UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

PAULO RICARDO RAMOS DA ROSA

Análise da razão entre precisão, área e consumo de potência das unidades multiplicadoras do vetor espacial do Gemmini

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Antonio Carlos Schneider Beck Filho

Porto Alegre
2024

RESUMO

A Inteligência Artificial nunca esteve tão popular quanto neste momento. Seu uso adentra diversos setores da sociedade, desde recomendações de busca em sites até carros autônomos e criação de entretenimento. Dentre os diversos recursos no universo de IA estão as Redes Neurais, uma estrutura de dados que simula o comportamento dos neurônios do corpo humano, muito utilizadas para reconhecimento de padrões e classificação de dados. Dado que as redes neurais são estruturas complexas e custosas em termos de hardware, estruturas para melhorar seu desempenho foram criadas, como o Gemmini.

O Gemmini é um gerador de aceleradores de redes neurais. Nele há um estrutura chamada vetor sistólico, que é um vetor bidimensional formado por unidades MAC (multiplica-acumula) cuja função é acelerar operações convolucionais.

Com o objetivo de buscar ganhos de área e potência no vetor sistólico do Gemmini, este trabalho irá alterar o vetor sistólico do Gemmini utilizando multiplicadores aproximativos, que são multiplicadores menos custosos em termos de hardware comparados a um multiplicador regular mas que apresentam um nível de perda na precisão do resultado, e analisar os efeitos dessa troca quanto a ganhos de área e potência e sobre a qualidade da inferência da rede neural, ou seja, o quanto a rede acerta sua predição. Ao final do trabalho, conclui-se que os vetores sistólicos testados com os melhores resultados obtiveram ganhos de potência de 3% com um aumento de área de 5% e perda de precisão de 1% ou ganho de potência de 17%, ganho de área de 4% e perda de precisão de 16,25%.

Palavras-chave: Gemmini. Multiplicadores aproximativos. Área. Potência. Inferência. Redes neurais.

ABSTRACT

Artificial Intelligence has never been as popular as it is today. Its use reaches many sectors of society, from website search recommendations to self-driving cars and even creation of entertainment. Between all the resources in the AI universe are the Neural Networks, a data structure that simulates the behaviour of the human body's neurons, very useful for recognizing patterns and data classification. Since neural networks are structures very complex and demanding in terms of hardware, structures to improve its performance were developed, like Gemmini.

Gemmini is a neural network accelerator generator. In Gemmini there is a structure called the systolic array, a bidimensional array made with MAC (multiply-accumulate) units, which its function is to accelerate convolutional operations.

With the objective of obtaining gains of area and power in Gemmini's systolic array, this assignment will change Gemmini's spatial array and use approximate multipliers, which are multipliers less expensive in terms of hardware compared to a regular multiplier but show a precision loss in its results, and analyze the effects of this replacement as for gains in area and power and about its inference quality, which is how much the network hits the prediction. At the end of this assignment, it concludes that the tested systolic arrays with the best results had a gains of 3% in power with an increase of 5% in area and precision loss around 1% or gain of 17% of power, gain of 4% in area and precision loss around 16,25%.

Keywords: Gemmini. Approximate Multipliers. Area. Power. Inference. Neural Networks.

LISTA DE FIGURAS

Figura 1.1 Exemplo de rede neural multicamadas 10
Figura 1.2 Funcionamento de um nodo de uma rede neural 11
Figura 1.3 Convolução de uma matriz 7x7 com um kernel 3x3 12
Figura 1.4 Ideia básica do elemento processador 13
Figura 1.5 Elemento processador *weight stationary* 14
Figura 1.6 Vetor sistólico 2D 14

Figura 2.1 Estrutura do Gemmini 15
Figura 2.2 Vetor sistólico 16

Figura 3.1 Exemplo de tabela verdade de um Meio Somador aproximativo trocando a porta XOR por uma porta OR 18
Figura 3.2 Gráficos comparativos de desempenho dos multiplicadores de 16 bits com sinal da EvoApproxLib com implementações precisas 20

Figura 4.1 dog.jpg 23
Figura 4.2 cat.jpg 23
Figura 4.3 ball.jpg 24
Figura 4.4 fish.jpeg 25
Figura 4.5 Somador completo de 1 bit 27
Figura 4.6 Mux 2:1 de 1 bit 28
Figura 4.7 Registrador de 1 bit 28
Figura 4.8 Flip-flop tipo D 29

LISTA DE TABELAS

Tabela 4.1 Multiplicadores utilizados para os testes. 21
Tabela 4.2 Resultados das execuções da rede mobilenet com multiplicadores aproximativos [Valor (Pontuação)] 21
Tabela 4.3 Resultados das execuções da rede resnet50 com multiplicadores aproximativos [Valor (Pontuação)] 22
Tabela 4.4 Predição googlenet_quantized para a imagem dog.jpg 22
Tabela 4.5 Predição googlenet_quantized para a imagem cat.jpg 24
Tabela 4.6 Predição googlenet_quantized para a imagem ball.jpg 25
Tabela 4.7 Predição googlenet_quantized para a imagem fish.jpeg 25
Tabela 4.8 Precisão da rede GoogLeNet_quantized utilizando diferentes multiplicadores aproximativos. 26
Tabela 4.9 Cálculos de potência e área dos vetores sistólicos. 29

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
ML	Machine Learning
MAC	Multiply-accumulate
GPU	Graphics Processing Unit
FPGA	Field-Programmable Gate Array
TPU	Tensor Processing Unit
NNP	Neural Network Processor
CPU	Central Processing Unit
SoC	System-on-chip
RoCC	Rocket Custom Co-processor
PE	Processing element
WS	Weight Stationary
OS	Output Stationary
NHWC	Number of samples, Height, Width, Channels
ONNX	Open Neural Network Exchange
CGP	Cartesian Genetic Programming
MAE	Mean Absolute Error
WCE	Worst-Case Absolute Error
MRE	Mean Relative Error
EP	Error Probability
CMOS	Complementary Metal Oxide Semiconductor

SUMÁRIO

1 INTRODUÇÃO 9
1.1 Sobre aprendizado de máquina 9
1.2 Sobre redes neurais 10
1.3 Aceleradores de IA 11
1.4 Vetor sistólico 12
2 O GEMMINI 15
2.1 Estrutura do Gemmini 15
2.2 Softwares para o Gemmini 16
3 MULTIPLICADORES APROXIMATIVOS 18
3.1 A biblioteca EvoApproxLib 19
4 TESTES E RESULTADOS 21
4.1 Valores obtidos - mobilenet e resnet50 21
4.2 Valores obtidos - googlenet_quantized ONNX 22
4.3 Taxa de precisão dos multiplicadores aproximativos na rede GoogLenet_quantized 26
4.4 Razão Potência x Área 26
5 CONCLUSÕES 30
REFERÊNCIAS 31

1 INTRODUÇÃO

A definição de Inteligência Artificial conhecida nos dias de hoje começou a ser moldada já nos primórdios da Ciência da Computação. Alan Turing, no artigo *Computing Machinery and Intelligence*, de 1950, propôs a pergunta: "Máquinas podem pensar?". Este artigo veio a definir o que hoje é conhecido como *Teste de Turing*, que define se uma máquina pode ou não ser considerada inteligente. Em 1955, é realizado o *Dartmouth Summer Research Project on Artificial Intelligence*, considerado o evento que lançou o termo *Artificial Intelligence*, bem como a Inteligência Artificial como campo de pesquisa. Foi nessa época que John McCarthy definiu a Inteligência Artificial como sendo "a ciência e engenharia de criar máquinas inteligentes". De uma forma mais geral, ela combina a Ciência da Computação com bases de dados robustas para resolver problemas. Esses problemas foram evoluindo ao longo dos anos e gerando subcategorias de estudos como, por exemplo, tomadas de decisões, processamento de linguagens naturais, percepção de dados, interpretação de sentimentos, representação de conhecimento e aprendizado de máquina.

