UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

FELIPE MARTIN SAMPAIO

Energy-Efficient Memory Architecture Design and Management for Parallel Video Coding

Thesis presented as partial requirement for the PhD degree in Computer Science.

Advisor: Prof. Dr. Sergio Bampi
Co-advisor: Prof. Dr. Bruno Zatt

Porto Alegre
2018

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Sampaio, Felipe Martin

Energy-Efficient Memory Architecture Design and Management for Parallel Video Coding / Felipe Martin Sampaio. -- 2018

169 f.

Orientador: Sergio Bampi.

Co-orientador: Bruno Zatt.

Tese (Doutorado) – Universidade Federal do Rio Grande do Sul, Instituto de Informática, Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2018.

1. High-Efficiency Video Coding. 2. On-Chip Memory Design. 3. Memory Management. 4. Application-Specific Knowledge. 5. Parallel Processing. I. Bampi, Sergio, orient. II. Zatt, Bruno, coorient. III. Título.

ACKNOWLEDGEMENTS

Trabalhar na área que escolhi para meu futuro, ter a oportunidade de contribuir com a comunidade acadêmica por meio do meu esforço de pesquisa, participar de eventos científicos onde pude conhecer os mais recentes avanços tecnológicos, experimentar a prática docente e me encantar com a possibilidade de ser um agente do transformação por meio da educação. Estes são apenas alguns dos benefícios, além de outros tantos, que tive durante o período em que fui estudante de doutorado. Iniciei apenas como um estudante e terminei como professor e amante da pesquisa científica. No primeiro ano (2013), quando trabalhei como bolsista em tempo integral do PPGC, pude me focar e trabalhar para desenvolver toda a base fundamental do que seria a minha Tese de doutorado. Quando fui nomeado e decidi por dividir minha jornada de trabalho como professor do Instituto Federal do Rio Grande do Sul (IFRS), sabia que conciliar ambas as atividades seria desafiador e que me exigiria muita dedicação. Durante o segundo ano 2014), pude dar continuidade aos trabalhos, tendo a oportunidade apresentar um dos meus artigos na conferência ISLPED, que ocorreu na cidade de San Diego, no estado da Califórnia, Estados Unidos. Em 2015 desenvolvi outra parte importante do meu projeto de doutorado, tendo a possibilidade de apresentá-lo no evento CASES, em Amsterdã. Ainda em 2016, ingressei no curso de Formação de Professores para os Componentes Curriculares da Formação Profissional, o que me habilitou como Licenciado em Ciência da Computação e me trouxe toda a base que faltava dentro das minhas atividades docentes. Diante deste breve relato da minha trajetória, agradeço ao Programa de Pós-Graduação em Computação da UFRGS, além do Campus Farroupilha do IFRS pela oportunidade de dar continuidade à minha formação. Juntamente com o CEFET-RS (ensino médio e técnico) e a UFPel (graduação), estas instituições me mostraram que é possível ter uma formação de qualidade por meio da dedicação de todos os agentes envolvidos. Me sinto privilegiado de ser egresso e de poder divulgar tudo de sensacional que estas instituições oferecem a seus estudantes. Me comprometo a utilizar todos os ensinamentos adquiridos sempre em prol do desenvolvimento humano, social, científico e tecnológico.

No início dos meus estudos no doutorado, ela era minha namorada e morava a mais de 1200 quilômetro de distância. Passados cinco anos, além de namorados nos tornamos (nesta ordem): companheiros, pais de um gato, moradores de um mesmo lar, noivos, pais (ansiosos pelo nascimento da nossa “pulga”), casados e pais efetivamente (agora com o nascimento daquela que encanta nossas vida todos os dias). Minha vida mudou muito durante este período, sendo que em todas ela esteve comigo. Amor, carinho, compreensão, companheirismo, esforço e dedicação são algumas palavras que descrevem minha esposa, Meiri Brum Lima Sampaio, fazendo com que eu tenha cada vez mais admiração por tudo que ela representa na minha vida. Quando me mudei para Farroupilha, ela me deu a maior prova de amor e veio para perto de mim; quando eu precisei trabalhar por finais de semana seguidos, ela segurou as pontas e me deu total apoio; quando eu estava cansado e precisava desabafar, ela sempre esteve ali para mim. Poderia, sem esforço algum, dedicar o espaço inteiro desta Tese para para agradecer cada momento que vivemos juntos; e certamente não seria suficiente. Amo o que construímos, amo o que vivemos, amo o que surgiu da gente e amo planejar e construir uma vida ao teu lado. Obrigado por tudo!

Há um pouco mais que um ano e cinco meses, em um final de tarde em Caxias do Sul, caiu a ficha de que, tudo o que eu tinha sentido até agora, não se comparava ao que estava por vir. Deste então, vivo todos os dias na companhia da guriazinha mais cativante, cheia de energia e com o sorriso mais lindo do mundo inteiro. Encontrar ela e receber um sorriso, um abraço apertado e um grito de “papai” renova as energias e dá um sentido especial para a vida. Trabalhar se torna mais fácil pois sei que a noite eu viverei isso. Espero que eu consiga oferecer

todas as oportunidades que eu tive para a formação da minha filha, e que ela possa escolher com autonomia seu futuro, sempre com respeito às pessoas e às diferenças.

Olhando para trás e com uma visão do mundo um pouco melhor do que já tive, me sinto privilegiado nas oportunidades que tive na vida. Não são todos (poucos, na verdade) que tem a oportunidade de estudar sem se preocupar com mais nada. A possibilidade de acesso a escolas de qualidade, o incentivo para me qualificar pessoal e profissionalmente, a certeza de que, se algo desse errado, teria um apoio incondicional que me ajudaria a superar qualquer dificuldade. Tudo isto foi proporcionado por duas pessoas maravilhosas que sempre se dedicaram muito e que sempre me apoiaram em todas as minhas decisões. Atualmente passamos pouco tempo juntos, mas a minha admiração por tudo que eles representam aumenta cada vez mais. Além de tudo, nada melhor que o exemplo para que os filhos cresçam se espelhando em seus pais. Diante disso, queria agradecer aos meus pais, Iracy e Toribio, e dizer que são meus ídolos para a vida: quero ser para minha filha tudo o que eles sempre foram para mim. Amo vocês!

Durante os últimos anos pude também acompanhar a evolução pessoal e profissional daquela que acompanhei crescer desde bebê. Foi muito especial poder participar da tua trajetória, mesmo que nos últimos 7 anos a distância. Além de irmãos, somos dindo/afilhada e, mais recentemente, comadre/compadre. Teu carinho por mim, pela Meiri e pela Aninha sempre me mostrou a pessoa maravilhosa, correta e dedicada que és. Que tu sigas sempre nesse caminho, sabendo que a qualquer momento pode contar com meu apoio!

Seguindo neste ritmo, gostaria de estender os agradecimentos a todos meus familiares que sempre serviram de base e sempre estiveram juntos comigo. Tenho muita sorte de ter no meu nome as marcas das famílias “Martin” e “Sampaio”: certamente o que eu me tornei hoje é o reflexo de tudo que vivi ao lado de vocês. Além disso, também entrei de “gaiato” nas famílias “Lima” e “Brum”, que mesmo não estando registradas no meu nome, tem um espaço especial em meu coração. Obrigado a todos!

Agradeço também aos meus orientadores, os quais contribuíram de forma essencial durante todo o período do doutorado. Ao professor Sergio Bampi, pela sempre disponibilidade em me auxiliar no desenvolvimento dos trabalhos, nas submissões dos artigos e nos encaminhamentos junto ao PPGC. Ao professor e colega Bruno Zatt, que aceitou o convite para ser co-orientador do trabalho e que sempre esteve disposto a trabalhar e dar valiosas contribuições a tudo que desenvolvi (mesmo com prazos extremamente exíguos…). Ao professor Muhammad Shafique, pela colaboração que mantivemos durante todo o meu trabalho de doutorado. A qualidade do trabalho não teria sido a mesma sem esta cooperação que tivemos durante estes anos. Me sinto honrado de trabalhar com vocês!

A amizade e o companheirismo tiveram uma parte importante nesta trajetória recente. Agradeço aos meus amigos que conviveram comigo mais diariamente como colegas de apartamento (Daniel, Diego, Rafael, Mateus e Bruno - estes dois últimos por períodos curtos, mas muito significativos). Aos meus amigos de fé, que me acompanham há mais de uma década, muito obrigado pela parceria, pelo companheirismo e pela cumplicidade. Em especial, alguns tiveram papéis especiais como padrinhos e madrinhas de casamento (Lucas, Carol, Igor, Juliana, Cícero, Duda e Diego). Além disso, dois participaram quase que diariamente das nossas vidas acompanhando e sendo referência no crescimento da Aninha: sei que os “didis” Lucas e Carol serão exemplos importantes para nossa pequena. Estendo estes destaques a todos os que me acompanharam (de maneira mais ou menos próxima) durante a minha trajetória: em especial a todos que integram a grande família Fifonde!

Meus colegas de trabalho tiveram papel fundamental durante a trajetória, sejam pelas discussões técnicas sobre assuntos relacionados ao trabalho, ou mesmo pelos momentos de descontração que me proporcionaram. Principalmente durante o primeiro ano do doutorado, quando estive em tempo integral no laboratório 215 do Instituto de Informática da UFRGS, tive colegas que, além de parceiros nas atividades de pesquisa, foram amigos que compartilharam comigo toda a rotina de morar em Porto Alegre (quase todos, assim como eu, se encontravam longe das famílias). Dividimos os anseios, as perspectivas e os sonhos durante este período importante de nossos processos formativos. Nos anos seguintes me inseri em um contexto diferente: sendo professor do Campus Farroupilha do Instituto Federal do Rio Grande do Sul. Em uma perspectiva diferente, pude conhecer e trabalhar junto com grandes profissionais da educação, nos quais pude me inspirar para este novo papel que comecei a ter. Todos os momentos de discussão, conversa e de descontração foram importantes para que eu pudesse conciliar todas as minhas atividades. Muito obrigado a todos do Campus Farroupilha que tiveram participação durante este processo!

Aproveito também para estender meu agradecimento a um grupo importante de colegas que tive durante os últimos quatro anos: os alunos que passaram por mim nas diferentes atividades de ensino, pesquisa e extensão que desenvolvi. Incrível chegar em sala de aula com a ideia de que o aprendizado seria adquirido por eles e perceber que fui eu quem mais aprendi nesse processo. Cada um com quem pude conviver certamente deixou sua marca na minha trajetória, e agradeço a todos pela possibilidade de contribuir com sua formação.

ABSTRACT

This Thesis presents the design of an energy-efficient hybrid scratchpad video memory architecture (called Hy-SVM) for parallel High-Efficiency Video Coding. Video coding stands out as a high complex part in the video processing applications. HEVC standard brought innovations that increase the memory requirements, mainly due to: (a) the novel coding structures, which aggravates the computational complexity by providing a wider range of possibilities to be analyzed; and (b) the high-level parallelism features provided by the Tiles partitioning, which provides performance acceleration, but, at the same time, strongly adds hard challenges to the memory infrastructure. The main bottleneck in terms of external memory transmission and on-chip storage is the reference frames data: which consists of already coded (and reconstructed) entire frames that must be stored and intensively accessed during the encoding process of future frames. Due to the large volume of data required to represent the reference frames, they are typically stored in the external memory (especially when high-definition videos are targeted). The proposed Hy-SVM architecture is inserted in a video coding system, which is based on multiple Tiles partitioning to enable parallel HEVC encoding: each Tile is assigned to a specific processing unit. The key ideas of Hy-SVM include: application-specific design and management; combined multiple levels of private and shared memories that jointly exploit intra-Tile and inter-Tiles data reuse; scratchpad memories (SPMs) as energy-efficient on-chip data storage; combined SRAM and STT-RAM hybrid memory (HyM) design. We propose a design methodology for Hy-SVM that leverages application-specific properties to properly define the HyMs parameters. In order to provide run-time adaptation (for both off- and on-chip parts), Hy-SVM integrates a memory management layer composed of: (1) *overlap prediction*, which has the goal of identifying the redundant memory access behavior by analyzing monitored past frames encoding to increase inter-Tiles data reuse exploitation; (2) *memory pressure management*, which aims on balancing the Tiles-accumulated memory pressure targeting on improving external memory communication channel usage; and (3) *lifetime-aware data management scheme* that alleviates STT-RAM SPMs of high bit-toggling write accesses to increase the their cells lifetime, as well as to reduce overhead issues related to poor write characteristics of STT-RAM. Application-specific knowledge was exploited by inheriting HEVC properties and performing run-time monitoring of memory accesses. Such information is used to properly design the on-chip video memories, as well as being utilized as input parameters of the run-time memory management layer. Based on the run-time decisions from the proposed Hy-SVM management strategies, Hy-SVM integrates distributed *memory access management units* (MAMUs) to control the access dynamics of private and shared SPMs. Additionally, *adaptive power management units* (APMUs) are able to strongly reduce on-chip energy consumption due to an accurate overlap prediction.

The experimental results demonstrate Hy-SVM overall energy savings over related works under various HEVC encoding scenarios. Compared to traditional data reuse schemes, like Level-C, the combined intra-Tile and inter-Tiles data reuse provides 69%-79% of energy reduction. Regarding related HEVC video memory architectures, the savings varied from 2.8% (worst case) to 67% (best case). From the external memory perspective, Hy-SVM can improve data reuse (by also exploiting inter-Tiles data redundancy), resulting on 11%-71%% of reduced off-chip energy consumption. Additionally, our APMUs contribute by reducing on-chip energy consumption of Hy-SVM by 56%-95%, for the evaluated HEVC scenarios. Thus, compared to related works, Hy-SVM presents the lowest on-chip energy consumption. The memory pressure management scheme can reduce the variations in the memory bandwidth by 37%-83% when compared to the traditional raster scan processing for 4- and 16-core parallelized HEVC encoder. The lifetime-aware data management significantly extends the STT-RAM lifetime, achieving 0.83 of normalized lifetime (near to the optimal case). Moreover, the overhead of implementing our management units insignificantly affects the performance and energy-efficiency of Hy-SVM.

Keywords: High-Efficiency Video Coding. Parallel Processing. On-Chip Memory Design. Memory Management. Application-Specific Knowledge. Emerging Memory Technologies and Organizations.

Projeto e Gerenciamento de Arquitetura de Memória Energeticamente Eficiente para Codificadores de Vídeo HEVC

RESUMO

Esta tese de doutorado apresenta o projeto de uma arquitetura de memória híbrida energeticamente eficiente baseada em memórias do tipo *scratchpad* (Hy-SVM) para a codificação paralela de vídeos segundo o padrão HEVC. A codificação de vídeo se destaca como uma parte extremamente complexa nas aplicações de processamento de vídeo. O padrão HEVC traz inovações que complicam fortemente os requerimentos de memória de tais aplicações, principalmente devido a: (a) novas estruturas de codificação, as quais agravam a complexidade computacional por proporcionarem muitas modos possíveis de codificação que devem ser analisados; além do (b) suporte de alto nível à paralelização da codificação por meio do particionamento das unidades de codificação em múltiplos *Tiles*, o qual provê a aceleração da performance dos codificadores, porém, ao mesmo tempo, adiciona grandes desafios à infraestrutura de memória. O principal gargalo em termos de comunicação com a memória externa e de armazenamento interno (dentro do chip do codificador) é dados pelas informações dos quadros de referência: que consiste em uma série de quadros completos já codificados (e reconstruídos) que devem ser mantidos em memória e acessados de forma intensa durante o processamento dos quadros futuros. Devido ao grande volume de dados que são necessários para representar os quadros de referência, estes são tipicamente armazenados na memória externa dos codificadores (principalmente quando vídeos de alta e ultra alta resolução são processados). A arquitetura proposta Hy-SVM está inserida em um sistema de codificação baseado no particionamento dos quadros do vídeo de entrada em múltiplos *Tiles*, de forma a habilitar a codificação paralela das informações segundo o padrão HEVC: neste cenário, cada *Tile* é assinalado para uma específica unidade de processamento do codificador HEVC, o qual executa o processamento dos diferentes *Tiles* em paralelo. A ideias chave da arquitetura Hy-SVM incluem: projeto e gerenciamento de memórias para a aplicação específica de codificação de vídeo; uso de múltiplos níveis de memórias privadas e compartilhadas, com o objetivo de explorar o reuso de dados intra-*Tile* e inter-*Tiles* de forma combinada; uso de memórias do tipo *scratchpad* (SPMs) para o armazenamento interno da informações de forma eficiente em termos de consumo de energia; projeto de memórias híbridas utilizando as tecnologias SRAM e STT-RAM como base. Uma metodologia de projeto é proposta para a arquitetura Hy-SVM, a qual aproveita propriedades específicas da aplicação para, de forma adequada, definir os parâmetros de projeto das memórias híbridas. De forma a prover adaptação em tempo de execução (para ambas as memórias *on-chip* e *off-chip*), a arquitetura Hy-SVM integra uma camada de gerenciamento composta pelas seguintes estratégias: (1) predição do *overlap* (sobreposição de acessos), o qual busca identificar o comportamento dos acessos redundantes entre diferentes unidades de processamento do codificador HEVC a partir da análise dos acessos à memória das codificações dos quadros passados do vídeo, com o objetivo de aumentar o potencial de exploração do reuso de dados inter-*Tiles*; (2) gerenciamento dos acessos à memória externa, responsável por balancear a vazão de dados com a memória acumulada entre as múltiplas unidades de processamento do codificador HEVC paralelo, com o objetivo de melhorar o uso do barramento de comunicação com a memória externa; e (3) gerenciamento de dados das SPMs implementadas a partir de células de memória STT-RAM, o qual alivia estas células de acessos de escrita com alta atividade de chaveamento dos bits armazenados, com o objetivo de aumentar o tempo de vide destas células, bem como reduzir as penalidades relativas à ineficiência dos acessos de escrita nas memórias STT-RAM. O conhecimento específico da aplicação foi utilizado nas estratégias de gerenciamento em tempo de execução das seguintes formas: explorando parâmetros da codificação HEVC e realizando monitorando em tempo real dos acessos à memória realizados pelo codificador. Estas informações são utilizadas tanto pelas técnicas de gerenciamento, quanto pelas metodologias de projeto das memórias. Baseadas nas decisões tomadas pela camada de gerenciamento, a arquitetura Hy-SVM integra unidades de gerenciamento de acessos à memória (*memory access management units* – MAMUs) para controlar as dinâmicas de acesso das memórias SPM privadas e compartilhadas. Além disso, unidades adaptativas de gerenciamento de potência (*adaptive power management units* – APMUs) são capazes de reduzir o consumo de energia interno do chip do codificador a partir das estimativas precisas de formação dos *overlaps*.

Os resultados obtidos por meio dos experimentos realizados demonstram economias de consumo energético da arquitetura Hy-SVM, quando comparada a trabalhos relacionados, sob diversos cenários de teste. Quando comparada a estratégias de reuso de dados tradicionais para codificadores de vídeo, como o esquema Level-C, a exploração do reuso de dados combinado nos níveis intra-*Tile* e inter-*Tiles* provê 69%-79% de redução de energia. Considerando as arquiteturas de memória de vídeo com foco no padrão HEVC, os ganhos variaram desde 2,8% (pior caso) até 67% (melhor caso). Da perspectiva do consumo de energia relacionado à comunicação com a memória externa, a arquitetura Hy-SVM é capaz de melhorar o reuso de dados (por explorar também o reuso de dados inter-*Tiles*), resultando em um consumo de energia *on-chip* 11%-17% menor. Além disso, as APMUs contribuem para reduzir o consumo de energia *on-chip* da arquitetura Hy-SVM em 56%-95%, para os cenários de teste analisados. Desta forma, comparada aos trabalhos relacionados, a arquitetura Hy-SVM apresenta o menor consumo energético *on-chip*. O gerenciamento da vazão da comunicação com a memória externa é capaz de reduzir as variações de largura de banda em 37%-83%, quando comparado à ordem tradicional de processamento, para cenários de teste com 4 e 16 *Tiles* sendo processados em paralelo pelo codificador HEVC. O gerenciamento de dados pôde, de forma significativa, estender o tempo de vida das células de memória STT-RAM, alcançando 0,83 de tempo de vida normalizado (métrica adotada para comparação, ficando muito próximo do caso ideal). Além disso, as sobrecargas causadas pela implementação das unidades de gerenciamento não afetam de foram significativa a performance e a eficiência energética da arquitetura Hy-SVM propostas por este trabalho.

Keywords: High-Efficiency Video Coding. Processamento Paralelo. Projeto de Memória On-Chip. Gerenciamento de Memória. Conhecimento Específico da Aplicação. Tecnologias e Organizações de Memórias Emergentes.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

2K	2560x1600 pixels resolution
ACT	Activation
AGU	Address Generation Unit
APMU	Adaptive Power Management Unit
AOM	Alliance for Open Media
AVC	Advanced Video Coding
B	Block
BL	Bitline
BLB	Bitline-bar
BT	Bit-toggling activity
BT_KEY	Bit-toggling activity key
BU	Basic Unit
CABAC	Context-Adaptive Binary Arithmetic Coding
CMOS	Complementary Metal-Oxide-Semiconductor
C_{Tile}	Tile classification (H-, M- or L-Type)
CTU	Coding Tree Unit
CU	Coding Unit
CurrOv	Current overlap
Dim	Dimension
D_{ME}	Motion Estimation distance factor
DM	Data management
DMT	Data Management Table
DPB	Decoded Picture Buffer
DRAM	Dynamic Random Access Memory
dSVM	Distributed Scratchpad Video Memory
EBT	Estimated bit-toggling activity
enHyV	Energy-Efficient Hybrid Video Memory
FPS	Frames per second
FQ	Forward quantization
FT	Forward transform
GOP	Group of Pictures
H	Height
HD	High Definition
HD1080	1920x1080 pixels resolution
HEVC	High-Efficiency Video Coding
HM	HEVC Test Model
HyM	Hybrid Memory
Hy-SVM	Hybrid Scratchpad Video Memory
IQ	Inverse quantization
IT	Inverse transform

LD	Low Delay
JVET	Joint Video Exploration Team
JVT-VC	Joint Video Team - Video Coding
LPDDR	Low-Power Dual-Data Rate
LS	Line size
LSB	Least-significant bit
MAMU	Memory Access Management Unit
MC	Motion Compensation
ME	Motion Estimation
MLC	Multi-level cell
MOT	Monitored Overlap Table
MPEG	Moving Picture Experts Group
MRAM	Magnetic Random Access Memory
MSB	Most-significant bit
MSD	Mean Squared Deviance
MSE	Mean Squared Error
MTJ	Magnetic Tunneling Junction
NB	Number of banks
NG	Number of CTU groups
NL	Number of lines
NMOS	N-Type Metal-Oxide-Semiconductor
NS	Number of sectors
NVM	Non-Volatile Memory
NVSim	Non-Volatile Simulator
PCM	Phase-Change Memory
PDF	Probability Density Function
PMOS	P-Type Metal-Oxide-Semiconductor
POT	Predicted Overlap Table
PredOv	Predicted overlap
PrivL1SPM	Private L1 Level Scratchpad Memory
PrivL2SPM	Private L2 Level Scratchpad Memory
PSNR	Peak Signal-to-Noise Ratio
PU	Prediction Unit
RA	Random Access
RAP	Random Access Point
RCDR	Reference-Centered Data Reuse
RD	Read
R-D	Rate-Distortion
RDO	Rate-Distortion Optimization
REF	Refresh
RefOv	Reference overlap

SA	Sense amplifier
SAD	Sum of Absolute Differences
SD	Standard Definition
SharedL2 SPM	Shared L2 Level Scratchpad Memory
SL	Sourceline
SLC	Single-Level Cell
SPM	Scratchpad Memory
SRAM	Static Random Access Memory
SS	Sector size
STBY	Standby
STT-RAM	Spin-Transfer Torque Random Access Memory
SW	Search Window
TB	Tiles boundary
TH	Threshold
TU	Transform Unit
TZ	Test Zone
UDH	Ultra-High Definition
W	Width
WA	Write amplifier
WE	Wakeup Energy
WL	Wordline
WP	Weighted prediction
WPP	Wavefront Parallel Processing
WR	Write
YCbCr	Luminance, Blue and Red Chrominance

SUMMARY

1. INTRODUCTION

Video processing emerges as the most used multimedia application in the embedded systems field, reaching billions of users mainly due to the popularization of mobile devices. The availability of very high definition video cameras significantly increases the multimedia processing requirements in such devices. Furthermore, the recent advances of streaming services motivate users to constantly share and view digital videos over the internet.

Experts from Cisco released on July 2017 a forecast about the impact of visual networking applications in the data traffic over the internet. The analyses comprehend actual traffic analysis of 2016 and trace prediction trend lines until 2021. The complete technical reports are available at (CISCO, 2017). A summary of the main forecasts regarding multimedia applications, focusing on video processing, is depicted in Figure 1.1.

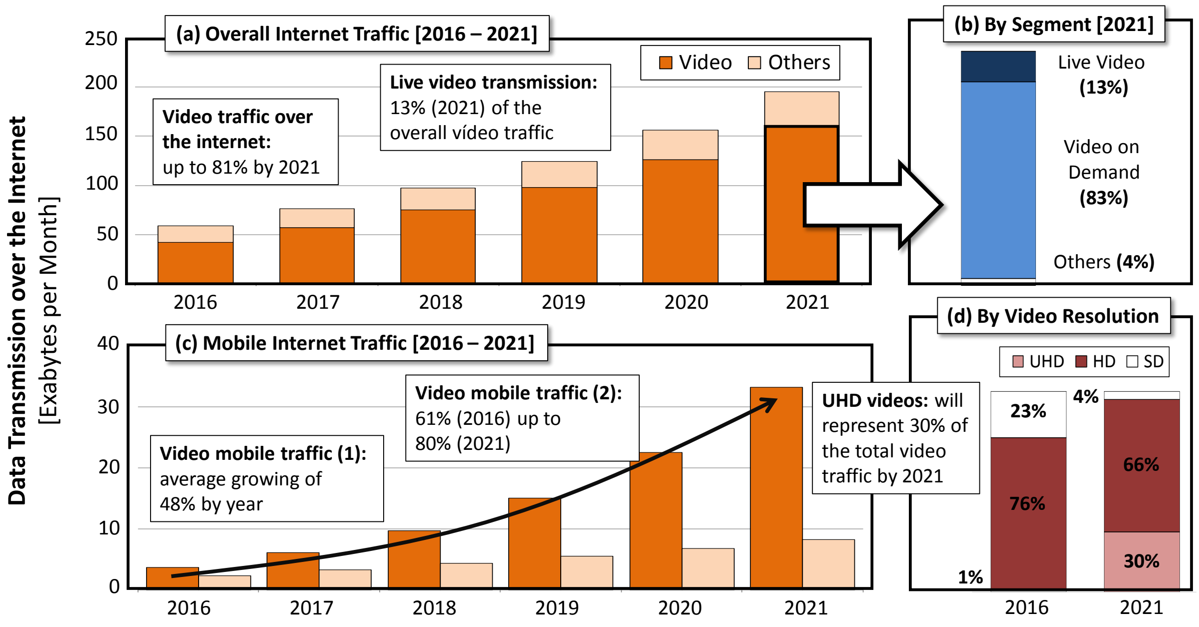


Figure 1.1: Internet traffic predictions from Cisco experts: (a) overall internet traffic trend (2016 to 2021); (b) video traffic by segment at 2021; (c) internet traffic trend considering only mobile devices (2016 to 2021); and (d) video traffic by resolution (Standard Definition (SD), High Definition (HD) and Ultra-High Definition (UHD)).

When considering the overall scenario (Figure 1.1a), the predictions say that 81% of all internet traffic will be occupied by video transmission in 2021. The considered segments of video applications were: video on demand (like Youtube, Netflix and another related video streaming services), live video transmission and others (like surveillance applications). Observing the predicted video traffic by segment at 2021, live video transmission will represent 13% (see Figure 1.1b). Live video applications lead to hard constraints for the processing, storage and transmission parts, since they impose real time requirements, aggravating the challenges for software and hardware designers to support it.

Figure 1.1c depicts the internet data traffic prediction considering only the mobile devices. Still, video processing stands out as the most representative application field: from 61% in 2016 up to 80% in 2021. In this analysis, it can be noted a stronger growth in the video traffic part, reaching 48% of increasing by year (on average). This trend endorses the need of efficient embedded multimedia processing (in terms of performance and energy) to support the growing demand for video applications on battery-powered devices.

Furthermore, the forecasts also trace the traffic considering the transmitted video resolutions (Figure 1.1d). In this scenario, it can be noted a strong increase in the Ultra-High Definition (UHD) video support. UHD videos, which represented only 1% of the total video transmission over the internet in 2016, will reach 30% of the total video traffic by 2021. Roughly, UHD videos have 2.5x and 9x more data than High Definition (HD) videos and Standard Definition (SD) videos. This leads to a significant complexity increasing for all steps inside the video processing system (storage, transmission, encoding and decoding).

1.1 Problems Definition

Advanced video processing algorithms introduce very high pressure on memory hierarchy, leading to undesirable energy and performance overhead (SHAFIQUE et al., 2012; ZATT et al., 2011a). Video codecs (encoders and decoders) are among the most complex and widely deployed video processing applications. Recently, the High-Efficient Video Coding (HEVC) (ISO/IEC-JCT1/SC29/WG11, 2013a) standard has been released to provide double coding efficiency compared to the H.264/AVC (its predecessor). However, this comes at the cost of increased computation time and by more than 40% (see Figure 1.2a). Besides employing novel complex coding tools, an HEVC encoder requires a significant amount of data from the off-/on-chip memories due to more memory intensive reference frames transmission for the prediction steps. On average, the memory demand is 2x-3x higher compared to the H.264/AVC (see Figure 1.2b). Thus, high off-/on-chip memory bandwidth along with larger on-chip video memories (to support bigger resolutions) leads to increased energy consumption in the HEVC encoders.

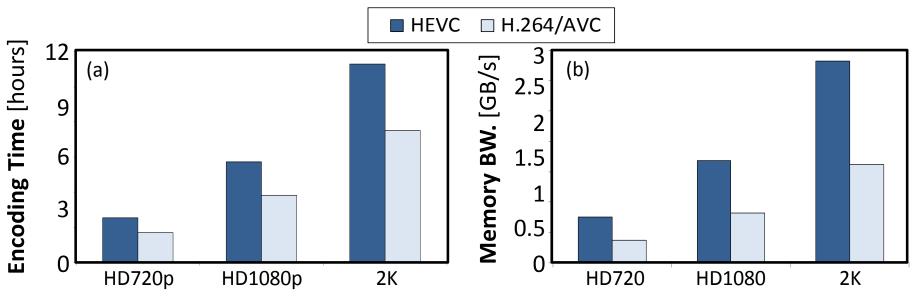


Figure 1.2: HEVC vs. H.264/AVC encoder (a) encoding time and (b) memory accesses comparison[1].

Furthermore, HEVC incorporate techniques to allow parallel execution, increasing the performance the encoding part, such as *Tiles* partitioning (MISRA et al., 2013). This strategy divides the frame into rectangular regions that can be encoded in parallel. Whereas providing speedup to encoding applications, such tools aggravate the energy consumption of the memory infrastructure (on- and off-chip parts; see Figure 1.3a and Figure 1.3b), posing new challenges for multimedia systems. The main Tiles-parallelized HEVC challenge is to efficiently exploit the inter-Tiles data reuse potential, which significantly increases as more parallelism is exploited (Figure 1.3d). In this work, we refer to this reference frame region that is redundantly accessed by more than one Tile processing as the *overlap* region (this concept will be extensively presented and discussed along the text). It can be also noted that, by increasing the parallelism employed during a HEVC encoding process, the external memory pressure[2] grows and becomes more unpredictable (see Figure 1.3c), imposing hard challenges to the access management.

[1] Average results for commonly used test sequences [13], 128x128 search window, H.264/AVC and HEVC test models, 300 frames.

[2] Also known as instant off-chip memory bandwidth.

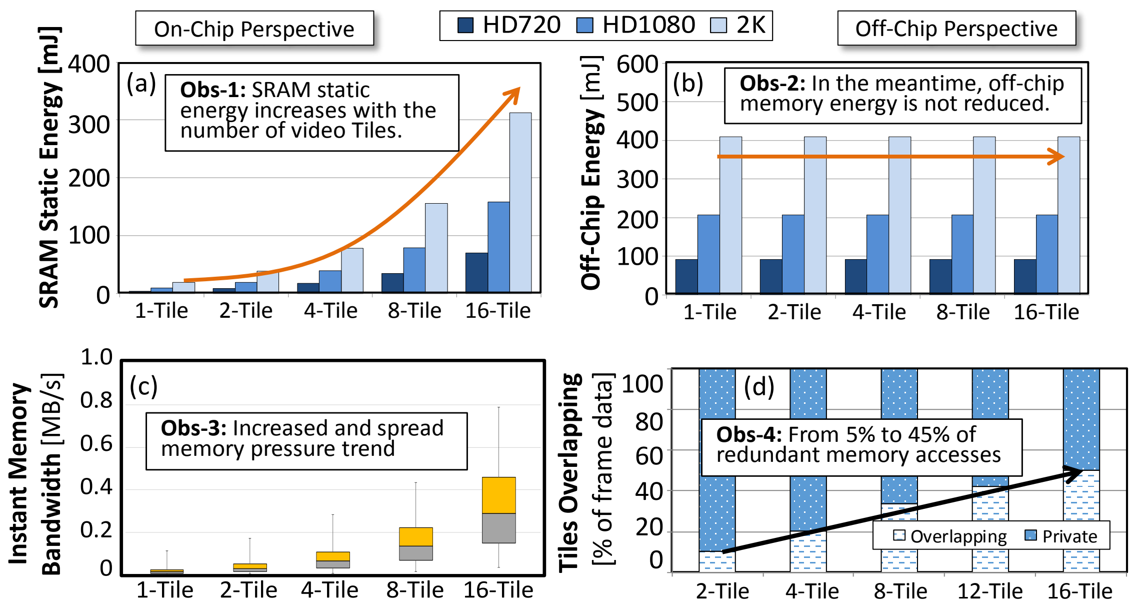


Figure 1.3: Memory requirements analysis for HEVC encoding.

Thereby, based on these preliminary evaluations, *there is a strong need for energy-efficient memory architectures targeting the viability of parallel features of HEVC*. Chapter 4 will present a detailed memory profiling of parallel executions of HEVC encoders to analyze specific behaviors and to specify the main energy bottlenecks.

1.2 Key Research Opportunities and Challenges

There are some important state-of-the-art research opportunities in the way of proposing memory architectures to enable energy-efficient HEVC execution on embedded manycore systems. Each one of these research fields brings hard challenges that will guide the proposed solutions of this work. A brief summary of these opportunities and the corresponding key challenges are presented as follows. A detailed review and further discussions of state-of-the-art works for each research area will be given at Chapter 3 .

Application-Specific Knowledge: Application-specific properties exploitation has been adopted by video coding community to base contributions in different research fields, such as computational complexity control units (CORREA et al., 2011, 2013), fast mode decision engines (LIAO; YANG; CHEN, 2016; PODDER; PAUL; MURSHED, 2016) and hardware accelerators design (BONATTO et al., 2017; CHO et al., 2015; M; SK, 2017). Recent trends demonstrated benefits of application-specific knowledge for memory design and management targeting low power video encoding (SAMPAIO et al., 2013a; SHAFIQUE et al., 2012; ZATT et al., 2011a, 2011b). In such works, specific video coding properties and video content characteristics are used as a basis for the proposed memory architectures. Therefore, *one important challenge is to leverage HEVC-specific knowledge to design and manage energy-efficient video memories for HEVC encoding*. Still, these works also takes advantage of the strong correlation between the memory access behaviors during the video encoding. Therefore, *another important challenge is to implement run-time monitoring systems to catch memory-related properties during the HEVC encoding process to provide helpful information for the management units to improve the memory energy efficiency*.

Memory Requirements for Parallel Video Coding: Even considering HEVC application-specific knowledge for memory architectures design to increase their energy efficiency, most of the published works does not consider the memory requirements for parallelized video coding. Important issues related to memory infrastructure in parallelized systems are related to: (1) restrictive access rates and increased memory pressure, due to multiple processing units requesting data in parallel; (2) conflicts related to the access of the same data from different processing units, leading to memory contention; and (3) inter-cores data accesses redundancy, requiring shared on-chip memories to minimize external memory communication. Therefore, *the challenge is to develop application-specific video memory architectures targeting energy efficiency considering parallel video coding requirements*.

Scratchpad Memories: A large body of research explored efficient cache organizations targeting multi/manycore processors. To overcome/alleviate the hardware overhead of caches, Scratch-Pad Memories (SPMs) evolved for energy-constrained embedded systems (BANAKAR et al., 2002). Instead of providing hardware support for map data/code from off-chip to on-chip memory, SPM allows designed/compiler to perform content management saving extra energy compared to complete caches under certain operating scenarios. Therefore, *the challenge here is to efficiently utilize SPMs by exploiting application-specific knowledge to enable energy savings in the SPMs design and access management*.

Hybrid Memory Design: Recently, the hybrid memory architectures for general purpose manycore processors have been emerged. The hybrid memory design utilizes emerging memory technologies (e.g., MRAM, STT-RAM (DONG et al., 2008)) in combination with traditional SRAM cells (ABE et al., 2012; KHAN; SHAFIQUE; HENKEL, 2013). Their goal is to reduce the impact of SRAM shortcomings, like low density and high static energy consumption. Generally, for general purpose applications, the emerging technologies are desired for last-level caches due to the low-static-energy and high-density features (CHEN et al., 2012). However, in reason of the lack of application-specific knowledge, these schemes are typically not efficient enough to support the high memory requirements of HEVC. Therefore, *there is a need of application-driven design for energy-efficient hybrid memories tailored towards HEVC executing in manycore processors*. This work exploits the STT-RAM (Spin-Transfer Torque RAM) usage in combination with SRAM, which is known to be a non-volatile memory (NVM), keeping data stored even if the cells are switched-off (no static energy consumption). Although all mentioned advantages of STT-RAM, its cell imposes hard energy and performance penalties during the write operations. In addition, STT-RAM cells have its lifetime dependent on the bit-toggling activity of the write operations (WU et al., 2010). Therefore, *there is a challenge here of properly data management policies (focusing on reducing the write activity) to increase the cells lifetime and to enable energy efficiency in a STT-RAM based hybrid memory system*.

1.3 Goals of this Thesis

The **major goal** of this thesis is to improve the energy efficiency for the memory infrastructure (off- and on-chip video memories) to enable Tiles-parallelized HEVC execution in embedded video processing applications. The proposed solutions are based on opportunities mostly brought by novel memory technologies and organizations, as discussed in the previous section.

Specific goals of this work are listed as follows. They will guide the insights and ideas for the designed memory architecture:

G1. Take advantage of application-specific knowledge of HEVC standard (i.e. its new coding tools) and video content properties to develop an energy-efficient video memory architecture;

G2. Consider specialized and more restricted memory requirements for energy-efficient HEVC parallel execution (multiple-Tiles feature of HEVC);

G3. Analyze, at design time, properties and offline memory access statistics of HEVC parallel executions to define design methodologies to improve the energy efficiency of on-chip video memories;

G4. Analyze, at run time, on-/off-chip memory-related access behaviors related to HEVC encoding process to adapt the memory management to the current requirements, in order to achieve higher energy savings;

G5. Utilize SPMs as energy-efficient on-chip video memories by exploiting the application-specific knowledge to simplify their management units circuitry;

G6. Design multi-level on-chip video memory architectures to both exploit intra-Tile and inter-Tile data reuse to allow further external memory energy savings;

G7. Exploit hybrid memory design, utilizing emerging STT-RAM technology, to minimize the SRAM energy-related shortcomings when large on-chip video memories are required for parallel HEVC execution, while handling endurance and write-inefficiency of STT-RAM cells.

G8. Balance Tiles-accumulate memory pressure to maximize the usage of the off-chip memory channel for reference data fetching;

1.4 Main Contributions

Figure 1.4 presents an overview of the contributions of this work, inside the adopted parallel HEVC execution system. To exploit multiple processing units system, uniform Tiles partitioning is defined for the input video. As these Tiles can be encoded independent to each other, they can be assigned to different processing units (as in Figure 1.4a). In the external memory (implemented using DRAM technology) is stored all data required for HEVC execution. Especially, the memory architectures focus on provide energy-efficient management for the reference frames (Figure 1.4b). Typically, for each target reference frame, each processing core will access data around its corresponding Tile position. In order to provide energy-efficient storage for the reference frames, this work implements a Hybrid Scratchpad Video Memory Architecture (Hy-SVM). The proposed memory architecture is designed based on methodologies that leverage application-specific knowledge (Figure 1.4d) and offline memory-related statistics from HEVC. Techniques that exploit monitored run-time memory-related statistics to increase the energy efficiency of Hy-SMVM are implemented by three different memory management schemes (Figure 1.4e): (1) Overlap Prediction, (2) Memory Pressure Management and (3) Lifetime-Aware Data Management. Based on run-time decisions of such schemes, Hy-SVM integrates distributed Memory Access Management Units (MAMUs) and Adaptive Power Management Units (APMUs), which effectively manages the access dynamics and the energy consumption, being adaptive to the input video content properties.

The overall ideas related to each one of the main contributions of this work are presented as follows:

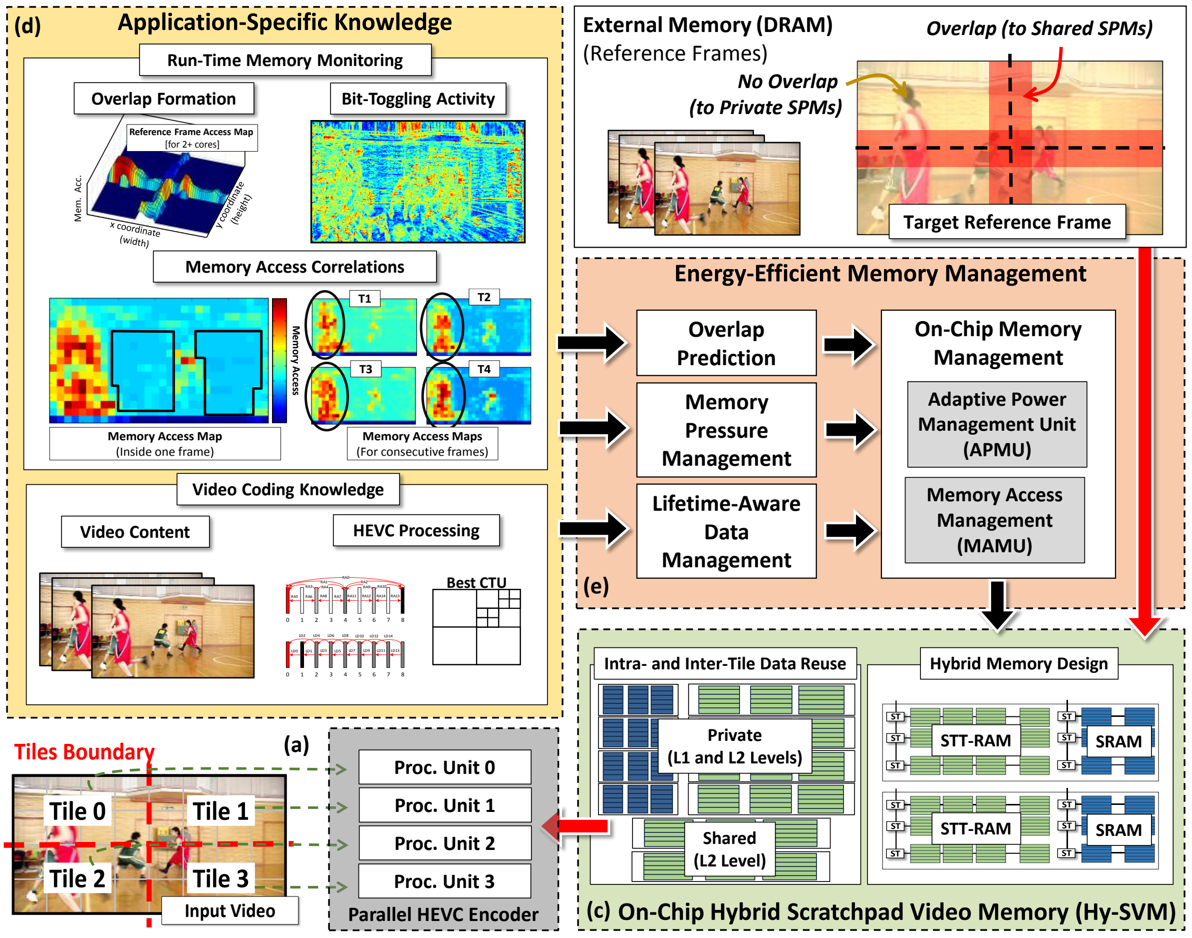

(d) Application-Specific Knowledge
Run-Time Memory Monitoring
Overlap Formation
Bit-Toggling Activity
Reference Frame Access Map [for 2+ cores]
Memory Access Correlations
Memory Access Map (Inside one frame)
Memory Access Maps (For consecutive frames)
T1
T2
T3
T4
Video Coding Knowledge
Video Content
HEVC Processing
Best CTU
External Memory (DRAM) (Reference Frames)
No Overlap (to Private SPMs)
Overlap (to Shared SPMs)
Target Reference Frame
Energy-Efficient Memory Management
Overlap Prediction
Memory Pressure Management
Lifetime-Aware Data Management
On-Chip Memory Management
Adaptive Power Management Unit (APMU)
Memory Access Management (MAMU)
(e)
Intra- and Inter-Tile Data Reuse
Private (L1 and L2 Levels)
Shared (L2 Level)
Hybrid Memory Design
STT-RAM
SRAM
(c) On-Chip Hybrid Scratchpad Video Memory (Hy-SVM)
Tiles Boundary
(a)
Tile 0
Tile 1
Tile 2
Tile 3
Input Video
Proc. Unit 0
Proc. Unit 1
Proc. Unit 2
Proc. Unit 3
Parallel HEVC Encoder

Figure 1.4: Overview of the novel contributions of this work.

1.4.1. *Energy-Efficient On-Chip Memory Design*

- *Hybrid Scratchpad Video Memory Architecture (Hy-SVM)*: that is composed of multiple levels of private and shared SPMs. It consists on (i) private L1[3] SPMs, implemented as SRAM arrays, to store the search window samples required for each HEVC processing unit; and (ii) private and shared L2 SRAM/STT-RAM hybrid memory SPMs (called HyMs) to provide reference frame level data reuse. The proposed design methodology leverages application-specific knowledge to define the hardware design parameters of SPMs.

1.4.2. *Energy-Efficient Memory Management Layer*

- *Overlap Prediction*: that leverages application-specific properties (e.g., history of past overlaps, video content and HEVC knowledge; see Figure 1.4d) to estimate the overlap characteristics for the next frame encoding. The overlap consists on the reference frame region that is accessed by two or more processing cores (each core processes one specific Tile). Figure 1.4d depicts the typical pattern of the formed overlap. Important overlap properties are related to its size, shape and access intensity distribution.

- *Lifetime-Aware Data Management*: that leverages application-specific properties to improve the STT-RAM cells endurance in the HyMs. The proposed data

[3] L1 and L2 in this work are not related to cache levels, but scratchpad memories implemented using SRAM-only (L1 level) or hybrid SRAM/STT-RAM (L2 level).

management scheme dynamically decides if the incoming reference frame block will be stored in the SRAM or STT-RAM portion. This decision is based on the estimated bit-toggling activity of each memory write operation in the HyMs, since most of STT-RAM shortcomings are related to write accesses.

- *Memory Pressure Management*: that leverages the memory access correlation within and across different Tiles (i.e. intra- and inter- Tiles correlations; see Figure 1.4d) to balance the instant memory pressure that is necessary for multiple cores to simultaneously access the external memory channel.
- *On-Chip Management Units of SPMs:* that implements memory access management units (MAMUs) and adaptive power management units (APMUs) to manage the data migration and the energy consumption of Hy-SVM (as in Figure 1.4e). Based on the overlap prediction output, the MAMUs implements read and write policies that manage the incoming Hy-SVM access to the corresponding SPM/HyM. Furthermore, APMUs can adapt the power gating strength according to the predicted overlap characteristics, which strongly depend on the video content.

1.5 Text Organization

This PhD Thesis is organized as follows:

Chapter 2 brings the basics related to video coding applications and further details of the state-of-the-art High-Efficiency Video Coding (HEVC), which are crucial to understand the ideas of the proposed video memory architectures. Additionally, this chapter presents the concepts related to the energy consumption of memory technologies for on-/off-chip memories. Initially, traditional Static-RAM (SRAM) is addressed. Following, the adopted emerging memory technologies for the proposed on-chip hybrid memory architectures are explained: Static Random Access Memory (SRAM) and Spin-Transfer Torque RAM (STT-RAM).

Chapter 3 presents the main ideas of state-of-the-art works that focus on improving energy efficiency of memory systems. Initially, a brief overview of general-purpose contributions regarding memory architectures for parallel processing is given. Then, application-specific works targeting video coding memory optimization are presented and discussed. After, detailed comparisons are performed with related energy-efficient video memory architectures.

Chapter 4 shows detailed HEVC memory evaluations to motivate the proposed on-chip memory architecture and energy-efficient management units. The main goal is to have a better understanding of the access behaviors of parallel HEVC to support the ideas. A key concept behind this work is duly defined: the overlap formation. Finally, the novel contributions are retaken by introducing the following technical chapters.

Chapter 5 introduces our Hybrid Scratchpad Video Memory Architecture (Hy-SVM), which is strongly based on the overlap exploitation to save external memory energy. Initially, an overview of the designed SPM levels and the adopted on- and off-chip memory organization models are presented. Then, offline statistical evaluations of HEVC and video content properties are performed, which will base the design of SPMs. Thus, the proposed methodology to define the memory parameters of all on-chip memory levels is described. Moreover, a logic organization of proposed hybrid SPMs is presented, relying on a joint combination of SRAM and STT-RAM portions to increase cells lifetime.

Chapter 6 properly describes the energy-efficient management layer of Hy-SVM. At first, the overlap prediction scheme is presented. The proposed strategy is based on performed evaluations from correlated characteristics from consecutive overlaps. After, the memory pressure management strategy is presented to properly manage the off-chip communication.

To manage STT-RAM write and to handle with endurance issues, the lifetime-aware data management scheme is explained. The effective hardware implementations of the proposed management schemes are integrated in the on-chip management units of Hy-SVM, composed be distributed memory access management units (MAMUs) and adaptive power management units (APMUs), which are described in the last part of the chapter.

Chapter 7 presents discussions regarding the experimental results, mostly presented in terms of energy consumption considering the on- and off-chip parts. The adopted methodologies for video coding evaluation, the used memory power models and the simulation infrastructure for parallel HEVC execution are also presented. The results of Hy-SVM are extensively compared to state-of-the-art related works and baseline implementations. Besides energy evaluations, external memory communication, overlap prediction accuracy and overhead analyses are also presented.

Chapter 8 concludes this thesis by presenting the final remarks. All contributions are summarized and the initially defined goals are discussed, based on the achieved results. As reflexive analysis from the PhD path, future research perspectives are presented.

2. BACKGROUND

Uncompressed video signals lead to a huge amount of data. Still, digital video usage has become even more ubiquous (SZE; BUDAGAVI; SULLIVAN, 2014). As results from this, video traffic is the biggest load on communication networks and data storage in a world-wide scenario. In this field, video compression has a key role of alleviating these constraints.

The H.264/AVC standard (ITU-T, 2013) still has important contribution in the video processing applications. However, considering the requirements imposed in its definition, H.264/AVC encoders are not enough scalable to meet the nowadays constant hunger for higher video quality (e.g. in terms of ultra-high resolutions, higher frame rates, and higher fidelity). In this scenario, the High-Efficiency Video Coding (HEVC) emerged (ISO/IEC-JCT1/SC29/WG11, 2013a), offering a major step forward to support these requirements.

At the first part, this chapter has the goal of introducing the main basic fundamental concepts involved in the video compression (aka. coding) applications **(Section 2.1)**. From this, the HEVC standard is discussed in deeper details **(Section 2.2)**. The emphases are for the HEVC structures and coding tools that are mostly related to the proposed memory architectures. For details regarding the other parts of HEVC processing, some references will be suggested.

In the second part, it will be given an introduction regarding the memory technologies and organizations adopted by the memory architectures designed by this work **(Section 2.3)**. The main goal is to present the memory cells internal design (at transistor level), as well as memory organization structures. Finally, energy consumption characteristics of all covered memory technologies will be discussed and compared.

2.1 Preliminaries on Video Coding

2.1.1. *Digital Video Characteristics*

A digital video is a sequence of static images (called *frames or pictures*) that, when exhibited at an enough temporal rate, gives a motion sensation to the viewer. In general, the enough frame rate to ensure a smooth motion perception considering the human visual system is around 30 frames per second (fps) (RICHARDSON, 2004). For nowadays video applications demand, which provides improved realism experience to users, the required frame rate for digital videos reaches 120 frames per second (SZE; BUDAGAVI; SULLIVAN, 2014). Each frame within the video is digitally represented by a two-dimensional matrix of *pixels*, with horizontal dimension of *W* (width) and vertical dimension of *H* (height).

The pixel is the digital data that stores the color and luminosity information of its corresponding position inside a frame. There are several color spaces that define the numerical representation of the pixels properties. Video coding applications are typically based on the YCbCr color space (see Figure 2.1). In such space, there are three different information: luminance (Y), blue chrominance (Cb) and red chrominance (Cr). The luminance channel (also referred as luma) represents the luminosity (light intense – gray scale fashion) of the picture. The chrominance matrices (or chroma) are related to the different color tones of the frame. The YCbCr color space is the preferred one for video coding applications in reason of its weak correlation between the channels (RICHARDSON, 2004). As consequence, compression tools can be applied separately for each component, enabling the exploitation of specific properties of each one. Video coding community commonly refers to each luma or chroma component of a pixel as *sample*. By following this, it is possible to define that one pixel has one luma, one blue chroma and one red choma sample associated with it.

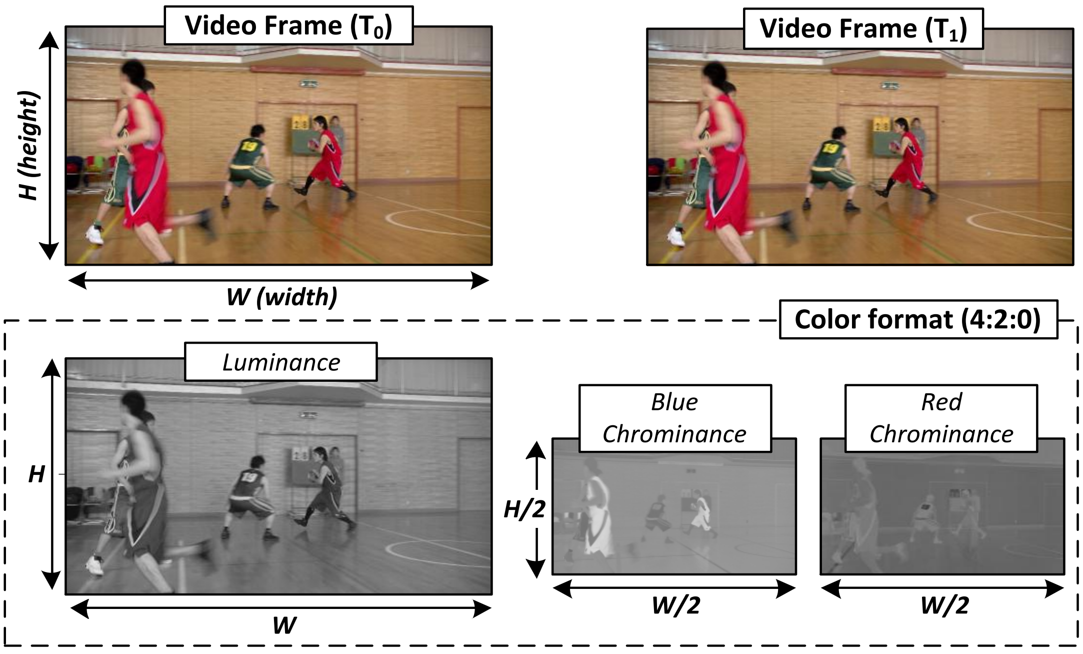


Figure 2.1: Basic concepts of digital videos: YCbCr color space representation of digital videos; 4:2:0 color format (supported by HEVC); temporal and spatial redundancies exemplification.

Digital videos, even in the raw format (without any compression), can have their samples matrices subsampled according with the predefined *color format*. When considering the YCbCr space, luma is the most important channel, regarding its visual contribution for the human eye perception. Therefore, the resolution of chroma channels can be reduced with imperceptible losses in the subject video quality. The most common formats are 4:4:4, 4:2:2 and 4:2:0. In 4:4:4 format (no subsampling), there is no chroma resolution reduction and each pixel has its own luma, blue chroma and red chroma samples. The 4:2:2 and 4:2:0 are subsampled ones, where the chroma matrices dimensions are reduced. In these cases, one chroma sample (red and blue) is used to express the color property of more than one pixel.

HEVC encoders widely adopt the 4:2:0 format as input digital video format (SULLIVAN et al., 2012). In this scenario, the blue and red chroma matrices are 2-subsampled in both horizontal and vertical dimensions (as depicted in Figure 2.1). As result, one chroma sample is inherited by four neighboring pixels. Therefore, by only applying this subsampled color format, the representation of one video frame is reduced by 50%.

Important characteristics from digital videos are related to the high data redundancy within them. The main goal of video encoders is to exploit such redundancies to reduce bit representation, while keeping the original visual quality; or, at least, minimizing the quality drops. The most representative data redundancy is noted when temporal-consecutive frames are observed (called *temporal redundancy*). Due to the high frame rate of digital videos (as already discussed), it is highly probable that almost the same scenario is represented by neighboring pictures (considering the exhibition order). Taking a 30-fps digital video as example, where consecutive pictures are captured with a time difference of 1/30 seconds (about 33 milliseconds), it is intuitive to conclude that most part of the scene is similar. The variation between adjacent frames, in this case, is mostly derived from the objects displacement inside the scenario during the capture time. This means that the objects stay inside the frame, but in a different position. Throughout the text, these characteristics will be referred as *video motion properties*. Video encoders strongly take advantage from the temporal redundancy by applying intricate motion search algorithms to capture these properties. The inter-frame prediction is the coding tool responsible for this exploitation (RICHARDSON, 2004).

In addition to the temporal redundancy, there is a substantive potential for data representation reduction when exploiting homogeneous regions in neighboring pixels inside a frame (called *spatial redundancy*). Picture areas that represent a scene background, or a clean and blue sky are practical examples of high homogeneity. Furthermore, even well-behavior textured regions present spatial redundancy that can be exploited. The spatial redundancy is exploited by the intra-frame prediction tool (RICHARDSON, 2004).

At last, there is also the entropic redundancy, which refers to the exploitation of different frequency of occurrences for distinct encoded symbols in a video. This issue is similarly handled with ordinary data compression algorithms, where high-frequent symbols are coded with less bits. Still, video coding tools apply specialized data compression algorithms to achieve higher compression rates (RICHARDSON, 2004).

Given some basics regarding digital videos characteristics, preliminaries about video coding are presented as follows.

2.1.2. *Hybrid Video Compression Model*

The HEVC, as well as recent previous video coding standards, is based on the hybrid video compression model, which is composed of the following functional blocks: (a) prediction operations (intra- and inter-frame predictions), (b) de-correlating transforms, (c) quantization and (d) entropy coding. Basically, the prediction blocks exploit already coded data (within or across video frames) to best represent the video region that is being coded. This predicted representation may not be exactly the same as the original. Thus, the difference (called residue) must considered in order to guarantee the quality of the coded video. Transforms and quantization strongly acts to reduce the residue representation. All generated information is, then, processed by the entropy coding part.

Figure 2.2 graphically depicts the hybrid video compression model flow to encode (compress) a given input digital video. The presented coding tools process the video by initially dividing the color matrices into smaller regions. At this part of the text, these smaller regions of the frame will be generically called *blocks*. One of the main HEVC innovations is the definition of very flexible data structures that enables the selection of variable-sized frame blocks. The goal is to adapt the block size to the video content properties: detailed frame regions can be split to smaller block sizes, while homogeneous areas can be grouped into larger blocks.

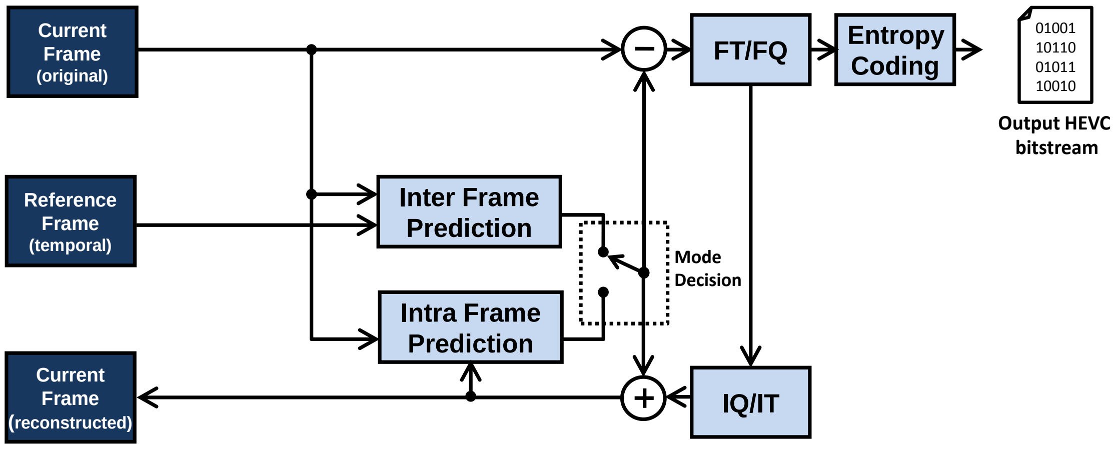


Figure 2.2: Block diagram of a video encoder (following the hybrid compression model).

Initially, the main goal of a video encoder is to analyze different ways of representing each block of the *current frame* (frame that is being currently processed) using already coded information: from the already coded blocks (1) of the current frame (spatial redundancy exploitation), or (2) of the past frames (temporal redundancy exploitation). This already processed information serves as reference to represent the current block. Thus, instead of including the whole pixels information in the final bitstream, the encoder will generate coding information to express the right way of reconstructing the current block in the decoder side.

The above presented task is executed by the prediction step and represents the main core of the hybrid video compression model. The intra-frame prediction is responsible to analyze the spatial correlation (within the current frame), while the inter-frame prediction analyzes the temporal motion properties (across the past frames). The prediction step generates two main information: the predicted block and prediction mode. The *predicted block* represents the best possible representation of the current block solely through the use of already coded reference blocks. In the same direction, the prediction mode is a video coding control entity that stores the selected way of using the reference data to re-generate the predicted block. Note that the prediction mode is important for the decoder side, since it has to reconstruct the predicted block to rebuild the original video. As already mentioned, the predicted block frequently differs from the original block. Hence, simply discarding this difference may result in significant losses during the encoding process. To handle this issue, this data (called *residual block*) is also sent into the output bitstream. In the decoder side, the predicted block is generated by having the prediction mode information, and further added to the residual block to rebuild the original block.

In hybrid video compression, there is a special treatment path to compress the residual data: composed of the forward transforms and quantization steps (FT/FQ modules in Figure 2.2). Before being processed by the entropy encoding, the residue is applied to mathematical transforms operations. The goal of the transforms is to convert the samples values from the spatial to the frequency domain, in order to de-correlate the residue and concentrate the high-frequency elements in a few coefficients. Over the transforms output, the quantization eliminates the small values associated to spectral components that are not perceptually relevant, typically generating sparse matrices of samples (with many near-zero values). It is important to highlight that the quantization inserts losses in the residual data. The strength of the quantization cuts is controlled by the Quantization Parameter (QP): higher is the QP higher is the quantization strength, leading to more losses during the encoding process. In the same vein, these higher losses are generally followed by higher compression rates. The QP is frequently used to adapt the required bandwidth to transmit the output bitstream over an unstable communication channel (VIZZOTTO et al., 2012).

At the end of the encoding flow, the entropy coding applies data compression algorithms over all generated data (residue, prediction modes, and other video coding control information) to reduce its representation and to pack them into pre-defined encoding units (SZE; BUDAGAVI; SULLIVAN, 2014). The entropy coding output is generally referred as *bitstream*, which is sent for properly transmission or storage.

Figure 2.3 depicts the block diagram of the decoder side. As input, the decoder receives the encoded bitstream and it is initially processed by the entropy decoding. In this step, all the encoding structures are decoded and reorganized to be sent to their respective modules. The residue part is reconstructed by the inverse quantization and transforms steps (IT/IQ module in Figure 2.3). The adopted prediction modes (decided during the encoding process) are forwarded to the intra- and inter-frame decoding. The task executed by these modules is to recreate the predicted block (the same as in the encoder side). By having the predicted block and the reconstructed residue block, the last step is to add them: generating the reconstructed block. This flow is performed for every block of the video, leading to the generation of the reconstructed video. Due to the quantization inserted drops, the reconstructed video is always different from the original one[4].

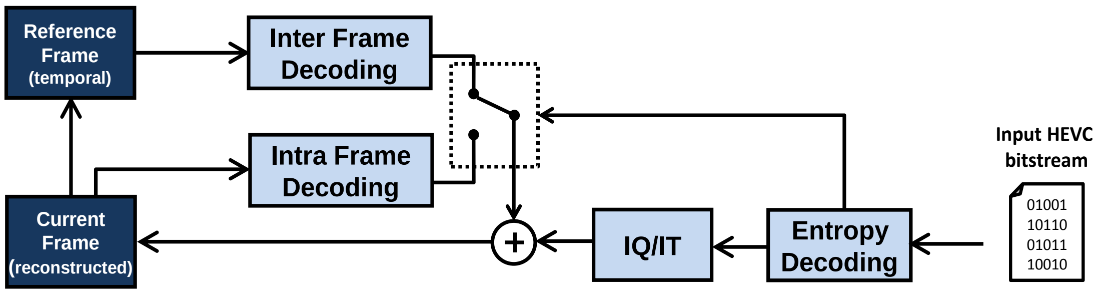


Figure 2.3: Block diagram of a video decoder (following the hybrid compression model).

Still observing the encoding process illustrated in Figure 2.2, there is a frame reconstruction loop inside the encoder. When analyzing the video decoder perspective, the reference data used to reconstruct the video frame blocks is rebuilt in a lossy scenario, since the quantization (applied in the encoder side) inserts errors in the residue. To ensure consistency in both sides, the reference information used by the decoder (to reconstruct the video) must be the same as the adopted by the encoder side (during the prediction steps). Hence, it is necessary to insert a reconstruction path to create the same references that will be used in the decoder. This reconstruction loop is composed of inverse versions of transforms and quantization steps (see Figure 2.2). The reconstructed residue, which is lossy due to the quantization, is added to the predicted block to compose the reconstructed frames. The reconstructed frames (also known as *reference frames*) must be locally stored during the encoding process, since they are used for the intra- and inter-frame predictions.

2.1.3. *Rate-Distortion (R-D) Cost*

The efficiency of compression algorithms are usually measured using the rate-distortion (R-D) cost (SULLIVAN; WIEGAND, 1998). The *rate* metric is related to the size (number of bits) of the generated output bitstream. The *distortion* corresponds to the objective quality measure of the drops inserted during the encoding process. The R-D cost is the usually adopted metric to express the *coding efficiency* of a video encoder. Equation (1) presents the R-D cost mathematical definition, where λ is the lagrangian parameter that correctly weights the tradeoff between distortion and bitrate size (depending on the adopted QP).

$$\mathrm{RD}_{\mathrm{Cost}} = D + \lambda R \qquad (1)$$

From this definition, it can be affirmed that the main optimization problem related to video compression algorithms is to minimize the R-D cost, leading to the maximization of the coding efficiency. It is important to note that every decision made during the encoding process will have repercussion (positive or negative) in the coding efficiency. Thus, the optimal coding efficiency is achieved when the R-D cost is used as metric for every required decision during the prediction steps over each current block of the input video. This leads to a huge complexity to video encoders. This method is called rate-distortion optimization and its adoption in a HEVC encoder will be presented in Section 2.2.2.

As follows, some widely-used distortion metrics are presented.

[4] Except when QP is equal to zero, which leads to a lossless compression. This scenario is rarely used in practice.

2.1.4. *Distortion Metrics*

The most used objective distortion metric is the Peak Signal-to-Noise Ratio (PSNR) (RICHARDSON, 2004), which is defined in Equation (2), where MAX is the maximum value that a sample can assume (2^n-1, where n is the number of bits of a sample), and the MSE is the Mean-Squared Error for image or block. The MSE is calculated as in Equation (3), where W and H represents the image (or block) horizontal and vertical dimensions, and O and R represent the original and the reconstructed luma or chroma samples, respectively. In these calculations, the MSE metric is the one that objectively expresses the difference (distortion) between the samples from the two frames or blocks. Another interesting distortion metric is the Sum of Absolute Differences (SAD), which is defined by Equation (4). SAD is a low-complexity metric that is intensive used in hardware implementations of video coding modules.

$$PSNR_{dB} = 10 \cdot \log_{10}\left(\frac{MAX^2}{\sqrt{MSE}}\right) \tag{2}$$

$$MSE = \frac{1}{W \cdot H}\sum_{i=0}^{W}\sum_{j=0}^{H}(R_{i,j} - O_{(i,j)})^2 \tag{3}$$

$$SAD = \sum_{i=0}^{W}\sum_{j=0}^{H}(R_{i,j} - O_{(i,j)}) \tag{4}$$

2.2 High-Efficiency Video Coding

The High-Efficiency Video Coding (HEVC) reunites the accumulated experience of around four decades of video coding community researches and barely three decades of international standardization. The result of such efforts was formally standardized as ITU-T Recommendation H.265 and ISOIEC International Standard 23008-2 (MPEG-H part 2). The first version of HEVC was completed in January 2013 (but formally released in a few months later) (ISO/IEC-JCT1/SC29/WG11, 2013a). Specifically to develop the HEVC standard, a patternship arrangement was formalized as a new joint organization, called Joint Collaborative Team on Video Coding (JCT-VC) (JCT-VC, 2017). The JCT-VC organized four meeting per year after its creation. Each of these events had hundreds of attendees and involved hundreds of submitted contributions that were analyzed to be incorporated in the final standard.

HEVC offers the same basic proposition today when compared to H.264/AVC, in the time of its development and subsequent release: double the compression efficiency. This means that HEVC is able to compress digital videos twice as much as H.264/AVC without any repercussion in the video quality. In the same perspective, HEVC achievements allows the support of higher resolutions, frame rates or video fidelity levels, and keep the output bitrate compliant with the transmission and storage infrastructure (SULLIVAN et al., 2012).

The HEVC encoding process follows the block-based hybrid video compression model, as already introduced in Section 2.1.2. In Figure 2.4 a typical HEVC encoder is presented. It can be noted a similar flow of operations when compared to the already discussed diagram of Figure 2.2. Initially, the input video frame is subdivided into fixed-size blocks (called Coding Tree Units in HEVC). Starting from the maximum size, it is subdivided into smaller coding blocks respecting a quadtree structure (called coding units). This structure allows high flexibility for variable-block size coding (properly explained in Section 2.2.1). Each coding unit is, then, processed by inter- and intra-frame prediction engines; generating prediction units. The inter-frame prediction has two main modules: the Motion Estimation, which is composed of a motion search engine to derive the motion properties of the current coding unit; and the Motion Compensation, which is responsible for getting Motion Estimation outputs (motion vector and reference frame index; further explained in Section 2.2.3) and generating the prediction information for the current coding unit. The mode decision selects the best prediction mode and, then, the residual information is calculated. The residue is processed by transform and quantization. HEVC also defines a quadtree structure for the transforms, organizing the residue into transform units to enable variable transform size operations. Finally, the quantized residue is encoded by the entropy coding, which implement the Context-Adaptive Binary Arithmetic Coding (CABAC in Figure 2.4). To generate the reconstructed samples, the quantized residue is applied to inverse transforms and quantization, added to the predicted coding unit and processed by in-loop filters. HEVC defines the Deblocking Filter and the Sample Adaptive Filter to improve subjective video quality by eliminating artifacts generated by the block-based compression tools (SULLIVAN et al., 2012). Note that the reconstructed blocks are used as spatial and temporal references for intra- and inter-frame prediction operations (see Figure 2.4). To support this, a Decoded Picture Buffer (DPB) is required to keep stored the reconstructed frames (also called reference frames). The frames in the DPB must be available to serve as reference or inter-frame predictions of next frames.

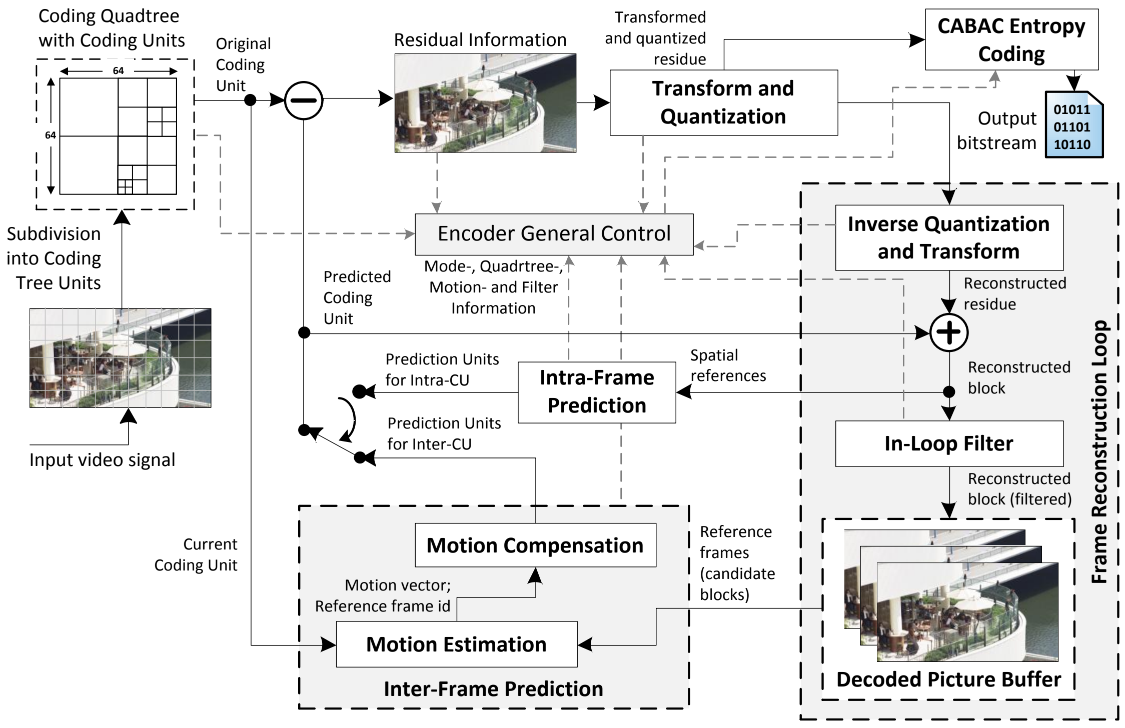

Coding Quadtree with Coding Units
64
64
Original Coding Unit
Residual Information
Transformed and quantized residue
CABAC Entropy Coding
Transform and Quantization
Output bitstream
01011
01101
10110
Subdivision into Coding Tree Units
Encoder General Control
Inverse Quantization and Transform
Predicted Coding Unit
Mode-, Quadrtree-, Motion- and Filter Information
Reconstructed residue
Prediction Units for Intra-CU
Intra-Frame Prediction
Spatial references
Reconstructed block
Input video signal
Prediction Units for Inter-CU
In-Loop Filter
Reconstructed block (filtered)
Frame Reconstruction Loop
Current Coding Unit
Motion Compensation
Reference frames (candidate blocks)
Motion vector; Reference frame id
Motion Estimation
Inter-Frame Prediction
Decoded Picture Buffer

Figure 2.4: HEVC block diagram for the encoding process
(with integrated frame reconstruction loop).

Source: SZE; BUDAGAVI; SULLIVAN, 2014

One of the main innovations of HEVC is related to a novel organization of the coding structures. In the H.264/AVC standard, the frame was initially partitioned in fixed-size macroblocks of 16x16 samples (WIEGAND et al., 2003). The prediction and transform steps were able to further partition a macroblock into smaller sizes: 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4 (smallest block size). In HEVC, the block partitioning starts with 64x64 size, which enables a better fit to higher resolutions videos. This 64x64 block is the start point of a quaternary-tree based partitioning structure, called *Coding Tree Unit* (CTU). During the prediction steps, the block is split into four smaller sized ones, called *Coding Units* (CU), and the prediction operations (intra- and inter-frames) are applied to the new blocks. This process is executed in a recursive way, until the smaller supported block size (8x8). Still, given a specific coding unit of a specific size inside the quaternary tree, the prediction step can further partition the block in eight different formats (four symmetric and four asymmetric). The information regarding the prediction steps (like prediction modes, motion information and adopted partition shape) are represented by a *Prediction Unit* (PU). Furthermore, when the residue is generated (difference between predicted and original samples), the transform module can also be applied to different block sizes, called *Transform Units*. The main goal such innovations is to enable high flexibility during the encoding process to adapt the compression tools to the video content. **Section 2.2.1** will present the HEVC coding structures in a deeper level of details.

The prediction step, responsible for exploiting the reference data to find the most efficient way of represent the current block information, is divided into intra- and inter-frame predictions. The intra-frame prediction, which decreases the spatial redundancy, uses the samples of already coded neighboring PUs (within the same frame) to predict the current one. HEVC supports 35 different modes for intra prediction: 33 angular modes, a flat mode, and a planar mode (SULLIVAN et al., 2012). In H.264/AVC, only nine modes were defined to 4x4 blocks and four modes for 16x16 blocks (WIEGAND et al., 2003).

Besides the spatial dependencies within a frame, the temporal redundancies (motion properties) are reduced by the inter-frame prediction. The core of this step is the Motion Estimation (ME), which is composed of a block-matching search engine that scans the reference frames to find for the most similar block to represent the current PU. To represent the displacement of the best-match block in the reference frame, the ME produces a motion vector for each PU partition. HEVC supports motion search in one quarter of pixel accuracy for luma samples and one eighth of pixel accuracy.

The compression gains obtained from the more flexible coding structures comes with the cost of increased computational complexity for the encoders. In HEVC scenario, without any complexity reduction technique, intra- and inter-prediction must be executed for all partitions of every PU possibility of each CU analyzed within the quad-tree structure of each CTU inside the video frame. For optimal coding efficiency, the Rate-Distortion Optimization (RDO) is adopted to determine CTU division and the best prediction modes for each selected PU. The RDO is based on calculating the R-D cost (explained in Section 2.1.3) for each PU possibility inside the CTU. This leads to a huge computational cost, since barely entire encoding and decoding flows are necessary to estimate the R-D metric for each analyzed PU. **Section 2.2.2** presents further explanations about the RDO inside the HEVC encoding.

Among all prediction operations, ME inherits the most computational complexity, being the main bottleneck in terms of execution time and required memory bandwidth. Thus, techniques to optimize these issues around ME execution have significant impact in the overall HEVC encoder. **Section 2.2.3** presents further details of inter-frame prediction, highlighting important concepts of ME.

To handle this high computational complexity, HEVC defines a special data partitioning way that facilitates parallel processing: the video frame division into rectangular regions called *Tiles*. The main idea is: coding units belonging to different Tiles have broken data dependency and, thus, can be processed in parallel. Therefore, all CTUs of a Tile can be assigned to a specific processing core. Another parallelism opportunity defined by HEVC is related to an alternative processing order of the coding units, called Wavefront Parallel Processing (WPP), where a much finer parallelism degree can be exploited by dividing the frame into rows of coding units (SZE; BUDAGAVI; SULLIVAN, 2014). **Section 2.2.4** further describes these parallelism features introduced by HEVC, focusing on the Tiles partitioning.

2.2.1. *HEVC Coding Structures*

2.2.1.1. Video Partitioning Structures

In HEVC, the input video sequence is divided into Group of Pictures (GOP). The GOP reunites at least two consecutive frames to constitute Random Access Points (RAP) from which the decoder can start decoding without direct dependency with any previous frames. Figure 2.5 illustrates the division of the frames sequence into GOPs of size four.

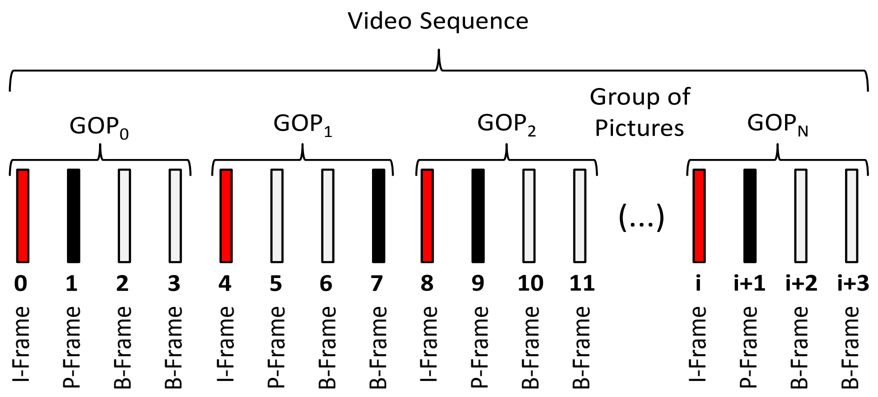

Video Sequence
GOP$_0$
GOP$_1$
GOP$_2$
Group of Pictures
GOP$_N$
(...)
0 1 2 3 4 5 6 7 8 9 10 11 i i+1 i+2 i+3
I-Frame P-Frame B-Frame B-Frame I-Frame P-Frame B-Frame B-Frame I-Frame P-Frame B-Frame B-Frame I-Frame P-Frame B-Frame B-Frame

Figure 2.5: Frames of a digital video grouped by GOPs.

The frames in a video sequence are classified as I-, P- or B-frames. In an I-frame, all CTUs are encoded only using spatial references, e.g., only intra-frame prediction is performed. This means that the decoding process for I-frames does not depends on the reconstructed data of other frames. I-frames are important for the first encoded frame and to insert RAPs to allow decoder synchronization, which is important for continuous video streaming applications. Further, P- and B-frames are encoded using both intra- and inter-frame predictions. The P-frames have temporal dependencies with only one reference frame, while B-frames utilize the bi-prediction that allows references to more than one reference frame.

2.2.1.2. Frame Partitioning Structures

Each frame is partitioned into square-shape *Coding Tree Units* (CTUs). The CTU represents the basic processing unit in HEVC and it is in that regard similar to the concept of a macroblock in prior video coding standards. HEVC does not set a fixed size for the CTU. However, typical HEVC applications adopted the largest possible size defined in the standard: 64x64. The CTU structure comprehends a quadtree structure, also referred as coding tree, which specifies the CTU division into *Coding Units* (CUs). Figure 2.6a depicts the quadtree structure of a CTU and exemplify its possible subdivision into variable-sized CUs. Similarly to the CTU, a CU consists in a square block of samples. At CTU level (depth 0 in Figure 2.6a), a flag into the bitstream indicates whether the complete CTU represents a CU or whether it will be divided into four equally-sized blocks. If the CTU is split, four new CUs are formed (depth 1). In a recursive way, each one of the new CUs can be further split into four new blocks (depths 2 and 3). This hierarchical subdivision process ends when the minimum CU size is reached. In typical HEVC applications, the minimum size of a CU is 8x8 samples. The final CTU division of the given example can be observed at Figure 2.6b.

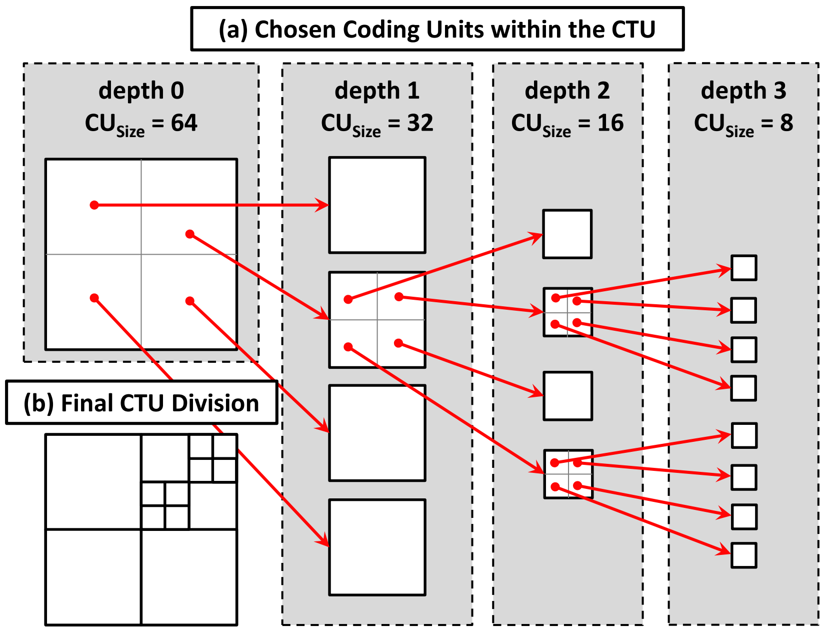


Figure 2.6: (a) Example of adopted CU sizes within a CTU and
(b) final CTU division for video frame texture adaptation.

This flexible encoding structure can adapt the CUs sizes according to the video content: larger CUs are used to exploit homogeneous areas, while smaller CUs can have a best fit into textured regions. When larger CUs are chosen, there are gains in the compression rates, since the CTU will be composed of fewer CUs and, this way, less coding control symbols are generated. However, if not well predicted, large CUs may incur in more residual information to be encoded, leading to higher distortion due to quantization cuts. In the other hand, CTUs divided into smaller CUs will require more control information. Analogously, smaller CUs better adapts the coding granularity to the video texture properties, reducing the inserted distortion in the residual treatment path. Therefore, the decisions involved in the CUs subdivision have a key role to provide high coding efficiency to HEVC encoding.

The CUs represent the encoding structures to which a coding mode is assigned. For each CU, it is decided whether the luma and chroma samples are predicted using intra- or inter-frame prediction. Figure 2.7a graphically illustrates this prediction dynamics for each CU of size 2Nx2N. All information related to the adopted prediction for the CU is represented by a Prediction Unit (PU). Note that different CUs within the same CTU may have different prediction modes (intra or inter). Additionally, for each one of prediction options, a CU can be further split during the prediction step. Each CU may be partitioned into two or four PUs, which are separately predicted. An important definition is that all PUs of a CU should be predicted with either as inter or as intra. Thus, an entire CU can be classified as inter-CU or intra-CU. HEVC supports eight different modes for partitioning a CU into PUs, as illustrated in Figure 2.7b: four symmetric and four asymmetric partitioning.

If a CU is signaled as intra, the corresponding PUs must stores one of the 35 supported spatial intra prediction modes for luma samples[5]. HEVC defines that intra-CUs can only support 2Nx2N and NxN partitions. For inter-CUs, all eight partitions can be assigned. Each PU of an inter-CU should store, among others, the motion vector and the used reference frame(s) index(es).

[5] For chroma channels, one of 5 available modes should be selected.

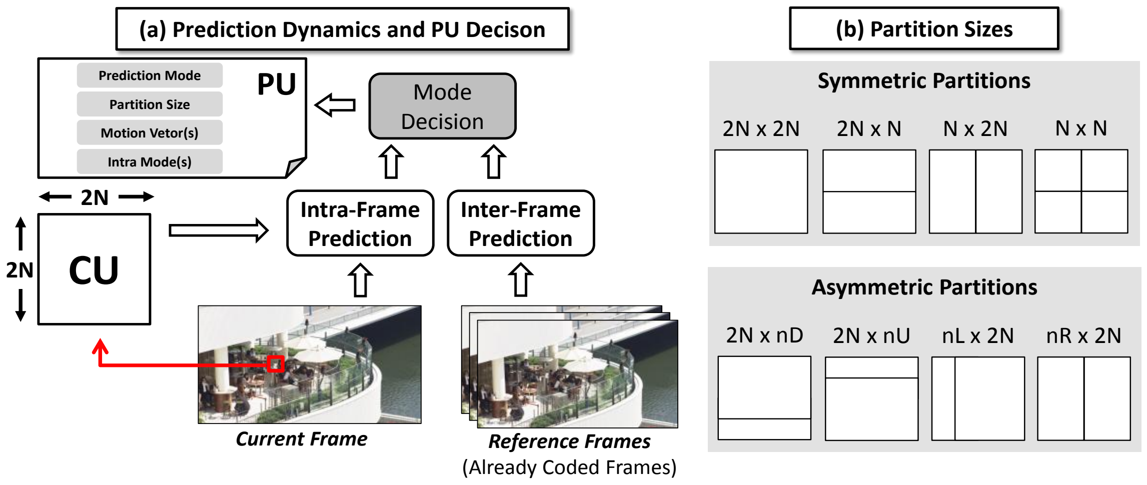


Figure 2.7: (a) Prediction dynamics for each CU within the coding tree and the generation of the PU; (b) Partition sizes allowed by HEVC into PUs (intra PUs only support 2Nx2N and NxN partitions).