1.1 Sobre aprendizado de máquina

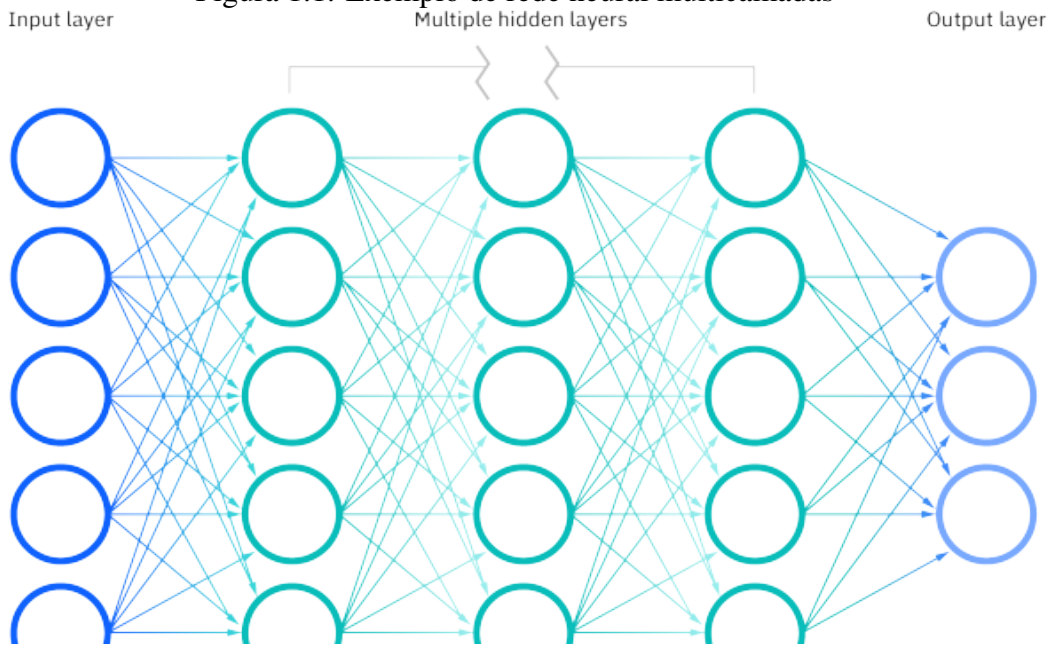
Termo popularizado por Arthur Samuel em 1959, o aprendizado de máquina (Machine Learning - ML) é definido por ele como "um campo de estudo que dá às máquinas a capacidade de aprender sem serem programadas de maneira explícita". De maneira simplificada, consiste em treinar a máquina para tomar decisões a partir de conhecimentos previamente adquiridos. O conhecimento é implementado como uma base de dados, que existe de diversos tipos, e a tomada de decisão é realizada por um algoritmo que, assim como a base de dados, varia para cada tipo de tarefa desejada. O algoritmo busca padrões na base de dados para então realizar predições de novos dados. Diversas aplicações utilizam ML, como filtragem de e-mails, sistemas de recomendações, reconhecimento de voz, visão computacional, modelos de linguagem, medicina, entre outros.

Dentre o ramo de aprendizado de máquina, diversos modelos estatísticos foram desenvolvidos e estudados até hoje, com as redes neurais sendo um dos mais importantes da área.

1.2 Sobre redes neurais

Redes neurais são um mecanismo de processamento de dados utilizado em Inteligência Artificial que simula o comportamento dos neurônios do cérebro humano. São redes compostas de nodos ligados em camadas que, a partir de certos valores de pesos, realizam operações para chegar a uma predição. Esses pesos e valores vão se modificando ao longo da execução, de maneira que a rede tenta compensar por seus erros, daí gerando sua característica de aprendizado. Geralmente, as redes neurais costumam ter uma camada inicial de entrada, uma camada final de saída e diversas camadas intermediárias chamadas de camadas ocultas, que são as camadas onde acontecerão as operações de classificação (AMAZON, 2023).

Figura 1.1: Exemplo de rede neural multicamadas

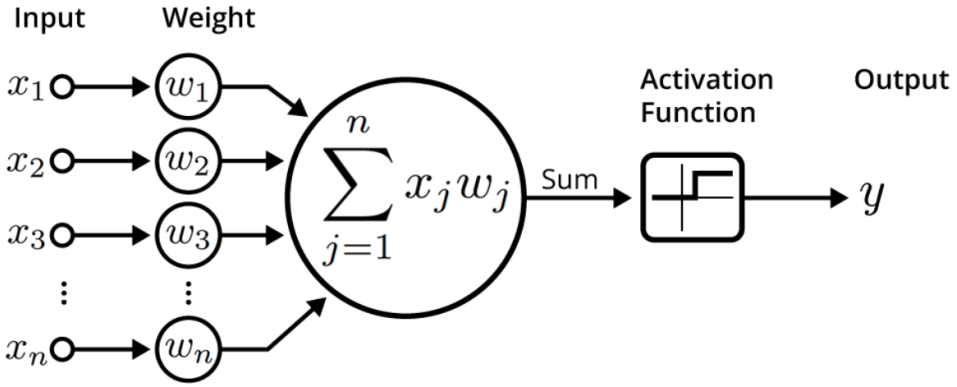


Fonte: (IBM, 2023)

Cada nodo da rede recebe um valor de entrada, realiza um cálculo e passa o resultado uma camada adiante. Esse valor de entrada é multiplicado por um valor de peso que vai determinar a relevância dessa entrada para o valor de saída. Os produtos das entradas são somados e então incluídos em uma função de ativação que irá gerar o valor de saída do nodo. A função de ativação (também chamada de função threshold) define se o valor obtido irá ou não ativar o neurônio, gerando um resultado através de limites bem definidos. Funções como sigmoide, retificação linear e tangente hiperbólica são exemplos de funções de ativação. O aprendizado da rede ocorre ajustando os pesos de cada nodo com

o método de *backpropagation*, que calcula o gradiente da função de erro da rede e vai compensando os pesos dos nodos de modo reverso camada por camada, o que caracteriza o seu nome.

Figura 1.2: Funcionamento de um nodo de uma rede neural



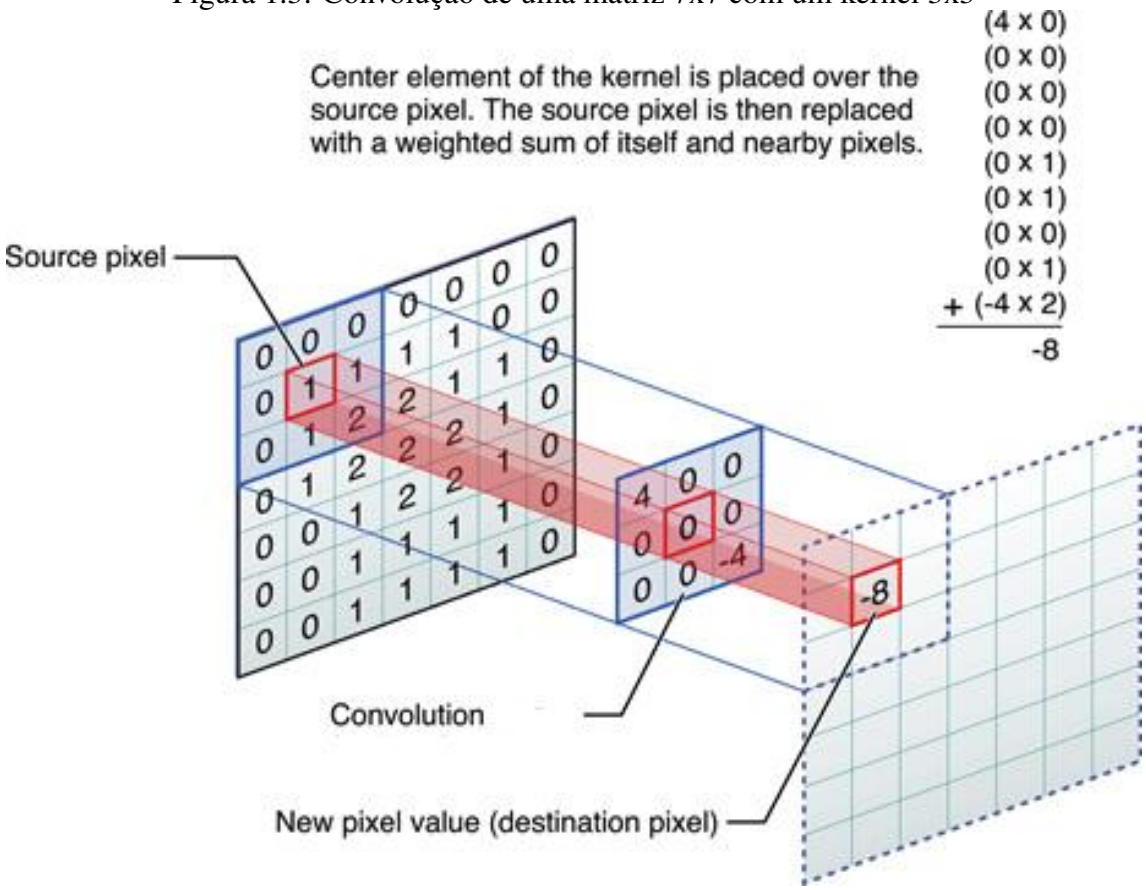
Fonte: (MCCULLUM, 2020)