The Mode Decision (gray module in Figure 2.7) is the responsible for deciding the best PUs for each CU inside the quadtree structure. Note that the decisions about the best prediction mode, allied to the already discussed decisions regarding the best CTU division into smaller CUs, represents the key challenge of the general control of a HEVC encoder. In a simplistic analysis, the prediction step of a 64x64 CTU has to analyze both intra- and inter-CU options for each level of the coding tree structure. This leads to one 64x64 CU, four 32x32 CUs, sixteen 16x16 CUs and sixty four 8x8 CUs (last level). For each one of these 85 possible CUs, two different partitioning ways must be evaluated by the intra-frame prediction, while eight are evaluated by inter-frame prediction. In total, 149 intra-PUs and 849 inter-PUs must be analyzed by the Mode Decision to process an entire CTU. Considering a 1080p input video sequence (1920x1080 pixels) that comprehends 506 CTUs, more than 500,000 PUs should be predicted and its coding efficiency analyzed. In HEVC reference software HM (ISO/IEC-JCT1/SC29/WG11, 2013b), the RDO is adopted as optimal scheme for the Mode Decision module, generating the upper-case scenario in terms of coding efficiency (considering the R-D cost metric). Details regarding the RDO will be presented in the next section.

2.2.2. *Rate-Distortion Optimization (RDO)*

The RDO represents an upper boundary in terms of coding efficiency, since its main strategy is to perform a complete depth-first search for all configurations of CUs and PUs. The goal is to ensure optimal decisions when encoding each CTU within a video frame.

RDO is based on the minimization of the R-D cost, which relates the bitrate size with the output video distortion (already discussed in Section 2.1.4). Figure 2.8 presents a recursive function (called *compressCU*) that depicts the idea of the RDO execution when analyzing a given coding unit *CU* of *d* depth in the quadtree structure. This function returns, at the end of its execution, the best R-D cost of encoding the current CU.

First, intra- and inter-frame predictions are triggered to exploit spatial and temporal dependencies. As result, the RDO estimates the R-D cost for the best PUs generated by the prediction steps for the current *CU* (RD_{Inter} and RD_{Intra} – *lines 1 and 2*). At this point, all partitioning ways for the PUs described in 2.2.1 were analyzed. Then, the RD_{Best} is calculated as the best (minimum) R-D cost between intra-CU or inter-CU choices (line 3). RD_{Best} represents the minimal R-D cost achieved by not splitting the current *CU* into smaller ones. To evaluate the cost of going through the next quadtree depth, the same function is recursively called for the next-depth *sub-CUs* (CU_0, CU_1, CU_2 and CU_3). The best R-D costs estimated for each one of the smaller CUs are accumulated in the $RD_{SubBest}$ (*lines 4-8*). It is important to note that the recursive calls to *compressCU* leads to execution of the same explained steps for each one of the sub-CUs; and this process is recursively executed until the smallest CU size is reached. At the end, the RDO returns the minimum R-D cost between the choice of not splitting the CU and the option of dividing the CU into sub-CUs (line *9*).

```
Algorithm: compressCU(Coding Unit: CU, Depth: d)
1.  $RD_{Inter}$ = checkRDCostInter(CU, d);
2.  $RD_{Intra}$ = checkRDCostIntra(CU, d);
3.  $RD_{Best}$ = min($RD_{inter}$, $RD_{intra}$);
4.  $RD_{SubBest}$ = 0;
5.  $RD_{SubBest}$ += compressCU($CU_0$, d+1);
6.  $RD_{SubBest}$ += compressCU($CU_1$, d+1);
7.  $RD_{SubBest}$ += compressCU($CU_2$, d+1);
8.  $RD_{SubBest}$ += compressCU($CU_3$, d+1);
9.  return min($RD_{SubBest}$, $RD_{Best}$);
```

Figure 2.8: Rate-Distortion Optimization (RDO) algorithm.

Besides the exploitation of all possible CU sizes within the quadtree, the RDO decisions for the best configuration are based on the R-D cost. This means that, for every analyzed intra- or inter-CU option, the HEVC encoder must estimates the final bitrate repercussion of such decision, as well as the impact on the reconstructed frame distortion. Based on the HEVC encoder diagram of Figure 2.4, the bitrate size estimation requires the residue generation, and posterior transforms, quantization entropy coding processing. Furthermore, the distortion knowledge can only be generated by calculating the residue, applying transforms, quantization and then all frame reconstruction loop process (highlighted in Figure 2.4): composed of inverse transforms, inverse quantization and final filtering operations. In summary, RDO implements a brute-force algorithm, which analyzes every possibility, and further uses the R-D cost as comparison metric, requiring high computation to generate the rate and distortion metric for each analyzed situation. For the above discussed reasons, RDO is not used in practical implementations of HEVC encoders. To handle with this issue, many researches aims on proposing complexity reduction compared to RDO (CORREA et al., 2012, 2013).

The most computational complex module of HEVC encoding is the inter-frame prediction step. Furthermore, the memory related issues of HEVC are strongly related to this module. The RDO usage as mode decision strategy highly aggravates these penalties, since inter-frame prediction must be executed for every tested inter-CU configuration. Section 4.1 motivates it by presenting a HEVC memory profiling that indicates the inter-frame prediction, particularly the Motion Estimation as the main memory bottleneck. As the focus of this work is to provide energy-efficiency memory architectures for HEVC encoding, the inter-frame prediction and the Motion Estimation aspects will be further discussed as follows. Additionally details if intra-frame prediction dynamics will not be presented in this text, and can be encountered in (SZE; BUDAGAVI; SULLIVAN, 2014).

2.2.3. *Inter-Frame Prediction*

In HEVC applications, there are two typical adopted prediction structures, which are depicted in Figure 2.9. The numbers in Figure 2.9 represents the encoding frame sequence, while the left-to-right frame disposition is related video exhibition order. In the *Low Delay* configuration, each frame is only able to use past frames (in the exhibition order) as temporal references. This leads to the same encoding and exhibition order. The *Random Access* configuration defines a hierarchical B structure, which defines temporal layers: represented by different shades of gray for the B-frames. Another detail is related to an alternative encoding order: initially, the first and the last frame are processed (first temporal layer; I- and P-frames, respectively); then, the fourth frame is encoded as a B-frame (second temporal layer), using reference frames from the previous layer; after, the frames of third temporal and the fourth (last) temporal layers are processed. The hierarchical B structure leads to different coding and exhibition orders. There is also the *Intra Only* configuration, where only I-frames are allowed (no inter-frame prediction is performed) and there are no dependencies between frames.

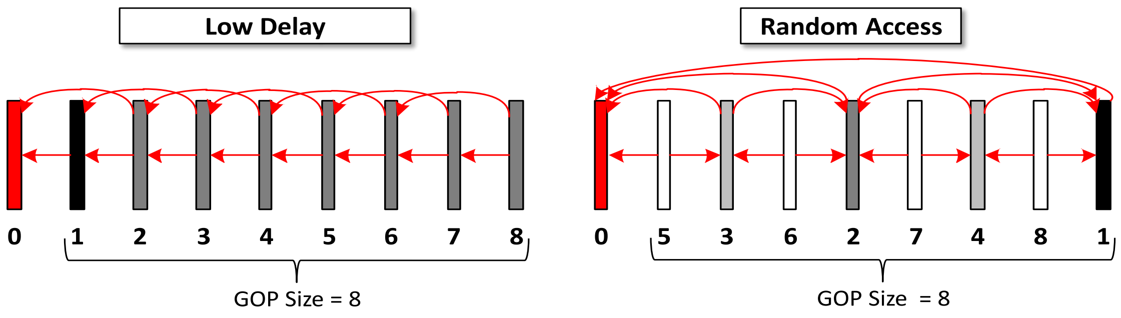


Figure 2.9: Common prediction structures utilized by HEVC applications: Random Access and Low Delay.

The arrows of Figure 2.9 indicate the inter-frame prediction direction: the arrows start from the current frame and arrive at the reference frames, indicating the temporal dependencies during the encoding process. To be used as reference, the frame must be previously encoded, reconstructed and entirely available in the DPB (typically assigned to an external memory in HEVC encoders).

An example: considering the Low Delay configuration and assuming that the frame 2 (exhibition and coding order) is, at the moment, the current frame to be processed by inter-frame prediction. This frame has temporal dependencies with frames 0 and 1, as depicted in Figure 2.9. This means that, in both encoding and decoding sides, frames 0 and 1 must be previously processed and reconstructed before frame 2 processing is started. When analyzing the frame 2 (coding order) in the Random Access structure, its processing depends on the frames 0 and 1 (coding order). In this case, the dependencies are past (frame 0) and future (frame 1) frames in the exhibition order. The utilization of future frames as reference is an important issue of recent video encoders.

Motion Estimation (ME) is the main core of the inter-frame prediction, being responsible to capture the temporal correlations between temporal neighboring frames within a video sequence. Figure 2.10 illustrates the ME processing. For each analyzed PU to be associated to an inter-CU in the current frame, the ME is applied according to the adopted prediction structure. The main goal of ME is to find the best match of each analyzed PU (called *current PU*) of the current frame using, as reference, one or more reconstructed frames (aka. reference frames). The optimal best match corresponds to the block in the reference frame that minimizes the R-D cost in the final HEVC processing. As already discussed, the optimal solution for the best coding efficiency is a computation-intensive task, since almost entire encoding and decoding flows must be executed to estimate the R-D cost for each possible matching block in the reference frames. As local decision in ME, the block that minimizes the residual information (difference between original and predicted blocks; see Section 2.1.2) is the chosen one as near-optimal result. In this sense, a low-complexity similarity metric is used to measure the amount of generated residue.

Using this metric as basis, the current PU is compared to a subset of blocks of the reference frames (called *candidate blocks*) and the most similar one is selected as the best match. As result, the ME delivers (1) a motion vector indicating the displacement between the current PU position and the best match, as well as (2) reference indexes that refers to the selected reference frames.

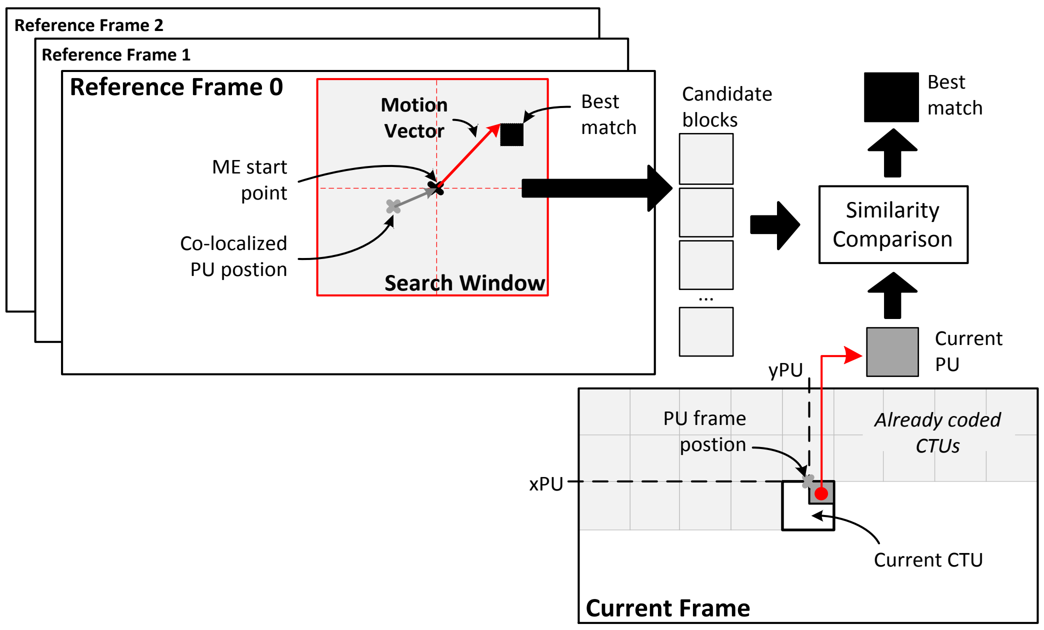


Figure 2.10: Basic concepts related to the Motion Estimation process.

Due to the typical adopted frame rates, the best match tends to be found in positions closed to the current PU position. Thus, the ME search typically starts in reference frames positions near to the co-located current PU position. Furthermore, there is also a demonstrated correlation between the motion properties between neighboring PUs. Typical ME engines adopts an initial computation to slightly change the ME start point according to the motion vectors generated during previous ME of neighboring PUs (called *motion predictors*; illustrated in Figure 2.10). To limit the ME search range, a common decision is to restrict the candidate blocks to a squared region of the reference frame called *search window*. If motion predictors analysis is supported to displace the ME start point, the search window is moved accordingly.

After the ME, the Motion Compensation (MC) utilizes the motion vector and the reference frame indexes to build the predicted block. The MC needs to fetch (from the DPB) the candidate block that was selected as the best match for the current PU. This is necessary for the posterior residue calculation, which must be sent to transforms and quantization steps. HEVC also exploits additionally techniques to increase inter-prediction efficiency, like fractional ME (with quarter-pixel precision for luma samples). These strategies will not be discussed in this text since they are out of the scope of this work.

In order to guide the ME search within the search window, several motion search algorithms have been proposed since previous video coding standards. The HEVC HM reference software implements two important ones: the Full Search, representing the exhaustive search option that leads to the optimal case; and the Test Zone (TZ) Search (PURNACHAND; ALVES; NAVARRO, 2012), being the heuristic-based solution that provides sub-optimal results.

The idea of Full Search is to compare the current PU with all possible candidate blocks inside the search window. By starting with the candidate block from the upper-left corner of the search window, the algorithm checks its similarity with the current PU. This process is repeated for the candidate block that begins one sample to the right. The search window is scanned in a raster order until the bottom-right corner of the search window is reached. Due to this exhaustive approach, Full Search achieves the best rate-distortion results. However, the number of comparisons grows in a quadratic order. As alternative, HM implements the TZ Search as a fast search algorithm, which is based in local greedy decisions with the goal of directing the search to iteratively catch the motion. Compared to the exhaustive search option, the TZ Search achieves speedup rates of 23x with insignificant losses in the coding efficiency (PURNACHAND; ALVES; NAVARRO, 2012.

Although the motion search algorithm directly affects the ME efficiency, the similarity criterion has also an important role. A widely adopted metric for ME similarity evaluation is the Sum of the Absolute Difference (SAD), which was already presented in Section 2.1.4.

2.2.4. *Parallelism Support of HEVC: Tiles*

An important innovation of HEVC is the definition high-level parallelization features to speedup the encoding process. This work focus on the coarse-grain parallel support of HEVC called *Tiles*. When Tiles usage is enabled, the picture is divided into rectangular-shaped groups of CTUs separated by vertical and/or horizontal boundaries (MISRA et al., 2013). The number of Tiles and the local of their boundaries can be defined for the entire sequence or changed from frame to frame (SZE; BUDAGAVI; SULLIVAN, 2014). Further, the Tiles can be partitioned in uniform or non-uniform ways. Figure 2.12 depicts an example of Tiles partitioning: composed of three rows and three columns, totalizing nine Tiles disposed in a non-uniform format.

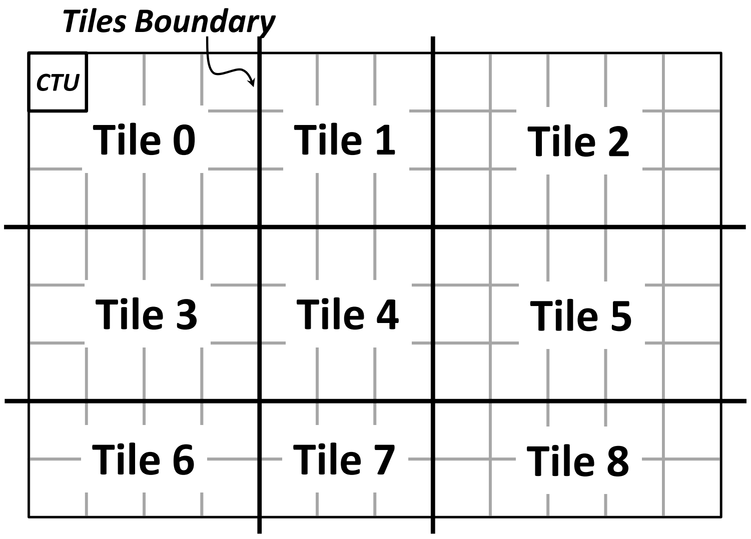


Figure 2.11: A video frame divided into nine Tiles.

Tiles boundaries do break parsing and prediction dependencies to that all CTUs within a Tile can be encoded independent from CTUs of other Tiles. Only filtering operations can still cross Tiles boundaries in order to prevent Tile border artifacts (SZE; BUDAGAVI; SULLIVAN, 2014).

Figure 2.12 depicts a practical 4-Tile partitioning and the corresponding association with a 4-core manycore processor. As already defined, the CTUs of different Tiles can be processed without any dependencies with each other. This offers the opportunity of parallel processing between such CTUs.

The main research field of this work is based on the Tiles partitioning support of HEVC encoders. There are several research challenges involved in the hardware and software support for parallel HEVC using multiple Tiles. This works focuses on solving the memory issues of supporting multiple processing cores (each one processing a specific Tile) simultaneously accessing the same infrastructure for storing and fetching data. As already motivated in Section 1.1 and further evaluated in Chapter 4, the increased computational complexity inserted by the novel coding structures of HEVC, allied to the possibility of parallel encoding, poses a challenge of ensuring energy efficiency to the encoder memory infrastructure to enable parallel HEVC encoding into embedded system applications. At the same time, the adopted memory hierarchy must ensure enough data bandwidth to properly feed the processing cores.

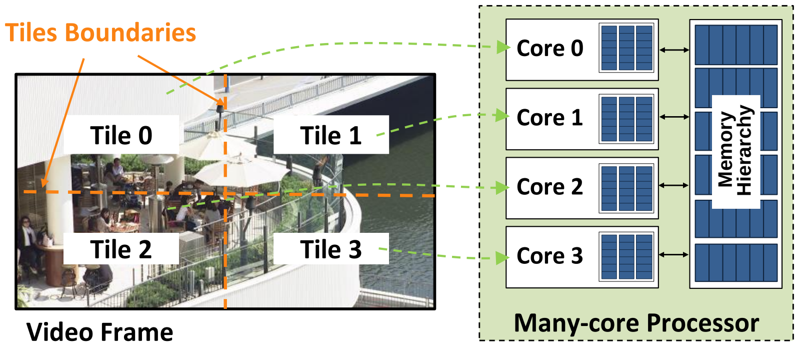


Figure 2.12: 4-Tile partitioning of a video frame and a typical processing assignment for a 4-core manycore processor.

The main propositions of this work are strongly based on exploiting novel memory technologies and organizations to compose energy-efficient video memories. Thus, the following sections introduce these memory technologies. The main intention is to define the main electrical characteristics that will base the ideas of the proposed memory architectures.

2.3 Memory Technologies

This section shows background concepts regarding the adopted memory technologies in this work. Initially, details regarding SRAM cell characteristics and its organization to compose memory arrays are presented. After, the properties of emerging STT-RAM cells are discussed.

2.3.1. *Static Random Access Memory (SRAM)*

2.3.1.1. *1-Bit SRAM Cell*

SRAM is the type of memory used as the building block of the most type of on-chip memories, like caches and scratchpad memories. As SRAM uses the same fabrication process as the nowadays processors, it leads to a simply integration onto the processor die. The basic 1-bit SRAM cell is implemented as two cross-coupled inverters, which are accessed by using two pass transistors, as depicted in Figure 2.13a. The cross-coupled connection builds a regenerative feedback that enables it to an indefinitely storage of one data bit. This configuration has one interface that allows either read or write operations, but not both simultaneously (JACOB; NG; WANG, 2010).

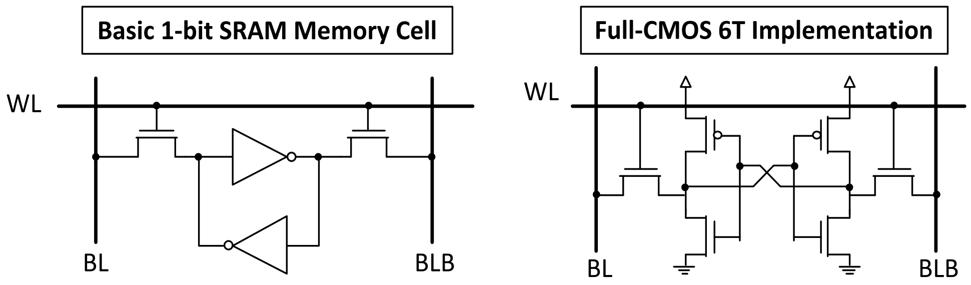


Figure 2.13: (a) Basic 1-bit SRAM cell organization and the (b) widely-adopted Full-CMOS 6T memory cell.

For a read operation, the *word line* (WL) is asserted and, as result, the stored bit is detected by the voltage differential between the bitline pair (BL and BLB). To write a specific bit into

SRAM cell, the BL pair is driven with a differential voltage from an external source to force the data onto the memory cell.

2.3.1.2. Multi-Bank SRAM Array

Figure 2.14 depicts a basic example of a N-bank SRAM memory organization. Each memory bank is composed of a matrix of SRAM cells (implemented as in Figure 2.13). The cells from the same memory line are connected by the same WL_{Row}, where *Row* represents the SRAM memory line position. Additionally, all memory cells that store the bit of same word position share the same bitlines. As memory peripherals, a row address decoder receives the input address line (3-bit address in the example of Figure 2.14) and activates only the corresponding word line. Sense and write amplifiers (SA and WA in Figure 2.14) are disposed to deal with differential encoding of bitlines and to drive the data in/out from/to input and output memory pins. Further, a precharge circuitry is required to ensure precharged bitlines at the beginning of read and write operations. To provide higher data bandwidth, the memory can be organized into multiple banks.

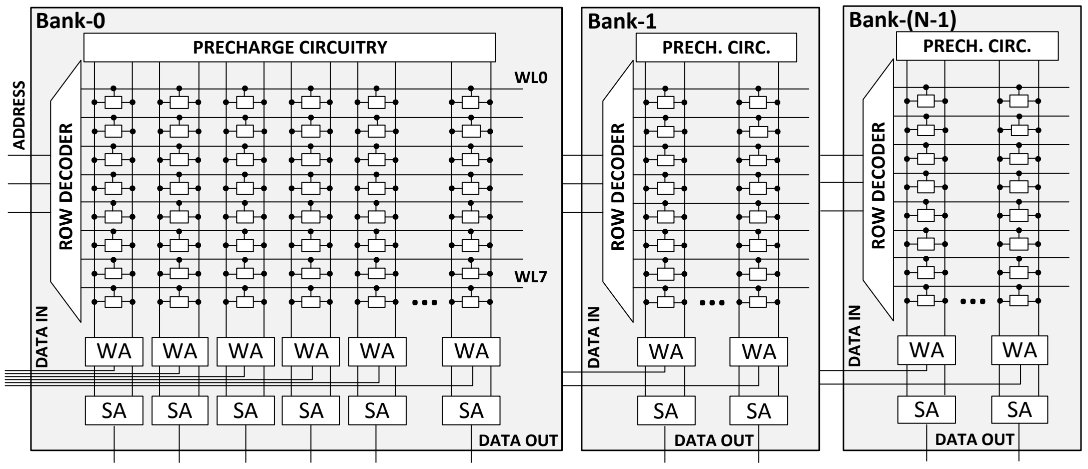


Figure 2.14: Simplistic example of an N-bank SRAM memory array: each bank composed of a matrix of 8x16 1-bit memory cells; each memory line stores a 16-bit memory word.

The following steps are necessary to perform a read operation: (1) the incoming address is decoded and one memory line is selected (WL_{Row} is set); (2) the access transistors of all memory cells of the selected WL_{Row} line are activated and the stored bits are passed to the bitlines (in a differential encoding); (3) the sense amplifiers are turned on to amplify the small difference voltage at each bitline pair into full-swing logic signals; and finally, (4) all bitlines are precharged to V_{DD} and get ready for next read/write operation.

The steps for a write access are: (1) the incoming address is decoded and one memory line is selected; (2) simultaneously to step 1, the write amplifiers generate the required voltage differential at bitlines to flip the memory cell state (when necessary); (3) cell flipping process takes place (when the stored bit is opposite to the value that is being written); and, at the end, (4) the precharging process reset the bitlines for the next access (CHENG; HUANG, 2005).

2.3.1.3. Energy Consumption in SRAM Arrays

There are two main sources of power dissipation for SRAM (also for CMOS circuits): static and dynamic energies; see Equation (5) (ZATT et al., 2016). The static power is a result of the leakage currents. When the input voltage is lower than the NMOS transistors threshold voltage, in an ideal case, the NMOS transistors do not conduce any current. However, in a real case

CMOS transistors do not completely block this current: called leakage current. The closer are the input and threshold voltages the stronger the leakage. The same happens for the PMOS transistors of SRAM cell. Equation (6) presents the static power dissipation formula, which is function of the leakage current ($I_{Leakage}$) and of the supply voltage (V_{DD}). It is important to note that the static energy is consumed even when no memory accesses are performed.

The dynamic power of SRAM cells is composed of two main components: the switching and the short circuit power components (see Equation (7)). The switching portion represents the power that is dissipated during flipping of the transistors state. It can be note in Equation (8) that the switching power (P_{Switch}) is function of: switching activity (α), operation frequency (*f*), load capacitance (C_L) and V_{DD}. Moreover, the short circuit power (Equation (9)) happens at the moment that the input transistor gate signal changes from ground to V_{DD}, and vice versa. There is a specific input voltage where both PMOS and NMOS transistors are conducing, leading to a short circuit current (I_{Short}).

$$P_{SRAM\_Cell} = P_{Static} + P_{Dynamic} \tag{5}$$

$$P_{Static} = I_{Leakage} \times V_{DD} \tag{6}$$

$$P_{Dynamic} = P_{Switch} + P_{Short} \tag{7}$$

$$P_{Switch} = \frac{1}{2}\alpha \times f \times C_L \times V_{DD}^2 \tag{8}$$

$$P_{Short} = I_{Short} \times V_{DD} \tag{9}$$

$$E_{SRAM\_Cell} = P_{SRAM\_Cell} \times t \tag{10}$$

The overall energy consumption of a SRAM cell is calculated by taken the power dissipation along the time (t); see formula in Equation (10). By analyzing the components that compose the energy consumption of a SRAM cell, some insights can be performed in order to reduce it. Adaptive power management can be performed by employing power-gate circuitry and proper management to turn of unused memory cells (V_{DD} equals to zero). Another way of managing energy is to reduce the switching activity: leading to reduced switching power (dynamic portion).

Recent manycore processors implements large SRAM arrays as last-level caches to provide inter-cores data reuse and to support the required memory bandwidth from massive parallel applications. In some cases, 128MB of on-chip storage is required for state-of-the arte manycore processors (SAMPAIO et al., 2015). Large SRAM arrays suffer with high static power dissipation due to the huge usage of memory cells, as well as their peripheral circuitry (previously explained in Section 2.3.1.2). Emerging memory technologies have emerged as an attractive alternative option for implementing large-sizes on-chip memories. The Spin-Transfer Torque RAM (STT-RAM) stands out as one of the most promising technology. This work adopted this technology to implement hybrid video memories aiming on taking advantage of the low-power features of this emerging technology. Details regarding STT-RAM memory technology and its multi-level cells design are presented in the following section.

2.3.2. *Spin-Transfer Torque RAM (STT-RAM)*

As an emerging memory technology, the Spin-Transfer Torque SRAM (STT-RAM) (DONG et al., 2008) provides higher density, better scalability and low static power features compared to the SRAM. In other aspects SRAM is still much more efficient, like in terms of write power and overall performance. Table 2.1 presents a subjective comparison between

SRAM and STT-RAM technologies[6], where the dark-gray cells represent the best scenario of each parameter.

Table 2.1: SRAM vs. STT-RAM Technologies (DONG et al., 2008)

Tech.	**Energy**			**Latency**		**Volatility**
	Static	**Read**	**Write**	**Read**	**Write**	
SRAM	HH	**L**	**L**	**L**	**L**	Volatile
STT-RAM	**L**	**L**	HH	**L**	H	**Non-Volatile**

The on-chip video memories have a particular property that facilitates the STT-RAM usage: *they have a relatively low write intensity compared to a very high read intensity* (SAMPAIO et al., 2014a). As the on-chip video memories implement data-reuse schemes for the search window samples, only a few data of the reference frame would be written to start the next CTU prediction. Once the needed data is stored on chip, the motion estimation massively accesses the on-chip video memory until the best match is found. As can be noticed in Table 2.1, the STT-RAM energy and performance are poor for write operations compared to that of the SRAM. Thus, video coding is a promising application for STT-RAM based hybrid memories.

STT-RAM is also known to be a non-volatile memory (NVM). This characteristic is very important for on-chip video memories, since parts of the memory may be switched-off (no static energy consumption) while keeping the data stored, leading to no extra external memory accesses to re-fetch the information. However, the NVM cells lifetime (aka. endurance property) *highly depends on the bit-toggling activity of the writing operations* (WU et al., 2010). If improperly balanced, the lifetime of a STT-RAM cell can be significantly reduced, compromising the overall memory system performance. Therefore, *there is a need for memory data management policies to increase the NVM lifetime in a hybrid video memory design*.

2.3.2.1. 1-Bit STT-RAM Cell

Spin-Transfer Torque RAM (STT-RAM) cell stores one logic bit in a *magnetic tunneling junction* (MTJ) – an oxide layer between two ferromagnetic layers. In this work, the “1T1J” structure was adopted for 1-bit STT-RAM cell, being composed of: (a) one NMOS access transistor with its gate connected to the WL (as in previously presented SRAM layout); (b) one MTJ that effectively stores the logical bit under magnetic principles. Figure 2.15 illustrates the schematic and the structural view of the 1T1J 1-bit STT-RAM cell. The source of the NMOS is connected to the source line (SL), and one side of the MTJ is connected to the bitline. The resistance value of the MTJ is determined by the relative magnetic field direction between these two layers (DONG et al., 2008). One layer has fixed magnetization (called reference layer). The other can have its magnetization changed due to a polarized programming current (called free layer). In a 1-bit STT-RAM cell, ‘low resistances’ due to parallel magnetization and ‘high resistances’ due to anti-parallel magnetization represent the logic bits ‘1’ and ‘0’, respectively (as shown in Figure 2.15c).

[6] The terms L, H and HH are used for a subjective comparison between STT-RAM and SRAM regarding its electrical characteristics: “L” means *low*, “H” means *high*, and “HH” means *very high*.

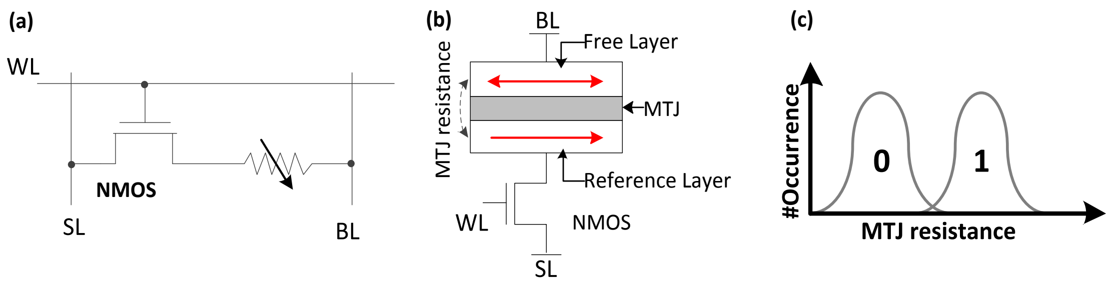


Figure 2.15: Spin-Transfer Torque RAM (STT-RAM) 1-bit cell (a) schematic view; (b) structural view; and (c) resistance range distribution to represent logic bits “0” and “1”.

The read operation respects the following steps: Initially, (1) a small negative voltage difference is applied on BL relative do SL; then, (2) this lead to a current passing through MTJ, which should be small enough to not trigger a write operation; (3) the sense amplifier compares this current with a reference one, deciding if a “0” or “1” logic bit is stored in the memory cell. To perform a write operation: (1) when “0” logic bit is being written, a positive voltage is established between SL and BL, and, for a “1” writing, vice versa; (2) the current amplitude necessary to reverse the direction of the ferromagnetic layer is determined by the MTJ size and the write pulse duration: the smaller is the MTJ or the longer the write pulse, the less the switching pulse is needed.

An important feature of recent STT-RAM technology achievements is the integration with CMOS-based SRAM arrays. The interface provided by STT-RAM cells (WR, BL and SL) is compliant to the SRAM cells (Section 2.3.1). Although fabrication process of STT-RAM incurs extra cost and additional fabrication complexity, mainly to integrate with CMOS logic, technology advances it this field have been enabling the utilization of this technology for both off- and on- chip memory arrays. In this work, it is proposed efficient ways of using STT-RAM as on-chip video memory, in combination of SRAM, to provide energy savings to the reference frames storage.

3. STATE-OF-THE-ART RELATED WORKS

This chapter has the goal of discussing the related works to situate the main contributions of this Thesis inside the different state-of-the-art research fields. Initially, related works for general-purpose processing are analyzed (**Section 3.1**). In this first part, scratchpad memories and hybrid memory design challenges and opportunities are discussed. Then, initiatives for parallelization of video coding are evaluated (**Section 3.2**), as well as the gap of memory-optimized solutions to support energy-efficient parallel HEVC implementations. In sequence, works that exploited application-specific video coding properties for memory optimization are discussed (**Section 3.3**). After that, energy-efficient video memory architectures for non-parallelized and Tiles-parallelized video coding are discussed in several aspects (**Section 3.4**). At this part, detailed comparison between the proposed ideas is performed. At the end, a description of preliminary works that bases Hy-SVM architecture is presented (**Section 3.5**).

3.1 General-Purpose Energy-Efficient Memory Works

Energy efficiency has been the target of lots of works in general-purpose multi-core systems in the last decades. In this section, two key memory design research fields are analyzed: the adoption of scratchpad memories and the exploration of emerging memory technologies in hybrid design. Before moving to application-specific exploitation of such techniques, brief discussions regarding general-purpose applications are presented.

3.1.1. *Scratchpad Memories*

As opportunity for application-specific applications, scratchpad memories (SPMs) overcome/alleviate the hardware overhead of caches. In SPMs, instead of providing hardware support for mapping data/code between off-chip and on-chip memory, the designer and/or the programmer/compiler are responsible to perform access management. If well designed, SPMs allow energy savings of up to 30% compared to complete cache memories (BANAKAR et al., 2002). SPMs are widely available to be used as high-performance and energy-efficiency on-chip storage option in nowadays processor chips (IBM RESEARCH, 2013; TEXAS INSTRUMENTS, 2017). Furthermore, recent advances on SPMs design and management techniques allowed its usage on nowadays system-on-a-chips and graphic processing units (ALVAREZ et al., 2015; HANSEN et al., 2017; MONAZZAH; FARBEH; MIREMADI, 2017; VILLEGAS et al., 2017). In this work, *we utilize SPMs as opportunity for designing application-specific on-chip video memories, enabling energy savings by exploiting the knowledge from the HEVC encoding*.

Power efficient management of these scratchpad memories is of key importance. External memory pressure and on-chip scratchpad memory management for high-performance manycore systems have been explored in (JEONG et al., 2012; PILLA et al., 2012). However, these works do not account for the application-specific properties, thus may not be efficiently applied to compose on-chip video memories.

3.1.2. *Hybrid Memory Design*

The hybrid memory design exploiting the development of emerging non-volatile memory technology has been research target during the last years (ABE et al., 2012; CHEN et al., 2012; DONG et al., 2008; JOG et al., 2012; KHAN; SHAFIQUE; HENKEL, 2013; LI et al., 2009; WU et al., 2010). These works provide a solid foundation to enable these emerging technologies feasible to be integrated with CMOS logic circuitry of nowadays embedded manycore processors. However, these works may not efficiently support the video coding high memory demand, since they did not take into account application-specific properties.

STT-RAM stands out as one of the most promising emerging memory technology, being target of industry research interests. Some commercial and academic initiatives highlight STT-RAM feasibility to be utilized: in standalone data storage systems (LAPEDUS, 2017; MERTENS, RON, 2018), as external memory alternative (JIN; SHIHAB; JUNG, 2014; KÜLTÜRSAY et al., 2013), as embedded storage option for specific applications (like multimedia, automotive and display panel) (MERTENS, 2017), as well as integrated into multi-core processors to implement last-level caches (AHN; YOO; CHOI, 2016; IMANI; PATIL; ROSING, 2016; KHOSHAVI et al., 2016).

One critical issue regarding non-volatile memories (like STT-RAM) is their limited endurance, which may lead to wear-our errors occurrence. It not well managed, such data loss may result, in the worst case, on application unexpected failures. Even with recent advances on fabrication process of non-volatile memories, state-of-the-art works have been proposed different management schemes aiming on alleviating STT-RAM cells from frequent bit toggles during write accesses (ARJOMAND et al., 2017; KIM; KIM; LEE, 2017; MIN et al., 2017; REED et al., 2017; YAZDANSHENAS et al., 2014); thus resulting on extended lifetime. The improved endurance achieved by these techniques are limited due to the requirements of considering general-purposed memory access behavior. In this work, *application-specific properties could be leveraged as opportunity to further increase STT-RAM cells lifetime*.

3.2 Tiles-Parallelized HEVC Works

3.2.1. *HEVC Hardware and Software Implementations*

HEVC performance and energy constraints have been addressed in recent state-of-the-art implementations. For the decoding part, efficient software (STROGENE.COM, 2018) and hardware (CHIANG et al., 2016; LIU et al., 2015; ZHOU et al., 2017) implementations are able to reach enough processing rates without strongly affecting the overall energy consumption.

However, the encoder reunites the most complex coding space exploration tools, imposing much more challenges for designing energy efficient implementations. In this context, parallelization features have been exploited to allow software and hardware for HEVC encoders (CHEN et al., 2016; CHO et al., 2015). In the same way that performance is improved, parallelization of the video encoding highly aggravates the energy consumption of HEVC encoders, especially for the off- and on-chip video memories. Thus, *the challenge is to increase the energy efficiency of the video memory infrastructure for parallelized HEVC encoders requirements*.

3.2.2. *HEVC Parallelization Strategies*

Several works have exploited HEVC high-level parallelization features to achieve performance speedup, especially for encoder implementations. In this context, different strategies were developed to properly define the best Tiles partitioning of video frames for parallel processing (BLUMENBERG et al., 2013; CHI et al., 2012; JIN; DAI, 2016; KHAN; SHAFIQUE; HENKEL, 2014; SHAFIQUE; KHAN; HENKEL, 2014).

In *Blumenberg et al.* (2013), variance maps from the raw video are used as hint to determine the best Tiles boundary locations. The scheme adopts a 2-step algorithm to group the higher correlated samples into the same Tile. The Tiles partitioning can be updated for each encoded frame. The main concern is related to determine the breaking points of spatial references that will result in lower coding efficiency losses.

Chi et al. (2012) provides efficient implementations of HEVC tools for parallel processing (Tiles and WPP). Afterwards, it was proposed a novel parallelization tool, called overlapped wavefront approach, which achieves higher performance and coding efficiency than Tiles and WPP schemes.

Khan (2014) relies estimating the total workload in an Intra Only HEVC encoder to determine the best Tiles partitioning. To adapt the Tiles-specific workload to the available processing capabilities, a scheme that manages the complexity knobs of the encoder were proposed. Further, to maximize the power efficiency, the operation frequency of each processing core is adapted depending upon the workload required for the corresponding Tile.

Shafique (2014) targets on minimizing the total power consumption by adapting the Tiles partitioning to activate an appropriate number of cores according to video-related processing demands. The best way of partitioning the video frame is the one that maximizes the coding efficiency and, at the same time, fulfills the throughput scenarios.

As can be noted by the above discussed schemes, they typically take into account the workload of each HEVC processing unit to define the best Tiles organization. Indeed, a well-balanced workload is an important factor when focusing on maximizing performance and minimizing power dissipation of a HEVC encoder. However, crucial issues regarding the storage and transmission of HEVC data in off-chip and on-chip video memories are not considered in these works. In a parallel-processing system, the memory infrastructure is highly required, leading to energy wasting if not properly designed and managed. Hence, *it can be observed a gap of energy-efficient video memories to support parallel HEVC hardware- and software-based implementations*.

Only *Jin (2016)* has concerns regarding the impacts of Tiles-based HEVC parallelization in the video memories. *Jin (2016)* proposed a content-adaptive Tiles partitioning to improve the HEVC compression efficiency under on-chip memory constraints. Based on maximum Tiles dimensions due to fixed on-chip memory size for the CTUs line buffer, the scheme employs a local competition optimization-based rectangular clustering scheme to partition the frames into a required number of Tiles adapting to video content variations. As shortcoming of this work, the proposed scheme focus on optimizing the memory design exploiting only intra-Tile data reuse, not considering the potential of inter-Tiles data reuse. Further, video memories characteristics of very high read intensity (compared to write operations) are not exploited in the work. Therefore, to address this gap *our work focus on exploiting the inter-Tiles data reuse, as well as intrinsic video memories characteristics (like high read access intensity).*

Furthermore, another important aspect is the external memory pressure that may be unbalanced due to different video properties at each Tile region within the video frame (as motivated in Section 4.2). If not well managed, it might lead to low/high power peak fluctuations and off-chip communication channel wasting. To address this issue, *we propose a memory pressure management, which integrates Hy-SVM, to properly balance the instant memory requirements in a Tiles-parallelized HEVC encoder*.

3.3 Application-Specific Memory Optimization for Video Coding

This section presents application-specific memory optimization schemes and architectures for video coding. As a widely used solution, traditional data-reuse schemes are presented and compared to the adopted strategy by Hy-SVM. Moreover, state-of-the-art reference fame compressing techniques for H.264/AVC and HEVC are presented and discussed, regarding their main shortcomings for parallel video coding. After, ME hardware architectures for HEVC-specific coding structures are presented. The main goal is to analyze the adopted on-chip memory infrastructure. At the end, video memories architectures for decoders are analyzed.

3.3.1. *Data-Reuse Schemes*

During the past decade, multiple works developed data-reuse schemes for the reference frames samples (CHEN et al., 2006; GRELLERT et al., 2011; TUAN; CHANG; JEN, 2002). In such works, the regularity of fetching the entire search window samples was exploited, even when fast search algorithms are adopted, which may not access the entire search window.

Tuan (2002) and Chen (2006) were the first works to exploit the data locality between neighboring search window samples in video coding applications. Level-C strategy (Figure 3.1a) relies on keep stored in on-chip video memory one entire search window. In doing so, due to the overlap between neighboring search windows (dark gray area in Figure 3.1a), just the remaining samples are required to be fetched from external memory. In Level-C, considering the traditional raster scan encoding order, a complete data reuse exploitation can be performed for one entire row of search windows. In a frame width level, Level-D (Figure 3.1b) scheme stores the entire row of search window in the on-chip video memory. Thus, besides exploiting the data reuse in the entire row of search windows (as Level-C), Level-D allows data reuse when the first block of the next row is processed (as depicted in Figure 3.1b). Moreover, several other schemes in different levels were also proposed in these works (Level A, B and C+). In the evaluations of this Thesis, Level-C and Level-D were adopted for comparison purposes.

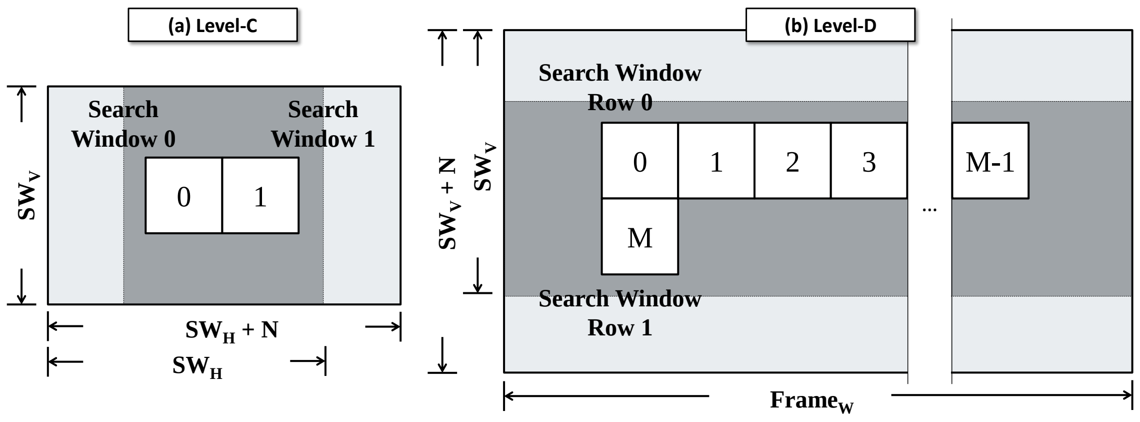


Figure 3.1: (a) Level-C and (b) Level-D schemes, which exploit intra-Tile data reuse in search window and frame width levels, respectively.

Sampaio (2013a) presented a data-reuse scheme in a different perspective (compared to Level-C and -D), when targeting multiview video coding scenarios. The proposed strategy, called Reference-Centered Data Reuse (RCDR) changes the ME processing order to reduce the number of times that one reference frame is fetched. In multiview video coding, the reference frame communication is aggravated due to the inter-view prediction, which exploits data redundancy between different cameras (views) for the multiview video. Due to this out-of-order processing, partial results are generated, which must have to be stored until can be discarded.

Table 3.1: Comparison of Data-Reuse Schemes

Work	**Target**	**Data-Reuse Level**	**Inter-Tiles Data Reuse?**	**Coding Order**
Tuan (2002) – Level-C	MPEG-2	Search window	No	Raster scan
Tuan (2002) – Level-D	H.264/AVC	Reference frame width (within the same Tile)	No	Raster scan
Sampaio (2013a) – RCDR	Multiview H.264/AVC	Search window	No	Reference-centered order
This Work – Hy-SVM	**Parallelized HEVC**	**Multi-level - Search window and (L1) Reference Frame (L2)**	**Yes**	**Balanced-pressure CTU re-scheduling**

Table 3.1 summarizes the above discussed data reuse schemes and compares them to the strategy employed by Hy-SVM architecture. In terms of data-reuse level, Hy-SVM is the first of proposing a multi-level approach, which jointly exploits search window (L1 SPMs) and reference frame (L2 SPMs) levels of data reuse. Moreover, inter-Tiles data reuse is a novelty introduced by the proposed video memory architecture, which increases the overall energy savings by reducing external memory communication. Hy-SVM also adopts a CTU re-scheduled order to balance the memory pressure in according to the video content properties.

3.3.2. *Reference Frame Compression Schemes*

Reference frame compressing strategies are exploited since previous video coding standards, and have the focus of reducing the data bandwidth from on-chip and off-chip memories by compressing the reference frames data (GUO et al., 2016; GUO; ZHOU; GOTO, 2014; LIAN et al., 2016a, 2016b; SAMPAIO et al., 2013b; SILVEIRA et al., 2015; ZHU et al., 2015).

Sampaio et al. (2013b) presented a lossy reference frame compressing scheme that employs a spatial prediction scheme to reduce provide near-zero representation of reference frame blocks, allowing efficient data compressing utilizing Huffman tables. The proposed strategy was developed for H.264/AVC multiview video coding memory requirements.

Silveira et al. (2015) implemented a low-complexity and lossless reference frame compression solution that performs intra-block double differential coding over 64x64 blocks to prepare them for static Huffman coding. Furthermore, hardware implementations of compressor e decompressor parts were designed, indicating power reduction in both on- and off-chip memory parts.

In *Guo (2014)*, a hybrid spatial-domain prediction is proposed, which is enhanced with additional modes to support various image characteristics. After that, efficient residual regrouping based on semi-fixed-length coding improves compression performance. A hardware implementation was designed to implement and evaluate the proposed techniques. Results indicate enough performance to support 3840x2160 HEVC encoding.

Zhu (2015) proposed an architecture that overcomes limitations of implementing frame re-compression techniques in HEVC video codecs. The work also provides easy connection with all video coding implementations.

Lian et al. (2016a) focused on improving the reference frame compression performance of state-of-the-art lossless algorithms, which noticeably degrades the external memory access latency. In doing so, the work developed an adaptive quantization oriented parallel lossless frame memory recompression algorithm. Experimental results demonstrate applicability of the designed hardware for compressor and decompressor parts for UHD videos, when utilized along with data reuse schemes.

In the same direction, *Lian et al. (2016b)* improved the previous work by strongly adapting the compression algorithm to the specifications of DRAM-based external memories. In this context, read and write dynamic, as well as page activation behavior are considered in the proposed scheme. Hence it improves off-chip energy savings compared to related works.

Guo et al. (2016) proposed a lossy reference frame compression algorithm that mostly focuses on minimizing the error propagation, which causes increased quality degradation of reference frames. As results, better and lower fluctuation in PNSR results were verified.

Even though these works can be applied to parallel HEVC encoding, they do not consider parallel memory accesses issues from different processing units, leading to: compromised scalability for increased parallelism, unsupported memory contention, and increased memory access latency. Further, on-chip memory energy is even aggravated since multiple on-chip logic circuitry must be inserted to keep the performance rates when various processing units are used in parallel.

3.3.3. *Motion Estimation Architectures with On-Chip Memory Design for HEVC*

Hardware design for motion estimation has been a key research challenge for recent video encoders. In this context, architectures for ME were designed targeting MPEG-2 and H.264/AVC specifications and requirements (PORTO; AGOSTINI; BAMPI, 2009; ROVATI et al., 2000). However, HEVC increased ME computational complexity by allowing more flexible block partitioning when compared to previous video coding standards (as already discussed in Section 2.2.1). Hence, it poses harder challenges for energy-efficient ME targeting HEVC encoders. To provide on-chip data storage support in such ME architectures, several works integrate video memory design (FAN et al., 2017; JOU; CHANG; CHANG, 2015; PARK et al., 2016; VAYALIL; KONG, 2017).

Fan et al. (2017) developed a hardware-oriented integer ME algorithm and the related hardware implementation. For the reference frames management, the designed architecture implements 2-D data reuse supported by horizontal and vertical reference SRAMs, along with on-chip memory reduction supported by 4x4 block compression.

Jou (2015) presented a joint algorithm and architecture design for ME that reduces the integer ME, complexity by selecting the most probable search directions and steps through statistical analysis. Besides, a novel fractional ME scheme reduces the interpolation filtering operations. These novel schemes contributes by increasing reference samples data reuse. It adopts a cache design as on-chip video memory, optimized by double Z scan indexed addressing to simplify access management.

Vayalil (2017) designed a full-search variable-block-size ME that reduces the memory requirements by following a Morton order for data reading and a sum of absolute differences reuse strategy.

Park et al. (2016) implemented a hardware architecture for ME using a modified reference data access skip (MRDAS) scheme for reducing the minimum memory bandwidth. Along with external memory communication reduction, coding efficiency is negligible degraded by the proposed technique.

None of the above discussed works support parallel execution of ME when multiple Tiles are defined. Parallelization of ME imposes the need of extra logic to implement multiple search engines, as well as requires specialized memory support to provide higher throughput rates. In doing so, the energy efficiency and performance of discussed ME architectures are significantly compromised when used in parallel HEVC encoder implementations.

3.4 Energy-Efficient Video Memory Architectures

The analysis of related energy-efficient video memory architectures is divided in three parts. At first, solutions designed for multiview video coding specific requirements are discussed. Then, video memories developed for HEVC are detailed and compared to the proposed Hy-SVM. At the end, two preliminary hardware design, developed as initial approaches of this PhD works, are detailed.

3.4.1. *H.264/MVC Related Works*

Energy-efficient video memory design and management were already important research focus since multiview video coding (MVC) extension of H.264/AVC standard was released. MVC encoders highly require from memory since inter-view prediction is employed to exploit disparity redundancies between frames from different views (cameras) (VETRO; WIEGAND; SULLIVAN, 2011). In this scenario, energy-efficient video memory architectures were strongly necessary. Several works focused on leveraging MVC memory behaviors and multiview video content properties to increase the energy savings of on- and off-chip parts of reference frame handling (SAMPAIO et al., 2013a; SHAFIQUE et al., 2012; ZATT et al., 2011a, 2011b).

Zatt et al. (2011a) presented a run-time adaptive energy-aware motion and disparity estimation architecture considering multiview video coding extension of H.264/AVC. The memory infrastructure incorporates data prefetching techniques for jointly reducing off- and on-chip memory energy consumption. Search maps from previous blocks processing are used to predict the search behavior for the next blocks. Power gating is adopted to shut down parts of the on-chip memory depending on the prediction. Unlike the already discussed data reuse schemes in Section 3.3.1, the search window is not completely fetched, avoiding unnecessary access of unused samples when fast search algorithms are not adopted.

An on-chip multi-banked video memory for motion and disparity estimation was proposed by *Zatt et al.* (2011b). The memory organization was driven by an extensive analysis of memory usage behavior for several videos. Memory restrictions for each block are derived from inter-frame and inter-view correlations. When low motion regions are processed, adaptive power gating works to reduce the energy supply of less probable unused on-chip memory sectors. Still, reference frame fetching is performed as the ME requires, which saves memory accesses for fast search algorithms, but leads to irregular access pattern in the off-chip memory.

Shafique et al. (2012) introduced an adaptive power management targeting on-chip video memories for multiview video coding. The energy-aware control checks the so called 3D-neighborhood for texture, motion and disparity properties to predict the behavior for the current block encoding. As in the previous discussed works, not the entire search window is fetched, causing irregular off-chip memory accesses.

Sampaio (2013a), besides contributing with the already discussed reference-centered data reuse scheme (see Section 3.3.1), also implemented an on-chip video memory architecture integrated with off-chip memory data organization and on-chip power management. In the off-chip perspective, a regular access pattern and reference frame data organization allowed reduced energy consumption by reducing DRAM page activation overhead. Further, a candidate blocks merging scheme was proposed to provide accurate hints to power gating control.

The H.264/MVC-based memory architectures are not scalable enough to be energy efficient for HEVC encoders due to its novel coding tools and complex video processing flow. By not taking into account the novel coding model of the advanced HEVC, these works are not able to achieve higher levels energy savings, as motivated in the evaluations of Chapter 4. Moreover, in Tile parallelized HEVC encoders, multiple processing units request data at the same time from the shared memory system. Thus, several other factors need to be taken into account, e.g., memory contention and memory access scheduling schemes. Still, inter-Tiles data reuse was not exploited by any of related works, which is required for energy-efficient parallel HEVC encoders.

3.4.2. *HEVC Related Works*

Khan (2013) proposed the first on-chip video memory architecture targeting HEVC encoders, called AMBER. It is based on a hybrid memory design utilizing STT-RAM technology. Additionally, it uses SRAM to implement FIFO buffers to hide the high write latency of STT-RAM cells. AMBER exploits the low leakage features of STT-RAM to store the entire DPB (i.e. all reference frames required to completely process one GOP of a video) in the on-chip video memory. Moreover, the search window size and the memory access pattern is leveraged as run-time parameters to apply adaptive power management, increasing the energy efficiency of AMBER.

Song (2015) implemented an on-chip memory architecture (called HVM) which combines SPMs and caches to achieve energy efficient data storage. A run-time prediction algorithm is proposed to effectively identify the most-frequent accessed memory regions in the search windows for processing individual CTUs. Depending on their intra- and inter-core reused, private or shared SPMs are accessed. An adaptive power gating scheme power offs SPM sectors with expired search windows, thus reducing static energy consumption.

Table 3.2 resumes the characteristics of presented energy-efficient video memory architectures for HEVC, relating their proposed schemes with Hy-SVM implemented strategies. AMBER has the limitation of not considering parallel video processing in its on-chip memory design and management units, which is inevitable to achieve high throughput. In terms of adopted on-chip memory organization. The three works utilize SPMs to simplify the access management circuitry. Besides, AMBER implements FIFOs as first level of memory to reduce write access latency. However, the implemented FIFOs are not effectively part of the storage system that may provide a high potential of energy-/performance-efficient design. Furthermore, AMBER stores all reference frames in the on-chip STT-RAM memories that incur high frequent-write, thus performing inefficient management under such scenarios and may only be feasible for a certain set of video resolutions. Another gap that is not addressed by AMBER is related to STT-RAM lifetime improvement, since all on-chip memory write accesses are performed in the STT-RAM part.

Table 3.2: Comparison of Energy-Efficient Video Memory Architectures Targeting HEVC

	Khan (2013) – AMBER	***Song (2015) – HVM***	***This Work – Hy-SVM***
Tiles-Parallelized HEVC Support?	No	Yes	Yes
On-Chip Memory Organization	FIFO and SPM	SPM and Cache	SPM
On-Chip Storage	Current CTU and entire Decoding Picture Buffer	Current search windows (distributed along private and shared SPMs and Caches)	Current search windows (private L1 SPMs) and reference frame (private and shared L2 SPMs)
Inter-Tiles Data Reuse	No	Yes	Yes
Memory Technologies	SRAM and STT-RAM	SRAM	SRAM and STT-RAM
Application-Specific Management Schemes	-	Load search window prediction algorithm	Overlap prediction, memory pressure management and lifetime-aware data management
On-Chip Management Units	Power-gating control (for memory cells) and clock-gating control (for ME engine)	Video memory management unit and power-gating control	Distributed MAMUs and APMUs

In another perspective, HVM integrates caches to store portions of the search window. SPMs are used to support a prefetching unit (called load window prediction algorithm). However, both private and shared caches/SPMs are logically organized in the same memory level, which means that the incoming memory access is directed either to private or shared memory array. In another vein, Hy-SVM defines two level of on-chip memories, which reduces the required off-chip memory communication. Hy-SVM adopts a light-weight caches to the second level to alleviate high-bit-toggling write operations to SPMs (implemented as STT-RAM), thus leading to improved endurance.

3.5 Preliminary Works from This Thesis

As first initiatives from this Thesis, two on-chip video memory architectures were proposed to initially exploit Tiles-parallelized opportunities to enable energy-efficient data management to HEVC encoders. The proposed Hy-SVM architecture is based on research opportunities and extended contributions from works.

Sampaio et al. (2014a) was pioneer on exploiting inter-Tiles data reuse to reduce external memory communication, thus improving the off-chip energy savings. The proposed architecture was called Distributed Scratchpad Video Memory Architecture (dSVM), being composed of private and shared memory arrays to enable support for intra-Tile and inter-Tiles data reuse, respectively. The adopted on-chip memory models are based on SPMs to simplify control circuitry. A policy for energy-efficient memory management was proposed, which was based on the identification of the overlap characteristics, which is estimated by a simple, but efficient, overlap prediction engine. An adaptive power management scheme was also proposed to shutdown memory cells outside the predicted overlap, increasing the on-chip static energy savings.

Sampaio et al. (2014c) improves dSVM achievements by developing a dedicated hybrid video memory architecture (called enHyV) focusing on parallel HEVC. enHyV combines SRAM and STT-RAM using private and shared SPMs. A design space exploration was performed to find the best optimization point to define the size of SRAM and STT-RAM memory arrays. Although enHyV implemented shared SPMs to support inter-Tiles data reuse, the focus was to demonstrate the contributions of STT-RAM technology to implement on-chip video memories. In doing so, just a simple overlap management unit integrates enHyV management layer. To deal with STT-RAM write inefficiency and endurance issues, it was proposed a data management scheme that reduces bit-toggling inefficiency during write accesses.

As limitation of dSVM, although inter-Tiles data reuse is exploited, its potential is not well exploited in this work, since adaptive management is not performed considering variable video content properties inside the same frame. Depending on the video properties (like low/high motion), energy may be wasted by not properly manage the shared video memories. Furthermore, to support on-chip overlap storage for reduced external memory communication, the additional SRAM to improve the data reuse brings extra static energy consumption. Therefore, merely using SRAM it becomes unfeasible when using a large number of processing units.

By exploiting STT-RAM advantages of low static energy consumption, enHyV could improve dSVM capability by keeping stored one entire reference frame. Combined shared and private SPMs allows joint intra-Tile and inter-Tiles data reuse. The shortcomings of enHyV are mainly related to the lack of efficient memory access and power management techniques to deal with video content dependent overlap formation characteristics.

4. HEVC MEMORY DYNAMICS AND OVERVIEW OF PROPOSED MEMORY ARCHITECTURE

This chapter presents preliminary analysis of memory dynamics of different HEVC encoding scenarios. In the first part, the goal is to characterize and motivate the main problem related to the memory infrastructure of HEVC encoders: the bottleneck caused by the intense reference frames transmission and the need of large on-chip video memories (**Sections 4.1 and 4.2**). The overlap formation, which represents a key concept for the proposed Hy-SVM architecture design and management strategies, is presented in **Section 4.3**. At the end, to link with the insights from the presented motivational analysis, the main contributions of this work are briefly introduced (**Section 4.4**) as start point for their technical description along the next chapters.

4.1 HEVC Memory Profiling

The external memory transmission to fetch the reference frames, as well as the on-chip storage to keep the data available for processing, are the main responsible for the high performance and energy restrictions in a HEVC encoder. These constraints are aggravated when real-time processing is required for very high resolutions (like 1080p, 2K and 4K) and high frame rates (like 60 fps and 120 fps). Furthermore, at the same time that parallelism can be exploited to meet these performance targets, it also leads to even higher penalties in the energy perspective, mainly when it is considered for the memory infrastructure.

Figure 4.1 presents some memory access evaluations of a HEVC encoder application: HEVC test model 11.0, using the TZ Search ME algorithm. In the overall HEVC encoder perspective (Figure 4.1a), note that the inter prediction is responsible for up to 80% of the memory accesses for the encoded videos. As already mentioned, the inter prediction must evaluate all CUs into the CTU structure. Besides, the ME search algorithms intensively access the memory to scan the reference frames (typically stored off-chip). Specifically in the inter prediction, only the reference frame fetching occupies the memory transmission channel in 45%. It is important to notice that TZ Search is a very fast and efficient ME algorithm, which reduces the memory communication in 23x, when compared to the exhaustive search (Full Search) (PURNACHAND; ALVES; NAVARRO, 2012). Thus, even when efficient ME algorithms are chosen, the memory requirements stills significantly affecting the encoding system.

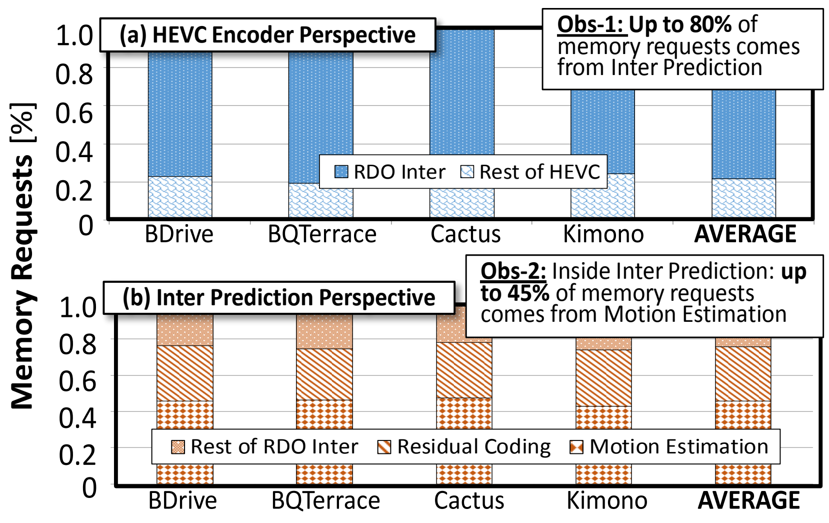


Figure 4.1: Memory requirements analysis for HEVC encoding.

4.2 Tiles-Accumulated Memory Pressure Evaluation

In this work, memory pressure is defined as the memory access requirement caused by a CTU processing during a specific time. When considering multiple processing units, the memory pressure may be (1) Tile-specific, or (2) accumulated (sum of all Tile-specific pressures). Typically, the motion estimation is performed in the traditional raster scan order (i.e., from top-left to bottom-right corner in row-by-row order). However, this may lead to unbalanced external memory pressure, as depicted in the 4-Tile example of Figure 4.2a. The maximum and minimum memory pressure peaks can be seen in Figure 4.2b. There are significant memory access variations compared to the average access case (that typically does not happen). This unbalanced memory pressure leads to high power peak dissipations and high instant memory bandwidth requirements, which may surpass the maximum availability constraints. Moreover, such unbalancing also leads to inefficient memory power management due to (1) fluctuations in the sleep durations, (2) frequent P_{ON}-P_{OFF} switching, and (3) memory usage prediction errors due to sudden access variations.

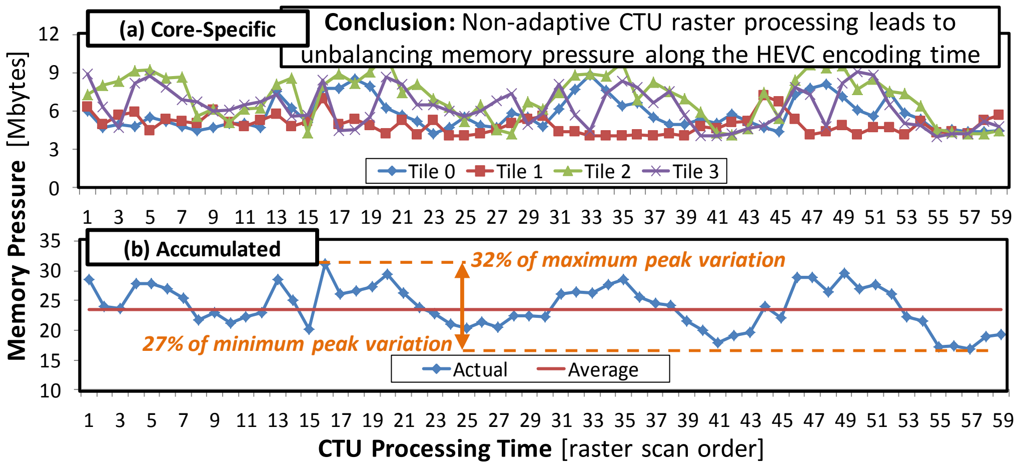


Figure 4.2: Memory pressure for (a) each processing unit; and (b) accumulated and average cases for *BasketballDrive*.

Therefore, *the key is to leverage application specific-properties to adapt and re-schedule the CTU processing in order to achieve the best possible memory pressure balancing.*

4.3 Inter-Tiles Data Reuse Evaluation

4.3.1. *Inter-Tiles Redundant Memory Access Evaluation*

When multiple Tiles are supported by a HEVC encoder, each processing unit is responsible to encode the CTUs of a specific Tile. Thus, the ME of each processing unit intensively searches in the reference frame to capture the motion properties. For the CTUs located near the Tiles boundaries, the ME search algorithms (depending on the motion direction) may access reference regions located across the Tiles limits in the reference frames. This leads to redundant memory access between adjacent Tiles processing. This data redundancy tends to grow for an increased number of Tiles (assuming 1 Tile per processing unit). The data redundant access trend is plotted for growing number of Tiles in the Figure 4.3. In the worst case, the redundant accesses reach 43% in a 16-Tile HEVC encoder. As larger is the memory accesses redundancy, more processing units must concurrently access the same reference data from the external memory without any data reuse. Therefore, *it may be beneficial to design dedicated on-chip memories for the data redundant regions to avoid external memory retransmission of the Tiles shared reference data, saving off-chip memory energy*.

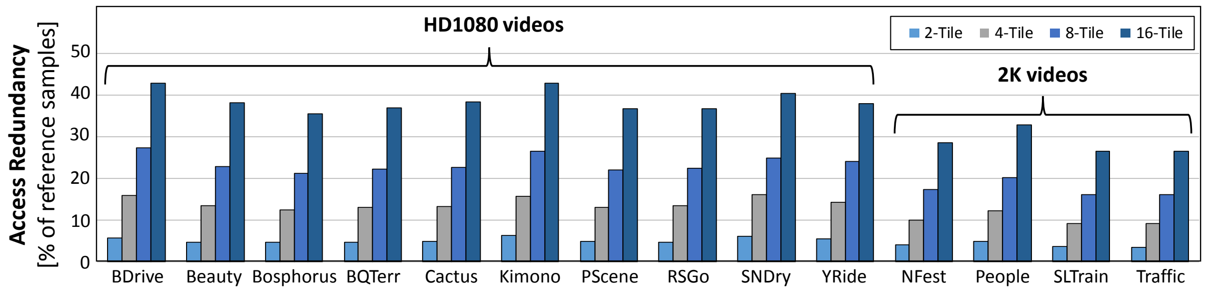


Figure 4.3: Redundant memory access between different processing units (assuming 1 Tile per unit) growing trend for increased parallelism.

It can also be noted that the inter-Tiles memory access redundancy may vary depending on the input video. For the HD1080 tested sequences, low-motion videos, like *BQTerrace* (4.7%-37%) and *Bosphorus* (4.8%-35%) tend to have less inter-Tiles access redundancy when compared to more complex sequences, like *BasketballDrive* (5.8%-43%) and *Kimono* (6.4%-43%). High-motion videos require more search steps to ME in order to find the most similar candidate block in the delimited search window within the reference frames. Thus, analyzing the PUs of CTUs near to the Tiles boundaries, it can be noted that ME may require more memory accesses across this barrier, leading to increased access redundancy between the parallel HEVC processing units. Therefore, besides designating on-chip video memories for shared reference data, *there is a strong need of application-driven design methodology for efficient SPMs sizing, as well as dynamic adaptation of memory management to be adaptive to different video content properties.*

The overlap concept, which is adopted in this work to exploit the characteristics of the formation of the inter-Tiles memory access redundancy inside the reference frames, is presented as follows.

4.3.2. *Overlap Concept*

The strategies for inter-Tiles data reuse are based on the *overlap concept*. The overlap is composed by reference frame samples that are accessed by two or more processing units during the parallel Tiles processing. The samples near to the Tile boundaries in the reference frame must be fetched/stored by multiple processing units, leading to external memory contention, redundant memory accesses and extra on-chip storage (causing energy wastage). An example in Figure 4.4a depicts the overlapping accesses performed for more than one processing unit (gray and black regions). When observing the memory access maps (Figure 4.4b), it is possible to notice that each processing unit will direct its searches to regions of reference frame according to the correspondent Tile position. When these access maps are merged and the intersection is analyzed (Figure 4.4c), the formed overlapping region can be observed.

Figure 4.6 presents the reference frame access maps of the overlap formations considering the *BasketballDrive* sequence when encoded with uniforms partitioning with 2, 4, 8 and 16 Tiles. Note that the overlap constitution is observed around the Tiles boundaries, being increasingly representative when more parallelism is adopted. The access maps confirm the same growing trend of previous analysis illustrated in Figure 4.3. Additionally, different shapes and access intensities can be observed along the different Tiles boundaries. In the example of the initial part of *BasketballDrive*, which concentrates the high motion CTUs in the left and bottom parts of the captured scene, the shape of the resultant overlap is itself heterogeneous. We can observe this variability (1) within the formation around one specific Tile boundary, as well as (2) between the formations of different Tiles boundaries. Therefore, to handle with this variable content-driven behavior, *specialized management for each Tiles-boundary overlap* *along with dynamic adaptation of variable overlap shapes is strongly required to achieve increase energy savings for both on- and off-chip memory parts.*

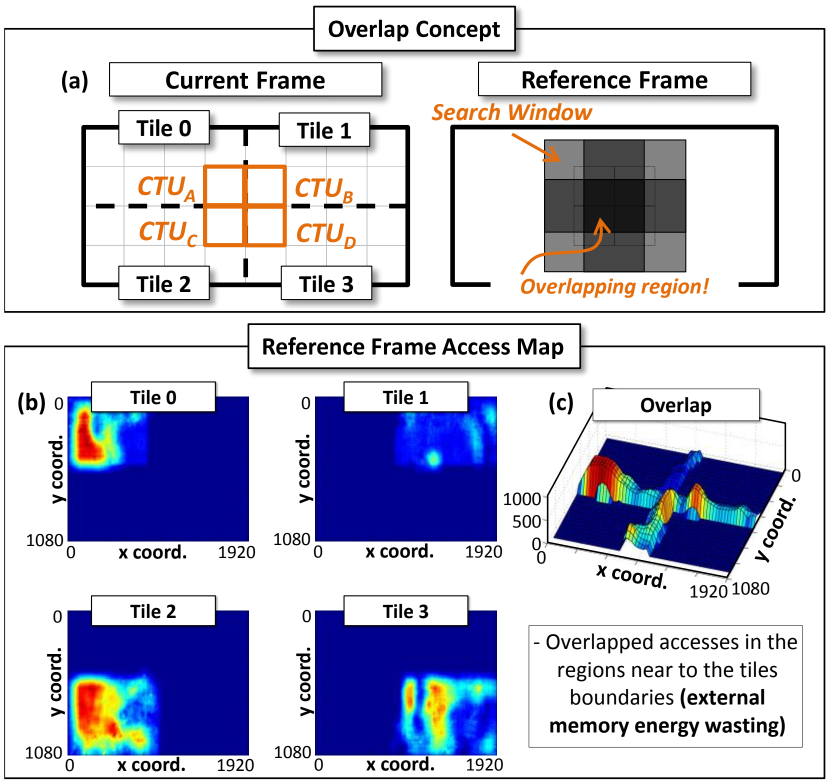


Figure 4.4: Overlap concept: (a) Example of Tile partitioning and of the overlap formation; reference frame access maps (b) for each Tile (specific for each processing unit) and (c) for the formed overlap.

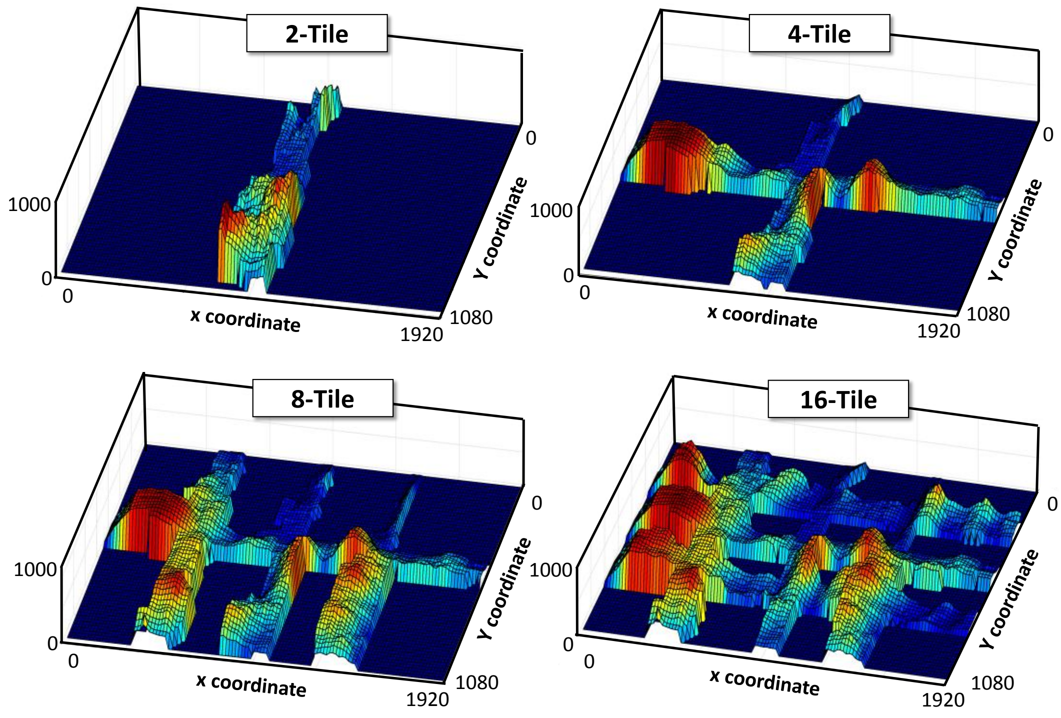


Figure 4.5: Reference frame access maps for the overlap formation in uniforms partitioning with 2, 4, 8 and 16 Tiles.

4.3.3. *Overlap Formation Parameters*

Figure 4.6 depicts an example of the overlap formation, as well as its main parameters. Let PU_A and PU_B be adjacent prediction units of CUs belonging to CTUs of different Tiles. Thus, they are encoded by different processing units that concurrently access the external memory for reference frame fetching. Typically, to determine the best ME start point, the most recent search algorithms (as TZ Search) for ME use past motion vectors as predictors for this purpose. Therefore, the motion properties of the Tiles boundaries neighborhood strongly affect the memory access behavior. In the example of Figure 4.6a, the predictors exploitation lead to a ME starting point at *PredA* and *PredB* for PU_A and PU_B, respectively. The ME search is typically limited by a maximum range, forming a squared search window surrounding the starting point. Inside this delimited area, the algorithm will search for the best match for the current PU. Still in the example, Figure 4.6b and Figure 4.6c depict the search window formation and the actual accessed regions during the ME execution for PU_A and PU_B, respectively. Note that the search pattern of TZ Search was used as case study. As result of the access merging for both PUs processing in the reference frame, the redundant memory access of these two ME searches are highlighted in Figure 4.6d (dark gray region). In this intersection, the reference frame samples are required for more than one processing unit, leading to redundant accesses (if not properly managed). This redundancy is called overlap in this work.

Besides the dynamic ME starting point, the video properties (motion field) near the Tiles boundaries may lead to memory access in different directions inside the search window. High-motion regions will cause a more distant ME starting point and will take the ME to longer searches. In this case, the overlap tend to have higher thickness and variable displacement (when consider the Tiles boundaries as the central position for overlap positioning). To properly subsidize the proposed overlap prediction scheme, detailed overlap characteristics exploitation is performed in Section 6.1.1.

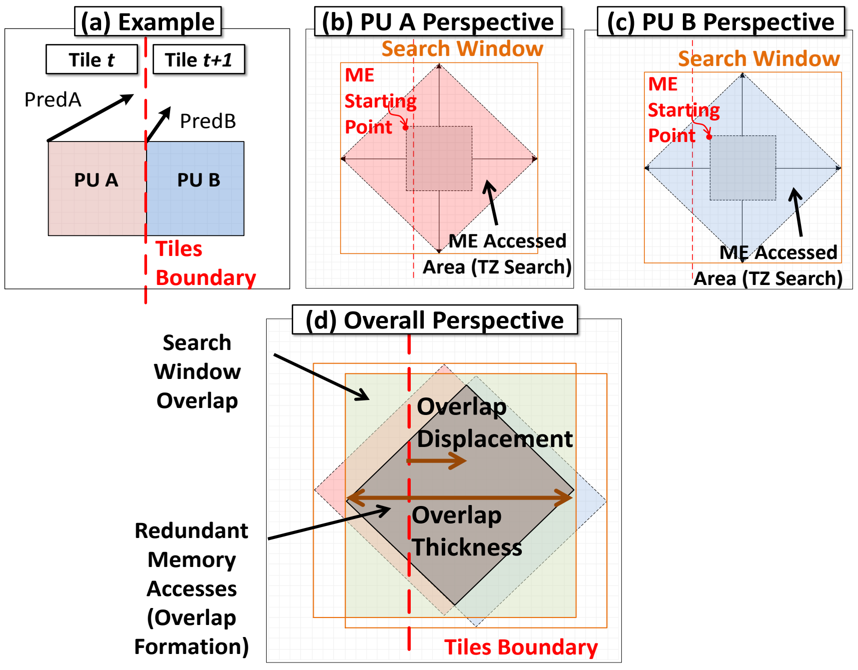


Figure 4.6: Overlap formation and its involved parameters.

In summary, the main involved parameters in the overlap formation are: (1) the maximum search delimitation (search window) and (2) the motion field of frame regions near the Tiles boundaries. The search window is typically a fixed parameters and known at design time. In contrast, the motion field is a video content property that should be analyzed at run time.

4.4 Overview of the Proposed Energy-Efficient Hybrid Scratchpad Video Memory Architecture and Its Run-Time Management Layer

This section introduces the technical content related to the proposed energy-efficient hybrid scratchpad video memory architecture and its run-rime management schemes. Figure 4.7 depicts an overall block diagram that introduces a parallel HEVC encoding system along with all contributions from this work. The system is based on multiple Tiles partitioning to enable parallel HEVC encoding: each Tile is assigned to a specific processing unit. As we focus on energy-efficient memory support, there is not any specific assumption for the HEVC parallel implementations, which may be designed as a multi-/many-core software or as ASIC-based hardware accelerators. The external main memory provides storage to all required data of HEVC encoding, like original and reference frames, instructions, HEVC coding structures, temporal variables, etc. In special, the entire DPB (e.g. all reference frames that are required to encode a video GOP) must be stored in the main memory.

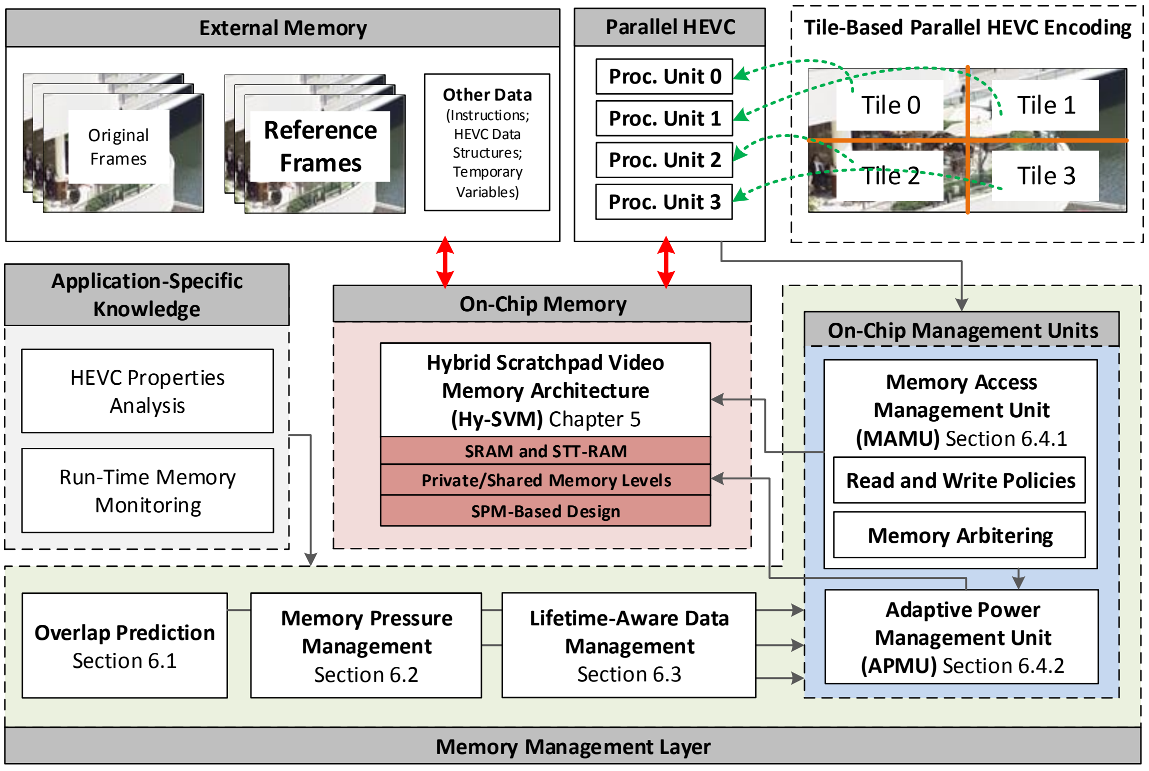

External Memory
Original Frames
Reference Frames
Other Data (Instructions; HEVC Data Structures; Temporary Variables)
Parallel HEVC
Proc. Unit 0
Proc. Unit 1
Proc. Unit 2
Proc. Unit 3
Tile-Based Parallel HEVC Encoding
Tile 0
Tile 1
Tile 2
Tile 3
Application-Specific Knowledge
HEVC Properties Analysis
Run-Time Memory Monitoring
On-Chip Memory
Hybrid Scratchpad Video Memory Architecture (Hy-SVM) Chapter 5
SRAM and STT-RAM
Private/Shared Memory Levels
SPM-Based Design
On-Chip Management Units
Memory Access Management Unit (MAMU) Section 6.4.1
Read and Write Policies
Memory Arbitering
Adaptive Power Management Unit (APMU) Section 6.4.2
Overlap Prediction Section 6.1
Memory Pressure Management Section 6.2
Lifetime-Aware Data Management Section 6.3
Memory Management Layer

Figure 4.7: Block diagram of the proposed memory architectures, divided according to on-chip memory design and memory management layer perspectives.

To allow energy-efficient on-chip storage of the reference frame samples, an on-chip hybrid scratchpad video memory (Hy-SVM) was designed, which employs private and shared hybrid memories to exploit combined intra-Tile and inter-Tiles data reuse. In the memory management layer, run-time adaptive schemes were developed relying on application-specific knowledge: (1) overlap prediction, (2) memory pressure management, and (3) lifetime-aware data management. Application-specific knowledge was exploited by: (1) inheriting HEVC properties and (2) performing run-time monitoring of memory accesses. Such information is used to properly design the on-chip video memories, as well as being utilized as input parameters for the schemes inside the memory management layer. As on-chip management units to control the data dynamics of Hy-SVM, distributed memory access management units (MAMUs) and adaptive power management units (APMUs) are implemented. MAMUs and

APMUs receive information from the run-time management schemes to improve the energy savings of Hy-SVM.

As follows, **Chapter 5** technically describes the hardware design of Hy-SVM. After that, **Chapter 6** presents the specific details of the energy-efficient memory management layer.

5. ON-CHIP HYBRID SCRATCHPAD VIDEO MEMORY ARCHITECTURE

This chapter introduces the on-chip hybrid scratchpad video memory architecture organization. At first, an overview of architecture design is presented (**Section 5.1**), highlighting hardware-specific details. Then, the on- and off-chip memory models adopted by the proposed architecture are descripted (**Section 5.2**). After, offline statistical evaluations of overlap formations and bit-toggling activities are performed to extract design-time parameters for designing the multiple level of SPMs (**Section 5.3**). The developed design methodology, which provides proper parameters to implement the on-chip SPMs, aiming on minimizing the energy consumption while guaranteeing the required reference data storage (**Section 5.4**).

5.1 Overview of Hy-SVM Architecture

Figure 5.1 depicts our hybrid scratchpad video memory architecture (Hy-SVM) and its energy-efficient management layer for parallel HEVC encoding. Each Tile is assigned to a specific processing unit. The proposed memory organization increases the energy efficiency of reference frames management (off-chip fetching and on-chip storage). The coarser lines in Figure 5.1 represent data connections, while finer lines illustrate the control flow between the modules.