Existem diversos tipos de classificações de redes neurais, usualmente ligadas ao fluxo de dados e/ou sua especificidade. Um tipo de rede muito utilizada para reconhecimento e classificação de imagens é a rede convolucional, onde as camadas ocultas realizam operações de convolução para, por exemplo, classificar os pixels de uma imagem e determinar o conteúdo dela. A convolução é uma operação que realiza diversas operações de multiplicação e soma (MAC), e se tratando de operações com alto potencial de paralelização e que são executadas em alto número, circuitos que otimizam essas operações foram desenvolvidos: os aceleradores de IA.

1.3 Aceleradores de IA

O processamento das redes neurais demanda alto uso de recursos de hardware, portanto circuitos especializados em trabalhar com IA e Deep Learning foram criados e são constantemente otimizados. Com a necessidade de portar essas redes para uso em, por exemplo, dispositivos embarcados e portáteis, onde o uso de bateria e economia de energia é crucial, ou de tempo real, onde o tempo de resposta precisa ser o mais otimizado possível, estes circuitos especializadas em realizar determinadas operações de maneira eficiente tornam-se essenciais. Essa é a ideia básica do acelerador de IA: são circuitos que otimizam a carga de trabalho para um processamento mais veloz. Alguns exemplos de Aceleradores de IA são: GPUs (desenvolvidas originalmente para processar gráficos,

Figura 1.3: Convolução de uma matriz 7x7 com um kernel 3x3



Fonte: (BASAVARAJAIAH, 2019)

mas podem ser usadas para processamento de IA devido seu alto grau de paralelismo), FPGAs (devido sua flexibilidade para criar curcuitos especializados), TPUs (circuitos integrados desenvolvidos pela Google para otimizar inferências de rede), CPUs (modelos modernos de CPU possuem aceleradores de IA para algumas tarefas) e NNPs (construídos especificamente para Deep Learning, são otimizados para operações com matrizes). NNPs se beneficiam muito de estruturas otimizadas para realizar operações MAC, chamados vetores sistólicos.

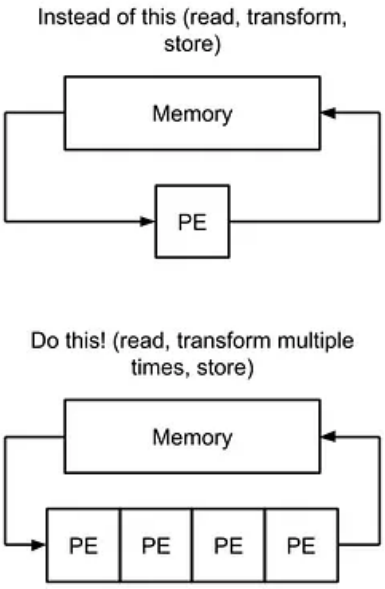
1.4 Vetor sistólico

Apesar de um protótipo de vetor sistólico ter sido utilizado no computador Colossus durante a Segunda Guerra Mundial, os vetores sistólicos foram independentemente introduzidos em 1979 por H. T. Kung (KUNG, 1982) e são estruturas desenvolvidas para paralelizar operações reduzindo a quantidade de ciclos necessários para resolvê-las. São formados por pequenos processadores simples chamados Elementos Processadores (Pro-

cessing Element - PE) que realizam uma específica operação. Tais elementos são interligados em série de forma linear ou bidimensional dependendo da finalidade do sistema. Os PEs também possuem registradores na saída para armazenar valores intermediários das operações tal como um sistema de pipeline, assim reduzindo o número de acessos à memória principal. Além de paralelismo e pipeline, o vetor sistólico tem como características o sincronismo e repetitividade (devido à construção simétrica do vetor sistólico, o número de ciclos necessários para concluir o percurso do vetor é bem definido), modularidade e localidade espacial e temporal. Como exemplos de aplicações que utilizam vetores sistólicos estão funções hash, compressão de dados, reconhecimento de objetos, processamento de imagens, reconhecimento de fala e redes convolucionais.

O PE pode ser construído de diversas maneiras de acordo com a aplicação que for utilizado, pois vetores sistólicos são estruturas caras de se fabricar e inflexíveis. Um modelo bem comum de PE é o *weight stationary*, que carrega os pesos dos nodos antes de realizar as operações matemáticas. Ao interligar esses elementos em duas dimensões, forma-se o vetor sistólico 2D, utilizado em aceleradores de redes neurais para realizar operações MAC.

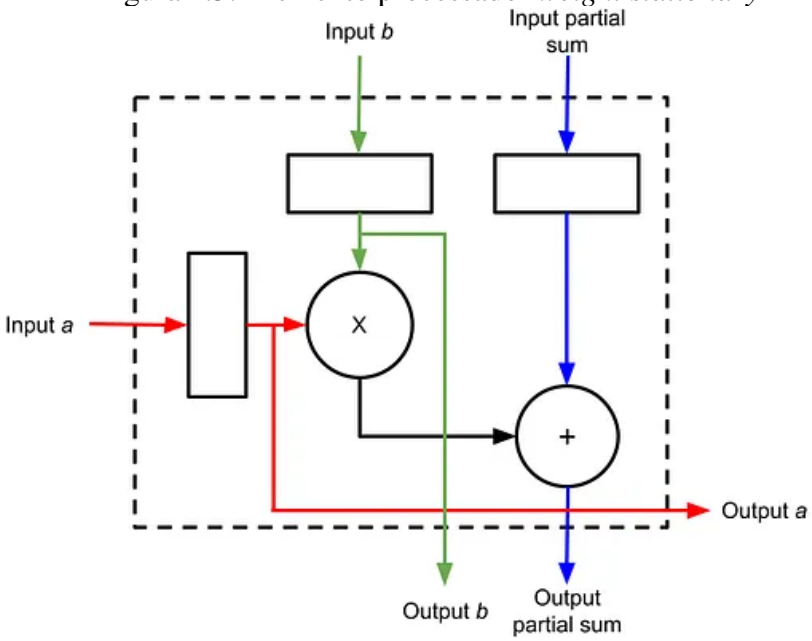
Figura 1.4: Ideia básica do elemento processador



Fonte: (PECCIA, 2022)

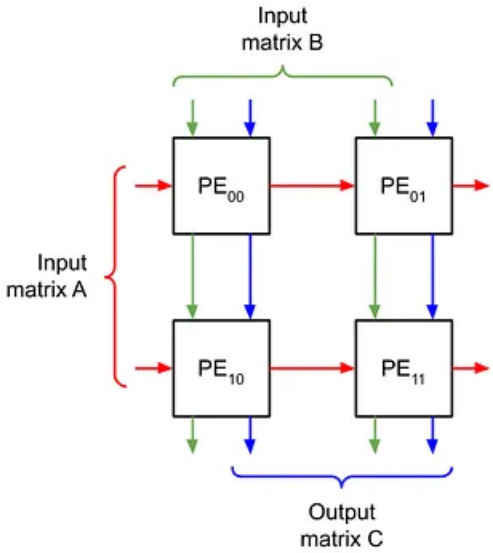
Conforme constatado, criar aceleradores de IA são custosos. Portanto uma ferramenta que permita gerar, testar e avaliar aceleradores é de suma importância para estudar e otimizar o funcionamento deles.

Figura 1.5: Elemento processador *weight stationary*



Fonte: (PECCIA, 2022)

Figura 1.6: Vetor sistólico 2D



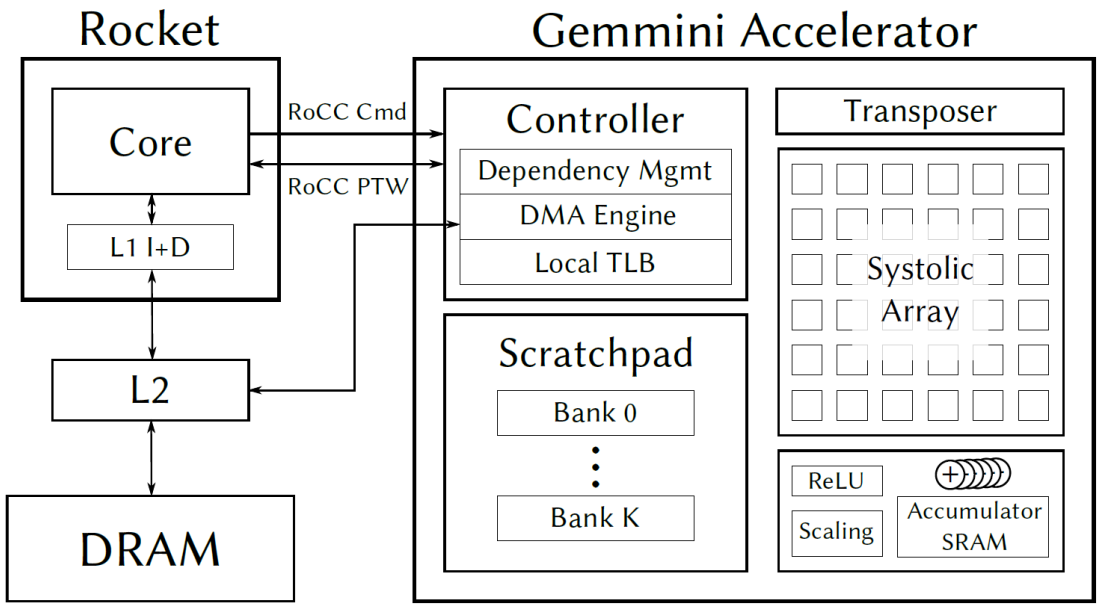
Fonte: (PECCIA, 2022)

2 O GEMMINI

O Gemmini (GENC et al., 2021) é um gerador de aceleradores de redes neurais full-stack altamente customizável. Nele é possível alterar diversas configurações de hardware e software para avaliar o desempenho da rede neural. O projeto Gemmini faz parte do framework Chipyard para desenvolvimento de SoCs baseados no processador RISC-V.

2.1 Estrutura do Gemmini

Figura 2.1: Estrutura do Gemmini



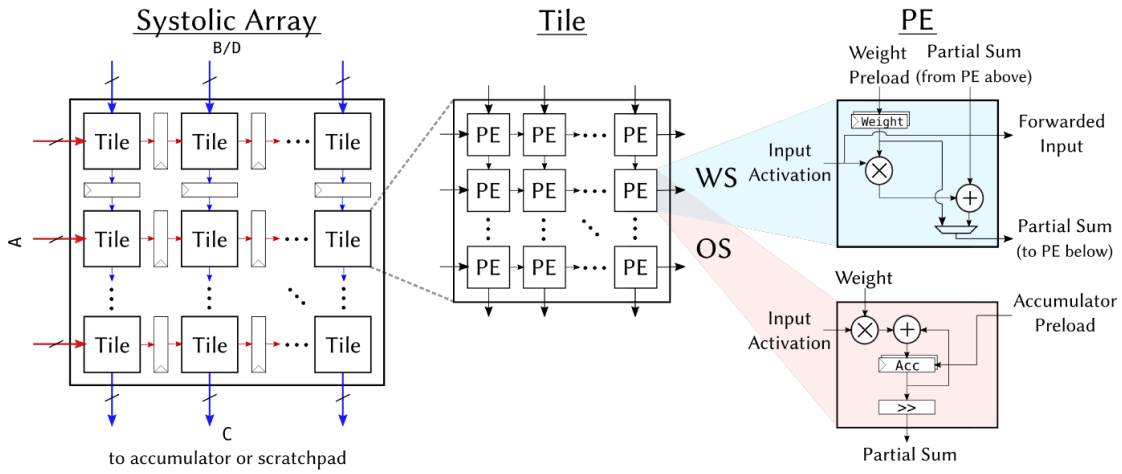
Fonte: (RESEARCH, 2021)

A estrutura do Gemmini pode ser dividida nas seguintes grandes seções:

- *Controller*: Unidade que faz o controle do fluxo do programa. Possui 3 subtipos de controle:
  - *LoadController*: Carrega os dados da memória principal para o Gemmini;
  - *ExecuteController*: Responsável por instruções de execução, como multiplicação de matrizes;
  - *StoreController*: Entrega os dados do Gemmini para a memória principal.
- *Scratchpad*: Onde fica guardadas as entradas e saídas do vetor sistólico.
- *Vetor Sistólico*: Um conjunto de unidades chamadas *Tiles* ligadas de maneira cru-

zada com registradores de pipeline entre elas. Dentro de cada Tile, existe um conjunto de elementos processadores que calculam paralelamente resultados parciais de uma operação e passam seu resultado adiante para a próxima unidade. No caso do Gemmini, os elementos processadores são unidades MAC utilizados para operações como multiplicação de matrizes e convoluções, por exemplo. O vetor sistólico realiza essas operações de maneiras distintas dependendo se a operação for com (Weight Stationary) ou sem peso (Output Stationary) do nodo da rede.

Figura 2.2: Vetor sistólico



Fonte: (RESEARCH, 2021)

2.2 Softwares para o Gemmini

Os programas que irão rodar no Gemmini devem incluir as bibliotecas do acelerador e suas funções devem ser explicitamente utilizadas, portanto cabe ao programador utilizá-las da maneira mais apropriada. Diversas funções estão disponíveis, mas existem 3 que são essenciais:

- *gemmini_mvin*: Função que carrega os dados da memória principal para o scratchpad do Gemmini;
- *gemmini_compute*: Função que utiliza o vetor sistólico para realizar as operações de multiplicação e soma com ou sem peso;
- *gemmini_mvout*: Função que devolve os dados do scratchpad para a memória principal.

O Gemmini possui vários programas inclusos no seu repositório, mas para a realização dos testes foram utilizadas as redes *resnet50* e *mobilenet*, que realizam uma série de convoluções para prever um conjunto de valores alvo. Também foi utilizada a rede *GoogLeNet* em versão quantificada para realizar a predição de algumas imagens.

3 MULTIPLICADORES APROXIMATIVOS

Circuitos aproximativos são circuitos que, ao utilizar componentes menos custosos em termos de hardware, geralmente apresentam uma queda na qualidade do resultado, porém com um ganho na área e no consumo de potência. Podem ser utilizados em sistemas que demandam resposta rápida sem se importar com pequenas perdas de valores, como processadores de imagem para embarcados, por exemplo. O relatório da EvoApproxLib (MRAZEK et al., 2017) menciona os seguintes princípios nos quais multiplicadores aproximativos são baseados: aproximação para encontrar produtos parciais utilizando uma estrutura mais simples para gerar produtos parciais; aproximação na árvore de produtos parciais ignorando alguns produtos parciais ou dividindo produtos parciais em diversos módulos e aplicando aproximação nos módulos menos significantes; usando contadores aproximados ou compressores na árvore do produto parcial; usando multiplicadores Booth aproximados ou compondo multiplicadores aproximativos complexos por meio de multiplicadores simples.

Figura 3.1: Exemplo de tabela verdade de um Meio Somador aproximativo trocando a porta XOR por uma porta OR.