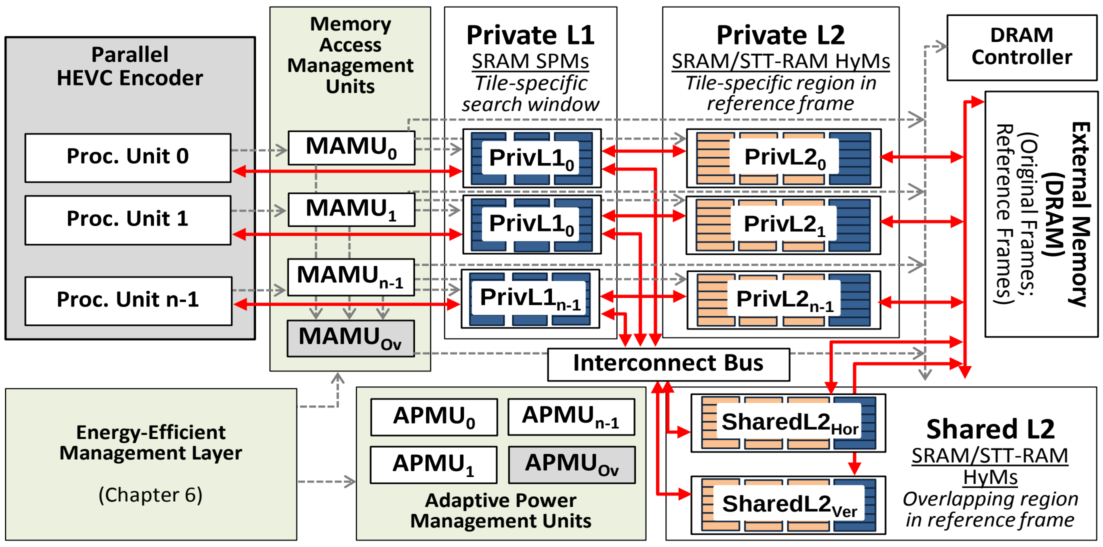


Figure 5.1: Block diagram of our hybrid scratchpad video memory architecture.

Our Hy-SVM architecture is organized as two levels of on-chip memory arrays:

L1 SPMs Level: N_{Tiles} private SPMs[7] (PrivL1) that store the search window samples for a specific processing unit, allowing intra-Tile data reuse between each CU processing. At this level, the SPMs are implemented as SRAM arrays, providing equally high performance and energy efficiency for read and write operations. Since PrivL1 SPMs represent smaller memory cells arrays, SRAM static energy consumption does not significantly affect the overall energy efficiency.

L2 SPMs Level: N_{Tiles} private (PrivL2) and $N_{TilesBoundaries}$ shared (SharedL2) hybrid memories (HyMs) that together can store one complete reference frame, providing combined intra- and inter-Tiles data reuse. Each L2 level HyM is designed as a combination of a STT-RAM SPM, exploiting STT-RAM high density and low static power features to implement large L2 data arrays; as well as a smaller portion of SRAM SPM, that will support high bit-

[7] Let N_{Tiles} be the number of Tiles and $N_{TilesBoundaries}$ be the number of Tiles boundaries.

toggling write activities to overcome STT-RAM write inefficiency. The PrivL2 stores the Tiles-specific region of the reference frame (accessed privately by the corresponding processing unit). Each HyM of PrivL2 has a direct data connection to the corresponding PrivL1 SPM. The SharedL2 HyMs are connected to the PrivL1 SPMs by an interconnect bus and it is responsible for the overlapping regions storage.

Along with SharedL2 HyMs, inter-Tiles data reuse is managed by a run-time overlap prediction that accurately estimates the redundant memory access behavior for the next ME. This prediction step is based on already monitored overlap formations from previous frames encoding. This knowledge is then forwarded to on-chip memory management hardware modules: (a) memory access management units (MAMUs) and (b) adaptive power management units (APMUs). They are responsible to effectively manage the on-chip memories by implementing a read/write policy, as well as proper power gating control over memory sectors. The goal is to achieve the best possible energy efficiency depending on the video content properties. Details regarding the energy-efficient memory management layer are presented in Chapter 6.

As follows, the adopted on- and off-chip memory models are described. The proposed Hy-SVM design methodology is strongly based on these defined parameters.

5.2 On- and Off-Chip Memory Models

Figure 5.2 depicts the adopted on- and off-chip memory models and the defined notations for the main involved parameters for the design part.

The external memory is composed of several banks. Each bank is a row-column matrix, where the number of rows represents the addressing space and the number of columns is directly related to the page size (P_{Size}). Each row-column intersection stores a memory word of size W_{Size}. Each memory access will initially cause a page activation, to pass the activated data to the page buffer. If consecutive accesses to a memory word are located in the same page, the memory controller needs just to address a specific column of the page buffer (called *burst read/write operations*). If other memory page is addressed, the current active page is precharged and a new page is activated. The external memory organization adopted in this work follows the LPDDR2 architecture. Further details regarding the operation flow of the selected off-chip memory model can be found at (MICRON TECHNOLOGY INC., 2001)

Every data transmission from/to memory is based on a fixed *basic access unit* (BU), which corresponds to a $BU_{Dim}*BU_{Dim}$ picture block of the reference frame (see Figure 5.2a). The samples of a BU are organized in a serialized way, so that all rows of one entire BU can be stored in the same memory page (see Figure 5.2b). Since the consecutive accesses to the same memory page lead to less page-activation energy overhead, improved energy efficiency can be achieved. Still, depending on the adopted BU_{Size} and P_{Size}, adjacent BUs of the reference frame can be organized in the same page. Thus, when external memory communication is required, then *several BUs are accessed in one burst operation* to increase the energy efficiency.

As on-chip SPM design, a multi-bank memory organization is adopted (see Figure 5.2c). Each SPM is composed of *NB* memory banks. To facilitate parallel access, each row of a BU is stored in a specific SPM bank. Hence, one line of a memory bank can store *LS* bits, equals to the size of one BU row (BU_{Dim} * NB_{Sample}[8]). The exception is the first SPM bank, which additionally stores control information for memory access management (explained in Section 5.4). A $Bank_i$ is composed of *NL* lines, grouped into *NS* memory sectors of *SS* bytes. The number

[8] In this work, we consider video sequences represented with 8-bit samples.

of BUs per sector ($N_{BUPerSector}$), which corresponds to the number of memory lines per sector ($N_{LinesPerSector}$), defines the power management granularity applied to the SPMs. The BU_{Dim} and $N_{BUsPerSector}$ are design-time parameters and should be carefully decided by the hardware designer.

Different sectors of the SPM can be individually power-gated using a multiple sleep-state transistor model. In the proposed techniques, it was adopted models supporting two (for STT-RAM arrays) and three (for SRAM arrays) power states: *OFF*, *Data Retentive (DR)* and *S3=ON*, where $E_{Static}(OFF) < E_{Static}(DR) < E_{Static}(ON)$. Still, each state have also increasing associated wake-up energies ($WE(OFF) > WE(DR) > WE(ON) = 0$). The electrical parameters of each power state to derive the static energy consumption and the overhead caused by the wakeup energies considers the characterization performed by (SINGH et al., 2007).

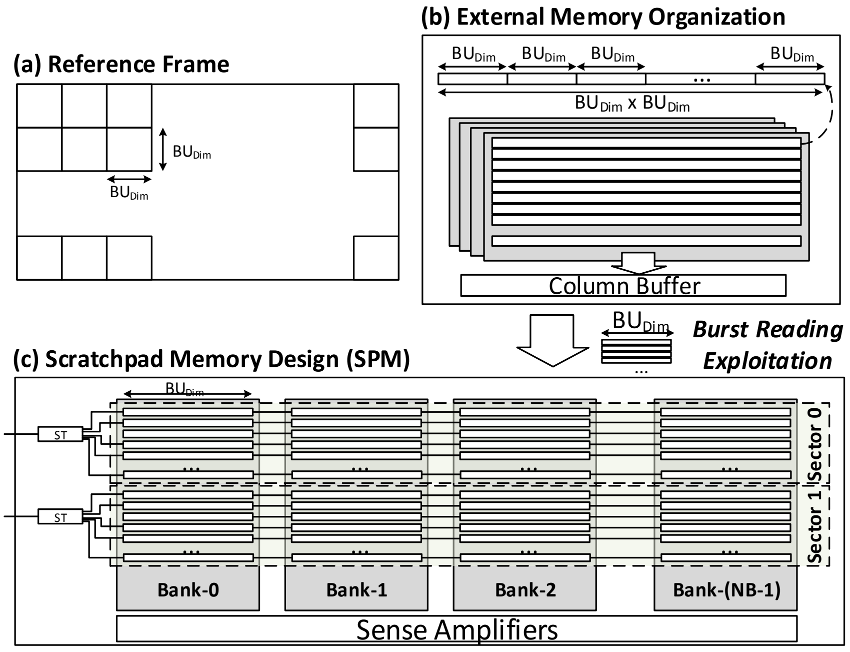


Figure 5.2: Adopted organization for the off- and on-chip memory parts.

Before moving forward to the physical and logical design, offline statistics-driven design space explorations were performed to improve the proposed design methodology of Hy-SVM with all possible application-specific knowledge of HEVC encoding.

5.3 Evaluations for Design Space Exploration of SPMs/HyMs Design

5.3.1. *Overlap Size Evaluation*

The proposed design methodology of Hy-SVM leverages the Tiles overlap behavior that depends on the search window size and on the video motion properties. Adaptive ME algorithms change the center of their searches by using spatial predictors (i.e. motion vectors of previously-coded CUs). Moreover, low motion CUs will lead to less memory access to search window samples. Hence, the optimal overlapping memory size for each video sequence follows a statistical distribution of the near-boundaries motion properties. Figure 5.3a depicts statistics of the Tiles overlap varying the search window size. On average, the overlap linearly increases with the growing of the maximum search range. The more or less concentrated distribution around the average size hints towards the video motion properties. Different regions near the Tile boundaries may have different motion characteristics, which leads to more or less memory access overlaps.

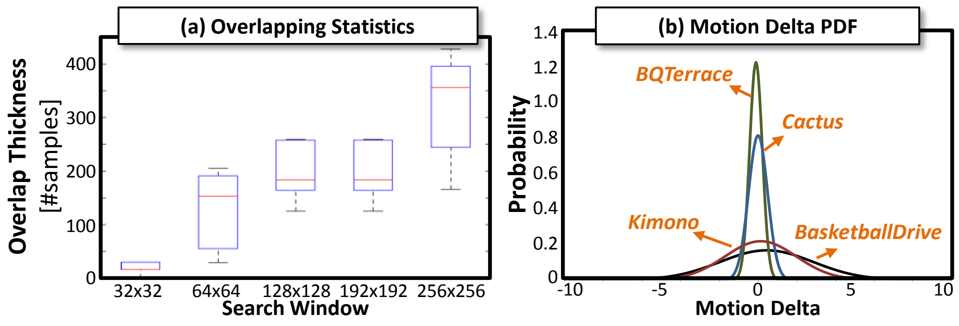


Figure 5.3: (a) Overlapping statistics for increasing search window size for evaluations with *BasketballDrive* test sequence; (b) motion delta distribution for several video sequences.

```
1.  determineMotionDelta(Video: V; TilePartitioning: TP):
2.  List$_\Delta$ = [ ];
3.  For all Frame ∈ V
4.      For all Tile$_{ID}$ ∈ TP
5.          PredMap[Tile$_{ID}$] = [ ];
6.          For all CU ∈ Tile$_{ID}$
7.              CU.performMotionEstimation();
8.              PredMap[Tile$_{ID}$].insert(CU.getUsedPredictor());
9.          End For
10.     For all TileBoundary$_{ID}$ ∈ TP
11.         //Let SideA and SideB the two tile boundary sides
12.         For all CU$_{SideA}$, CU$_{SideB}$ ∈ TileBoundary$_{ID}$
13.             PredA := PredMap[Tile$_{SideA}$][CU$_{SideA}$][Coord$_{ID}$];
14.             PredB := PredMap[Tile$_{SideB}$][CU$_{SideB}$][Coord$_{ID}$];
15.             $\Delta_{Value}$ := |PredA – PredB|;
16.             List$_\Delta$.append(Delta$_{Value}$);
17.         End For
18.     End For
19. End For
20. {$\mu_\Delta$, $\sigma_\Delta$} = norm_dist(List$_\Delta$);
21. return {$\mu_\Delta$, $\sigma_\Delta$};
```

Figure 5.4: Motion knowledge extraction for SharedL2 SPM sizing.

To statistically define the motion property near a specific Tile boundary of a given video, it was defined the Δ_{Motion} (motion delta) metric as being the *video correlated parameter used for determining the overlap size*, as presented in Figure 5.4. For each frame of the video and for each defined Tile boundary, the algorithm obtains the used ME spatial predictors (*lines 7-8*). The difference of the predictors used by the near-boundary CUs from the two Tile boundary sides (*SideA and SideB*) is then calculated (*lines 12-17*). This difference will represent the access search range of *SideA* CUs in the *SideB* reference frame region, and vice-versa. The Probability Density Function (PDF) of the Δ_{Motion} metric is then calculated (*line 20*), where μ_Δ and σ_Δ are the statistical average and standard deviation, respectively, of the motion delta parameters extracted from video encoding. The PDFs for HD1080p test sequences are plotted in Figure 5.3b. It can be noted diverse behaviors depending on the input video: high motion videos like *BasketballDrive* and *Kimono* present more spread distributions, while low motion videos like *Cactus* and *BQTerrace* have more concentrated distributions.

5.3.2. *Design Space Exploration of HyMs*

As already discussed, STT-RAM presents low static energy consumption while having high density. It allows us to designate the most part of the HyM to be composed of the STT-RAM array. In the meantime, BUs from the reference frame that cause high bit-toggling activity strongly decrease the STT-RAM lifetime, minimizing its non-volatility advantage. Thus, a small portion of SRAM is used to handle with these BUs. Although SRAM does not degrade from bit-toggling activity, it costs a large area and a high static energy consumption. Therefore, the main challenge involved in the HyMs design *is to leverage application-specific properties to design a well-balanced combination of SRAM and STT-RAM to minimize the static energy consumption whereas increasing the STT-RAM cells lifetime*. The bit-toggling activity (*BTA*) during a HyM write operation of a basic unit BU_0 over an already stored BU_1 is defined as the number of bits that toggles during the operation divided by the total number of written bits, as in Equation (11). BU_{Dim} is the horizontal/vertical BU dimension and NB_{Sample} the number of bits per sample. The *toggling_bits* function returns the number of collocated bits that are different between two numbers.

$$\mathrm{BTA}(\mathrm{BU}_0, \mathrm{BU}_1) = \frac{\sum_{y=0}^{\mathrm{BU_{Dim}}} \sum_{x=0}^{\mathrm{BU_{Dim}}} \mathrm{toggling\_bits}(\mathrm{BU}_{0(x,y)}, \mathrm{BU}_{1(x,y)})}{\mathrm{BU_{Dim}}^2 * \mathrm{NB_{Sample}}} \qquad (11)$$

Example: Figure 5.4 illustrates the bit-toggling activity map resultant from the write accesses during the replacement process between two consecutive reference frames (depicted in the left part). We can note that the higher activities correspond to the higher motion and textured areas of the video. In contrast, lower bit-toggling occurrences are related to more static and homogeneous frame parts. Therefore, the video content properties must be taken into account when dimensioning and managing HyMs.

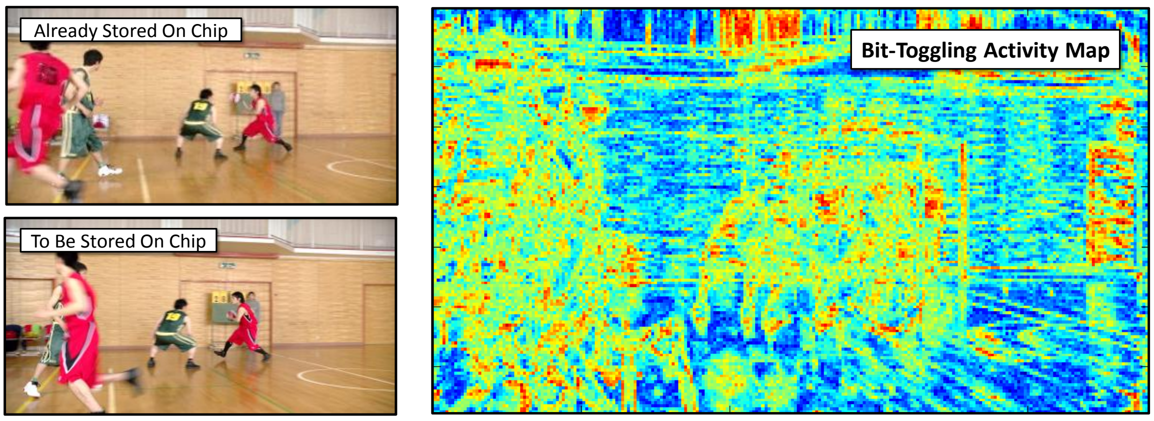


Figure 5.5: Example of reference frame replacement (for *BasketballDrive* sequence) and its corresponding bit-toggling activity map.

Figure 5.6 depicts the design space exploration controlled by an external parameter: the bit-toggling threshold (BT_{TH}). Reference frame basic units that lead to bit-toggling activities lower than BT_{TH} are assigned to STT-RAM, while higher values will direct the BU to SRAM. Our exploration varies the $\mathrm{BT_{TH}}$ from 0 (no activity) to 1 (maximum activity, all bits toggle) in steps of 0.01. We analyze our two optimization target variables: STT-RAM lifetime (Figure 5.6a) and SRAM size (Figure 5.6b), since it is known that the static energy efficiency is limited by the amount of SRAM cells (as discussed in Section 2.3). To find the best design point, we analyze an efficiency plot that relates both variables (see Figure 5.6c). We run this exploration for a set of video test sequences following our evaluation methodology (described in Section 7.1.1). The maximum efficiency point was discovered when $BT_{TH}=0.24$. Using this design point, we have that the SRAM usage factor (α_{SRAM}) is equal to 35% and the STT-RAM lifetime can be improved near to the optimal case (when no bit toggles): 0.83 normalized lifetime, as detailed in Section 7.5. From the Hy-SVM perspective, α_{SRAM}=35% means that the SRAM array will be sized as 35% of the STT-RAM capacity. Note that the α_{SRAM} factor is used for L2 HyMs design (*PrivL2* and *SharedL2*).

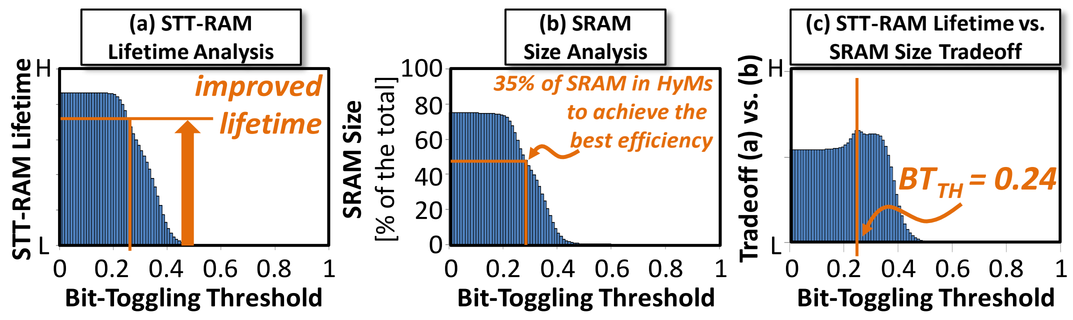


Figure 5.6: Design space exploration for joint (a) STT-RAM lifetime and (b) SRAM size optimization, resulting on a (c) tradeoff analysis.

The toggling activity at bit level is also exploited in Hy-SVM. Figure 5.7 depicts the accumulated statistics for toggling occurrences for each bit position using *ParkScene* and *NebutaFestival* test video sequences. Extended analyses considering other video sequences are presented in the Appendix B. It can be noticed near-zero bit-toggling activity for the two most significant bits (MSB) of the two sequences. Therefore, it means that even for BUs with high average bit-toggling activities, the two MSB toggle with a very low probability. This property is explored by always storing the two MSB in the STT-RAM, this way reducing the SRAM size (saving further static energy) while not penalizing the STT-RAM cells lifetime. This enables us to realize a fine-grained hybrid memory organization.

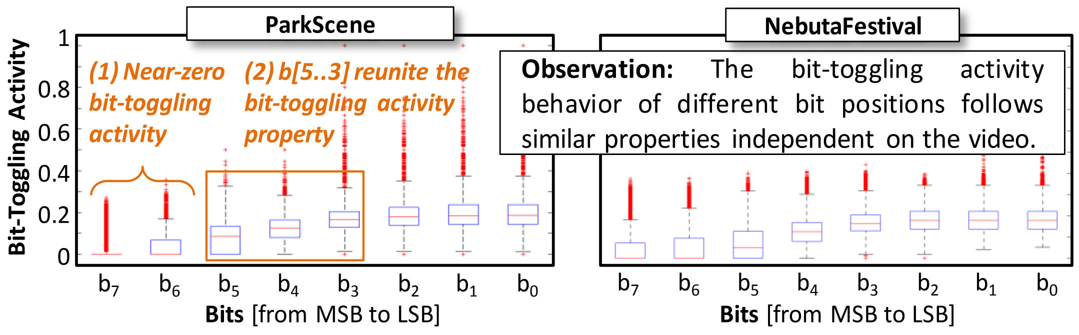


Figure 5.7: Bit-toggling activity of different bit positions.

Another important aspect from Figure 5.7 is that the bit range from b_5 to b_3 inherits the bit-toggling activity of the entire 8-bit sample. It means that only these three specific bits of the two involved data need to be compared to approximate the bit-toggling activity of this write operation. It is exploited by generating a bit-toggling key (BT_KEY) composed of only these three bits of some specific samples from reference frame BUs. This key aims to serve as an identifier that must be stored by the lifetime-aware data management unit. *The goal is to design an energy-efficient way to estimate the bit-toggling activity of each write operation.* Details regarding this data management unit are given in Section 6.3.

5.4 Design Methodology of SPMs/HyMs

5.4.1. *Overlap Sizing Parameters*

Equations (12) and (13) define the Tiles overlap sizing formula for the overlap thickness ($Ov_{Thickness}$) and length (Ov_{Length}). These formula are used at design-time to properly derive the SharedL2 SPMs parameters. The $Ov_{Thickness}$ is calculated from the search window width or height, since it defines the maximum range ME can reach when searching in the reference frames. Additionally, ME start point can be displaced by prior analysis from neighboring motion predictors. Thus, the search window center can vary according to the motion field of Tiles boundaries. To represent that, an off-line statistical parameter Δ_{Motion} is inserted to scale the overlap thickness to be adapted to the average case of test sequences. The *Ov_{Length}* is related to the frame width or height, when overlaps are formed around horizontal or vertical Tiles boundaries, respectively.

$$\mathrm{Ov_{Thickness}}(\mathrm{TB_{ID}}) = \begin{cases} \Delta_{\mathrm{Motion}} \times \mathrm{SW_W}, & \text{if vertical boundary} \\ \Delta_{\mathrm{Motion}} \times \mathrm{SW_H}, & \text{if horizontal boundary} \end{cases} \tag{12}$$

$$\mathrm{Ov_{Length}}(\mathrm{TB_{ID}}) = \begin{cases} \mathrm{Frame_H}, & \text{if vertical boundary} \\ \mathrm{Frame_W}, & \text{if horizontal boundary} \end{cases} \tag{13}$$

5.4.2. *L1 Level SPMs Design*

Based on the memory organization defined in Section 5.2, we determine the sizing for the SPMs Levels in the proposed Hy-SVM architecture. As already explained, all SPMs (PrivL1, PrivL2, and SharedL2 levels) are composed of *BU_{Dim}* memory banks as in Equation (14), which allows parallel access of one entire BU. However, the other SPM parameters are different depending on the Hy-SVM level.

$$\mathrm{PrivL1_{NB}} = \mathrm{PrivL2_{NB}} = \mathrm{SharedL2_{NB}} = \mathrm{BU_{Dim}} \tag{14}$$

The PrivL1 SPMs store Tile-specific search window samples, requiring *$PrivL1_{NL}$* memory lines, as expressed by Equation (15). The first memory bank of a PrivL1 SPM must store, besides the first BU row, three control data: the horizontal and vertical BU frame position, and a validate bit (as in Equation (16)). This information is important for MAMU to properly manage hit and miss occurrences. In total, *$PrivL1_{LS}$* bits are required for each memory bank, where *NB_{Sample}* is the number of bits per reference frame sample. The power management of PrivL1 SPMs, which is based on longer sleep duration opportunities from balanced memory pressure (as properly explained in Section 6.4.2.1), is applied for each defined SPM sector. The already defined *$N_{LinesPerSector}$* parameter indicates the adopted management level (as already discussed in Section 5.2). Thus, Equation (17) defines the sector size *$PrivL1_{SS}$* and Equation (18) presents the number of sectors within a PrivL1 SPM (*$PrivL1_{NS}$*). These parameters directly affects the overhead of implementing APMUs power maps.

$$\mathrm{PrivL1_{NL}} = \mathrm{N_{BUsPerPrivL1}} = \left\lceil \frac{\mathrm{SW_W} \times \mathrm{SW_H}}{\mathrm{BU_{Size}}} \right\rceil \tag{15}$$

$$\mathrm{PrivL1_{LS}} = \begin{cases} (|\mathrm{BU_{XPos}}| + |\mathrm{BU_{YPos}}| + 1) + \mathrm{BU_{Dim}} \times \mathrm{NB_{Sample}} & \text{if } \mathrm{Bank_0} \\ \mathrm{BU_{Dim}} * \mathrm{NB_{Sample}} & \text{otherwise} \end{cases} \tag{16}$$

$$\mathrm{PrivL1_{SS}} = \mathrm{N_{LinesPerSector}} \times \mathrm{BU_{Size}} \tag{17}$$

$$\mathrm{PrivL1_{NS}} = \frac{\mathrm{PrivL1_{NL}}}{\mathrm{PrivL1_{SS}}} \tag{18}$$

5.4.3. *L2 Level HyMs Design*

The L2 level of Hy-SVM completely stores one reference frame, by having its samples distributed along the PrivL2 and SharedL2 HyMs. As illustrated in Figure 5.8, hybrid memory design is exploited in L2 level of Hy-SVM by implementing HyMs containing: (a) one STT-RAM SPM, which is designed to have specific memory lines for all BUs within Tile-specific region of reference frame (in case of PrivL2), or within the overlapping regions (in case of SharedL2); and (b) one SRAM SPM, which provides storage only for BUs estimated with high bit-toggling activities, alleviating the STT-RAM part of high-cost write operations.

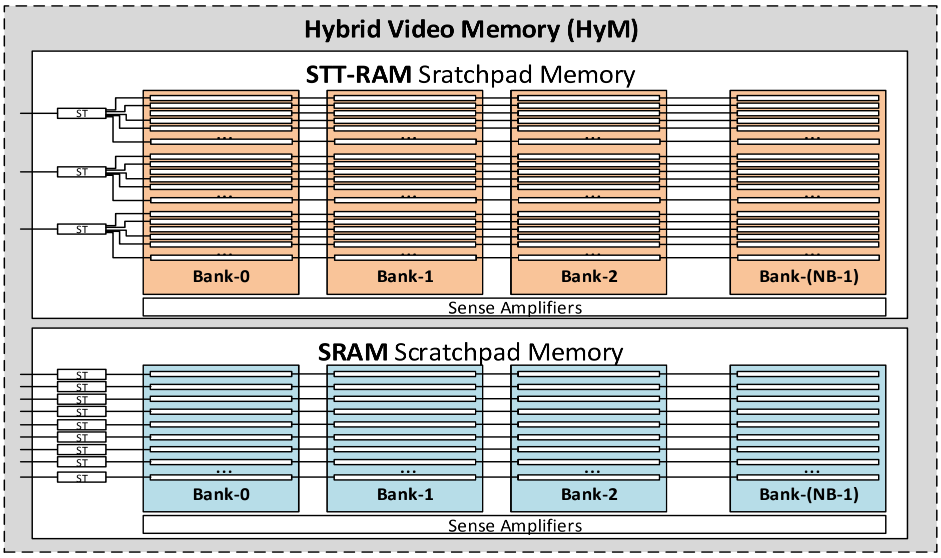


Figure 5.8: Hybrid video memories (HyMs) physical organization for L2 level (PrivL2 and SharedL2).

As definitions from the design space exploration HyMs (presented in Section 5.3.2) we have: (a) the extraction of the α_{SRAM} factor, which represents the size of SRAM portion (related to HyM total size) that leads to the best optimization point between STT-RAM lifetime and on-chip static energy consumption; and (b) the design decision to always store the two MSB of all reference frame samples in the STT-RAM SPM of each HyM.

The first calculations refer to the STT-RAM SPMs of PrivL2 and SharedL2 HyMs, which are designed to store all corresponding BUs. The SRAM SPMs design will be described based on STT-RAM defined parameters.

STT-RAM SPMs Design: The $PrivL2_{STT\text{-}NL}$ number of memory lines depends on the frame resolution and the number of Tiles, as expressed in Equation (19). In another perspective, the SharedL2 SPMs requires $SharedL2_{STT\text{-}NL}$ lines, which is related to the $Ov_{Thickness}$ and Ov_{Length} overlap parameters; see Equation (20). The PrivL2 STT-RAM SPMs are designed to guarantee that all BUs within the same reference frame have a specific associated memory line. Thus, it is not necessary to keep stored the frame position coordinates of the stored BU in a specific SPM line. To ensure correct hit/miss detection by MAMU, a validate bit is stored alongside the first BU row in $Bank_0$. The same scheme is adopted for SharedL2 SPMs. Thus, the line size of each L2 STT-RAM SPM bank ($L2_{STT\text{-}LS}$) is defined in Equation (21). Our APMU acts on L2 level of Hy-SVM to reduce on-chip static energy consumption. Note that as L2 SPMs are implemented as STT-RAM arrays, the shutdown operation of specific memory sectors does not imply on off-chip memory re-fetching, due to the non-volatile nature of STT-RAM cells. The power gating is applied for each memory sector. In doing so, the sector size $L2_{STT\text{-}SS}$, which is the same for PrivL2 and SharedL2 STT-RAM SPMs, is defined according to this design-time parameter, as in Equation (22). As result, the number of memory sectors ($PrivL2_{STT\text{-}NS}$ and $SharedL2_{STT\text{-}NS}$), which directly affects the APMU design, is defined in Equations (23) and (24), respectively.

$$\mathrm{PrivL2_{STT-NL}} = \mathrm{N_{BUsPerPrivL2-STT}} = \left\lceil \frac{\mathrm{Frame_W} \times \mathrm{Frame_H}}{\mathrm{BU_{Size}} \times \mathrm{N_{Tiles}}} \right\rceil \quad (19)$$

$$\mathrm{SharedL2_{STT-NL}} = \mathrm{N_{BUsPerSharedL2-STT}} = \left\lceil \frac{\mathrm{Ov_{Thickness}} \times \mathrm{Ov_{Length}}}{\mathrm{BU_{Size}}} \right\rceil \quad (20)$$

$$L2_{STT-LS} = \begin{cases} BU_{Dim} + 1 & \text{if } Bank_0 \\ BU_{Dim} & \text{otherwise} \end{cases} \quad (21)$$

$$L2_{STT-SS} = N_{LinesPerSector} \times BU_{Size} \quad (22)$$

$$PrivL2_{STT-NS} = \frac{PrivL2_{STT-NL}}{L2_{STT-SS}} \quad (23)$$

$$SharedL2_{STT-NS} = \frac{SharedL2_{STT-NL}}{L2_{STT-SS}} \quad (24)$$

SRAM SPMs Design: Considering the SRAM part, the design methodology is applied equally for PrivL2 and SharedL2 HyMs by using the adopted sizing for the STT-RAM as parameter. The number of lines of each SRAM SPM bank (*$PrivL2_{SRAM-NL}$* and *$SharedL2_{SRAM-NL}$*) is derived from applying the offline statistics-based α_{SRAM} factor to the already defined *$PrivL2_{STT-NL}$* and *$SharedL2_{STT-NL}$* parameters of STT-RAM part, as defined in Equations (25) and (26). Since only a small part of the data is assigned to SRAM (to alleviate STT-RAM from high bit-toggling activities), the horizontal and vertical BU frame positions, as well as a validate bit, must be stored along with the reference data. Furthermore, as the two MSB of all samples are always stored in the STT-RAM SPMs, the line size *$L2_{SRAM-LS}$* can be reduced, as defined in Equation (27). To provide a fine-grain power management for the SRAM SPMs, the power states are assigned specifically for each memory line (as illustrated in Figure 5.8), leading to the number of sectors (*$L2_{SRAM-NS}$*) equals to *$L2_{SRAM-NL}$*.

$$PrivL2_{SRAM-NL} = N_{BUsPerPrivL2-SRAM} = \lceil PrivL2_{STT-NL} * \alpha_{SRAM} \rceil \quad (25)$$

$$SharedL2_{SRAM-NL} = N_{BUsPerSharedL2-SRAM} = \lceil SharedL2_{STT-NL} * \alpha_{SRAM} \rceil \quad (26)$$

$$L2_{SRAM-LS} = \begin{cases} (|BU_{XPos}| + |BU_{YPos}| + 1) + BU_{Dim} * (NB_{Sample} - 2) & \text{if } Bank_0 \\ BU_{Dim} * (NB_{Sample} - 2) & \text{otherwise} \end{cases} \quad (27)$$

To provide a proper hardware infrastructure for the management layer of Hy-SVM, small on-chip memory blocks are designed. They serve to keep stored: (a) the monitored and predicted overlap representations (Predicted and Monitored Overlap Tables – POTs and MOTs); the power maps for APMU operation (frame- and CTU-level power maps); as well as a data management table (DMT) to manage HyMs write operations. This hardware data structures will be explained along with the related management schemes in next chapter.

6. ENERGY-EFFICIENT MEMORY MANAGEMENT LAYER

This chapter introduces the memory management layer, which enables run-time adaptation and improved energy savings to the designed Hy-SVM architecture. Initially, details regarding the overlap prediction scheme is presented in **Section 6.1**. An accurate prediction of the overlap characteristics (size, shape and displacement) is of key importance to enable energy-efficient inter-Tiles data reuse. In a different perspective, unbalanced memory pressure caused when considering Tiles-accumulated memory requirements is treated by the memory pressure management scheme, which is properly defined in **Section 6.2**. Well-balanced memory transmission is extremely necessary to avoid off-chip power peaks and allow a better prediction for the sleep durations of on-chip SPM sectors. STT-RAM lifetime and write access inefficiency is handled by the lifetime-aware data management scheme, explained in **Section 6.3**. In addition, distributed on-chip management blocks composed of memory access management units (MAMUs) and adaptive power management units (APMUs) are presented in **Section 6.4**. They implemented hardware support of data structures and control logic to properly manage the data dynamics of Hy-SVM.

6.1 Overlap Prediction

We focus our evaluations and proposed overlap prediction scheme in the Low Delay (LD) and Random Access (RA) prediction structures, as illustrated in Figure 6.1. Each arrow denotes a prediction dependency evaluated by the ME, starting from the current frame and pointing to the used reference frame. We assign an overlap identification (Ov_{ID}) for each prediction dependency. Further, another important parameter is the distance between the current and reference frame of each Ov_{ID}, represented by the notation D_{ME}. It is defined as the absolute difference between the picture exhibition order number between the two frames: $D_{ME}(Ov_{ID}) = |F_{Curr} - F_{Ref}|$ (depicted in the bottom of Figure 6.1). For example, the D_{ME} of the prediction *RA2* is calculated as $D_{ME}(RA2) = |4 - 8| = 4$.

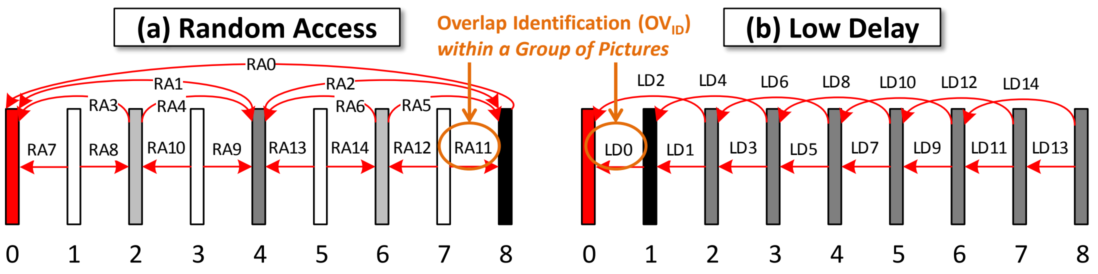


Figure 6.1: Overlap identification (Ov_{ID}) in (a) Random Access and (b) Low Delay HEVC encoder configurations.

Note that each ME will lead to a formation of an overlapping region. The characteristics of the overlaps were evaluated to base our run-time overlap prediction scheme (Section 6.1.1). An accurate estimation of such properties is important (a) to improve inter-Tiles data reuse (exploited by the SharedL2 SPMs), as well as (b) to provide less-frequent *ON-OFF* switching activities, leading to higher energy savings for our adaptive power management scheme. To support the variability of the overlap characteristics, a light-weight overlap data representation is proposed (Section 6.1.2). The overlap prediction scheme is described in Section 6.1.3.

6.1.1. *Overlap Correlation Evaluations*

Memory analyses were performed with the goal of identifying correlated parameters of overlap formations between consecutive MEs. The evaluations consider three important overlap characteristics: size, shape, and displacement.

Analysis-1 (Overlap Size): Figure 6.2 presents an evaluation of the overlap size by exploiting MEs with different D_{ME} parameters. In this case, we are interested in the number of redundant memory accesses within a reference frame depending on the absolute value of the distance D_{ME}. Thus, this analysis does not consider the prediction direction. We can note that the overlap size reduces when lower D_{ME} MEs are executed. Therefore, an insight is to leverage the size of past overlaps in our prediction scheme. In doing so, the relation between the D_{ME} factors must be taken into account to scale the predicted overlap accordingly. Our APMU can exploit it by dynamically applying relaxed or aggressive power gating to improve the SPMs energy efficiency according to predicted memory demand.

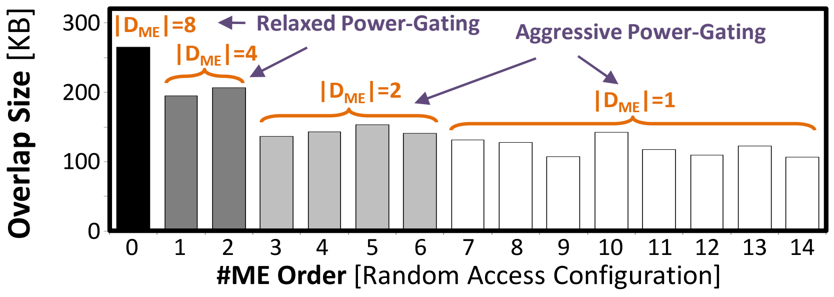


Figure 6.2: Example of overlap sizing variation for several temporal distances (D_{ME} factor).

Analysis-2 (Overlap Shape): Besides the size, another important aspect is the overlap shape, which may significantly change along the overlap length. We can note this dynamic behavior in Figure 6.3, where the shape varies according to the video content. Not exploiting this variation may lead to inefficiency to memory energy consumption (on- and off-chip parts). Furthermore, there is a significant similarity between the shapes of overlaps when analyzing consecutive ME processing, as can be noted in Figure 6.3. Therefore, the shape characteristics of previous formations can be used as reference to improve the prediction accuracy for the next overlaps.

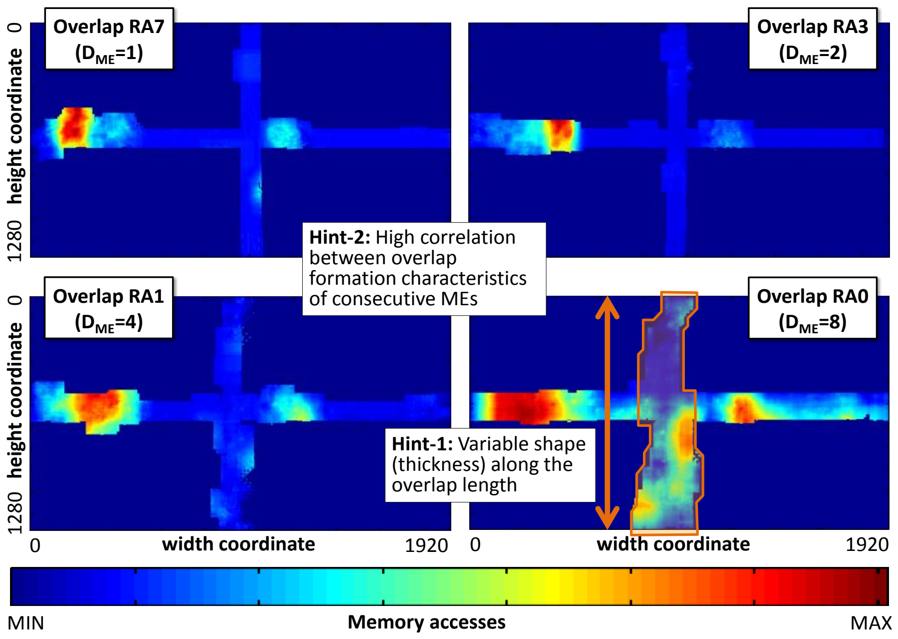


Figure 6.3: Correlation between consecutive overlaps (RA5, RA3, RA1 and RA0), considering Random Access prediction structure.

Analysis-3 (Overlap Displacement): Figure 6.4 presents an analysis (Probability Density Function charts) comparing the overlap displacement for ME steps with different D_{ME} factors. The displacement, in this evaluation, was measured by the distance of the center of the actual overlap regarding the Tiles boundary. When we compare the generated overlap between MEs with the same prediction direction (Figure 6.4a), we can note that higher D_{ME} factors lead to higher and spreader overlap displacements. In another vein, ME operations with lower D_{ME} values lead overlaps centered nearer the Tiles boundary, as well with a more concentrated behavior. In Figure 6.4b, comparing MEs with opposite prediction directions (D_{ME} values with different signals), we can observe opposite displacements in the formed overlaps. Therefore, regarding this aspect, our insight is to leverage the past overlap displacement weighted by the difference of D_{ME} factors.

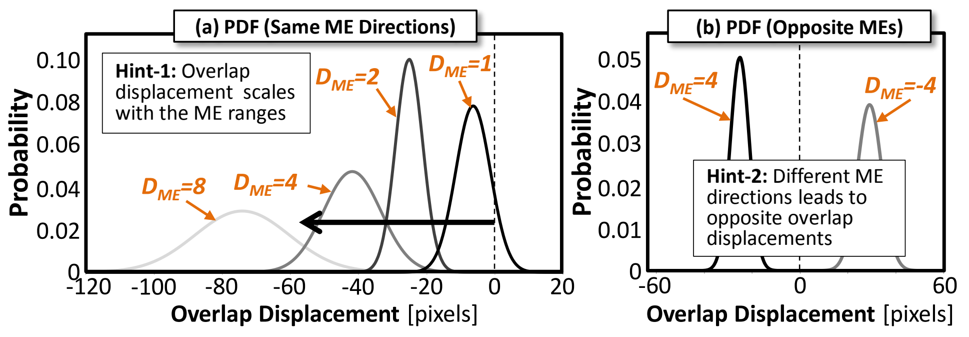


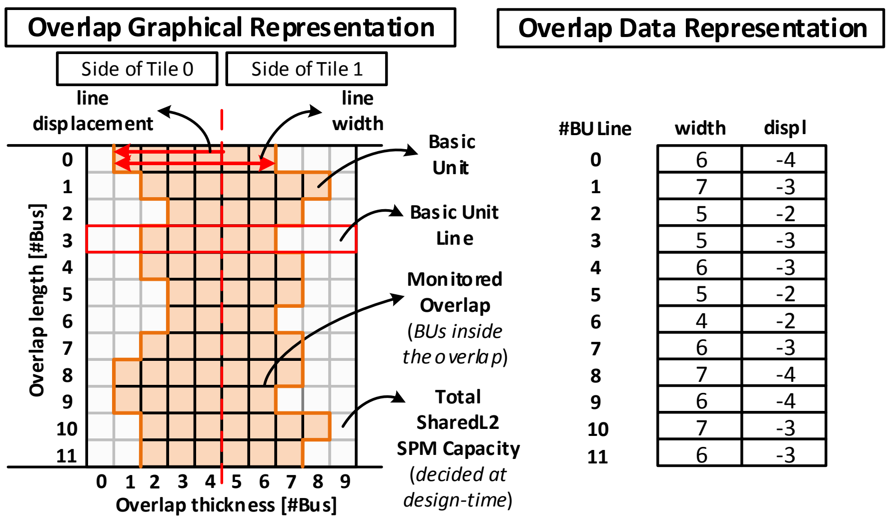
Figure 6.4: Overlap displacement correlation analysis.

To exploit the discussed overlaps correlation in our prediction unit, we implement an overlap representation that properly models the absolute size, the variable shape and displacement properties.

6.1.2. *Overlap Representation*

Equation (28) models an overlap as an ordered set of tuples, each one containing width (*width$_i$*) and displacement *(displ$_i$)* information of a specific basic unit line *i* within the overlap. The level of representation is based on the adopted BU dimension, being compliant to Hy-SVM organization.

$$\mathrm{Ov_{ID}} = \left\{(\mathrm{width_i}, \mathrm{displ_i}), \forall\, BU\ line\ i\ \middle|\ 0 \le \mathrm{i} < \mathrm{Ov_{Length}}/\mathrm{BU_{Dim}}\right\} \tag{28}$$



#BU Line	width	displ
0	6	-4
1	7	-3
2	5	-2
3	5	-3
4	6	-3
5	5	-2
6	4	-2
7	6	-3
8	7	-4
9	6	-4
10	7	-3
11	6	-3

Figure 6.5: (a) Graphical and (b) data representation of an overlap by our energy-efficient management of Hy-SVM.

Figure 6.5 illustrates an example of overlap representation. In the graphical view (Figure 6.5a), we can observe the possibility of modeling the variations of width and displacement along the overlap length. For each BU line, the width is related to the overlap thickness (in number of BUs), while the displacement is expressed as the distance of the first BU from the Tiles boundary center. Hence, the design-time parameters Ov_{Length} and $Ov_{Thickness}$ related to the overlap thickness and length (previously defined in Section 5.2), utilized to design the SharedL2 SPMs, are refined to provide a more accurate representation of the actual formed overlap. The mapping between the graphical and the data representation is presented in Figure 6.5b. We can note that each BU line within the overlap has associated width and displacement information.

Considering a 4-Tile HD1080p HEVC encoder, 256x256 search window size, and BU_{Dim}=8, the overlap representation for the horizontal Tiles boundary requires 480 bytes, while the vertical overlap occupies 136 bytes. These values represent a negligible overhead, especially when comparing to the hardware resources required to implement L1 and L2 SPM Levels.

6.1.3. *Overlap Prediction Scheme*

Our scheme is inspired on the video coding idea of selecting several references (past coded information) to predict the behavior of the data that is being processed. Therefore, for each overlap that is being predicted (called *current overlap*), the information of past monitored overlap formations (called *reference overlaps*) are exploited. As result, an estimation of the formation characteristics for the current overlap is generated (called *predicted overlap*). In this context, there are two key data structures: the *Monitored Overlaps Table* (MOT) and the *Predicted Overlap Table* (POT).

Figure 6.6 depicts the flowchart of our overlap prediction scheme, as well as its integration with Hy-SVM on-chip management units (detailed presented in Sections 6.4 and 6.4.2). During a ME processing, our Memory Monitoring Unit monitors the inter-Tiles redundant accesses. This unit utilizes one bitmap for each HEVC processing unit to identify the accessed BUs within a reference frame. As result, the bitmaps are combined, and the monitored overlap representation (presented in the previous section) is generated and stored in the MOT. There is a specific MOT_{TB} for each Tile boundary *TB*, which is responsible to store a historic of the past monitored overlap formations of this specific boundary.

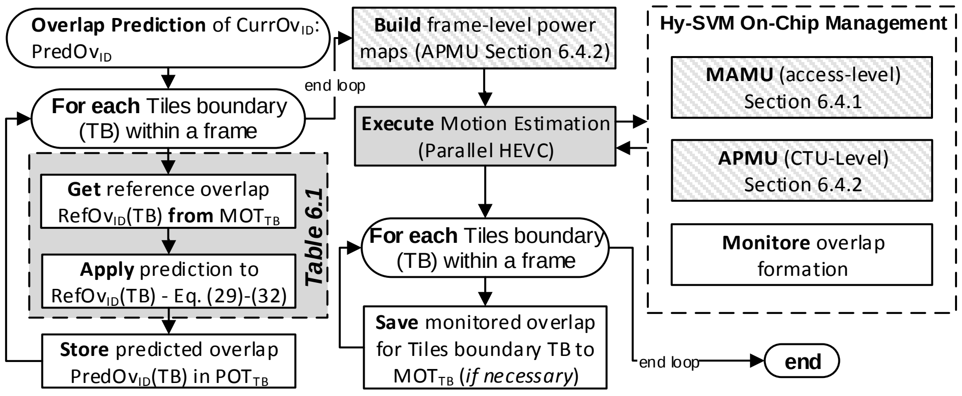

Overlap Prediction of CurrOv_ID: PredOv_ID
For each Tiles boundary (TB) within a frame
Get reference overlap RefOv_ID(TB) from MOT_TB
Apply prediction to RefOv_ID(TB) - Eq. (29)-(32)
Table 6.1
Store predicted overlap PredOv_ID(TB) in POT_TB
end loop
Build frame-level power maps (APMU Section 6.4.2)
Execute Motion Estimation (Parallel HEVC)
For each Tiles boundary (TB) within a frame
Save monitored overlap for Tiles boundary TB to MOT_TB (if necessary)
end loop
end
Hy-SVM On-Chip Management
MAMU (access-level) Section 6.4.1
APMU (CTU-Level) Section 6.4.2
Monitore overlap formation

Figure 6.6: Flowchart of our run-time overlap prediction scheme and its relation to Hy-SVM on-chip management units.

For each Tiles boundary within a frame, our scheme accesses the MOT_{TB} to get the reference overlap *$RefOv_{ID}(TB)$* that will be used for the prediction of *$CurrOv_{ID}(TB)$*. To minimize the MOTs size and guarantee the best possible correlation between current and reference overlaps, proposes a prediction assignment based on the correlations of MEs, considering the Random Access and Low Delay configurations. The overlap identifications follow the notations defined in Table 6.1. Based on this assignment, a prediction operation is applied to estimate the predicted overlap based on the monitored information from the selected reference overlap. The prediction process is based on one of four possible operations: *downscale*, *upscale, invert* or *copy*; as defined in Eqs. (29)-(32), where the α and β are offline statistical factors that were extracted by experimental analysis using real-world video coding scenarios[9].

Table 6.1: Overlap Prediction Assignment for *Random Access* and *Low Delay* HEVC Encoder Configurations

Random Access			Low Delay		
Curr. Ov$_{ID}$	**Prediction Operation**	**Ref. Ov$_{ID}$**	**Curr. Ov$_{ID}$**	**Prediction Operation**	**Ref. Ov$_{ID}$**
RA0	off-line stats. (if first frame) or copy RA0 from previous GOP		LD0	off-line stats. (if first frame) or copy RA0 from previous GOP	
RA1	downscale(α)	RA0	LD1	copy	LD0
RA2	invert	RA1	LD2	upscale(β)	LD1
RA3	downscale(α)	RA1	LD3	copy	LD1
RA4	invert	RA3	LD4	upscale(β)	LD3
RA5	downscale(α)	RA2	LD5	copy	LD3
RA6	invert	RA5	LD6	upscale(β)	LD5
RA7	downscale(α)	RA3	LD7	copy	LD5
RA8	invert	RA7	LD8	upscale(β)	LD7
RA9	downscale(α)	RA4	LD9	copy	LD7
RA10	invert	RA9	LD10	upscale(β)	LD9
RA11	downscale(α)	RA5	LD11	copy	LD9
RA12	invert	RA11	LD12	upscale(β)	LD11
RA13	downscale(α)	RA6	LD13	copy	LD11
RA14	invert	RA13	LD14	upscale(β)	LD13

$$\begin{array}{l}\underline{\mathrm{downscale}(\mathrm{PredOv_{ID}}, \mathrm{RefOv_{ID}}, \alpha):}\\ \mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{width} \leftarrow \lceil \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{width} \times \alpha \rceil\\ \mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{displ} \leftarrow \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{displ},\\ \forall\ \mathrm{BU\ line\ i}, where\ \alpha < 0\end{array} \qquad (29)$$

$$\begin{array}{l}\underline{\mathrm{upscale}(\mathrm{PredOv_{ID}}, \mathrm{RefOv_{ID}}, \beta):}\\ \mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{width} \leftarrow \lceil \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{width} \times \beta \rceil\ and\\ \mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{displ} \leftarrow \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{displ},\\ ,\forall\ \mathrm{BU\ line\ i}, where\ \beta > 0\end{array} \qquad (30)$$

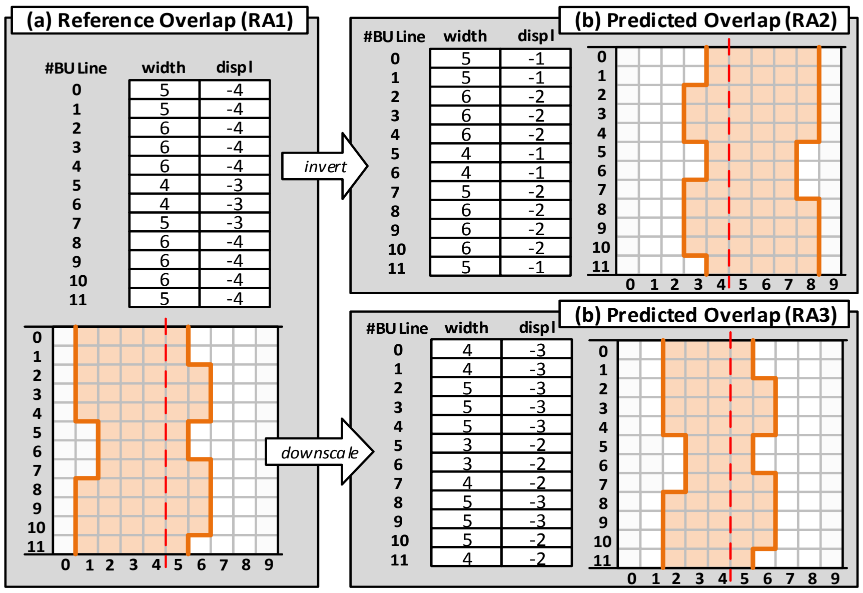
$$\begin{array}{l}\underline{\mathrm{invert}(\mathrm{PredOv_{ID}}, \mathrm{RefOv_{ID}}):}\\ \mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{width} \leftarrow \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{width}\ and\\ \mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{displ} \leftarrow -(\mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{width} + \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{displ}), \forall\ \mathrm{BU\ line\ i}\end{array} \qquad (31)$$

$$\begin{array}{l}\underline{\mathrm{copy}(\mathrm{PredOv_{ID}}, \mathrm{RefOv_{ID}}):}\\ \mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{width} \leftarrow \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{width}\ and\\ \mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{displ} \leftarrow \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{displ}, \forall\ \mathrm{BU\ line\ i}\end{array} \qquad (32)$$

Example-1: Let the RA1 prediction dependency be processed by ME and the inter-Tiles redundant memory accesses be monitored, generating the *RefOv$_{RA1}$* (illustrated in Figure 6.7a). From the proposed prediction assignment for Random Access configuration presented in Figure 6.1, the selected prediction operation for the next *PredOv$_{RA2}$* and *PredOv$_{RA3}$* are *invert* and *downscale*, respectively. In Figure 6.7b we can observe the result of invert prediction operation, where the overlap is displaced from left to right part related to the Tiles boundary. In this case, the RA1 and RA2 have the same absolute value of *D$_{ME}$*, but with different signals: *D$_{ME}$(RA1)=4* and *D$_{ME}$(RA2)=-4*. In this case, due to the opposite motion directions, the overlap formations tends to be displaced (as motivated in Analysis-2 of Section 6.1.1). The estimation of

[9] We adopted: *α=0.75* and *β=1.25* in our experiments.

PredOv$_{RA3}$, reduces the overlap width parameters by a *α=0.75* factor. The prediction dependencies RA1 and RA3 have same direction but different distances: *D$_{ME}$(RA1)=4* and *D$_{ME}$(RA2)=2*. In doing so, the overlap formation for RA3 tends to be smaller than RA1, since lower motion fields will be detected.


(a) Reference Overlap (RA1)
invert
downscale
(b) Predicted Overlap (RA2)
(b) Predicted Overlap (RA3)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9

(a) Reference Overlap (RA1)

#BU Line	width	displ
0	5	-4
1	5	-4
2	6	-4
3	6	-4
4	6	-4
5	4	-3
6	4	-3
7	5	-3
8	6	-4
9	6	-4
10	6	-4
11	5	-4

(b) Predicted Overlap (RA2)

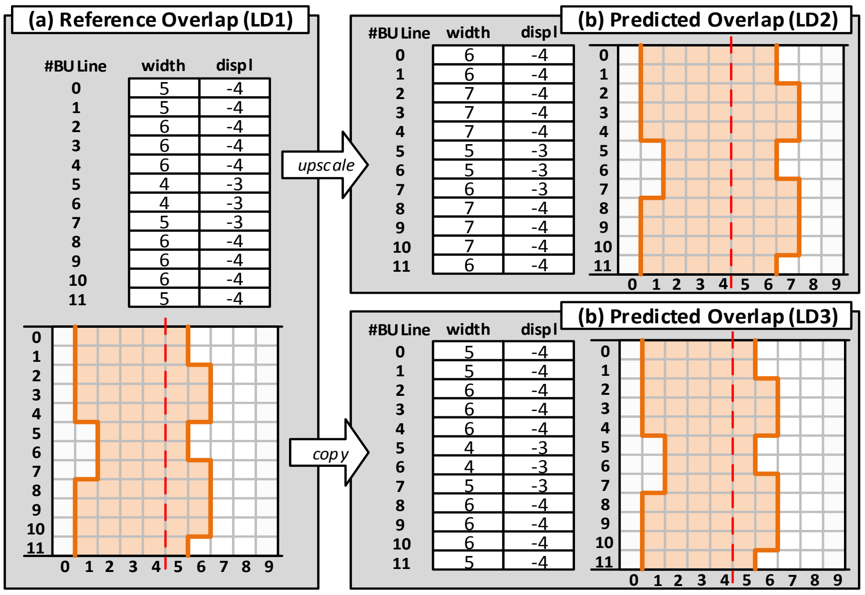
#BU Line	width	displ
0	5	-1
1	5	-1
2	6	-2
3	6	-2
4	6	-2
5	4	-1
6	4	-1
7	5	-2
8	6	-2
9	6	-2
10	6	-2
11	5	-1

(b) Predicted Overlap (RA3)

#BU Line	width	displ
0	4	-3
1	4	-3
2	5	-3
3	5	-3
4	5	-3
5	3	-2
6	3	-2
7	4	-2
8	5	-3
9	5	-3
10	5	-2
11	4	-2

Figure 6.7: Example of overlap prediction operations when estimating RA2 (invert) and RA3 (downscale; using α=0.75) formations from RA1 monitored reference overlap.

Example-2: In the perspective of Low Delay HEVC configuration, consider that LD1 prediction was computed and the *RefOv$_{LD1}$* was monitored (depicted in Figure 6.8a). According to the adopted overlap prediction assignment of Table 6.1, the estimation steps of *PredOv$_{LD2}$* and *PredOv$_{LD3}$* are performed from the *upscale* (Figure 6.8b) and *copy* (Figure 6.8c) operations, respectively. The upscale is applied since the overlap *Ov$_{LD2}$* is generated during a ME of D$_{ME}$=2, while the monitored overlap *RefOv$_{LD1}$* has *D$_{ME}$=1*. This means that higher motion tends to be observed in *Ov$_{LD2}$* (compared to *Ov$_{LD1}$*), leading to a higher probability of increased overlap.


(a) Reference Overlap (LD1)
upscale
copy
(b) Predicted Overlap (LD2)
(b) Predicted Overlap (LD3)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9

(a) Reference Overlap (LD1)

#BU Line	width	displ
0	5	-4
1	5	-4
2	6	-4
3	6	-4
4	6	-4
5	4	-3
6	4	-3
7	5	-3
8	6	-4
9	6	-4
10	6	-4
11	5	-4

(b) Predicted Overlap (LD2)

#BU Line	width	displ
0	6	-4
1	6	-4
2	7	-4
3	7	-4
4	7	-4
5	5	-3
6	5	-3
7	6	-3
8	7	-4
9	7	-4
10	7	-4
11	6	-4

(b) Predicted Overlap (LD3)

#BU Line	width	displ
0	5	-4
1	5	-4
2	6	-4
3	6	-4
4	6	-4
5	4	-3
6	4	-3
7	5	-3
8	6	-4
9	6	-4
10	6	-4
11	5	-4

Figure 6.8: Example of overlap prediction operations when estimating LD2 (invert) and LD3 (upscaling; using α=1.25) formations from LD1 monitored reference overlap.

The predicted overlap *$PredOv_{ID}(TB)$* is stored in the POT_{TB} to become available for MAMUs and APMUs operations. Details regarding the implementation of Hy-SVM on-chip management units are given as follows.

6.2 Memory Pressure Management

This section describes the proposed memory pressure management scheme, which focuses on balancing the instant memory bandwidth in order to optimize the external memory communication channel usage. As consequence of a well-balanced memory pressure, longer sleep durations for the PrivL1 SPMs can be exploited to additionally save on-chip static energy (as presented in Section 6.4.2.1).

Initially, application-specific evaluations are presented to capture the main correlations of memory accesses in two perspectives: intra-frame (among spatial neighboring CTUs inside the same frame) and inter-frame (among CTUs of consecutive frames). Based on this knowledge, the memory pressure management scheme is presented, being composed of three main steps: (1) memory pressure prediction, (2) run-time statistics-based CTU memory classification, and (3) CTU re-scheduling.

6.2.1. *Intra- and Inter-Frame Memory Accesses Correlations*

If it is possible to accurately predict the memory requirements for a given CTU, it can be exploited by a power manager to balance the memory pressure in a very efficient way. In case of high frame rates (30-60 fps), significant temporal correlation exists, i.e., the neighboring frames have similar memory access behavior, as depicted in Figure 6.9. Additionally, high video frame resolutions (e.g., FullHD=1920x1080 to 4K=3840x2160) increase the spatial correlations between neighboring CTUs within the same frame. Furthermore, note that the memory pressure for each CTU also depends upon their corresponding video content characteristic (like texture and motion content). Therefore, *the key is to leverage the knowledge from the monitored memory pressure of spatially- and temporally-neighboring CTUs to obtain a high quality prediction of the actual memory pressure for a given CTU.*

While balancing the memory pressure is important from the external memory perspective, it is also crucial to take care of the Tile-private on-chip SPMs. In this case, long sleep durations (and consequently more static energy savings) can be achieved by consecutively encoding CTUs with similar video content properties (like texture and motion), thus similar memory pressure. Figure 6.10 shows Tiles with *less* memory requirements (like Tile 1) and *more* memory demands (like Tile 2). In this case, longer sleep durations and higher energy savings can be obtained for the SPM of processing the Tile 1. Furthermore, re-scheduled CTU processing orders for a well-balanced memory pressure tends to group similar properties CTUs to be consecutively encoded, providing even higher sleep durations (as demonstrated in Section 6.4.2.1). Hence, an important *challenge here is how to leverage the CTU re-schedule for memory pressure balancing and increased sleep durations for efficient PrivL1 SPM power management.*

Therefore, the main goal of the proposed memory pressure management is *to leverage application-specific properties for memory pressure balancing and, additionally, PrivL1 SPM's static energy reduction targeting parallelized HEVC encoding*. The following sections will describe all modules of the proposed scheme.

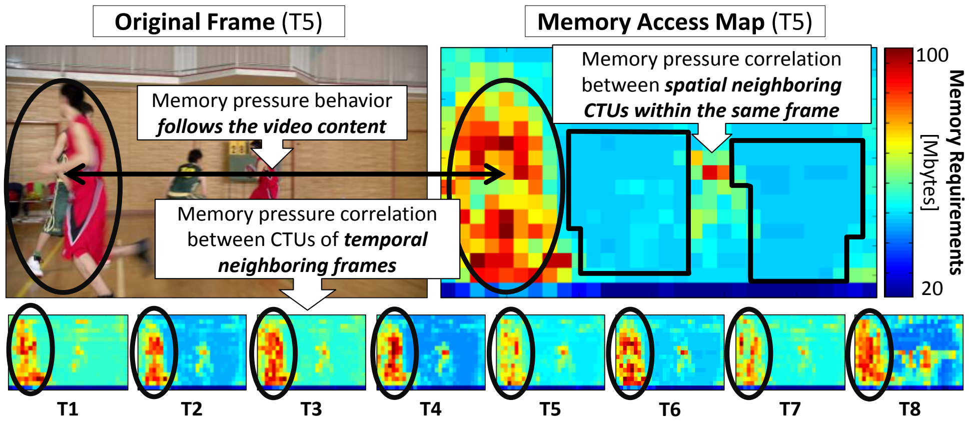

Original Frame (T5)
Memory Access Map (T5)
Memory pressure behavior follows the video content
Memory pressure correlation between CTUs of temporal neighboring frames
Memory pressure correlation between spatial neighboring CTUs within the same frame
100
20
Memory Requirements [Mbytes]
T1
T2
T3
T4
T5
T6
T7
T8

Figure 6.9: Video content and neighborhood correlation analysis for *BasketballDrive* test sequence.

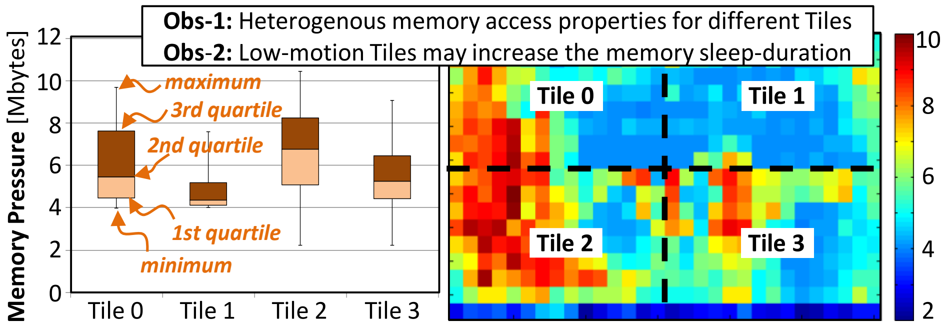


Figure 6.10: Intra-Tile memory pressure analysis for *BasketballDrive*.

6.2.2. *Overview of Memory Pressure Management Scheme*

Considering the already discussed analyzed memory access correlations, it was proposed a memory pressure management scheme for parallelized HEVC. The key is to leverage the memory access correlation *within* and *across* different Tiles (i.e. Intra- and Inter-Tile correlations; as discussed in previous section).

The proposed memory pressure management scheme is composed of the following three modules: (a) memory pressure prediction, (b) run-time statistics-based CTU memory classification, and (c) CTU re-scheduling for memory pressure balancing. The run-time memory monitoring unit feeds the statistics about the current memory requirements to the system.

The proposed memory balancing management is composed of the following parts:

- *Memory Pressure Prediction:* that leverages the monitored memory pressure of Tiles in the previously encoded CTUs in order to accurately predict the memory requirements for Tiles in the current frame.
- *Run-Time Statistics-Based CTU Memory Classification:* that dynamically adapts the parameters involved in the memory management scheme according to the predicted memory pressure statistics.
- *CTU Re-Scheduling:* the proposed strategy groups the CTUs of a Tile into variable-size groups (called CTU-groups). The size of the CTU-groups depends on the Tile-specific motion activity properties. Depending upon the predicted memory pressure, the CTU-groups are scheduled to closely meet the target pressure.

6.2.3. *Memory Pressure Prediction Algorithm*

As demonstrated in Section 6.2.1, highly correlated memory pressure may exist (1) among spatial neighboring CTUs (within the same frame); and (2) among CTUs of temporal neighboring frames. Therefore, based on the actual memory usage of previously processed CTUs (*ActualMem*), the prediction algorithm estimates the memory requirements of the CTUs in the current frame[10]. Figure 6.11 depicts an example of used CTU predictors in the current and reference frames. Four spatial predictors from the current frame and nine temporal predictors from each reference frame are selected as input to a weighted prediction.

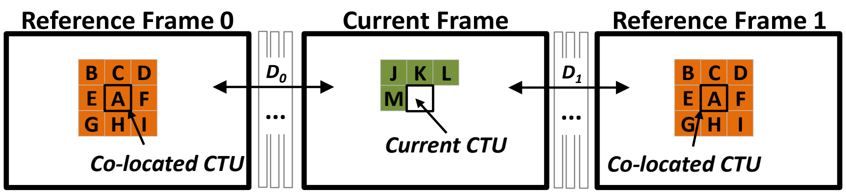


Figure 6.11: Example: spatial and temporal predictors selecting.

Equations (33) and (34) presents the spatial and temporal predictors selected for a given CTU: *Pred*$_{Temp}$ and *Pred*$_{Spatial}$, respectively. The letters *A-M* correspond to the spatial and temporal predictors depicted in Figure 6.11. As statistical parameters for the prediction, different weighting factors[11] was applied according to the spatial location of the predictor related to the current CTU position. Possible cases of CTU position are: center (α_C), horizontal/vertical (α_A), and diagonal (α_D). Equations (35)-(37) present the weighted prediction formula for predicting the memory pressure considering a given CTU. The weighting factors were statistically generated based on the memory access correlations of real video test sequences. First, the predicted memory pressure considering only the temporal references is estimated: *PredMem*$_{Temp}$ in Equation (35). Then, the spatial predictors are used to calculate the *PredMem*$_{Spatial}$, as in Equation (36). Finally, both spatial and temporal predictions are used to derive the predicted memory pressure for the given CTU: *PredMem* in Equation (37).

$$\mathrm{Pred_{Temp}}(\mathrm{F_{Ref}}) := \mathrm{WP}(\mathrm{ActualMem}(\mathrm{F_{Ref}}[\mathrm{A...I}]), [\alpha_{C,}\alpha_{A,}\alpha_D]) \tag{33}$$

$$\mathrm{Pred_{Spatial}} := \mathrm{WP}(\mathrm{ActualMem}(\mathrm{F_{Curr}}[\mathrm{J...M}]), [\alpha_{A,}\alpha_D]) \tag{34}$$

$$\mathrm{PredMem_{Temp}} = \sum_{\forall\, \mathrm{F_{Ref}}} \left\{ \left[\sum_{\mathrm{P_T} \in \mathrm{Pred_{Temp}}(\mathrm{F_{Ref}})} (\mathrm{P_T}) \right] * \frac{1}{\mathrm{D}[\mathrm{F_{Ref}}]} \right\} \tag{35}$$

$$\mathrm{PredMem_{Spatial}} = \sum_{\mathrm{P_S} \in \mathrm{Pred_{Spatial}}} (\mathrm{P_S}) \tag{36}$$

$$\mathrm{PredMem(CTU)} = \mathrm{WP}(\mathrm{PredMem_{Temp}}, \mathrm{PredMem_{Spatial}}, [\alpha_{S,}\alpha_T]) \tag{37}$$

When some predictors are unavailable (e.g., in case of CTUs at the frame boundaries) the weighted prediction is performed only with the available predictors. The predicted memory requirements of the CTUs need to be analyzed to classify each video frame, Tile and CTU-groups to characterize their memory access behavior.

6.2.4. *Run-Time Statistics-Based CTU Memory Classification*

As motivated in Section 4.2, in order to avoid the memory pressure imbalance problem of traditional raster scan order processing, the proposed scheme re-schedules the order of CTU

[10] A current frame refers to the frame being encoded at that moment.

[11] Statistically defined parameters using the experimental methodology described in Section 7.1: αC=0.5, αA=0.3, αD=0.2, αS=0.5, and αT=0.5.

evaluations for motion estimation. To achieve this, the scheme partitions the CTUs of a Tile into so-called *CTU-groups*, which are rectangular regions of CTUs such that, all CTUs of a given CTU-group are processed consequently; see an example in Figure 6.12. The goal is to assign CTUs with similar memory requirements/pressure into one group while balancing the overall memory pressure of Tiles.

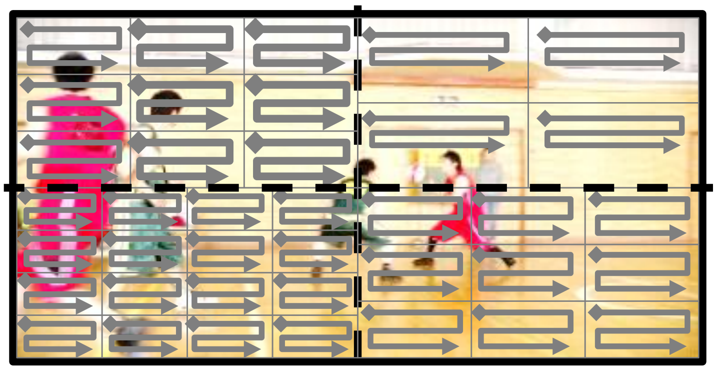


Figure 6.12: Example: CTU-groups division for re-scheduling.

The memory access distribution follows specific properties (i.e., motion and texture) of each video sequence. Hence, the video properties are used to decide the number of CTU-groups. The proposed scheme adapts the number of CTU-groups at frame level according to the predicted memory access distribution during each Tile processing. At first, a base number of groups is defined, N_B in Equation (38). It is based on the Probability Density Function (PDF) of the predicted memory pressures at frame level (μ_F is the average, σ_F is the standard deviation) and the average number of CTUs per Tile ($\mathrm{N}_{\mathrm{CTUPerTile}}$). Later on, it is defined the actual number of groups for each Tile (N_G in Equation (39)) by comparing the predicted memory access distribution of a given Tile (μ_T, σ_T) with that during the overall frame encoding. Tiles with spread memory pressure distributions are divided into more CTU-groups to enable fine-grained management (first clause of Equation (39)). The goal is to have a fine-grain management because it may have very diverse memory behaviors within a Tile. In contrast, Tiles with concentrated memory pressure distribution (second clause of Equation (39)) lead to few (but large-sized) CTU-groups as their texture and motion properties tend to be correlated inside such a Tile. The decision of having smaller CTU-groups must be carefully taken because the SPM data reuse among adjacent CTUs is not available between each CTU-group processing, causing efficiency loss in the SPM data management. Due to the CTU order inside one CTU-group (see Figure 6.12), the SPMs are more efficient for large-groups.

$$N_B = \left\lceil \left(\frac{\sigma_F}{\mu_F}\right) \mathrm{N}_{\mathrm{CTUPerTile}} \right\rceil \qquad (38)$$
$$\text{where: } \{\mu_F, \sigma_F\} = \mathrm{PDF}(\mathrm{PredMem(CTU)} \,|\, \forall\ \mathrm{CTU} \in \mathrm{Frame}\ F)$$

$$\mathrm{N_G(T)} = \begin{cases} \lceil \mathrm{N_B} + [(\sigma_T/\mu_T) - (\sigma_F/\mu_F)]\mathrm{N_{CTUPerTile}} \rceil & \text{if } (\sigma_T > \sigma_F) \\ \lceil \mathrm{N_B} - [(\sigma_F/\mu_F) - (\sigma_T/\mu_T)]\mathrm{N_{CTUPerTile}} \rceil & \text{otherwise} \end{cases} \qquad (39)$$
$$\text{Where: } \{\mu_T, \sigma_T\} = \mathrm{PDF}(\mathrm{PredMem(CTU)} \,|\, \forall\ \mathrm{CTU} \in \mathrm{Tile}\ T)$$

The predicted memory pressure distribution is used to classify the Tile in terms of motion property. By comparing the average behavior of each Tile-specific distribution to the overall frame distribution, Equation (40) defines three categories: *H-type* (high motion), *M-type* (medium motion), and *L-type* (low motion). Moreover, each CTU-group also has its own PDF (given in Equation (41)) that will be used for the re-scheduling decision during the memory pressure balancing.

$$C_{Tile}(T) = \begin{cases} (\mu_T \geq \mu_F + 0.5\sigma_F), (H)\ \text{High} \\ (\mu_F + 0.5\sigma_F > \mu_T > \mu_F - 0.5\sigma_F), (M)\ \text{Medium} \\ (\mu_F + 0.5\sigma_F > \mu_T), (L)\ \text{Low} \end{cases} \tag{40}$$

$$\{\mu_G, \sigma_G\} = \text{PDF}(\text{PredMem}(\text{CTU}) | \forall\ \text{CTU} \in \text{CTUgroup}\ G) \tag{41}$$

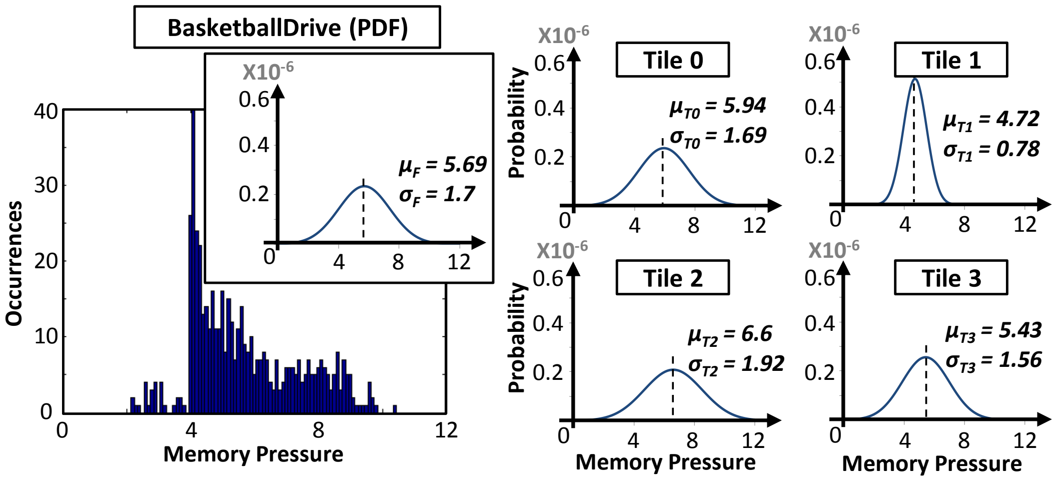


Figure 6.13: Memory pressure statistics for each Tile of the *BasketballDrive* test sequence (PDFs and histogram).

An Example: Figure 6.13 presents the run-time statistics of the predicted memory pressure of a frame in the HD1080 *BasketballDrive* video encoded with 4 Tiles. The N_{Base} value, which is only dependent on the overall frame statistics, is calculated using Equation (38), i.e. $N_{Base}=6$. Using Equation (39), the number of CTU-groups at is calculated: $N_G(0)=6$, $N_G(1)=2$, $N_G(2)=8$, $N_G(3)=4$. Using Equation (40), the motion classification of Tiles are: $C_{Tile}(0)$*=M-type*, $C_{Tile}(1)$*=L-type*, $C_{Tile}(2)$*=H-type*, and $C_{Tile}(3)$*=M-type*.

The above analysis and predicted memory pressure statistics are used by the CTU re-scheduling algorithm for memory pressure balancing and by the APMU of PrivL1 SPMs.

6.2.5. *CTU Re-Scheduling Scheme*

The goal of the CTU re-scheduling is to balance the accumulated memory pressure at the Tiles level, reducing the mean squared deviance (MSD) related to the average memory pressure (ideal case). Different number of CTU-groups lead to variable-sized groups, containing more or less CTUs within each Tile. The proposed scheme also classifies the Tiles according to the motion properties in three classifications C_{Tile}*={H-type, M-type, L-type}* using the Equation (40). Different Tile types will contribute in different ways for the accumulated balancing: *H-type* Tiles start by occupying the most part of the memory bandwidth, *M-type* Tiles contribute by median memory occupation, and the *L-type* Tiles aim to alleviate the memory pressure. The main task is to schedule the CTU-groups processing.

Figure 6.14 depicts the proposed CTU-groups scheduling functionality that is called at two points: (1) at the initial frame processing, when the decision about CTU-groups scheduling has not already taken, and (2) at the end of one CTU-group processing, when a new group must be scheduled. The call for this routine is performed at Tile-level, when the algorithm analyzes the current scenario to take the best decision. So, the input parameters are the *ID* of the Tile ($Tile_{ID}$) and the list of CTU-groups ($L_{CTU\text{-}Groups}$) that are inside the target Tile (*line 1*). For the first frame of the video, there are no temporal references for memory predictors, so the traditional raster scan order is performed (*lines 2-3*). If it is not the first frame, all memory predictions and run-time memory-related classifications are performed at the beginning of the frame processing. In case of the first CTU-group scheduling, the algorithm takes the motion Tile classification C_{Tile} into account to decide the CTU-group that will be next coded ($G_{ToBeCoded}$) (*lines 6-9*). Otherwise, the adaptive scheme analyzes the gap (*gapAccumPress*) between the current memory pressure (*currMemPress*) and an approximate average case prediction (*averageAccumPress* in *line 11*). So, the algorithm selects the CTU-group which has the predicted memory pressure and that has the best fit to the predicted gap (*lines 11-14*). After this decision, the CTU-group is removed from the non-coded groups list and the CTUs according are encoded according to the CTU-groups internal processing order depicted in Figure 6.12 (*line 17*).

```
1.  scheduleCTUGroup(Tile: $Tile_{ID}$, List of CTU-groups: $L_{CTU\text{-}Groups}$)
2.  If first frame Then
3.     $G_{ToBeCoded}$ := $L_{Groups}$.first();    //CTU-group equals to Tile
4.  Else   //not the first frame
5.     If frame start Then   //run-time statistical knowledge of Tiles
6.        $Tile_{class}$ := $C_{Tile}(Tile_{ID})$;  //Equation (40) – statistical classification
7.        Case($Tile_{Class}$ = L-type): $G_{ToBeCoded}$ := $L_{CTU\text{-}Groups}$.min();
8.        Case($Tile_{Class}$ = M-type): $G_{ToBeCoded}$ := $L_{CTU\text{-}Groups}$.median();
9.        Case($Tile_{Class}$ = H-type): $G_{ToBeCoded}$ := $L_{CTU\text{-}Groups}$.max();
10.    Else
11.       averageAccumPress:= $\sum_{T=0}^{N_{Tiles}}(\mu_T)$ ; //sum of av. pressures
12.       currAccumPress := getCurrentMemoryPressure();  //monitoring
13.       gapAccumPress := averageAccumPress – currAccumPress;
14.       $G_{ToBeCoded}$ := (G | $\mu_G$ has the best fit to gapAccPress);
15.    End If;
16. End If;
17. $L_{CTU\text{-}Groups}$ .remove($G_{ToBeCoded}$); encode($G_{ToBeCoded}$);
```

Figure 6.14: CTUs re-scheduling algorithm.

6.3 Lifetime-Aware Data Management

The MAMU implements a special treatment during a write access in L2 HyMs (PrivL2 and SharedL2), which is driven by the proposed lifetime-aware data management scheme. As already discussed, this scheme aims on improving STT-RAM cells endurance (i.e., extending the lifetime) to prevent Hy-SVM from wear-out errors. At the same time, as previously presented in Section 5.3, the design methodology leveraged application-specific properties to find the best possible optimization point of SRAM size within the HyMs, based on statistics of bit-toggling activities using real cases of input video sequences. The main goal of our lifetime-aware data management scheme is to provide an energy- and performance-efficient way of predicting the bit-toggling activity relying on the knowledge from the video content. Based on this, the incoming BU is directed either to STT-RAM or SRAM SPM, excepting by the two MSB of each sample that are always written in STT-RAM part.

To estimate as simple as possible the bit-toggling activity during a write operation, the bit-toggling key BT_KEY was defined as in Equation (42). This key is generated at the moment before the write operation and it consists in a set of wires getting the bits from b_5 to b_3 of specific samples resultant from the *downsample8* operation from the reference frame BU. The choice of these specific bits was taken due to the conclusion (2) from Figure 5.7. The *downsample8* function selects equally-spaced samples of a BU, reducing its representation resolution by 8 times. As example, considering an 8x8 BU, the prior 64-sample block is down-sampled to compose an 8-sample key[12]. Due to the spatial correlation between near pixels of a video frame, it is possible to discard many ones and still maintain the bit-toggling activity property. Thus, the BT_KEY will have 3x8=24 bits. The proposed estimated bit-toggling activity (EBT) calculates the number of bits that differs between the BT_KEY of the two involved BUs, as in Equation (43). This strategy is developed to avoid a complete read operation to fetch the entire

[12] To facilitate the understanding, the explanations onwards will consider BUs of 8x8 samples (BU_{Dim}=8).

BU to, just then, perform the bit-toggling activity evaluation. The *BT_KEY* of each stored reference BU is stored in a very fast special table, called Data Management Table (DMT). Besides the BT_KEY, the DMT also stores a flag indicating whether the corresponding reference BU is stored in the STT-RAM or SRAM array (called presence bit). A practical example is depicted in the Figure 6.17, where each one of the nine reference BUs has a DMT entry with its corresponding presence bit value. Thus, the DMT line consists in BL_{DMT} bits, as in Equation (44). Figure 6.15 presents 2D maps and histograms to show the high correlation between the actual bit-toggling activity (BT) and the estimated one (EBT). For a BU_{Size} equals to 8, the number of required bits is reduced to derive the toggling activity by ~22x using the EBT metric. The circuit to compute the number of bits that differ between two BT_KEY can be implemented with 24 1-bit XOR gates and a tree of 1-bit full adders, not representing neither energy nor performance significant penalty for the HyMs.

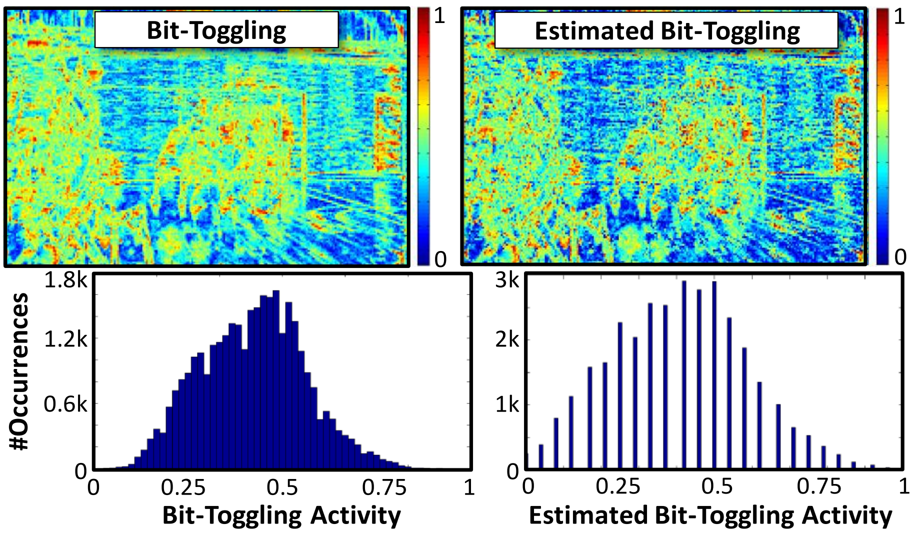


Figure 6.15: Statistical correlation between BT and EBT metrics.

$$\mathrm{BT\_KEY(BU)} = concat([\mathrm{b}_5..\mathrm{b}_3] \mid \mathrm{b} \in downsample8(\mathrm{BU})) \tag{42}$$

$$\mathrm{EBT(BU_0, BU_1)} = \mathrm{toggling\_bits(BT\_KEY(BU_0), BT\_KEY(BU_1))/24} \tag{43}$$

$$\mathrm{BL_{DMT}} = 1 + 24 = 25\mathrm{bits} \tag{44}$$

Figure 6.16 presents the data management steps for a HyM write operation. First, the BT_KEY for the BU that is being written is generated (*line 2*). Then, the BT_KEY of the already stored BU must be retrieved from the DMT (*line 4*) and the estimated activity α_{EBT} is then calculated (*line 5*). The α_{EBT} is then compared with the offline statistical defined threshold BT_{TH} (*line 6*). In the case that α_{EBT} is higher than BT_{TH}, the BU to be written is divided to be partially stored in the SRAM and STT-RAM *(lines 7-10*). In this case, SRAM dynamic energy is sacrificed to increase the STT-RAM cells lifetime. It is demonstrated in the experimental results discussion (Chapter 7) that this spent energy is very small compared to the overall savings provided by Hy-SVM. For α_{EBT} lower than BT_{TH}, the BU is completely written into the STT-RAM cells (*line 13*). The DMT is updated with the new $Bit_{Presence}$ (*lines 11-14*) and with the *BT_KEY* of the written BU (*line 16*).