TRUTH TABLE OF APPROXIMATE HALF ADDER

Inputs		Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	$Carry$	Sum	$Carry$	Sum	
0	0	0	0	0 ✔	0 ✔	0
0	1	0	1	0 ✔	1 ✔	0
1	0	0	1	0 ✔	1 ✔	0
1	1	1	0	1 ✔	1 ✗	1

Fonte: (VENKATACHALAM; KO, 2017)

O estudo *ConfAx: Exploiting Approximate Computing for Configurable FPGA CNN Acceleration at the Edge* (KOROL et al., 2022) demonstrou que utilizando unidades aproximativas foi possível obter um ganho de potência de até 1.65 vezes e um ganho de energia de até 1.44 vezes com perda de precisão menor que 1% na média. Já o estudo *Low-power approximate MAC unit* (ESPOSITO; STROLLO; ALIOTO, 2017) propôs uma unidade MAC convolucional aproximativa com ganho de potência de mais de 60% comparado com uma unidade MAC precisa e com perdas aceitáveis de qualidade nas imagens. Também há o estudo *Energy-Efficient Approximate MAC Unit* (ADAMS;

VENKATACHALAM; KO, 2019), que propõe uma arquitetura de unidade MAC que, comparado com uma unidade precisa, reduz a área em 67% e o consumo de potência em 49%.

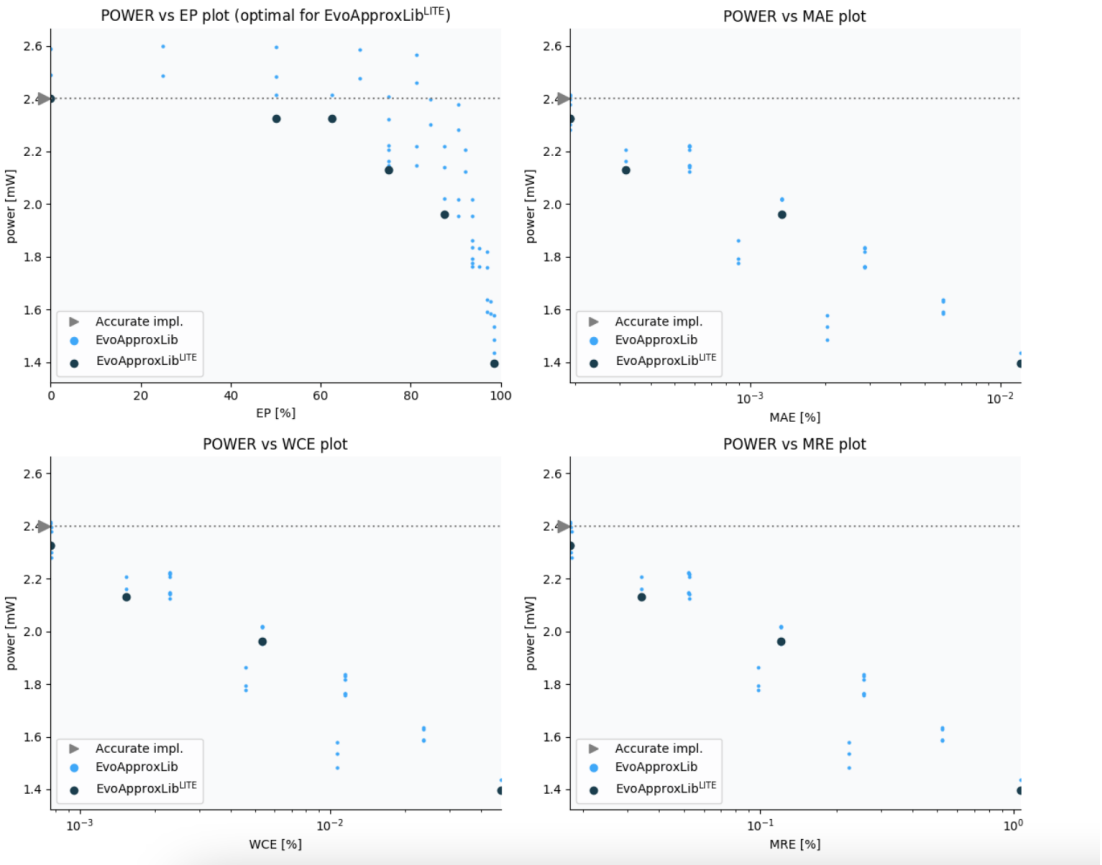
3.1 A biblioteca EvoApproxLib

Para realização dos testes foi utilizada a biblioteca de multiplicadores aproximativos *EvoApproxLib* (MRAZEK et al., 2017), que possui tanto modelos aproximativos em software quanto em hardware. A biblioteca já possui documentada dados como:

- *MAE*: Erro médio Absoluto;
- *WCE*: Erro absoluto de Pior Caso;
- *MRE*: Erro médio Relativo;
- *EP*: Probabilidade de Erro;
- *Power*: Consumo de Potência em mW;
- *Area*: Área ocupada no chip em um^2.

A EvoApproxLib é uma evolução da EvoApproxLib8b, que é uma biblioteca de somadores e multiplicadores aproximativos de 8 bits, adicionando somadores e multiplicadores de 12 e 16 bits com e sem sinal. As implementações foram criadas utilizando a técnica de Programação Genética Cartesiana (CGP), uma forma de Programação Genética que utiliza grafos para representar estruturas computacionais, e recebe o termo "Cartesiano"pois os programas são representados por grades bidimensionais de nodos. A validação dos circuitos foi realizada com a suite de benchmarks *AxBench* e a tecnologia dos circuitos é TSMC de 180 nanômetros.

Figura 3.2: Gráficos comparativos de desempenho dos multiplicadores de 16 bits com sinal da EvoApproxLib com implementações precisas



Fonte: (TECHNOLOGY, 2017)

4 TESTES E RESULTADOS

Os testes foram feitos utilizando redes neurais implementadas em C e em modelo ONNX. As redes *resnet50* e *mobilenet* possuem uma implementação em C e a rede *GoogLeNet* em versão quantificada é utilizada como modelo ONNX. Na implementação do Gemmini das redes resnet50 e mobilenet o objetivo é acertar um conjunto de quatro valores. Além disso, também é calculdada a pontuação de cada valor. Quanto maior o score, mais a rede acredita estar correta. Já a rede ONNX GoogLeNet quantificada realiza a predição de uma imagem JPG de, no máximo, 224x224 pixels e mostra os 5 resultados com as maiores probabilidades de imagem. A versão quantificada da rede GoogLeNet foi utilizada por questões de compatibilidade com os multiplicadores aproximativos e por ser executada de maneira mais rápida sacrificando pouco de sua precisão: com as amostras de imagens testadas a versão completa da GoogLeNet obteve uma taxa de precisão Top-1 de 87,84% enquanto a versão quantificada teve precisão Top-1 de 76,22%. A configuração de hardware *baselineInferenceConfig* do Gemmini foi utilizada em todos os testes. Essa configuração possui um vetor sistólico de dimensões 16x16 com 1 nível de hierarquia.

Os multiplicadores utilizados foram:

Tabela 4.1: Multiplicadores utilizados para os testes.

	MAE (%)	WCE (%)	MRE (%)	EP (%)	power (mW)	area (um^2)
HG4	0.00	0.00	0.00	0.00	2.400	2614.0
GQU	0.00019	0.00076	0.018	50.00	2.326	2764.2
GQV	0.00019	0.00076	0.018	62.50	2.325	2760.4
HF7	0.00032	0.0015	0.034	75.00	2.130	2576.5
GSM	0.0013	0.0053	0.12	87.50	1.961	2495.7
GV3	0.012	0.048	1.06	98.44	1.396	1932.6

4.1 Valores obtidos - mobilenet e resnet50

Os resultados obtidos para as redes mobilenet e resnet50 foram:

Tabela 4.2: Resultados das execuções da rede mobilenet com multiplicadores aproximativos [Valor (Pontuação)]

	Valor base	HG4	GQU	GQV	HF7	GSM	GV3
Predição 1	75 (127)	75 (127)	76 (69)	65 (53)	327 (127)	456 (-47)	0 (-128)
Predição 2	900 (127)	900 (127)	900 (112)	327 (37)	34 (127)	456 (-33)	0 (-128)
Predição 3	125 (103)	125 (103)	116 (96)	62 (55)	327 (127)	733 (-51)	0 (-128)
Predição 4	897 (98)	897 (98)	897 (60)	79 (34)	34 (127)	327 (-28)	0 (-128)

Na implementação da mobilenet, os valores esperados são 75, 900, 125 e 897. O multiplicador HG4, como esperado, obteve um resultado idêntico ao valor base, já que ele não possui perdas. Já o multiplicador GQU obteve êxito nas predições 2 e 4 e nas predições 1 e 3 teve uma taxa de erro de , respectivamente, 1,33% e 7,2%, no total com uma taxa de erro média de 4,26%. O multiplicador GQV obteve resultados mais disparidos dos esperados, chegando a uma taxa de erro média de 54,64%. Todavia, os multiplicadores HF7, GSM e GV3 tiveram péssimos resultados, com as respectivas taxas de erro média de 172,5%, 276,8% e 100%.

Tabela 4.3: Resultados das execuções da rede resnet50 com multiplicadores aproximativos [Valor (Pontuação)]

	Valor base	HG4	GQU	GQV	HF7	GSM	GV3
Predição 1	75 (45)	75 (45)	75 (48)	75 (44)	36 (1)	490 (37)	0 (0)
Predição 2	900 (43)	900 (43)	900 (47)	900 (69)	0 (1)	489 (16)	0 (0)
Predição 3	641 (40)	641 (40)	641 (39)	63 (30)	36 (1)	490 (30)	0 (0)
Predição 4	897 (57)	897 (57)	897 (58)	897 (57)	363 (1)	722 (25)	0 (0)

Os valores esperados na rede resnet50 são 75, 900 641 e 897. Novamente, o multiplicador HG4 não possui erros. Aqui, o multiplicador GQU também obteve erro zero, somente com pequenas diferenças na pontuação. O multiplicador GQV teve bom desempenho nessa rede, com uma taxa de erro média de 22,54%. Os multiplicadores HF7, GSM e GV3 tem taxas de erro média de, respectivamente, 76,47%, 160,4% e 100%.

4.2 Valores obtidos - googlenet_quantized ONNX

Os testes da rede googlenet_quantized foram realizados utilizando quatro imagens. As tabelas a seguir mostram as predições encontradas pela rede para cada multiplicador e a pontuação que vai de zero a um. A execução no Spike foi realizada com pré-processamento *caffe2* e otimização nível 99, que realiza a transformação NHWC, mesmo formato de armazenamento de dados utilizado pelo TensorFlow, da Google.

Tabela 4.4: Predição googlenet_quantized para a imagem dog.jpg

BASE/HG4	GQU	GQV	HF7	GSM	GV3
0.010029 Newfoundland, Newfoundland dog	0.010988 Newfoundland, Newfoundland dog	0.010905 Newfoundland, Newfoundland dog	0.023353 Staffordshire bullterrier, Staffordshire bull terrier	0.007105 Great Dane	0.006701 spatula
0.017131 curly-coated retriever	0.018481 curly-coated retriever	0.023470 curly-coated retriever	0.032730 flat-coated retriever	0.022115 Newfoundland, Newfoundland dog	0.008669 bottlecap
0.083233 Great Dane	0.059236 Great Dane	0.043859 Great Dane	0.037524 curly-coated retriever	0.070950 curly-coated retriever	0.009017 sunglasses, dark glasses, shades
0.288698 flat-coated retriever	0.309341 flat-coated retriever	0.330764 flat-coated retriever	0.092641 Labrador retriever	0.334895 Labrador retriever	0.010080 syringe
0.548024 Labrador retriever	0.559998 Labrador retriever	0.551420 Labrador retriever	0.104044 Chesapeake Bay retriever	0.530900 flat-coated retriever	0.012701 corn

Figura 4.1: dog.jpg

Fonte: (RESEARCH, 2021)

A tabela 4.4 mostra os resultados para a imagem 4.1. A coluna *BASE/HG4* mostra os valores obtidos com o multiplicador padrão do C e do multiplicador da biblioteca EvoApproxLib. A primeira coluna mostra que a rede teve como valores mais expressivos os 54,8% de certeza que a imagem se trata de um Labrador e 28,8% de chance de ser um Flat-coated Retriever. Aqui, os multiplicadores aproximativos GQU e GQV tiveram excelentes resultados, com o GQU chegando a 56% de certeza de ser um Labrador e 30,9% de chance de ser um Flat-coated Retriever, e o GQV tendo 55,1% de certeza de ser um Labrador e 33% de certeza de ser um Flat-coated Retriever. O multiplicador GSM teve desempenho razoável, com 33,48% de chance de ser um Labrador e 53% de chance de ser um Flat-coated Retriever. Já o HF7 teve desempenho insatisfatório, com apenas 9% de chance de ser Labrador e 10% de chance de ser um Chesapeake Bay Retriever, uma raça que sequer estava no top 5 dos outros multiplicadores. Por último, o multiplicador GV3 fracassou completamente na tarefa, chegando a conclusão de que a imagem tem 1,2% de chance de ser milho ou 1% de chance de ser uma seringa.

Figura 4.2: cat.jpg

Fonte: (RESEARCH, 2021)

Tabela 4.5: Predição googlenet_quantized para a imagem cat.jpg

BASE/HG4	GQU	GQV	HF7	GSM	GV3
0.005367 tiger, Panthera tigris	0.004981 tiger, Panthera tigris	0.004272 tiger, Panthera tigris	0.065223 lynx, catamount	0.006908 window screen	0.006701 spatula
0.007940 lynx, catamount	0.012484 lynx, catamount	0.016889 lynx, catamount	0.088626 window screen	0.078528 lynx, catamount	0.008669 bottlecap
0.185519 Egyptian cat	0.208846 Egyptian cat	0.212267 tiger cat	0.089080 tiger cat	0.189827 tiger cat	0.009017 sunglasses, dark glasses, shades
0.186814 tiger cat	0.209870 tiger cat	0.221418 Egyptian cat	0.199755 tabby, tabby cat	0.323848 Egyptian cat	0.010080 syringe
0.608403 tabby, tabby cat	0.557362 tabby, tabby cat	0.537299 tabby, tabby cat	0.227991 Egyptian cat	0.381938 tabby, tabby cat	0.012701 corn

Para a imagem 4.2, a tabela 4.5 mostra que os três valores mais expressivos do multiplicador sem perdas são 60,8% para a chance de ser um gato malhado, 18,6% de chance de ser um gato tigrado e 18,5% de chance de ser um gato egípcio. Novamente os multiplicadores GQU e GQV tiveram desempenho interessante. O multiplicador GQU apresentou 55,73% de chance de ser um gato malhado, 20,9% de chance de ser um gato tigrado e 20,8% de chance de ser um gato egípcio. Já o multiplicador GQV teve 53,72% de probabilidade de ser um gato malhado, 22,14% de probabilidade de ser um gato egípcio e 21,22% de probabilidade de ser um gato egípcio. O multiplicador GSM obteve valores com menos disparidade, o que demonstra um grau maior de incerteza da predição. Mesmo assim, a rede ainda acredita majoritariamente que a imagem se trata de um gato malhado, com 38,19% de chance, contra 32,38% de chance de ser um gato egípcio e 18,98% de chance de ser um gato tigrado. O multiplicador HF7, contudo, teve como item de maior probabilidade o gato egípcio, com 22,79% de chance e em segundo, com 19,97% de chance, o gato malhado. Por fim, o multiplicador GV3 obteve os mesmos resultados errôneos da imagem dog.jpg.