The HyM read operation is much simpler than the write case, since no decision must be taken. The DMT is just accessed to get the presence bit and, depending on this, STT-RAM or SRAM/STT-RAM will be accessed. Finally, the data is forwarded to the requesting processing unit by the access management unit. As the data management unit increases the STT-RAM lifetime, the power management unit can power-gate unused cells with minimized risk of data re-fetching from external memory.

```
manageWrite(*Hybrid Memory:* HyM; *BasicUnit:* $BU_{ToBeWritten}$;
 *Basic Unit Positions:* x, y)
$Key_{ToBeWritten}$ := BT_KEY($BU_{ToBeWritten}$);   //generate key - Equation (42)
$Address_{Data}$ := *genPhysicalAddress*(x, y);   //calculate physical address
$Key_{ToBeReplaced}$ := DMT[$Address_{Data}$][23..0]; //get already stored key
$\alpha_{EBT}$ := EBT($Key_{ToBeWritten}$, $Key_{ToBeReplaced}$);   //estimate activity - Equation (43)
**If** ($\alpha_{EBT} > BT_{TH}$) **Then**        //high bit-toggling data
    $BU_{STT\text{-}RAM}$ := ((b[7..6] | ∀ b ∈ $BU_{ToBeWritten}$);   //2-bit split
    $BU_{SRAM}$ := (b[5..0] | ∀ b ∈ $BU_{ToBeWritten}$)        //6-bit split
    $HyM_{STT\text{-}RAM}$[$Address_{Data}$].*write*($BU_{STT\text{-}RAM}$);   **//STT-RAM write**
    $HyM_{SRAM}$.*write*(x, y, $BU_{SRAM}$);        **//SRAM write**
    DMT[$A_{Data}$][25].*write*('0');              //DMT update – Presence Bit
**Else**                                  //low bit-toggling data
    $HyM_{STT\text{-}RAM}$[$Address_{Data}$].*write*($BU_{ToBeWritten}$);   **//STT-RAM write**
    DMT[$Address_{Data}$][25].*write*('1');       //DMT update – Presence Bit
**End If;**
DMT[$A_{Data}$][23..0].*write*($Key_{ToBeWritten}$);              //DMT update
```

Figure 6.16: Data management for a HyM write operation.

To exemplify the lifetime-aware data management actuation, the logical organization of HyMs is demonstrated in Figure 6.17, where a set of 3x3 reference BUs are taken as example. As decision from the data management unit, the BUs (0,0), (1,1) and (2,2) are considered to provoke high bit-toggling activity and must be partially stored in the SRAM SPM; while the remaining BUs are completely stored in the STT-RAM SPM. For data management purposes, a Data Management Table (DMT in and Figure 6.17) was also designed.

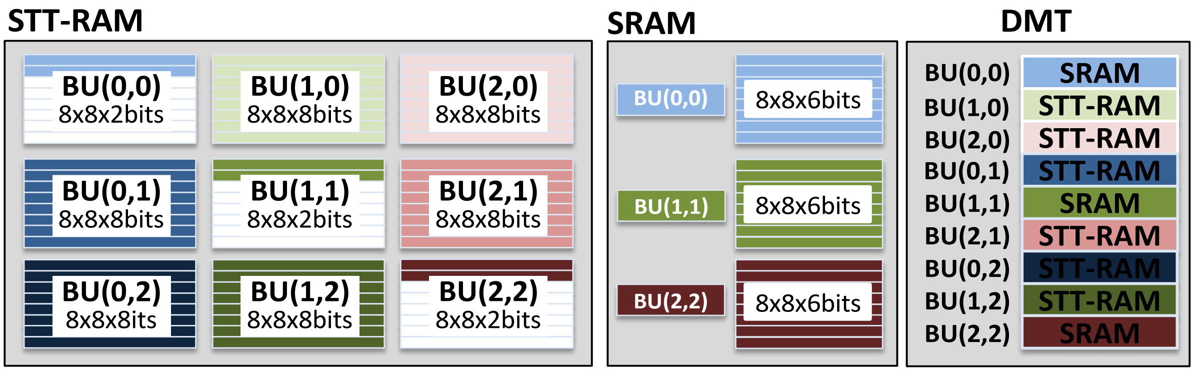


Figure 6.17: Example of a HyM data assignment.

6.4 On-Chip Memory Management Units

Figure 6.18 presents the block diagram of the proposed on-chip Hy-SVM management units. As example, Figure 6.18 illustrates hardware details of a 4-Tile HEVC case study, which has one horizontal (*Hor*) and one vertical (*Ver*) Tiles boundary. The main goal is to rely on accurate overlap prediction to employ energy-efficient memory access and power management to designed SPMs in Hy-SVM architecture. As previously explained, the overlap prediction leverages past overlap formations of past MEs, which are kept stored in the MOT. The proposed memory monitoring is the unit responsible for capturing the inter-Tiles redundant memory access behavior. The prediction unit stores the predicted overlap of a corresponding Tiles boundary in the POTs.

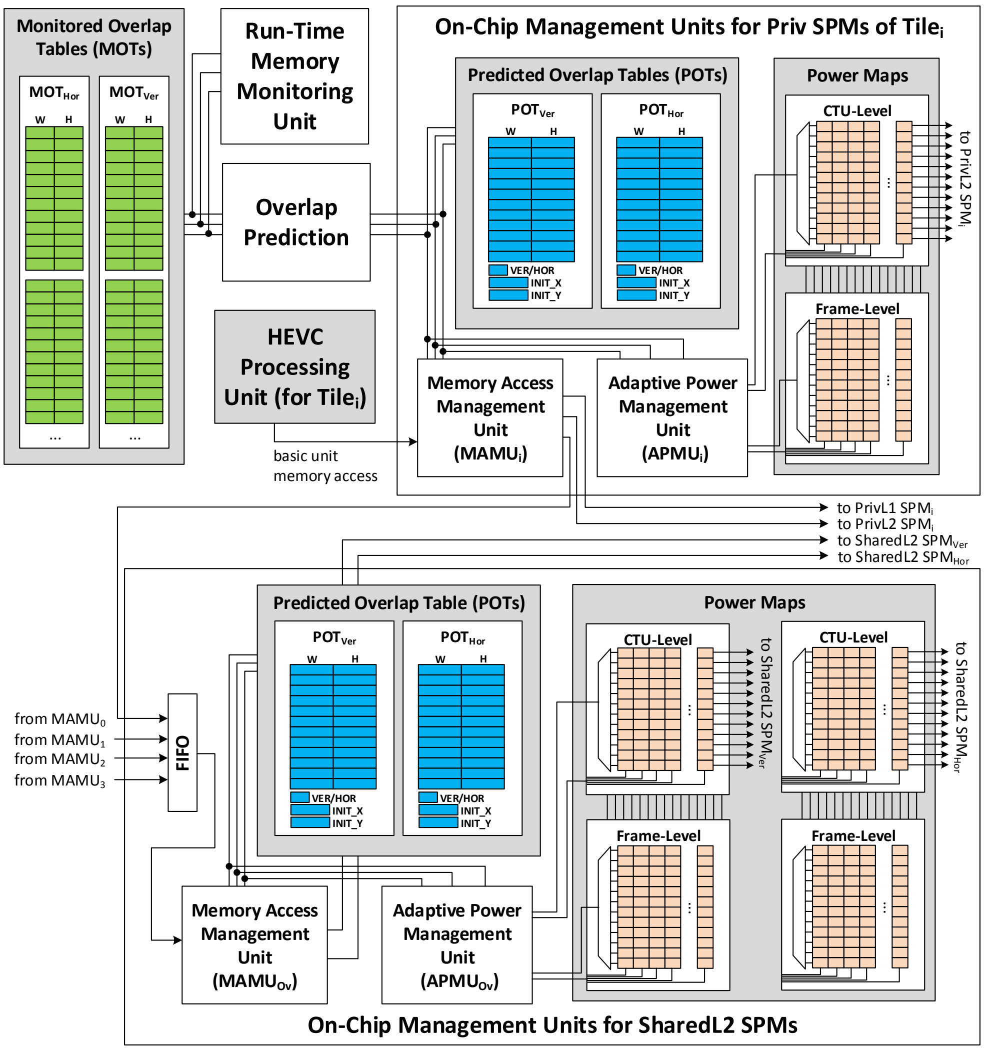
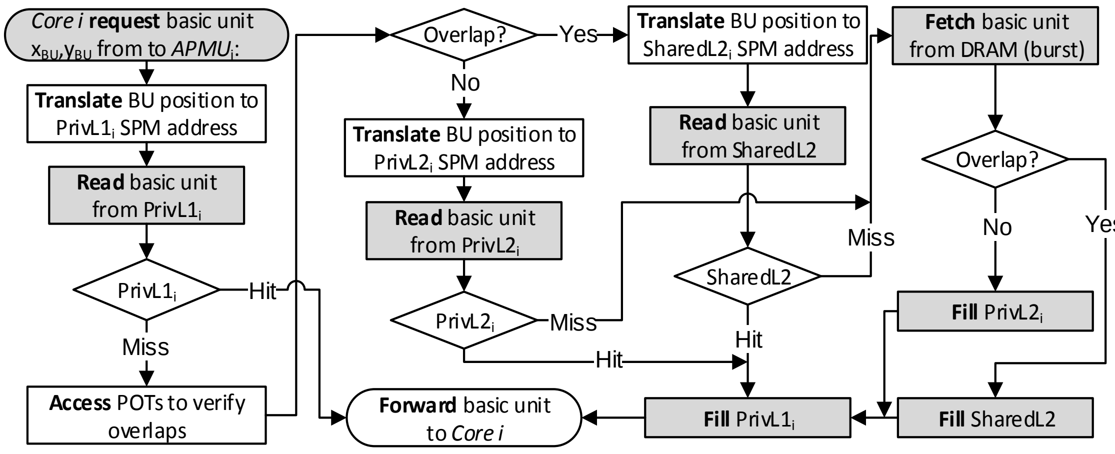

Monitored Overlap Tables (MOTs)
MOT_{Hor}
MOT_{Ver}
Run-Time Memory Monitoring Unit
Overlap Prediction
HEVC Processing Unit (for $Tile_i$)
basic unit memory access
On-Chip Management Units for Priv SPMs of $Tile_i$
Predicted Overlap Tables (POTs)
POT_{Ver}
POT_{Hor}
VER/HOR
INIT_X
INIT_Y
Power Maps
CTU-Level
Frame-Level
to PrivL2 SPM_i
Memory Access Management Unit ($MAMU_i$)
Adaptive Power Management Unit ($APMU_i$)
to PrivL1 SPM_i
to PrivL2 SPM_i
to SharedL2 SPM_{Ver}
to SharedL2 SPM_{Hor}
from $MAMU_0$
from $MAMU_1$
from $MAMU_2$
from $MAMU_3$
FIFO
Predicted Overlap Table (POTs)
to SharedL2 SPM_{Ver}
to SharedL2 SPM_{Hor}
Memory Access Management Unit ($MAMU_{Ov}$)
Adaptive Power Management Unit ($APMU_{Ov}$)
On-Chip Management Units for SharedL2 SPMs

Figure 6.18: Block diagram of our on-chip Hy-SVM management units integrated to the run-time overlap prediction and memory monitoring units.

Each HEVC processing unit *i* has an associated instance of memory access management unit ($MAMU_i$) and of adaptive power management unit ($APMU_i$). These modules utilize the predicted overlap, available in the POTs, to provide energy-efficient management of the PrivL1 and PrivL2 SPMs for the processing unit *i*. The $MAMU_i$ receives a memory access request and, based on a read/write policy, translates the address and forwards the operation to either PrivL1 or PrivL2. Further, if the incoming access is related to a basic unit inside any predicted overlap, the request is forwarded to a $MAMU_{Ov}$, responsible of managing the SharedL2 SPMs accesses. As the private MAMUs require the knowledge of the predicted overlaps, each unit has an instance of the POTs (as in Figure 6.18). The $APMU_i$ analyzes the POT content and HEVC parameters to build the power maps for the PrivL2 SPM_i. The power maps are directly connected to the sleep-transistors that control the power state of each sector of STT-RAM array. Additionally, a specific $APMU_{Ov}$ module manages the power gating operation of SharedL2 SPMs. Details regarding MAMUs and APMUs implemented schemes are given as follows.

6.4.1. *Memory Access Power Management Unit (MAMU)*

Our MAMU implements a read/write policy (see flowchart of Figure 6.19) that takes advantage from the Tiles overlap to increase the data-reuse of the reference frame samples. When HEVC processing unit *i* requests a BU of positions x_{BU} and y_{BU} to Hy-SVM, as first step, the MAMU translates the BU frame positions to PrivL1 SPM address space. Then, it performs a PrivL1 SPM$_i$ access to check for hit/miss occurrence. In case of a hit, the BU is forwarded to the processing unit. Otherwise, a miss leads to the access of L2 level SPMs. At this point, the MAMU checks along with the predicted overlaps if the requested BU belongs to one Tiles overlapping region. Assuming that the data is inside an overlap related to the Tiles intersection *TB*, the corresponding SharedL2 SPM$_{TB}$ is then accessed. In this case, inter-Tiles data reuse is exploited, since the processing of all Tiles that share the Tiles boundary *TB* may request the same data. For non-overlapping regions, the PrivL2 SPM$_i$ is accessed, leading to intra-Tile data reuse. Note that for each data request, either a ShreadL2 SPM or a PrivL2 SPM is accessed. If a L2 hit is verified, the data is forwarded to the processing unit and the PrivL1 SPM$_i$ is filled with the requested BU. In case of a L2 miss, the BU must be fetched from the external memory and written to either PrivL2$_i$ or SharedL2$_{TB}$ SPM (depending on the predicted overlap) and the PrivL1 SPM$_i$. After that, the data is forwarded to the Tiles-specific HEVC processing unit.



Core i request basic unit x_BU,y_BU from to APMU_i:
Translate BU position to PrivL1_i SPM address
Read basic unit from PrivL1_i
PrivL1_i
Hit
Miss
Access POTs to verify overlaps
Overlap?
Yes
No
Translate BU position to PrivL2_i SPM address
Read basic unit from PrivL2_i
PrivL2_i
Miss
Hit
Forward basic unit to Core i
Translate BU position to SharedL2_i SPM address
Read basic unit from SharedL2
SharedL2
Miss
Hit
Fill PrivL1_i
Fetch basic unit from DRAM (burst)
Overlap?
No
Yes
Fill PrivL2_i
Fill SharedL2

Figure 6.19: Flow of our memory access management unit with read/write policy.

Case-study example: Figure 6.20 depicts an example of data migration for our read/write policy in four different cases, considering a 2-Tile HEVC encoding system.

(a) In the beginning, the on-chip SPMs are empty and each request will lead to external memory fetching (L1 and L2 misses). Figure 6.20 shows that predicted overlap is analyzed to determine whether the reference BU is stored in the PrivL2 SPM$_i$ or in the SharedL2 SPM$_{TB}$. The PrivL1 SPM$_i$ is always filled with the fetched data. During the frame encoding, due to the intra-Tile (PrivL1 and PrivL2 SPMs) and inter-Tiles (SharedL2 SPMs) reused data, more hits occur and even less external memory communication is needed.

(b) The second case of Figure 6.20 depicts Tiles-centering CTUs processing where only the PrivL2 SPMs are accessed (only intra-Tile data reuse). Note that all accesses inside this case are directed to reference frame BUs outside the predicted overlap. We can also observe some PrivL1 SPMs hits, which avoid L2 SPMs accessing and external memory fetching.

(c) The third case illustrates accesses from CTUs located close to the Tiles boundary. In this scenario, L2 memory hits are verified for both PrivL2 and SharedL2 SPMs (i.e., combined intra- and inter-Tiles data reuse). This case represents the best energy efficiency when requiring L2 level access.

(d) The last scenario of Figure 6.20 presents the best case of energy efficiency, where all memory accesses result on PrivL1 hits.

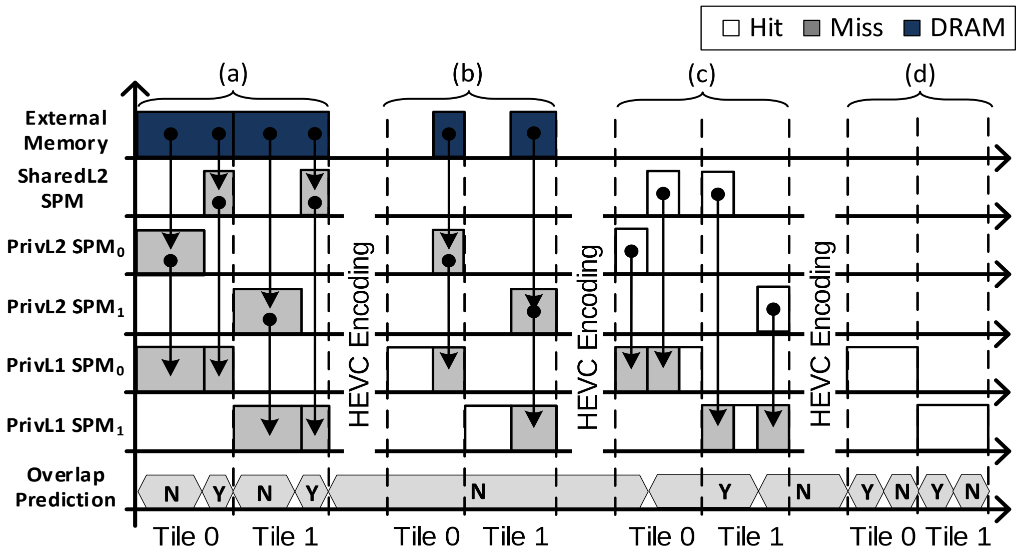


Figure 6.20: An example of data interaction for a 2-Tile HEVC encoding.

6.4.2. *Adaptive Power Management Unit (APMU)*

6.4.2.1. Adaptive Power Management of L1 SPMs

The APMU for PrivL1 SPMs, implemented with SRAM cells, monitors each private SPM usage to capture the current video motion property and power-gate less-likely used sectors to save on-chip static energy. At this level, it was considered a memory technology with three power states: *ON*, *DR (Data Retentive*) and *OFF*.

Evaluations of Figure 6.21illustrate that it is possible to *increase the potential of long sleep durations once the memory pressure is balanced*. For example, Figure 6.21a presents the SPM usage for the processing unit 1 when encoding the *BasketballDrive* sequence. The SPM usage (SPM_{Usage}) calculated as the percentage of accessed SPM memory positions (measured by the memory monitoring unit) during one ME operation (*AccSPM*), see Equation (45). As shown in Figure 6.21b, the SPM usage for the entire CTU can be determined as the Probability Density Function of the SW_{Usage} values of all blocks within the CTU, see Equation (46).

$$\mathrm{SPM}_{\mathrm{Usage}}(\mathrm{CU}_{\mathrm{ID}}) = \mathrm{AccSPM}(\mathrm{CU}_{\mathrm{ID}})/\mathrm{S}_{\mathrm{SPM}} \tag{45}$$

$$\begin{gathered}\mathrm{SPM}_{\mathrm{UsagePDF}}(\mathrm{CTU}) = \{\sigma_{\mathrm{SPM}}, \mu_{\mathrm{SPM}}\} = \\ \mathrm{PDF}(\mathrm{SPM}_{\mathrm{Usage}}(\mathrm{Block}_{\mathrm{Node}}) | \forall\, \mathrm{Block}_{\mathrm{Node}} \in \mathrm{CTU})\end{gathered} \tag{46}$$

At the beginning of a CTU encoding, the algorithm predicts the number of the memory sectors that can be put into different power state (i.e., N_{ON}, N_{DR} and N_{OFF}). As basis for this prediction, it was analyzed (1) the actual search window usage for previously processed CTUs (e.g., CTU_{ID-3}, CTU_{ID-2}, and CTU_{ID-1}); (2) the predicted usage for the current CTU_{ID} and the next CTU_{ID+1} and CTU_{ID+2}. The goal is to have the knowledge of the past, present and predicted future memory requirements to increase the on-chip static energy savings while minimizing the overhead for memory sectors waking-up. Figure 6.21b presents an example of SPM usage PDFs and the corresponding power states assignment.

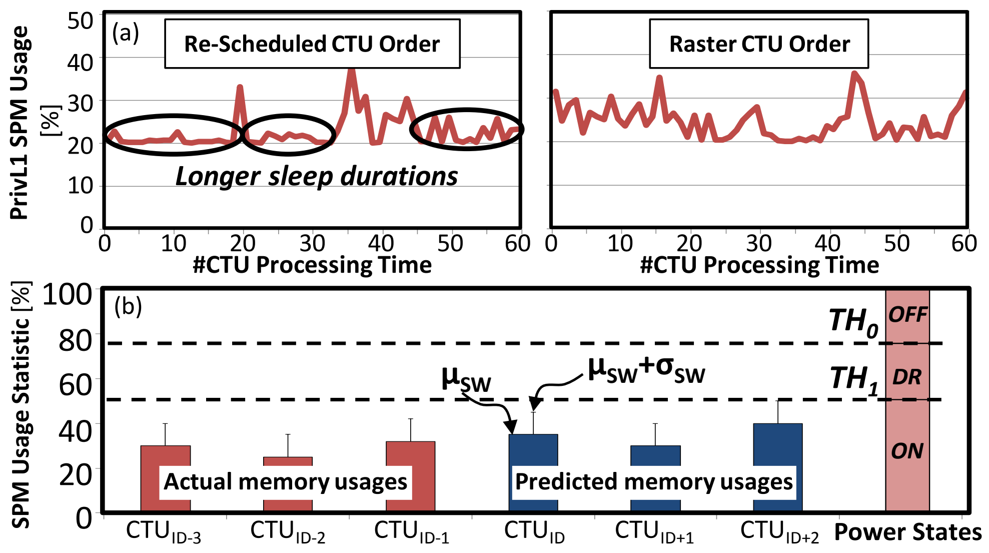


Figure 6.21: (a) Increased memory pressure correlation;
(b) power states determination based on the SPM usage PDFs.

Figure 6.22 presents the APMU policy for PrivL1 SPMs. The actual SPM usage PDFs of the past CTUs ($List_{ActualSPMUsagePDF}$) and the next predicted SPM usage PDFs ($List_{PredSPMUsagePDF}$) are used to determine the power states of the SPM sectors (*lines* 3-5). As in Figure 6.21, the scheme defines two thresholds (TH_0 and TH_1) based on the average and standard deviation of all cited PDFs (*lines 6-7*). Afterwards, the SPM sectors corresponding to each power states are derived (*lines 8-9*). The physical assignment of the power states to the SPM cells is performed at the beginning of every block processing within a CTU (*lines 10-13*). In the case of data retransmission is required (SPM cells wake-up from the *OFF* state), the control unit inserts stalls in the execution pipeline. Still, this penalty implies a negligible energy/performance overhead since in the experiments the worst-case scenario is observed <0.2% times.

```
1.  managePowerSPM (Tile: $Tile_{ID}$, CTU: $CTU_{ID}$)
2.  $PowerMap_{SPM} := \Phi$; $N_{ON} := 0$; $N_{DR} := 0$; $N_{OFF} := 0$;
3.  $List_{ActualSPMUsagePDF} := (SPM_{UsagePDF}\ (ActualMem(CTU_{ID})\ |\ ID \in \{-3..-1\}))$;
4.  $List_{PredSPMUsagePDF} := (SPM_{UsagePDF}\ (PredMem(CTU_{ID})\ |\ ID \in \{0..2\}))$;
5.  $List_{PDF}$.append($List_{ActualSPMUsagePDF}$, $List_{PredSPMUsagePDF}$);
6.  $TH0 := max(\mu_{SPM}+3.\sigma_{SPM}\ |\ (\mu_{SPM},\sigma_{SPM}) \in \{List_{PDF}\})$;   //TH's definition
7.  $TH1 := max(\mu_{SPM}+1.\ \sigma_{SPM}\ |\ (\mu_{SPM},\sigma_{SPM}) \in \{\ List_{PDF}\})$;
8.  $N_{OFF} := (1-TH0)*N_{Sec}$; $N_{DR} := (TH0-TH1)\ *N_{Sec}$; $N_{ON} := TH1\ *N_{Sec}$;
9.  $PowerMap_{SPM}$.assignPowerStates($N_{ON}$, $N_{DR}$, $N_{OFF}$);
10. For all Block $\in CTU_{ID}$
11.     SPM[$Tile_{ID}$].powerGate($PowerMap_{SPM}$);   //apply power gating
12.     encode(Block);
13. End For;
```

Figure 6.22: Adaptive on-chip power management of SPMs.

6.4.2.1. Adaptive Power Management of L2 SPMs

Our APMU leverages the predicted overlaps and the search limits of current CTUs to further reduce the static energy consumption of Hy-SVM. The SPMs in the L2 level of Hy-SVM (implemented as STT-RAM) were designed to operate in two power states: *ON* ($V_{ON}=V_{DD}$ volts) and *OFF* ($V_{OFF}=0$ volts). Due to the non-volatility characteristic of STT-RAM, the data is kept stored in the memory cell even when *OFF* state is assigned (differently from SRAM cells). Further, L2 SPMs are typically significantly larger than L1 SPMs, leading to higher energy consumption. In doing so, our APMU concentrates effort in L2 SPMs, resulting in a great impact in the Hy-SVM overall on-chip energy (as demonstrated in Chapter 7).

In Hy-SVM, the L2 Level can store an entire reference frame, exploiting STT-RAM reduced leakage power and providing high intra-Tile and inter-Tiles data reuse, leading to reduced external memory energy. Besides, the ME required memory accesses for all CUs within a CTU is limited to a search window, which represents a small portion of the whole reference frame. Our APMU scheme relies on estimates the search limits for the entire CTU processing, which combines the search window of the ME for all CUs. The CTU search limits are defined as a squared region of BUs of $\lceil (\mathrm{SL_{Dim}} \times \mathrm{SL_{Dim}})/\mathrm{BU_{Size}} \rceil$ size, where $\mathrm{SL_{Dim}} = \mathrm{CTU_{Size}} + \mathrm{SW_{Dim}}$.

Figure 6.23 and Figure 6.24 present the flowcharts for the APMU when managing PrivL2 and SharedL2 STT-RAM SPMs within the HyMs. Our schemes act at the beginning of frame processing (frame level; first flowcharts), as well as before the encoding of each CTU (CTU level; second flowcharts). Note that CTU-level step of APMU may be executed at different time stamps for each Tile processing, since it depends on the execution time spent to encode CTUs with distinct properties in different processing units.

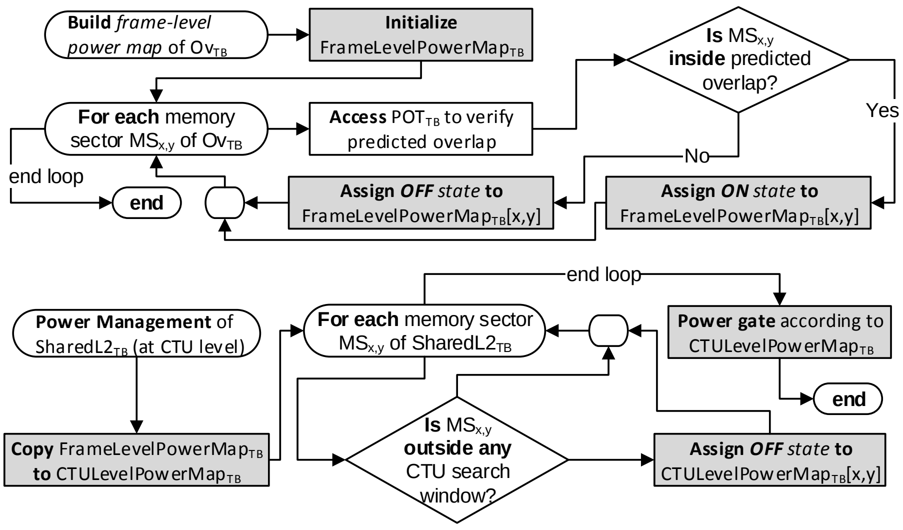

Build frame-level power map of Ov_TB
Initialize FrameLevelPowerMap_TB
Is MS_x,y inside predicted overlap?
Yes
No
For each memory sector MS_x,y of Ov_TB
Access POT_TB to verify predicted overlap
end loop
end
Assign OFF state to FrameLevelPowerMap_TB[x,y]
Assign ON state to FrameLevelPowerMap_TB[x,y]
end loop
Power Management of SharedL2_TB (at CTU level)
For each memory sector MS_x,y of SharedL2_TB
Power gate according to CTULevelPowerMap_TB
end
Copy FrameLevelPowerMap_TB to CTULevelPowerMap_TB
Is MS_x,y outside any CTU search window?
Assign OFF state to CTULevelPowerMap_TB[x,y]

Figure 6.23: Flow of the proposed adaptive power management unit of SharedL2 SPMs.

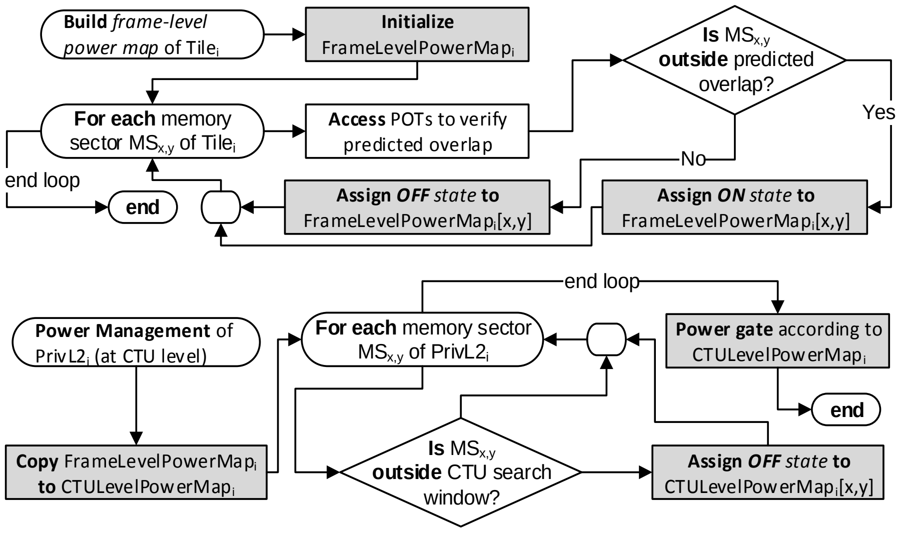
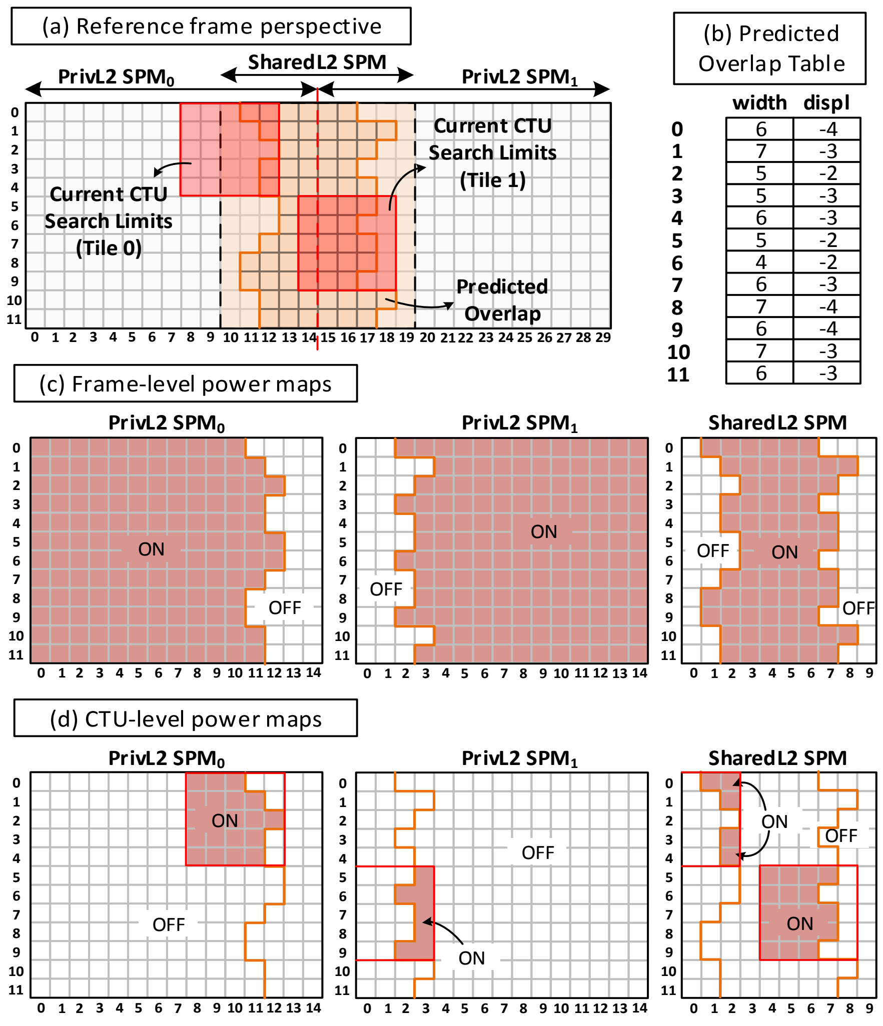

Build frame-level power map of Tile_i
Initialize FrameLevelPowerMap_i
Is MS_x,y outside predicted overlap?
Yes
No
For each memory sector MS_x,y of Tile_i
Access POTs to verify predicted overlap
end loop
end
Assign OFF state to FrameLevelPowerMap_i[x,y]
Assign ON state to FrameLevelPowerMap_i[x,y]
end loop
Power Management of PrivL2_i (at CTU level)
For each memory sector MS_x,y of PrivL2_i
Power gate according to CTULevelPowerMap_i
end
Copy FrameLevelPowerMap_i to CTULevelPowerMap_i
Is MS_x,y outside CTU search window?
Assign OFF state to CTULevelPowerMap_i[x,y]

Figure 6.24: Flow of the proposed adaptive power management unit of PrivL2 SPMs.

For an ease explanation of our concepts, an example of the APMU operation is illustrated in Figure 6.25. In the first part, the adopted 2-Tile HEVC encoding scenario is represented at reference frame perspective (Figure 6.25a). The current CTU search limits of Tile 0 and Tile 1 are depicted, as well as the predicted overlap (stored in the POT using the proposed representation, as in in Figure 6.25b).

At the beginning of frame processing, the APMU builds one frame-level power map for each L2 SPMs. For the PrivL2 SPMs, the memory sectors outside the predicted overlap have associated power state set as *ON*. Otherwise, the *OFF* state is assigned. As previously discussed, L2 accesses are directed either to a PrivL2 SPM or a SharedL2 SPM, depending on the predicted formation of the overlap. The APMU frame-level power map building process is the opposite for the SharedL2 SPMs, as presented in Figure 6.25c. The frame-level power maps are not directly assigned to the sleep-transistors of the L2 SPMs, being start points to compose the CTU-level power maps. At CTU level, our scheme checks the frame-level power map against the search limits of the current CTUs. Note that the PrivL2 SPMs must be checked against the search limits of its corresponding Tile processing, while the SharedL2 SPMs must be analyzed considering the search limits of all HEVC processing units. The SPM sectors outside the search limits are set as *OFF* state, resulting on CTU-level power maps of Figure 6.25d. By assigning *ON* state for the STT-RAM sectors inside the CTU search limit, we ensure long sleep durations during one entire CTU processing.



	width	displ
0	6	-4
1	7	-3
2	5	-2
3	5	-3
4	6	-3
5	5	-2
6	4	-2
7	6	-3
8	7	-4
9	6	-4
10	7	-3
11	6	-3

Figure 6.25: Example: adaptive power management of STT-RAM L2 SPMs for a 2-Tile scenario.

7. EXPERIMENTAL RESULTS AND DISCUSSIONS

This chapter discusses the experimental results and compares them to related works. At first, detailed experimental setup is presented in **Section 7.1**, organized in terms of adopted video coding evaluation methodology, memory simulation methodology and on- and off-chip memory power models. The discussion of our experimental results is composed of several parts. Initially, (in **Section 7.2**) a detailed energy consumption profiling traced, which separately evaluates off- and on-chip energy savings of Hy-SVM, as well as combining both parts to compose an overall scenario. After that, (in **Section 7.3**) the proposed management units regarding the overlap exploitation are evaluated in terms of prediction accuracy. Then, (in **Section 7.4**) an external memory communication evaluation is performed to derive the improvements of the proposed memory pressure management in terms of balanced data transmission. The lifetime-aware data management scheme contributions on improving STT-RAM cells lifetime is evaluated in **Section 7.5**. Finally, an overhead analysis of implementing the proposed management units along with Hy-SVM is discussed in **Section 7.6**.

7.1 Experimental Setup

7.1.1. *Video Coding Evaluation Methodology*

The experimental analyses related to the video coding are based on the recommended HEVC common test conditions (ISO/IEC-JCT1/SC29/WG11, 2012) using, primarily, the HEVC test model (HM 11.0) (ISO/IEC-JCT1/SC29/WG11, 2013b). The HM is the official reference software that is completely compliant with the HEVC standard, containing all state-of-the-art coding tools provide by the standard. Experiments were executed for different parallelization scenarios (multiple Tiles, which each Tile executes on a dedicated processing unit). The experiments include the exploitation of a wide range search window dimensions: 128x128, 192x192, and 256x256. To provide different video content characteristics to properly evaluate the efficiency of the proposed memory architectures, an extensive set of test video sequences with distinct properties were evaluated: *BasketballDrive (BDrive), Beauty, Bosphorus, BQTerrace (BQTerr), Cactus, Kimono, ParkScene (PScene), ReadySteadyGo (RSGo), ShakeNDry (SNDry),* and *YachtRide (YRide)* - HD1080 (1920x1080 pixels); *NebutaFestival (NFest), PeopleOnStreet (People), SteamLocomotiveTrain (SLTrain)* and *Traffic* - 2K (2560x1600 pixels). Other important encoder specifications are: GOP=8, QP=32, CABAC, Random Access configuration, and TZ Search algorithm.

7.1.2. *Memory Simulation Methodology*

To capture the memory access profiling of the used HEVC encoding applications (considering multiple level general-purpose cache memories), it was used the *callgrind* and *cachegrind* tools of *valgrind* simulator (VALGRIND DEVELOPERS, 2017).

To simulate the access dynamics of the proposed memory architectures, several custom simulators were developed. As main input of these simulators, memory accesses traces from the used HEVC encoding applications were extracted. For each scheme inside the proposed memory architectures, a software-based modeling was inserted in the developed simulators. To provide the best possible accuracy of the memory energy estimation in terms of comparison with related works, the used simulation environment incorporates widely used memory models for all adopted technologies: SRAM and STT-RAM in the on-chip memory perspective; and DRAM for the external memory. The developed simulation environment was built under an open-source license and it is available for usage in the used project repository[13].

[13] Custom simulators developed in this work are available at https://bitbucket.org/felsamps/

7.1.3. *Off- and On-Chip Power Models*

To perform the electrical characterization for the adopted 32nm SRAM memory arrays, the CACTI 6.5 tool was used (HP LABS, 2008). Regarding the STT-RAM based on-chip memories, the evaluations consider the generated parameters of NVSim tool (XIANGYU DONG et al., 2012). The SPM design parameters resultant from the proposed Hy-SVM design methodology for the evaluated video coding scenarios are summarized in Appendix A. Furthermore, based on the designed SPM organizations, Appendix A presents the power and latency characterization, which was performed using the above presented tools (CACTI for SRAM and NVSim for STT-RAM).

For the main memory, it was adopted one 4-Gbit Low-Power DDR2 module (MICRON TECHNOLOGY INC., 2011). The energy components of a LPDDR2 were estimated using the main memory accesses of each application and the technology data from Micron (MICRON TECHNOLOGY INC., 2005, 2005). The main specifications are: Vdd=1.2V, Freq=533MHz, W_{Size}=32 bits, P_{Size}=512B, N_{Rows}=16K and $N_{Columns}$=2K. The total energy is derived by the composition of six components: page activation energy (E_{ACT}), write energy (E_{WR}), read energy (E_{RD}), I/O pins energy (E_{DQ}), refresh energy (E_{REF}) and standby energy (E_{STBY}). In the experimental analysis along this paper, the assumption is that the memory will always operate in the *ACT* state and the standby energy will be equivalent to the P_{ACT_STBY} component.

All control-flow hardware blocks were synthesized using Cadence synthesis flow using ST 65nm standard-cells library.

7.1.4. *Comparison-Purpose Baseline Hy-SVM Variations*

Besides the comparison with related works, we implemented baseline variations of Hy-SVM to measure the efficiency of our design decisions. Hence, three alternative comparison-purpose architectures were evaluated in our experiments:

- *All-SRAM*: adopts the SRAM technology for all SPMs in Hy-SVM architecture. The goal is to evaluate the benefits (static energy consumption) and shortcomings (poor write efficiency) of using STT-RAM in L2 SPMs;
- *Priv-Only*: avoids the usage of Shared SPMs, thus exploiting only intra-Tile data reuse. The main purpose is to evaluate the impacts of SharedL2 SPMs, as well as the overlap management efficiency;
- *No-APMU*: avoids the proposed power management over L2 SPMs. The goal is to evaluate the contributions of APMU in the on-chip energy savings of Hy-SVM;
- *No-DM*: avoids the lifetime-aware data management and the SRAM SPMs within L2 HyMs. With this variation, the goal is to evaluate the advantages (increased STT-RAM lifetime) and disadvantages (energy consumption overhead) of implementing this endurance optimization technique inside Hy-SVM architecture.

7.2 Energy Efficiency Evaluation

7.2.1. *Off-Chip Energy Results*

In order to analyze different parameters and their impacts in the Hy-SVM off-chip energy savings, the experimental results are organized in three different (but connected) evaluations.

7.2.1.1. Analysis-1: Parallelism and HEVC Prediction Structure

Figure 7.1 depicts the first evaluation of off-chip energy savings of Hy-SVM compared to related implementations. The analysis presents the savings for 2-, 4-, 8- and 16-Tile with Low

Delay and Random Access HEVC configurations. To observe the behavior for different search window sizes, the energy results were separated for 128x128, 192x192 and 256x256 dimensions. In this first evaluation, the presented savings were calculated as the average scenario of all tested video sequences.

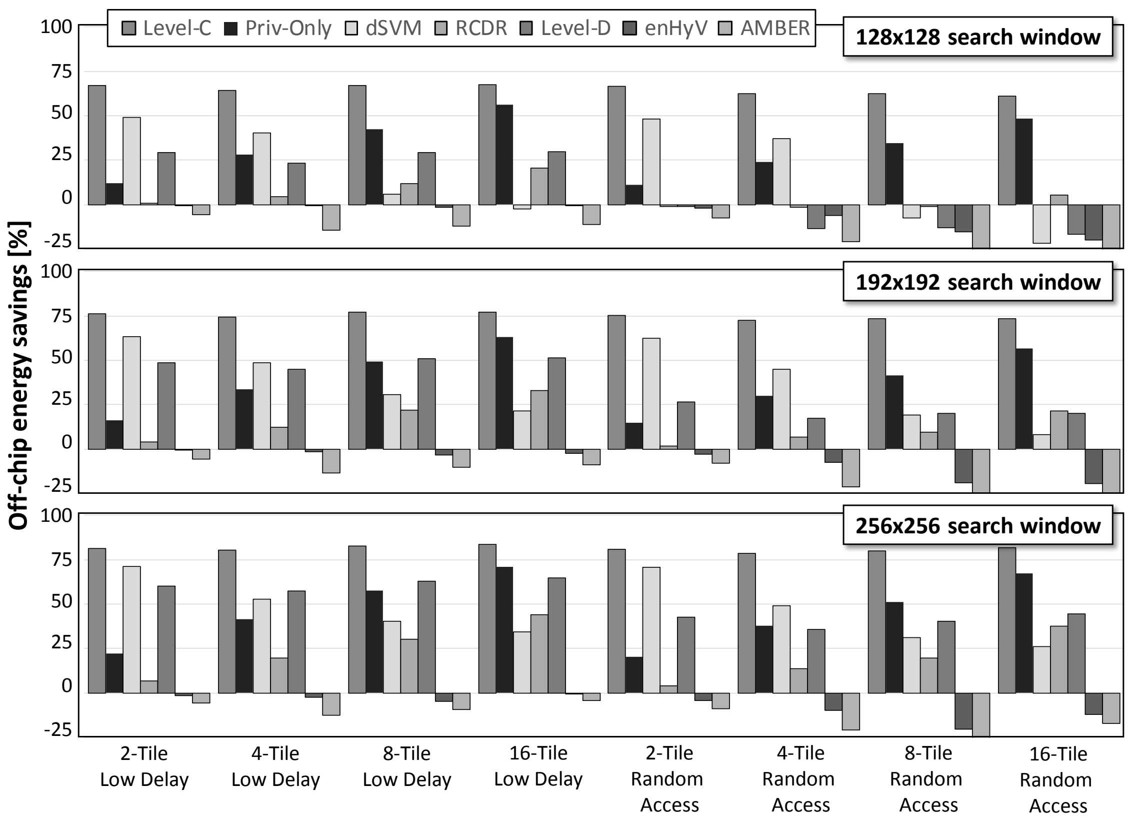


Figure 7.1: Off-chip energy savings of Hy-SVM compared to related works for increased number of Tiles and for Random Access and Low Delay HEVC configurations (average scenario of tested video sequences).

Level-C (CHEN et al., 2006) represents the upper bound results, since it only exploits intra-Tile data reuse in search window level. Hy-SVM can reduce up to 66%, 76% and 82% the off-chip energy consumption compared to Level-C, when considering 128x128, 192x192 and 256x256 search windows, respectively. In this case, the energy savings remain stable when increased number of Tiles are used. Still, the adopted HEVC prediction configuration does not affect the achieved gains.

Regarding dSVM (SAMPAIO et al., 2014a) architecture, which also exploits joint intra-Tile and inter-Tiles data reuse, our Hy-SVM can achieve savings of up to 49%, 63% and 71% for the tested search window sizes. The use of STT-RAM allows Hy-SVM energy-efficient on-chip storage of entire reference frame samples. Hence, it strongly impacts the external memory communication, as dSVM adopts intra-Tile data reuse in search window level. Since dSVM strongly increases the on-chip memory when increased number of Tiles is adopted, the Hy-SVM savings is reduced for increased parallelism. In some extreme cases, dSVM has improved Hy-SVM off-chip energy efficiency, like for 16-Tile scenario using 128x128 search window size. For the other cases, Hy-SVM still is able of achieving improved results. As we demonstrate in next section, the extra energy consumed by Hy-SVM larger on-chip SPMs is compensated by STT-RAM benefits and improved power management.

Comparing with RCDR (SAMPAIO et al., 2013a) data reuse strategy, it can be noted increased savings when higher parallelism levels are adopted in HEVC encoders. In the best case scenarios, Hy-SVM can reduce the off-chip energy by 21%, 33% and 44% when considering 16-Tile Low Delay settings. RCDR achieves improved energy efficiency for Random Access prediction structure since its implemented reference-centered alternative processing order can better exploit the reuse of the same reference frame to perform more consecutive MEs. However, due to RCDR search window level date reuse, Hy-SVM could overcome the off-chip energy savings by employing a more complete access redundancy support.

When analyzing Level-D, Hy-SVM can improve off-chip energy efficiency by up to 51%, on average in the best case scenario (256x256 search window). In this case, the increased parallelism does not significantly affect the energy reduction. When analyzing the worst case of Hy-SVM (128x128 search window and Random Access structure), Level-D overcomes the energy savings by 11%. In the remaining cases, Hy-SVM can reach better results. However, Level-D was not developed to support alternative CTUs processing scheduling, since its data reuse strategy is completely dependent on the traditional raster scan order. In its turn, Hy-SVM multi-level data reuse (search window level in L1 SPMs and reference frame level in L2 SPMs) can be adaptive to the run-time adaptive alternative CTUs processing order proposed to balance the accumulated memory pressure of all HEVC processing units. The results of Level-D in Figure 7.1 considers the traditional raster scan order, which is not be the same for alternative re-scheduled CTUs processing orders.

AMBER (KHAN; SHAFIQUE; HENKEL, 2013) and enHyV (SAMPAIO et al., 2014c) achieved improved off-chip energy savings when compared to Hy-SVM: up to 12% and 22% on average, respectively. AMBER fully exploits reference frame level data reuse, avoiding data re-fetching from external memory during a frame processing. However, to support Tiles-parallelized HEVC, AMBER requires the multiplication of its on-chip video memories, which strongly affects its on-chip energy efficiency (as discussed in next sections). enHyV implements data-reuse schemes in the same levels of Hy-SVM, without a proper management of the overlap formation. In the external memory perspective, enHyV provides a more complete support for inter-Tiles redundant accesses, leading to reduced SharedL2 misses. Still, Hy-SVM can achieve competitive off-chip energy results and, additionally, implements an overlap management that strongly reduces the SharedL2 SPMs on-chip energy. Considering an overall energy analysis, which combines off- and on-chip energy parts, Hy-SVM surpasses AMBER and enHyV implementation due to more efficient on-chip power management (discussed in Section 7.2.2).

Compared to our *Priv-Only* baseline implementation, the SharedL2 SPMs contributes by reducing from 11% (2-Tile) up to 71% (16-Tile) the external memory energy consumption, on average. Note that the achieved savings increase when more Tiles are used (higher parallelism), due to the well-exploited increased inter-Tiles data reuse potential by Hy-SVM. We demonstrate in the next sections that the on-chip energy required for SharedL2 SPMs is strongly reduced by our energy-efficient management schemes, resulting on savings when compared to *Priv-Only*.

Observing the average results comparing the two evaluated HEVC prediction structures, we can note that Hy-SVM can achieve better energy efficiency when Low Delay is adopted: the savings are 5%-10% higher when compared to Random Access scenario. The main reason is related to a more predictable behavior of the overlap formations in Low Delay, since it organizes the frame dependencies adopting the same DME factors for each processed frame (as can be seen in Figure 6.1, presented in the beginning of Section 6.1). It is demonstrated in the analyses of 7.3, when the accuracy of the proposed overlap prediction is evaluated under different test conditions.

7.2.1.2. *Analysis-2: Search Window Size and Video Resolution*

Another important remark from Analysis-1 is the increased savings achieved by Hy-SVM when increase search window is analyzed. For a proper exploitation of this parameter, as well the impact of the video resolution, Figure 7.2 presents the off-chip energy savings for growing search window and video dimensions.

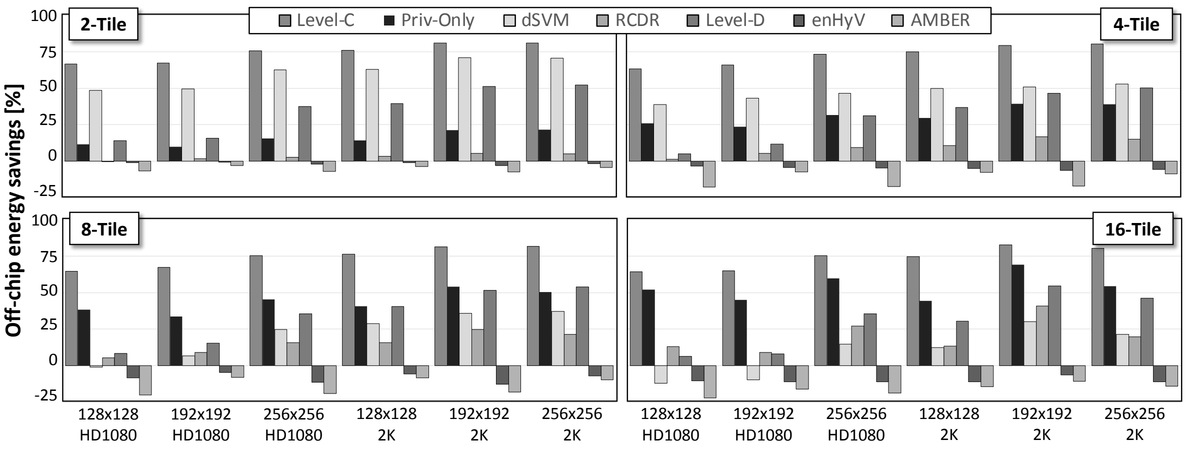


Figure 7.2: Off-chip energy savings of Hy-SVM compared to related works for increased search window size and video resolution (average scenario of HD1080 and 2K tested video sequences and of Random Access and Low Delay structures).

In a general perspective, it can be note an growing behavior of the achieved energy efficiency when increased search windows and video resolutions are adopted. Starting by Level-C, the achieved Hy-SVM reduction is improved by 12% and 11%, on average, when increasing the search window size for HD1080 and 2K resolution videos, respectively. In case of the savings regarding dSVM architecture, the average gains are 63% (HD1080) and 52% (2K). For RCDR, larger search windows lead to an increase of energy savings of 119% (HD1080) and 78% (2K). The same behavior is noted when compared to Level-D: 180% (HD1080) and 119% (2K). When compared to AMBER and enHyV, which achieve better results than Hy-SVM, the energy savings behavior is not strongly related to the search window size, since all these architecture implement reference frame level data reuse.

7.2.1.3. *Analysis-3: Video Sequences Characteristics*

Besides the different energy saving results for different video resolutions, the varied video characteristics, like low/high motion and texture properties, may affect the efficiency of the proposed Hy-SVM architecture. Figure 7.3 depicts the off-chip energy savings for all tested video sequences, separated for 2-, 4-, 8- and 16-Tile HEVC scenarios. In this analysis, the 192x192 search window size is fixed and the average of the achieved savings for Random Access and Low Delay configurations is considered.

We observe that the off-chip energy savings may vary according to video content properties. HEVC encoding of high motion sequences (like *Kimono* and *BasketballDrive (BDrive)* in Figure 7.3) lead to larger overlaps, since the motion search reaches more distant reference frame samples. In these cases, our Hy-SVM architecture is able to exploit this increased inter-Tiles data reuse potential and save external memory communication. In contrast, low motion videos like *Bosphorus* and *Traffic*, lead to smaller overlap formations. In these cases, inter-Tiles data reuse potential is itself lower, leading to reduced energy efficiency of implementing SharedL2 SPMs to exploit this data redundancy. Therefore, the energy savings of Hy-SVM compared to related works is reduced. As discussed in the motivational analysis of Section 4.3, it was a premise for Hy-SVM design this run-time adaptivity of the proposed management schemes to the video content characteristics, enabling dynamic higher/lower energy savings depending on the captured potential of inter-Tiles data reuse.

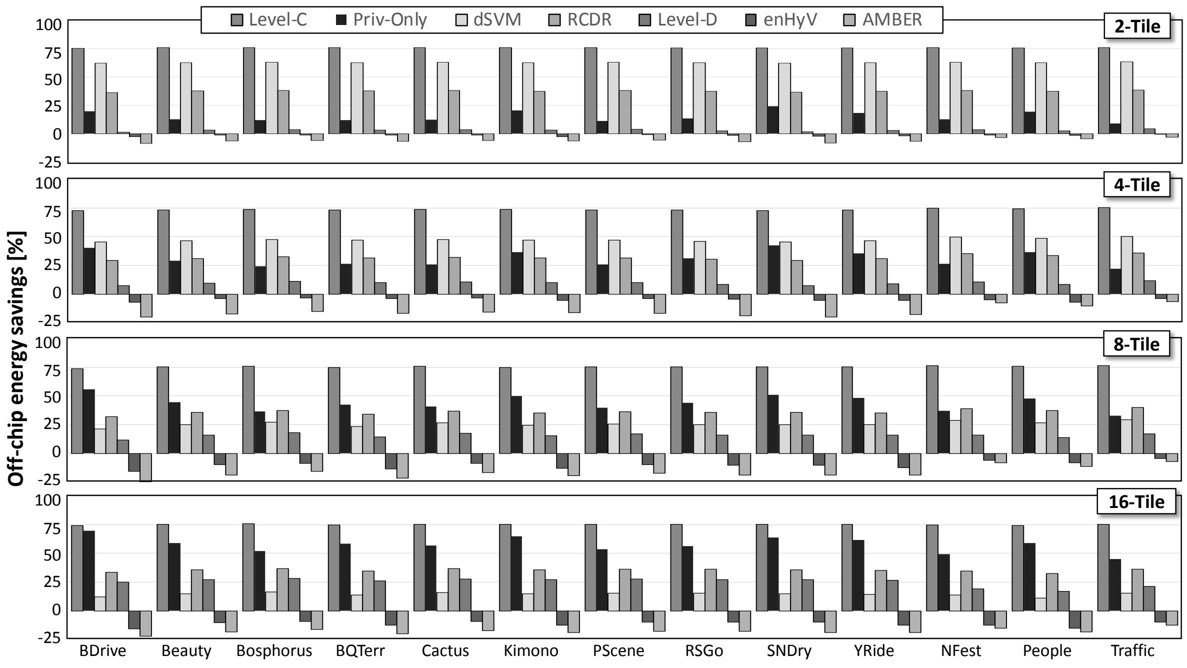


Figure 7.3: Off-chip energy savings of Hy-SVM compared to related works for different input video sequences (fixed 192x192 search window size; average scenario between Random Access and Low Delay structures).

7.2.2. *On-Chip Energy Results*

7.2.2.1. *Overall On-Chip Energy Savings*

Figure 7.4 shows the on-chip energy analysis of Hy-SVM compared to related works and baseline. In this evaluation, the energy consumption was normalized to All-SRAM results, which represents the worst-case scenario among the related works. Besides, the analysis also compares the on-chip energy for the three selected search window sizes. The energy measurements in Figure 7.4 represent the average scenario of HD1080 and 2K tested videos. The results of this analysis are barely the same for Random Access and Low Delay structures. The chosen prediction structure for HEVC encoding does not affect the static energy consumption, since the design methodology of HyMs generates the same SPM parameters for both cases. In terms of dynamic energy (related to write and read accesses), Random Access typically requires higher number of on-chip SPM accesses, which leads to higher dynamic energy consumption. However, as static portion represents the major part of the total on-chip memory energy consumption, the total energy consumption difference between Random Access and Low Delay structures becomes insignificant.

From enHyV perspective, Hy-SVM achieves, in the best case, on-chip energy savings of up to 82%-95% (HD1080-2K), 73%-92% and 64%-89% for 2-Tile scenarios using 128x128, 192x192 and 256x256 search windows, respectively. The gains over enHyV are mostly due to an improved power management, which relies on overlap prediction to increase the on-chip static energy savings. An accurate prediction allows Hy-SVM to shut down SPM sectors outside the overlap (for SharedL2 SPMs) or inside the overlap (for PrivL2 SPMs) to reduce on-chip energy in according to input video content characteristics. In the worst cases (HD1080 16-Tile scenario), Hy-SVM still overcomes enHyV on-chip energy savings by 9% (256x256), 14% (192x192) and 28% (128x128).

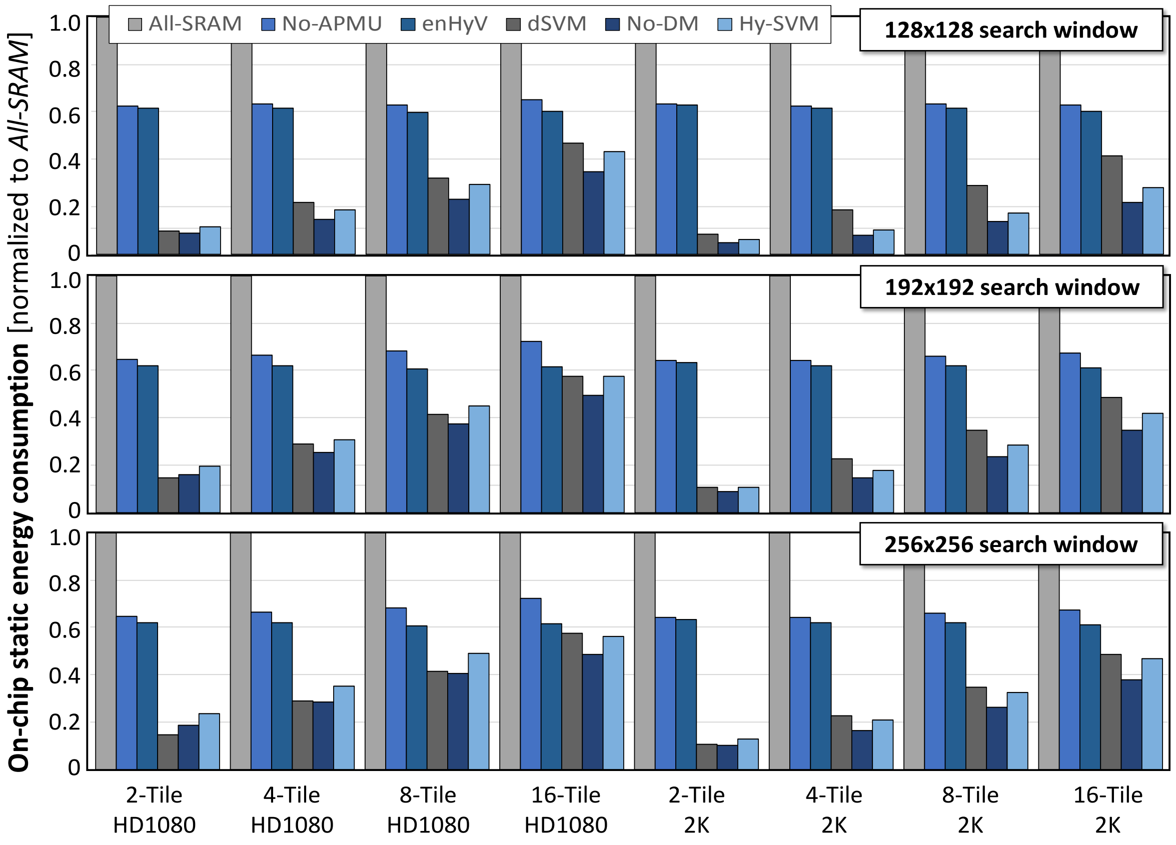


Figure 7.4: On-chip energy consumption of Hy-SVM compared to related works.

When compared to dSVM, our Hy-SVM architecture presents competitive on-chip energy consumption. dSVM surpasses Hy-SVM gains when lower parallelism levels are adopted. For instance, when considering 2-Tile scenarios of Figure 7.4, the energy consumption is up to 65% (256x256), 46% (192x192) and 21% (128x128) lower than Hy-SVM. On another perspective, when growing number of Tiles is adopted, Hy-SVM overcomes dSVM in terms of on-chip energy efficiency, saving up to 8% (256x256), 25% (192x192) and 45% (256x256) compared to dSVM. Combining these with the previous off-chip energy results (overall perspective presented in next section), Hy-SVM is able reduce the energy of external memory communication with competitive on-chip energy consumption.

Compared to baseline *All-SRAM* and *No-APMU* implementations, our Hy-SVM can achieve up to 94% and 95% (2-Tile), 90% and 84% (4-Tile), 83% and 72% (8-Tile), and 73% and 56% (16-Tile) energy reduction, respectively. The savings related to *All-SRAM* are related to the STT-RAM low leakage power dissipation. The savings compared to *No-APMU* represent the efficiency of our power management over the two levels of SPMs of Hy-SVM.

When observing the *No-DM* baseline implementation results, we note that complete Hy-SVM consumes from 18%-30% more on-chip energy. This overhead is related to the SRAM SPMs that were inserted in PrivL2 and SharedL2 HyMs to alleviate STT-RAM from high bit-toggling write accesses, leading to improved STT-RAM cells lifetime (as demonstrated in Section 7.5).

The authors of AMBER did not informed the on-chip static energy savings lead by the proposed power-gating scheme. Thus, it is not possible to perform a fair comparison with Hy-

SVM, so AMBER results were not inserted in the analysis of Figure 7.4[14]. AMBER represents an upper bound scenario in terms of on-chip energy consumption, since it is necessary to replicate the storage of the reference frames for each Tile processing. Considering this, Hy-SVM consumes, on average, 95% (128x128), 92% (192x192) and 90% (256x256) less on-chip energy than AMBER. To have an idea of on-chip memory optimization of Hy-SVM compared to AMBER, we compare it to the *No-APMU* baseline version of Hy-SVM. In this analysis, we can also note savings of up to 58%-69% (2-Tile), 68%-81% (4-Tile), 81-88% (8-Tile) and 87%-92% (16-Tile), on average, when processing HD1080 and 2K videos, respectively. Therefore, even when not considering the gains achieved by the APMU, the proposed design methodology overcomes AMBER in terms of on-chip memory energy optimization.

7.2.2.2. *On-Chip Energy Savings in PrivL1 SPMs*

Figure 7.5a depicts the on-chip static energy savings specific of PrivL1 SPMs. On average, the proposed scheme saves 56% of on-chip energy by power gating the unused and less-likely used memory sectors. The wake-up energies overhead is already included into the results of Figure 7.5.

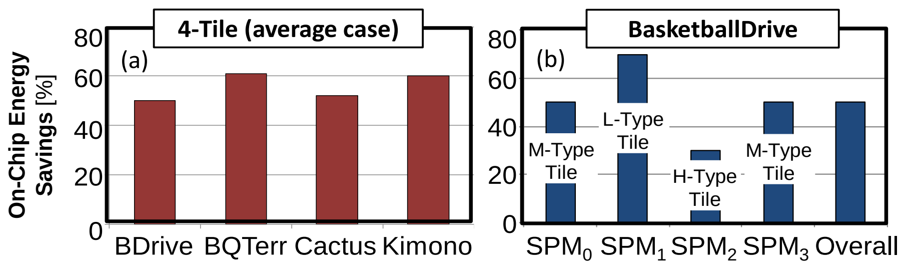


Figure 7.5: On-chip static energy reduction due to adaptive power management of PrivL1 SPMs.

The achieved energy reductions are high in case of low-motion Tiles. It incurs in longer sleep durations due to consecutive processing of CTU with similar texture and motion. This behavior is demonstrated in Figure 7.5b where the total energy savings are decomposed for each PrivL1 SPM that is responsible to handle with Tile-specific search window. In this analysis, we adopted the same scenario used as example during the explanation of the memory pressure management scheme (Section 6.2). The low-motion Tiles provide the highest savings while the medium- and high-motion Tiles required more energy due to higher memory usage as a result of an extensive search. When considering SPMs, the energy/performance overhead of waking up the memory cells are negligible, since one block of the search window is continuously accessed during one ME operation over a given block of the CTU. Thus, the energy/performance penalty is completely amortized, not leading to significant overhead for the overall memory system.

7.2.3. *Overall Energy Results*

Table 7.1 presents the overall energy savings of Hy-SVM for three different scenarios. The total energy is computed (sixth column of Table 7.1) by the composition of off- and on-chip parts, including the control hardware that implements the management schemes.

[14] To have a comparison with AMBER on-chip energy, we estimated the size of its on-chip memories and, based on the defined design methodology, extracted the energy consumption components using the same methodology than Hy-SVM.

Table 7.1: Overall Energy Savings of Hy-SVM Compared to Related Works

Solution	On-Chip Mem. [KB]	On-Chip Energy [mJ]	Off-Chip BW. [MB/s]	Off-Chip Energy [mJ]	Total Energy [mJ]	Savings Hy-SVM [%]
***Scenario 1:** 4-Tile HD1080, 192x192 search window and Low Delay*						
Level-C (TUAN; CHANG; JEN, 2002)	256	121	435	5708	5829	**69.3%**
Level-D (TUAN; CHANG; JEN, 2002)	240	110	128	1681	1790	**0.1%**
dSVM (SAMPAIO et al., 2014a)	631	285	218	2854	3139	**43.0%**
AMBER (KHAN; SHAFIQUE; HENKEL, 2013)	8100	4072	99	1297	5369	**66.7% (58.7%*)**
enHyV (SAMPAIO et al., 2014c)	3496	726	111	1453	2179	**18.0%**
Our No-APMU	3496	763	111	1453	2216	**19.3%**
Our No-DM	2656	249	111	1453	1702	**-5.0%**
Our Hy-SVM	3496	313	112	1475	1788	**-**
***Scenario 2:** 4-Tile 2K, 192x192 search window and Low Delay*						
Level-C (TUAN; CHANG; JEN, 2002)	256	121	840	8806	8927	**72.6%**
Level-D (TUAN; CHANG; JEN, 2002)	320	210	235	2466	2676	**8.6%**
dSVM (SAMPAIO et al., 2014a)	776	440	420	4403	4844	**49.5%**
AMBER (KHAN; SHAFIQUE; HENKEL, 2013)	16000	4144	195	2048	6192	**60.5% (44.4%*)**
enHyV (SAMPAIO et al., 2014c)	6358	1310	200	2100	3410	**28.3%**
Our No-APMU	6358	1344	200	2100	3444	**29.0%**
Our No-DM	4776	260	200	2100	2360	**-3.6%**
Our Hy-SVM	6358	330	202	2116	2446	**-**
***Scenario 3:** 8-Tile 2K, 256x256 search window and Random Access*						
Level-C (TUAN; CHANG; JEN, 2002)	800	345	1172	30488	30833	**78.5%**
Level-D (TUAN; CHANG; JEN, 2002)	320	210	272	7073	7283	**8.8%**
dSVM (SAMPAIO et al., 2014a)	1720	884	340	8842	9726	**31.7%**
AMBER (KHAN; SHAFIQUE; HENKEL, 2013)	32000	8288	195	5081	13369	**50.3% (48.3%*)**
enHyV (SAMPAIO et al., 2014c)	7442	1588	200	5212	6800	**2.4%**
Our No-APMU	7442	1695	200	5212	6907	**3.9%**
Our No-DM	5720	668	200	5212	5881	**-12.9%**
Our Hy-SVM	7442	830	223	5810	6640	**-**

savings of No-APMU over AMBER

Compared to Level-C scheme, even this presenting the smallest on-chip video memory (i.e., lower on-chip energy), Hy-SVM can reach overall energy savings of 69%-79%. These gains are mainly related to the reduction of 5.2 times in the external memory energy by exploiting inter-Tiles data reuse. When analyzing Level-D strategy, we note competitive results: from 0.1% to 9% of savings. Level-D also implements reference frame level data reuse (as Hy-SVM), but it stores only one row of search windows on chip: this leads to a balanced usage on-chip storage and off-chip memory bandwidth. However, as already discussed, these improvements in the overall memory energy efficiency of Level-D are strongly based on regular raster scan order for CTUs processing. When alternative methods of CTUs re-scheduling are required, like the implemented in the proposed memory pressure management integrated in the Hy-SVM architecture, the energy savings of Level-D are compromised.

Regarding the dSVM architecture, the hybrid multiple levels of SPMs allows Hy-SVM total memory energy savings of 31%-50% for the tested scenarios. Even requiring more and larger

SPMs, Hy-SVM is able to reduce on-chip energy consumption by adopting improved power management, compared to dSVM. Moreover, multiple levels of SPMs allowed reduced off-chip memory bandwidth and, consequently, improved off-chip energy efficiency. When comparing to AMBER, which achieves the best off-chip energy results, Hy-SVM reaches up to 66.7% of total energy savings. If we do not consider the APMU and the overlap management of Hy-SVM, the *No-APMU* baseline version can, still, achieve overall energy reduction compared to AMBER: 59%, 44% and 48% for scenarios 1, 2 and 3, respectively. As previously discussed, enHyV also achieves lower external memory energy consumption than Hy-SVM. However, in an off- and on-chip combined perspective, Hy-SVM surpasses enHyV by achieving up to 28% of improved energy efficiency.

Finally, the proposed management layer, composed of overlap prediction, and on-chip MAMUs and AMPUs, can improve the energy efficiency of Hy-SVM by up to 29% (best case of scenario 2). The insertion of SRAM SPMs in L2 HyMs, as well as all involved hardware circuitry to support the proposed lifetime-aware data management represents, in the overall energy perspective, an overhead from 4% up to 13% in terms of overall energy consumption.

7.3 Overlap Prediction Accuracy Evaluation

Figure 7.6 depicts the metrics used for the accuracy evaluation of the proposed overlap prediction scheme. For each ME, our memory simulation environment captures the predicted overlap (exemplified in Figure 7.6a), as well as the actual overlap formation (Figure 7.6b). Based on these, a prediction accuracy map is built (as in Figure 7.6c). For each BU inside the SharedL2 HyM storage area for a specific Tiles boundary, which is dimensioned by the design-time calculated parameters $Ov_{Thickness}$ and Ov_{Length}, our methodology classifies it as a prediction hit, as an over-prediction, or as an under-prediction. One prediction hit means that the target BU was correctly estimated as inside/outside the overlap formation, thus indicating the accuracy of the overlap prediction scheme. One over prediction means that one BU, which was initially predicted as being part of the overlap, is not inside the actual overlap formation. This case leads to Hy-SVM on-chip energy wasting, since one entire SPM sector is assigned with *ON* state and no inter-Tiles data reuse is verified. In another perspective, one under prediction signifies that one BU inside the actual formed overlap was not predicted accordingly. In this scenario, off-chip energy wasting is verified since inter-Tiles data reuse is not exploited.

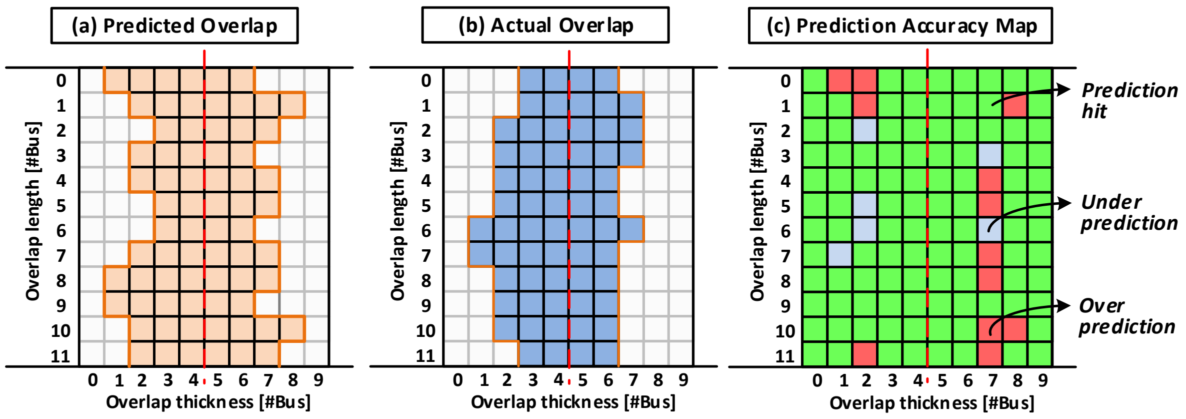


Figure 7.6: Representation of the adopted methodology for overlap prediction accuracy evaluation.

Figure 7.7 depicts the accuracy evaluation of the proposed overlap prediction, compared to a baseline scenario where no prior prediction is performed and all SharedL2 HyM BUs are assumed to be part of the overlap. In this analysis, the most important factors that impact the accuracy evaluation are the number of Tiles, the search window size and the adopted HEVC prediction structure. The percentage values of Figure 7.7 are related to the average cases for all tested video sequences.

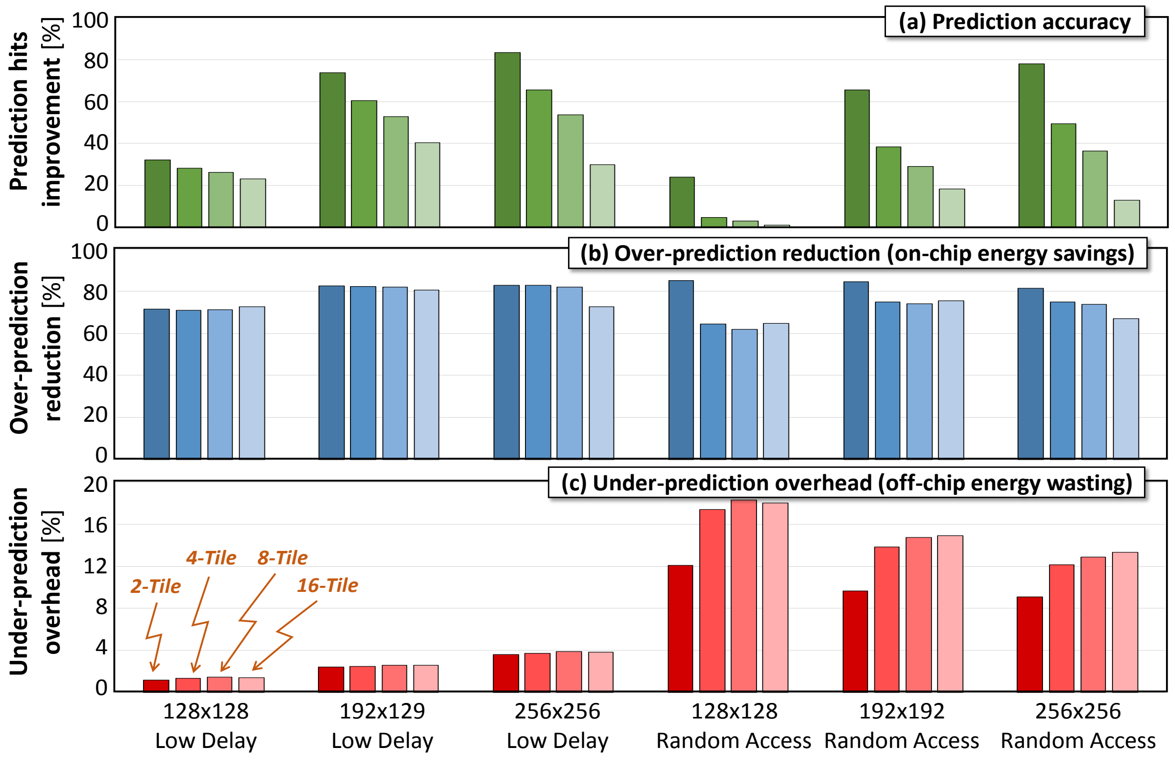


Figure 7.7: Overlap prediction evaluation, in terms of: (1) prediction accuracy, (2) over-prediction reduction and (3) under-prediction overhead.

The first analysis (Figure 7.7a) presents the improvements in term of prediction hits. The best cases are observed for lower number of Tiles, achieving up to 32%, 74% and 83% of increased accuracy (2-Tile Low Delay scenarios). It can also be noted a growing accuracy trend when larger search windows are used. Regarding the HEVC prediction structure, the proposed prediction scheme presents best results when Low Delay is selected. As already indicated in previous evaluations, Low Delay prediction dependencies have a more predictable behavior when compared to Random Access.

In the second analysis (Figure 7.7b), the over prediction reduction is evaluated. It is important to notice that the case where the entire SharedL2 HyM BUs are estimated as overlap represents the worst case of over prediction. Compared to this baseline, the proposed overlap prediction strategy is able to reduce the over predictions by 81%, 75%, 74%, and 72% on average for the tested 2-, 4-, 8-, and 16-Tile scenarios. The behavior of the over prediction indexes are similar for all tested search windows and HEVC prediction structures.

At the end, the overhead of under prediction occurrences are analyzed in Figure 7.7c. Note that the adopted baseline case, where all BUs of a SharedL2 SPM are assumed to be inside the overlap, does not have under prediction occurrences. Considering the Low Delay structure, only 1%-4%, on average, of under predicted BUs were verified. Due to the more unpredicted behavior of Random Access prediction dependencies, higher overhead was verified (from 12% to 16%, on average). As already demonstrated in the energy efficiency evaluation of Hy-SVM, this overhead does not imply on significant penalties, when compared to related works.

7.4 Off-Chip Memory Communication Evaluation

Besides the energy efficiency improvements achieved by the proposed Hy-SVM architecture, another aspect addressed by this work is related to the unbalanced behavior of the external memory communication when adopting traditional raster scan order of CTU processing. Thus, this section evaluates the results achieved by the memory pressure management scheme with regarding to the CTU re-scheduling strategy that aimed to provide a more balanced external memory communication.

MSD metric: Let the $Mem_{[0...m]}$ be the discretized memory pressure measurements along the time. The mean squared deviance (MSD) calculates the squared different between each memory pressure measured point and the *Mem* average value (μ_{Mem}), as in Equation (47).

$$\mathrm{MSD}\left(\mathrm{Mem}_{[0\ldots m]}\right)=\frac{1}{|\mathrm{Mem}|}\sum_{i=0}^{m}\left(\mu_{\mathrm{Mem}}-\mathrm{Mem}_{i}\right)^{2} \tag{47}$$

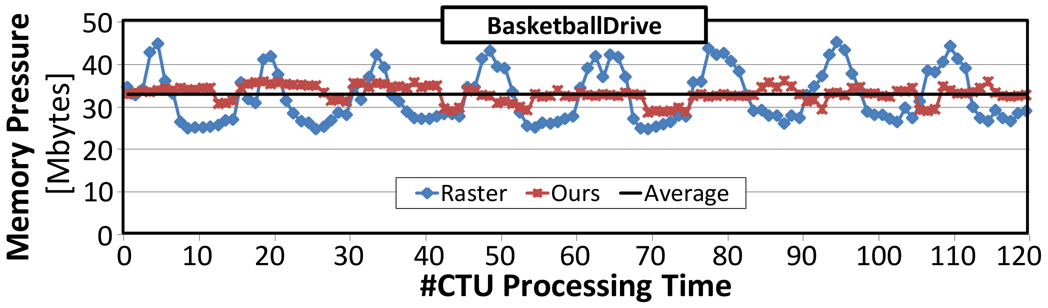


Figure 7.8: Accumulated memory pressure results.

Figure 7.8 presents a temporal evaluation of the memory pressure comparing (1) the traditional CTU raster processing order; (2) application-specific memory pressure balancing scheme using CTU-rescheduling; and (3) the optimal corner case where the memory pressure is continuously equals to the average pressure. The case (3) is a theoretical approximation used to evaluate the gaps of the schemes related to the best possible balancing case. Figure 7.8 shows that the proposed scheme balances the pressure for each processing unit. Compared to the traditional raster order, the maximum-minimum peak variations are reduced from 27%-32% to 9%-13%, respectively. The scheme proposed in this work achieves this balancing by effectively predicting the memory requirements, capturing the Tile-specific properties, and managing the processing order.

Figure 7.9 presents the results regarding the memory pressure balance. As already discussed, more Tiles potentially leads to more unbalanced accumulated memory pressure, since more concurrent memory accesses are performed during each time slot. In this scenario, there is a high probability of having very different motion properties being processed by different units at the same time. So, the balancing gap when more Tiles are used is higher. The proposed scheme successfully exploits this potential, as shown in Figure 7.9. The average MSD efficiency reduction ranges from 37% to 83%, for 4 to 16 Tiles. Therefore, the application-specific memory power management is efficiently scalable when working with an increased number of Tiles.

Figure 7.9c depicts a frame-by-frame MSD reduction analysis. During the first frame processing, as only spatial references can be used as input for the memory pressure predictor, the scheme achieves results close to the original raster order. However, by acquiring the temporal knowledge, the scheme fits the CTU-Groups accordingly to capture the motion properties and achieves increased memory pressure balancing for the other remaining frames.

Thus, the scheme can better balance the accumulated memory pressure by up to 49% in the case of 4-Tile *BasketballDrive* scenario (Figure 7.9c).

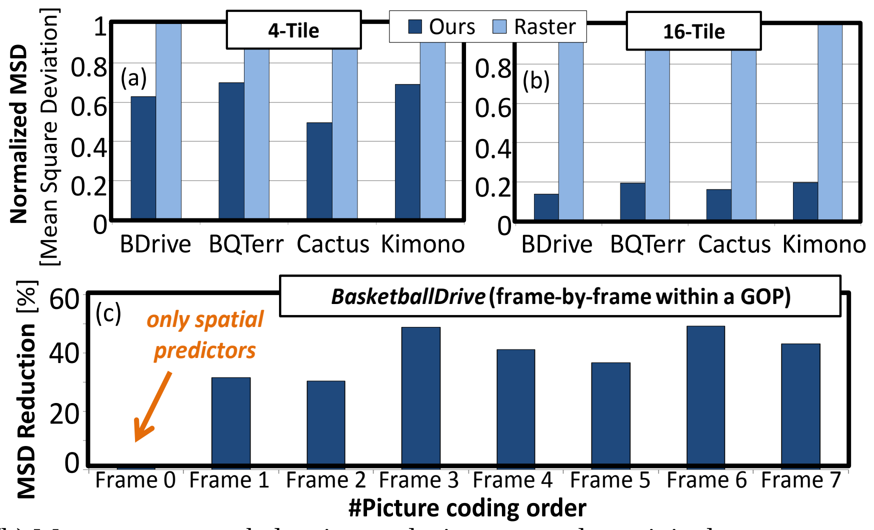


Figure 7.9: (a)(b) Memory pressure balancing analysis compared to original raster scan order and (c) frame-by-frame analysis (for Random Access prediction structure).

7.5 STT-RAM Lifetime Evaluation

The data management unit of Hy-SVM significantly improves the STT-RAM cells lifetime, as demonstrated in Figure 7.10.

In this analysis, the normalized STT-RAM lifetime is plotted bordered by the lifetime of Hy-SVM without any data management (*No-DM* baseline variation) and by the best case scenario, where no bit toggles occur during write operations (1.0 value). On average, it has a normalized lifetime of 0.83, nearer to the best case than the Hy-SVM basic approach without any management. The data management of Hy-SVM can achieve higher lifetime improvements for low-textured videos, like *Kimono* with 0.85 normalized lifetime. In another vein, highly detailed scenes lead to high bit-toggling activities, requiring high SRAM usage to alleviate STT-RAM cells. As Hy-SVM was designed for the average case, lifetime is less improved for this kind of videos. However, even for the worst case scenario, the scheme still can improve the lifetime (0.79 lifetime for *Traffic*).

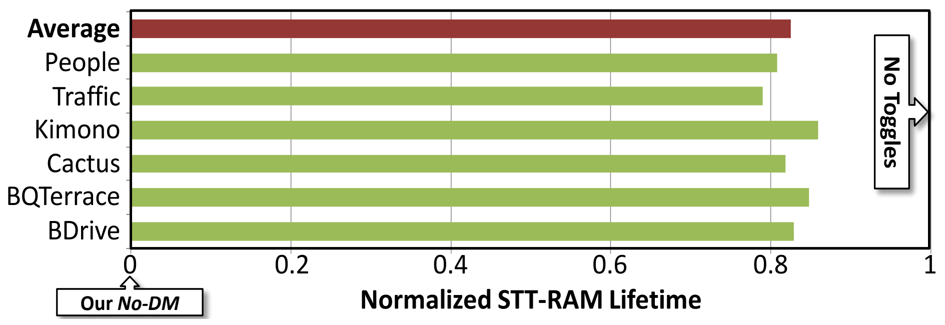


Figure 7.10: Normalized STT-RAM cells lifetime.

7.6 Overhead Evaluation

Figure 7.11 presents an overhead analysis of the implemented management techniques in Hy-SVM. We utilize the *All-SRAM* and *Priv-Only* baseline implementations to discuss the overhead in terms of access latency (Figure 7.11a) and dynamic energy consumption (Figure 7.11b).

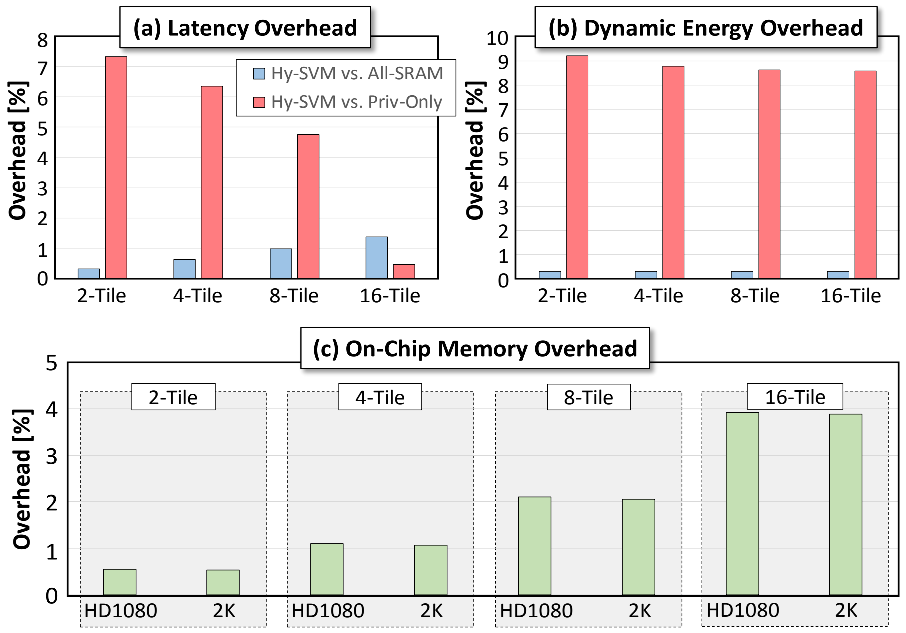


Figure 7.11: Overhead analysis in terms of: (a) latency, (b) dynamic energy and (c) extra on-chip memory size.