Figura 4.3: ball.jpg

Fonte: (RESEARCH, 2021)

A imagem 4.3 é uma imagem mais simples, e isso ocasionou em inferências com

Tabela 4.6: Predição googlenet_quantized para a imagem ball.jpg

BASE/HG4	GQU	GQV	HF7	GSM	GV3
0.000004 croquet ball	0.000005 croquet ball	0.000008 croquet ball	0.026198 head cabbage	0.000021 croquet ball	0.006701 spatula
0.000008 racket, racquet	0.000012 racket, racquet	0.000018 racket, racquet	0.028501 Petri dish	0.000089 racket, racquet	0.008669 bottlecap
0.000014 Granny Smith	0.000022 Granny Smith	0.000041 Granny Smith	0.134568 lemon	0.000318 lemon	0.009017 sunglasses, dark glasses, shades
0.000084 lemon	0.000100 lemon	0.000144 lemon	0.214231 Granny Smith	0.000551 Granny Smith	0.010080 syringe
0.999885 tennis ball	0.999856 tennis ball	0.999783 tennis ball	0.301460 tennis ball	0.998978 tennis ball	0.012701 corn

um alto grau de precisão. A versão base do multiplicador chegou a 99,9885% de certeza de se tratar de uma bola de tênis. Aqui os multiplicadores aproximativos GQU e GQV tiveram uma taxa de erro entre 0,015% e 0,02%, com as respectivas probabilidades de 99,9856% e 99,9783%. O multiplicador GSM teve uma taxa de erro de 0,1%, chegando a uma probabilidade de 99,8978%. Já o multiplicador HF7 teve uma precisão bem mais baixa, com 30,14% de probabilidade de ser uma bola de tênis, 21,42% de probabilidade de se tratar de uma maçã verde e 13,45% de probabilidade de ser um limão. O multiplicador GV3 mais uma vez não conseguiu realizar uma predição minimamente aceitável.

Figura 4.4: fish.jpeg

Fonte: https://fishinfoblog.blogspot.com/2012/08/fish.html

Tabela 4.7: Predição googlenet_quantized para a imagem fish.jpeg

BASE/HG4	GQU	GQV	HF7	GSM	GV3
0.008300 sea slug, nudibranch	0.006406 sea slug, nudibranch	0.006494 sea slug, nudibranch	0.026107 sea anemone, anemone	0.006830 goldfish, Carassius auratus	0.006701 spatula
0.009112 sea anemone, anemone	0.009669 sea anemone, anemone	0.011424 sea anemone, anemone	0.043195 coral reef	0.020319 sea anemone, anemone	0.008669 bottlecap
0.011859 coral reef	0.012070 coral reef	0.013239 coral reef	0.074871 rock beauty, Holocanthus tricolor	0.023174 coral reef	0.009017 sunglasses, dark glasses, shades
0.026349 rock beauty, Holocanthus tricolor	0.029110 rock beauty, Holocanthus tricolor	0.038195 rock beauty, Holocanthus tricolor	0.148312 goldfish, Carassius auratus	0.083282 rock beauty, Holocanthus tricolor	0.010080 syringe
0.892855 anemone fish	0.901724 anemone fish	0.884878 anemone fish	0.183296 anemone fish	0.837470 anemone fish	0.012701 corn

Por fim, a imagem 4.4 obteve os resultados da tabela 4.7. A predição mais significativa da rede com o multiplicador padrão e com o multiplicador HG4 para essa imagem foi de 89,28% de chance de ser um peixe-palhaço. Para o multiplicador GQU o valor foi ainda maior: 90,17%. Já o multiplicador GQV chegou aos 88,48% de probabilidade de ser um peixe-palhaço. O multiplicador GSM obteve 83,74% de certeza de ser um peixe-

palhaço e o multiplicador HF7 teve 18,32% de certeza de ser um peixe-palhaço e 14,83% de certeza de ser um peixe dourado. O multiplicador GV3 teve 0% de acerto mais uma vez.

4.3 Taxa de precisão dos multiplicadores aproximativos na rede GoogLenet_quantized

Levando em consideração que as respostas esperadas para as imagens dog.jpg (4.1), cat.jpg (4.2), ball.jpg (4.3) e fish.jpeg (4.4) eram, respectivamente, Labrador Retriever, Tabby cat, Tennis ball e Anemone Fish, é possível calcular uma média de precisão da rede utilizando os diversos multiplicadores aproximativos usados. Calculando as médias dos valores obtidos chega-se aos valores da tabela 4.8. O multiplicador GV3 não consta na tabela pois obteve precisão zero.

Tabela 4.8: Precisão da rede GoogLeNet_quantized utilizando diferentes multiplicadores aproximativos.

	dog.jpg (Labrador Retriever) (%)	cat.jpg (Tabby cat) (%)	ball.jpg (Tennis ball) (%)	fish.jpeg (Anemone fish) (%)	AVG (%)
BASE/HG4	54.8024	60.8403	99.9885	89.2855	76.22
GQU	55.9998	55.7362	99.9856	90.1724	75.47
GQV	55.142	53.7299	99.9783	88.4878	74.3345
HF7	9.2641	19.9755	30.1460	18.3296	19.4288
GSM	33.4895	38.1938	99.8978	83.747	63.83

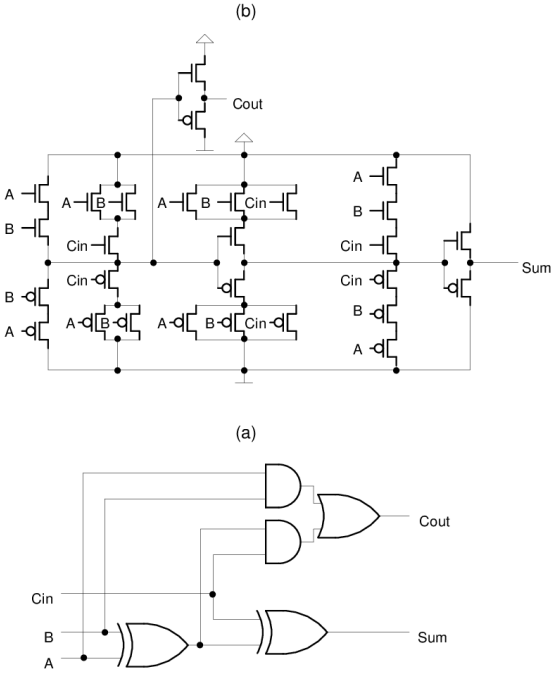
Realizando a média das porcentagens das predições dos valores esperados, percebe-se que rodar a rede com o multiplicador base do C ou o multiplicador HG4 apresentou uma precisão média de 76,22%. Os multiplicadores aproximativos, como esperado, apresentaram uma precisão média mais baixa que a versão base. Mesmo assim, foram médias bem próximas, com o multiplicador GQU alcançando uma média de 75,47%, o multiplicador GQV com 74,33% e o multiplicador GSM com impressionantes 63,83%. A precisão média do multiplicador HF7, todavia, foi baixa, chegando a 19,42%.

4.4 Razão Potência x Área

Conforme mostra a figura 2.2, o vetor sistólico do Gemmini é composto de *tiles* separados por registradores de pipeline. Dentro de cada tile está o Elemento Processador, que de acordo com a configuração atribuída ao Gemmini, pode ser único ou bidimensio-

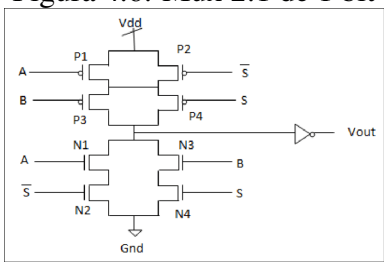
nal. Dentro do PE estão dois circuitos: um para uso em aplicações weight-stationary e outro para aplicações output-stationary. No circuito weight-stationary existe um registrador, um multiplicador, um somador e um multiplexador 2:1, e no circuito output-stationary estão um multiplicador, um somador, um registrador e pode haver um registrador com shift, caso for utilizados valores float32 no vetor. Nos testes realizados, todos os valores são inteiros de 16 bits, portanto não foi utilizado registrador com shift. Os valores de área e consumo de potência dos multiplicadores da EvoApproxLib são fornecidos pela própria biblioteca. Já que a EvoApproxLib possui um multiplicador de 16 bits com sinal que não possui perdas, assume-se que o multiplicador de 16 bits do C é tão grande e gasta tanta potência quanto o multiplicador da EvoApproxLib. Para o cálculo de área e potência dos demais componentes foram utilizados os dados de área e potência de um somador de 8 bits com sinal da EvoApproxLib sem perdas e foi feito o seguinte cálculo: um somador completo de 1 bit com lógica CMOS possui 28 transistores, portanto um somador completo de 8 bits possui 28x8 = 224 transistores. Dividindo a área do somador da EvoApproxLib que é de 70,4 um^2 pela quantidade de transistores chega-se a área de 70,4/224 = 0,314um^2. Realizando o mesmo cálculo para a potência, o valor encontrado é de 0,00015mW por transistor.

Figura 4.5: Somador completo de 1 bit



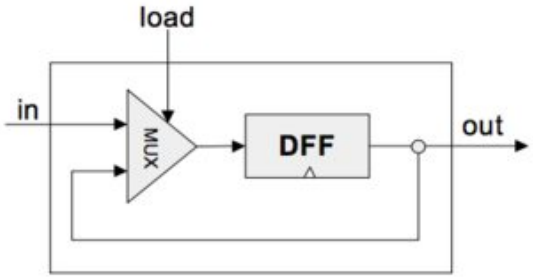
Fonte: https://www.researchgate.net/figure/1-bit-full-adder-descriptions-a-Logical-b-Transistor-level_fig2_2705927

Figura 4.6: Mux 2:1 de 1 bit



https://www.researchgate.net/figure/21-MUX-using-CMOS-logic-only_fig4_349211497

Figura 4.7: Registrador de 1 bit



Fonte: https://cs.stackexchange.com/questions/74000/1-bit-register-with-data-flip-flop-doesnt-store-bit

Com esses valores de área e potência por transistor foram calculadas área e potência dos outros componentes. Os componentes que não variaram durante os testes possuem os seguintes valores:

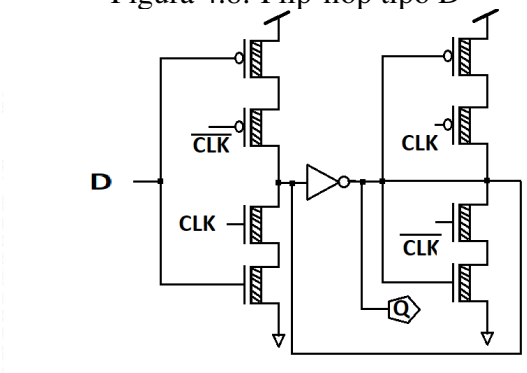
- *Somador 16 bits com sinal*: Potência = 0,0672mW, Área = 140,8 um^2;
- *Mux 2:1 16 bits*: Potência = 0,0192mW, Área = 40,192um^2;
- *Registrador 16 bits*: Potência = 0,0384mW, Área = 80,384um^2.

Para o cálculo de área e potência do vetor sistólico completo deve-se levar em consideração os registradores de pipeline, que, para um vetor de dimensões 16x16, serão um total de 480 registradores e 256 PEs, com cada PE contendo 2 multiplicadores, 2 somadores, 2 registradores e 1 Mux. Utilizando os valores obtidos com as quantidades que estão no vetor sistólico 16x16, chega-se às seguintes fórmulas para cálculo de potência e área:

- *Potência do vetor sistólico* = 512 x (Potência do multiplicador) + 77,4144;
- *Área do vetor sistólico* = 512 x (Área do multiplicador) + 162119,68.

Utilizando os dados dos multiplicadores fornecidos pela EvoApproxLib chegamos aos valores de vetores sistólicos da tabela 4.9:

Figura 4.8: Flip-flop tipo D



Fonte: https://www.researchgate.net/figure/Static-CMOS-type-DFF-using-CNFET_fig2_349710358

Tabela 4.9: Cálculos de potência e área dos vetores sistólicos.

	Potência (mW)	Ganho Potência (%)	Area (um^2)	Ganho Área (%)
BASE/HG4	1306.2144	0	1500487.68	0
GQU	1268.3264	2.9	1577390.08	-5.12
GQV	1267.8144	2.93	1575444.48	-4.99
HF7	1167.9744	10.58	1481287.68	1.27
GSM	1081.4464	17.2	1439918.08	4.03
GV3	792.1664	39.35	1151610.88	23.25

5 CONCLUSÕES

Com o experimento realizado, de acordo com os valores da tabela 4.9 e da tabela 4.8 tem-se as seguintes conclusões:

- O vetor sistólico com o multiplicador GQU obteve um ganho de potência de 2,9%, um aumento de área de 5,12% com uma diferença na média de precisão de 0,98% comparado com o multiplicador padrão;
- O vetor sistólico com o multiplicador GQV teve ganho de potência de 2,93%, aumento de área de 4,99% e uma diferença na precisão de 2,47% em comparação com o multiplicador padrão;
- O vetor sistólico com o multiplicador HF7 teve ganho de potência de 10,58%, ganho de área de 1,27% e diferença de 74,5% na precisão;
- O vetor sistólico utilizando o multiplicador GSM teve ganho de potência de 17,2%, ganho de área de 4,03% e diferença de 16,25% na precisão;
- O vetor sistólico com o multiplicador GV3 obteve ganho de potência de 39,35%, ganho de área de 23,25% e diferença de precisão de 100%.

Os multiplicadores GQU e GQV obtiveram baixas perdas de precisão, mas com um certo aumento de área. Com um ganho de potência de quase 3%, essas alternativas podem ser interessantes dependendo do projeto envolvido. O multiplicador GSM teve um alto ganho de potência e um bom ganho de área, devendo ser avaliado se a perda de precisão de 16,25% vale a pena para a aplicação. O multiplicador HF7 obteve o desempenho menos impressionante, sendo pior que o multiplicador GSM em todos os aspectos. Por fim, o multiplicador GV3 apesar de ter expressivos ganhos de potência e área se torna inútil devido ao seu desempenho nulo nas inferências.

Multiplicadores aproximativos podem ser uma alternativa viável para um projeto embarcado em que se busca ganhos de potência e área, desde que a perda de precisão no resultado não comprometa expressivamente o funcionamento do sistema, com o balanço entre potência, área e precisão variando de aplicação para aplicação.

REFERÊNCIAS

ADAMS, E.; VENKATACHALAM, S.; KO, S.-B. Energy-efficient approximate mac unit. **IEEE International Symposium on Circuits and Systems (ISCAS)**, May 2019.

AMAZON. **AWS - What is a neural network?** 2023. <https://https://aws.amazon.com/what-is/neural-network/>. Accessed on 2023-08-22.

BASAVARAJAIAH, M. **6 basic things to know about Convolution**. 2019. <https://medium.com/@bdhuma/6-basic-things-to-know-about-convolution-daef5e1bc411>. Accessed on 2023-08-22.

ESPOSITO, D.; STROLLO, A. G. M.; ALIOTO, M. Low-power approximate mac unit. **Ph.D. Research in Microelectronics and Electronics (PRIME)**, p. 81–84, Jul 2017.

GENC, H. et al. Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration. In: **Proceedings of the 58th Annual Design Automation Conference (DAC)**. [S.l.: s.n.], 2021.

IBM. **IBM - What is a neural network?** 2023. <https://www.ibm.com/topics/neural-networks>. Accessed on 2023-08-22.

KOROL, G. et al. Confax: Exploiting approximate computing for configurable fpga cnn acceleration at the edge. **IEEE International Symposium on Circuits and Systems (ISCAS)**, p. 1650–1654, May 2022.

KUNG, H. T. Why systolic architectures? **Computer**, v. 15, n. 1, p. 37–46, Jan 1982.

MCCULLUM, N. **Deep Learning Neural Networks Explained in Plain English**. 2020. <https://www.freecodecamp.org/news/deep-learning-neural-networks-explained-in-plain-english/>. Accessed on 2024-02-08.

MRAZEK, V. et al. Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods. **Design, Automation Test in Europe Conference Exhibition (DATE)**, p. 258–261, Mar 2017.

PECCIA, F. **Hardware Accelerators for Neural Networks**. 2022. <https://medium.com/towards-data-science/accelerating-neural-networks-on-hardware-baa3c14cd5ba>. Accessed on 2023-08-22.

RESEARCH, U. B. A. **Gemmini Website**. 2021. <https://github.com/ucb-bar/gemmini>. Accessed on 2023-08-18.

TECHNOLOGY, C. R. Faculty of information. **EvoApproxLib Website**. 2017. <https://ehw.fit.vutbr.cz/evoapproxlib/>. Accessed on 2023-08-18.

VENKATACHALAM, S.; KO, S.-B. Design of power and area efficient approximate multipliers. **IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS**, v. 25, n. 5, p. 1782–1786, May 2017.