Compared to *All-SRAM*, the overhead of inefficient STT-RAM write operations represents (on average) only 0.3% in terms of latency, and 0.8% in dynamic energy. Since PrivL1 SPMs have high hit rates (more than 95%), combined to low write intensity of video memories, STT-RAM write penalty in Hy-SVM can be reduced. The comparison with *Priv-Only* version aims to evaluate the MAMU inserted overhead of analyzing the POTs to direct the incoming access either to PrivL2 or SharedL2 SPM. Still, the high L1 hit rates strongly reduces the overhead of overlap management, since it runs when L2 level access is required. Additionally, the hardware required for overlap management is composed of small tables and requires simple logic operations. As result, we can notice an overhead (on average) in the latency of only 4.7%, as well as 8.8% in the dynamic energy. Complementary, Figure 7.11c shows the extra on-chip memory size required to implement the proposed energy-efficient management layer of Hy-SVM. In this analysis, we compute the size of monitored and predicted overlap tables (POTs and MOTs), as well as the frame-level and CTU-level power maps. As result, the overhead achieves only 4% in the worst-case scenario (16-Tile).

8. CONCLUSIONS

This PhD work focused on designing an energy-efficient memory architecture (Hy-SVM) to enable parallel HEVC encoding. HEVC standard innovates on providing a light-weight parallelization feature: the Tiles partitioning. In the meantime that the adoption of multiple Tiles accelerates the HEVC encoding process, it strongly aggravates the memory issues, which were already restrictive for a non-parallelized coding scenario.

The proposed strategies were developed to efficiently support the novel coding structures and parallelization capabilities; *addressing G2 specific goal*[15]. For the proposed on-chip video memory architecture, both with respect to design methodology and with respect to their properly management schemes, energy efficiency could be achieved by leveraging application-specific properties from video coding applications; *addressing G1 specific goal*. The application knowledge was mostly exploited by implementing run-time monitoring units in the proposed architectures. The goal was to collect important hints to properly manage the off- and on-chip energy consumption. Some run-time analyzed information were: (a) redundant memory access to reference frames between different Tiles, (b) memory accesses correlations of CTUs within and across different Tiles, (c) Tile-specific and Tiles-accumulated memory pressure, and (d) bit-toggling activity during reference frames replacement; *addressing G4 specific goal*. Furthermore, HEVC encoding parameters also served as important information to optimize the on-chip memory design and all proposed management schemes. Some examples are: adopted prediction structures, decided CTU division along the coding tree structure, and video content properties (like motion and texture characteristics).

The main contributions of this work were mostly based on research opportunities brought by recent advances in memory technologies and organization models. The use of scratchpad memory, which strongly increased the potential of energy reduction of designed Hy-SVM architecture, was utilized as base of the on-chip data storage organization; *addressing G5 specific goal*. The Hy-SVM architecture implements private and shared SPMs to provide energy-efficient storage of Tile-specific (search window and Tile-specific reference frame region) and Tiles-shared (redundant reference frame region) data. By adopting a multi-level organization, L1 (composed of private SRAM SPMs) and L2 (composed of private and shared SRAM/STT-RAM hybrid SPMs - HyMs), Hy-SVM was able to provide a complete support of intra-Tile and inter-Tiles data reuse, incurring on significantly reduction of off-chip data communication to fetch the reference data when compared to video memories designed by related works; *addressing G6 specific goal*. Hy-SVM architecture relies on hybrid memory design, by taking advantage from SRAM and STT-RAM benefits to potentiate energy savings to the on-chip HyMs. A proper design space methodology was performed to measure the best distribution between SRAM and STT-RAM portions; *addressing G7 specific goal*. Moreover, offline statistical evaluations resulted on important knowledge that was inherited by the proposed design methodology of HyMs circuit-level parameters; *addressing G3 specific goal*.

To guarantee energy efficiency, adaptive management schemes provided run-time adaptation to different memory requirements of video coding (caused by variable properties of input video sequences). Initially, the redundant memory accesses of reference data from different processing units (called overlap) have been exploited by the overlap prediction management. It run-time estimates the overlap properties for the next frame processing and provides dynamic adaptation to the HyMs. Moreover, the unbalanced external memory bandwidth, cause by HEVC parallel encoding of distinct video coding properties that leads to

[15] To link the concluding remarks of this chapter with the prior PhD goals (defined in Section 1.3), the labels assigned to them are referred in this chapter.

unstable memory demands, was the target of the proposed memory pressure management; *addressing G8 specific goal*. The main goal was to re-schedule the CTU processing order to obtain a well-balanced memory pressure, leading to a better usage of the external memory channel. Endurance issues of STT-RAM cells were addressed by the proposed lifetime-aware data management; *addressing G7 specific goal*. This scheme was integrated in the energy-efficient management layer of Hy-SVM.

The data migration dynamics along the multiple SPM levels and the external memories were controlled by the distributed memory access management units (MAMUs). The MAMUs had the role of manage the SPM access according to the decisions taken by the memory management layer. In this sense, MAMUs implemented proper read and write policies in order to maximize the inter-Tiles data reuse and STT-RAM cells lifetime, as well as contribute to the memory pressure balancing. On-chip energy consumption was managed by the adaptive power management units (APMUs), which could significantly reduce the on-chip static consumption by leveraging more balanced memory requirements, leading to longer sleep durations to L1 private SPMs sectors, and providing accurate estimation of the overlap characteristics, resulting on energy savings for L2 private and shared SPMs.

8.1 Summary of Experimental Results

Several tested conditions of HEVC encoding affected the energy consumption behavior of Hy-SVM. In this sense, extensive analyses were performed using different video resolutions, search window sizes, prediction structures, number of Tiles, and video content properties.

In an overall scenario, the Hy-SVM architecture outperformed the related works when analyzing combined on-chip and off-chip energy consumption. The best case among the evaluated conditions, the overall energy reductions reached 79% (over Level-C), 9% (over Level-D), 50% (over dSVM), 59% (over AMBER), and 28% (over enHyV). These savings were achieved by the combined reduction provided by Hy-SVM in the on-chip and off-chip energy portions. In the off-chip perspective, the joint intra-Tile and inter-Tiles data reuse supported by the multiple levels of SPMs could strongly reduce the external memory communication. Thus, Hy-SVM surpassed some related works by 82% (over Level-C), 71% (over dSVM), 44% (over RCDR) and 51% (over Level-D). In contrast, AMBER and enHyV achieved better off-chip energy results: 22% and 12% of savings compared to Hy-SVM, on average. However, these reductions came with a cost of extra on-chip energy consumption. In this perspective, Hy-SVM outperformed all related works, reaching from 9% (worst case) to 95% (best case) of on-chip energy savings representing the best results in all evaluated scenarios. The contribution of each proposed strategy within Hy-SVM was evaluated due to the definition of baseline variations, which serves as comparison purpose. The utilization of STT-RAM in the hybrid memory design (in combination with SRAM arrays) increased the on-chip energy savings by up to 73%-94%. Furthermore, APMUs could leverage well balanced memory requirements and an accurate overlap estimation to contribute with up to 56%-95% of reduction (only on-chip part), and up to 29% (overall consumption).

Most of the memory energy consumption improvements of Hy-SVM were resultant from accurate estimations from the overlap prediction scheme, which could be verified by proper accuracy analyses. For the evaluated scenarios, the prediction strategy improved the accuracy by 83% (in the best case), compared to a baseline case where no management is performed. The cases where over predictions occur could be reduced by 72%-81%, and the under prediction overhead reached 1%-16%, on average. Well-balanced off-chip memory communication was achieved by the memory pressure management scheme, which re-schedules the CTUs processing order leading to 37%-83% of mean squared deviance reduction. STT-RAM cells lifetime was significantly extended by the proposed data management strategy. In this case, the experimental results pointed to a 0.83 normalized lifetime, corresponding to important improvements on cells endurance. The overhead of integrating all related management layers were also performed in terms of energy consumption, access latency and on-chip dynamic energy, and extra on-chip memory cells. For all evaluated scenarios, the measured overhead did not imply on significantly affect Hy-SVM energy efficiency.

Considering the achieved energy savings, the contributions in terms of balanced memory pressure and extended memory cells lifetime, as well the insignificant overhead of implementing Hy-SVM run-time management schemes, *it was demonstrated the feasibility of energy-efficient multimedia processing supporting parallel execution in state-of-the-art HEVC encoders, thus addressing the main goal of this PhD Thesis*.

8.2 Publications during the PhD Work

As results of this PhD research, , several scientific publications were concluded. In the first part of PhD activities, two preliminary video memories implementations initially exploited the potential of SPMs and hybrid memory design (combination of SRAM and STT-RAM) to improve energy efficiency of reference data storage in parallel HEVC. Thus, the Distributed Scratchpad Video Memory (dSVM) architecture and its related management schemes were published in the Design, Automation and Test in Europe Conference and Exhibition – DATE 2014 (SAMPAIO et al., 2014a). Besides the memory design, this work also introduced the overlap concept and proposed a first version of overlap prediction scheme. Moreover, as energy-efficient hybrid video memory (enHyV) architecture and its involved management parts were published in the IEEE/ACM International Conference on Computer-Aided Design – ICCAD 2014 (SAMPAIO et al., 2014c). In this work, lifetime optimization of STT-RAM was firstly exploited. The memory pressure management scheme was the subject of a paper published in the IEEE/ACM International Symposium on Low Power Electronics and Design – ISLPED 2014 (SAMPAIO et al., 2014b). Improvements on multi-level hybrid SPMs organization and on the overlap prediction strategy, as well as detailed memory access power management units, were part of the contributions that compose a submitted paper to the IEEE Transactions on Circuits and Systems for Video Technology (TCSVT). As result from all these efforts, we proposed the Hy-SVM architecture in this PhD thesis, which integrates the hybrid memories design and the application-driven management layers published in these partial works.

An initial version of an approximation-aware multi-level STT-RAM memory architecture, based on resilience evaluations of video coding applications, was published in the International Conference on Compilers, Architecture and Synthesis for Embedded Systems – CASES 2015 (co-located event of the Embedded Systems Week – ESWeek) (SAMPAIO et al., 2015). It composes an initial effort within the approximate storage research field, representing a starting point for future works, as detailed as follows.

The list of publications and the complete published papers are presented in Annex A.

8.3 Future Works

This Thesis explored different research topics in order to improve the energy efficiency of parallel HEVC requirements regarding its memory infrastructure. In this sense, there are several other research challenges that should be addressed, as well as opportunities from recent advances that can be applied to video coding scenario. They are summarized as follows:

8.3.1. *Approximate storage for video coding applications*

Video coding can be classified as a resilient application when considering the reference frames as a resilient data. Thus, approximate storage can be exploited over such data to achieve energy savings during the memory operations. In this case, energy efficiency can be improved by tolerating a controlled level of error occurrences with the goal of simplifying correction routines.

Recently, state-of-the-art works have explored data approximations for energy reductions in main memories (JUNG et al., 2016; LIU et al., 2011; SAMPSON et al., 2013). *Jung et al. (2016)* performs a review regarding possibilities of approximation in DRAM memories. As a practical implementation, *Liu et al. (2011)* extends the data refresh interval of DRAM memories to potentially save energy consumption while assuming wear-out errors. *Sampson et al. (2013)* uses approximate storage in PCM-based main memories by reducing write pulses and leading to wear-out errors.

An ongoing research during this PhD work leveraged approximate storage opportunities from resilient application were also exploited. An application-aware strategy was proposed by *Sampaio et al. (2015)*, which means that other resilient applications besides video coding can take advantage from its improved energy efficiency. Unreliable multi-level cells (MLC) of STT-RAM were adopted as memory infrastructure. The MLC design is a promising alternative to single-level cells (SLC). In a MLC, one physical memory cell is able to store more than one logic bit. Recent studies (BI et al., 2013; ZHANG et al., 2012) have demonstrated the feasibility of MLC-based design of STT-RAM towards scalability and energy efficiency for larger banks. However, due to the process variations, memory arrays based on MLC STT-RAM have more frequent error occurrences during memory read and write operations. Thus, the energy consumption can be compromised by the required extra circuitry to guarantee the reliability of the memory system.

In the last years, state-of-the-art on-chip memory implementations that rely on approximate storage using advances of STT-RAM technologies have been proposed (MONAZZAH et al., 2017; RANJAN et al., 2017; ZHAO et al., 2017). *Monazzah et al. (2017)* and *Ranjan et al. (2017)* developed general-purpose memories using selective approximation strengths to optimize the energy efficiency of last-level caches. In an application-driven perspective, *Zhao et al. (2017)* designed an MLC STT-RAM architecture for on-chip storage for image applications.

Video coding applications have specific resilience behavior, which is mostly dependent on the video content properties. Moreover, this variability may occur between different sequences, as well as inside the same video, becoming even more intricate the exploitation of error tolerance techniques to maximize the energy efficiency while minimizing the coding efficiency drops. Thus, *an important key challenge is to leverage specific video coding resilience properties to enable approximation storage for the exploration of reliability-energy-quality tradeoffs in on- and off-chip video memories*. Furthermore, any data approximation in video encoders should also guarantee error-free execution for the non-resilient kernels, ensuring no critical failures during the execution.

8.3.2. *Memory requirements evaluation for next-generation video coding standards*

Next-generation video coding standards are currently being developed by joint groups coordinated by video standardization committees and world-wide companies involved on video processing systems.

One effort is conducted by a joint group composed of experts from ITU-T and ISO/IEC called "Joint Video Exploration Team" (JVET, 2018). By starting from HEVC reference software, this group is evaluating novel coding tools to improve HEVC standard, focusing on improve coding efficiency. The contributions from video processing research groups and involved companies are integrated in the JEM reference software. As the same flow of H.264/AVC and HEVC standards definition, extensive evaluations will be performed and the most promising strategies will be incorporated in the final standard.

A parallel investigation is performed by a consortium of leading Internet companies, such as Amazon, Apple, ARM, Cisco, Facebook, Google, IBM, Intel, Microsoft, Mozilla, Netflix and NVIDIA, called Alliance for Open Media (AOM, 2018). The goal is to develop next-generation media formats, codecs and technologies. Alliance members bring their collective technology and expertise to meet growing Internet demand for top-quality video, audio, imagery and streaming across devices of all kinds and for users worldwide. As first initiative from this project, royalty-free video codec specification and open source implementation was provided, named as AV1. In contrast to the definition of previous video coding standards, this is the first time that an open video codec standardization was guided by commercial interests.

At the same time that important efforts are noticed to improve HEVC coding efficiency during the definition of the next-generation video coding standards, performance and energy issues of such novel tools should be extensively analyzed. Novel strategies for parallelization have been proposed, which may highly require support from the memory system. Therefore, besides coding efficiency and performance optimization evaluations, *there is also a strong need for memory requirements assessment of the novel tools proposed by the next-generation video codecs*.

REFERENCES

ABE, K. et al. **Novel hybrid DRAM/MRAM design for reducing power of high performance mobile CPU**. 2012 International Electron Devices Meeting. **Anais**... In: 2012 INTERNATIONAL ELECTRON DEVICES MEETING. dez. 2012

AHN, J.; YOO, S.; CHOI, K. Prediction Hybrid Cache: An Energy-Efficient STT-RAM Cache Architecture. **IEEE Transactions on Computers**, v. 65, n. 3, p. 940–951, mar. 2016.

ALVAREZ, L. et al. **Runtime-Guided Management of Scratchpad Memories in Multicore Architectures**. 2015 International Conference on Parallel Architecture and Compilation (PACT). **Anais**... In: 2015 INTERNATIONAL CONFERENCE ON PARALLEL ARCHITECTURE AND COMPILATION (PACT). out. 2015

AOM. **Home**. Disponível em: <http://aomedia.org/>. Acesso em: 19 fev. 2018.

ARJOMAND, M. et al. **Leveraging value locality for efficient design of a hybrid cache in multicore processors**. 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). **Anais**... In: 2017 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD). nov. 2017

BANAKAR, R. et al. **Scratchpad memory: a design alternative for cache on-chip memory in embedded systems**. Proceedings of the Tenth International Symposium on Hardware/Software Codesign, 2002. CODES 2002. **Anais**... In: PROCEEDINGS OF THE TENTH INTERNATIONAL SYMPOSIUM ON HARDWARE/SOFTWARE CODESIGN, 2002. CODES 2002. 2002

BI, X. et al. **Unleashing the potential of MLC STT-RAM caches**. Proceedings of the International Conference on Computer-Aided Design. **Anais**...IEEE Press, 2013Disponível em: <http://dl.acm.org/citation.cfm?id=2561913>. Acesso em: 13 jun. 2014

BLUMENBERG, C. et al. **Adaptive content-based Tile partitioning algorithm for the HEVC standard**. Picture Coding Symposium (PCS), 2013. **Anais**... In: PICTURE CODING SYMPOSIUM (PCS), 2013. dez. 2013

BONATTO, L. V. M. et al. **Low-power Multi-size HEVC DCT Architecture Proposal for QFHD Video Processing**. Proceedings of the 30th Symposium on Integrated Circuits and Systems Design: Chip on the Sands. **Anais**...: SBCCI '17.New York, NY, USA: ACM, 2017Disponível em: <http://doi.acm.org/10.1145/3109984.3109987>. Acesso em: 15 fev. 2018

CHEN, C.-Y. et al. Level C+ data reuse scheme for motion estimation with corresponding coding orders. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 16, n. 4, p. 553–558, abr. 2006.

CHEN, D. et al. Viewer-Aware Intelligent Efficient Mobile Video Embedded Memory. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. PP, n. 99, p. 1–13, 2018.

CHEN, K. et al. A Novel Wavefront-Based High Parallel Solution for HEVC Encoding. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 26, n. 1, p. 181–194, jan. 2016.

CHEN, Y.-T. et al. **Dynamically reconfigurable hybrid cache: An energy-efficient last-level cache design**. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012. **Anais**...2012Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6176431>. Acesso em: 2 maio. 2013

CHENG, S.-P.; HUANG, S.-Y. **A low-power SRAM design using quiet-bitline architecture**. 2005 IEEE International Workshop on Memory Technology, Design, and Testing (MTDT'05).

Anais... In: 2005 IEEE INTERNATIONAL WORKSHOP ON MEMORY TECHNOLOGY, DESIGN, AND TESTING (MTDT'05). ago. 2005

CHI, C. C. et al. Parallel Scalability and Efficiency of HEVC Parallelization Approaches. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1827–1838, dez. 2012.

CHIANG, P. T. et al. A QFHD 30-frames/s HEVC Decoder Design. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 26, n. 4, p. 724–735, abr. 2016.

CHO, S. et al. Efficient In-Loop Filtering Across Tile Boundaries for Multi-Core HEVC Hardware Decoders With 4 K/8 K-UHD Video Applications. **IEEE Transactions on Multimedia**, v. 17, n. 6, p. 778–791, jun. 2015.

CISCO. **Visual Networking Index (VNI) Forecast Highlights Tool**. Disponível em: <https://www.cisco.com/c/m/en_us/solutions/service-provider/vni-forecast-highlights.html>. Acesso em: 6 nov. 2017.

CORREA, G. et al. Complexity control of high efficiency video encoders for power-constrained devices. **IEEE Transactions on Consumer Electronics**, v. 57, n. 4, p. 1866–1874, nov. 2011.

CORREA, G. et al. **Motion compensated tree depth limitation for complexity control of HEVC encoding**. 2012 19th IEEE International Conference on Image Processing (ICIP). **Anais**... In: 2012 19TH IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP). 2012

CORREA, G. et al. **Coding Tree Depth Estimation for Complexity Reduction of HEVC**. 2013 Data Compression Conference. **Anais**... In: 2013 DATA COMPRESSION CONFERENCE. mar. 2013

DONG, X. et al. **Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) As a Universal Memory Replacement**. : DAC '08.New York, NY, USA: ACM, 2008Disponível em: <http://doi.acm.org/10.1145/1391469.1391610>. Acesso em: 4 nov. 2017

FAN, Y. et al. A Hardware-Oriented IME Algorithm for HEVC and its Hardware Implementation. **IEEE Transactions on Circuits and Systems for Video Technology**, v. PP, n. 99, p. 1–1, 2017.

GRELLERT, M. et al. **A multilevel data reuse scheme for Motion Estimation and its VLSI design**. 2011 IEEE International Symposium of Circuits and Systems (ISCAS). **Anais**... In: 2011 IEEE INTERNATIONAL SYMPOSIUM OF CIRCUITS AND SYSTEMS (ISCAS). maio 2011

GUO, L. et al. **Frame-level quality and memory traffic allocation for lossy embedded compression in video codec systems**. 2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW). **Anais**... In: 2016 IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA EXPO WORKSHOPS (ICMEW). jul. 2016

GUO, L.; ZHOU, D.; GOTO, S. A New Reference Frame Recompression Algorithm and Its VLSI Architecture for UHDTV Video Codec. **IEEE Transactions on Multimedia**, v. 16, n. 8, p. 2323–2332, dez. 2014.

HANSEN, H. E. et al. **A shared scratchpad memory with synchronization support**. 2017 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC). **Anais**... In: 2017 IEEE NORDIC CIRCUITS AND SYSTEMS CONFERENCE (NORCAS): NORCHIP AND INTERNATIONAL SYMPOSIUM OF SYSTEM-ON-CHIP (SOC). out. 2017

HP LABS. **HP Labs : CACTI**. Disponível em: <http://www.hpl.hp.com/research/cacti/>. Acesso em: 6 nov. 2017.

IBM RESEARCH. **The Cell Project - IBM**. CT002. Disponível em: <http://researcher.watson.ibm.com/researcher/view_group.php?id=2649>. Acesso em: 29 jul. 2015.

IMANI, M.; PATIL, S.; ROSING, T. **Low power data-aware STT-RAM based hybrid cache architecture**. 2016 17th International Symposium on Quality Electronic Design (ISQED). **Anais**... In: 2016 17TH INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN (ISQED). mar. 2016

ISO/IEC-JCT1/SC29/WG11. **Common test conditions and software reference configurations**, 2012.

ISO/IEC-JCT1/SC29/WG11. **High Efficiency Video Coding (HEVC) text specification draft 10**, 2013a.

ISO/IEC-JCT1/SC29/WG11. **High Efficiency Video Coding (HEVC) Test Model 13 (HM 13) Encoder Description**, 2013b.

ITU-T. **ITU-T Recommendation H.264 (05/2003): advanced video coding for generic audiovisual services**, 2013.

JACOB, B.; NG, S.; WANG, D. **Memory Systems: Cache, DRAM, Disk**. [s.l.] Elsevier Science, 2010.

JCT-VC. **JCT-VC - Joint Collaborative Team on Video Coding**. Disponível em: <http://www.itu.int:80/en/ITU-T/studygroups/2017-2020/16/Pages/video/jctvc.aspx>. Acesso em: 5 nov. 2017.

JEONG, M. K. et al. **A QoS-aware memory controller for dynamically balancing GPU and CPU bandwidth use in an MPSoC**. Proceedings of the 49th Annual Design Automation Conference. **Anais**...ACM, 2012Disponível em: <http://dl.acm.org/citation.cfm?id=2228513>. Acesso em: 14 mar. 2014

JIN, X.; DAI, Q. Clustering-Based Content Adaptive Tiles Under On-chip Memory Constraints. **IEEE Transactions on Multimedia**, v. 18, n. 12, p. 2331–2344, dez. 2016.

JIN, Y.; SHIHAB, M.; JUNG, M. **Area, Power, and Latency Considerations of STT-MRAM to Substitute for Main Memory**. Proc. ISCA. **Anais**...2014Disponível em: <http://www.utdallas.edu/~jung/uploads/Main/stt-mram-study.pdf>. Acesso em: 27 jul. 2015

JOG, A. et al. **Cache revive: architecting volatile STT-RAM caches for enhanced performance in CMPs**. Proceedings of the 49th Annual Design Automation Conference. **Anais**...2012Disponível em: <http://dl.acm.org/citation.cfm?id=2228406>. Acesso em: 17 jul. 2013

JOU, S. Y.; CHANG, S. J.; CHANG, T. S. Fast Motion Estimation Algorithm and Design for Real Time QFHD High Efficiency Video Coding. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 25, n. 9, p. 1533–1544, set. 2015.

JUNG, M. et al. **Invited: Approximate computing with partially unreliable dynamic random access memory #x2014; Approximate DRAM**. 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC). **Anais**... In: 2016 53ND ACM/EDAC/IEEE DESIGN AUTOMATION CONFERENCE (DAC). jun. 2016

JVET. **JVET JEM software | JVET**. Disponível em: <https://jvet.hhi.fraunhofer.de/>. Acesso em: 19 fev. 2018.

KHAN, M. U. K.; SHAFIQUE, M.; HENKEL, J. **AMBER: Adaptive energy management for on-chip hybrid video memories**. 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). **Anais**... In: 2013 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD). nov. 2013

KHAN, M. U. K.; SHAFIQUE, M.; HENKEL, J. **Software architecture of High Efficiency Video Coding for many-core systems with power-efficient workload balancing**. 2014 Design, Automation Test in Europe Conference Exhibition (DATE). **Anais**... In: 2014 DESIGN, AUTOMATION TEST IN EUROPE CONFERENCE EXHIBITION (DATE). mar. 2014

KHOSHAVI, N. et al. Read-Tuned STT-RAM and eDRAM Cache Hierarchies for Throughput and Energy Enhancement. **arXiv:1607.08086 [cs]**, 27 jul. 2016.

KIM, H.; KIM, S.; LEE, J. Write-Amount-Aware Management Policies for STT-RAM Caches. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 25, n. 4, p. 1588–1592, abr. 2017.

KÜLTÜRSAY, E. et al. **Evaluating STT-RAM as an energy-efficient main memory alternative**. 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). **Anais**... In: 2013 IEEE INTERNATIONAL SYMPOSIUM ON PERFORMANCE ANALYSIS OF SYSTEMS AND SOFTWARE (ISPASS). abr. 2013

LAPEDUS, M. **Semiconductor Engineering .:. Four Foundries Back MRAM**. Disponível em: <https://semiengineering.com/four-foundries-back-mram/>. Acesso em: 20 fev. 2018.

LI, J. et al. **Hybrid cache architecture with disparate memory technologies**. ACM SIGARCH Computer Architecture News. **Anais**...2009Disponível em: <http://dl.acm.org/citation.cfm?id=1555761>. Acesso em: 2 maio. 2013

LIAN, X. et al. Parallel Content-Aware Adaptive Quantization Oriented Lossy Frame Memory Recompression for HEVC. **IEEE Transactions on Circuits and Systems for Video Technology**, v. PP, n. 99, p. 1–1, 2016a.

LIAN, X. et al. Lossless Frame Memory Compression Using Pixel-Grain Prediction and Dynamic Order Entropy Coding. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 26, n. 1, p. 223–235, jan. 2016b.

LIAO, W.; YANG, D.; CHEN, Z. **A fast mode decision algorithm for HEVC intra prediction**. 2016 Visual Communications and Image Processing (VCIP). **Anais**... In: 2016 VISUAL COMMUNICATIONS AND IMAGE PROCESSING (VCIP). nov. 2016

LIU, S. et al. **Flikker: Saving DRAM Refresh-power Through Critical Data Partitioning**. Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems. **Anais**...: ASPLOS XVI.New York, NY, USA: ACM, 2011Disponível em: <http://doi.acm.org/10.1145/1950365.1950391>. Acesso em: 27 jul. 2015

LIU, T. M. et al. **Energy and area efficient hardware implementation of 4K Main-10 HEVC decoder in Ultra-HD Blu-ray player and TV systems**. 2015 IEEE International Conference on Multimedia and Expo (ICME). **Anais**... In: 2015 IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME). jun. 2015

M, M. A. B.; SK, N. M. **High Performance Integer DCT Architectures for HEVC**. 2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems (VLSID). **Anais**... In: 2017 30TH INTERNATIONAL CONFERENCE ON VLSI DESIGN AND 2017 16TH INTERNATIONAL CONFERENCE ON EMBEDDED SYSTEMS (VLSID). jan. 2017

MERTENS, R. **GlobalFoundries: 22nm eMRAM technology is now available, prototyping to start in Q1 2018 | MRAM-Info**. Disponível em: <https://www.mram-info.com/globalfoundries-22nm-emram-technology-now-available-prototyping-start-q1-2018>. Acesso em: 20 fev. 2018.

MERTENS, RON. **Everspin starts to produce commercial 40nm 256Mb STT-MRAM chips | MRAM-Info**. Disponível em: <https://www.mram-info.com/everspin-starts-produce-commercial-40nm-256mb-stt-mram-chips>. Acesso em: 20 fev. 2018.

MICRON TECHNOLOGY INC. **TN-46-03 – Calculating DDR Memory System Power**Micron Technology Inc., , 2001. Disponível em: <https://www.micron.com/~/media/documents/products/technical-note/dram/tn4603.pdf?la=en>. Acesso em: 10 maio. 2017

MICRON TECHNOLOGY INC. **TN-46-12: Mobile DRAM Power-Saving Features/Calculations**Micron Technology Inc., , 2005. Disponível em: <https://www.micron.com/~/media/documents/products/technical-note/dram/tn4612.pdf>. Acesso em: 10 maio. 2017

MICRON TECHNOLOGY INC. **4Gb: x16, x32 Mobile LPDDR2 SDRAM S4**Micron Technology Inc., , 2011. Disponível em: <https://www.micron.com/~/media/documents/products/data-sheet/dram/mobile-dram/low-power-dram/lpddr2/u80m_4gb_mobile_lpddr2_s4_sdram.pdf>. Acesso em: 11 maio. 2017

MIN, C. et al. **Extending the lifetime of object-based NAND flash device with STT-RAM/DRAM hybrid buffer**. 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC). **Anais**... In: 2017 22ND ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE (ASP-DAC). jan. 2017

MISRA, K. et al. An Overview of Tiles in HEVC. **IEEE Journal of Selected Topics in Signal Processing**, v. 7, n. 6, p. 969–977, dez. 2013.

MONAZZAH, A. M. H. et al. **QuARK: Quality-configurable approximate STT-MRAM cache by fine-grained tuning of reliability-energy knobs**. 2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). **Anais**... In: 2017 IEEE/ACM INTERNATIONAL SYMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN (ISLPED). jul. 2017

MONAZZAH, A. M. H.; FARBEH, H.; MIREMADI, S. G. OPTIMAS: Overwrite Purging Through In-Execution Memory Address Snooping to Improve Lifetime of NVM-Based Scratchpad Memories. **IEEE Transactions on Device and Materials Reliability**, v. 17, n. 3, p. 481–489, set. 2017.

PARK, S. et al. **An efficient motion estimation hardware architecture using Modified Reference Data Access(MRDAS) skip algorithm for high Efficiency Video Coding(HEVC) encoder**. 2016 IEEE 6th International Conference on Consumer Electronics - Berlin (ICCE-Berlin). **Anais**... In: 2016 IEEE 6TH INTERNATIONAL CONFERENCE ON CONSUMER ELECTRONICS - BERLIN (ICCE-BERLIN). set. 2016

PILLA, L. L. et al. **A hierarchical approach for load balancing on parallel multi-core systems**. Parallel Processing (ICPP), 2012 41st International Conference on. **Anais**...IEEE, 2012Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6337573>. Acesso em: 14 mar. 2014

PODDER, P. K.; PAUL, M.; MURSHED, M. Fast Mode Decision in the HEVC Video Coding Standard by Exploiting Region with Dominated Motion and Saliency Features. **PLOS ONE**, v. 11, n. 3, p. e0150673, 10 mar. 2016.

PORTO, R.; AGOSTINI, L.; BAMPI, S. **Hardware Design of the H.264/AVC Variable Block Size Motion Estimation for Real-Time 1080HD Video Encoding**. 2009 IEEE Computer Society Annual Symposium on VLSI. **Anais**... In: 2009 IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI. maio 2009

PURNACHAND, N.; ALVES, L. N.; NAVARRO, A. **Improvements to TZ search motion estimation algorithm for multiview video coding**. 2012 19th International Conference on Systems, Signals and Image Processing (IWSSIP). **Anais**... In: 2012 19TH INTERNATIONAL CONFERENCE ON SYSTEMS, SIGNALS AND IMAGE PROCESSING (IWSSIP). abr. 2012

RANJAN, A. et al. **STAxCache: An approximate, energy efficient STT-MRAM cache**. Design, Automation Test in Europe Conference Exhibition (DATE), 2017. **Anais**... In: DESIGN, AUTOMATION TEST IN EUROPE CONFERENCE EXHIBITION (DATE), 2017. mar. 2017

REED, E. et al. **Probabilistic Replacement Strategies for Improving the Lifetimes of NVM-based Caches**. Proceedings of the International Symposium on Memory Systems. **Anais**...: MEMSYS '17.New York, NY, USA: ACM, 2017Disponível em: <http://doi.acm.org/10.1145/3132402.3132433>. Acesso em: 2 fev. 2018

RICHARDSON, I. E. **H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia**. [s.l.] Wiley, 2004.

ROVATI, F. S. et al. An innovative, high quality and search window independent motion estimation algorithm and architecture for MPEG-2 encoding. **IEEE Transactions on Consumer Electronics**, v. 46, n. 3, p. 697–705, ago. 2000.

SAMPAIO, F. et al. **Energy-efficient memory hierarchy for Motion and Disparity Estimation in Multiview Video Coding**. Design, Automation Test in Europe Conference Exhibition (DATE), 2013. **Anais**... In: DESIGN, AUTOMATION TEST IN EUROPE CONFERENCE EXHIBITION (DATE), 2013. mar. 2013a

SAMPAIO, F. et al. **Content-adaptive reference frame compression based on intra-frame prediction for multiview video coding**. 2013 IEEE International Conference on Image Processing. **Anais**... In: 2013 IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING. set. 2013b

SAMPAIO, F. et al. **dSVM: Energy-efficient distributed Scratchpad Video Memory Architecture for the next-generation High Efficiency Video Coding**. Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014. **Anais**... In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION (DATE), 2014. mar. 2014a

SAMPAIO, F. et al. **Content-driven memory pressure balancing and video memory power management for parallel High Efficiency Video Coding**. 2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). **Anais**... In: 2014 IEEE/ACM INTERNATIONAL SYMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN (ISLPED). ago. 2014b

SAMPAIO, F. et al. **Energy-efficient architecture for advanced video memory**. 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). **Anais**... In: 2014 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD). nov. 2014c

SAMPAIO, F. et al. **Approximation-aware Multi-Level Cells STT-RAM cache architecture**. 2015 International Conference on Compilers, Architecture and Synthesis for

Embedded Systems (CASES). **Anais**... In: 2015 INTERNATIONAL CONFERENCE ON COMPILERS, ARCHITECTURE AND SYNTHESIS FOR EMBEDDED SYSTEMS (CASES). out. 2015

SAMPSON, A. et al. **Approximate Storage in Solid-state Memories**. Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture. **Anais**...: MICRO-46.New York, NY, USA: ACM, 2013Disponível em: <http://doi.acm.org/10.1145/2540708.2540712>. Acesso em: 31 jul. 2015

SHAFIQUE, M. et al. **Adaptive power management of on-chip video memory for Multiview Video Coding**. 2012 49th ACM/EDAC/IEEE Design Automation Conference (DAC). **Anais**... In: 2012 49TH ACM/EDAC/IEEE DESIGN AUTOMATION CONFERENCE (DAC). jun. 2012

SHAFIQUE, M.; KHAN, M. U. K.; HENKEL, J. **Power efficient and workload balanced tiling for parallelized high efficiency video coding**. 2014 IEEE International Conference on Image Processing (ICIP). **Anais**... In: 2014 IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP). out. 2014

SILVEIRA, D. et al. Efficient reference frame compression scheme for video coding systems: algorithm and VLSI design. **Journal of Real-Time Image Processing**, p. 1–21, 11 dez. 2015.

SINGH, H. et al. Enhanced Leakage Reduction Techniques Using Intermediate Strength Power Gating. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 15, n. 11, p. 1215–1224, nov. 2007.

SONG, C.; JU, L.; JIA, Z. **Hybrid scratchpad and cache memory management for energy-efficient parallel HEVC encoding**. 2015 33rd IEEE International Conference on Computer Design (ICCD). **Anais**... In: 2015 33RD IEEE INTERNATIONAL CONFERENCE ON COMPUTER DESIGN (ICCD). out. 2015

STROGENE.COM. **Strongene - HEVC/H.265 Decoder**. Disponível em: <http://www.xhevc.com/en/hevc/decoder/download.jsp>. Acesso em: 9 jan. 2018.

SULLIVAN, G. J. et al. Overview of the High Efficiency Video Coding (HEVC) Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1649–1668, dez. 2012.

SULLIVAN, G. J.; WIEGAND, T. Rate-distortion optimization for video compression. **IEEE Signal Processing Magazine**, v. 15, n. 6, p. 74–90, nov. 1998.

SZE, V.; BUDAGAVI, M.; SULLIVAN, G. J. (EDS.). **High Efficiency Video Coding (HEVC)**. Cham: Springer International Publishing, 2014.

TEXAS INSTRUMENTS. **TMS370CX7X from Texas Instruments**. Disponível em: <http://www.ti.com/mcu/docs/mcuorphan.tsp?contentId=15364>. Acesso em: 29 jul. 2015.

TUAN, J.-C.; CHANG, T.-S.; JEN, C.-W. On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 12, n. 1, p. 61–72, jan. 2002.

VALGRIND DEVELOPERS. **Valgrind Home**. Disponível em: <http://valgrind.org/>. Acesso em: 6 nov. 2017.

VAYALIL, N. C.; KONG, Y. VLSI Architecture of Full-Search Variable-Block-Size Motion Estimation for HEVC Video Encoding. **IET Circuits, Devices Systems**, v. 11, n. 6, p. 543–548, 2017.

VETRO, A.; WIEGAND, T.; SULLIVAN, G. J. Overview of the Stereo and Multiview Video Coding Extensions of the H.264/MPEG-4 AVC Standard. **Proceedings of the IEEE**, v. 99, n. 4, p. 626–642, abr. 2011.

VILLEGAS, A. et al. Lightweight Hardware Transactional Memory for GPU Scratchpad Memory. **IEEE Transactions on Computers**, v. PP, n. 99, p. 1–1, 2017.

VIZZOTTO, B. B. et al. **A Model Predictive Controller for Frame-Level Rate Control in Multiview Video Coding**. 2012 IEEE International Conference on Multimedia and Expo. **Anais**... In: 2012 IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO. jul. 2012

WIEGAND, T. et al. Overview of the H.264/AVC video coding standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 13, n. 7, p. 560–576, jul. 2003.

WU, X. et al. Design exploration of hybrid caches with disparate memory technologies. **ACM Transactions on Architecture and Code Optimization**, v. 7, n. 3, p. 1–34, 1 dez. 2010.

XIANGYU DONG et al. NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 31, n. 7, p. 994–1007, jul. 2012.

YAZDANSHENAS, S. et al. Coding Last Level STT-RAM Cache for High Endurance and Low Power. **IEEE Computer Architecture Letters**, v. 13, n. 2, p. 73–76, jul. 2014.

ZATT, B. et al. **Run-time adaptive energy-aware Motion and Disparity Estimation in Multiview Video Coding**. 2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC). **Anais**... In: 2011 48TH ACM/EDAC/IEEE DESIGN AUTOMATION CONFERENCE (DAC). jun. 2011a

ZATT, B. et al. **A low-power memory architecture with application-aware power management for motion amp; disparity estimation in Multiview Video Coding**. 2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). **Anais**... In: 2011 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD). nov. 2011b

ZATT, B. et al. **3D Video Coding for Embedded Devices: Energy Efficient Algorithms and Architectures**. [s.l.] Springer New York, 2016.

ZHANG, Y. et al. **Multi-level cell STT-RAM: Is it realistic or just a dream?** Proceedings of the International Conference on Computer-Aided Design. **Anais**...ACM, 2012Disponível em: <http://dl.acm.org/citation.cfm?id=2429498>. Acesso em: 13 jun. 2014

ZHAO, H. et al. **Approximate image storage with multi-level cell STT-MRAM main memory**. 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). **Anais**... In: 2017 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD). nov. 2017

ZHOU, D. et al. An 8K H.265/HEVC Video Decoder Chip With a New System Pipeline Design. **IEEE Journal of Solid-State Circuits**, v. 52, n. 1, p. 113–126, jan. 2017.

ZHU, J. et al. **An independent bandwidth reduction device for HEVC VLSI video system**. 2015 IEEE International Symposium on Circuits and Systems (ISCAS). **Anais**... In: 2015 IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS). maio 2015

APPENDIX A – CHARACTERIZATION OF DESIGNED ON-CHIP HYBRID VIDEO MEMORIES

Table A.1: Design-Decision Parameters of Hy-SVM Design Methodology

Input Parameters	
BU Dim.	8
BUs (Lines) per Sector	4
Bits per Sample	8
Motion Delta	2
Line Size (bits)	64
Sector Size (bits)	256

Table A.2: Sizing of Evaluated Hy-SVM Design Parameters (STT-RAM SPMs)

Scenario Parameters					**STT-RAM SPMs**			
Mem. Level	**SPM ID**	**#Tiles**	**Search Window**	**Video Resolution**	**Size (Bytes)**	**Size (KB)**	**Number of Lines (NL)**	**Number of Sectors (NS)**
PrivL2	**PrivL2-1**	2	-	HD1080	1036800	1013	16200	4050
	PrivL2-2	2		2K	2048000	2000	32000	8000
	PrivL2-3	4		HD1080	518400	506	8100	2025
	PrivL2-4	4		2K	1024000	1000	16000	4000
	PrivL2-5	8		HD1080	259200	253	4050	1013
	PrivL2-6	8		2K	512000	500	8000	2000
	PrivL2-7	16		HD1080	129600	127	2025	506
	PrivL2-8	16		2K	256000	250	4000	1000
SharedL2-Ver	**SharedL2-Ver-1**	-	128x128	HD1080	138240	135	2160	540
	SharedL2-Ver-2		128x128	2K	204800	200	3200	800
SharedL2-Hor	**SharedL2-Hor-1**	-	128x128	HD1080	245760	240	3840	960
	SharedL2-Hor-2		128x128	2K	138240	135	2160	540

Table A.3: Sizing of Evaluated Hy-SVM Design Parameters (RAM SPMs)

Scenario Parameters					**STT-RAM SPMs**			
Mem. Level	**SPM ID**	**#Tiles**	**Search Window**	**Video Resolution**	**Size (Bytes)**	**Size (KB)**	**Number of Lines (NL)**	**Number of Sectors (NS)**
PrivL1	**PrivL1-1**	-	128x128	-	36864	36	576	144
	PrivL1-2		192x192		65536	64	1024	256
	PrivL1-3		256x256		331776	324	5184	1296
PrivL2	**PrivL2-1**	2	-	HD1080	362880	354	5670	1418
	PrivL2-2	2		2K	716800	700	11200	2800
	PrivL2-3	4		HD1080	181440	177	2835	709
	PrivL2-4	4		2K	358400	350	5600	1400
	PrivL2-5	8		HD1080	90720	89	1418	354
	PrivL2-6	8		2K	179200	175	2800	700
	PrivL2-7	16		HD1080	45360	44	709	177
	PrivL2-8	16		2K	89600	88	1400	350
SharedL2-Ver	**SharedL2-Ver-1**	-	-	HD1080	48384	47	756	189
	SharedL2-Ver-4			2K	71680	70	1120	280
SharedL2-Hor	**SharedL2-Hor-1**	-	-	HD1080	86016	84	1344	336
	SharedL2-Hor-4			2K	48384	47	756	189

Table A.4: Power and Latency Components of Evaluated STT-RAM SPMs

SPM ID	Static Power (mW)	Dynamic Energy (pJ)		Latency (ns)	
		Read	Write	Read	Write
PrivL2-1	257,558	185,82	638,208	1,984	10,69
PrivL2-2	1018	276,83	729,131	1,8	10,604
PrivL2-3	128,561	133,596	589,01	1,9	10,653
PrivL2-4	257,558	185,82	638,208	1,984	10,69
PrivL2-5	64,281	103,368	558,782	1,877	10,637
PrivL2-6	128,561	133,596	589,01	1,9	10,653
PrivL2-7	32,1405	100,902	557,808	1,547	10,482
PrivL2-8	64,281	103,368	558,782	1,877	10,637
SharedL2-Ver-1	63,781	100,902	557,808	1,547	10,482
SharedL2-Ver-2	64,281	103,368	558,782	1,877	10,637
SharedL2-Hor-1	64,281	103,368	558,782	1,877	10,637
SharedL2-Hor-2	128,561	133,596	589,01	1,9	10,653

Table A.5: Power and Latency Components of Evaluated SRAM SPMs

SPM ID	Static Power (mW)	Dynamic Energy (pJ)		Latency (ns)	
		Read	Write	Read	Write
PrivL1-1	17,0668	0,118908	0,326939	0,979512	0,997519
PrivL1-2	30,1414	0,144643	0,144643	1,29467	1,29467
PrivL1-3	43,1164	0,253412	0,497958	1,29934	1,29934
PrivL2-1	420,034	1,1613	1,3594	3,56072	3,56072
PrivL2-2	819,35	1,70061	1,66746	5,05218	5,05218
PrivL2-3	210,017	0,755956	0,95405	2,61526	2,61526
PrivL2-4	420,034	1,1613	1,3594	3,56072	3,56072
PrivL2-5	109,736	0,395138	1,02507	2,28963	2,28963
PrivL2-6	210,017	0,755956	0,95405	2,61526	2,61526
PrivL2-7	55,0989	0,278891	0,592609	1,46829	1,46829
PrivL2-8	109,736	0,395138	1,02507	2,28963	2,28963
SharedL2-Ver-1	109,736	0,395138	1,02507	2,28963	2,28963
SharedL2-Ver-2	210,017	0,755956	0,95405	2,61526	2,61526
SharedL2-Hor-1	55,0989	0,278891	0,592609	1,46829	1,46829
SharedL2-Hor-2	109,736	0,395138	1,02507	2,28963	2,28963

APPENDIX B – DETAILED BIT-TOGGLING ACTIVITY AND DESIGN SPACE EXPLORATION FOR INCREASED STT-RAM LIFETIME

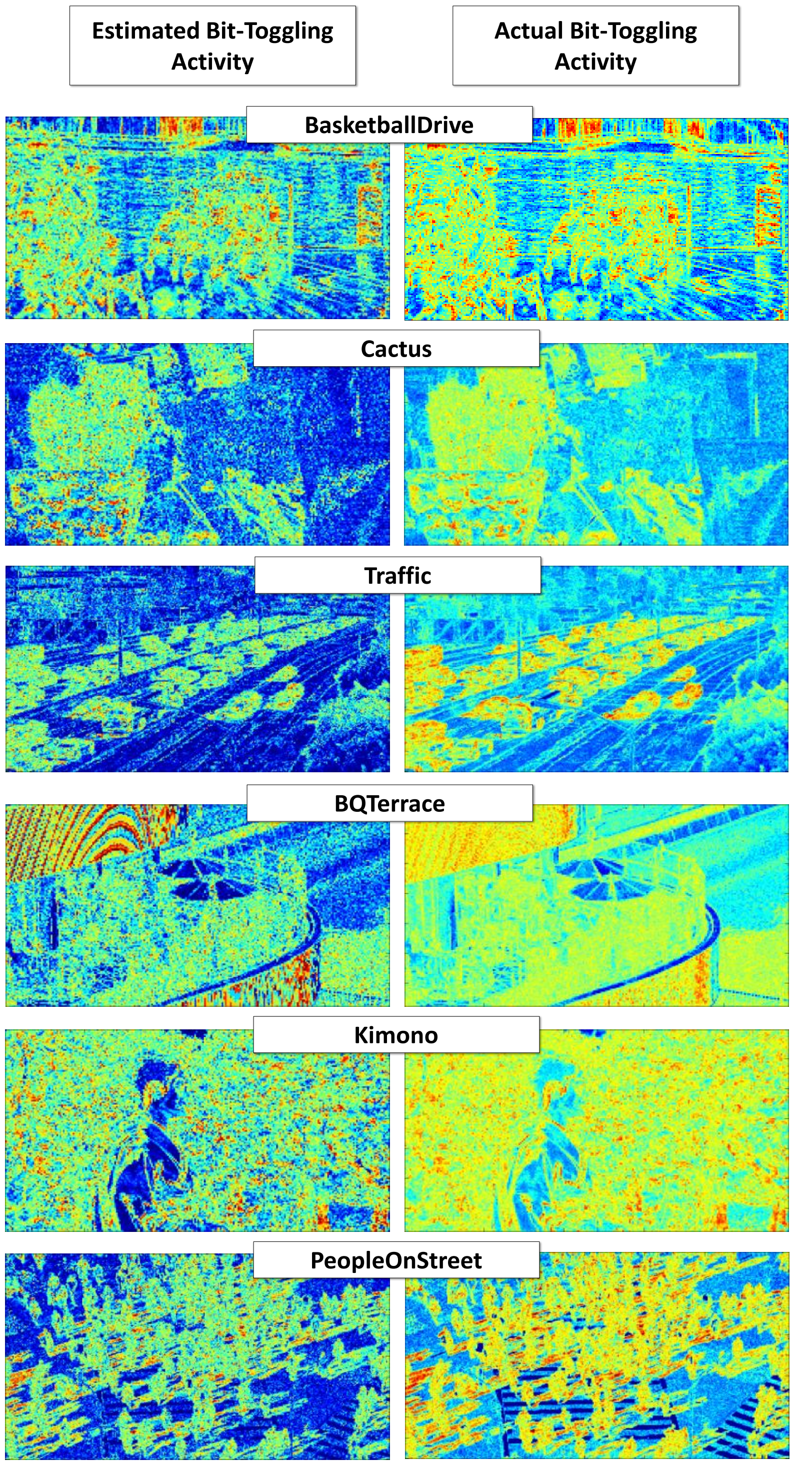


Figure B.1: Bit-toggling activity estimation and actual bit-toggling occurrences for all evaluated test sequences.

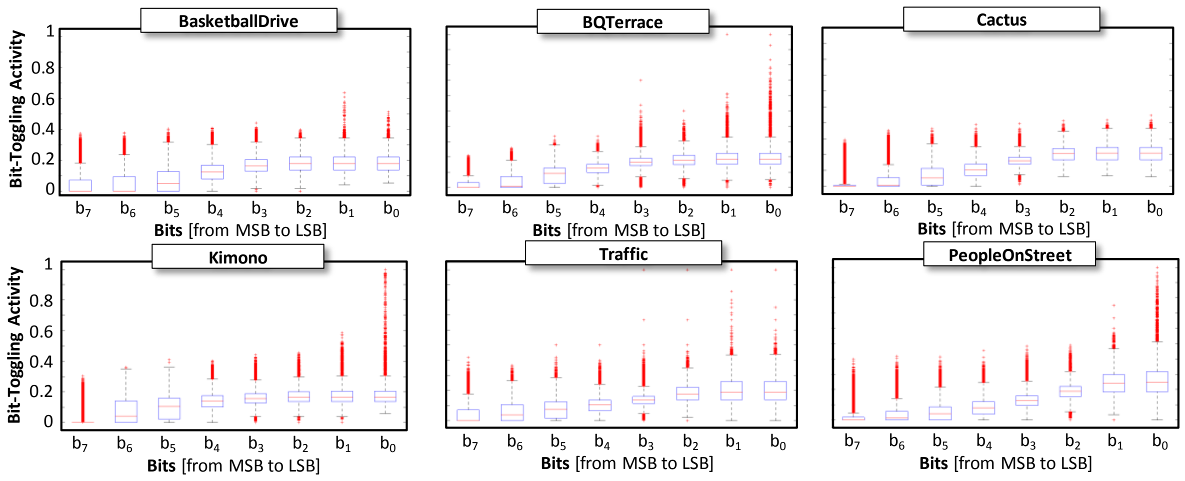


Figure B.2: Bit-toggling activity per bit using the evaluated test sequences.

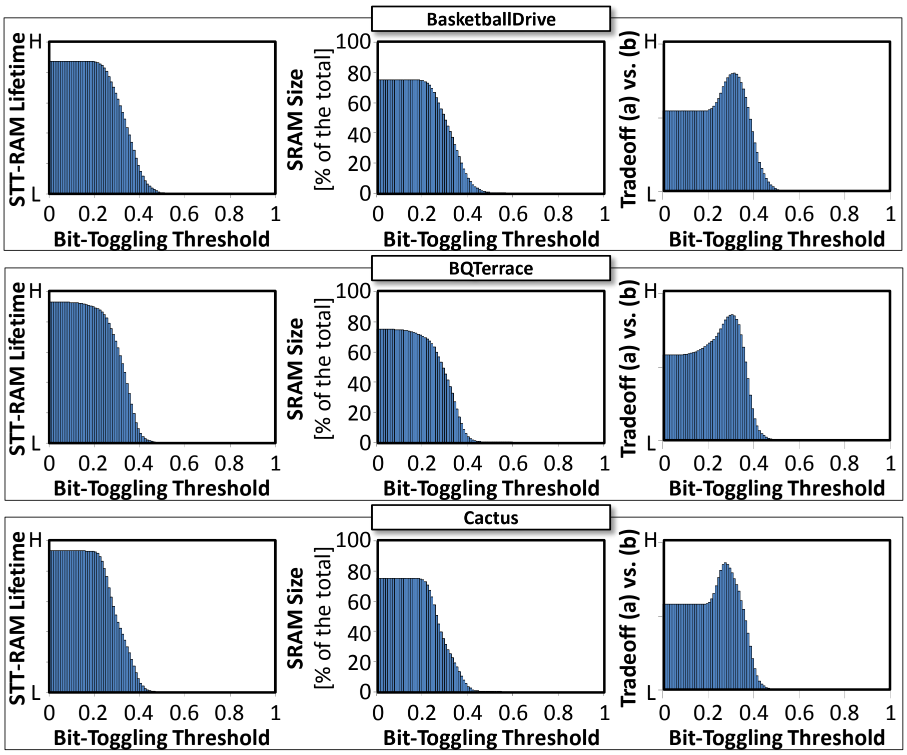


Figure B.3: Lifetime improvements and SRAM size for the BasketballDrive, BQTerrace and Cactus test sequences.

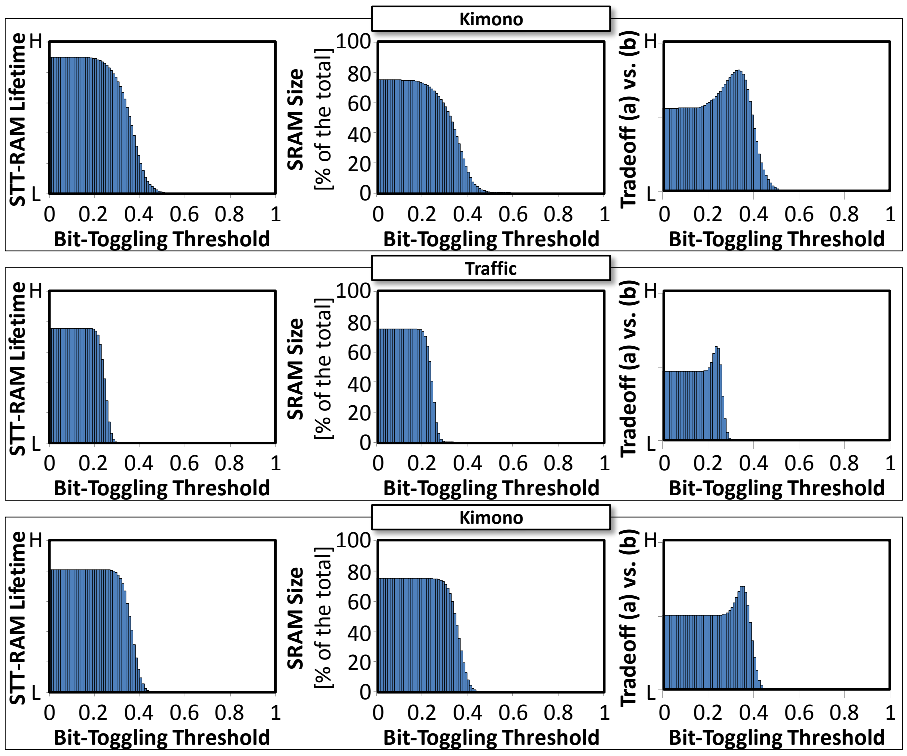


Figure B.4: Lifetime improvements and SRAM size for the Kimono, Traffic and Kimono test sequences.

ANNEX A – LIST OF PUBLICATIONS DURING THIS PHD WORK

SAMPAIO, F.; SHAFIQUE, M.; ZATT, B.; BAMPI, S.; HENKEL, J. **dSVM: Energy-efficient distributed Scratchpad Video Memory Architecture for the next-generation High Efficiency Video Coding**. Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014. **Anais**... In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION (DATE), 2014. mar. 2014

SAMPAIO, F.; SHAFIQUE, M.; ZATT, B.; BAMPI, S.; HENKEL, J. **Content-driven memory pressure balancing and video memory power management for parallel High Efficiency Video Coding**. 2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). **Anais**... In: 2014 IEEE/ACM INTERNATIONAL SYMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN (ISLPED). ago. 2014

SAMPAIO, F. ; SHAFIQUE, M.; ZATT, B.; BAMPI, S.; HENKEL, J. **Energy-efficient architecture for advanced video memory**. 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). **Anais**... In: 2014 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD). nov. 2014

SAMPAIO, F. ; SHAFIQUE, M.; ZATT, B.; BAMPI, S.; HENKEL, J.. **Approximation-aware Multi-Level Cells STT-RAM cache architecture**. 2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES). **Anais**... In: 2015 INTERNATIONAL CONFERENCE ON COMPILERS, ARCHITECTURE AND SYNTHESIS FOR EMBEDDED SYSTEMS (CASES). out. 2015

SAMPAIO, F.; SHAFIQUE, M.; ZATT, B.; BAMPI, S.; HENKEL, J. Hybrid Scratchpad Video Memory Architecture for Energy-Efficient Parallel HEVC. IEEE Transactions on Circuits and Systems for Video Technology (TCSVT). **(submitted)**

dSVM: Energy-Efficient Distributed Scratchpad Video Memory Architecture for the Next-Generation High Efficiency Video Coding

Felipe Sampaio[1], Muhammad Shafique[2], Bruno Zatt[3], Sergio Bampi[1], Jörg Henkel[2]
[1]Informatics Institute, PPGC, Federal University of Rio Grande do Sul (UFRGS), Brazil
[2]Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
[3]GACI, PPGC, CDTec, Federal University of Pelotas (UFPel), Brazil
{felipe.sampaio, bampi}@inf.ufrgs.br, bzatt@inf.ufpel.edu.br, {muhammad.shafique, henkel}@kit.edu

***Abstract*— An energy-efficient distributed Scratchpad Video Memory Architecture (dSVM) for the next-generation parallel High Efficiency Video Coding is presented. Our dSVM combines private and overlapping (shared) Scratchpad Memories (SPMs) to support data reuse within and across different cores concurrently executing multiple parallel HEVC threads. We developed a statistical method to size and design the organization of the SPMs along with a supporting memory reading policy for energy efficiency. The key is to leverage the HEVC and video content knowledge. Furthermore, we integrate an adaptive power management policy for SPMs to manage the power states of different memory parts at run time depending upon the varying video content properties. Our experimental results illustrate that our dSVM architecture reduces the overall memory energy consumption by up to 51%-61% compared to parallelized state-of-the-art solutions [11]. The dSVM external memory energy savings increase with an increasing number of parallel HEVC threads and size of search window. Moreover, our SPM power management reacts to the current video properties and achieves up to 54% on-chip leakage energy savings.**

Keywords—Video Memory, Scratchpad Memory, HEVC, Application-Specific Optimizations, Energy Efficiency, Adaptivity.

I. INTRODUCTION

To bridge the increasing gaps between the processor and memory scaling/speed in many-cores era with memory-intensive applications, specialization of memory architectures has become one of most important design issues. Multiple cores simultaneously accessing the same memory infrastructure incur high energy consumption and contention. Meanwhile, embedded multi-/many-core processors are subjected to stringent energy constraints. These issues intricate when executing memory-intensive applications like video coding, image matching, etc.

The *High Efficiency Video Coding* (HEVC) is the next-generation video coding standard [1] that provides double compression compared to its predecessor H.264/AVC. However, this comes at the cost of >40% more computation effort compared to the H.264 encoder as shown by our experimental analysis in Fig. 1a. This increased complexity is due to the novel *Coding Tree Unit* (CTU) structure [2] and a plethora of new prediction modes that result in an increase mode decision space [3]. Moreover, these new coding features lead to >2x more memory accesses compared to H.264/AVC due to more intensive reference frames storage and transmission (as in Fig. 1b). *A large amount of off-/on-chip memory accesses and large-sized on-chip memories lead to high energy consumption in HEVC encoders*. To achieve high performance, HEVC encoders can be parallelized on multi-/many-core processing platforms. However, this may lead to further increase in the energy consumption and memory pressure due to *multiple encoding cores requiring the same data* from the memory infrastructure, posing new challenges for the embedded multimedia systems.

A large body of research explored efficient cache organizations and on-chip memory architectures for general purpose multi-/many-core processors [18]. To overcome/alleviate the hardware overhead of caches, Scratch-Pad Memories (SPMs) evolved for energy-constrained embedded systems [19]. Instead of providing hardware support for mapping data/code from off-chip to on-chip memory, SPM allows designer/compiler to perform content management saving up to 30% of energy compared to complete caches under certain operating scenarios [19][1]. *The challenge is to efficiently utilize the SPMs.*

Considering the above-discussed memory issues of HEVC, general-purpose techniques for SPM management [20]-[22] may not be energy efficient. Recent trends demonstrated benefits of application-specific SPMs management for low-power H.264 video encoding for single core or ASIC-based systems [4]-[7]. However, these works lack support for many-cores which are more memory restrictive and do not address memory contention in private vs. shared memories for cores synchronization. Moreover, these works do not account for the novel coding model of the advanced HEVC that can be leveraged to achieve even higher energy savings as we will motivate in Section I.B.

In summary, *there is a strong need for application-specific memory design targeting energy-efficient high efficiency video encoding on embedded multi-/many-core platforms*. Our goal is to leverage the application-specific characteristics of the emerging HEVC standard to increase the potential of energy savings.

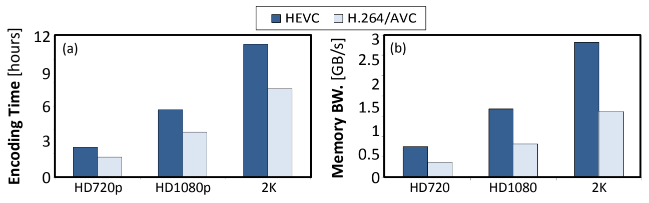


Fig. 1 HEVC vs. H.264/AVC encoder (a) encoding time; (b) memory accesses. (average results for commonly used test sequences [13], 128x128 search window, H.264/AVC and HEVC test models, 300 frames)

Before moving further, we will present basics of HEVC to the level of details necessary to understand our novel contribution.

A. *Overview of HEVC Coding Tools and Related Memory Issues*

To facilitate parallelization with minimal quality loss, the standardization committee (JCT-VC) introduced the novel concept of *Tiles*[2] in HEVC, which is different from slices that are used for video streaming [17]. Tiles divide one video frame into rectangular regions that can be coded independent of each other, thus increasing the thread level parallelism [15][16]. Fig. 2 presents an example of 4-Tile partitioning. Each Tile is assigned to a specific core *without* any data dependency with another Tile processing.

The inter-frame prediction with Motion Estimation (ME) is the most complex processing step in the HEVC encoder as it corresponds to >80% of the computation time and energy consumption of HEVC encoders. ME searches the best match of a block from the current frame in a set of so-called reference frames[3]. The search is performed in a restricted *search window*. The reference frames are typically stored in the external/off-chip memory while the search windows are stored in on-chip memories. Due to the memory management for fetching the search window samples from the off-chip and increased leakage energy for keeping them in the on-chip memory, the ME becomes the most

[1] *Examples:* IBM Cell Processor [23], ARM10E [24], TI TMS370CX7X [25], etc.
[2] These are video Tiles, i.e. different from hardware tiles in many-core processors.
[3] These are previously coded and reconstructed frames.

978-3-9815370-2-4/DATE14/©2014 EDAA

memory–intensive processing block [4]-[7]. As a result, 70%-90% of the ME energy is spent in the off-chip and on-chip memories (leakage and dynamic) [4]-[7]. Furthermore, multiple Tiles amplify the memory pressure since more data must be fetched/stored during the same time instant.

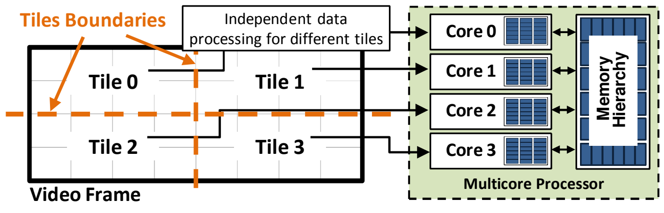


Fig. 2 Video Tiles and multicore organization for parallel HEVC.

Another novel coding tool of the HEVC that aggravates the memory problem is variable-sized *Coding Units* (CUs). The HEVC decision is based on a quad-tree structure (see Fig. 3). The root for the decision is the 64x64 CU, called *coding-tree block* (CTB). The encoder is responsible for deciding what is the best partitioning for the current CTB that provides the best coding efficiency, in terms of bitrate and coded video quality. Current HEVC draft also supports 32x32, 16x16 and 8x8 CU sizes [17].

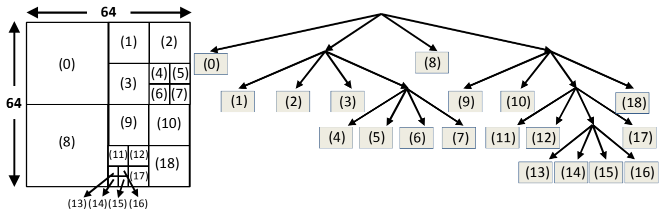


Fig. 3 An example of HEVC coding tree unit organization

B. Goals and Motivational Analysis

The main goal of our work is *to leverage the application-specific knowledge of the emerging HEVC standard (i.e. its new coding tools) and video content properties to develop an energy-efficient SPM-based on-chip video memory*. The key is (1) to analyze and exploit the memory access behavior in the video Tiles-based processing; and (2) the overlapping reference samples that define the shared access patterns for different cores processing different Tiles. The samples near to the tile boundaries in the reference frame must be fetched/stored by multiple cores, leading to external memory contention, redundant memory access and extra on-chip storage (causing energy wastage). An example in Fig. 4a depicts the overlapping accesses performed for more than one tile processing core (gray and black regions).

In the following, we highlight important memory related issues during the Tile-based HEVC processing with the help of our experimental case study and expose the potential of application-specific optimization with the help of several observations.

Analysis-1: The overlapping regions tend to grow for an increased number of Tiles (assuming 1 Tile per core). The overlap size trend is plotted for growing number of Tiles in the Fig. 4b. In the worst case, the overlap reaches 50% in a 16-core HEVC encoder. As larger is the overlapping area, more cores must concurrently access the same reference data from the external memory without any data reuse. Therefore, *it may be beneficial to design dedicated SPMs for the overlapping regions to avoid external memory retransmission of the Tiles shared reference data, saving off-chip memory energy*.

Analysis-2: Although the ME is performed within a search window, the search algorithm may not require all the samples. For instance, the TZ search algorithm in the HEVC software [14] does not necessarily explores the entire search window analysis [6]. Moreover, adaptive ME algorithms feature changing centering of the search window depending upon the already coded neighboring CUs. As a result, the *Tiles overlap shape may substantially vary according to the video content* as shown in Fig. 5. Furthermore, the *samples inside the overlapping regions have different access intensities*. It shows that, not all parts of the on-chip video memory (storing the overlapping samples) will be accessed for every CU depending on the video content. Even for the accessed sectors, the access distribution is different depending on the video content characteristics. Therefore, *the key is to leverage the overlapping memory access knowledge to predict the unused or less-frequently used memory sectors for adaptive power management of the SPMs.*

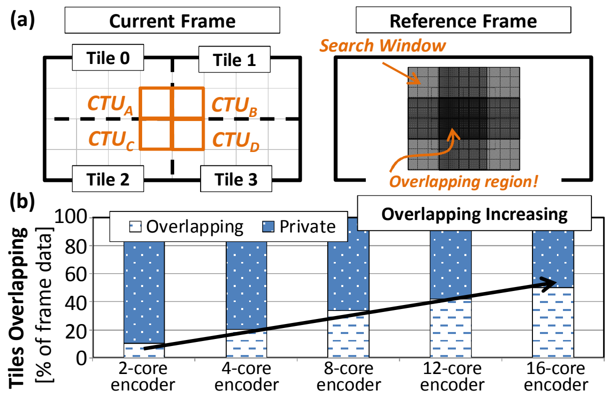


Fig. 4 (a) Example of Tile partitioning and of the overlapping problem; (b) Evaluation of overlapping accesses for different number of Cores (HD1080p; "BasketballDrive" sequence; 127x127 search window)

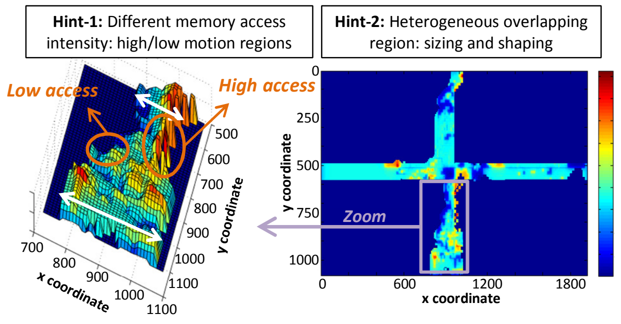


Fig. 5 Distribution of the overlapping samples (HD1080p; "BasketballDrive" sequence; 127x127 search window)

C. Our Novel Contributions

We propose an energy efficient distributed Scratchpad video memory architecture (dSVM) for the next-generation High-Efficiency Video Coding (HEVC) exploiting the video Tiles based parallel processing on multi/many-core processors. It employs:

- **A Distributed Scratchpad Video Memory Architecture (Section III)** that integrates several *private and overlapping (shared) SPMs* to support intra-Tile and inter-Tiles data reuse, respectively, among various cores. We develop a scheme that leverages the offline statistical analysis of HEVC and video content to size and design the organization of SPMs. A reading policy is designed for energy-efficient data fetching.
- **Adaptive Power Management of dSVM (Section IV)** that takes into account the size and the shape of the predicted overlapping area to select appropriate sleep states for different regions of private and overlapping SPMs.

We evaluate the energy efficiency of our dSVM architecture for various recommended test video sequences for different number of Tiles.

To the best of authors' knowledge, this is the first work towards energy-efficient on-chip memory hierarchy for the emerging Tile-based parallel HEVC encoders.

II. MEMORY MODELS AND NOTATIONS

Every data transmission from/to memory is based on a fixed *basic access unit* (BU), which corresponds to a $BU_{Size}*BU_{Size}$ picture block. When external memory communication is required, then *several BUs are accessed in one burst operation* to increase the energy efficiency.

On-Chip SRAM Organization Model: We adopt a bank-based partitioning Scratchpad memory (SPM) model to allow for parallel data accesses; see Fig. 6. Each SPM is composed of N_B number of banks. To facilitate parallel reading, different rows of a BU are stored in parallel banks. A bank B_i is composed of N_S sectors of size S_S. Each sector has N_L number of lines of size S_L.

Different sectors of the SPM can be individually power-gated using a multiple sleep-state transistor model supporting four power states [12]: *S0=OFF*, *[S1,S2]=Data Retentive* and *S3=ON*, where $E_{Static}(S0) < E_{Static}(S1) < E_{Static}(S2) < E_{Static}(S3)$. Still, each state have also increasing associated wake-up energies ($WE(S0) > WE(S1) > WE(S2) > WE(S3) = 0$).

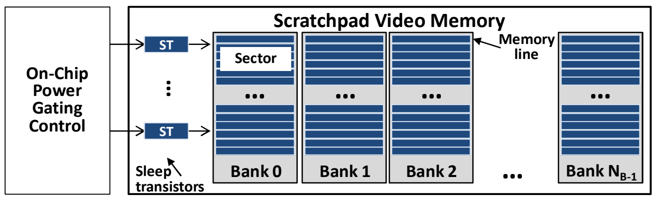


Fig. 6 Organization model of our scratchpad video memory

Off-Chip DDR DRAM Model: For energy estimation of the off-chip memory, we adopt the DDR (Dual Data Rate) DRAM model depicted in [9]-[10]. The total power is derived by the composition of six components: (1) page activation energy (E_{ACT}), (2) write energy (E_{WR}), (3) read energy (E_{RD}), (4) I/O pins energy (E_{DQ}), (5) refresh energy (E_{REF}), and (6) standby energy (E_{STBY}). In the experimental analysis, we assume that the memory will always operate in the active state and the standby energy will be equivalent to the E_{ACT_STBY} component.

III. ARCHITECTURE OF OUR DISTRIBUTED SCRATCHPAD VIDEO MEMORY

Fig. 7 depicts the block diagram of our distributed Scratchpad video memory architecture (dSVM) for multi-core HEVC encoding. Each core[4] is assigned the processing of one out of the *n* video Tiles. The SPMs are used to store different parts of the reference frame used for ME or other encoding blocks. We propose two levels of SPMs:

1) A core-private SPM (PrivSPM) to store the search window data corresponding to each CU for intra-Tile data reuse, and
2) A core-shared SPM (OvSPM) to store the Tiles overlapping reference data for inter-Tiles data reuse.

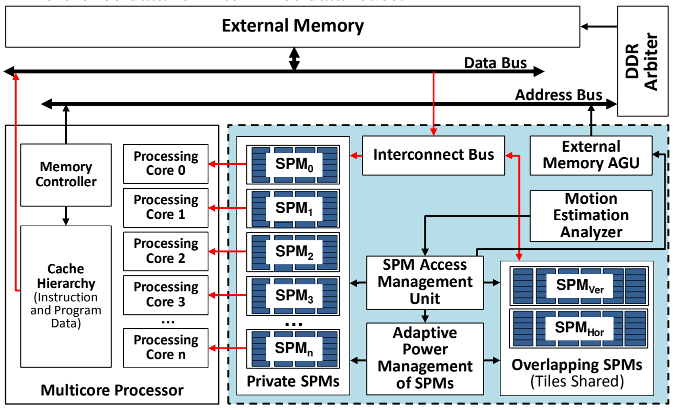


Fig. 7 Our dSVM architecture integrated in a HEVC encoder

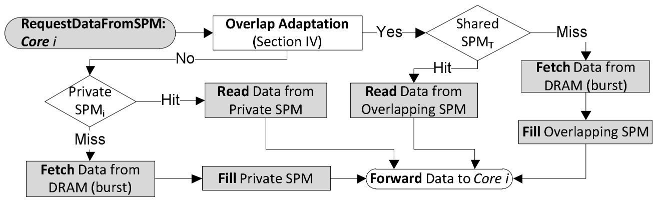
[4] A core has a small private instruction and data cache to store the program code and common data (like variables). The SPM is used for large data like reference frame.

Each core sends the search window data requests to the SPM access management module using the vertical/horizontal frame coordinates. The SPM access management unit will schedule the memory accesses according to our DRAM/SPM reading policy (see Section III.A). The overlap patters and size is extracted by the Motion Estimation Analyzer and forwarded to (i) the SPM access management module to map the overlapping region to the on-chip *OvSPMs*; and (ii) the adaptive power management unit for selecting an appropriate sleep state for the idle SPM regions. If external memory access is required, the frame positions are translated to physical DRAM memory position addresses by the Address Generation Unit (External Memory AGU in Fig. 7). The adaptive power management unit analyzes the Tiles overlap size to adaptively predict the less-likely accessed or idle memory sectors of the *PrivSPMs* and *OvSPMs* and to select an appropriate sleep state in order to save SPM leakage energy.

In the following sections, we detail the SPM access management module, SPM sizing and design, and adaptive power management policy.

A. SPM Access Management Unit: Reading Policy and External Memory Arbitering

Our SPM access management unit implements the memory reading policy (see flowchart of Fig. 8) that takes advantage from the tiles overlap to increase the data-reuse of the reference frames samples. If a core *i* requests data from the SPM memory organization, as the first step, the SPM access management unit checks along with the overlap prediction if the requested data potentially belongs to one tiles overlapping region. Assuming that the data is inside an overlap related to the tiles intersection *T*, the corresponding cores-shared $OvSPM_T$ is then accessed. In this case, the inter-Tiles data reuse is exploited, since all tiles that share the tile boundary *T* may request the same data. For non-overlapping regions, the $PrivSPM_i$ is accessed, leading to intra-Tile data reuse. Note that for each core data request, either the shared ($OvSPM_T$) or the private ($PrivSPM_i$) memory is accessed. In the case of a hit, the data is simply forwarded to core *i*. In case of a miss, the data must be fetched from the external memory and forwarded to the core *i*. For improved energy efficiency, the SPM access management unit requests a burst of samples from the DRAM memory, which reduces the DRAM page activation energy and amortizes the initial latency for memory random access [7]. Furthermore, the corresponding SPM is filled with the fetched data. To handle parallel accesses to the *OvSPM*, we employ a priority based scheduling.



RequestDataFromSPM: Core i
Overlap Adaptation (Section IV)
Yes
No
Shared SPM$_T$
Miss
Hit
Fetch Data from DRAM (burst)
Fill Overlapping SPM
Private SPM$_i$
Hit
Miss
Read Data from Private SPM
Read Data from Overlapping SPM
Fetch Data from DRAM (burst)
Fill Private SPM
Forward Data to Core i

Fig. 8 Flow of our SPM access management unit with the reading policy

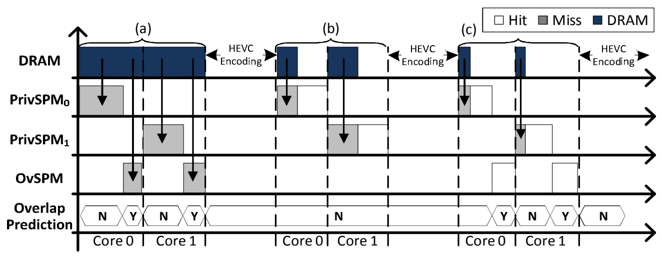


Fig. 9 An example of data interaction for a 2-core system

An Example: Fig. 9 illustrates an example for our memory reading policy in three different cases for a 2-core encoding system.

a) In the beginning, the on-chip SPMs are empty and each request will lead to external memory fetching (*OvSPM* and *PrivSPM* misses). Fig. 9 shows that the overlap prediction is analyzed to determine whether the reference data is stored in the $PrivSPM_i$ or in the $OvSPM_T$. During the frame processing, due to the intra-Tile and inter-Tiles reused data, more hits occur and even less external memory communication is needed.
b) The second case in Fig. 9 depicts tile-centering CUs processing where only the *PrivSPMs* is accessed (i.e. only intra-Tile data reuse).
c) The last case shows the best case of energy efficiency, where memory hits are observed for both *PrivSPMs* (i.e. intra-Tile data reuse) and *OvSPMs* (i.e. inter-Tiles data reuse) accessing.

B. *Design of Scratchpad Video Memories*

A key challenge is to determine an appropriate size and organization of different SPMs (*PrivSPMs* and *OvSPMs*) to optimize for leakage and dynamic energy. We propose an application-guided methodology that exploits the statistical analysis of memory access behavior Tile-parallelized HEVC in order to increase the energy efficiency of our dSVM architecture.

Our methodology leverages the Tiles overlap behavior that depends on the search window size and the video motion properties. Adaptive ME algorithms change the center of their search windows by using spatial predictors (i.e., motion vectors of previously-coded CUs). Moreover, low motion CUs will lead to less search window usage. Hence, the optimal overlapping memory size for each video sequence follows a statistical distribution of the near-boundaries ME motion predictors. Fig. 10a depicts statistics of the tiles overlap varying the search window size. On average, the overlap linearly increases with the increase n the search range. The more or less concentrated distribution around the average size hints towards the video motion properties. Different regions near the tile boundaries have different motion characteristic, which leads to more or less memory access overlaps.

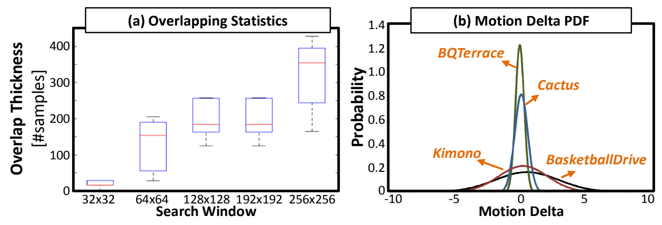


Fig. 10 (a) Overlapping statistics for increasing search window size for the "BasketballDrive" test video sequence;
(b) motion delta distribution for several test video sequences

To statistically define the motion property near a specific tile boundary of a given video, we define the Δ_{Motion} (motion delta) metric as being the *video correlated parameter used for determining the overlap size*, as presented in Fig. 11. For each frame of the video and for each defined Tile boundary, the algorithm obtains the used ME spatial predictors (*lines 7-8*). The difference of the predictors used by the near-boundary CUs from the two Tile boundary sides (*SideA* and *SideB*) is then calculated (*lines 12-17*). This difference will represent the access search range of *SideA* CUs in the *SideB* reference frame region, and vice-versa. The Probability Density Function (PDF) of the motion delta metric is then calculated (*line 20*), where μ_Δ and σ_Δ are the statistical average and standard deviation, respectively, of the motion delta parameters extracted from the video. The PDFs for HD1080p test sequences are plotted in Fig. 10b. We can note diverse behaviors depending on the input video: high motion videos like *BasketballDrive* and *Kimono* present more spread distributions, while low motion videos like *Cactus* and *BQTerrace* have more concentrated distributions. Using the motion parameter and the search window dimension, we define the Tiles overlap sizing formula for the overlap thickness ($Ov_{Thickness}$) and length (Ov_{Length}) in Eq. (1)-(2), respectively. The signal of the motion delta represents the video motion direction near the target tiles boundary. Negative values mean that we have opposite motion directions, which decreases the overlap size, while positive motion delta values increases the range of the overlap.

```
1.  determineMotionDelta(Video: V; TilePartitioning: TP):
2.  List_Δ = [ ];
3.  For all Frame ∈ V
4.    For all Tile_ID ∈ TP
5.      PredMap[Tile_ID] = [ ];
6.      For all CU ∈ Tile_ID
7.        CU.performMotionEstimation();
8.        PredMap[Tile_ID].insert(CU.getUsedPredictor());
9.      End For
10.   For all TileBoundary_ID ∈ TP
11.     //Let SideA and SideB the two tile boundary sides
12.     For all CU_SideA, CU_SideB ∈ TileBoundary_ID
13.       PredA := PredMap[Tile_SideA][CU_SideA][Coord_ID];
14.       PredB := PredMap[Tile_SideB][CU_SideB][Coord_ID];
15.       Δ_Value := |PredA – PredB|;
16.       List_Δ.append(Delta_Value);
17.     End For
18.   End For
19. End For
20. {μ_Δ, σ_Δ} = norm_dist(List_Δ);
21. return {μ_Δ, σ_Δ};
```

Fig. 11 Motion knowledge extraction for overlapping SPM sizing

$$\mathrm{Ov}_{\mathrm{Thickness}}(\mathrm{TileBoundary}_{\mathrm{ID}}) = 2 \times \mathrm{SW} + \Delta_{\mathrm{Motion}} \quad \text{where: } \Delta_{\mathrm{Motion}} = \mu_\Delta + 2 * \sigma_\Delta \tag{1}$$

$$\mathrm{Ov}_{\mathrm{Length}}(\mathrm{TileBoundary}_{\mathrm{ID}}) = \begin{cases} \mathrm{H}_{\mathrm{Frame}} & \text{if vertical} \\ \mathrm{W}_{\mathrm{Frame}} & \text{if horizontal} \end{cases} \tag{2}$$

Based on statistical evaluations and the memory organization model defined in the Section II, we determine the physical sizing for SPMs in our dSVM; see Eq. (3)-(7). For the overlapping data, each Tile boundary will leads to a specific *OvSPM* design. Our sizing formulation is based on the definition of the BU size (BU_{Size}), which is the smaller unit that can be accessed. For instance, a BU_{Size} equals to 16 means that the smaller data transmission unit is one 16x16 reference block. The BU size is a design decision for efficient power management depending on the adopted search window dimension. One BU in the overlap is mapped to a specific memory line (composed of $OvSPM_{SL}$ bytes) along the $OvSPM_{NB}$ memory banks. Each *OvSPM* sector groups specific rows of the BUs along the overlap thickness ($OvSPM_{SS}$). One entire line of BUs is completely stored into a group of same positioned sectors along the $OvSPM_{NB}$ memory blocks. In total, each *OvSPM* has $OvSPM_{NS}$, to store the complete overlapping data.

$$\mathrm{N}_{\mathrm{OvSPM}} = \mathrm{N}_{\mathrm{TilesBoundaries}} \tag{3}$$

$$\mathrm{OvSPM}_{\mathrm{S_L}} = \mathrm{BU}_{\mathrm{Size}} \tag{4}$$

$$\mathrm{OvSPM}_{\mathrm{N_B}} = \mathrm{BU}_{\mathrm{Size}} \tag{5}$$

$$\mathrm{OvSPM}_{\mathrm{S_S}} = \lceil \mathrm{Ov}_{\mathrm{Thikness}}/\mathrm{BU}_{\mathrm{Size}} \rceil * \mathrm{S_L} \tag{6}$$

$$\mathrm{OvSPM}_{\mathrm{N_S}} = \mathrm{OvSPM}_{\mathrm{N_B}} * \lceil \mathrm{Ov}_{\mathrm{Thikness}}/\mathrm{BU}_{\mathrm{Size}} \rceil \tag{7}$$

The *PrivSPM* stores the search window samples, as expressed in Eq. (8)-(12). The data organization is similar to that presented for the *OvSPMs* except that the *PrivSPM* must store core-private search window instead of Tile overlaps.

$$\mathrm{N}_{\mathrm{PrivSPM}} = \mathrm{N}_{\mathrm{Tiles}} \tag{8}$$

$$\mathrm{PrivSPM}_{\mathrm{S_L}} = \mathrm{BU}_{\mathrm{Size}} \tag{9}$$

$$\mathrm{PrivSPM}_{\mathrm{N_B}} = \mathrm{BU}_{\mathrm{Size}} \tag{10}$$

$$\mathrm{PrivSPM}_{\mathrm{S_S}} = \lceil \mathrm{SW_H}/\mathrm{BU}_{\mathrm{Size}} \rceil * \mathrm{S_L} \tag{11}$$

$$\mathrm{PrivSPM}_{\mathrm{N_S}} = \mathrm{PrivSPM}_{\mathrm{N_B}} * \lceil SW_V/BU_{Size} \rceil \tag{12}$$

IV. ADAPTIVE POWER MANAGEMENT OF SPMS

In case where the overlap size is reduced when low motion is captured around the tiles boundary, we propose an adaptive power management scheme for the *OvSPM* in our dSVM architecture to reduce its leakage energy. Furthermore, *PrivSPMs* are less accessed when CUs near the Tile boundaries are encoded since most memory requests are actually performed in the *OvSPMs*. Therefore, our scheme power-gates the *PrivSPMs* regions that are not accessed due to the overlap intersection.

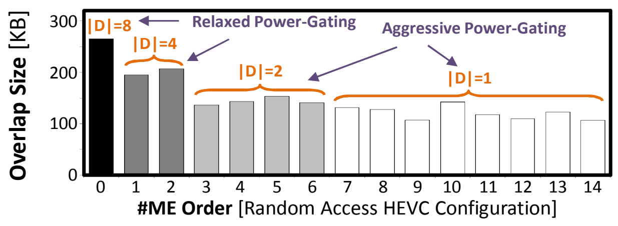


Fig. 12 Overlap sizing variation for several temporal distances (D factor).

To capture the impact of temporal distance for overlap size prediction, we define a term "D" as the distance between the current and reference frames. This distance directly affects our overlap prediction. More distant frames (i.e. high D values) lead to high overlap size due to more intense motion activity. Smaller overlaps can be noted when D is decreasing. Fig. 12 illustrates the overlap size for MEs with different D factors. Our power management selects an appropriate sleep state according to the motion behavior: relaxed power-gating (i.e. putting idle sectors in data retentive modes) is used when we have high motion overlaps. In case of low motion overlaps, aggressive power-gating (i.e. putting sectors in power-OFF mode) is applied to save more leakage energy.

```
1.  managePowerOverlapSPM (Frame: $F_{Current}$, $F_{Reference}$;
    TileBoundary: $TileBoundary_{ID}$)   //frame-level management
2.  currOverlapUsage := 0;
3.  {$\mu_{Usage}$, $\sigma_{Usage}$} := getOverlapUsages();   //run-time statistics
4.  $D_{ME}$ := getPoc($F_{Current}$) – getPoc($F_{Reference}$);   //overlap prediction (lines 3-4)
5.  PredOv($TileBoundary_{ID}$) := $\begin{cases} \mu_{Usage} - \sigma_{Usage} & \text{If } D_{ME} = 1 \\ \mu_{Usage} & \text{If } 2 \geq D_{ME} \geq 3 \\ \mu_{Usage} + \sigma_{Usage} & \text{If } 4 \geq D_{ME} \geq 7 \\ \mu_{Usage} + 2.\,\sigma_{Usage} & \text{If } D_{ME} \geq 8 \end{cases}$
6.  $PowerMap_{Ov}$(x,y) := $\begin{cases} S0 & \text{If } (x,y) \in \text{to predicted overlap} \\ S3 & \text{otherwise} \end{cases}$
7.  For all CTU ∈ {$Tile_0$, $Tile_1$, ..., $Tile_{n-1}$}   //CTU-level management
8.      SearchLimits := getSearchLimits(CTU);
9.      $PowerMap_{Ov}$(x,y) := $\begin{cases} PowerMap_{Ov} & \text{If } (x,y) \notin (\text{PredOv} \cap \text{SearchLimits}) \\ S1 & \text{Else If } (x,y) \text{ shared by 2 tiles} \\ S2 & \text{Else If } (x,y) \text{ shared by} > 2\ tiles \end{cases}$
10.     SPM[$TileBoundary_{ID}$].powerGate($PowerMap_{Ov}$);
11.     currOverlapUsage += performMotionEstimation();
12. End For
13. store(currOverlapUsage);
14. return;
```

Fig. 13 Adaptive power management policy for the Overlapping SPM.

```
1.  managePowerPrivateSPM(Frame Tile: $Tile_{ID}$)
2.  (∀ (x,y) ∈ $PowerMap_{SW}$, $PowerMap_{SW}$(x,y) := S3);
3.  For all CTU ∈ $TIle_{ID}$   //CU level power-gating
4.      For all $TileBoundary_{ID}$ ∈ TilePartitioning
5.          $PowerMap_{SW}$(x,y):= $\begin{cases} S0 & \text{if } (x,y) \in \text{PredOv}(TileBoundary_{ID}) \\ PowerMap_{SW} & \text{otherwise} \end{cases}$
6.      End For
7.      $SPM_{Priv}$[$Tile_{ID}$].powerGate($PowerMap_{SW}$);
8.      performMotionEstimation();
9.  End For
10. return;
```

Fig. 14 Adaptive power management policy for the Private SPM .

Fig. 13 depicts our adaptive power management policy for the *OvSPM*. At frame level, online statistics of overlap SPMs usages for previous ME are generated (*line 2*). As shown in Fig. 12, using the "D" factor of the current ME as parameter, we predict the current overlap size (*line 5*). For all SPM lines outside the predicted overlap, the *OFF* state (S0) is assigned to the *$PowerMap_{Ov}$* corresponding position; otherwise, the ON state is assigned (S3). In CTU processing level (*line 7),* our power management checks for the non-accessed *OvSPM* positions that are inside the overlap prediction to put them in *data retentive* states (*lines 8-9*). S2 state is assigned for positions potentially accessed by more than two Tiles, while S1 state is used for overlap positions shared for only two Tiles. The overlap usage for the current ME is updated at every CTU processing (*line 11*) and saved to be used for future overlap predictions (*line 13*).

The adaptive power management policy for the *PrivSPMs* is depicted in Fig. 14. At the beginning of a CTU processing, it checks for intersected positions between the core-private search window and any predicted overlap. For each intersection, it power-gates the corresponding PrivSPM positions *(line 5* in Fig. 14). Note, both *OvSPM* and *PrivSPM* managements work in parallel in our dSVM system.

V. Experimental Results

A. Experimental Setup

The experimental analysis is based on the recommended HEVC common test conditions [13] using the HEVC test model (HM 11.0) [14]. We execute the experiments for 4-Tile and 8-Tile scenarios (each Tile executes on a dedicated core) with five search window dimensions: 64x64, 96x96, 128x128, 192x192, and 256x256. Six test video sequences with different properties were evaluated: *BasketballDrive (BDrive)*, *BQTerrace*, *Cactus* and *Kimono* (HD1080p: 1920x1080), *PeopleOnStreet (People)* and *NebutaFestival (Nebuta)* (2K: 2500x1600). Other encoder specifications are: GOP=8, CABAC, FRExt, Random Access configuration, and TZ Search algorithm.

For memory energy evaluation, we use the CACTI 6.5 leakage/dynamic energies estimation for a 32nm SRAM-based SPM. The leakage reduction and wake-up energies were derived from the analytical model presented in [12]. The 4-Gbit Low-Power DDR2 (LPDDR2) DRAM MT42L128M16D1GU-25WT electrical specifications [8] were used to determine all external memory energy components mentioned in Section II. As a design decision for combined coarse- and fine-grained SPM management, considering the most widely used video resolutions and search window sizes (as listed above), we adopt *BU_{Size}=16*.

To evaluate the savings of our dSVM architecture, we select two other comparison partners: (a) SPMs with Level C-based data reuse for each core, and (b) our dSVM with only the PrivSPMs and no shared OvSPMs. The energy evaluations consider the first 30 consecutive frames of each test video sequence.

B. Energy Savings

Tab. 1 presents the overall energy evaluation with a breakdown of off-chip and on-chip memory energy consumption.

Tab. 1 Overall Energy Consumption Evaluation

	SPMs Size [KB]	On-Chip Energy [mJ]	Off-Chip Energy [mJ]	Overall Energy [mJ]	Savings dSVM [%]
***Scenario 1:** 4-Tile HD1080, 129x129 search window*					
Level C [11]	144	16	587	603	36%
Our PrivSPM Only	144	16	469	485	21%
Our dSVM	614	33	351	384	-
***Scenario 2:** 8-Tile HD1080, 129x129 search window*					
Level C [11]	288	33	587	620	61%
Our PrivSPM Only	288	32	462	494	51%
Our dSVM	1098	63	179	242	-

Tab. 1 shows that our complete dSVM architecture provides the best energy efficiency for the two tested scenarios. Considering the accumulated size of SPM blocks (private plus overlapping), the dSVM architecture presents the highest memory usage. However, our adaptive power management is able to significantly reduce the leakage consumption and accordingly adapting the power states to the predicted overlap size and shape. Therefore, the dSVM architecture can reduce the on-chip energy consumption being competitive with the related non-shared memories approaches. Furthermore, this slight on-chip energy overhead is amortized by significant savings in the external memory transfers that leads to overall savings of 21%-36% compared to Level C and our PrivSPM Only solution (scenario 1), respectively. In the scenario 2, our energy savings even increase to 51%-61% compared to Level C and our PrivSPM Only solution, respectively. Note that our dSVM architecture provides increasing overall savings when more Tiles (i.e. parallel HEVC threads) are used (2x higher savings, on average). Extrapolating our results for more than 8 video Tiles (as more inter-Tiles data reuse potential can be exploited), our dSVM can achieve even higher memory energy savings.

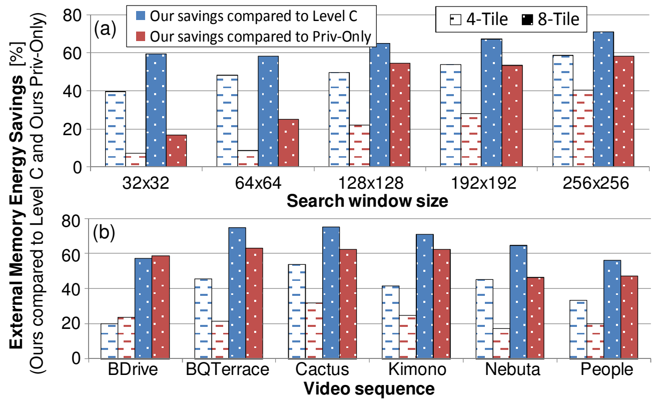


Fig. 15 External memory energy savings for 4-tile and 8-tile scenarios: (a) average savings for all sequences varying the search window size and (b) savings for each tested sequence (128x128 search window size)

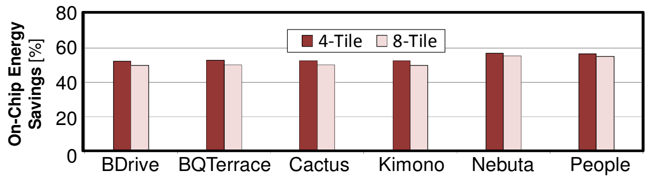


Fig. 16 Leakage energy savings due to our dynamic power management of the dSVM architecture (128x128 search window size)

Off-Chip Memory Energy Savings: Fig. 15 depicts the external memory energy savings of our dSVM for different search window sizes, input video test sequences, and the number of parallelized Tiles. Fig. 15 shows that as more Tiles are used, more external memory energy is saved due to the larger overlap. Our dSVM architecture supersedes other comparison partners by exploiting our novel concept of *both* intra-Tile and inter-Tiles data reuse. The dSVM savings increase with the growing search window from 7% to 58% for the 4-Tiles partitioning and from 17% to 71% for the 8-Tiles partitioning. This due to exploiting the shared memory accesses coming from different processing cores. Furthermore, there are savings also vary depending upon the video content: low motion videos leads to less overlap and less potential of reduction. In the best case, the *Cactus* sequence achieves an external memory energy reduction of 55% and 74% for 4-Tiles and 8-Tiles partitioning (using 128x128 search window size).

On-Chip Memory Energy Savings: Fig. 16 depicts the on-chip leakage energy savings of our dSVM architecture due to our adaptive power management policy. On average, our policy reduces the leakage energy by 54% and 52%, considering 4-Tile and 8-Tile scenarios. Part of the savings is related to the *PrivSPMs* energy management, which captures the intersections of the search window positions with the any predicted overlap. Regarding the *OvSPMs*, our scheme can significantly reduce the leakage energy for low motion ME, where the overlap tends to be small.

VI. CONCLUSION

This work presented a distributed Scratchpad Video Memory Architecture for the next-generation parallel High Efficiency Video Coding. It exploits intra- and inter- video Tile level data reuse jointly through private and shared SPMs of different cores executing parallel HEVC threads. The SPM design is based on application-specific knowledge of HEVC and statistical analysis of memory access behavior w.r.t. the video content properties. To further reduce the leakage energy, we integrated an adaptive power management policy for SPMs that exploit the prediction of the overlapping accesses from different cores and their relationship to the video content properties. Our dSVM architecture provides up to 61% reduction in the overall memory energy and 54% in the leakage energy compared to state-of-the-art. Our proposed contribution enables energy-efficient multimedia systems supporting multiple threads of the next-generation HEVC encoder.

ACKNOWLEDGMENTS

This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89); http://invasic.de and partly supported by DAAD and CAPES as part of the PROBRAL project "VideoArch^{3D} - Power-Efficient Techniques for 3D-Multimedia".

REFERENCES

[1] B. Bross, W. J. Han, J. R. Ohm, G. J. Sullivan, T. Wiegand, “High Efficiency Video Coding (HEVC) text specification draft 7”, May 2012.
[2] D. Marpe, et al, “Video compression using nested quadtree structures, leaf merging, and improved techniques for motion representation and entropy coding,” IEEE TCSVT, vol. 20, no. 12, pp. 1676–1687, 2010.
[3] B. M. T. Pourazad, C. Doutre, M. Azimi, P, Nasiopoulos, “HEVC: The New Gold Standard for Video Compression: How Does HEVC Compare with H.264/AVC?,” IEEE CEM, pp. 36-46, 2012.
[4] B. Zatt, M. Shafique, F. Sampaio, L. Agostini, S. Bampi, J. Henkel, "Run-time adaptive energy-aware motion and disparity estimation in multiview video coding", IEEE/ACM/EDA DAC, pp. 1026-1031, 2011.
[5] M. Shafique, B. Zatt, F. L. Walter, S. Bampi, J. Henkel, “Adaptive Power Management of On-Chip Video Mamory for Multiview Video Coding”, IEEE/ACM/EDA DAC, pp. 866-875, 2012.
[6] B. Zatt, M. Shafique, S. Bampi, J. Henkel, “A Low-Power Memory Architecture with Application-Aware Power Management for Motion & Disparity Estimation in Multiview Video Coding”,IEEE/ACM ICCAD, pp. 40-47, 2011.
[7] F. Sampaio, B. Zatt, M. Shafique, J. Henkel, S. Bampi, “Energy-Efficient Memory Hierarchy for Motion and Disparity Estimation in Multiview Video Coding”, IEEE/ACM DATE, pp. 665-670, 2013.
[8] Micron. “4Gb: x16, x32 Mob. LPDDR2 SDRAM S4”. Rev. N 05/13 EN, 168p, 2013.
[9] Micron. “TN-46-03 – Calc. DDR Mem. System Power”. Rev. B 3/05 EN, 26p, 2005.
[10] Micron. “TN-46-12: Mob. DRAM Power-Sav. Features/Calc.”, 10p, 2009.
[11] C.-Y. Chen, C.-Y. Chen, C.-T. Huang, L.-G. Chen. “Level C+ Data Reuse Scheme for Motion Estimation with Corresponding Coding Orders”, IEEE TCSVT, vol. 16, no. 4, p. 553-558, 2006.
[12] H. Singh, L. Agarwal, D. Sylvester, K.J. Nowka, "Enhanced leakage reduction techniques using intermediate strength power gating", IEEE TVLSI, vol. 15, no. 11, pp. 1215-1224, 2007.
[13] F. Bossen, “Common test conditions and software reference configurations”, ITU-T/ISO/IEC JCTVC-K1100, October 2012.
[14] JCT-VC. HEVC Software SVN, 2011. Available in: <https://hevc.hhi.fraunhofer.de/>
[15] Misra, K.; Segall, A.; Horowitz, M.; Xu, S.; Fuldseth, A.; Zhou, M., "An overview of tiles in HEVC," IEEE JSTSP, no.99, 2013
[16] C. Blumenberg, D. Palomino, B. Zatt, S. Bampi. “Adaptive Content-Based Tile Partitioning Algorithm for the HEVC Standard”, PCS, p. 185-188, 2014.
[17] JCT-VC, “High Efficiency Video Coding (HEVC) text spec. draft 10”, 2013.
[18] Iyengar, A., "Design and performance of a general-purpose software cache," IEEE IPCCC, vol., no., pp.329,336, 1999.
[19] R. Banakar, S. Steinke, B.-S. Lee, M. Balakrishnan, P. Marwedel, “Scratchpad Memory: Design Alternative for Cache on-chip Memory in Embedded Systems,” CODES+ISSS, pp. 73–78, 2002.
[20] K. Bai and A. Shrivastava, “Automatic and efficient heap data management for limited local memory multicore architectures,” IEEE DATE, 2013, 2013, pp. 593-598.
[21] N. Deng, W. Ji, J. Li, F. Shi, and Y. Wang, “A Novel Adaptive Scratchpad Memory Management Strategy,” IEEE RTCSA pp. 236–241, 2009.
[22] B. Egger, S. Kim, C. Jang, J. Lee, S. L. Min, and H. Shin, “Scratchpad Memory Management Techniques for Code in Embedded Systems without an MMU” IEEE TC, vol. 59, no. 8, pp.1047-1062, 2010.
[23] IBM, “The Cell Project”, Last Accessed: Sep. 2013, <http://researcher.watson.ibm.com/researcher/view_project.php?id=2649>.
[24] ARM, “ARM10 Family: An Overview”, pp. 11, 2005
[25] Texas Instruments, “TMS370CX7X from Texas Instruments”, <www.ti.com/mcu/docs/mcuorphan.tsp?contentId=15364>.

Content-Driven Memory Pressure Balancing and Video Memory Power Management for Parallel High Efficiency Video Coding

Felipe Sampaio[1], Muhammad Shafique[2], Bruno Zatt[3], Sergio Bampi[1], Jörg Henkel[2]
[1]Informatics Institute, PPGC, Federal University of Rio Grande do Sul (UFRGS), Brazil
[2]Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
[3]GACI, PPGC, CDTec, Federal University of Pelotas (UFPel), Brazil
{felipe.sampaio, bampi}@inf.ufrgs.br, bzatt@inf.ufpel.edu.br, {muhammad.shafique, henkel}@kit.edu

ABSTRACT

We present a novel content-driven memory pressure balancing and video memory power management scheme for parallel High Efficiency Video Coding (HEVC). The key is to leverage the application-specific knowledge to balance the (instant) access pressure on Scratchpad-based Video Memories (SVMs) for parallelized video processing. Our scheme accurately predicts the memory requirements of each processing core based on monitored memory usage and leverages this knowledge to perform a categorization of different video regions. Afterwards, it employs an adaptive policy for memory pressure balancing by rescheduling encoding of different video blocks based on their categories. This balancing also facilitates our scheme to perform efficient power-gating of unused parts of SVMs. Experimental results show that our scheme reduces the variations in the memory pressure by 37%-83% when compared to the traditional raster scan processing for 4- and 16-core parallelized HEVC encoder. Our content-driven power management saves 56% (on average) of SVM leakage energy.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems; B.3.2 [**Design Styles**]: Cache memories

Keywords

On-chip memory, memory pressure reduction, application-specific optimization, HEVC, video coding, adaptivity, power management, low-power, energy.

1. INTRODUCTION AND RELATED WORK

Parallelization of video processing applications under stringent energy budget is a significant challenge for the next-generation embedded manycore multimedia systems. Moreover, the memory hierarchy consumes a significant portion of the chip footprint and power/energy in such systems. Meeting these constraints becomes quite intricate when considering the escalating complexity of emerging video coding standards, like HEVC [1].

The *High Efficiency Video Coding* (HEVC) standard [1] aims at providing 2x higher compression efficiency compared to that of the state-of-the-art H.264/AVC standard. To achieve this, HEVC introduces novel data structures and coding tools that increase the computational effort by 40% and memory requirements by >2x compared to H.264/AVC (see Figure 1a). To alleviate this increased computation, HEVC provides parallelization support in form of *Video Tiles* that are independently processed on different cores. However, this further complicates the memory design in an embedded multimedia system through the following means: (1) More on-chip video memories are required to feed the processing cores that incur an increase in the leakage and dynamic energy. (2) External memory pressure is increased since multiple cores try to access the data at the same time, thus also leading to an increase in the off-chip energy. Moreover, the number of scenarios with unbalanced memory pressure may increase due to the run-time variation of the video content (as shown in Figure 1b) that lead to high instant power dissipation and may surpass the maximum available memory bandwidth. Therefore, *it is crucial to balance the memory pressure while performing efficient power management of video memories in parallel HEVC encoding*.

Recently, the use of scratchpad memories has proliferated in the manycore systems (like in IBM Cell [10]) as power-efficient on-chip memories to complement or replace large-sized shared caches [8]. The scratchpad memories avoid energy overhead of tags and write replacement management to provide >30% energy reduction compared to a full cache design [8]. Power efficient management of these scratchpad memories is of key importance. External memory pressure and on-chip scratchpad memory management for high-performance manycore systems have been explored in [12][13]. However, these works do not account for the application-specific properties, thus may not be efficiently employed for on-chip video memories. From the application-driven perspective, several works proposed dedicated power management schemes for video encoding regarding both off-/on-chip video memories [2]-[4]. However, these works lack support for *parallel* HEVC video encoding and corresponding memory constraints. Therefore, these techniques may perform inefficient under scenarios with (1) unbalanced memory pressure during parallel HEVC encoding (as we motivate in Section 1.2); and (2) simultaneously accessed multiple on-chip memories.

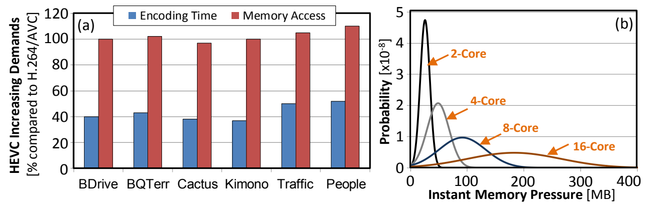


Figure 1. (a) HEVC increasing demands compared to H.264/AVC (b) memory pressure Probability Density Function for *BasketballDrive*.

Summarizing, *the challenge is to obtain balanced pressure for off- and on-chip memories based on multiple Scratchpad-based Video Memories (SVMs) used by different Video Tiles in parallel HEVC encoding and to provide efficient SVM power management by exploiting this knowledge* .

Before proceeding further, we first provide preliminaries of HEVC.

1.1 HEVC Preliminaries

The HEVC introduces the *Coding-Tree Unit* (CTU, e.g., a 64x64 block) as a basic encoding entity within a video frame. The CTU is divided using a recursive splitting into blocks of NxN or 2Nx2N sizes (e.g., 32x32, 16x16 and so on) [1]. An example partitioning is shown in Figure 2b. The Motion Estimation is performed for all possible blocks. For each block it searches for the most similar block within a *search window* in one or more reference frames (i.e. already encoded and reconstructed frames).

ISLPED'14, August 11 - 13 2014, La Jolla, CA, USA
Copyright 2014 ACM 978-1-4503-2975-0/14/08…$15.00.
http://dx.doi.org/10.1145/2627369.2627615

The search window is defined as the maximum range of motion search in both horizontal and vertical directions. This motion search process for multiple blocks may consume up to 90% of the total HEVC encoding energy [2]. Besides CTUs, HEVC supports rectangular *Video Tiles* (each containing multiple CTU) that can be processed in parallel without any data dependency [7]. Figure 2a presents an example of a 2x2 Video Tiles configuration (4-Tile scenario) for a video frame with 8x4 CTUs.

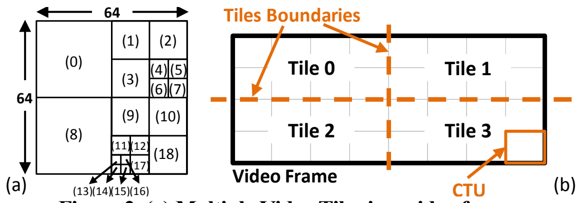


Figure 2. (a) Multiple Video Tiles in a video frame; (b) An example CTU partitioning.

1.2 Motivational Case Studies

We have performed an experimental analysis (see experimental setup in Section 4) for (1) memory pressure and access imbalance when processing multiple Video Tiles concurrently; (2) memory pressure and access correlations; and (3) Intra-Video Tile access behavior. These analyses provide a foundation for our novel contributions.

1) Memory Pressure Analysis: We define memory pressure as the memory access requirement caused by a CTU processing during a specific time. When considering multiple processing cores, the memory pressure may be (1) core-specific, or (2) accumulated (sum of all core-specific pressures). Typically, the motion estimation is performed in the traditional raster scan order (i.e., from top-left to bottom-right corner in row-by-row order). However, this may lead to unbalanced external memory pressure, as depicted in the 4-Tile example of Figure 3a. The maximum and minimum memory pressure peaks can be seen in Figure 3b. There are significant memory access variations compared to the average access case (that typically does not happen). This unbalanced memory pressure leads to high power peak dissipations and high instant memory bandwidth requirements, which may surpass the maximum availability constraints. Moreover, such unbalancing also leads to inefficient memory power management due to (1) fluctuations in the sleep durations, (2) frequent P_{ON}-P_{OFF} switching, and (3) memory usage prediction errors due to sudden access variations. Therefore, *the key is to leverage application specific-properties to adapt and re-schedule the CTU processing in order to achieve the best possible memory pressure balancing.*

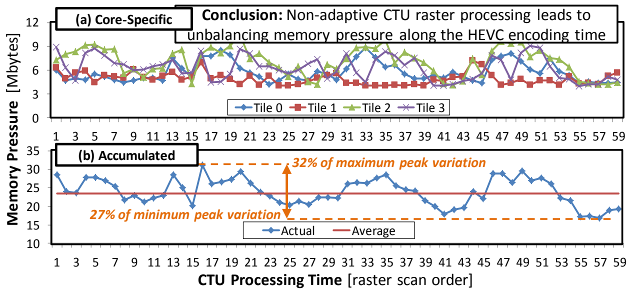


Figure 3. Memory pressure for (a) each processing core; and (b) accumulated and average cases for *BasketballDrive*.

2) Spatial/Temporal Neighboring Analysis: If it is possible to accurately predict the memory requirements for a given CTU, it can be exploited by a power manager to balance the memory pressure in a very efficient way. In case of high frame rates (30-60 fps), significant temporal correlation exists, i.e. the neighboring frames have similar memory access behavior, as depicted in Figure 4. Additionally, high video frame resolutions (e.g., FullHD=1920x1080 to 4K=3840x2160) increase the spatial correlations between neighboring CTUs within the same frame. Furthermore, we can note that the memory pressure for each CTU also depends upon their corresponding video content characteristic (like texture and motion content). Therefore, *the key is to leverage the knowledge from the monitored memory pressure of spatially- and temporally-neighboring CTUs to obtain a high quality prediction of the actual memory pressure for a given CTU.*

3) Intra-Video Tile Memory Analysis: While balancing the memory pressure is important from the external memory perspective, it is also crucial to take care of the core-private on-chip SVMs. In this case, long sleep durations (and consequently more leakage energy savings) can be achieved by consecutively encoding CTUs with similar video content properties (like texture and motion), thus similar memory pressure. Figure 5 shows Video Tiles with *less* memory requirements (like Video Tile 1) and *more* memory demands (like Video Tile 2). In this case, longer sleep durations and higher energy savings can be obtained for the SVM of core processing the Video Tile 1. Furthermore, re-scheduled CTU processing orders for a well-balanced memory pressure tends to group similar properties CTUs to be consecutively encoded, providing even higher sleep durations (as we will demonstrate in Section 3). Hence, the *key challenge here is how to leverage the CTU re-schedule for memory pressure balancing and increased sleep durations for efficient SVM power management.*

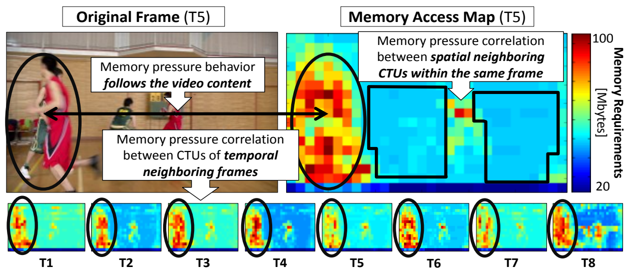


Figure 4. Video content and neighborhood correlation analysis for *BasketballDrive* test sequence.

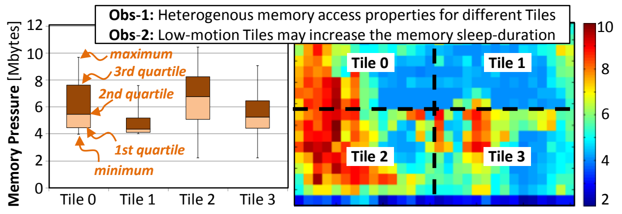


Figure 5. Intra-Tile memory pressure analysis for *BasketballDrive.*

The goal of our work is *to leverage application-specific properties for memory pressure balancing and SVM's leakage energy reduction targeting parallelized HEVC encoding*.

1.3 Our Novel Contributions

We propose content-driven memory pressure balancing along with SVM power management for HEVC parallelized on manycore processors. The key is to leverage the memory access correlation *within* and *across* different Video Tiles (i.e. Intra- and Inter- Video Tile correlation). Our scheme employs:

- **A Memory Pressure Prediction Algorithm (Section 2.1)** that leverages the monitored memory pressure of Video Tiles in the previously encoded CTUs in order to accurately predict the memory requirements for Video Tiles in the current frame.
- **Run-Time Statistics-Based CTU Memory Classification (Section 2.2)** that dynamically adapts the parameters involved in our memory power management scheme according to the predicted memory pressure statistics.
- **CTU Re-Scheduling for Memory Pressure Balancing (Section 2.3)** our scheme groups the CTUs of a *Video Tile* into

variable-size groups (called CTU-groups). The size of the CTU-groups depends on the Video Tile-specific motion activity properties. Depending upon the predicted memory pressure, we schedule the CTU-groups to closely meet the target pressure.

- **Content-Driven Power Management of SVMs (Section 3):** since the CTU-groups may also exhibit similar properties blocks, our scheme analyzes the predicted memory usage of different CTU to increase the potential of the sleep-duration of different SVM regions and thereby increasing the leakage energy savings.

To the best of authors' knowledge, this is the first work towards *managing the memory pressure in parallel video processing* that exploits the video content properties and memory access correlation.

1.4 Overview of Our Memory System

Figure 6 depicts the overall system with our content-driven memory power management. To support HEVC encoding parallelized using *n Video Tiles*, our system has (1) a multicore processor with *n* cores and (2) a memory infrastructure containing *n* SVMs, such that every core has its private on-chip SVM for search window storage used during the motion estimation process. The SVMs are connected to the external memory by data/address bus interfaces. Our content-driven memory pressure balancing scheme is composed of the following three modules: (a) memory pressure prediction, (b) run-time statistics-based CTU memory classification, and (c) CTU re-scheduling for memory pressure balancing. Furthermore, our memory management system also employs a content-driven power management of SVMs. It leverages the run-time statistical analysis performed by (a) and (b). A memory monitoring unit feeds the statistics about the current memory requirements to our system.

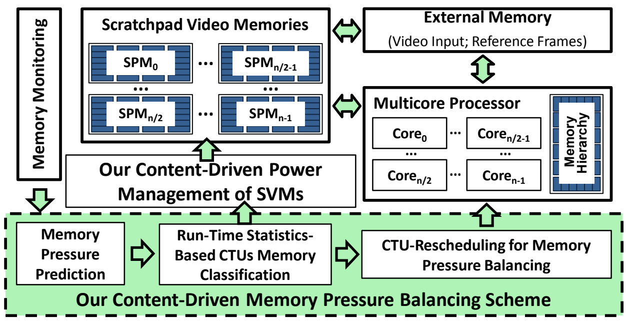

Memory Monitoring
Scratchpad Video Memories
SPM0
SPMn/2-1
SPMn/2
SPMn-1
External Memory
(Video Input; Reference Frames)
Multicore Processor
Core0
Coren/2-1
Coren/2
Coren-1
Memory Hierarchy
Our Content-Driven Power Management of SVMs
Memory Pressure Prediction
Run-Time Statistics-Based CTUs Memory Classification
CTU-Rescheduling for Memory Pressure Balancing
Our Content-Driven Memory Pressure Balancing Scheme

Figure 6. n-Tile HEVC encoding system with our application-specific memory power management scheme.

Although traditional scratchpad memories require programmer driven control, recent works have demonstrated run-time management of these memories where data allocation is managed by a virtual manager, like [8][9]. In our case, instead of explicitly passing the control to the programmer, we have an application-specific hardware management of these SVMs (which is much simpler compared to the management circuitry of cache memories).

2. CONTENT-DRIVEN MEMORY PRESSURE BALANCING SCHEME

2.1 Memory Pressure Prediction

As demonstrated in Section 1.2, highly correlated memory pressure may exist (1) among spatial neighboring CTUs (within the same frame); and (2) among CTUs of temporal neighboring frames. Therefore, based on the actual memory usage of previously processed CTUs (*ActualMem*), our prediction algorithm estimates the memory requirements of the CTUs in the current frame[1]. Figure 7 depicts an example of used CTU predictors in the current and reference frames. Four spatial predictors from the current frame and nine temporal predictors from each reference frame are selected as input to a weighted prediction. Eq. (1)-(2) presents the spatial and temporal predictors selected for a given CTU: *Pred*$_{Temp}$ and *Pred*$_{Spatial}$, respectively. The letters *A-M* correspond to the spatial and temporal predictors depicted in Figure 7. As statistical parameters for the prediction, we apply different weighting factors[2] according to the spatial location of the predictor related to the current CTU position. Possible cases of CTU position are: center (α_C), horizontal/vertical (α_A), and diagonal (α_D). Eq. (3)-(5) present the weighted prediction formula for predicting the memory pressure considering a given CTU. The weighting factors were statistically generated based on the memory access correlations of real video test sequences. First, the predicted memory pressure considering only the temporal references is estimated: *PredMem*$_{Temp}$ in Eq. (3). Then, the spatial predictors are used to calculate the *PredMem*$_{Spatial}$, as in Eq. (4). Finally, both spatial and temporal predictions are used to derive the predicted memory pressure for the given CTU: *PredMem* in Eq. (5).

$$\mathrm{Pred}_{\mathrm{Temp}}(\mathrm{F}_{\mathrm{Ref}}) := \mathrm{WP}(\mathrm{ActualMem}(\mathrm{F}_{\mathrm{Ref}}[\mathrm{A}\ldots\mathrm{I}]), [\alpha_{C,}\alpha_{A,}\alpha_D]) \quad (1)$$

$$\mathrm{Pred}_{\mathrm{Spatial}} := \mathrm{WP}(\mathrm{ActualMem}(\mathrm{F}_{\mathrm{Curr}}[\mathrm{J}\ldots\mathrm{M}]), [\alpha_{A,}\alpha_D]) \quad (2)$$

$$\mathrm{PredMem}_{\mathrm{Temp}} = \sum_{\forall\, \mathrm{F}_{\mathrm{Ref}}} \left\{ \left[\sum\nolimits_{\mathrm{P_T} \in \mathrm{Pred}_{\mathrm{Temp}}(\mathrm{F}_{\mathrm{Ref}})} (\mathrm{P_T}) \right] * \frac{1}{\mathrm{D}[\mathrm{F}_{\mathrm{Ref}}]} \right\} \quad (3)$$

$$\mathrm{PredMem}_{\mathrm{Spatial}} = \sum\nolimits_{\mathrm{P_S} \in \mathrm{Pred}_{\mathrm{Spatial}}} (\mathrm{P_S}) \quad (4)$$

$$\mathrm{PredMem(CTU)} = \mathrm{WP}(\mathrm{PredMem}_{\mathrm{Temp}}, \mathrm{PredMem}_{\mathrm{Spatial}}, [\alpha_{S,}\alpha_T]) \quad (5)$$

When some predictors are unavailable (e.g., in case of CTUs at the frame boundaries) the weighted prediction is performed only with the available predictors.

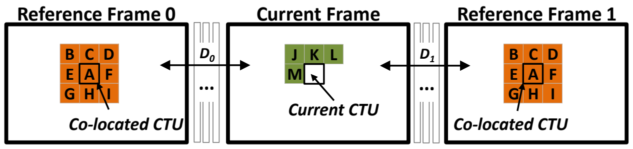


Figure 7. Example: spatial and temporal predictors selecting.

The predicted memory requirements of the CTUs need to be analyzed to classify each video frame, Video Tile and CTU-groups to characterize their memory access behavior.

2.2 Run-Time Statistics-Based CTU Memory Classification

As motivated in Section 1.2, in order to avoid the memory pressure imbalance problem of traditional raster scan order processing, our scheme re-schedules the order of CTU evaluations for motion estimation. To achieve this, our scheme partitions the CTUs of a Video Tile into so-called *CTU-groups*, which are rectangular regions of CTUs such that, all CTUs of a given CTU-group are processed consequently; see an example in Figure 8. The goal is to assign CTUs with similar memory requirements/pressure into one group while balancing the overall memory pressure of Video Tiles.

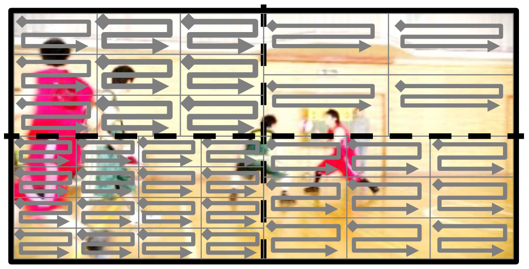


Figure 8. Example: CTU-groups division for re-scheduling.

The memory access distribution follows specific properties (i.e., motion and texture) of each video sequence. Hence, we use the video

[1] A current frame refers to the frame being encoded at that moment.

[2] Statistically defined parameters using the experimental methodology described in Section 4: α_C=0.5, α_A=0.3, α_D=0.2, α_S=0.5, and α_T=0.5.

properties to decide the number of CTU-groups. Our scheme adapts the number of CTU-groups at frame level according to the predicted memory access distribution of Video Tiles. At first, a base number of groups is defined, N_B in Eq. (6). It is based on the Probability Density Function (PDF) of the predicted memory pressures at frame level (μ_F is the average, σ_F is the standard deviation) and the average number of CTUs per Video Tile ($\mathrm{N_{CTUPerTile}}$). Later on, we define the actual number of groups for each Video Tile (N_G in Eq. (7)) by comparing the predicted memory access distribution of a given Video Tile (μ_T, σ_T) with that of the with the overall frame. Video Tiles with spread memory pressure distributions are divided into more CTU-groups to enable fine-grained management (first clause of Eq. (7)). The goal is to have a fine-grain management because we may have very diverse memory behaviors within a Video Tile. In contrast, Video Tiles with concentrated memory pressure distribution (second clause of Eq. (7)) lead to few (but large-sized) CTU-groups as their texture and motion properties tend to be correlated inside such a Video Tile. The decision of having smaller CTU-groups must be carefully taken because the SVM data reuse among adjacent CTUs is not available between each CTU-group processing, causing efficiency loss in the SVM data management. Due to the CTU order inside one CTU-group (see Figure 8), the SVMs are more efficient for large-groups.

$$N_B = \lceil(\sigma_F/\mu_F)\ \mathrm{N_{CTUPerTile}}\rceil \\ \text{where: } \{\mu_F, \sigma_F\} = \mathrm{PDF}(\mathrm{PredMem}(\mathrm{CTU}) | \forall\ \mathrm{CTU} \in \mathrm{Frame}\ F) \tag{6}$$

$$\mathrm{N_G(T)} = \begin{cases} (\sigma_T > \sigma_F), \lceil \mathrm{N_B} + [(\sigma_T/\mu_T) - (\sigma_F/\mu_F)]\mathrm{N_{CTUPerTile}} \rceil \\ (\sigma_T \le \sigma_F), \lceil \mathrm{N_B} - [(\sigma_F/\mu_F) - (\sigma_T/\mu_T)]\mathrm{N_{CTUPerTile}} \rceil \end{cases} \tag{7}$$
$$\text{Where: } \{\mu_T, \sigma_T\} = \mathrm{PDF}(\mathrm{PredMem}(\mathrm{CTU}) | \forall\ \mathrm{CTU} \in \text{Video Tile } T)$$

The predicted memory pressure distribution is used to classify the Video Tile in terms of motion property. By comparing the average behavior of each Video Tile-specific distribution to the overall frame distribution, Eq. (8) defines three categories: *H-type* (high motion), *M-type* (medium motion), and *L-type* (low motion). Moreover, each CTU-group also has its own PDF (given in Eq. (9)) that will be used for the re-scheduling decision during the memory pressure balancing.

$$\mathrm{C_{Tile}(T)} = \begin{cases} (\mu_T \ge \mu_F + 0.5\sigma_F), \text{(H) High} \\ (\mu_F + 0.5\sigma_F > \mu_T > \mu_F - 0.5\sigma_F), \text{(M) Medium} \\ (\mu_F + 0.5\sigma_F > \mu_T), \text{(L) Low} \end{cases} \tag{8}$$

$$\{\mu_G, \sigma_G\} = \mathrm{PDF}(\mathrm{PredMem}(\mathrm{CTU}) | \forall\ \mathrm{CTU} \in \mathrm{CTUgroup}\ G) \tag{9}$$

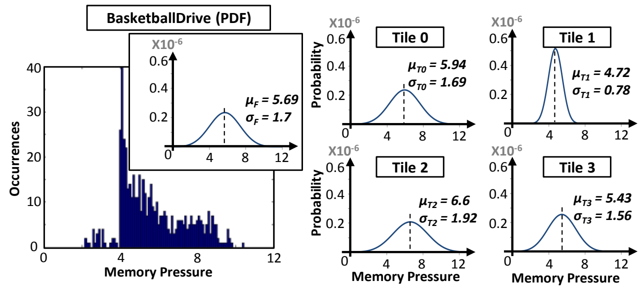


Figure 9. Memory pressure statistics for each Video Tile of the *BasketballDrive* test sequence (PDFs and histogram).

An Example: Figure 9 presents the run-time statistics of the predicted memory pressure of a frame in the HD1080 *BasketballDrive* video encoded with 4-Tiles. The N_{Base} value, which is only dependent on the overall frame statistics, is calculated using Eq. (6), i.e. $N_{Base}=6$. Using Eq. (7), the number of CTU-groups at is calculated: $N_G(0)=6$, $N_G(1)=2$, $N_G(2)=8$, $N_G(3)=4$. Using Eq. (8), the motion classification of Video Tiles are: $C_{Tile}(0)$=*M-type*, $C_{Tile}(1)$=*L-type*, $C_{Tile}(2)$=*H-type*, and $C_{Tile}(3)$=*M-type*.

The above analysis and predicted memory pressure statistics are used by our CTU re-scheduling algorithm for memory pressure balancing and by the corresponding power management policy.

2.3 CTU Re-Scheduling for Memory Pressure Balancing

The goal of our CTU re-scheduling is to balance the accumulated memory pressure at the Video Tiles level, reducing the mean squared deviance (MSD) related to the average memory pressure (ideal case). Different number of CTU-groups leads to variable-sized groups, containing more or less CTUs within each Video Tile. Our scheme also classifies the Video Tiles according to the motion properties in three classifications C_{Tile}=*{H-type, M-type, L-type}* using the Eq. (8). Different Video Tile types will contributes in different ways for the accumulated balancing: *H-type* Video Tiles start by occupying the most part of the memory bandwidth, *M-type* Video Tiles contribute by median memory occupation, and the *L-type* Video Tiles aim to alleviate the memory pressure. The main task of our scheme is to schedule the CTU-groups processing.

Figure 10 depicts our CTU-groups scheduling functionality that is called at two points: (1) at the initial frame processing, when the decision about CTU-groups scheduling has not already taken, and (2) at the end of one CTU-group processing, when a new group must be scheduled. The call for this routine is performed at Video Tile-level, when the algorithm analyzes the current scenario to take the best decision. So, as input parameters we have the *ID* of the Video Tile $Tile_{ID}$ and the list of CTU-groups ($L_{CTU\text{-}Groups}$) that are inside the target Video Tile (*line* 1). For the first frame of the video, there are no temporal references for memory predictors, so the traditional raster scan order is performed (*lines* 2-3). If it is not the first frame, all memory predictions and run-time memory-related classifications are performed at the beginning of the frame processing. In case of the first CTU-group scheduling, the algorithm takes the motion Video Tile classification C_{Tile} into account to decide the CTU-group that will be next coded ($G_{ToBeCoded}$) (lines 6-9). Otherwise, our adaptive scheme analyzes the gap (*gapAccumPress*) between the current memory pressure (*currMemPress*) and an approximate average case prediction (*averageAccumPress* in *line 11*). So, the algorithm selects the CTU-group which has the predicted memory pressure and that has the best fit to the predicted gap (*lines 11-14*). After this decision, the CTU-group is removed from the non-coded groups list and the CTUs according are encoded according to the CTU-groups internal processing order depicted in Figure 8 (*line 17*).

```
1.  scheduleCTUGroup(Video Tile: Tile_ID, List of CTU-groups: L_CTU-Groups)
2.  If first frame Then
3.      G_ToBeCoded := L_Groups.first();   //CTU-group equals to Video Tile
4.  Else   //not the first frame
5.      If frame start Then   //run-time statistical knowledge of Video Tiles
6.          Tile_class := C_Tile(Tile_ID);   //Eq. (8) – statistical classification
7.          Case(Tile_Class = L-type): G_ToBeCoded := L_CTU-Groups.min();
8.          Case(Tile_Class = M-type): G_ToBeCoded := L_CTU-Groups.median();
9.          Case(Tile_Class = H-type): G_ToBeCoded := L_CTU-Groups.max();
10.     Else
11.         averageAccumPress:= $\sum_{T=0}^{N_{Tiles}}(\mu_T)$ ; //sum of av. pressures
12.         currAccumPress := getCurrentMemoryPressure();   //monitoring
13.         gapAccumPress := averageAccumPress – currAccumPress;
14.         G_ToBeCoded := (G | $\mu_G$ has the best fit to gapAccPress);
15.     End If;
16. End If;
17. L_CTU-Groups .remove(G_ToBeCoded); encode(G_ToBeCoded);
```

Figure 10. CTUs re-scheduling algorithm.

Besides memory pressure balancing, we also develop an on-chip power management that controls the low-power states of different blocks of the SVMs while increasing their sleep durations.

3. CONTENT-DRIVEN POWER MANAGEMENT

Our power management policy monitors each core's private SVM usage to capture the current video motion property and power-gate less-likely used sectors to save on-chip leakage energy.

Memory Power Model: We consider a memory technology with three power states: P_{ON}, P_{DR} *(Data Retentive)* and P_{OFF}, where:

$V_{ON}=V_{dd}$, $V_{DR}=0.3*V_{dd}$ and $V_{OFF}=0$, and the wake-up energies (WE) for power states transitions[3] are $WE_{T0}=1/2*C_{Circuit}*V_{dd}^2$, and $WE_{T1}=0.65*WE_{T0}$. V_{dd} is the memory supply voltage, and $C_{Circuit}$ is the total capacitance of the memory [5]. Our SVMs are divided into N_{Secs} memory sectors that are power gated by the same sleep transistor. One memory sector supports a 16x16 search window block ($S_{Sector}=16*16*8bits=2048bits$). This sectors organization *allows fine-grain memory management* during the encoding, since variable blocks sizes are processed and very accurate memory power states assignment is required. The SVMs are sized to *store one complete search window in a private way for each core*. Hence, we have $N_{SVM}=N_{Tiles}=N_{Cores}$ number of SVMs and each SVM has $S_{SVM}=(SW_H+64)*(SW_V+64)*8$ Kbits, where SW_V and SW_H are the search window vertical and horizontal dimensions.

Run-Time SVM Usage Analysis: Our evaluations in Figure 11 illustrate that we can *increase the potential of long sleep durations once the memory pressure is balanced*. For example, Figure 11a presents the SVM usage for the core 1 when encoding the *BasketballDrive* sequence. The SVM usage (SVM_{Usage}) calculated as the percentage of accessed SVM memory positions (measured by our memory monitoring unit) during one ME operation (*AccSVM*), see Eq. (10). As shown in Figure 11(b), the SVM usage for the entire CTU can be determined as the Probability Density Function of the SW_{Usage} values of all blocks within the CTU, see Eq. (11).

$$\mathrm{SVM_{Usage}(CU_{ID}) = AccSVM(CU_{ID})/S_{SVM}} \tag{10}$$

$$\mathrm{SVM_{UsagePDF}(CTU) = \{\sigma_{SVM}, \mu_{SVM}\} = PDF(SPM_{Usage}(Block_{Node}) | \forall\ Block_{Node} \in CTU)} \tag{11}$$

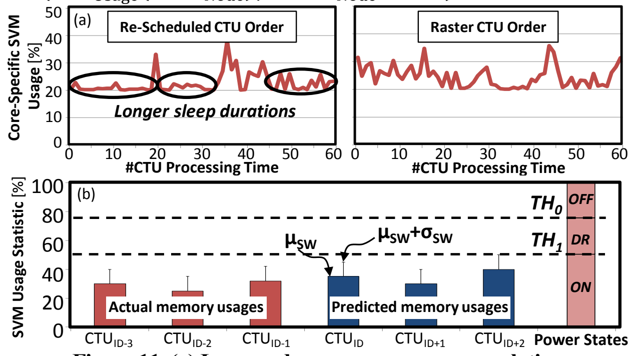


Figure 11. (a) Increased memory pressure correlation; (b) power states determination based on the SVM usage PDFs.

Our Power Management Scheme: At the beginning of a CTU encoding, the algorithm predicts the number of the memory sectors that can be put into different power state (i.e., N_{ON}, N_{DR} and N_{OFF}). As basis for this prediction, we analyze (1) the actual search window usage for previously processed CTUs (e.g., CTU_{ID-3}, CTU_{ID-2}, and CTU_{ID-1}); (2) the predicted usage for the current CTU_{ID} and the next CTU_{ID+1} and CTU_{ID+2}. The goal is to have the knowledge of the past, present and predicted future memory requirements to increase the on-chip leakage energy savings while minimizing the overhead for memory sectors waking-up. Figure 11(b) presents an example of SVM usage PDFs and the corresponding power states assignment.

Figure 12 presents our power management policy. The actual SVM usage PDFs of the past CTUs ($List_{ActualSVMUsagePDF}$) and the next predicted SVM usage PDFs ($List_{PredSVMUsagePDF}$) are used to determine the power states of the SVM sectors (*lines* 3-5). As in Figure 11, we define two thresholds (TH_0 and TH_1) based on the average and standard deviation of all cited PDFs (*lines 6-7*). Afterwards, the SVM sectors corresponding to each power states are derived (*lines 8-9*). The physical assignment of the power states to the SVM cells is performed at the beginning of every block processing within a CTU (*lines 10-13*). In the case data retransmission is required (SVM cells wake-up from the P_{OFF} state), the control unit inserts stalls in the execution pipeline. Still, this penalty implies a negligible energy/performance overhead since in our experiments the worst-case scenario is observed <0.2% times.

```
1.  managePowerSVM (VideoTile: $Tile_{ID}$, CTU: $CTU_{ID}$)
2.  $PowerMap_{SVM} := \Phi$; $N_{ON} := 0$; $N_{DR} := 0$; $N_{OFF} := 0$;
3.  $List_{ActualSVMUsagePDF} := (SVM_{UsagePDF}(ActualMem(CTU_{ID}) | ID \in \{-3..-1\}))$;
4.  $List_{PredSVMUsagePDF} := (SVM_{UsagePDF}(PredMem(CTU_{ID}) | ID \in \{0..2\}))$;
5.  $List_{PDF}$.append($List_{ActualSVMUsagePDF}$, $List_{PredSVMUsagePDF}$);
6.  $TH0 := max(\mu_{SVM}+3.\sigma_{SVM} | (\mu_{SVM},\sigma_{SVM}) \in \{List_{PDF}\})$;   //TH's definition
7.  $TH1 := max(\mu_{SVM}+1.\ \sigma_{SVM} | (\mu_{SVM},\sigma_{SVM}) \in \{List_{PDF}\})$;
8.  $N_{OFF} := (1-TH0)*N_{Sec}$; $N_{DR} := (TH0-TH1)*N_{Sec}$; $N_{ON} := TH1*N_{Sec}$;
9.  $PowerMap_{SVM}$.assignPowerStates($N_{ON}$, $N_{DR}$, $N_{OFF}$);
10. For all Block $\in CTU_{ID}$
11.     SVM[$Tile_{ID}$].powerGate($PowerMap_{SVM}$);   //apply power gating
12.     encode(Block);
13. End For;
```

Figure 12. On-chip power management of SVMs.

4. RESULTS AND DISCUSSIONS

4.1 Experimental Methodology

The experiments are performed using the HEVC software (HM 11.0) using the common test conditions adopted by the video coding community [6]. Four HD1080p (1920x1080) test video sequences with different properties were adopted: *BasketballDrive (BDrive)*, *BQTerrace (BQTerr)*, *Cactus* and *Kimono*. We consider 4-/16-Tile scenarios, 128x128 search window size (typical dimension for HD1080p [7]), GOP=8, FRExt, CABAC, and TZ Search algorithm for motion estimation. We use 4 and 16 threads (i. e., Video Tiles), each executing on a dedicated/specific processing core. Therefore, we use 4-, and 16-core x86 processor in our setup. Table 1 presents the on-chip SVM parameters as per the model defined in Section 3.

Table 1. On-Chip SVMs Sizing Parameters

SVM Sizing Parameter	Value
Number of SVMs	4, 8, 16 (one SVM per core)
SVM Sector Size	2048 bits = 256 B
SVM Size	(128+64) * (128+64) * 1B = 36 KB
Memory Size (4-Core)	114 KB
Memory Size (16-Core)	576 KB

We developed a custom simulator that takes the HM 11.0 memory traces for each thread (independent Video Tile) as input and estimates the accumulated memory pressure and the on-chip leakage energy. Our simulator contains memory models for the external memory and for the on-chip SVMs. For the external memory, we used a Low-Power DDR2 DRAM (LPDDR2) memory model (from Micron technical specification [14][15]) to derive the memory pressure. For the on-chip memory leakage energy estimation, we extracted the electrical parameters (for the 65nm SRAM technology node) using the CACTI 6.5 tool [11], as well as the multiple power states model described in Section 3.

MSD metric: Let the $Mem_{[0\ldots m]}$ be the discretized memory pressure measurements along the time. The mean squared deviance (MSD) calculates the squared different between each memory pressure measured point and the *Mem* average value (μ_{Mem}), as in Eq. (12).

$$\mathrm{MSD}(\mathrm{Mem}_{[0\ldots m]}) = \frac{1}{|\mathrm{Mem}|}\sum_{i=0}^{m}(\mu_{Mem} - Mem_i)^2 \tag{12}$$

4.2 Memory Pressure Balancing Results

Figure 13 presents a temporal evaluation of the memory pressure comparing (1) the traditional CTU raster processing order; (2) our application-specific memory pressure balancing scheme using CTU-rescheduling; and (3) the optimal corner case where the memory pressure is continuously equals to the average pressure. The case (3) is a theoretical approximation used to evaluate the gaps of our and the traditional schemes related to the best possible balancing case. Figure 13 shows that our scheme balances the pressure for each processing core. Compared to the traditional raster order, the

[3] Power states transitions: *T0* (P_{OFF}→P_{ON}), and *T1* (P_{DR}→P_{ON}).

maximum-minimum peak variations are reduced from 27%-32% to 9%-13%, respectively. Our scheme achieves this balancing by effectively predicting the memory requirements, capturing the Video Tile-specific properties, and managing the processing order.

Figure 14 presents our results regarding the memory pressure balance. As already discussed, more Video Tiles potentially leads to more unbalanced accumulated memory pressure, since more concurrent memory accesses are performed during each time slot. In this scenario, there is a high probability of having very different motion properties being processed by different cores at the same time. So, the balancing gap when more Video Tiles are used is higher. Our scheme successfully exploits this potential, as shown in Figure 14. The MSD efficiency reduction ranges from, on average, 37% to 83%, for 4 to 16 Video Tiles. Therefore, our application-specific memory power management is efficiently scalable when working with an increased number of Video Tiles.

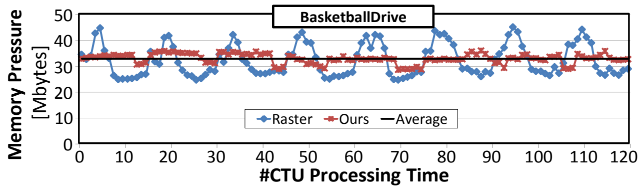


Figure 13. Accumulated memory pressure results of our scheme.

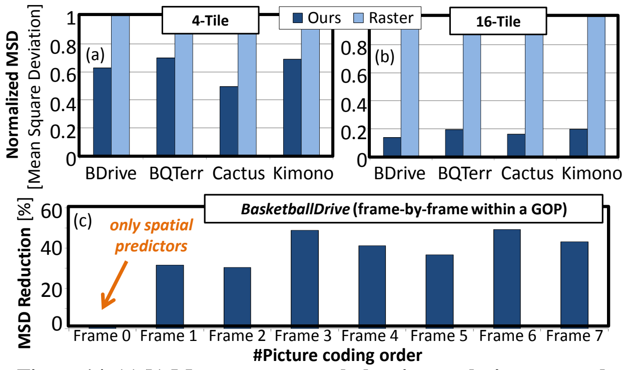


Figure 14. (a)(b) Memory pressure balancing analysis compared to the original raster scan order and (c) frame-by-frame analysis.

Figure 14c depicts a frame-by-frame MSD reduction analysis. During the first frame processing, as only spatial references can be used as input for our memory pressure predictor, our scheme achieves results close to the original raster order. However, by acquiring the temporal knowledge, our scheme fits the CTU-Groups accordingly to capture the motion properties and achieves increased memory pressure balancing for the other remaining frames. Thus, we can increase the accumulated memory pressure balancing by up to 49% in the case of 4-Tile *BasketballDrive* scenario (Figure 14c).

4.3 On-Chip Leakage Energy Savings

Figure 15a depicts the on-chip leakage energy savings of our content-driven SVM power management policy for different video sequences. On average, our scheme saves 56% of on-chip energy by power gating the unused and less-likely used memory sectors. The wake-up energies overhead is already included into the results of Figure 15. Our energy reductions are high in case of the low-motion Video Tiles by achieving longer sleep durations due to consecutive processing of CTU with similar texture and motion. This behavior is demonstrated in Figure 15b where the total energy savings are decomposed for each core-private SVM. The low-motion Video Tiles provide the highest savings while the medium- and high-motion Video Tiles required more energy due to higher memory usage as a result of an extensive search. When considering SVMs, the energy/performance overhead of waking up the memory cells are negligible, since one block of the search window is continuously accessed during one ME operation over a given block of the CTU. Thus, the energy/performance penalty is completely amortized, not leading to significant overhead for the overall memory system.

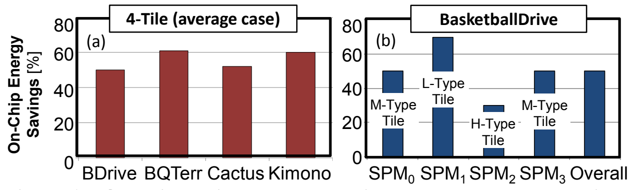


Figure 15. On-chip static energy reduction due to our content-driven power management of SVMs.

5. CONCLUSIONS

This work presented a content-driven memory pressure balancing scheme with an integrated power management policy. Our scheme is composed of: (1) a prediction unit that estimates the memory pressure due to the monitoring of past CTUs encoding; (2) a run-time statistics-based CTU memory classification that adapts the involved parameters of our schemes to the current video content; (3) a memory pressure balancing strategy that adaptively changes the CTU processing order to reduce the accumulated memory pressure variations; and (4) a power management policy that analyzes the actual and predicted memory usage for the CTUs to accordingly power-gate unused (or less-likely used) video memory sectors. Our experimental results demonstrated that our scheme can reduce the memory pressure peak variation by 37%-83% compared to the state-of-the-art raster processing order, for 4-/16-core processors. The SVM leakage energy is reduced by 56%. This work illustrates that the reducing memory pressure and on-chip SVM leakage energy are crucial for parallel HEVC on real-world embedded systems.

6. AKNOLEDGEMENTS

This work was partly supported by DFG as part of Transregional. Collaborative Research Centre "Invasive Computing" (SFB/TR 89); and partly by DAAD/CAPES as part of PROBRAL project VideoArch^{3D}.

7. REFERENCES

[1] JCT-VC, "High Efficiency Video Coding (HEVC) text specification draft 10 (for FDIS & Consent)", Doc.: JCTVC-L1003_v9, 2013.
[2] B. Zatt, M. Shafique, F. Sampaio, L. Agostini, S. Bampi, J. Henkel, "Run-time adaptive energy-aware motion and disparity estimation in multiview video coding", IEEE DAC, pp. 1026-1031, 2011.
[3] M. Shafique, B. Zatt, S. Bampi, J. Henkel, "Adaptive Power Management of On-Chip Video Mem. for Multiview Video Coding", DAC, pp. 866-875, 2012.
[4] F. Sampaio, B. Zatt, M. Shafique, J. Henkel, S. Bampi, "Energy-Efficient Memory Hierarchy for Motion and Disparity Estimation in Multiview Video Coding", IEEE DATE, pp. 665-670, 2013.
[5] H. Singh et al., "Enhanced leakage reduction techniques using intermediate strength power gating", IEEE Transactions on Very Large Scale Integration, vol. 15, no. 11, pp. 1215-1224, 2007.
[6] F. Bossen, "Common test conditions and software reference configurations", ITU-T/ISO/IEC JCTVC-K1100, 2012.
[7] K. Misra et al., "An overview of tiles in HEVC," JSTSP, no.99, 2013.
[8] D. Cho et al., "Adaptive Scratch Pad Memory Management for Dynamic Behavior of Multimedia Applications," TCAD, v.28, n.4, pp.554-567, 2009.
[9] I. Issenin, et al. "Data-Reuse-Driven Energy-Aware Cosynthesis of Scratch Pad Memory and Hierarchical Bus-Based Communication Architecture for Multiprocessor Streaming Applications". TCAD, v. , n. , pp. 1439-1452, 2008.
[10] IBM, "The Cell Project", Last Accessed: Sep. 2013, <researcher.watson.ibm.com/researcher/view_project.php?id=2649>.
[11] S. Thoziyoor, N. Muralimanohar, J.-H. Ahn, and N. P. Jouppi, "CACTI 5.1 technical report," HP Labs, Tech. Rep. HPL-2008-20, 2008.
[12] M. Jeong, et al. "A QoS-aware memory controller for dynamically balancing GPU and CPU bandwidth use in an MPSoC," DAC, pp. 850–855., 2012.
[13] L. L. Pilla, et al. "A hierarchical approach for load balancing on parallel multi-core systems," In: ICPP'12, pp. 118–127, 2012.
[14] Micron. "4Gb: x16, x32 Mob. LPDDR2 SDRAM S4", 168p, 2013.
[15] Micron. "TN-46-03 – Calculating DDR Mem. System Power"., 26p, 2005.

Energy-Efficient Architecture for Advanced Video Memory

Felipe Sampaio[1], Muhammad Shafique[2], Bruno Zatt[3], Sergio Bampi[1], Jörg Henkel[2]
[1]Informatics Institute, PPGC, Federal University of Rio Grande do Sul (UFRGS), Brazil
[2]Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
[3]GACI, PPGC, CDTec, Federal University of Pelotas (UFPel), Brazil
{felipe.sampaio, bampi}@inf.ufrgs.br, bzatt@inf.ufpel.edu.br, {muhammad.shafique, henkel}@kit.edu

***Abstract*— An energy-efficient hybrid on-chip video memory architecture (enHyV) is presented that combines private and shared memories using a hybrid design (i.e., SRAM and emerging STT-RAM). The key is to leverage the application-specific properties to efficiently design and manage the enHyV. To increase STT-RAM lifetime, we propose a data management technique that alleviates the bit-toggling write occurrences. An adaptive power management is also proposed for static-energy savings. Experimental results illustrate that enHyV reduces on-chip static memory energy compared to SRAM-only version of enHyV and to state-of-art AMBER hybrid video memory [9] by 66%-75% and 55%-76%, respectively. Furthermore, negligible external memory energy consumption is required for reference frames communication (98% lower than state-of-the-art Level C+ technique [18]). Our data management significantly improves the enHyV STT-RAM lifetime, achieving 0.83 of normalized lifetime (near to the optimal case). Our hybrid memory design and management incur low overhead in terms of latency and dynamic energy.**

I. Introduction

Advanced video processing algorithms introduce very high pressure on the memory hierarchy, leading to undesirable energy and performance overheads [2][3]. Therefore, battery-powered applications incorporate dedicated video memories to provide enough data bandwidth while reducing energy consumption. Video codecs are among the most complex and widely deployed video processing applications. Recently, the next-generation High-Efficiency Video Coding (HEVC) [1] has been released that provides double coding efficiency compared to the H.264/AVC. However, this comes at the cost of increased computation time by more than 40% [8]. Besides employing novel complex coding tools, an HEVC encoder requires a significant amount of data from the off-/on-chip memories due to more intensive reference frames transmission for the prediction. On average, the memory demand is 2x-3x higher compared to that of the H.264/AVC [8]. High off-/on-chip memory bandwidth along with larger on-chip video memories (to support bigger resolutions) leads to increased energy consumption in HEVC encoders. Additionally, the HEVC has data parallelism support to provide high processing rates. However, parallel processing *tightens the memory energy restrictions due to multiple cores accessing the same memory infrastructure simultaneously*, which aggravates the memory pressure.

Recently, the *hybrid memory architectures* for general purpose manycore processors have evolved that utilize emerging memory technologies (e.g., MRAM, STT-RAM [12]; see Section I.B) in combination with traditional SRAM cells [9][11]. Their goal is to reduce the impact of SRAM shortcomings like low density and high static energy consumption. Typically, for general purpose applications, the emerging technologies are desired for the last-level cache due to the low-static-energy and high-density features [10]. However, due to lack of application-specific knowledge, these schemes are not efficient enough to support the high memory requirements of HEVC.

Therefore, *there is a need of application-driven design for energy-efficient and performance-aware hybrid memories tailored towards HEVC executing on manycore processors*.

Before we introduce our novel contributions, we will present our motivational memory analysis of HEVC encoders followed by a brief overview of the emerging memory technologies in comparison to the traditional SRAM.

A. *Memory Energy Bottleneck in HEVC*

The new coding structure of HEVC divides the video frame into flexible block sizes following a quad-tree structure called *coding-tree unit* (CTU) [7]. Typically the CTU partitioning starts from the maximum allowable block size of 64x64 pixels into several *coding units* (CU) of sizes 32x32, 16x16 and 8x8 pixels. Fig. 1(a) depicts an example of CTU partitioning. For each CU, the *motion estimation* searches the most similar block within a delimited squared portion of reference frames, called search window [7]. The motion estimation processes each possible CU inside a CTU, thus resulting in the most time and energy consuming module of an HEVC encoder. It requires 70%-80% of the encoding time and consumes 80%-90% of the total energy [2][3]. Furthermore, from the memory perspective, the reference frames fetching from the external memory and its on-chip storage lead to significant energy consumption (>92% of the motion estimation energy) [2][3].

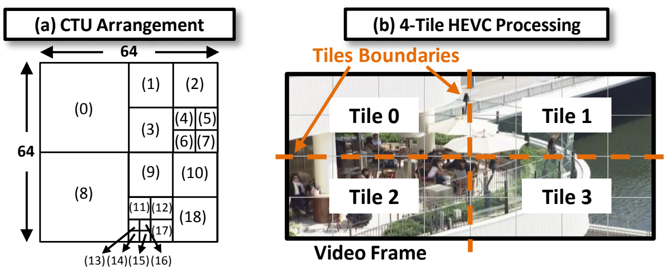


Fig. 1 (a) Example of CTU partitioning into variable-sized CUs; (b) 4-Tile partitioning of a video frame.

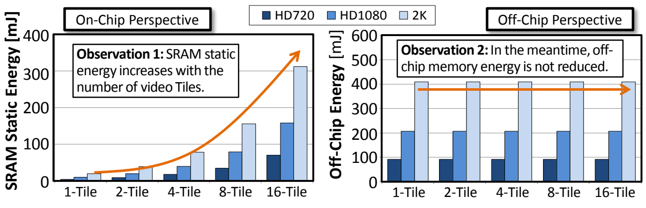


Fig. 2 SRAM on-chip static energy vs. DRAM off-chip energy analyses using reference-frame-on-chip strategy.

To exploit multiple cores in a manycore system, the HEVC provides a light-weight data parallelism support that divides the video frame into rectangular regions called *video Tiles* [7]; see Fig. 1(b). These video Tiles can be encoded independent to each other without any spatial dependencies, thus can be parallelized on multiple cores. The video Tile-parallelized HEVC aggravates the video memory bottleneck (as shown in Fig. 2 for a larger number of video Tiles), leading to large on-chip video memories and high static energy consumption. Typically, the entire reference frame for each video Tiles processing is fetched to the on-chip SRAM-based video memories (i.e., reference-frame-on-chip scenario in Fig. 2). Fig. 2 illustrates a significant increase in the SRAM static energy when increasing the number of used video Tiles whereas the external memory bandwidth is not even reduced. An important observation can be made: *the on-chip video memories based on SRAM have low energy efficiency when larger on-chip video memories are required for parallel video Tiles processing support*. In general, this would also be valid for other multi-threaded video processing workloads. Hybrid memory design has emerged as a promising solution to address the SRAM limitations, i.e., low density and high static energy.

B. *SRAM and Emerging STT-RAM Memory Technologies*

As an emerging memory technology, the Spin-Transfer Torque SRAM (STT-RAM) [12] provides higher density, better scalability and low static power features compared to the SRAM. In other aspects SRAM is still much more efficient, like in terms of write power and overall performance. Table I presents a subjective comparison between SRAM and STT-RAM technologies[1], where the dark-gray cells represent the best scenario of each parameter.

TABLE I SRAM VS. STT-RAM TECHNOLOGIES [12]

	Energy			**Latency**		**Volatility**
	Static	**Read**	**Write**	**Read**	**Write**	
SRAM	HH	**L**	**L**	**L**	**L**	Volatile
STT-RAM	**L**	**L**	HH	**L**	H	**Non-Volatile**

The on-chip video memories have a particular property that facilitates the STT-RAM usage: *they have a relatively low write intensity compared to a very high read intensity* [8]. As the on-chip video memories implement data-reuse schemes for the search window samples, only a few data of the reference frame would be written to start the next CTU prediction. Once the needed data is stored on chip, the motion estimation massively accesses the on-chip video memory until the best match is found. As can be noticed in Table I, the STT-RAM energy and performance are poor for write operations compared to that of the SRAM. Thus, video coding is a promising application for STT-RAM based hybrid memories.

STT-RAM is also known to be a non-volatile memory (NVM). This characteristic is very important for on-chip video memories, since parts of the memory may be switched-off (no static energy consumption) while keeping the data stored, leading to no extra external memory accesses to re-fetch the information. However, the NVM cells lifetime (aka. endurance property) *highly depends on the bit-toggling activity of the writing operations* [13]. If improperly balanced, the lifetime of a STT-RAM cell can be significantly reduced, compromising the overall memory system performance. Therefore, *there is a need for memory data management policies to increase the NVM lifetime in a hybrid video memory design*.

In summary, the high bit-toggling writing operations need to be directed to the SRAM cells, while lower bit-toggling writes are the preferred ones for the NVM part. *The knowledge of the bit-toggling can be accurately predicted using the video content and application-specific properties*. In our HEVC processing system, reference frames are read from/written to the on-chip video memory. Thus, the *key is to exploit the application-specific properties of HEVC reference frames to write the data on either NVM or SRAM parts of the hybrid on-chip video memory to increase the NVM cells lifetime*.

C. *Overview of Our Novel Concepts and Contributions*

In this work, we leverage the application-specific properties to design and manage an energy-efficient hybrid on-chip video memory architecture (enHyV, Fig. 3a) that is composed of several small hybrid memory modules (HyMs, Fig. 3b). To address SRAM limitations (low density and high static energy), enHyV integrates both SRAM and STT-RAM where the STT-RAM cells are intensively used to increase the overall energy efficiency. We demonstrate the applicability and benefits with the help of a parallel High Efficiency Video Coding (HEVC).

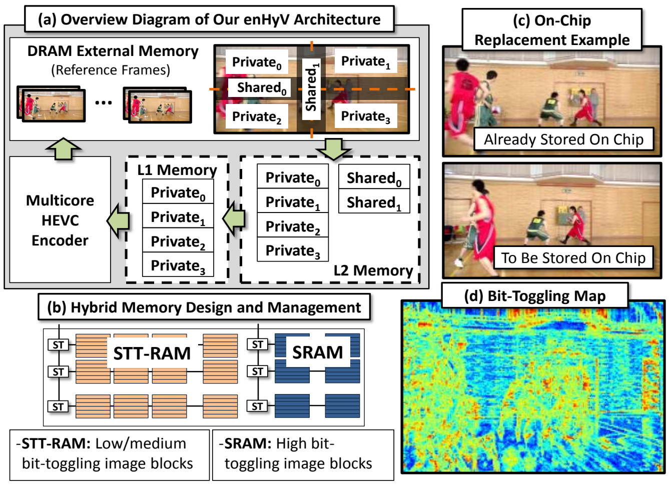


Fig. 3 Overview of our proposed enHyV architecture.

Our novel contributions in a nutshell are:

A Hybrid On-Chip Video Memory Architecture (enHyV; Section III): It is composed of multiple levels of private and shared HyMs, as shown in Fig. 3a. It consists of (1) private L1[2] HyMs to store the search window samples required for each HEVC processing core, (2) private and shared L2 HyMs to implement the reference frame level data reuse. The HyMs are managed by our energy-efficient management units. Our design methodology is based on offline statistical analyses using recommended test video sequences (that are different from the ones used for evaluation to avoid data biasing) [19]. Moreover, the

[1] The terms L, H and HH are used for a subjective comparison between STT-RAM and SRAM regarding its electrical characteristics: "L" means *low*, "H" means *high*, and "HH" means *very high*.

[2] L1 and L2 in this work do not refer to cache levels, but hybrid memory levels implemented as scratchpad memories.

sizing and energy-efficient management of HyMs are proposed to determine how much SRAM and STT-RAM cells are used at each memory level.

Energy-Efficient Management of enHyV (Section IV): It leverages application-specific properties to improve the STT-RAM cells endurance and to manage the energy consumption. As the lifetime is directly correlated with the bit-toggling activity (see Fig. 3c, d) during the write operations, we propose a dynamic data management that dynamically decides if the incoming reference frame block will be stored in the SRAM or STT-RAM cells. Furthermore, an adaptive power management technique exploits the high-endurance STT-RAM cell to switch off less used portions of L2 HyMs to obtain increased on-chip energy savings.

To the best of authors' knowledge, this is the first work of exploiting hybrid memory for multi-threaded video processing.

II. Related Works

The hybrid memory design exploiting the development of emerging non-volatile memory technology has been research target during the last years [9]-[16]. These works provide a solid foundation to enable these emerging technologies feasible to be integrated with CMOS logic circuitry of nowadays embedded manycore processors. However, these works may not efficiently support the video coding high memory demand, since they did not take into account application-specific properties. During the past decade, multiple works developed dedicated memory architectures for the H.264/AVC [18] and content-driven complexity and energy reduction for motion estimation [4][5]. The H.264/AVC-based memory architectures are not scalable enough to be energy efficient for HEVC encoders due to its novel coding tools and complex video processing flow. In another scenario, application-specific properties were exploited to reduce computational complexity and energy consumption (both off/on-chip parts) targeting the video coding [2][3] . In video Tile parallelized HEVC encoders, multiple cores request data at the same time from the shared memory system. Thus, several other factors need to be taken into account, e.g., memory contention, coherence protocols, and memory access scheduling schemes. The work in [8] designed a SRAM-based distributed memory architecture. The additional SRAM to improve the data reuse brings extra static energy consumption. Therefore, merely using SRAM it becomes infeasible when using a large number of parallel cores. The first dedicated hybrid memory design for video coding (called AMBER) was presented in [9]. It uses SRAM only as a FIFO buffer to hide the high write latency of STT-RAM cells, but not effectively being part of the storage system that may provide a high potential of energy-/performance-efficient design. AMBER stores all reference frames in the on-chip STT-RAM memories that incur high frequent-write, thus performing inefficient management under such scenarios and may only be feasible for a certain set of video resolutions. In particular, AMBER has two key limitations: (1) *it does not support parallel video processing*, which is inevitable to achieve high throughput; and (2) *it does not target lifetime improvements* and all write accesses are performed in the STT-RAM part.

III. Hybrid On-Chip Video Memory Architecture

Fig. 4 depicts the block diagram of our hybrid on-chip video memory architecture (enHyV) and its energy-efficient management scheme for parallel video processing. Each video Tile is assigned to a specific processing core. The enHyV is designed to increase the energy efficiency of video frames management (off-chip fetching and on-chip storage). Note, as a case study, we use parallelized HEVC encoders but the concepts are equally beneficial for other multi-threaded video processing applications. Our enHyV architecture is organized as two levels of hybrid memories (HyMs):

L1 Level: *n* private hybrid memories[3] (*PrivL1*) that store the search window samples, allowing intra-Tile data reuse between each CU processing.

L2 Level: *n* private HyMs (*PrivL2*) and *m* shared HyMs (*SharedL2*) that together can store one complete reference frame, providing combined intra-/inter-Tiles data reuse. The *SharedL2* is connected to the *PrivL1* by an interconnect bus and it is responsible for the overlapping regions storage (as in Fig. 3a). The *PrivL2* stores the remaining data (accessed by only one core). Each HyM of *PrivL2* has a direct connection to the corresponding *PrivL1* HyM.

In addition to two levels of HyMs, an *Access Management Unit* and a DRAM *Access Generator* are designed to jointly manage the off-/on-chip memories data interaction. The detailed data interaction is explained in the following.

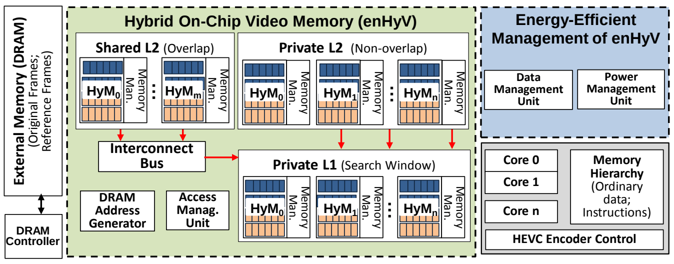


Fig. 4 The block diagram of our enHyV architecture.

A. *enHyV Access Management*

The access management unit starts to execute when a processing core *i* requests data from enHyV. Depending on the CU size being processed, different sizes of data blocks must be requested. In this sense, our enHyV management splits the requested data into several fixed-size data blocks and, for each one, it starts checking each HyM level. If a *PrivL1* HyM_i hit is verified, the data is simply forwarded to the core *i* and no more action is required. Otherwise, if a miss occurs, the L2 memories must be verified. *SharedL2* or *PrivL2* HyMs are mutual-exclusively accessed depending whether the requested data is in an overlap region or not. A L2 hit avoids external memory fetching and the data can be forwarded to the core *i*. Furthermore, the data is written into *PrivL1* HyM_i, increasing the intra-Tile data reuse. In case of L2 miss, the DRAM access generator is triggered to perform sequential external memory accesses to fetch reference frames data. The fetched data is then written into the corresponding *SharedL2*/*PrivL2* and *PrivL1* HyMs and passed to the requesting core.

Next we present design space exploration results using statistical analysis of video properties to properly design and size the SRAM and STT-RAM arrays for each involved HyM of enHyV.

[3] Let *n* be the number of Tiles and *m* be the number of Tiles boundaries.

B. *Design Space Exploration of Hybrid Memories (HyMs)*

As already discussed, STT-RAM presents low static energy consumption while having high density. It allows us to designate the most part of the HyM to be composed of the STT-RAM array. In the meantime, blocks from the reference frame that cause high bit-toggling activity strongly decrease the STT-RAM lifetime, minimizing its non-volatility advantage. Thus, a small portion of SRAM is used to handle with these blocks. Although SRAM does not degrade from bit-toggling activity, it costs a large area and a high static energy consumption. Therefore, the main challenge involved in the HyMs design *is to leverage application-specific properties to design a well-balanced combination of SRAM and STT-RAM to minimize the static energy consumption whereas increasing the STT-RAM cells lifetime.* We define the bit-toggling activity (*BT*) during a HyM write operation of a block B_0 over an already stored block B_1 as the number of bits that toggles during the operation divided by the total number of written bits, as in Eq. (1). B_{Size} is the horizontal/vertical block size and NB_{Sample} the number of bits per sample. The *toggling_bits* function returns the number of collocated bits that are different between two numbers.

$$\mathrm{BT}(\mathrm{B}_0, \mathrm{B}_1) = \frac{\sum_{y=0}^{\mathrm{B}_{\mathrm{Size}}} \sum_{x=0}^{\mathrm{B}_{\mathrm{Size}}} toggling\_bits(\mathrm{B}_{0(x,y)}, \mathrm{B}_{1(x,y)})}{\mathrm{B}_{\mathrm{Size}}^{\ 2} * \mathrm{NB}_{\mathrm{Sample}}} \qquad (1)$$

Fig. 5 depicts the design space exploration controlled by an external parameter: the bit-toggling threshold ($\mathrm{BT_{TH}}$). Reference frame blocks that lead to bit-toggling activities lower than $\mathrm{BT_{TH}}$ are assigned to STT-RAM, while higher values will direct the block to SRAM. Our exploration varies the $\mathrm{BT_{TH}}$ from 0 (no activity) to 1 (maximum activity, all bits toggle) in steps of 0.01. We analyze our two optimization target variables: STT-RAM lifetime (Fig. 5a) and SRAM size (Fig. 5b), since we know that the static energy efficiency is limited by the amount of SRAM cells (as discussed in Section I.B). To find the best design point, we analyze an efficiency plot that relates both variables (see Fig. 5c). We run this exploration for a set of video test sequences following our evaluation methodology (described in Section V). The maximum efficiency point was discovered when BT_{TH}=0.24. Using this design point, we have that the SRAM usage factor (α_{SRAM}) is equal to 35% and the STT-RAM lifetime can be improved near to the optimal case (when no bit toggles): 0.83 normalized lifetime, as detailed in Section VI.C. From the enHyV perspective, α_{SRAM}=36% means that the SRAM array will be sized as 36% of the STT-RAM capacity. Note that the α_{SRAM} factor can be used for any type of HyM design (*PrivL1*, *PrivL2* and *SharedL2*).

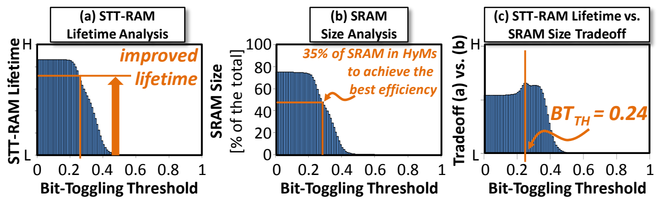


Fig. 5 (c) Design space exploration for joint (a) STT-RAM lifetime and (b) SRAM size optimization

The toggling activity at bit level is also exploited in enHyV. Fig. 6 depicts the accumulated statistics for toggling occurrences for each bit position using *ParkScene* and *NebutaFestival* test video sequences [19]. It can be noticed that we have near-zero bit-toggling activity for the two most significant bits (MSB) of the two sequences. Therefore, it means that even for blocks with high average bit-toggling activities, the two MSB toggle with a very low probability. We exploit this property by always storing the two MSB in the STT-RAM, this way reducing the SRAM size (saving further static energy) while not penalizing the STT-RAM cells lifetime. This enables us to realize a fine-grained hybrid memory organization.

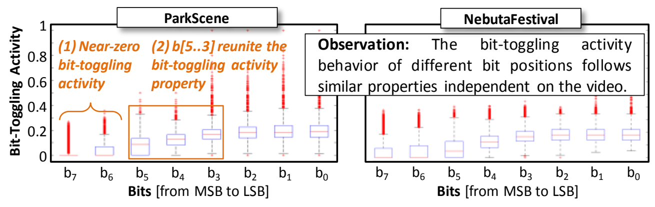


Fig. 6 Bit-toggling activity of different bit positions.

Another important aspect from Fig. 6 is that the bit range from b_5 to b_3 inherits the bit-toggling activity of the entire 8-bit sample. It means that we only need to compare these three specific bits of the two involved data to approximate the bit-toggling activity of this write operation. We exploit it by generating a bit-toggling key (BT_KEY) composed of only these three bits of some specific samples from reference frame blocks. This key aims to serve as an identifier that must be stored by the data management unit. *Our goal is to design an energy-efficient way to estimate the bit-toggling activity of each write operation.* Details regarding this data management unit are given in Section IV.A.

C. *HyM Physical and Logical Organization*

Fig. 7 presents the physical organization of the HyMs. The size of the HyMs ($Size_{HyM}$) is different depending on the enHyV level, as in Eq. (2): *PrivL1* HyMs must store a complete search window sized according to horizontal and vertical dimensions SW_H and SW_V; *PrivL2* HyM size depends on the video dimensions ($Frame_H$ or $Frame_V$, depending on the video Tiles direction) and the number of used video Tiles (N_{Tiles}); *SharedL2* must support the overlapping region size, which varies according to the search window dimension (SW_H or SW_V) and the frame dimension. The B_{Size} in Eq. (3) refers to the basic access unit used for every enHyV memory transaction. Our sizing methodology works for any value of B_{Size}. It is interesting for our specific HEVC case to use B_{Size} as 8, since 8x8 is the smallest CU size allowed in the HEVC latest working draft [1].

$$\mathrm{HyM_{Size}} = \begin{cases} \lceil (\mathrm{SW_H} * \mathrm{SW_V}) / \mathrm{B_{Size}} \rceil & \text{If } PrivL1 \\ \lceil (\mathrm{Frame_H} / \mathrm{Frame_W}) / (\mathrm{B_{Size}} * \mathrm{N_{Tiles}}) \rceil & \text{If } PrivL2 \\ \lceil (\mathrm{SW_V}/2) * \mathrm{H_{Frame}} / \mathrm{B_{Size}} \rceil & \text{If } SharedL2 \end{cases} \qquad (2)$$

$$\mathrm{B_{Size}} = Smaller\ CU\ Size = 8 \qquad (3)$$

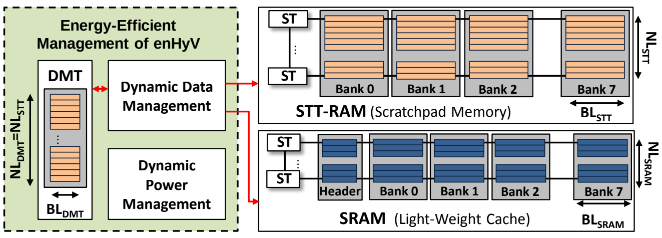


Fig. 7 Hybrid memory (HyM) organization.

The sizing formulas of the STT-RAM part are expressed in Eq. (4)-(6). The STT-RAM array is designed as a scratchpad memory and each B_{Size}xB_{Size} reference frame block has a fixed and unique position. This part must be sized to store all possible reference blocks: (i) the entire search window for PrivL1 HyMs and (ii) the corresponding reference frame part for *PrivL2* and *SharedL2*. Thus, the STT-RAM size ($Size_{STT}$) follows the total size of the HyM, expressed by the already calculated HyM_{Size}. The STT-RAM memory part is composed of B_{Size} parallel banks. Each bank has NL_{STT} memory lines of NB_{STT} bits. The set of memory lines of the same row composes one reference frame block. With this design, one complete B_{Size}xB_{Size} block can be completely written/read in one clock cycle. Equations (7)-(9) present the sizing variables of the SRAM portion. Since only a few part of the data is assigned to SRAM (to alleviate STT-RAM from high bit-toggling activities), we design it as a fully-associative light-weight cache memory, as in Fig. 7. The data array size of SRAM cache ($Size_{SRAM}$) represents a fraction of the HyM_{Size}, using the already calculated α_{SRAM} factor. The header of each line is composed of the *(x,y)* spatial coordinates of the reference block. Similar to the STT-RAM design, one entire row of lines of the B_{Size} banks stores one complete reference block. Since the two MSB are always stored in the STT-RAM array, the memory line size NL_{SRAM} is smaller than the NL_{STT}, reducing by 25% the data array size.

$$\mathrm{Size_{STT}} = \mathrm{HyM_{Size}} \quad (4)$$
$$\mathrm{BL_{STT}} = \mathrm{B_{Size}} * \mathrm{NB_{Sample}} = 8 * 8\mathrm{bits} = 64\mathrm{bits} \quad (5)$$
$$\mathrm{NL_{STT}} = \mathrm{Size_{STT}}\,/(\mathrm{B_{Size}} * \mathrm{BL_{STT}}) \quad (6)$$
$$\mathrm{Size_{SRAM}} = \lceil \mathrm{HyM_{Size}} * \alpha_{\mathrm{SRAM}} * 0.75 \rceil \quad (7)$$
$$\mathrm{BL_{SRAM}} = \mathrm{B_{Size}} * (\mathrm{NB_{Sample}} - 2) = 6 * 8\mathrm{bits} = 48\mathrm{bits} \quad (8)$$
$$\mathrm{NL_{SRAM}} = \mathrm{Size_{SRAM}}/(\mathrm{B_{Size}} * \mathrm{BL_{SRAM}}) \quad (9)$$

The logical organization of our HyMs is demonstrated in Fig. 8, where a set of 3x3 reference blocks are taken as example. As decision from the data management unit, the blocks (0,0), (1,1) and (2,2) are considered to provoke high bit-toggling activity and must be partially stored in the SRAM cache; while the remaining blocks are completely stored in the STT-RAM scratchpad memory. For data management purposes, a Data Management Table (DMT in Fig. 7 and Fig. 8) was also designed.

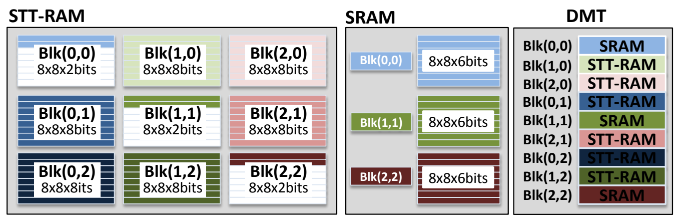


Fig. 8 HyM data assignment example

The DMT design is fully explained in Section IV along with our energy-efficient management of enHyV.

IV. Energy-Efficient Management of enHyV

Besides the enHyV memory organization and access management unit, we also developed an energy-efficient management of the designed HyMs. The data management unit aims to increase the STT-RAM cells lifetime, allowing much more effective power management. Details about the schemes are given as follows.

A. Data Management Unit

To estimate as simple as possible the bit-toggling activity during a write operation, we define the bit-toggling key BT_KEY as in Eq. (10). This key is generated at the moment before the write operation and it consists in a set of wires getting the bits from b_5 to b_3 of eight specific samples resultant from the *downsamplingTo8* operation from the reference frame block. The choice of these specific bits was taken due to the conclusion (2) from Fig. 6. The *downsamplingTo8* function selects eight equally-spaced samples of a block, reducing its representation resolution. In our case, due to the spatial correlation between near pixels of a video frame, it is possible to discard many ones and still maintain the bit-toggling activity property. Thus, the BT_KEY will have 3x8=24 bits. The proposed estimated bit-toggling activity (EBT) calculates the number of bits that differs between the BT_KEY of the two involved blocks, as in Eq. (11). This strategy is developed to avoid a complete read operation to fetch the entire block to, just then, perform the bit-toggling activity evaluation. We store the BT_KEY of each stored reference block in a very fast special table, called Data Management Table (DMT). Besides the BT_KEY, the DMT also stores a flag indicating whether the corresponding reference block is stored in the STT-RAM or SRAM array (called presence bit). A practical example is depicted in and Fig. 8, where each one of the nine reference blocks has a DMT entry with its corresponding presence bit value. Thus, the DMT line consists in BL_{DMT} bits, as in Eq. (12). Fig. 9 presents 2D maps and histograms to show the high correlation between the actual bit-toggling activity (BT) and the estimated one (EBT). For a B_{Size} equals to 8, we reduce the number of required bits to derive the toggling activity by ~22x using the EBT metric. The circuit to compute the number of bits that differ between two BT_KEY can be implemented with 24 1-bit XOR gates and a tree of 1-bit full adders, not representing neither energy nor performance significant penalty for the HyMs.

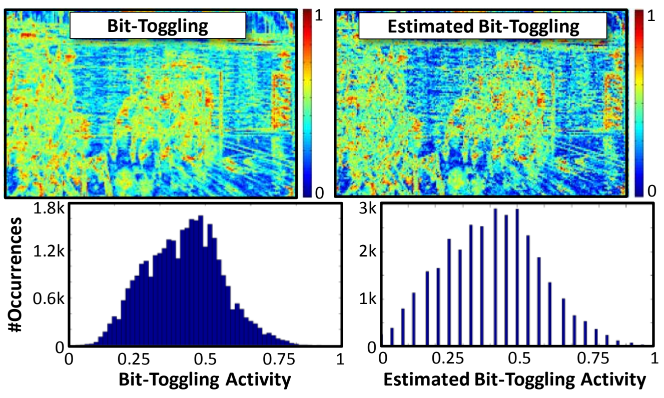


Fig. 9 Statistical correlation between BT and EBT metrics.

$$\mathrm{BT\_KEY(B)} = concat([\mathrm{b_5 .. b_3}] \,|\, \mathrm{b} \in downsampleTo8(\mathrm{B})) \quad (10)$$
$$\mathrm{EBT(B_0, B_1)} = \mathrm{toggling\_bits(BT\_KEY(B_0), BT\_KEY(B_1))/24} \quad (11)$$
$$\mathrm{BL_{DMT}} = 1 + 24 = 25\mathrm{bits} \quad (12)$$

Fig. 10 presents the data management steps for a HyM write operation. First, the BT_KEY for the block that is being written is generated (*line 2*). Then, the BT_KEY of the already stored block must be retrieved from the DMT (*line 4*) and the estimated activity α_{EBT} is then calculated (*line 5*). The α_{EBT} is then compared with the offline statistical defined threshold BT_{TH} (*line 6*). In the case that

α_{EBT} is higher than BT_{TH}, the block to be written is divided to be partially stored in the SRAM and STT-RAM (*lines 7-10*). In this case, we sacrifice SRAM dynamic energy to increase the STT-RAM cells lifetime. We demonstrate in the results section that this spent energy is very small compared to the overall savings provided by enHyV. For α_{EBT} lower than BT_{TH}, the block is completely written into the STT-RAM cells (*line 13*). The DMT is updated with the new $Bit_{Presence}$ (*lines 11-14*) and with the BT_KEY of the written block (*line 16*).

The HyM read operation is much simpler than the write case, since no decision must be taken. The DMT is just accessed to get the presence bit and, depending on this, STT-RAM or SRAM/STT-RAM will be accessed. Finally, the data is forwarded to the requesting core by the access management unit. As the data management unit increases the STT-RAM lifetime, the power management unit can power-gate unused cells with minimized risk of data re-fetching from external memory.

```
1.  manageWrite(Hybrid Memory: HyM; Data 8x8 Block: $Block_{ToBeWritten}$;
     8x8 Block Positions: x, y)
2.  $Key_{ToBeWritten}$ := BT_KEY($Block_{ToBeWritten}$);   //generate key - Eq. (10)
3.  $Address_{Data}$ := genPhysicalAddress(x, y);   //calculate physical address
4.  $Key_{ToBeReplaced}$ := DMT[$Address_{Data}$][23..0]; //get already stored key
5.  $\alpha_{EBT}$ := EBT($Key_{ToBeWritten}$, $Key_{ToBeReplaced}$);   //estimate activity - Eq. (11)
6.  If ($\alpha_{EBT} > BT_{TH}$) Then        //high bit-toggling data
7.      $Block_{STT-RAM}$ := ((b[7..6] | $\forall$ b $\in$ $Block_{ToBeWritten}$);  //2-bit split
8.      $Block_{SRAM}$ := (b[5..0] | $\forall$ b $\in$ $Block_{ToBeWritten}$)     //6-bit split
9.      $HyM_{STT-RAM}$[$Address_{Data}$].write($Block_{STT-RAM}$);   //STT-RAM write
10.     $HyM_{SRAM}$.write(x, y, $Block_{SRAM}$);          //SRAM write
11.     DMT[$A_{Data}$][25].write('0');                //DMT update – Presence Bit
12. Else                                           //low bit-toggling data
13.     $HyM_{STT-RAM}$[$Address_{Data}$].write($Block_{ToBeWritten}$);  //STT-RAM write
14.     DMT[$Address_{Data}$][25].write('1');          //DMT update – Presence Bit
15. End If;
16. DMT[$A_{Data}$][23..0].write($Key_{ToBeWritten}$);               //DMT update
```

Fig. 10 Data management for a HyM write operation.

B. Power Management Unit

Our HyMs were designed to be able to operate in two power states: *ON* (V_{ON}=V_{DD} volts) and *OFF* (V_{OFF}=0 volts). Due to the non-volatility characteristic of STT-RAM, the data is kept stored in the memory cell even when *OFF* state is assigned. This is not the case for SRAMs, leading to a data loss and requiring later an external memory re-fetching. Typically, the *PrivL2* and *SharedL2* HyMs are very much larger than *PrivL1* HyMs, leading to significantly higher on-chip energy consumption. In this sense, our power management concentrates effort in the L2 HyMs, resulting in a great impact in the enHyV overall static energy (as demonstrated by our experimental results in Section VI).

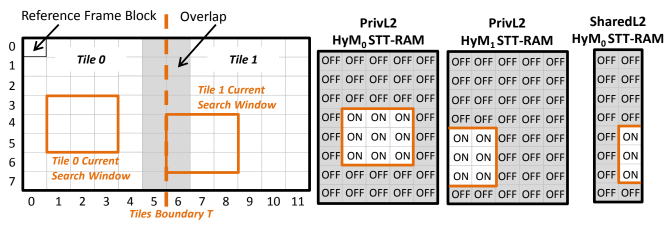


Fig. 11 Example: Power management for STT-RAM for a 2-Tile scenario.

The STT-RAM power management is performed at CTU level and it is depicted in Fig. 11. At the beginning of a CTU processing, our scheme checks all STT-RAM positions against the search window limits. Note that the *PrivL2* HyMs must be checked only against the search window of its corresponding core, while the *SharedL2* HyMs must be analyzed considering all cores that share this overlapping region. For the intersecting area, all related STT-RAM cells are assigned as *ON*. Otherwise, the *OFF* state is assigned. This assignment can be observed in the Fig. 11 example. It can be noted that all memory lines inside the search window limits are switched on, even not knowing if all data will be requested from *PrivL1*. It facilitates the control unit and provides low read latency when any of the reference blocks inside the search window limits are retrieved from the L2 HyMs. Since all CUs processing inside the CTU will perform memory access inside the search window limits, we guarantee long sleep durations to amortize the wake-up energy and latency overhead.

We also developed a power management strategy for the SRAM part. Our scheme acts at frame-level for SRAM management. Since the bit-toggling activity can vary depending on the video, we may have different SRAM usages. In order to adapt to the video content, our scheme run-time analyzes the SRAM usage Probability Density Function (PDF) for previous frames to estimate the usage for the next reference frame fetching. The TH_{SRAM} threshold determines the amount of SRAM to be switched to the *ON* state, while the other parts are assigned to *OFF* state. The TH_{SRAM} is run-time calculated as $TH_{SRAM}=\mu+\sigma$, where μ (average) and σ (standard deviation) are parameters from PDF of SRAM usage statistics during past frames processing. In case the SRAM usage surpasses the estimated TH_{SRAM}, wake-up energy and latency are required to assign *ON* power state. However, due to the very high similarity of consecutive frames, we also guarantee a long sleep duration and significant on-chip static energy savings.

V. Evaluation Methodology

A custom simulator was developed to capture the video memory access traces. Based on adopted memory power models, it estimates the energy consumption for on- and off- chip memory parts, as well as the STT-RAM cells lifetime improvement and access latency overhead. Details regarding our memory electrical models and video coding configurations are given as follows.

Table II SRAM and STT-RAM Characterization for 65nm [12]

Parameter	SRAM	STT-RAM	STT-RAM Ratio
Area (*F=feature size*)	$146F^2$	$37F^2$	3.94x smaller
Static Power (mW/mm^2)	25.2	2.7	9.33x lower
Read Latency (*ns*)	2.795	2.795	-
Write Latency (*ns*)	2.795	11.287	4x higher
Read Dyn. Energy (*nJ*)	0.151	0.155	-
Write Dyn. Energy (*nJ*)	0.151	2.942	15.5 higher

A. On-Chip SRAM and STT-RAM Power Models

The CACTI 6.5 tool [24] was used for SRAM energies/ latencies for 32nm memories. We adopt the static energy reduction and wake-up latencies/energies from the analytical model depicted in [23]. We adopted the STT-RAM memory characterization from [12]. This work evaluated STT-RAM and developed scaling factors to be compared with SRAM. Table II presents the used scaling factor in our work to evaluate the performance and energy efficiencies of enHyV different technologies.

B. External LPDDR DRAM Memory Model

The memory is composed of several banks. Each bank is a row-column matrix, where the number of rows (N_{Rows}) represents the addressing space and the number of columns ($N_{Columns}$) is directly related to the page size (P_{Size}). Each row-column intersection stores a memory word of size W_{Size}. Each memory access will initially cause a page activation, to pass the activated data to the page buffer. If consecutive accesses are located in the same page, the memory controller needs just to address a specific column of the page buffer (called *burst read/write operations*). If other memory page is addressed, the current active page is precharged and a new page is activated.

The 4-Gbit Low-Power DDR2 (LPDDR2) DRAM MT42L128M16D1GU-25WT [20] chip was used. The main specifications are: *Vdd*=1.2V, *Freq*=533MHz, W_{Size}=32bits, P_{Size}=512B, N_{Rows}=16K and $N_{Columns}$=2K. The total energy is derived by the composition of six components: page activation energy, write energy, read energy, I/O pins energy, refresh energy and standby energy [21][22].

C. Video Coding Experimental Setup

Our experimental setup for HEVC evaluation considers recommended test conditions [19] using the HEVC test model (HM 11.0) [6]. We execute our experiments for 4-Tile and 8-Tile scenarios (each video Tile executes on a dedicated processing core). As inputs for our experiments, we select six video test sequences from the JCT-VC recommended test video benchmark [19]: *BasketballDrive* (*BDrive*), *BQTerrace*, *Cactus*, and *Kimono* (HD1080: 1920x1080), and *PeopleOnStreet* (*People*) and *Traffic* (2K: 2560x1600). Note that we use different video sequences for the analysis and design space exploration of HyMs: *ParkScene* (HD1080); *NebutaFestival* and *SteamLocomotive* (2K). By using different sets of video benchmarks in the design and evaluation parts, we guarantee that our results are not biased towards the design decisions using the same videos. Other encoder specifications are: 128x128 search window (default value in HM 11.0), GOP=8, CABAC, FRExt, Random Access configuration, and TZ Search block matching for the motion estimation.

For comparison purposes, we also implemented a SRAM-only version of enHyV, where the STT-RAM arrays are implemented as SRAM. Additionally, it is also used for comparison to the state-of-the-art AMBER hybrid memory for *non-parallelized* HEVC [9]. In the case of AMBER, as their STT-RAM memory arrays do not support parallel access, we replicate the entire memory infrastructure according to the number of used video Tiles. For external memory energy evaluation, the Level C+ traditional data-reuse scheme [9] was used to estimate our savings.

VI. Experimental Results and Discussions

A. On-Chip Energy Results

Fig. 12 depicts the static energy consumption results of our enHyV architecture compared to the SRAM-only solution and to the state-of-the-art related works. Our hybrid design, even not taking into account the power management savings, is able to reduce the leakage energy by 50%-62%, compared to the SRAM-only case for 4-Tile and 8-Tile scenarios, respectively. Furthermore, our power management unit is able to improve by 33% the static energy reduction, achieving 66%-75% of savings related to SRAM-only. Also, as our power management is adaptive to the video content due to the proper analysis of the SRAM usage of already frames processing, it provides improved reductions when low motion videos are encoded, like *Traffic* (up to 80% of savings for the 4-Tile scenario). Considering the state-of-the-art AMBER hybrid video memory architecture [9], the enHyV consumes 55%-76% less static energy consumption, on average. AMBER stores all reference frame on-chip, which leads to very large STT-RAM arrays. Even with the low leakage energy consumption of STT-RAM memory cells (around 90% lower than SRAM [12]), AMBER is not efficient when video Tiles are processed in parallel. For the 8-Tile scenario, AMBER consumes the highest static energy, which is up to 4.28x higher than enHyV.

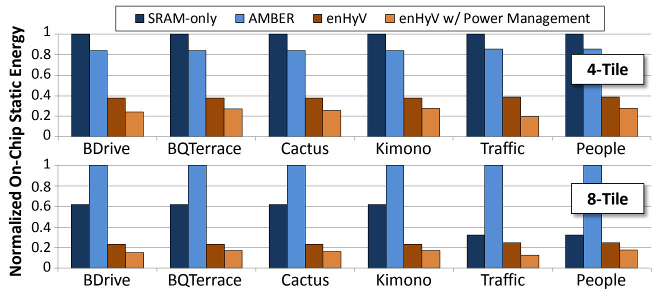


Fig. 12 On-chip static energy comparison of enHyV with related works.

B. Off-Chip Energy Results

The external memory energy analysis is presented in Fig. 13. As enHyV can completely reuse the samples within the same reference frame, we can achieve huge savings compared to Level C+: 98% on average for the HD and 2K tested videos. As Level C+ employs search window data reuse, its gains are limited when parallelized video Tiles are processed, since inter-Tiles data reuse is not exploited. On the other hand, AMBER completely avoids the need of external memory communication to fetch the reference frames, since the complete decoding picture buffer is implemented as on-chip STT-RAM memories. Observing Fig. 13, enHyV architecture consumes near-zero off-chip energy for reference frames transmission. In the meantime, as already discussed, our well-balanced hybrid design of SRAM and STT-RAM, with the power management unit, can reduce the on-chip energy compared to AMBER. Furthermore, enHyV also improves STT-RAM lifetime, guaranteeing efficient STT-RAM power management.

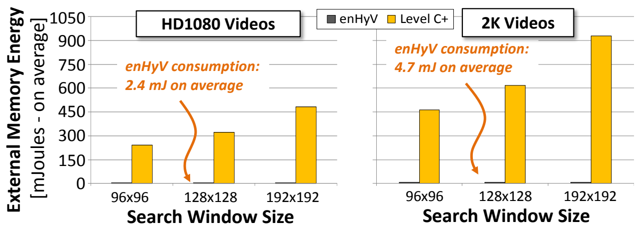


Fig. 13 External memory energy consumption of enHyV compared to Level C+ [9] (average for 4-/8-Tile scenarios).

C. STT-RAM Lifetime Results

The data management unit of enHyV significantly improves the STT-RAM cells lifetime, as demonstrated in Fig. 14. In this analysis, we plot the normalized STT-RAM lifetime bordered by the lifetime of enHyV without any data management (zero value) and by the best case scenario, where no bit toggles occurs during write operations (1.0 value). On average, we have a normalized lifetime of 0.83, nearer to the best case than the enHyV basic approach without any management. Our data management of enHyV can achieve higher lifetime improvements for low-textured videos, like *Kimono* with 0.85 normalized lifetime. In another vein, highly detailed scenes lead to high bit-toggling activities, requiring high SRAM usage to alleviate STT-RAM cells. As we design enHyV for the average case, lifetime is less improved for this kind of videos. However, even for the worst case scenario, our scheme still can improve the lifetime (0.79 lifetime for *Traffic*).

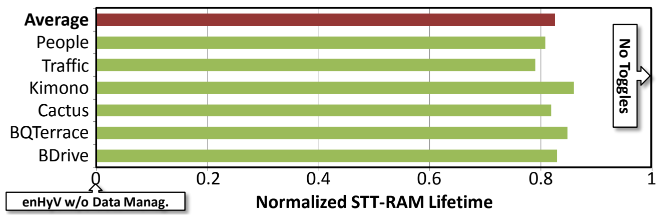


Fig. 14 Normalized STT-RAM cells lifetime.

D. *enHyV Overhead: Access Latency and Dynamic Energy*

STT-RAM poor performance and energy efficiencies when writing along with our energy-efficient management scheme imposes overhead to the access latency and dynamic energy during the read/write operations. Fig. 15**Erro! Fonte de referência não encontrada.** depicts these overheads. As already discussed in the motivational section, video memories have the characteristic of being high read-intensity applications. Thus, the write overhead is amortized by the high amount of read operations. In terms of average access latency, we noticed 1.3% of increased latency on the average case. For the dynamic energy perspective, the overhead is 7.7% on average. Taking all these small overhead into account, our enHyV architecture still achieves the highest energy efficiency compared to state-of-the-art works with insignificant access latency overhead.

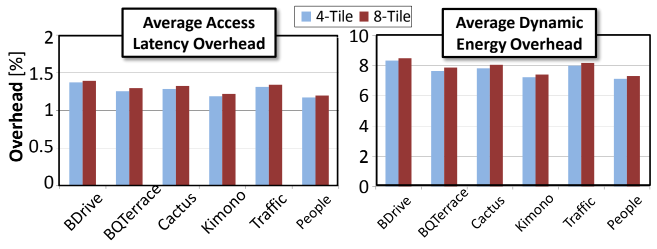


Fig. 15 enHyV overhead analysis: HyMs access latency and dynamic energy.

VII. Conclusions

This work presented an energy-efficient hybrid on-chip video memory (enHyV) architecture for advanced multi-threaded video processing applications and demonstrated the practical benefits using the next-generation parallelized High Efficiency Video Coding application. Small SRAM arrays were designed in combination with STT-RAM to alleviate the bit-toggling activity during write operations, increasing STT-RAM cells lifetime. For energy-efficient management, an adaptive data management unit selects either SRAM or STT-RAM to be written depending on the estimated bit-toggling activity of each write operation. We additionally employed a power management unit that saves static energy from both SRAM and STT-RAM parts. Our experimental results showed that enHyV design could achieve average static energy savings of 66%-75% and 55%-76% compared to SRAM-only version of enHyV and state-of-the-art AMBER architecture, respectively. External energy consumption could be significantly minimized compared to traditional Level C+ scheme (98% reduced), resulting on negligible off-chip communication overhead. Our data management unit could significantly improve the STT-RAM cells lifetime, achieving 0.83 normalized lifetime. Our enHyV architecture enables energy-efficient implementations of parallelized video processing applications.

VIII. Acknowledgements

This work was partly supported by DFG as part of Transregional. Collaborative Research Centre "Invasive Computing" (SFB/TR 89).

IX. References

[1] JCT-VC, “High Efficiency Video Coding (HEVC) text specification draft 10 (for FDIS & Consent)”, Doc.: JCTVC-L1003_v9, 2013.
[2] B. Zatt, M. Shafique, F. Sampaio, S. Bampi, J. Henkel., "Run-time adaptive energy-aware motion and disparity estimation in multiview video coding", ACM/EDA/IEEE DAC, pp. 1026-1031, 2011.
[3] M. Shafique, B. Zatt, S. Bampi, J. Henkel. “Adaptive Power Management of On-Chip Video Mamory for Multiview Video Coding”, ACM/EDA/IEEE DAC, pp. 866-875, 2012.
[4] M. Shafique et al, "An HVS-based Adaptive Computational Complexity Reduction Scheme for H.264/AVC video encoder using Prognostic Early Mode Exclusion," IEEE/ACM DATE , pp.1713-1718, 2010.
[5] M. Shafique et al., "enBudget: A Run-Time Adaptive Predictive Energy-Budgeting scheme for energy-aware Motion Estimation in H.264/MPEG-4 AVC video encoder," IEEE/ACM DATE, pp.1725-1730, 2010.
[6] JCT-VC. HEVC Software SVN, 2011. Available in: <https://hevc.hhi.fraunhofer.de/ >
[7] B. Pourazad et al., “HEVC: The New Gold Standard for Video Compression: How Does HEVC Compare with H.264/AVC?,” IEEE CeM, pp. 36-46, 2012.
[8] F. Sampaio, M. Shafique, B. Zatt, S. Bampi, J. Henkel, “dSVM: Energy-Efficient Dist. Scratchpad Video Memory Architecture for the Next-Generation High Efficiency Video Coding”, IEEE/ACM DATE, 2014.
[9] M. U. K. Khan, M. Shafique, J. Henkel. “AMBER: Adaptive Energy Management for On-Chip Hybrid Video Memories”. IEEE/ACM ICCAD, pp. 405-412, 2013.
[10] Y.-T. Chen et al., "Dyn. reconfigurable hybrid cache: An energy-efficient last-level cache design", IEEE/ACM DATE, vol., no., pp.45,50, 12-16, 2012.
[11] K. Abe et al, "Novel hybrid DRAM/MRAM design for reducing power of high performance mobile CPU," IEDM, pp.10-13, 2012.
[12] X. Dong et al., "Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement," ACM/EDA/IEEE DAC, pp.554,559, 2008.
[13] X. Wu et al. “Hybrid Cache Architecture with Disparate Memory Technologies” IEEE/ACM ISCA. pp. 34-45, 2009.
[14] A. Jog et al., “Cache revive: architecting volatile STT-RAM caches for enhanced performance in CMPs,” ACM/EDA/IEEE DAC, 2012, pp. 243–252.
[15] X. Wu et al, “Design exploration of hybrid caches with disparate memory technologies,” ACM TACO, vol. 7, no. 3, pp. 1–34, Dec. 2010.
[16] Li, L. Zhang et al., “Hybrid cache architecture with disparate memory technologies,” in ACM SIGARCH CAN, 2009, vol. 37, pp. 34–45.
[17] A. Jadidi, "High-endurance and performance-efficient design of hybrid cache arch. through adaptive line replacement", ACM/IEEE ISLPED, p.79,84, 2011.
[18] C.-Y. Chen et al. “Level C+ Data Reuse Scheme for Motion Estimation with Corresponding Coding Orders”, IEEE TCSVT, v. 16, n. 4, p. 553-558, 2006.
[19] F. Bossen, “Common test conditions and software reference configurations”, ITU-T/ISO/IEC JCTVC-K1100, October 2012.
[20] Micron. “4Gb: x16, x32 Mobile LPDDR2 SDRAM S4”, 168p, 2013.
[21] Micron. “TN-46-03 Calc. DDR Mem. System Power”. Rev. B 3/05 EN, 2005.
[22] Micron. “TN-46-12 DRAM Power-Saving Features/Calcs”. Rev. B 5/09, 2009.
[23] H. Singh et al., "Enhanced leakage reduction techniques using intermediate strength power gating", IEEE TVLSI, vol. 15, no. 11, pp. 1215-1224, 2007.
[24] S. Thoziyoor et al, “CACTI 5.1 tech. report,” HP Labs, 2008.

Approximation-Aware Multi-Level Cells STT-RAM Cache Architecture

ABSTRACT

Current manycore processors exhibit large on-chip last-level caches that may reach sizes of 32MB – 128MB and incur high power/energy consumption. The emerging Multi-Level Cells (MLC) STT-RAM memory technology improves the capacity and energy efficiency issues of large-sized memory banks. However, MLC STT-RAM incurs non-negligible protection overhead to ensure reliable operations when compared to the Single-Level Cells (SLC) STT-RAM.

In this paper, we propose an approximation-aware MLC STT-RAM cache architecture, which is partially-protected to restrict the reliability overhead and in turn leverages variable resilience characteristics of different applications for adaptively curtailing the protection overhead under a given error tolerance level. It thereby improves the energy-efficiency of the cache while meeting the reliability requirements. Our cache architecture is equipped with a latency-aware hardware module for double-error correction. To achieve high energy efficiency, approximation-aware read and write policies are proposed that perform approximate storage management while tolerating some errors bounded within the user-provided tolerance level. The architecture also facilitates run-time control on the quality of applications' results. We perform a case study on the next-generation advanced video encoding applications that exhibit memory-intensive functional blocks with varying resilience properties and inherent support for parallelism.

Experimental results demonstrate that our approximation-aware MLC STT-RAM based cache architecture can improve the energy efficiency compared to state-of-the-art fully-protected caches (7%-19%, on average), while incurring minimal quality penalties in the output (-0.219% to -0.426%, on average). Furthermore, our architecture supports complete error protection coverage for all cache data when processing non-resilient application. The hardware overhead to implement our approximation-aware management negligibly affects the energy efficiency (0.15%-1.3% of overhead) and the access latency (only 0.02%-1.56% of overhead) of our architecture.

1. INTRODUCTION

Advanced manycore processors have tight memory energy budget when processing massively parallel applications. The on-chip memory infrastructure (typically composed of cache memory hierarchy) must alleviate the main memory data communication by employing multiple levels (up to L4, in some cases) of caches. The landscape of recent on-chip systems (see Fig. 1) shows that the memory consumes a significant portion of the footprint, for instance, up to 128MB in IBM Power 8.

Traditional 6T-SRAM cell based memories incur large area and high leakage power consumption [23][24]. Non-volatile memory technologies have emerged as an attractive alternative option for implementing large-sized on-chip memories. The Spin-Transfer Torque RAM (STT-RAM) stands out as one of the most promising technologies. It provides high scalability, improved endurance and resilience to soft errors, and reduced leakage power consumption [20]. Due to its inherent shortcomings of asymmetric read/write behavior and poor write access efficiency, STT-RAM has been mostly preferred for implementing the last-level caches [23]. Typically, last-level caches require large memory banks and have low read/write intensity characteristic (compared to the other cache levels). Therefore, in STT-RAM based caches, in order to increase the density of memory cells, *multi-level cells* (MLC) design is a promising design alternative to *single-level cell* (SLC). In MLC STT-RAM, one physical memory cell is able to store more than one logic bit. Recent studies [7][10] have demonstrated the feasibility of MLC-based design of STT-RAM towards scalability for larger caches banks.

For the MLC support in STT-RAM technology, the resistance range of the magnetic tunnel junction (MTJ) is further discretized to store more than 2 logic states. However, due to process variations, memory arrays based on MLC STT-RAM have tight sense margin(s) between adjacent resistance states [7]. It leads to more frequent error occurrences during memory read and write operations. Even though it is a more energy-efficient solution, MLC STT-RAM requires extra circuitry to guarantee the reliability of the memory system. Therefore, a key research challenge *is to design energy-efficient fault-tolerant cache memories to enable multi-level cell STT-RAM usage in advanced manycore processors.*

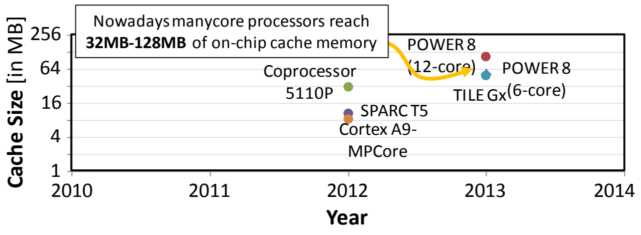


Fig. 1. Trend of cache memory size for practical case studies of manycore processors.

1.1 State-of-the-Art and Key Research Challenges

Several works during the past decade proposed reliability optimization schemes for SLC-based cache memories [12][13][15]. The work [12] proposes a fault-tolerant cache architecture considering magnetic-RAM (MRAM) technology – first generation of STT-RAM. In work [15], fault tolerance techniques are proposed for management of voltage-scaled cache faults. A fault-tolerant scratchpad memory (FTSPM) is proposed by [13], using non-volatile memory technologies targeting the vulnerability to soft-errors for the application data. However, none of these works considers large-sized MLC-based memory designs. Furthermore, MLC-based memories require specialized errors handling due to their specific behavior on fault occurrences (as detailed in the background section).

Recently reliability optimization for MLC STT-RAM based memories have gathered community's interest as a major research challenge [7][8][9][10][11][14]. The works in [7][10] present an intensive design exploration (at the transistor level) for analysis of reliability characteristics of MLC cells. The proposals in [8][10] employ wear-rate leveling techniques to improve the reliability of STT-RAM cells by improving their endurance, leading to less frequent wear-out errors. In [14], the same endurance problem is targeted by assuming wear-out cells and applying error correction computation over this faulty data. The work in [9] is the most recent one to deal with MLC STT-RAM memories issues. However, the [9] strategy is implemented at memory-cell level and thereby incurs a significant (hardware and energy) overhead for large-sized cache memories as every MLC cell must require associated hardware for fault protection.

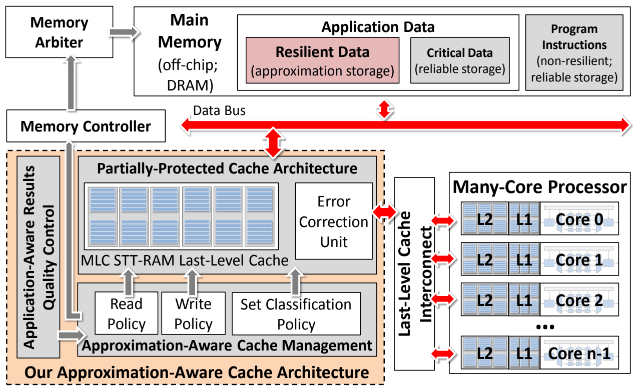
The goal of this paper is to achieve energy-efficient reliability optimization in MLC STT-RAM based caches through *selective approximations of the storage data* depending upon the applications' resilience level and user-provided error tolerance level. The concept of selective data approximation allows for sub-optimal results that facilitates simplification of the error-protection hardware. *The key challenge is to maximize the quality of the applications' results while at the same time minimizing the energy consumption*.

Recently, state-of-the-art have explored data approximations for energy reductions in DRAM-based main memories [2][3][4]. The works [3][4] extend the data refresh interval of DRAM memories to potentially saving energy consumption while assuming wear-out errors. In [2], approximate storage is used in PCM-based main memories by reducing write pulses and leading to wear-out errors. *To the best of authors' knowledge, application of approximate computing in MLC STT-RAM based on-chip memories/caches and exploration of corresponding reliability–energy–quality tradeoffs have yet not been explored*.

Definition – Approximate Storage: In this work, we use the term *approximate storage* for the memory access operations that are not protected against read/write errors in MLC STT-RAMs and may potentially exhibit bit errors.

1.2 Our Novel Contributions and System Overview

This paper addresses the above-discussed issues by introducing a novel design and management of approximation-aware MLC STT-RAM cache architecture. Fig. 2 illustrates the architectural overview of our cache architecture highlighting its key components along with its integration in an on-chip manycore system. We allow the programmer to insert source-code annotations for the identification of resilient and critical data (as and enabled through approximation-aware compilers [5][6]).



Memory Arbiter
Main Memory (off-chip; DRAM)
Application Data
Resilient Data (approximation storage)
Critical Data (reliable storage)
Program Instructions (non-resilient; reliable storage)
Memory Controller
Data Bus
Partially-Protected Cache Architecture
Error Correction Unit
MLC STT-RAM Last-Level Cache
Read Policy
Write Policy
Set Classification Policy
Approximation-Aware Cache Management
Application-Aware Results Quality Control
Our Approximation-Aware Cache Architecture
Last-Level Cache Interconnect
Many-Core Processor
L2
L1
Core 0
Core 1
Core 2
Core n-1

Fig. 2. Overview of our approximation-aware MLC STT-RAM cache architecture and its integration in a manycore.

Our key novel contributions in a nutshell are:

1) *A Partially-Protected MLC STT-RAM Cache Architecture (Section 4.1)* that adopts a set-associative cache organization, where the first *n-1* memory banks store the *n-1* blocks of the cache sets. This part is implemented with 2-bit MLC STT-RAM cell. The n^{th} memory bank is dynamically selected to store either the last block of the cache set or the error correction codes (as part of the error protection system). To guarantee less intrinsic error occurrences, the n^{th} memory bank is implemented by SLC cells. Our cache architecture is equipped with the following three key components.

2) *A Latency-Aware Error Correction Unit (Section 4.2)* that ensures error free execution by protecting memory operations of application's critical data. To avoid performance degradation, our design targets reduced critical delay. It is based on a double-error correction algorithm and it is composed of two parts: (i) the encoding part executed at the time of cache write operations, which generates the error correction codes; and (ii) the decoding part applied at the time of the cache read operations and it is responsible of correcting the cache block data when any fault is detected.

3) *An Approximation-Aware Cache Management Unit (Section 4.3)* that classifies a cache set as 'reliable' (i.e. this set needs to be stored with reliability) or 'unreliable' (i.e. this set does no necessarily require reliability and can tolerate approximation errors). It afterwards, exploits this knowledge to perform approximate storage on the 'unreliable' marked sets and thereby skipping the error correction functionality during the read/write for these sets. To enable this approximation-aware read and write access policies are developed to handle reliable and approximate data storage.

4) *An Application-Aware Quality Control Unit (Section 4.4)* that monitors the objective quality of the application's output to adapt the strength of the approximate storage, thus enabling a variable approximation control. Different applications may exhibit distinct approximate storage responses. Moreover, the same application may demonstrate different output quality profiles depending on the changing inputs. Our scheme adapts the error protection strength to ensure better output quality with an insignificant overhead.

2. BACKGROUND: MLC STT-RAM MEMORY

Spin-Transfer Torque RAM (STT-RAM) cell stores one logic bit in a *magnetic tunneling junction* (MTJ) – an oxide layer between two ferromagnetic layers (see Fig. 3). The resistance value of the MTJ is determined by the relative magnetic field direction between these two layers [25]. One layer has fixed magnetization (called reference layer). The other can have its magnetization changed due to a polarized programming current (called free layer). In a *Single-Level Cell* (SLC) design (as shown in Fig. 3a), 'low resistances' due to parallel magnetization and 'high resistances' due to anti-parallel magnetization represent the logic bits '1' and '0', respectively.

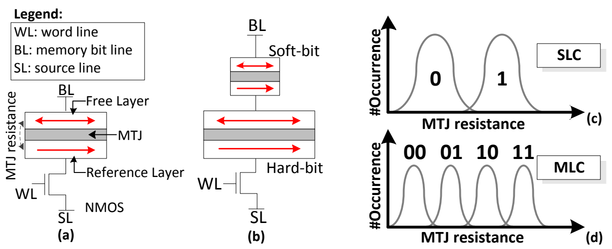


Fig. 3. (a) SLC STT-RAM and (b) MLC STT-RAM cells design; resistance distributions of (c) SLC and (d) MLC cells.

To compose a *Multi-Level Cell* (MLC), the read and write operations must realize four or more resistances states in the same device. In this work we consider the 2-bit MLC STT-RAM cells as proposed by [7][10]; see Fig. 3b. To obtain this, two MTJ with different sizes are vertically stacked along with one NMOS access transistor. The large MTJ stores the "hard-bit", whereas the small MTJ has the "soft-bit". As they are designed to have the same resistance-area product, the small MTJ will lead to higher resistances states than the larger MTJ. The four resistance states (to represent the four logic combinations "00", "01", "10" and "11") are the combination of the magnetization direction of the two MTJ. Fig. 3c and Fig. 3d demonstrate the resistance range of each logic state, comparing SLC and MLC approaches. Due to process variations, tight sense margins in MLC cells between two adjacent resistance states have been observed.

Process variations and thermal fluctuations affect the reliability of MLC STT-RAM memories [9]. When compared to SLC, MLC suffers from relatively more error occurrences. These errors can be classified into two categories: *write errors* and *read errors*. In an MLC cell, 'write errors' happen when a small programming current is applied to flip only the 'soft-bit', but due to process variability the 'hard-bit' accidentally changes at the same time. As this problem did not appear at SLC cells, 'write error' management is a critical issue only when designing MLC-based memories. Furthermore, the 'read errors' are mainly caused by sensing errors, for example, when the resistance state cannot be correctly read. This type of error may also happen when using SLC cells. However, MLC aggravate this problem due to smaller sense margin between adjacent resistances states. As it can be seen in Fig. 3c and Fig. 3d, the resistance states that correspond to "0" and "1" in SLC cells are further partitioned in the MLC cells, to store the second logic bit. Due to variability, there are resistance values where an erroneous logic state may be interpreted, leading to a sensing error. *The combination of write and read errors for an MLC STT-RAM cell may jointly lead to errors probabilities of 10^{-2} to 10^{-4}* [9]. Thus, it becomes infeasible to protect every memory position, especially in case of large on-chip caches where the protection overhead would be non-negligible.

3. CASE STUDY: RESILIENCE EVALUATION FOR A VIDEO ENCODING APPLICATION

Several applications from image/video/vision processing and recognition/data mining domains contain *resilient kernels* (i.e. compute-intensive functional blocks) [1] that may tolerate data errors without violating their core functionality. Such errors in resilient kernels typically do not lead to critical failures during their execution. However, the quality of the final results may be deteriorated. In this work, we denote such a data set for a resilient kernel as *resilient data*. In the following, we perform a case study to evaluate the resilience nature of different kernels of an important advanced video encoding application called the High-Efficiency Video Coding (HEVC). Video encoders have widely proliferated in various application domains, for instance, security, automotive, consumer, and internet streaming (predicted to cover up to 70%-80% of the internet traffic by 2017) [35].

The HEVC encoder exhibits various memory-intensive functional blocks (leading to high energy consumption) with variable resilience characteristics, thus making it a well-suited benchmark application for our approximation-aware cache architecture. Furthermore, HEVC inherently supports parallelism, thus enabling parallel processing on multiple cores and creating simultaneous cache access patterns by different cores.

3.1 HEVC Encoder Preliminaries

The High-Efficiency Video Coding (HEVC) [22] has been developed to provide 2x better compression compared to H.264/AVC. It employs *rate-distortion optimization* (RDO) in order to maximize the compression rates with while minimizing the video objective quality drops (distortion) [28]. The RDO heuristics explore various coding configurations for each block in a video frame. Therefore, HEVC inherently supports a quality-energy design space.

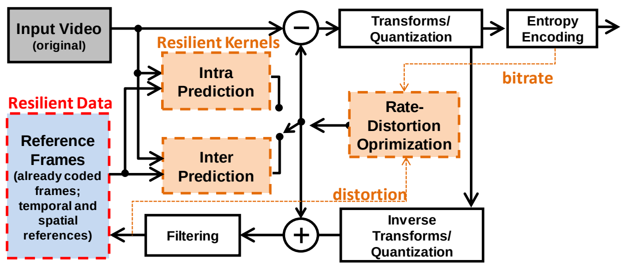


Fig. 4. Block diagram of an HEVC encoder illustrating the resilient kernels and data; white blocks are non-resilient.

Each block in a video frame video can be encoded using the content information from already coded frames (inter prediction), or using the data from the same frame (intra prediction). These already-coded frames are called *reference frames* and require a significant amount of on-chip memory to realize fast energy-efficient video coding architectures, especially in case of bigger resolutions like full-HD (1920x1080 pixels), 2K (2560x1600), and 4K (3840x2160) videos [33]. In case of inter-prediction, each block in the 'current frame' is searched in the 'reference frames' through a very compute- and memory-intensive operation called *motion search*. To achieve high compression rates HEVC tries to match the block structure according to the object shape through a recursive partitioning process which exponentially increases the motion search complexity as it has to be performed for every block partitioning. Therefore, to achieve high performance, HEVC supports a light-weight data-parallelism support by dividing the video frame into so-called *Video Tiles* (see Fig. 5), which are rectangular regions in a video frame that can be encoded independently (i.e. without any spatial dependencies) in parallel on different cores in a manycore system [28].

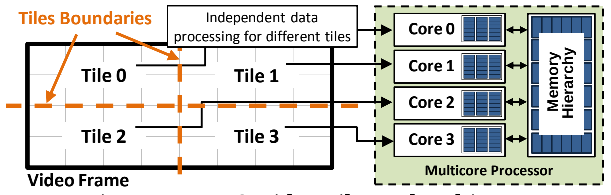


Fig. 5. HEVC Video Tiles and multicore.

3.2 HEVC Memory Access Analysis

The memory operations required to access the reference frames' data from the main memory represent the main memory bottleneck in HEVC (as depicted in Fig. 6), which is aggravated in case multiple cores are accessing the same memory simultaneously. This is concentrated in the inter prediction process, specifically during the motion search and can vary depending on the adopted motion search algorithm and the texture/motion properties of the input video content. *Exhaustive search* (for the best possible coding efficiency) and *fast search* (for higher performance with near-optimal coding efficiency) algorithms can be used depending upon the system constraints. In both scenarios, the reference frames transmission remains the main memory bottleneck in HEVC encoders.

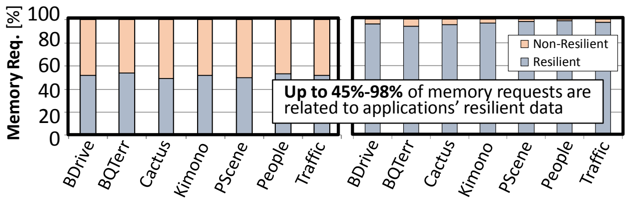


Fig. 6 Memory access profiling of the HEVC encoder.

3.3 HEVC Resilience Analysis

Though reference frames are the most intensively accessed memory data, it is also primarily used by the most resilient functional blocks of the HEVC encoders. This means that errors can be tolerated during the reference frames accesses, leading only to drops in encoding efficiency (application's results in terms of its core functionality of compression efficiency) and may not even affect the output correctness (user-visible output). At the same time, to ensure error free execution, the data required in other functional blocks (like transforms, quantization, entropy encoding and filtering operations – see Fig. 4) cannot tolerate faults (thus denoted as the *critical data*) and must be stored/processed with high reliability.

Insight-1: The existence of critical and resilient data is an important characteristic of HEVC encoders, which is also common in different other optimization-based applications, like image and audio processing, graphical processing and classification algorithms [1][5][6]. In such applications, the memory infrastructure must ensure the correctness for the critical data. For resilient data, however, data approximations can be employed to save energy consumptions while tolerating errors bounded under the user-provided tolerance levels. Therefore, *a memory infrastructure based on approximation storage for the resilient data must, at the same time, facilitate high reliability levels for the critical data so that it can be deployed in a real-world scenario where cases of opportunities for variable approximations and requirements of correct data exist simultaneously*.

Fig. 7a depicts the quality drop in the resilient data in HEVC encoder applications in the presence of approximation errors (summary of numerous data approximation experiments). In this evaluation, the HM HEVC encoder [21] was used. The same trend was observed in the fast x265 encoding application [31]. Detailed description of our experimental setup is given in Section 5.

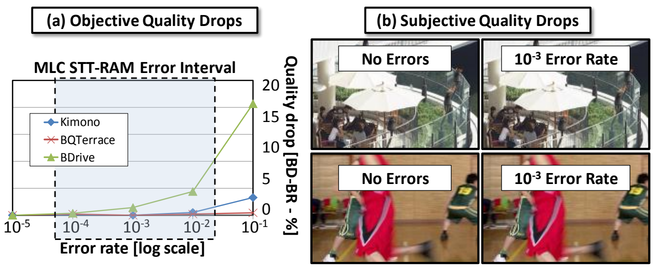


Fig. 7 Error injection evaluation over resilient data in HEVC encoders.

Insight-2: It can be noted that, for the typical error probability of unprotected MLC STT-RAM cells (10^{-4} to 10^{-2}), the data approximation errors for the resilient kernel affected the quality of the application results by 0.5%, on average. Fig. 7b presents example encoded frames of two video sequences: (1) the *BQTerrace* sequence representing the case of minimal quality drop of our analysis (0.2%); and (2) the *BasketballDrive* sequence, which leads to the highest quality loss (2.3% on average). Note, even for the worst case, the output quality loss is not significant. Therefore, an insight from this analysis is that *it is possible to save energy by not protecting the memory operations related to the resilient data, with insignificant penalties on the application's results*.

Insight-3: The potential and impact of approximate storage exploitation can vary depending on the applications' specific characteristics. Even for the same application, when processing different inputs, a varied behavior in output quality degradation may be noticed. Hence, it is important to vary the approximation strength to facilitate run-time varying demands for distinct approximation strengths. We denote this as *variable approximation control*. Considering the knowledge of applications' resilience, user has to specify a error tolerance level, as adopted by the approximate computing community [1][2][3][4]. In summary, *there is a need for dynamic adaptation of the approximation strength for different data storages due to diverse resilience properties of different applications and their varying nature for different input sets*.

4. OUR APPROXIMATION-AWARE MLC STT-RAM CACHE ARCHITECTURE

Our approximation-aware MLC STT-RAM cache architecture adapts the error protection for different regions of the cache according with its data resilience properties. Once approximate storage can be applied, our architecture dynamically skips error protection routines for those cache sectors. As we motivated in Section 4, controlled approximate storage of resilient data does not necessarily affect the quality of applications' results.

In this section, we first present the architecture overview of our partially-protected cache architecture (Section 4.1) followed by the latency-aware error correction unit (Section 4.2). Furthermore, the approximation-aware cache management unit is proposed for energy-efficient read- and write- operations (Section 4.3). An application-aware output quality control unit (Section 4.4) is presented to adapt the approximate storage strength to improve the output quality.

Example Parameters for the Ease-of-Concept Explanation: Without the loss in the generality, for the ease of explanation of our novel concepts with the help of example figures, we assume a cache organization with the following parameters: 8-way set associative cache; processor word size of 64 bits; cache block size of 64 bytes (512 bits); main memory address bus of 29 bits (which addresses 4 GB of data); last-level cache capacity of 64 MB (compliant with Fig. 1 cache size trend for recent manycore processors); 2^{17} rows at each memory bank, supporting equally 2^{17} cache sets. Still, our strategies can be applied for any cache architectural configuration.

4.1 Partially-Protected Cache Architecture

As depicted in Fig. 8, our partially-protected cache architecture is based on a set-associative cache organization.

According to our assumptions, we have 8 memory banks (from Bank-0 to Bank-7) to store all blocks of all cache sets (way-0 to way-7). We use a memory array composed of 2-bit MLC STT-RAM to implement all fields of the cache line: valid bit, tag and data arrays. As explained in Section 2, MLC cells are more susceptible to errors during read or write operations, compared to traditional SLC ones. To ensure reliability for critical data, we designate the last memory bank to store error correction codes (ECCs) for error protection. This specific bank (Bank-7) is implemented as SLC cells (highlighted in red in Fig. 8), to provide better intrinsic reliability. In this special memory bank, the ECCs of each data of the seven other cache blocks are stored. Thus, at every read and write operation, these ECCs must be accessed to serve as input for the error correction unit. We realize approximate storage by avoiding error protection of applications' resilient data. In this case, the memory position of Bank-7 is enabled for data storage, dynamically increasing the associativity of these cache sets (from 7 to 8).

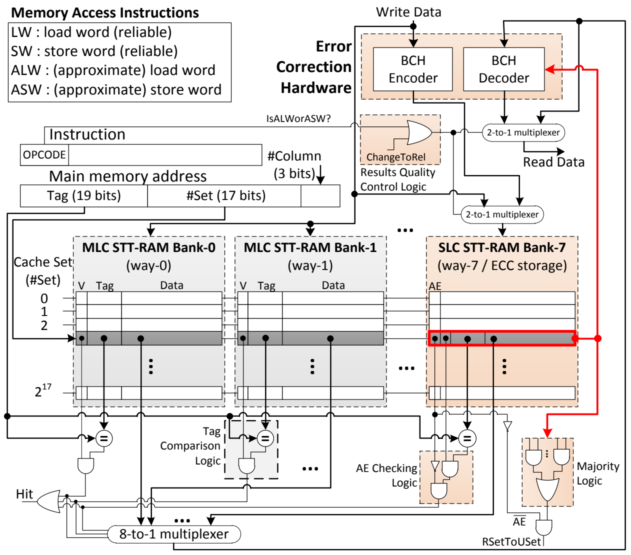


Fig. 8. Our partially-protected cache architecture.

At run-time, each set of the cache is classified according to its reliability support. A set is classified as *reliable set* (*RSet*) when its cache blocks must be protected (critical data). Otherwise, a set is classified as *unreliable set* (*USet*). In this case, no protection is ensured and the error correction unit is bypassed. In both cases, the first seven memory banks remain storing application data as common cache blocks. The Bank-7 cache blocks have different bit-organization depending on the set classification, as depicted in Fig. 9. First, we have an extra flag (called *approximation enable bit - AE*) which will determine if the whole set must be interpreted as *RSet* (*AE=0*) or *USet* (*AE=1*). If the set is classified as *RSet*, the corresponding row of Bank-7 stores ECCs for error protection of the data inside the other 7 cache blocks of this set. In our example, the ECCs have 64 bits to protect one cache block (512 bits). A simple logic is inserted for *AE* checking, where the data read from Bank-7 is skipped for the tag comparison when an *RSet* is accessed (*AE Checking Logic* in Fig. 8). In the other case, in a *USet* configuration, the Bank-7 serves as the way-7 cache block of the set, which contains: valid bit, tag (19 bits) and data columns (8 columns of 64 bits). When a set is classified as a *USet*, we improve the average associativity of the cache. However, data stored in a *USet* is not protected against read and write errors.

Additionally, our partially-protected cache architecture interacts with the management units using some control signals. The *RSetToUSet* flag indicates when a given cache set has more resilient data than critical data. This is required for dynamic reclassification of the sets (as properly descripted in Section 4.3). This signal is generated by the *Majority Logic*, as shown in Fig. 8. The signal *ChangeToRel* is an input that comes from the application-aware output quality control. This flag signalizes that an approximate storage must be converted into a reliable one, triggering the error correction unit to protect the memory operation. Details regarding this adaptive control are given in Section 4.4.

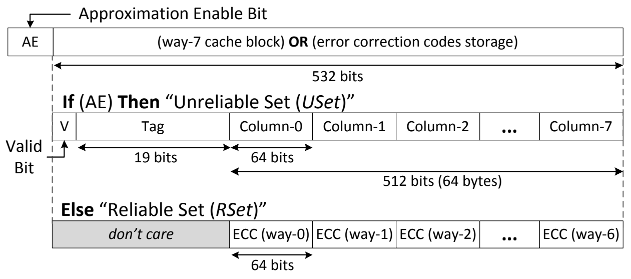


Fig. 9. Bit-level organization of cache line for reliable and unreliable sets.

4.2 Latency-Aware Error Correction Unit

Simple-error correction (SEC) schemes, like SEC Hamming, are not capable to support the error rates of MLC STT-RAM cells [9][30]. Thus, double-error correction (DEC) algorithms are required to ensure the protection for the applications' critical data. We select the DEC *Bose-Chandhuri-Hocquenghem* (BCH) algorithm [32] to be integrated with our partially-protected cache architecture. BCH codes comprehend a class of powerful error-correction cyclic algorithms and are typically employed in communication systems, but not being used for memory-based applications due to their data redundancy and latency overhead for the BCH decoder part. However, recent works, like [30], demonstrated several simplifications for binary BCH codes that make it practical for memory-based systems. In this case, most of the arithmetic operators can be reduced to 2-bit XOR logic gates. It allows for a fully combinational design and thereby reducing the intrinsic latency of cyclic-based coding algorithms.

The BCH encoder generates the ECC from the input data. This part does not represent any bottleneck, since it can be simply implemented with a binary tree of XOR gates. The latency related problems are in the BCH decoder since it needs to re-generate the ECC from the input data and check this ECC with the old one already stored in the Bank-7. Fig. 10 depicts the architecture for the BCH decoder part. As can be noted, a BCH decoder contains one instance of a BCH encoder. After the re-computation of the ECC, it is compared with the old ECC (*Comparator* module), composing an error vector that will be matched with predefined error patterns (*Error Locator* module). As a result, a bit-vector representing the location of the errors is passed to the *Error Corrector* module, which will flip the faulty bits to obtain the corrected data.

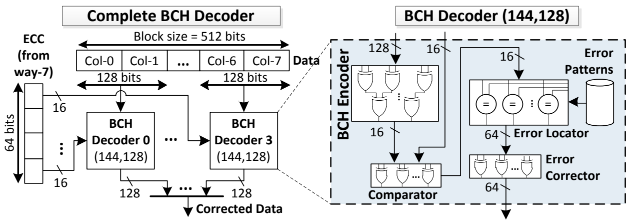
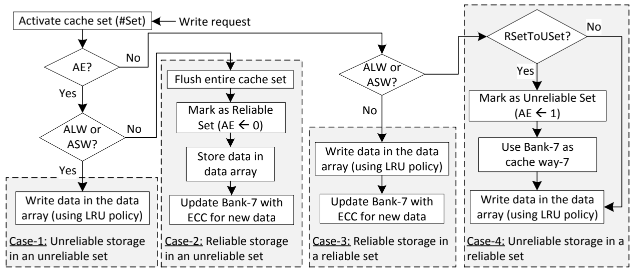


Fig. 10 BCH-based error correction hardware design

A binary logic *(n, k)* input block is a *k*-bit subspace of a binary *n*-bit input. Thus, an *n*-bit codeword contains *k*-bit of data and *r* (where *r* = *k* – *n*) bits for ECCs [30][32]. In our case, we have to protect the cache blocks (512 bits) of each set. The ECCs, in turn, must be stored in the Bank-7 (532 bits). In an *RSet*, we need to store ECCs for seven cache blocks, leading to a maximum of 76 bits for each ECC. The larger is the input data of a BCH encoder/decoder, the longer is the critical path due to deeper XOR trees to compute all bits. Since error protection must be applied in memory operation of all critical data, longer critical paths for BCH encoder/decoder may compromise the overall memory access latency. The best optimization point is the one that generates the larger ECC closer to the maximum allowed (76 bits) and, at the same time, reduces the critical path by splitting the error correction hardware into multiple BCH encoders/decoders with shorter critical paths. More BCH modules used lead to high leakage power. However, compared to the area occupied by the memory data array of typical last-level caches, the leakage of the complete error correction unit is insignificant (as demonstrated in Section 6.1). By exploring the design space of BCH algorithm for ECC generation, we can find the following solution as an efficient one: the block size division into 128-bit words, leading to 4 parallel BCH encoders/decoders, generating 16-bit ECCs, each one (as depicted in Fig. 10). Thus, the ECC for one cache block is 16x4=64 bits, and the ECC for the entire *RSet* is 64x7=448 bits.

4.3 Approximation-Aware Cache Management Unit

Fig. 11 depicts the proposed write policy, integrated in the approximation-aware cache management unit of our partially-protected cache architecture.



Activate cache set (#Set)
Write request
AE?
No
Yes
ALW or ASW?
No
Yes
Write data in the data array (using LRU policy)
Case-1: Unreliable storage in an unreliable set
Flush entire cache set
Mark as Reliable Set (AE ← 0)
Store data in data array
Update Bank-7 with ECC for new data
Case-2: Reliable storage in an unreliable set
ALW or ASW?
No
Write data in the data array (using LRU policy)
Update Bank-7 with ECC for new data
Case-3: Reliable storage in a reliable set
RSetToUSet?
No
Yes
Mark as Unreliable Set (AE ← 1)
Use Bank-7 as cache way-7
Write data in the data array (using LRU policy)
Case-4: Unreliable storage in a reliable set

Fig. 11. Write access management

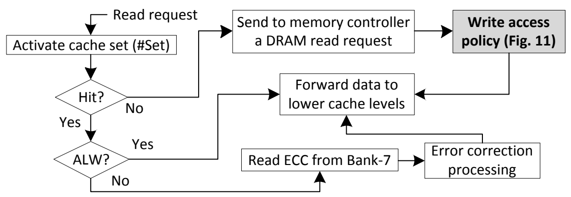
As the first step, the data of the entire set is activated and read (for hit/miss evaluation). Considering the *AE* flag (set classification) and the type of memory instruction (reliable or unreliable storage), four cases can be encountered:

Case-1: *Unreliable storage of resilient data (ALW or ASW) in a USet (AE=1)*. In this case, no protection is applied and the data is simply written in the cache, using the least-recently used (LRU) replacement policy.

Case-2: *Reliable storage of critical data (LW or SW) in a USet (AE=1)*. Reliable storage is not suitable to be performed in a *USet*, since Bank-7 is used as an extra cache block. Thus, our approximation management unit takes the following steps: (1) the entire cache set is flushed and (2) reclassified as an *RSet* (*AE* is assigned to 0). Bank-7 memory row is now responsible for storing the ECCs for error protection. Furthermore, the data is written in the cache, the ECCs are generated by the BCH encoder, and stored in Bank-7 (to ensure future reliable operations for the data).

Case-3: *Reliable storage of critical data (LW or SW) in an RSet (AE=0)*. The data is simply written in the cache, the BCH encoder computes the ECCs, to be stored in Bank-7.

Case-4: *Unreliable storage of resilient data (ALW or ASW) in an RSet (AE=0).* As an *RSet* can support unreliable storages, our management does not immediately reclassify the set as a *USet*. The reclassification will only occur when the unreliable storages in a set surpassing the reliable storages. To capture this condition, we analyze the signal *RSetToUSet*, which is the result of the Majority Logic (see Fig. 8). There is no need of ECC generation, since unreliable storages skip error protection. When the approximate memory access does not lead to a reclassification to a *USet*, a special ECC is generated indicating that the given cache block of that set corresponds to resilient data. This special ECC signalizes that the error correction unit can be bypassed, even when the read set is classified as an *RSet*.



Read request
Activate cache set (#Set)
Send to memory controller a DRAM read request
Write access policy (Fig. 11)
Hit?
No
Yes
Forward data to lower cache levels
ALW?
Yes
No
Read ECC from Bank-7
Error correction processing

Fig. 12. Read access management.

Fig. 12 presents our read access management policy for our partially-protected cache architecture. When a read request occurs, the corresponding set is activated and the corresponding rows of all memory banks are read. If the read access leads to a miss (the tag checking fails), a request is sent to the memory controller in order to fetch the required data from the main memory. Once the data is read from the main memory, it must be written to the cache. Our policy follows the already described write policy to properly store the incoming data. Then, the data is forward to the next cache levels. If a hit is verified, two cases need to be considered: (1) if it is an approximate memory instruction, the accessed data is simply forward to the next cache levels; otherwise, (2) if the memory access instruction requires a reliable access, then an extra step of error correction processing is required. This step is performed by the BCH decoder module, as already explained in Section 4.2.

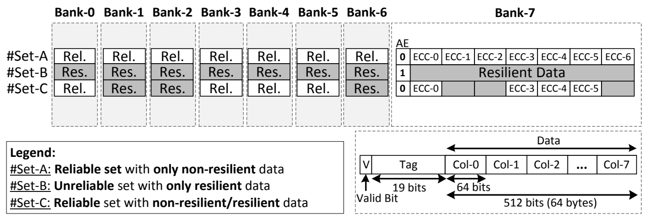


Fig. 13. An example of different set configurations and allocation for resilient and critical data

An Example: Fig. 13 depicts a data allocation example. We have three highlighted cache sets (*#Set-A*, *#Set-B* and *#Set-C*). The *#Set-A* and *#Set-C* are classified as *RSet* (*AE=0*), and the *#Set-B* is assigned as a *USet*. We can note two examples of *RSet* allocations: in the case of *#Set-A*, all cache blocks are storing reliable data, and Bank-7 is filled with the ECCs for the seven cache blocks of the set. In case of *#Set-C*, the approximate and reliable data are jointly stored in cache blocks of the same set. As already explained, the set is only reclassified to *USet* if the number of unreliable storages surpasses the number of reliable ones (inside the target set). In this case, as we skip error protection routines for the approximate data, and the ECC fields in Bank-7 corresponding to the resilient data are set to a special ECC, identifying this fact. The *#Set-B* is an example of a *USet*, where Bank-7 is used as way-7 block of the cache. The *USets* are only suitable for unreliable storages, since we must have to ensure error free execution by increasing the reliability when managing application's critical data. In this case, the average associativity of the cache is increased, improving the number of hits and, consequently, reducing main memory communication.

To give a behavioral idea of the partial protection proposed in our approximation-aware MLC STT-RAM cache architecture, Fig. 14 depicts the system properties when a resilient kernel of an application starts running. As premise of our system, we must ensure high reliability when storing critical data. In this case, our partially-protected cache organization sacrifices the last memory bank to store ECCs to ensure the data reliability. As a result, the percentage of *USets* is near to 0% and the average associativity is 7. Further, the latency-aware error correction unit is used for all read and write memory operations. When a resilient kernel starts processing, bursts of unreliable storages (ALW and ASW memory instructions) will fill the cache sets with resilient data. Hence, the existing *RSets* are gradually reclassified as *USets*. Following the same trend, the overall cache associativity will increase from 7 to 8, as ECCs can be discarded due to bursts of unreliable storage. As the resilient kernel finishes, our cache architecture adaptively reclassifies the groups to *RSets*, activating the error correction hardware to protect the MLC cells against errors. This adaptive run-time reclassification is crucial to achieve energy-efficient error protection, while ensuring reliability for the critical data.

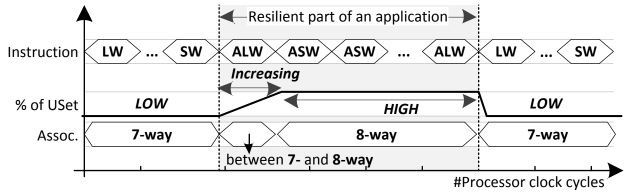


Fig. 14. Timing diagram of our approximation-aware partially-protected cache architecture.

4.4 Application-Aware Output quality Control

While saving energy is a crucial issue, the quality of application's output results must also be taken into account. Thus, our cache architecture also employs an application-aware output quality control scheme. At design-time determined intervals, the quality level of application's output is forwarded by the application to the quality control unit. As different applications have different output characteristics, the application user/developer defines an objective quality measure for the application's results. Under user-defined constraints, our architecture can adapt the error correction coverage. The strategy is to store a brief history of past registered objective quality levels. When a descendent trend is observed, the quality control unit acts by interpreting unreliable storages to reliable ones, thus trading off energy with the improved quality level. Otherwise, when the achieved quality level is above the quality constraint, the protection for more unreliable storages can be activated.

The block diagram of our quality control unit is depicted in Fig. 15. The signal *ChangeToRel* is used as the interface between the output quality control and the cache architecture (see Fig. 8). By setting this signal to '1', an unreliable storage will be protected. We define a TH_{App} parameter, as the probability of protecting (becoming reliable) for an unreliable storage. Thus, at each application update with a new quality measurement, two signals can be generated: (1) *IncTH*, indicating that a decreasing trend is observed and more protection should be applied; and (2) *DecTH*, which alleviates the protection by capturing the increased quality case. As initial value, TH_{App} is set to zero. We update the TH_{App} with increments (if *IncTH=1*) or decrements (if *DecTH=1*) in steps of 0.05.

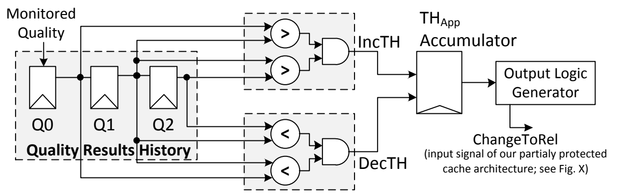


Fig. 15. Application-aware quality control unit.

Hardware Overhead: This application-aware quality control unit requires a very simple implementation: three registers to store the brief history of objective qualities (*Q0*, *Q1* and *Q2* in Fig. 15), four subtractors to perform the comparisons, and two AND-2 logic gates. Furthermore, an accumulator stores the update value of TH_{App} parameter. Finally, simple output logic, to translate the TH_{App} to the signal *ChangeToRel*, is required.

5. EXPERIMENTAL METHODOLOGY / SETUP

Two different HEVC encoders were used as benchmark: (1) the HEVC *HM* implementation [21] by the JCT-VC group; and (2) the *x265* open-source application [31]. The resilient kernels in HEVC encoder are greatly affected by the motion search technique used during the inter-prediction step. In our experiments, we evaluated two different algorithms: *exhaustive search* and *fast search* (that adopts an heuristic evaluation choices to avoid local minima). The Full Search algorithm was used as the *exhaustive search* in both *HM* and *x265*. For the *fast search*, *HM* uses the *TZ Search* algorithm, whereas *x265* implements the *hexagonal search* algorithm. As input test sequences, six different test videos (recommended by the standardization committee) in two different high definition resolutions were considered: *BasketballDrive (BDrive)*, *BQTerrace (BQTerr)*, *Cactus* and *Kimono* (full-HD: 1920x1080 pixels); *Traffic* and *PeopleOnStreet* (*People*) (2K: 2560x1600) [16]. For results objective quality evaluation of the selected video encoding applications, we considered the *Bjontegaard Delta* metric (BD-BR) [26] as recommended by the standardization committee and widely adopted in the video coding community. The results quality drop evaluation is done through error injection according to the MLC STT-RAM error model. Note that the wear-out errors evaluation is outside the scope of this work.

Table I Hardware-Related Parameters Adopted in Our Experiments

Manycore Processor Parameter		
Instruction Set Architecture	x86	
Number of Cores	16	
Cache Parameters		
	L1 Cache (Private)	**Last-Level Cache (Shared)**
Technology	SRAM	2-bit MLC STT-RAM
Design	4-way	8-way (ours)
Size	32 KB	64 MB
Read Latency (ns)	1.425	2.263
Write Latency (ns)	1.425	15.095
Leakage Power. (mW)	1.137	808.423
Read En. (pJ/acc)	48	1.497
Write En. (pJ/acc)	48	878
Main Memory Parameters		
Model	LPDDR2 - MT42L128M16D1GU-25WT	
Access Latency	300 ns	
Energy Components	Refresh and Standby (fixed); Page Activate, Read, Write and I/O Pins (variable)	

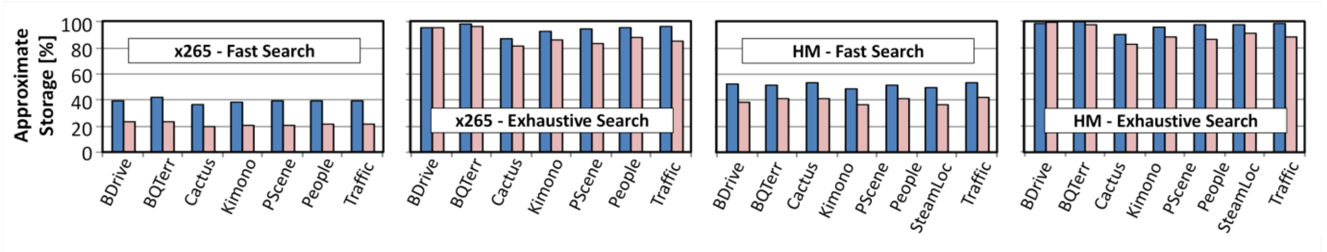


Fig. 16 Approximate-storage characterization for the used HEVC encoding applications (HM and x265).

For our experiments, we use the *callgrind* and *cachegrind* tools of *valgrind* simulator [29] to extract the memory access traces for each cache memory level and for the main memory. Table I illustrate the parameters used in our experiments for L1 and last-level caches, such that, the selection of parameter values follow the prominent state-of-the-art works [11][12][14][15] and commercially available manycore processors. The cache latencies and energy parameters of Table I were extracted using CACTI (for SRAM) [27] and NVSim (for STT-RAM) [34]. Works in MLC STT-RAM cells physical exploitation were considered to derive the values for MLC design [7][10]. For the main memory, we consider one 4-Gbit Low-Power DDR2 module [17]. The energy components of a LPDDR2 were estimated using the main memory accesses of each application and the technology data from Micron [18][19]. All control-flow hardware blocks, including the error correction module based on BCH algorithm was synthesized using Cadence synthesis flow using ST 65nm standard-cells library.

5.1 Resilience Characterization for HM and x265

Before moving forward to the experimental results discussion, it is important to characterize the used benchmark applications in terms of resilience of their memory accesses operations. The characterization was performed for HM and x265 applications, when encoding videos with fast and exhaustive motion search engines. Fig. 16 depicts the percentage of approximate memory access, when writing and reading resilient data. On average for our case-study applications, approximate storage operations represents 31%, 92%, 46% and 94%, for x265 and HM using fast and exhaustive motion search engines, respectively.

6. RESULTS AND DISCUSSIONS

For evaluation purpose, we define an alternate implementation that avoids approximation storage by assuming every cache memory access as a reliable operation. Thus, the latency and energy overhead of ensuring error protection is always observed. The goal is to evaluate the energy efficiency of our partially-protected cache architecture against fully-protected memory designs, like [11]-[15].

6.1 Energy Efficiency Analysis

Table 2 presents the synthesis results of the error correction module, in terms of delay and power consumption for BCH encoder and decoder modules. As explained in Section 4.2, the most complex part is the BCH decoder due to the internal BCH encoder and comparison logic. Therefore, the critical delay and the dynamic energy of the BCH encoder directly affect the write operations of our approximation-aware cache architecture. The BCH encoder, otherwise, increases the latency and dynamic energy of read operations. Due to our partially-protected approach, these overhead can be reduced when performing approximations for the resilient data through error skipping.

Table 2 Error Correction Module Synthesis

	BCH Encoder	BCH Decoder	Error Correction
Technology	ST 65nm		
Delay (ns)	0.47	1.49	-
Leakage Power (mW)	0.045	1.129	1.174
Dynamic Power (mW)	2.164	47.67	-
Dynamic Energy (pJ/acc)	5.085	350.375	-

To demonstrate the energy efficiency of the proposed approximation-aware cache architecture based on MLC STT-RAM memory, we extract the energy consumption of each part, separately: *on-chip energy* (leakage and dynamic parts), *off-chip energy* (due to main memory accesses), and the *overall energy*. Table 3 summarizes the energy results for the benchmark applications averaged over all test video sequences when executing *fast search* algorithm.

Table 3 Energy-Efficiency Analysis of our Partially-Protected MLC STT-RAM Cache Architecture

	HM Encoder	x265 Encoder
On-Chip Leakage Energy (mJ)		
Last-Level Cache	808.42	
Error Correction Unit	1.174	
Total	809.60	
On-Chip Dynamic Energy (mJ)		
Fully-Protected	15.01	1.06
Ours – Partially-Protected	12.02	0.98
Savings	20%	8%
Off-Chip Energy (mJ)		
Fully-Protected	1,429.44	183.77
Ours – Partially-Protected	1,000.61	113.94
Savings	30%	38%
Overall Energy (mJ)		
Fully-Protected	2,254.04	994.43
Ours – Partially-Protected	2,093.81	974.13
Overall Savings	19%	7%

In terms of on-chip leakage energy, we can note an insignificant increase (+0.15%) when the error correction module is inserted to compose our partially-protected cache architecture. Although with core local L1 cache memory, the total L1 leakage energy is significantly smaller compared to the last-level cache (representing less than 1%). Fig. 17 depicts the leakage trend overhead for varied cache memory sizes (4MB-64MB): 0.15%-1.3%, considering the average of used applications.

Our partially-protected cache architecture can save dynamic energy by 8%-20% through avoiding error protection of memory operations for the resilient data, compared the fully-protected cache. Fig. 18 presents a detailed analysis of on-chip dynamic energy consumption of our cache architecture for all analyzed video test sequences: on average, our architecture achieves 21% and 30% dynamic energy savings for the tested applications when running *fast search* and *exhaustive search*, respectively.

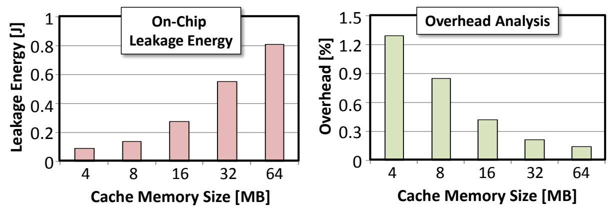


Fig. 17 Leakage energy analysis.

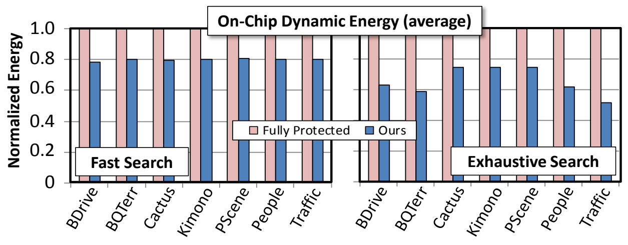


Fig. 18 On-chip energy efficiency analysis for (a) dynamic energy; and (b) leakage energy.

In the off-chip energy part, our architecture improves the energy efficiency by enabling cache sets with an extra block, compared with fully-protected cache. This leads to less last-level cache misses, incurring in the reduced off-chip energy consumption for main memory fetching and storage. In our experiments, our partially-protected cache architecture can reach savings of 30%-38% compared to the fully-protected architecture. Note, the main memory access operations are significantly more costly (in terms of latency and energy consumption) compared to the on-chip cache access operations. Thus, our approximation-aware cache architecture takes care of this by *leveraging resilient kernels of the applications to increase the cache sets associativity*, assuming approximation storage in these sets. At the same time, we ensure error free execution by protecting the reliable memory operations. We discuss these aspects further in Section 6.3.

On average, our approximation-aware cache architecture together with the external memory provides energy savings of 7% and 19% compared to full-protected cache, for *HM* and *x265*, respectively.

6.2 Overhead Analysis: Access Latency

Besides the demonstrated energy efficiency of our architecture, we also estimate the overhead of equipping the last-level cache with the proposed novel hardware modules. Fig. 19 depicts the latency and dynamic energy overhead of implementing extra hardware for error protection for reliable memory access operations.

When the approximate storage is exploited, our approximation-aware read and write policies detect it and skip any error protection procedures. In this case, no overhead is paid for error correction, leading to no overhead in the access latency. For reliable memory operations, error protection is then applied. A reliable write operation incurs the following additional latencies: (1) BCH encoder latency to generate the ECC for the new cache block; (2) write access of MLC STT-RAM array to write the new data; and (3) an extra write access to the SLC STT-RAM *Bank-7* update the ECC of the modified cache set. Still, these accesses can be done in parallel, leading to reduced overhead. When a reliable read is performed, the access latency of our architecture comprehends: read access from MLC STT-RAM array to get the accessed data and the ECC from SLC-based *Bank-7*, and BCH decoder to ensure the corrected data in the cache memory output. The BCH encoder will add only 0.47 ns to the MLC STT-RAM cache write latency, leading to an overhead of just 3.1%. The BCH decoder, in turn, adds 1.49 ns to the cache read latency. In our experiments, the average overhead in terms of the memory access latency incurred by the insertion of our approximation-aware management is 0.01%-0.72% and 0.02%-1.56%, considering *x265* and *HM* benchmarks using *exhausting search* and *fast search*, respectively.

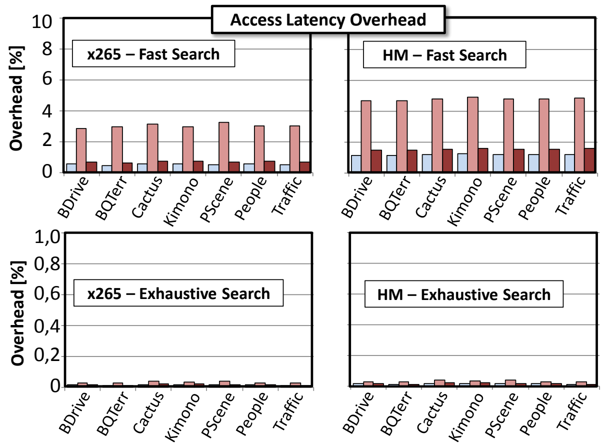


Fig. 19 Memory access latency overhead analysis.

6.3 Quality Analysis of Application's Results

After demonstrating the energy efficiency of our approximation-aware cache architecture due to reliability optimization exploiting resiliency properties, we must also evaluate the quality drops of applications' results. For all tested applications and input video sequences, *the increased reliability of our error protection engine for critical data ensured error-free execution for all experiments.*

Table 4 Quality Analysis of Applications' Results.

Video Sequence	BD-BR Drops [%]	
	x265	HM
BasketballDrive	-0.182	-0.536
BQTerrace	-0.067	-0.451
Cactus	-0.146	-0.482
Kimono	-0.297	-0.471
ParkScene	-0.177	-0.268
PeopleOnStreet	-0.357	-0.564
AVERAGE	***-0.219***	***-0.426***

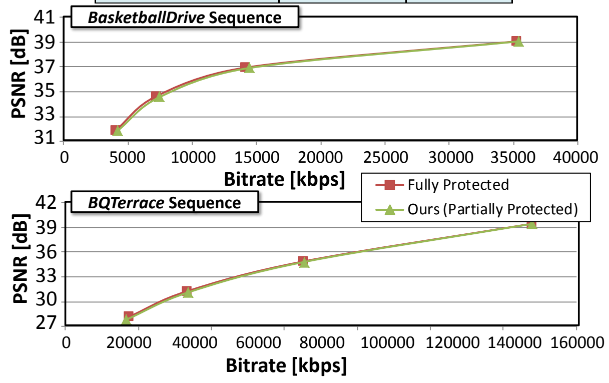


Fig. 20 RD curves for quality analysis (HM application).

Table 4 summarizes the evaluation of the quality loss of our partially-protection architecture. In Fig. 20, a more detailed rate-distortion analysis using two complex test video sequences: *BasketballDrive* and *BQTerrace*. To improve the achieved quality results, we proposed an application-aware quality control unit that enables variable approximation control. The energy consumed by this module is already counted in the energy efficiency analysis, presented in Section 6.1. As already demonstrated, the overhead to implement this adaptive control is negligible. We can note from Table 4 that our variable approximation control minimizes the quality loss that even becomes insignificant when comparing to the best-case scenario: where all approximate read/write operations are protected. Our experiments show that quality drops varies from 0.173% to 0.485%, on average for *x265* and *HM* applications. It can be noted in Fig. 20 where the rate-distortion curves are practically overlapped.

Therefore, we accomplish our goal of maximizing energy efficiency when optimizing the reliability of MLC STT-RAM caches while maximizing the applications' output quality.

7. CONCLUSIONS

This work proposed an approximation-aware cache architecture that leverages MLC STT-RAM density and low-power features to design large-sized caches for advanced manycore processors. To solve MLC STT-RAM reliability issues, our architecture is partially-protected to reduce reliability overhead by leveraging resilience properties of applications. The goal is to increase the energy efficiency while meeting the reliability requirements. Our architecture integrates a latency-aware double error-correction unit to guarantee error protection for applications' critical data, ensuring error free execution. Approximation-aware read- and write policies exploit approximate storage and organize them along the reliable and unreliable cache positions. An adaptive control takes care of the applications' output quality. We performed case studies on two next-generation advanced video encoding applications that exhibit memory-intensive functional blocks with variable resilience properties and parallelism support.

Experimental results over various test videos demonstrated the improved energy efficiency (on average 7%-19%) of our approximate-aware MLC STT-RAM based cache architecture compared to fully-protected caches. At the same time, these gains incur in minimal quality penalties in the output (quality loss from -0.219% to -0.426%). The proposed error protection module ensured complete error-free execution by providing full coverage when processing non-resilient critical application function. Furthermore, the overhead of implementing our approximation-aware management negligibly impacts the energy-efficiency (0.15%-1.3% of on-chip leakage) and the access latency (0.01%-1.56%) of our cache architecture.

8. REFERENCES

[1] V. K. Chippa, S.T. Chakradar, K. Roy, A. Raughunathan. "Analysis and Characterization of Inherent Application Resilience for Approximate Computing", In: Design Automation Conference, pp. 1-9, 2013.

[2] A. Sampson, J. Nelson, K. Strauss, L. Ceze. "Approximate Storage in Solid-State Memories". In: International Symposium on Microarchitecture, pp. 25-35, 2013.

[3] J. Lucas, M. Alvarez-Mesa, M. Andersch, B. Juurlink. "Sparkk: Quality-Scalable Approximate Storage in DRAM", In: The Memory Forum, pp. 1-9, 2014.

[4] S. Liu, K. Pattabiraman, T. Moscibroda, B.Zorn. "Flikker: Saving DRAM Refresh-power through Critical Data Partitioning", In: Conference onArchitectural Support for Programming Languages and Operating Systems, pp. 213-224, 2011.

[5] A. Mishra, R. Barik, S. Paul. "iACT: A Software-Hardware Framework for Understanding the Scope of Approximate Computing", In: Workshop on Approximate Computing Across the System Stack, 2014.

[6] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, D. Grossman. "EnerJ: Approximate Data Types for Safe and General Low-Power Computation", In: Conference on Programming Language Design and Implementation, pp. 164-174, 2011.

[7] X. Bi, M. Mao, D. Wang, H. Li. "Unleashing the Potential of MLC STT-RAM Caches", In: International Conference on Computer-Aided Design, pp. 429-436, 2013.

[8] M. Zhao, L. Jiang, Y. Zhang, C. J. Xue. "SLC-enabled Wear Leveling for MLC PCM Considering Process Variation", In: Design Automation Conference, pp. 1-6, 2014.

[9] W. Wen, et al. "State-Restrict MLC STT-RAM Designs for High-Reliable High-Performance Memory System", In: Design Automation Conference, pp. 1-6, 2014.

[10] Y. Zhang, L. Zhang, W. Wen, G. Sun, Y. Chen. "Multi-level Cell STT-RAM: Is It Realistic or Just a Dream?", In: International Conference on Computer-Aided Design, pp. 1-6, 2014.

[11] Y. Chen, W.-F. wong, H. Li, C.-K. Koh, Y. Zhang, W. Wen. "On-Chip Caches Built on Multilevel Spin-Torque RAM Cells and Its Optimizations", In: Journal on Emerging Technologies in Computing Systems, pp. 16:1-16:22, 2013.

[12] C.-K. Koh, et al. "The Salvage Cache: A fault-tolerant cache architecture for next-generation memory technologies", In: IEEE ICCD, pp. 268-274, 2009.

[13] A. M. H. Monazzah, et al. "FTSPM: A Fault-Tolerant ScratchPad Memory", In: DSN, 2013.

[14] R. Azevedo, eta l. "Zombie Memory: Extending Memory Lifetime by Reviving Dead Blocks", In: ACM/IEEE ISCA, pp. 452-463, 2013.

[15] M. Gottscho, et al. "Power/Capacity Scaling: Energy Savings with Simple Fault-Tolerant Caches", In: ACM/IEEE/SIGDA DAC, pp. 1-6, 2014

[16] F. Bossen, "Common test conditions and software reference configurations", ITU-T/ISO/IEC JCTVC-K1100, October 2012.

[17] Micron. "4Gb: x16, x32 Mobile LPDDR2 SDRAM S4", 168p, 2013.

[18] Micron. "TN-46-03 Calc. DDR Mem. System Power". Rev. B 3/05 EN, 2005.

[19] Micron. "TN-46-12 DRAM Power-Saving Features/Calcs". Rev. B 5/09, 2009.

[20] A. Jadidi, "High-endurance and performance-efficient design of hybrid cache arch. through adaptive line replacement", ACM/IEEE ISLPED, p.79-84, 2011.

[21] JCT-VC. HEVC Software SVN, 2011. Available in: <https://hevc.hhi.fraunhofer.de/ >

[22] JCT-VC, "High Efficiency Video Coding (HEVC) text specification draft 10 (for FDIS & Consent)", Doc.: JCTVC-L1003_v9, 2013.

[23] Y.-T. Chen et al., "Dyn. reconfigurable hybrid cache: An energy-efficient last-level cache design", IEEE/ACM DATE, vol., no., pp.45,50, 12-16, 2012.

[24] K. Abe et al, "Novel hybrid DRAM/MRAM design for reducing power of high performance mobile CPU," IEDM, pp.10-13, 2012.

[25] X. Dong et al., "Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement," ACM/EDA/IEEE DAC, pp.554,559, 2008.

[26] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves", VCEG Contribution VCEG-M33, Austin, April 2001.

[27] S. Thoziyoor et al, "CACTI 5.1 tech. report," HP Labs, 2008.

[28] B. Pourazad et al., "HEVC: The New Gold Standard for Video Compression: How Does HEVC Compare with H.264/AVC?," IEEE CeM, pp. 36-46, 2012.

[29] Valgrind^{TM} Developers. Valgrind Home. Available in: <http://valgrind.org/>

[30] R. Naseer, J. Draper. "Parallel Double Error Correcting Code Design to Mitigate Multi-Bit Upsets in SRAMs", In: Solid-State Circuits Conference, p. 222-225, 2008.

[31] x265. "x265 HEVC High Efficiency Video Coding H.265 Encoder". Available in: <http://x265.org/>.

[32] J. Wiley and Sons, "Error Correction Coding: Mathematical Methods and Algorithms", 2005.

[33] I. Richardson, "H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia", Wiley, 2003.

[34] X. Dong, C. Xu, Y. Xieand, N.P. Jouppi. "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory." In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, p. 994-1007, 2012.

[35] CISCO. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018. [S.l.]. 2013.

Hybrid Scratchpad Video Memory Architecture for Energy-Efficient Parallel HEVC

Felipe Sampaio, Bruno Zatt, Muhammad Shafique, Jörg Henkel, and Sergio Bampi

***Abstract*— A hybrid scratchpad video memory (Hy-SVM) for energy-efficient Tiles-parallelized High-Efficiency Video Coding (HEVC) is presented herein. The key ideas of Hy-SVM include: application-specific design and management; combined multiple levels of private and shared memories that jointly exploits intra-Tile and inter-Tiles data reuse; scratchpad memories (SPMs) as on-chip data storage; SRAM and STT-RAM hybrid design. We propose a design methodology for Hy-SVM that leverages application-specific properties to properly define the SPMs parameters. The inter-Tiles data reuse potential of parallel HEVC is exploited by our run-time overlap prediction scheme, which identifies the redundant memory access behavior by analyzing monitored past frames encoding. Based on the predicted overlap characteristics, Hy-SVM integrates memory access management units (MAMUs) to control the access dynamics to the private/shared SPM levels. Furthermore, adaptive access management units (APMUs) can strongly reduce on-chip energy consumption due to the predicted overlap formation. The experimental results demonstrate Hy-SVM overall energy savings of 55%-92% (4-Tile) and 43%-94% (8-Tile) when compared to related works. From the external memory perspective, Hy-SVM can improve data reuse, resulting in 24%-35% of off-chip energy consumption. Additionally, our APMU contributes by reducing on-chip energy consumption of Hy-SVM by 83%, on average. Thus, compared to related works, Hy-SVM presents the lowest on-chip energy consumption. Moreover, the overhead of implementing our management units insignificantly affects the performance- and energy-efficiency of Hy-SVM.**

***Index Terms*—Video Memory, Scratchpad, HEVC, Application-Specific Optimization, Energy Efficiency, Adaptivity.**

I. Introduction

In multimedia processing systems, the video compression (aka. video coding) has a key role, being responsible for reducing the video data representation to enable efficient storage and transmission. The High Efficiency Video Coding (HEVC) is the state-of-the-art standard [1] that provides double compression compared to its predecessor H.264/AVC [2]. However, this comes with a cost of more than 40% computational effort increase when compared to H.264/AVC in the encoding part [3]. The increased complexity of HEVC results from the novel coding data structures and a plethora of new prediction modes, resulting in a wider decision space [4].

Manuscript received February 08th, 2018.

F. Sampaio (fmsampaio@inf.ufrgs.br) and S. Bampi (bampi@inf.ufrgs.br) are with the PPGC, Informatics Institute (II), Federal University of Rio Grande do Sul (UFRGS), Brazil.

M. Shafique (muhammad.shafique@tuwien.ac.at) is with the Institute of Computer Engineering, Vienna University of Technology (TU Wien), Austria.

B. Zatt (zatt@inf.ufpel.edu.br) is with Center of Technology Development (CDTec), Federal University of Pelotas (UFPel), Brazil.

J. Henkel (henkel@kit.edu) is with the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany.

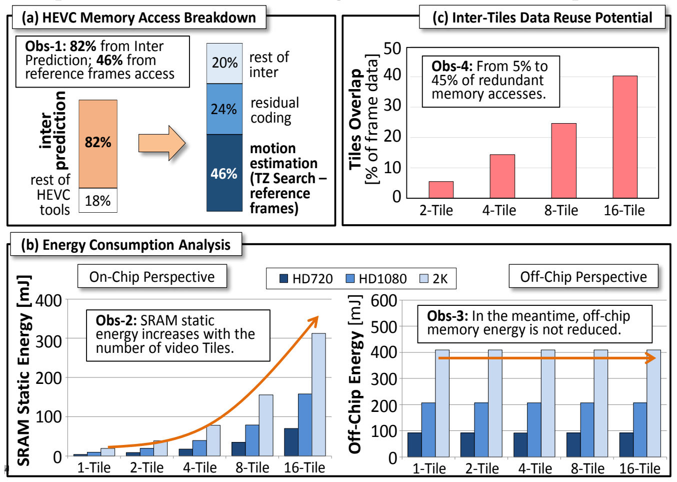


Fig. 1 - Memory requirements analysis for HEVC encoding.

From memory perspective, these new coding features lead to 2-3x more memory communication compared to H.264/AVC [3]. The memory bottleneck in HEVC encoders is related to the access to already processed (and reconstructed) video frames, called reference frames. In this aspect, up to 50% of off-/on-chip memory accesses are required for reference frames reading and writing (Fig. 1a) leading to high memory-related energy consumption during HEVC encoding. To face the increased computational complexity, HEVC defines high-level parallelization strategies, such as *Tiles* partitioning [5], which divides the frame into rectangular regions that can be encoded in parallel. Whereas providing speedup to encoding applications, such tools aggravate the energy consumption of the memory infrastructure (on- and off-chip parts; see Fig. 1b), posing new challenges for multimedia systems. The main Tiles-parallelized HEVC challenge is to efficiently exploit the inter-Tiles data reuse potential, which significantly increases as more parallelism is exploited (Fig. 1c). In this work, we refer to this reference frame region that is redundantly accessed by more than one Tile processing as the *overlap* region.

Thereby, *there is a strong need for energy-efficient memory architectures which are able to exploit the data reuse potential stemming from parallel features of HEVC.*

A. Key Research Challenges and Opportunities

General-purpose memory hierarchies, like [6], [7], have compromised energy efficiency when facing the state-of-the-art HEVC encoding high memory requirements and specialized dynamics [3]. Application-specific video memories have been focus of research works since previous video coding standards, like MPEG-2 [8], [9] and H.264/AVC [10]–[13]. The main goal is to increase the energy efficiency of video memories relying on video coding knowledge. However, these works lack support for parallel video coding, which is more memory restrictive and do not address memory contention in private vs. shared memories for data synchronization. Hence, *the challenge is to leverage application-specific knowledge as opportunity for designing dedicated energy-efficient video memories for parallel video coding*.

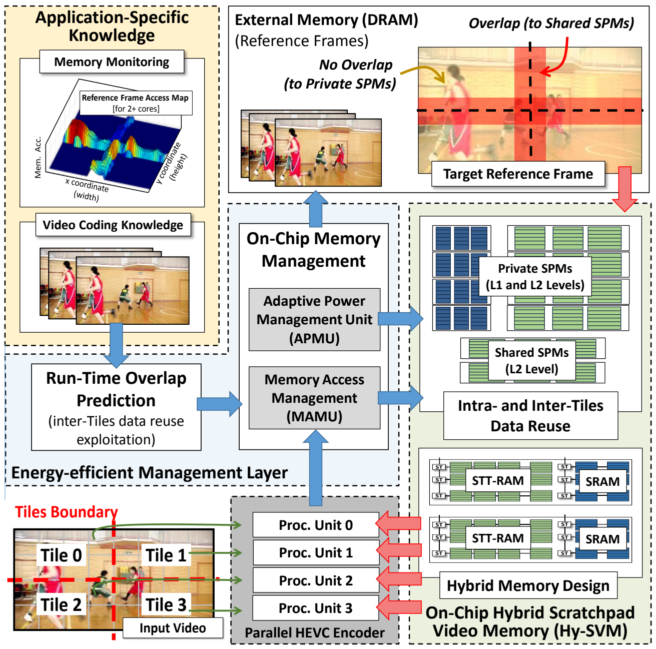
As opportunity for application-specific applications, scratchpad memories (SPMs) overcome/alleviate the overhead of caches. In SPMs, instead of providing hardware support for mapping data/code from off-chip to on-chip memory, the designer and/or the compiler are responsible to perform access management. Due to application-specific knowledge exploitation, SPMs allows energy savings of up to 30% compared to complete cache memories [14]. SPMs are widely available to be used as performance- and energy-efficient on-chip storage option in nowadays processor chips [13][14]. In this work, *we utilize SPMs as opportunity for designing application-specific on-chip video memories, enabling energy savings by exploiting the knowledge from the HEVC encoder.*

In another perspective, hybrid memory design exploiting the industry advances of alternative memory technologies has been research target during the last decade [17]–[20]. In hybrid on-chip memory design, emerging memory technologies are used in combination with traditional Static-RAM (SRAM) cells. The goal is to reduce the impact of SRAM shortcomings, like low density and high static energy consumption. In this context, the Spin-Transfer Torque RAM (STT-RAM) stands out as a promising technology. Recent academy and industry advances serve as a solid foundation to enable STT-RAM to be integrated with CMOS logic circuitry of nowadays general-purpose processors or ASIC-based implementations [19]–[21]. Still, as these works did not take into account application-specific properties, they may not efficiently support the video coding memory demand and data transmission characteristics. *The challenge here is to exploit application-driven design of hybrid on-chip video memories tailored towards parallel HEVC.*

B. Overview of Our Main Contributions

Considering previous discussions, *the goal of this work is to provide energy efficiency for the memory infrastructure in Tiles-parallelized HEVC encoding*.

To this end, we designed a *hybrid scratchpad video memory architecture (called Hy-SVM)* that relies on joint inter- and intra-Tiles data reuse and hybrid memory design (based on combined SRAM and STT-RAM scratchpad memories) to allow energy-efficient storage in HEVC encoders. Furthermore, a memory management layer that leverages application-specific knowledge is proposed. A key concept in this work is the *overlap* formation, representing the main characteristics of the reference frame region that is accessed by more than one Tile processing. This concept is properly defined in Section II.A.


Application-Specific Knowledge
Memory Monitoring
Reference Frame Access Map [for 2+ cores]
Mem. Acc.
x coordinate (width)
Y coordinate (height)
Video Coding Knowledge
External Memory (DRAM) (Reference Frames)
Overlap (to Shared SPMs)
No Overlap (to Private SPMs)
Target Reference Frame
On-Chip Memory Management
Adaptive Power Management Unit (APMU)
Memory Access Management (MAMU)
Run-Time Overlap Prediction (inter-Tiles data reuse exploitation)
Energy-efficient Management Layer
Private SPMs (L1 and L2 Levels)
Shared SPMs (L2 Level)
Intra- and Inter-Tiles Data Reuse
STT-RAM
SRAM
Hybrid Memory Design
On-Chip Hybrid Scratchpad Video Memory (Hy-SVM)
Tiles Boundary
Tile 0
Tile 1
Tile 2
Tile 3
Input Video
Proc. Unit 0
Proc. Unit 1
Proc. Unit 2
Proc. Unit 3
Parallel HEVC Encoder
Fig. 2 – Overview diagram of our main contributions

An overview of Hy-SVM and their integration with Tiles-parallelized HEVC encoder is depicted in Fig. 2. It employs:

1) <u>Hybrid Scratchpad Video Memory (Hy-SVM) Architecture (Section IV)</u> that is composed of multiple levels of private and shared SPMs, as depicted in Fig. 2. It consists on (i) private L1[1] SPMs, implemented as SRAM arrays, to store the search window samples required for each HEVC processing unit; and (ii) private and shared L2 STT-RAM SPMs to provide reference frame level data reuse. The proposed design methodology leverages application-specific knowledge to define the hardware design parameters of SPMs.
2) <u>Run-Time Overlap Prediction (Section V)</u> that relies on application-specific knowledge (e.g., monitored past overlaps, video content and HEVC parameters; see Fig. 2) to estimate the overlap characteristics for the next frame encoding, improving the inter-Tiles data reuse potential. The predicted overlap properties are related to its size, shape and displacement.
3) <u>On-Chip Hy-SVM Management (Section VI)</u> that implements *memory access management units* (MAMUs) and *adaptive power management units* (APMUs) to manage the energy consumption of Hy-SVM (as in Fig. 2). Based on the overlap prediction output, the MAMUs implements read and write policies that manage the incoming Hy-SVM access to the corresponding SPM. Furthermore, APMUs can adapt the power gating strength according to the predicted overlap characteristics, which strongly depend on the video content.

[1] L1 and L2 in this work are not related to cache levels, but scratchpad memories implemented using either SRAM (L1 level) or STT-RAM (L2 level).

Paper organization: Section II presents background concepts regarding HEVC and STT-RAM; Section III discusses the main advantages/shortcomings of state-of-the-art works; Section IV introduces our Hy-SVM architecture, as well as the adopted organization models and the design methodology of SPMs; Section V presents our run-time overlap prediction scheme; Section VI describes our on-chip management units: MAMU and APMU; Section VII shows the experimental methodology; Section VIII discusses the experimental results and compares our savings with related works and baseline implementations; and, finally, Section IX concludes this work.

II. Background

This section introduces some preliminary concepts regarding HEVC and STT-RAM memory technology.

A. HEVC Background and Overlap Concept

The new coding structure of HEVC divides the video frame into flexible block sizes following a quad-tree structure called *coding-tree unit* (CTU) [4]. Typically the CTU partitioning starts from the maximum allowable block size of 64x64 pixels, and then explores breaking it into several *coding units* (CU) of sizes 32x32, 16x16 and 8x8 pixels. Fig. 3 depicts an example of CTU partitioning. For each CU, the encoder selects the best *prediction unit* generated by either intra or inter prediction steps. Inside the inter prediction, the *motion estimation* (ME) is the most complex and memory consuming module. The ME searches for the most similar block within a delimited portion of reference frames, called search window [4]. The ME processes each possible CU inside a CTU, thus resulting in the most time and energy consuming module of an HEVC encoder.

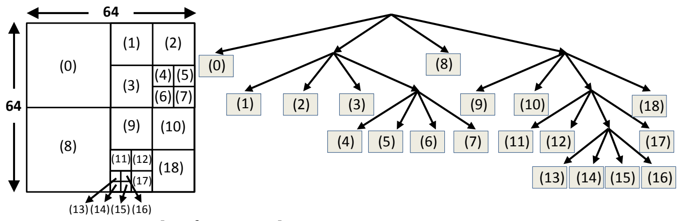


Fig. 3 - An example of HEC coding tree organization.

To exploit multiple cores in a many-core system, the HEVC provides a light-weight data parallelism support that divides the video frame into rectangular regions called *Tiles* [5]. In Fig. 4b, the 8x4 CTUs video frame is partitioned into 4 Tiles (in a 2x2 fashion). These Tiles can be encoded independent of each other without any spatial dependencies, thus can be parallelized on multiple processing units: either general-purpose processing cores or ASIC hardware accelerator encoding units.

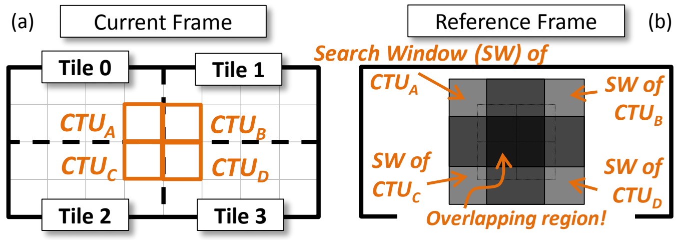


Fig. 4 - (a) 4-Tile partitioning and (b) overlap formation in a reference frame.

The samples near to the Tiles boundaries in the reference frame must be fetched by multiple Tiles processing, leading to external memory contention, redundant memory access and extra on-chip storage (causing energy wasting). An example in Fig. 4b depicts the *overlap* formation being the reference frame region that is accessed for encoding more than one Tile.

B. STT-RAM Background

A Spin-Transfer Torque RAM (STT-RAM) cell stores one logic bit in a magnetic tunneling junction – a very thin oxide layer interposed between two ferromagnetic layers. The resistance value of this junction is determined by the relative magnetic field direction between these two layers [22]. One layer has fixed magnetization (reference layer), while the other can have its magnetization changed due to a polarized programming current (called free layer). Thus, low resistances due to parallel and high resistances due to anti-parallel magnetizations represent the logic bits '0' and '1', respectively.

As an alternative to SRAM in the on-chip perspective, STT-RAM provides higher density, better scalability, non-volatile behavior, and low static power features compared to the SRAM. In other aspects SRAM is still more efficient, by having a lower write power and by its overall performance. Table I presents a subjective comparison between SRAM and STT-RAM technologies[2], where the dark-gray cells represent the best scenario of each parameter.

Table I SRAM vs. STT-RAM Technologies [22]

	Energy			Latency		Volatility
	Static	Read	Write	Read	Write	
SRAM	HH	**L**	**L**	**L**	**L**	Volatile
STT-RAM	**L**	**L**	HH	**L**	H	**Non-Volatile**

On-chip video memories require large arrays to implement data reuse schemes, which is aggravated for parallel video coding. The most promising STT-RAM characteristic is the low static energy consumption, becoming a suitable memory technology to implement such on-chip memories.

Another interesting property of on-chip video memories that facilitates the STT-RAM usage is: they have a relatively low write intensity compared to a very high read intensity. As the on-chip video memories implement data-reuse schemes for the search window samples, only a few data of the reference frame would be written to start the next CTU prediction. Once the needed data is stored on chip, the ME massively accesses the on-chip video memory until the best match is found. As can be noticed in Table I, the STT-RAM energy and performance are poor for write operations compared to that of the SRAM. Thus, video coding is a promising application for STT-RAM based hybrid memories.

STT-RAM has the advantage of a non-volatile memory (NVM). This characteristic is also very important for on-chip video memories, since parts of the memory may be switched-off (to eliminate static energy consumption) while keeping the data stored, leading to no extra external memory accesses to save and to re-fetch the data.

[2] "L" means *low*, "H" means *high*, and "HH" means *very high*.

III. Related Works

Since the HEVC release, several works were developed with the goal of exploiting its high-level parallelization features. In this context, different strategies have the goal of properly defining the best partitioning of the video frame into Tiles for parallel processing [23]–[27]. The proposed schemes typically take into account the workload of each processing unit to define the best tiling configuration. Only the strategy developed in [24] addresses the impacts of Tiles-based HEVC parallelization in the video memories. However, the proposed scheme focus on optimizing the memory design exploiting only intra-Tile data reuse, not considering the potential of inter-Tiles data reuse. Further, video memories characteristics of very high read intensity (compared to write operations) are not exploited in the work. Therefore, to address this gap *this work focus on exploiting the inter-Tiles data reuse, as well as intrinsic video memories characteristics, namely their high read access intensity*.

Several recent works exploited application-specific knowledge to propose energy-efficient memory design and management for HEVC. Reference frame compressing strategies are exploited in HEVC by [28]–[30] focusing on reducing the data bandwidth from on-chip and off-chip memories by compressing the reference frames data. Even though these works can be applied to parallel video encoding, they do not exploit parallel memory accesses from different processing units, leading to compromised scalability for increased parallelism. Further, on-chip memory energy is even aggravated, since multiple on-chip logic circuits must be inserted to implement the compressing/decompressing steps for the stored/fetched data. Dedicated video memories for HEVC encoding were already proposed by [3], [31]–[33]. In dSVM [3] and [31], SRAM-based distributed memory architectures were designed. The additional SRAM to improve the data reuse brings extra static energy consumption. Therefore, merely adding more SRAM becomes unfeasible when using a large number of processing cores. Dedicated hybrid video memories for video coding were developed in [32] and [33]. AMBER [32] uses SRAM only as FIFO buffers to hide the high write latency of STT-RAM cells, but not effectively being part of the storage system that may provide a high potential of energy-efficient design. Additionally, AMBER does not support parallel video processing, which is inevitable to achieve high processing throughput. enHyV architecture [33] combines SRAM and STT-RAM using private and shared SPMs. A design space exploration was performed to find the best optimization point between energy efficiency and STT-RAM endurance. However, the inter-Tiles data reuse potential is not properly exploited in this work, since no adaptive management is performed depending on the video content. Depending on the video properties (like low/high motion), energy may be wasted by not properly managing the shared video memories.

IV. Hybrid Scratchpad Video Memory Architecture (Hy-SVM)

Fig. 5 depicts our hybrid scratchpad video memory architecture (Hy-SVM) and its energy-efficient management layer for parallel HEVC encoding. Each Tile is assigned to a specific processing unit. The proposed memory organization increases the energy efficiency of reference frames management (off-chip fetching and on-chip storage). The coarser lines in Fig. 5 represent data connections, while finer lines illustrate the control flow between the modules.

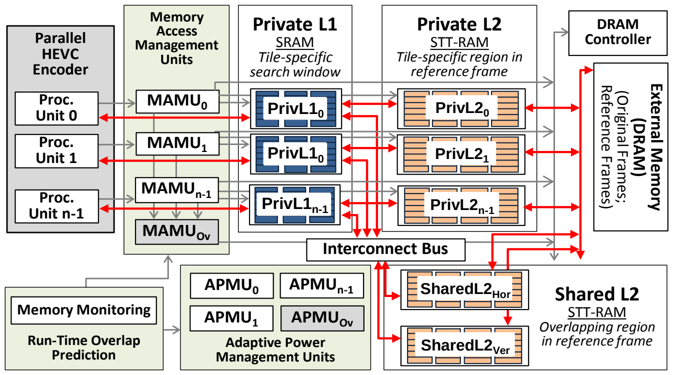


Fig. 5 – Block diagram of our hybrid scratchpad video memory architecture.

Our Hy-SVM architecture is organized as two levels of SPMs:

L1 SPMs Level: N_{Tiles} private SPMs[3] (PrivL1) that store the search window samples for a specific processing unit, allowing intra-Tile data reuse between each CU processing. At this level, the SPMs are implemented as SRAM arrays, providing equally high performance and energy efficiency for read and write operations. Since PrivL1 SPMs represent smaller memory cells arrays, SRAM static energy consumption does not significantly affect the overall energy efficiency.

L2 SPMs Level: N_{Tiles} private SPMs (PrivL2) and $N_{TilesBoundaries}$ shared SPMs (SharedL2) that together can store one complete reference frame, providing combined intra- and inter-Tiles data reuse. All L2 level SPMs are designed using STT-RAM technology, exploiting its high density and low static power features to implement large L2 data arrays. The PrivL2 stores the Tiles-specific region of the reference frame (accessed privately by the corresponding processing unit). Each SPM of PrivL2 has a direct data connection to the corresponding PrivL1 SPM. The SharedL2 SPMs are connected to the PrivL1 SPMs by an interconnect bus and it is responsible for the overlapping regions storage.

Along with SharedL2 SPMs, inter-Tiles data reuse is managed by a run-time overlap prediction that accurately estimates the redundant memory access behavior for the next ME. This prediction step is based on already monitored overlap formations from previous frames encoding. This knowledge is then forwarded to on-chip memory management hardware modules: (a) memory access management units (MAMUs) and (b) adaptive power management units (APMUs). They are responsible to effectively manage the SPMs by implementing a

[3] Let N_{Tiles} be the number of Tiles and $N_{TilesBoundaries}$ be the number of Tiles boundaries.

read/write policy, as well as proper power gating control over SPM sectors. The goal is to achieve the best possible energy efficiency depending on the video content properties. Details regarding the energy-efficient memory management layer are presented in Sections V and VI.

A. Adopted Memory Models and Notations Definition

Fig. 6 depicts the adopted on- and off-chip memory models that support Hy-SVM design. Every data transmission between the SPM and the external memory is based on a fixed basic unit (BU), which corresponds to the squared BU_{Dim} x BU_{Dim} picture block of the reference frame (see Fig. 6a). The samples within a BU are organized in a serialized way, so that all rows of an entire BU can be stored in the same external memory page (see Fig. 6b). Since consecutive accesses to the same memory page lead to less page-activation overhead, improved energy efficiency can be achieved.

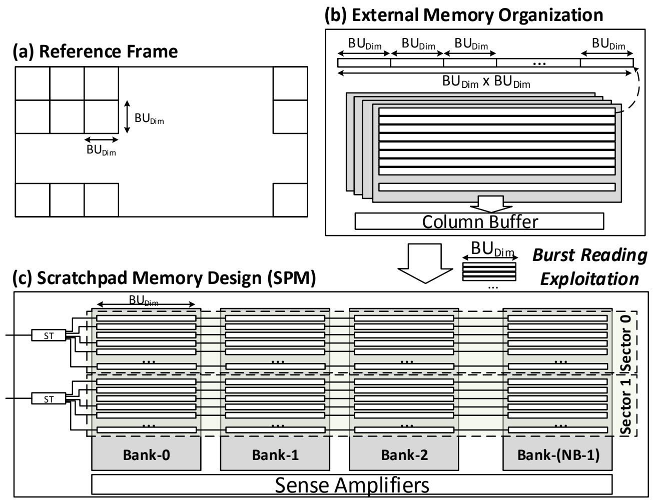


Fig. 6 – Adopted organization for the off- and on-chip memory parts.

As on-chip SPM design, a multi-bank memory organization is adopted (see Fig. 6c). Each SPM is composed of *NB* memory banks. To facilitate parallel access, each row of a BU is stored in a specific SPM bank. Hence, one line of a memory bank can store *SL* bytes, equals to the size of one BU row (BU_{Dim} bytes) [4]. The exception is the first SPM bank, which additionally stores control information for memory access management (explained in Section IV.B). A $Bank_i$ is composed of *NL* lines, grouped into *NS* memory sectors of *SS* bytes. The number of BUs per sector ($\underline{N_{BUPerSector}}$), which corresponds to the number of memory lines per sector ($N_{LinesPerSector}$), defines the power management granularity applied to the SPMs. The BU_{Dim} and $N_{BUsPerSector}$ are design-time parameters and should be carefully decided by the hardware designer.

B. Design of Hybrid Scratchpad Video Memories

Eq. (1)-(2) define the Tiles overlap sizing formula for the overlap thickness ($Ov_{Thickness}$) and length (Ov_{Length}). These formulas are used at design-time to properly derive the SharedL2 SPMs parameters. The $Ov_{Thickness}$ is calculated from the search window width or height, since it defines the maximum range ME can reach when searching in the reference frames. Additionally, ME start point can be displaced by prior analysis from neighboring motion predictors. Thus, the search window center can vary according to the motion field of Tiles boundaries. To represent that, an off-line statistical parameter Δ_{Motion} is inserted to scale the overlap thickness to be adapted to the average case of test sequences[5]. The Ov_{Length} is related to the frame width or height, when overlaps are formed around horizontal or vertical Tiles Boundaries, respectively.

$$\mathrm{Ov_{Thickness}}(\mathrm{TB_{ID}}) = \begin{cases} \Delta_{\mathrm{Motion}} \times \mathrm{SW_W}, & \text{if vertical boundary} \\ \Delta_{\mathrm{Motion}} \times \mathrm{SW_H}, & \text{if horizontal boundary} \end{cases} \tag{1}$$

$$\mathrm{Ov_{Length}}(\mathrm{TB_{ID}}) = \begin{cases} \mathrm{Frame_H}, & \text{if vertical boundary} \\ \mathrm{Frame_W}, & \text{if horizontal boundary} \end{cases} \tag{2}$$

Based on the memory organization defined in Section IV.A, we determine the physical sizing for the SPMs Levels in the proposed Hy-SVM architecture. As already explained, all SPMs (PrivL1, PrivL2, and SharedL2 levels) are composed of BU_{Dim} memory banks as in Eq. (3), which allows parallel access of one entire BU. However, the other SPM parameters are different depending on the Hy-SVM level.

$$\mathrm{PrivL1_{NB}} = \mathrm{PrivL2_{NB}} = \mathrm{SharedL2_{NB}} = \mathrm{B_{Dim}} \tag{3}$$

The PrivL1 SPMs store Tile-specific search window samples, requiring $PrivL1_{NL}$ memory lines, as expressed by Eq. (4). The first memory bank of a PrivL1 SPM must store, besides the first BU row, three control data: the horizontal and vertical BU frame position, and a validate bit (as in Eq. (5)). This information is important for MAMU to properly manage hit and miss occurrences. The PrivL1 level does not have associated power management, thus not requiring the overhead of sleep-transistors and memory sectors definition.

$$\mathrm{PrivL1_{NL}} = \mathrm{N_{BUsPerPrivL1}} = \left\lceil \frac{\mathrm{SW_W} \times \mathrm{SW_H}}{\mathrm{BU_{Size}}} \right\rceil \tag{4}$$

$$\mathrm{PrivL1_{SL}} = \begin{cases} (|\mathrm{BU_{XPos}}| + |\mathrm{BU_{YPos}}| + 1) + \mathrm{BU_{Dim}} & \text{if Bank}_0 \\ \mathrm{BU_{Dim}} & \text{otherwise} \end{cases} \tag{5}$$

The L2 level completely stores one reference frame, by having its samples distributed along the PrivL2 and SharedL2 SPMs. The PrivL2 SPMs stores the Tile-specific reference frame region, while the SharedL2 SPMs must support the overlapping regions size. The $PrivL2_{NL}$ number of memory lines depends on the frame resolution and the number of Tiles, as expressed in Eq. (6). In another perspective, the SharedL2 SPMs requires $SharedL2_{NL}$ lines, which is related to the $Ov_{Thickness}$ and Ov_{Length} overlap parameters; see Eq. (7). The PrivL2 SPMs of Hy-SVM are designed to guarantee that all BUs within the same reference frame have a specific associated memory line. Thus, it is not necessary to keep stored the frame position coordinates of the stored BU in a specific SPM line. To ensure correct hit/miss detection by MAMU, a validate bit is stored alongside the first BU row in $Bank_0$. The same scheme is adopted for SharedL2 SPMs, as defined in Eq. (8). Still, as in PrivL2, only one validate bit must be stored for each BU.

[4] In this work, we consider video sequences represented with 8-bit samples.

[5] In this work we adopted the same statistical method to determine Δ_{Motion} than [3].

$$\mathrm{PrivL2_{NL}} = \mathrm{N_{BUsPerPrivL2}} = \left\lceil \frac{\mathrm{Frame_W} \times \mathrm{Frame_H}}{\mathrm{BU_{Size}} \times \mathrm{N_{Tiles}}} \right\rceil \qquad (6)$$

$$\mathrm{SharedL2_{NL}} = \mathrm{N_{BUsPerSharedL2}} = \left\lceil \frac{\mathrm{Ov_{Thickness}} \times \mathrm{Ov_{Length}}}{\mathrm{BU_{Size}}} \right\rceil \qquad (7)$$

$$\mathrm{PrivL2_{SL}} = \mathrm{SharedL2_{SL}} = \begin{cases} \mathrm{BU_{Dim}} + 1 & \text{if } \mathrm{Bank_0} \\ \mathrm{BU_{Dim}} & \text{otherwise} \end{cases} \qquad (8)$$

Our adaptive power management strongly acts on L2 level of Hy-SVM to reduce on-chip static energy consumption. Note that as L2 SPMs are implemented as STT-RAM arrays, the shutdown of specific memory sectors does not imply on off-chip memory re-fetching, due to the non-volatile nature of STT-RAM cells. The power gating is applied for each memory sector. The already defined $N_{LinesPerSector}$ parameter indicates the adopted management level (as already discussed in Section IV.A). In doing so, the values of $PrivL2_{SS}$ and $SharedL2_{SS}$ are defined according to this design-time parameter, as in Eq. (9). As result, the number of memory sectors ($PrivL2_{NS}$ and $SharedL2_{NS}$), which directly affects the APMU design, is defined in Eq. (10) and (11), respectively.

$$\mathrm{PrivL2_{SS}} = \mathrm{SharedL2_{SS}} = \mathrm{N_{LinesPerSector}} \times \mathrm{BU_{Size}} \qquad (9)$$

$$\mathrm{PrivL2_{NS}} = \frac{\mathrm{PrivL2_{NL}}}{\mathrm{L2_{SectorSize}}} \qquad (10)$$

$$\mathrm{SharedL2_{NS}} = \frac{\mathrm{SharedL2_{NL}}}{\mathrm{L2_{SectorSize}}} \qquad (11)$$

The following sections describe the memory management layer, which improves the energy efficiency of Hy-SVM architecture. First, the run-time overlap prediction scheme is presented. Then, the on-chip memory management units, MAMU and APMU, are explained.

V. Run-Time Overlap Prediction

We focus our evaluations and proposed overlap prediction scheme in the HEVC-defined Low Delay (LD) and Random Access (RA) prediction structures, as illustrated in Fig. 7. Each arrow denotes a prediction dependency evaluated by the ME, starting from the current frame and pointing to the used reference frame. We assign an overlap identification (Ov_{ID}) for each prediction dependency. Further, another important parameter is the distance between the current and reference frame of each Ov_{ID}, represented by the notation D_{ME}. It is defined as the absolute difference between the picture exhibition order number between the two frames: $\mathrm{D_{ME}}(\mathrm{Ov_{ID}}) = |\mathrm{F_{Curr}} - \mathrm{F_{Ref}}|$ (depicted in the bottom of Fig. 7). For example, the D_{ME} of the prediction $RA2$ is calculated as $D_{ME}(RA2) = |4 - 8| = 4$.

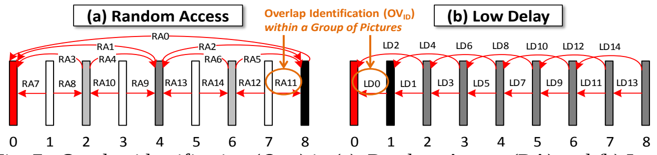


Fig. 7 - Overlap identification (Ov_{ID}) in (a) Random Access (RA) and (b) Low Delay (LD) HEVC encoder configurations.

Note that each ME will lead to the formation of an overlapping region. The characteristics of the overlaps were evaluated to base our run-time overlap prediction scheme (Section V.A). An accurate estimation of such properties is important (a) to improve inter-Tiles data reuse (exploited by the SharedL2 SPMs), as well as (b) to provide less-frequent *ON-OFF* switching activities, leading to higher energy savings for our adaptive power management scheme. To support the variability of the overlap characteristics, a light-weight overlap data representation is proposed (Section V.B). The overlap prediction scheme is described in Section V.C.

A. Overlaps Correlation Evaluation

Memory analyses were performed with the goal of identifying correlated parameters of overlap formations between consecutive MEs. The evaluations consider three important overlap characteristics: size, shape, and displacement.

Analysis-1 (Overlap Size): Fig. 8 presents an evaluation of the overlap size by exploiting MEs with different D_{ME} parameters. In this case, we are interested in the number of redundant memory accesses within a reference frame depending on the absolute value of the distance D_{ME}. Thus, this analysis does not consider the prediction direction. We can note that the overlap size reduces when lower D_{ME} MEs are executed. Therefore, an insight is to leverage the size of past overlaps in our prediction scheme. In doing so, the relation between the D_{ME} factors must be taken into account to scale the predicted overlap accordingly. Our APMU can exploit it by dynamically applying relaxed or aggressive power gating to improve the SPMs energy efficiency according to predicted memory demand.

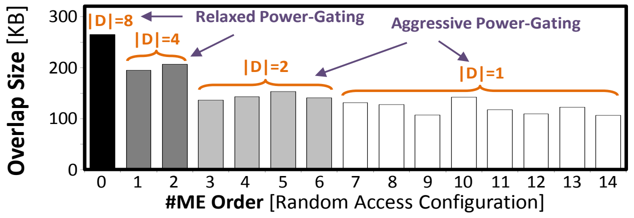


Fig. 8 – Example of overlap sizing variation for several temporal distances (D factor).

Analysis-2 (Overlap Shape): Besides the size, another important aspect is the overlap shape, which may significantly change along the overlap length. We can note this dynamic behavior in Fig. 9, where the shape varies according to the video content. Not exploiting this variation may lead to inefficiency to memory energy consumption (on- and off-chip parts). Furthermore, there is a significant similarity between the shapes of overlaps when analyzing consecutive ME processing, as can be noted in Fig. 9. Therefore, the shape characteristics of previous formations can be used as reference to improve the prediction accuracy for the next overlaps.

Analysis-3 (Overlap Displacement): Fig. 10 presents an analysis (Probability Density Function charts) comparing the overlap displacement for ME steps with different D_{ME} factors. The displacement, in this evaluation, was measured by the distance of the center of the actual overlap regarding the Tiles boundary. When we compare the generated overlap between MEs with the same prediction direction (Fig. 10a), we can note that higher D_{ME} factors lead to higher and spreader overlap displacements. In another vein, ME operations with lower D_{ME} values lead overlaps centered nearer the Tiles boundary, as well with a more concentrated behavior. In Fig. 10b, comparing MEs with opposite prediction directions (D_{ME} values with different signals), we can observe opposite displacements in the formed overlaps. Therefore, regarding this aspect, our insight is to leverage the past overlap displacement weighted by the difference of D_{ME} factors.

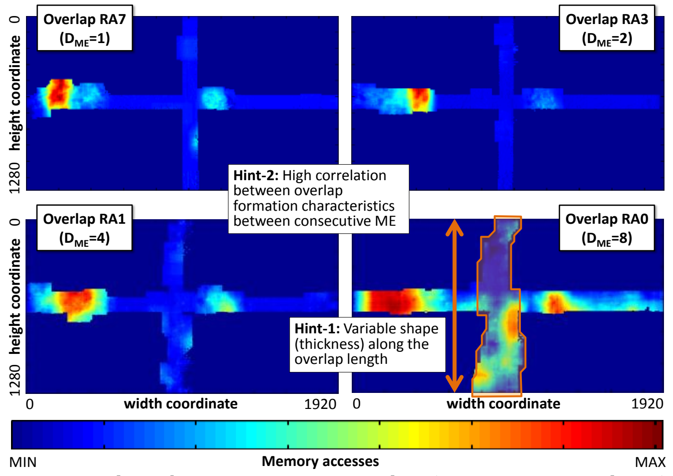


Fig. 9 - Correlation between consecutive overlaps (RA7, RA3, RA1 and RA0), considering Random Access prediction structure.

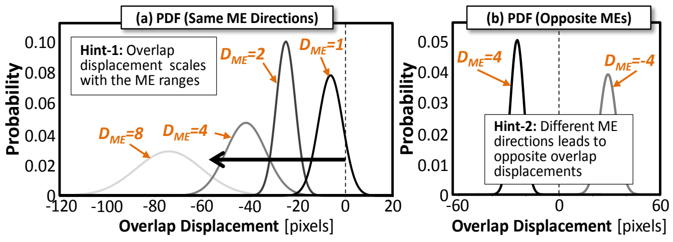


Fig. 10 - Overlap displacement correlation analysis.

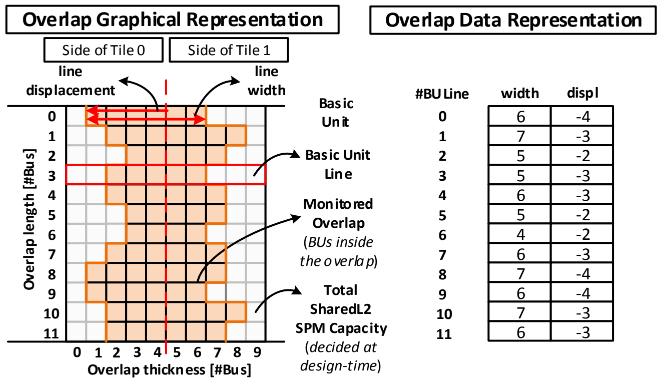
To exploit the discussed overlaps correlation in our prediction unit, we implement an overlap representation that properly models the absolute size, and the variable shape and displacement properties.

B. Overlap Representation

Eq. (12) models an overlap as an ordered set of tuples, each one containing width (*width$_i$*) and displacement (*displ$_i$*) information of a specific basic unit line *i* within the overlap. The level of representation is based on the adopted BU dimension, being compliant to Hy-SVM organization.

$$Ov_{ID} = \{(width_i, displ_i), \forall\, BU\ line\ i \mid 0 \le i < Ov_{Length}/BU_{Dim}\} \tag{12}$$

Fig. 11 illustrates an example of overlap representation. In the graphical view (Fig. 11a), we can observe the possibility of modeling the variations of width and displacement along the overlap length. For each BU line, the width is related to the overlap thickness (in number of BUs), while the displacement is expressed as the distance of the first BU from the Tiles boundary center. Hence, the design-time parameters Ov_{Length} and $Ov_{Thickness}$ related to the overlap thickness and length (previously defined in Section IV.B), utilized to design the SharedL2 SPMs, are refined to provide a more accurate representation of the actual formed overlap. The mapping between the graphical and the data representation is presented in Fig. 11b. We can note that each BU line within the overlap has associated width and displacement information.



Overlap Graphical Representation
Side of Tile 0
Side of Tile 1
line displacement
line width
Basic Unit
Basic Unit Line
Monitored Overlap (BUs inside the overlap)
Total SharedL2 SPM Capacity (decided at design-time)
Overlap length [#Bus]
Overlap thickness [#Bus]
Overlap Data Representation

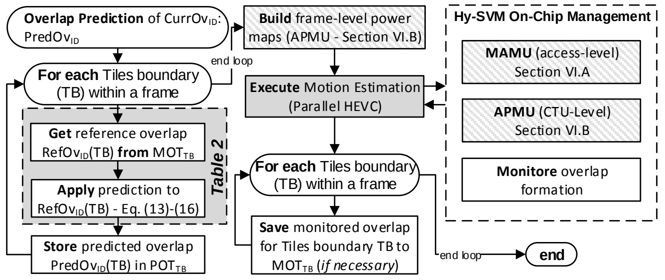
#BU Line	width	displ
0	6	-4
1	7	-3
2	5	-2
3	5	-3
4	6	-3
5	5	-2
6	4	-2
7	6	-3
8	7	-4
9	6	-4
10	7	-3
11	6	-3

Fig. 11 – (a) Graphical and (b) data representation of an overlap by our energy-efficient management of Hy-SVM.

Considering a 4-Tile HD1080p HEVC encoder, 256x256 search window size, and BU_{Dim}=8, the overlap representation for the horizontal Tiles boundary requires 480 bytes, while the vertical overlap occupies 136 bytes. These values represent a negligible overhead, especially when comparing to the hardware resources required to implement L1 and L2 SPM Levels (as demonstrated in Section VIII.D).

C. Overlap Prediction Scheme

Our scheme is inspired on the video coding idea of selecting several references (past coded information) to predict the behavior of the data that is being processed. Therefore, for each overlap that is being predicted (called *current overlap*), the information of past monitored overlap formations (called *reference overlaps*) are exploited. As result, an estimation of the formation characteristics for the current overlap is generated (called *predicted overlap*). In this context, there are two key data structures: the *Monitored Overlaps Table* (MOT) and the *Predicted Overlap Table* (POT).



Overlap Prediction of CurrOv$_{ID}$: PredOv$_{ID}$
For each Tiles boundary (TB) within a frame
Get reference overlap RefOv$_{ID}$(TB) from MOT$_{TB}$
Apply prediction to RefOv$_{ID}$(TB) - Eq. (13)-(16)
Table 2
Store predicted overlap PredOv$_{ID}$(TB) in POT$_{TB}$
end loop
Build frame-level power maps (APMU - Section VI.B)
Execute Motion Estimation (Parallel HEVC)
For each Tiles boundary (TB) within a frame
Save monitored overlap for Tiles boundary TB to MOT$_{TB}$ (if necessary)
end loop
end
Hy-SVM On-Chip Management
MAMU (access-level) Section VI.A
APMU (CTU-Level) Section VI.B
Monitore overlap formation

Fig. 12 – Flowchart of our run-time overlap prediction scheme and its relation to Hy-SVM on-chip management units.

Fig. 12 depicts the flowchart of our overlap prediction scheme, as well as its integration with Hy-SVM on-chip management units (detailed presented in Section VI). During a ME processing, our Memory Monitoring Unit monitors the inter-Tiles redundant accesses. This unit utilizes one bitmap for each HEVC processing unit to identify the accessed BUs within a reference frame. As result, the bitmaps are combined, and the monitored overlap representation (presented in the previous section) is generated and stored in the MOT. There is a specific MOT_{TB} for each Tile boundary *TB*, which is responsible to store a historic of the past monitored overlap formations of this specific boundary.

For each Tiles boundary within a frame, our scheme accesses the MOT_{TB} to get the reference overlap $RefOv_{ID}(TB)$ that will be used for the prediction of $CurrOv_{ID}(TB)$. To minimize the MOTs size and guarantee the best possible correlation between current and reference overlaps, Table II proposes a prediction assignment based on the correlations of MEs, considering the Random Access and Low Delay configurations. The overlap identifications follow the notations defined in Fig. 7. Based on this assignment, a prediction operation is applied to estimate the predicted overlap based on the monitored information from the selected reference overlap. The prediction process is based on one of four possible operations: *downscale*, *upscale*, *invert* or *copy*; as defined in Eqs. (13)-(16), where the α and β are offline statistical factors that were extracted by experimental analysis using real-world video coding scenarios[6].

TABLE II OVERLAP PREDICTION ASSIGNMENT FOR *RANDOM ACCESS* AND *LOW DELAY* HEVC ENCODER CONFIGURATIONS

Random Access			Low Delay		
Curr. Ov$_{ID}$	**Prediction Operation**	**Ref. Ov$_{ID}$**	**Curr. Ov$_{ID}$**	**Prediction Operation**	**Ref. Ov$_{ID}$**
RA0	off-line stats. (if first frame) or copy RA0 from previous GOP		LD0	off-line stats. (if first frame) or copy RA0 from previous GOP	
RA1	downscale(α)	RA0	LD1	copy	LD0
RA2	invert	RA1	LD2	upscale(β)	LD1
RA3	downscale(α)	RA1	LD3	copy	LD1
RA4	invert	RA3	LD4	upscale(β)	LD3
RA5	downscale(α)	RA2	LD5	copy	LD3
RA6	invert	RA5	LD6	upscale(β)	LD5
RA7	downscale(α)	RA3	LD7	copy	LD5
RA8	invert	RA7	LD8	upscale(β)	LD7
RA9	downscale(α)	RA4	LD9	copy	LD7
RA10	invert	RA9	LD10	upscale(β)	LD9
RA11	downscale(α)	RA5	LD11	copy	LD9
RA12	invert	RA11	LD12	upscale(β)	LD11
RA13	downscale(α)	RA6	LD13	copy	LD11
RA14	invert	RA13	LD14	upscale(β)	LD13

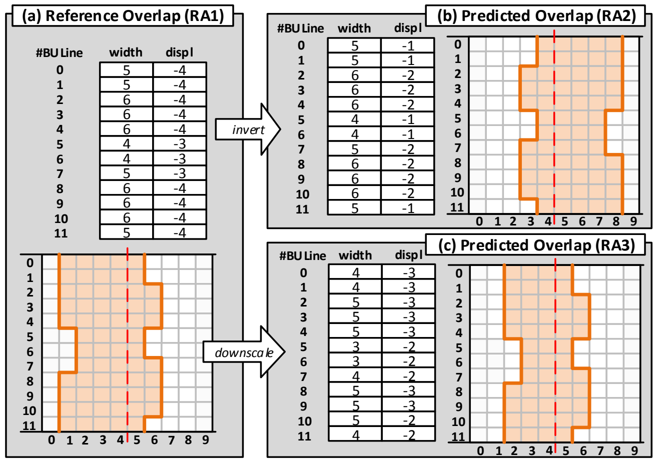
$$\begin{aligned}&\underline{\mathrm{downscale}(\mathrm{PredOv_{ID}}, \mathrm{RefOv_{ID}}, \alpha):}\\&\mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{width} \leftarrow \lceil \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{width} \times \alpha \rceil \textit{ and}\\&\mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{displ} \leftarrow \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{displ},\\&\forall\ \mathrm{BU\ line\ i}, \textit{where } \alpha < 0\end{aligned} \tag{13}$$

$$\begin{aligned}&\underline{\mathrm{upscale}(\mathrm{PredOv_{ID}}, \mathrm{RefOv_{ID}}, \beta):}\\&\mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{width} \leftarrow \lceil \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{width} \times \beta \rceil \textit{ and}\\&\mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{displ} \leftarrow \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{displ},\\&, \forall\ \mathrm{BU\ line\ i}, \textit{where } \beta > 0\end{aligned} \tag{14}$$

$$\begin{aligned}&\underline{\mathrm{invert}(\mathrm{PredOv_{ID}}, \mathrm{RefOv_{ID}}):}\\&\mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{width} \leftarrow \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{width} \textit{ and}\\&\mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{displ} \leftarrow -(\mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{width} +\\&\qquad\qquad \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{displ}), \forall\ \mathrm{BU\ line\ i}\end{aligned} \tag{15}$$

$$\begin{aligned}&\underline{\mathrm{copy}(\mathrm{PredOv_{ID}}, \mathrm{RefOv_{ID}}):}\\&\mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{width} \leftarrow \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{width} \textit{ and}\\&\mathrm{PredOv_{ID}}[\mathrm{i}].\mathrm{displ} \leftarrow \mathrm{RefOv_{ID}}[\mathrm{i}].\mathrm{displ}, \forall\ \mathrm{BU\ line\ i}\end{aligned} \tag{16}$$

Example: Consider the RA1 prediction dependency was processed by ME and the inter-Tiles redundant memory accesses was monitored, generating the $RefOv_{RA1}$ (illustrated in Fig. 13a). From the proposed prediction assignment for Random Access configuration presented in Table II, the selected prediction operation for the next $PredOv_{RA2}$ and $PredOv_{RA3}$ are *invert* and *downscale*, respectively. In Fig. 13b we can observe the result of invert prediction operation, where the overlap is displaced from left to right part related to the Tiles boundary. In this case, the RA1 and RA2 have the same absolute value of DME, but with different signals: $D_{ME}(RA1)=4$ and $D_{ME}(RA2)=-4$. In this case, due to the opposite motion directions, the overlap formations tends to be displaced (as motivated in Analysis-2 of Section V.A). The estimation of $PredOv_{RA3}$, reduces the overlap width parameters by a $\alpha=0.75$ factor. The prediction dependencies RA1 and RA3 have same direction but different distances: $D_{ME}(RA1)=4$ and $D_{ME}(RA2)=2$. In doing so, the overlap for RA3 tends to be smaller than RA1, since lower motion fields will be detected.


(a) Reference Overlap (RA1)
(b) Predicted Overlap (RA2)
(c) Predicted Overlap (RA3)
invert
downscale
0 1 2 3 4 5 6 7 8 9

(a) Reference Overlap (RA1)

#BU Line	width	displ
0	5	-4
1	5	-4
2	6	-4
3	6	-4
4	6	-4
5	4	-3
6	4	-3
7	5	-3
8	6	-4
9	6	-4
10	6	-4
11	5	-4

(b) Predicted Overlap (RA2)

#BU Line	width	displ
0	5	-1
1	5	-1
2	6	-2
3	6	-2
4	6	-2
5	4	-1
6	4	-1
7	5	-2
8	6	-2
9	6	-2
10	6	-2
11	5	-1

(c) Predicted Overlap (RA3)

#BU Line	width	displ
0	4	-3
1	4	-3
2	5	-3
3	5	-3
4	5	-3
5	3	-2
6	3	-2
7	4	-2
8	5	-3
9	5	-3
10	5	-2
11	4	-2

Fig. 13 – Example of overlap prediction operations when estimating RA2 (invert) and RA3 (downscale; using α=0.75) from RA1 reference overlap.

The predicted overlap $PredOv_{ID}(TB)$ is stored in the POT_{TB} to become available for MAMUs and APMUs operations. Details regarding the implementation of Hy-SVM on-chip management units are given as follows.

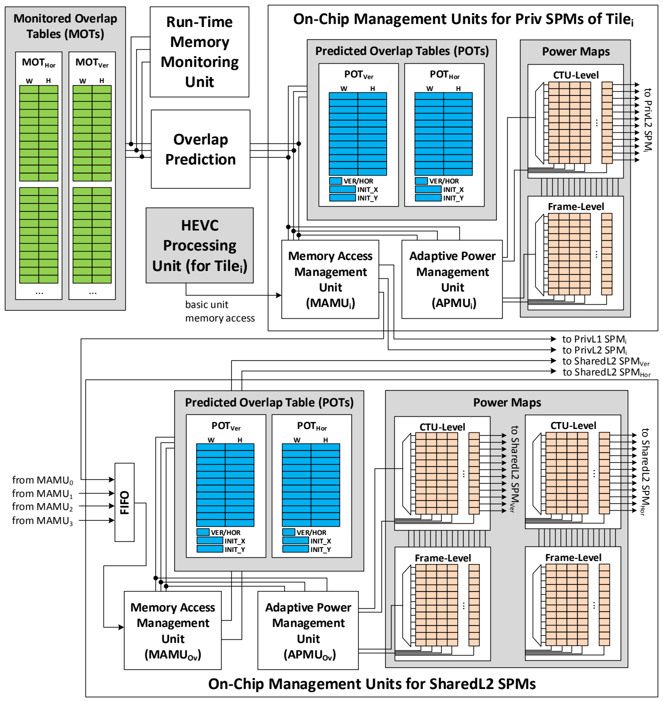
VI. ON-CHIP HY-SVM MANAGEMENT UNITS

Fig. 14 presents the block diagram of the proposed on-chip Hy-SVM management units. As example, Fig. 14 illustrates hardware details of a 4-Tile HEVC case study, which has one horizontal (*Hor*) and one vertical (*Ver*) Tiles boundary. The main goal is to rely on accurate overlap prediction to employ energy-efficient memory access and power management to designed SPMs in Hy-SVM architecture. As previously explained, the overlap prediction leverages past overlap formations of past MEs, which are kept stored in the MOT. The proposed memory monitoring is the unit responsible for capturing the inter-Tiles redundant memory access behavior. The prediction unit stores the predicted overlap of a corresponding Tiles boundary in the POTs.

Each HEVC processing unit *i* has an associated instance of memory access management unit ($MAMU_i$) and of adaptive power management unit ($APMU_i$). These modules utilize the predicted overlap, available in the POTs, to provide energy-

[6] We adopted: $\alpha=0.75$ and $\beta=0.8$ in our experiments.

efficient management of the PrivL1 and PrivL2 SPMs for the processing unit *i*. The MAMU$_i$ receives a memory access request and, based on a read/write policy, translates the address and forwards the operation to either PrivL1 or PrivL2. Further, if the incoming access is related to a basic unit inside any predicted overlap, the request is forwarded to a MAMU$_{Ov}$, responsible of managing the SharedL2 SPMs accesses. As the private MAMUs require the knowledge of the predicted overlaps, each unit has an instance of the POTs (as in Fig. 14). The APMU$_i$ analyzes the POT content and HEVC parameters to build the power maps for the PrivL2 SPM$_i$. The power maps are directly connected to the sleep-transistors that control the power state of each sector of STT-RAM array. Additionally, a specific APMU$_{Ov}$ module manages the power gating operation of SharedL2 SPMs. Details regarding MAMUs and APMUs implemented schemes are given as follows.

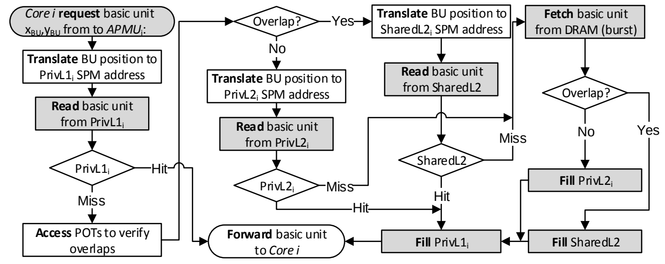


Monitored Overlap Tables (MOTs)
MOT$_{Hor}$
MOT$_{Ver}$
Run-Time Memory Monitoring Unit
Overlap Prediction
HEVC Processing Unit (for Tile$_i$)
basic unit memory access
On-Chip Management Units for Priv SPMs of Tile$_i$
Predicted Overlap Tables (POTs)
POT$_{Ver}$
POT$_{Hor}$
Power Maps
CTU-Level
Frame-Level
to PrivL2 SPM$_i$
Memory Access Management Unit (MAMU$_i$)
Adaptive Power Management Unit (APMU$_i$)
to PrivL1 SPM$_i$
to PrivL2 SPM$_i$
to SharedL2 SPM$_{Ver}$
to SharedL2 SPM$_{Hor}$
from MAMU$_0$
from MAMU$_1$
from MAMU$_2$
from MAMU$_3$
FIFO
Predicted Overlap Table (POTs)
to SharedL2 SPM$_{Ver}$
to SharedL2 SPM$_{Hor}$
Memory Access Management Unit (MAMU$_{Ov}$)
Adaptive Power Management Unit (APMU$_{Ov}$)
On-Chip Management Units for SharedL2 SPMs

Fig. 14 – Block diagram of our on-chip Hy-SVM management units integrated to the run-time overlap prediction and memory monitoring units.

A. *Memory Access Management Unit (MAMU)*

Our MAMU implements a read/write policy (see flowchart of Fig. 15) that takes advantage from the Tiles overlap to increase the data-reuse of the reference frame samples. When HEVC processing unit *i* requests a BU of positions x_{BU} and y_{BU} to Hy-SVM, as first step, the MAMU translates the BU frame positions to PrivL1 SPM address space. Then, it performs a PrivL1 SPM$_i$ access to check for hit/miss occurrence. In case of a hit, the BU is forwarded to the processing unit. Otherwise, a miss leads to the access of L2 level SPMs. At this point, the MAMU checks along with the predicted overlaps if the requested BU belongs to one Tiles overlapping region. Assuming that the data is inside an overlap related to the Tiles intersection *TB*, the corresponding SharedL2 SPM$_{TB}$ is then accessed. In this case, inter-Tiles data reuse is exploited, since the processing of all Tiles that share the Tiles boundary *TB* may request the same data. For non-overlapping regions, the PrivL2 SPM$_i$ is accessed, leading to intra-Tile data reuse. Note that for each core data request, either a ShreadL2 SPM or a PrivL2 SPM is accessed. If a L2 hit is verified, the data is forwarded to the processing unit and the PrivL1 SPM$_i$ is filled with the requested BU. In case of a L2 miss, the BU must be fetched from the external memory and written to either PrivL2$_i$ or SharedL2$_{TB}$ SPM (depending on the predicted overlap) and the PrivL1 SPM$_i$. After that, the data is forwarded to the Tiles-specific HEVC processing unit.



Core i request basic unit x_{BU},y_{BU} from to APMU$_i$:
Translate BU position to PrivL1$_i$ SPM address
Read basic unit from PrivL1$_i$
PrivL1$_i$
Hit
Miss
Access POTs to verify overlaps
Overlap?
Yes
No
Translate BU position to PrivL2$_i$ SPM address
Read basic unit from PrivL2$_i$
PrivL2$_i$
Miss
Hit
Translate BU position to SharedL2$_i$ SPM address
Read basic unit from SharedL2
SharedL2
Miss
Hit
Fetch basic unit from DRAM (burst)
Overlap?
No
Yes
Fill PrivL2$_i$
Fill SharedL2
Fill PrivL1$_i$
Forward basic unit to Core i

Fig. 15 – Flow of our memory access management unit with read/write policy.

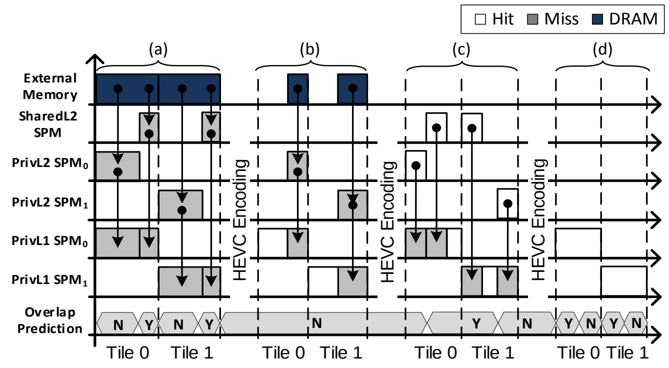


Fig. 16 – An example of data interaction for a 2-Tile HEVC encoding.

Case-study example: Fig. 16 depicts an example of data migration for our read/write policy in four different cases, considering a 2-Tile HEVC encoding system.

(a) In the beginning, the on-chip SPMs are empty and each request will lead to external memory fetching (L1 and L2 misses). Fig. 16 shows that predicted overlap is analyzed to determine whether the reference BU is stored in the PrivL2 SPM$_i$ or in the SharedL2 SPM$_{TB}$. The PrivL1 SPM$_i$ is always filled with the fetched data. During the frame encoding, due to the intra-Tile (PrivL1 and PrivL2 SPMs) and inter-Tiles (SharedL2 SPMs) reused data, more hits occur and even less external memory communication is needed.

(b) The second case of Fig. 16 depicts Tiles-centering CTUs processing where only the PrivL2 SPMs are accessed (only intra-Tile data reuse). Note that all accesses inside this case are directed to reference frame BUs outside the predicted overlap. We can also observe some PrivL1 SPMs hits, which avoid L2 SPMs accessing and external memory fetching.

(c) The third case illustrates accesses from CTUs located close to the Tiles boundary. In this scenario, L2 memory hits are verified for both PrivL2 and SharedL2 SPMs (i.e. combined intra- and inter-Tiles data reuse). This case represents the best energy efficiency when requiring L2 level access.

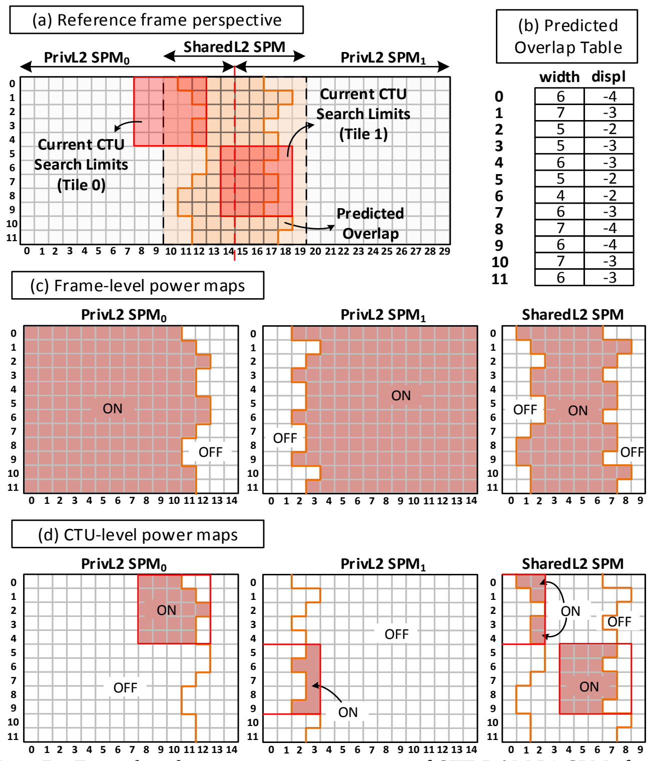
(d) The last scenario of Fig. 16 presents the best case of energy efficiency, where all memory accesses result on PrivL1 hits.

B. *Adaptive Power Management Unit (APMU)*

Our APMU leverages the predicted overlaps and the search limits of current CTUs to further reduce the static energy consumption of Hy-SVM. The SPMs in the L2 level of Hy-SVM (implemented as STT-RAM) were designed to operate in two power states: *ON* ($V_{ON}=V_{DD}$ volts) and *OFF* ($V_{OFF}=0$ volts). Due to the non-volatility characteristic of STT-RAM, the data is kept stored in the memory cell even when *OFF* state is assigned (differently from SRAM cells). Further, L2 SPMs are typically significantly larger than L1 SPMs, leading to higher energy consumption. In doing so, our APMU concentrates effort in L2 SPMs, resulting in a great impact in the Hy-SVM overall on-chip energy (as demonstrated in Section VIII).

In Hy-SVM, the L2 Level can store an entire reference frame, exploiting STT-RAM reduced leakage power and providing high intra-Tile and inter-Tiles data reuse, leading to reduced external memory energy. Besides, the ME required memory accesses for all CUs within a CTU is limited to a search window, which represents a small portion of the whole reference frame. Our APMU scheme relies on estimates the search limits for the entire CTU processing, which combines the search window of the ME for all CUs. The CTU search limits are defined as a squared region of BUs of $\lceil(\mathrm{SL_{Dim}} \times \mathrm{SL_{Dim}})/\mathrm{BU_{Size}}\rceil$ size, where $\mathrm{SL_{Dim}} = \mathrm{CTU_{Size}} + \mathrm{SW_{Dim}}$.

An example of the APMU operation is illustrated in Fig. 17. In the first part, the adopted 2-Tile HEVC encoding scenario is represented at reference frame perspective (Fig. 17a). The current CTU search limits of Tile 0 and Tile 1 are depicted, as well as the predicted overlap (stored in the POT using the proposed representation, as in in Fig. 17b).


(a) Reference frame perspective
PrivL2 SPM$_0$
SharedL2 SPM
PrivL2 SPM$_1$
Current CTU Search Limits (Tile 0)
Current CTU Search Limits (Tile 1)
Predicted Overlap
(b) Predicted Overlap Table
(c) Frame-level power maps
ON
OFF
(d) CTU-level power maps

	width	displ
0	6	-4
1	7	-3
2	5	-2
3	5	-3
4	6	-3
5	5	-2
6	4	-2
7	6	-3
8	7	-4
9	6	-4
10	7	-3
11	6	-3

Fig. 17 – Example: adaptive power management of STT-RAM L2 SPMs for a 2-Tile scenario.

At the beginning of frame processing, the APMU builds one frame-level power map for each L2 SPMs. For the PrivL2 SPMs, the memory sectors outside the predicted overlap have associated power state set as *ON*. Otherwise, the *OFF* state is assigned. As previously discussed, L2 accesses are directed either to a PrivL2 SPM or a SharedL2 SPM, depending on the predicted formation of the overlap. The APMU frame-level power map building process is the opposite for the SharedL2 SPMs, as presented in Fig. 17c. The frame-level power maps are not directly assigned to the sleep-transistors of the L2 SPMs, being start points to compose the CTU-level power maps. At CTU level, our scheme checks the frame-level power map against the search limits of the current CTUs. Note that the PrivL2 SPMs must be checked against the search limits of its corresponding Tile processing, while the SharedL2 SPMs must be analyzed considering the search limits of all HEVC processing units. The SPM sectors outside the search limits are set as *OFF* state, resulting on CTU-level power maps of Fig. 17d. By assigning *ON* state for the STT-RAM sectors inside the CTU search limit, we ensure long sleep durations during one entire CTU processing.

VII. Evaluation Methodology

A custom simulator was developed to capture the video memory access traces. Based on adopted memory power models, it estimates the energy consumption for on- and off-chip parts, as well as the overhead of implementing our run-time management schemes.

Regarding the design-time parameters of Hy-SVM (defined in Section IV.A), we defined BU_{Dim} as 8 due to the smallest possible CU size defined by HEVC [1]. To provide balanced fine- and coarse-grain power management, the $N_{BUsPerSector}$ is set as 4. In doing so, each sleep transistor determine the power state of one 16x16 reference frame region stored in the SPM.

A. *Memory Power Models*

The CACTI 6.5 tool [34] was used for on-chip SRAM energies/latencies considering 32nm memory cells. The 32nm STT-RAM electrical parameters were extracted using the NVSim tool [35]. To estimate static energy reduction, as well as wake-up latencies/energies, we adopted the analytical model proposed in [36]. As external memory, the 4-Gbit Low-Power DDR2 (LPDDR2) DRAM MT42L128M16D1GU-25WT [37] chip was used. The main specifications are: *VDD=1.2V*, *Freq=533MHz*, word size of 32 bits, page size equals to 512 bytes, 16K rows and 2K columns. The total energy is derived by the composition of six components: page activation energy, write energy, read energy, I/O pins energy, refresh energy and standby energy [38], [39].

B. *Video Coding Parameters*

The experimental evaluations are based on the recommended HEVC common test conditions [40] using the HEVC test model (HM 13.0) [41]. We perform analyses for 2, 4, 8 and 16 uniform Tile partitioning scenarios based on a 128x128 search window size. In total, fourteen different video sequences with distinct properties were evaluated: *BasketballDrive (BDrive)*, *Beauty*, *Bosphorus*, *BQTerrace (BQTerr)*, *Cactus*, *Kimono*, *ParkScene (PScene)*, *ReadySteadyGo (RSGo)*, *ShakeNDry (SNDry)* and

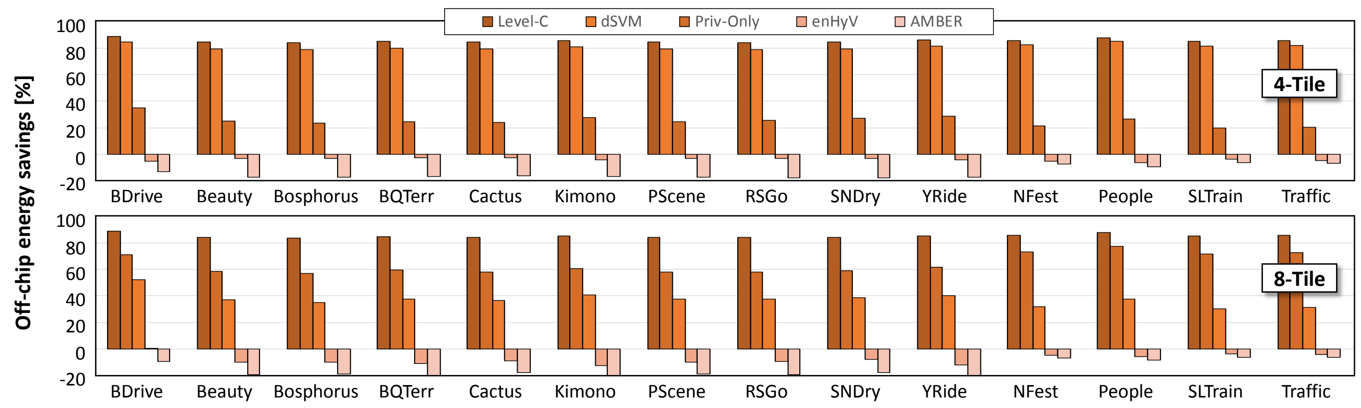


Fig. 18 – Off-chip energy savings of Hy-SVM compared to related works (128x128 search window size).

YachtRide (YRide) - HD1080 (1920x1080 pixels); *NebutaFestival (NFest)*, *PeopleOnStreet (People)*, *SteamLocomotiveTrain (SLTrain)* and *Traffic* (2K – 2560x1600 pixels). Other HEVC settings are: GOP=8, CABAC, FRExt, Random Access configuration, and TZ Search algorithm.

C. *Comparison-Purpose Baseline Hy-SVM Architectures*

Besides the comparison with related works, we implemented baseline variations of Hy-SVM to measure the efficiency of our design decisions. Hence, three alternative comparison-purpose architectures were evaluated in our experiments:

- *All-SRAM*: adopts the SRAM technology for all SPMs in Hy-SVM architecture. The goal is to evaluate the benefits (static energy consumption) and shortcomings (poor write efficiency) of using STT-RAM in L2 SPMs;
- *Priv-Only*: avoids the usage of Shared SPMs, thus exploiting only intra-Tile data reuse. The main purpose is to evaluate the impacts of SharedL2 SPMs, as well as the overlap management efficiency;
- *No-APMU*: avoids the proposed power management over L2 SPMs. The goal is to evaluate the contributions of APMU in the on-chip energy savings of Hy-SVM.

VIII. Experimental Results

The discussion of our experimental results is composed of four parts: initially, (in subsection A) the energy consumption regarding the external memory is analyzed; then, (in B) the on-chip energy savings are evaluated; later, (in C) the overall energy is computed and compared with related works and baseline implementations; finally, (in D) an overhead analysis is presented in terms of access latency and dynamic energy, as well as required extra on-chip memory size.

A. *Off-Chip Energy Results*

Fig. 18 depicts the off-chip energy savings of Hy-SVM compared to related implementations. The analysis was performed for 4-Tile and 8-Tile HEVC encoding considering different video sequences.

As first remark from the experiments, we observe that the off-chip energy savings may vary according to video content properties. HEVC encoding of high motion sequences (like *Kimono* and *BasketballDrive* in Fig. 18) lead to larger overlaps, since the motion search reaches more distant reference frame samples. In these cases, our Hy-SVM architecture is able to exploit this increased inter-Tiles data reuse potential and save external memory communication.

Level-C [9] represents the upper bound results, since it only exploits intra-Tile data reuse in search window level. Hy-SVM can save up to 85% off-chip energy compared to Level-C, on average. Regarding dSVM [3] architecture, which also exploits joint intra-Tile and inter-Tiles data reuse, our Hy-SVM can achieve savings of up to 80% and 61%, for 4- and 8- Tiles respectively. The use of STT-RAM allows Hy-SVM the energy-efficient on-chip storage of entire reference frame samples. Hence, it strongly impacts in the external memory communication, since dSVM adopts intra-Tile data reuse in search window level. As we demonstrated in next section, the extra energy consumed by Hy-SVM larger on-chip SPMs is compensated by STT-RAM benefits and improved power management. AMBER [32] and enHyV [33] achieved increased off-chip energy savings when compared to Hy-SVM: up to 9% and 16% on average, respectively. AMBER fully exploits reference frame level data reuse, avoiding data re-fetching from external memory during a frame processing. However, to support Tiles-parallelized HEVC, AMBER requires the multiplication of its on-chip video memories, which strongly affects its on-chip energy efficiency (as discussed in next sections). enHyV implements data-reuse schemes in the same levels of Hy-SVM, without a proper management of the overlap formation. In the external memory perspective, enHyV provides a more complete support for inter-Tiles redundant accesses, leading to reduced SharedL2 misses. Still, Hy-SVM can achieve competitive off-chip energy results and, additionally, implements an overlap management that strongly reduces the SharedL2 SPMs on-chip energy.

Compared to our *Priv-Only* baseline implementation, the SharedL2 SPMs contributes by reducing up to 24% (4-Tile) and 35% (8-Tile) the external memory energy consumption, on average. Note that the achieved savings increase when more Tiles are used (higher parallelism), due to the well-exploited increased inter-Tiles data reuse potential by Hy-SVM. We demonstrate in the next sections that the on-chip energy required for SharedL2 SPMs is strongly reduced by our energy-efficient management schemes, resulting on savings when compared to *Priv-Only*.

B. *On-Chip Energy Results*

Fig. 19 shows the on-chip energy analysis of Hy-SVM compared to related works and baseline. In this evaluation, AMBER represents the upper bound case, since it replicates the storage of the reference frame for each Tile processing. In this case, Hy-SVM consumes 97% less on-chip energy than AMBER. The authors of AMBER did not informed the on-chip static energy savings lead by the proposed power-gating scheme. In doing so, it is not possible to perform a fair comparison with Hy-SVM. However, when comparing with the *No-APMU* baseline version of Hy-SVM, we can also note savings of 69%-83% (4-Tile) and 82-89%% (8-Tile), on average, when processing HD1080 and 2K videos, respectively. From enHyV perspective, Hy-SVM achieves on-chip energy savings from 66% (HD1080/8-Tile) up to 87% (2K/4-Tile). The gains over enHyV are mostly due to an improved power management, which relies on overlap management, as well as to enHyV extra SRAM memory to improve STT-RAM cells lifetime. When compared to dSVM, our Hy-SVM architecture presents similar on-chip energy consumption: 28% lower for 2K videos and 31% higher for HD1080 videos. Combining this with the previous off-chip energy results, Hy-SVM is able reduce the energy of external memory communication with similar on-chip energy consumption (demonstrated in next section).

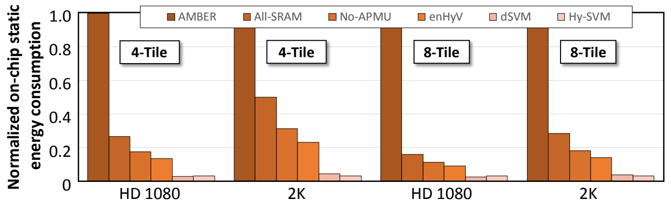


Fig. 19 – On-chip energy consumption of Hy-SVM compared to related works.

Compared to baseline All-SRAM and No-APMU implementations, our Hy-SVM can achieve up to 90% (4-Tile) and 83% (8-Tile) energy reduction. The savings related to All-SRAM are related to the STT-RAM low leakage power dissipation. The savings compared to No-APMU represent the efficiency of our power management over L2 level of SPMs.

C. *Overall Energy Results*

Table III presents the overall energy savings of Hy-SVM for 4-Tile and 8-Tile scenarios. The total energy is computed (fifth column of Table III) by the composition of off- and on-chip parts, including the control hardware that implements the management schemes.

Compared to Level-C scheme, even this presenting the smallest on-chip video memory (i.e. lower on-chip energy), Hy-SVM can reach overall energy savings of 69%-77%. These gains are mainly related to the reduction of 5.2 times in the external memory energy by exploiting inter-Tiles data reuse. Regarding the dSVM architecture, the multiple levels of SPMs allows Hy-SVM total memory energy savings of 43%-72%, for 4- and 8-Tile respectively. When comparing to AMBER, which achieves the best off-chip energy results, Hy-SVM reaches up to 94% of total energy savings. If we do not consider the APMU and the overlap management of HySVM, the No-APMU baseline version can, still, achieve overall energy reduction compared to AMBER (55% for 4-Tile; and 51% for 8-Tile scenarios). As previously discussed, enHyV also achieves lower external memory energy consumption than Hy-SVM. However, in an off- and on-chip combined perspective, Hy-SVM surpasses enHyV by achieving 58%-61% of improved energy efficiency. Finally, the proposed management layer, composed of overlap prediction, and on-chip MAMUs and AMPUs, can improve the energy efficiency of Hy-SVM by 64% and 58% in 8-Tile and 4-Tile parallel HEVC, respectively.

TABLE III OVERALL ENERGY SAVINGS OF HY-SVM COMPARED TO RELATED WORKS

Solution	**On-Chip Mem. [KB]**	**On-Chip Energy [mJ]**	**Off-Chip Energy [mJ]**	**Total Energy [mJ]**	**Savings Hy-SVM [%]**
Scenario 1: *4-Tile HD 1080, 128x128 search window*					
Level-C [9]	144	68	1,379	1,447	**77%**
dSVM [3]	519	107	1,072	1,179	**72%**
AMBER [32]	8,100	4,072	183	4,255	**79%*** **92%**
enHyV [33]	3,384	542	198	740	**55%**
Our No-APMU	2,544	711	198	908	**64%**
Our Hy-SVM	2,544	124	206	330	-
Scenario 2: *8-Tile HD 1080, 128x128 search window*					
Level-C [9]	288	137	1,360	1,497	**69%**
dSVM [3]	933	189	615	804	**43%**
AMBER [32]	16,200	8,144	183	8,327	**87%*** **94%**
enHyV [33]	3,892	739	193	932	**51%**
Our No-APMU	3,168	906	193	1,100	**58%**
Our Hy-SVM	3,168	252	208	461	-

**savings of No-APMU over AMBER*

D. *Overhead Analysis*

Fig. 20 presents an overhead analysis of the implemented management techniques in Hy-SVM. We utilize the *All-SRAM* and *Priv-Only* baseline implementations to discuss the overhead in terms of access latency (Fig. 20a) and dynamic energy consumption (Fig. 20b). Compared to *All-SRAM*, the overhead of inefficient STT-RAM write operations represents (on average) only 0.3% in terms of latency, and 0.8% in dynamic energy. Since PrivL1 SPMs have high hit rates (more than 95%), combined to low write intensity of video memories (as discussed in Section II.B), STT-RAM write penalty in Hy-SVM can be reduced. The comparison with *Priv-Only* version aims to evaluate the MAMU inserted overhead of analyzing the POTs to direct the incoming access either to PrivL2 or SharedL2 SPM. Still, the high L1 hit rates strongly reduces the overhead of overlap management, since it runs when L2 level access is required. Additionally, the hardware required for overlap management is composed of small tables and requires simple logic operations. As result, we can notice an overhead (on average) in the latency of only 4.7%, as well as 8.8% in the dynamic energy. Complementary, Fig. 20c shows the extra on-chip memory size required to implement the proposed energy-efficient management layer of Hy-SVM. In this analysis, we compute the size of monitored and predicted overlap tables (POTs and MOTs), as well as the frame-level and CTU-level power maps. As result, the overhead achieves only 4% in the worst-case scenario (16-Tile).

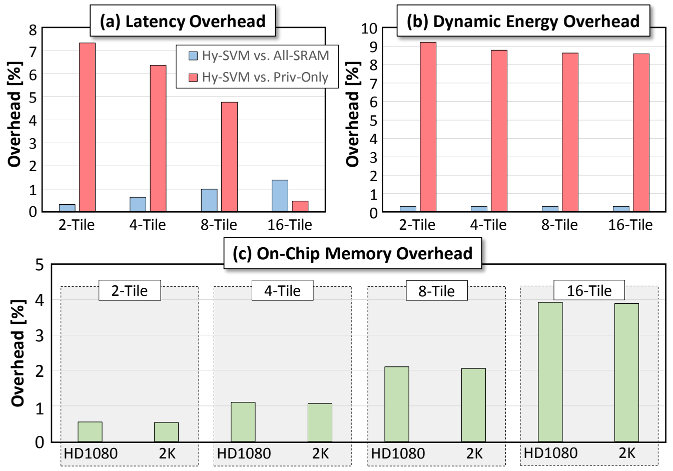


Fig. 20 – Overhead analysis in terms of: (a) latency, (b) dynamic energy and (c) extra on-chip memory size.

IX. Conclusions

This work presented a hybrid scratchpad video memory architecture for parallelized High-Efficiency Video Coding. It exploits opportunities from application-specific memory access behavior, combining scratchpad memories and hybrid memory design to improve the energy efficiency w.r.t. the memory infrastructure of parallel HEVC video encoders. We designed multiple levels of private and shared on-chip SPMs to fully exploit intra-Tile and inter-Tiles data reuse. For the SPMs implementation, we proposed a design methodology based on extracted application-specific properties. To improve the energy savings, we propose a memory management layer, which exploits the existing overlapping regions within the reference frames. In this context, our management layer is composed of the run-time overlap prediction scheme, as well as on-chip control units: memory access management and adaptive power management units. Our architecture provides from 51% up to 94% of energy savings compared to recent related works. The proposed Hy-SVM enables energy-efficient multimedia processing supporting parallel execution in state-of-the-art HEVC encoders.

References

[1] ISO/IEC-JCT1/SC29/WG11, “High Efficiency Video Coding (HEVC) text specification draft 10.” 2013.

[2] ITU-T, “ITU-T Recommendation H.264 (05/2003): advanced video coding for generic audiovisual services.” 2013.

[3] F. Sampaio, M. Shafique, B. Zatt, S. Bampi, and J. Henkel, “dSVM: Energy-efficient distributed Scratchpad Video Memory Architecture for the next-generation High Efficiency Video Coding,” in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, 2014, pp. 1–6.

[4] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[5] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou, “An Overview of Tiles in HEVC,” *IEEE J. Sel. Top. Signal Process.*, vol. 7, no. 6, pp. 969–977, Dec. 2013.

[6] M. Zhang, V. M. Stojanovic, and P. Ampadu, “Reliable Ultra-Low-Voltage Cache Design for Many-Core Systems,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 59, no. 12, pp. 858–862, Dec. 2012.

[7] A. Chakraborty, H. Homayoun, A. KHAJED, N. DUTT, A. ELTAWIL, and F. KURDAHI, “Multi-Copy Cache: A Highly Energy Efficient Cache Architecture,” *CECS UC Irvine Tech. Rep. CECS-TR-10-05*, 2010.

[8] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, “On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 61–72, Jan. 2002.

[9] C.-Y. Chen, C.-T. Huang, L.-G. Chen, and L.-G. Chen, “Level C+ data reuse scheme for motion estimation with corresponding coding orders,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 553–558, Apr. 2006.

[10] B. Zatt, M. Shafique, S. Bampi, and J. Henkel, “A low-power memory architecture with application-aware power management for motion amp; disparity estimation in Multiview Video Coding,” in *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2011, pp. 40–47.

[11] B. Zatt, M. Shafique, F. Sampaio, L. Agostini, S. Bampi, and J. Henkel, “Run-time adaptive energy-aware Motion and Disparity Estimation in Multiview Video Coding,” in *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2011, pp. 1026–1031.

[12] F. Sampaio, B. Zatt, M. Shafique, L. Agostini, S. Bampi, and J. Henkel, “Energy-efficient memory hierarchy for Motion and Disparity Estimation in Multiview Video Coding,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, 2013, pp. 665–670.

[13] M. Shafique, B. Zatt, F. L. Walter, S. Bampi, and J. Henkel, “Adaptive power management of on-chip video memory for Multiview Video Coding,” in *2012 49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2012, pp. 866–875.

[14] R. Banakar, S. Steinke, B.-S. Lee, M. Balakrishnan, and P. Marwedel, “Scratchpad memory: a design alternative for cache on-chip memory in embedded systems,” in *Proceedings of the Tenth International Symposium on Hardware/Software Codesign, 2002. CODES 2002*, 2002, pp. 73–78.

[15] Texas Instruments, “TMS370CX7X from Texas Instruments,” 2017. [Online]. Available: http://www.ti.com/mcu/docs/mcuorphan.tsp?contentId=15364. [Accessed: 29-Jul-2015].

[16] IBM Research, “The Cell Project - IBM,” 22-Mar-2013. [Online]. Available: http://researcher.watson.ibm.com/researcher/view_group.php?id=2649. [Accessed: 29-Jul-2015].

[17] D. Niu, C. Xu, N. Muralimanohar, N. P. Jouppi, and Y. Xie, “Design trade-offs for high density cross-point resistive memory,” in *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, 2012, pp. 209–214.

[18] N. Khoshavi, X. Chen, J. Wang, and R. F. DeMara, “Read-Tuned STT-RAM and eDRAM Cache Hierarchies for Throughput and Energy Enhancement,” *ArXiv160708086 Cs*, Jul. 2016.

[19] J. Ahn, S. Yoo, and K. Choi, “Prediction Hybrid Cache: An Energy-Efficient STT-RAM Cache Architecture,” *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 940–951, Mar. 2016.

[20] H. Kim, S. Kim, and J. Lee, “Write-Amount-Aware Management Policies for STT-RAM Caches,” *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 25, no. 4, pp. 1588–1592, Apr. 2017.

[21] M. Lapedus, “Semiconductor Engineering .:. Four Foundries Back MRAM,” 23-Aug-2017. [Online]. Available: https://semiengineering.com/four-foundries-back-mram/. [Accessed: 29-Jan-2018].

[22] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen, “Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) As a Universal Memory Replacement,” New York, NY, USA, 2008, pp. 554–559.

[23] C. Blumenberg, D. Palomino, S. Bampi, and B. Zatt, “Adaptive content-based Tile partitioning algorithm for the HEVC standard,” in *Picture Coding Symposium (PCS), 2013*, 2013, pp. 185–188.

[24] X. Jin and Q. Dai, “Clustering-Based Content Adaptive Tiles Under On-chip Memory Constraints,” *IEEE Trans. Multimed.*, vol. 18, no. 12, pp. 2331–2344, Dec. 2016.

[25] C. C. Chi *et al.*, “Parallel Scalability and Efficiency of HEVC Parallelization Approaches,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1827–1838, Dec. 2012.

[26] M. Shafique, M. U. K. Khan, and J. Henkel, “Power efficient and workload balanced tiling for parallelized high efficiency video coding,” in *2014*

IEEE International Conference on Image Processing (ICIP), 2014, pp. 1253–1257.

[27] M. U. K. Khan, M. Shafique, and J. Henkel, "Software architecture of High Efficiency Video Coding for many-core systems with power-efficient workload balancing," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2014, pp. 1–6.

[28] L. Guo, D. Zhou, and S. Goto, "A New Reference Frame Recompression Algorithm and Its VLSI Architecture for UHDTV Video Codec," *IEEE Trans. Multimed.*, vol. 16, no. 8, pp. 2323–2332, Dec. 2014.

[29] J. Zhu, L. Guo, D. Zhou, S. Kimura, and S. Goto, "An independent bandwidth reduction device for HEVC VLSI video system," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 609–612.

[30] X. Lian, Z. Liu, W. Zhou, and Z. Duan, "Parallel Content-Aware Adaptive Quantization Oriented Lossy Frame Memory Recompression for HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. PP, no. 99, pp. 1–1, 2016.

[31] C. Song, L. Ju, and Z. Jia, "Hybrid scratchpad and cache memory management for energy-efficient parallel HEVC encoding," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*, 2015, pp. 712–719.

[32] M. U. K. Khan, M. Shafique, and J. Henkel, "AMBER: Adaptive energy management for on-chip hybrid video memories," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 405–412.

[33] F. Sampaio, M. Shafique, B. Zatt, S. Bampi, and J. Henkel, "Energy-efficient architecture for advanced video memory," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2014, pp. 132–139.

[34] HP Labs, "HP Labs : CACTI," 2008. [Online]. Available: http://www.hpl.hp.com/research/cacti/. [Accessed: 06-Nov-2017].

[35] Xiangyu Dong, Cong Xu, Yuan Xie, and N. P. Jouppi, "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.

[36] H. Singh, K. Agarwal, D. Sylvester, and K. J. Nowka, "Enhanced Leakage Reduction Techniques Using Intermediate Strength Power Gating," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 15, no. 11, pp. 1215–1224, Nov. 2007.

[37] Micron Technology Inc., "4Gb: x16, x32 Mobile LPDDR2 SDRAM S4." Micron Technology Inc., 2011.

[38] Micron Technology Inc., "TN-46-12: Mobile DRAM Power-Saving Features/Calculations." Micron Technology Inc., 2005.

[39] Micron Technology Inc., "TN-46-03 – Calculating DDR Memory System Power." Micron Technology Inc., 2001.

[40] ISO/IEC-JCT1/SC29/WG11, "Common test conditions and software reference configurations." 2012.

[41] ISO/IEC-JCT1/SC29/WG11, "High Efficiency Video Coding (HEVC) Test Model 13 (HM 13) Encoder Description." 2013.