SALÃO DE INICIAÇÃO CIENTÍFICA XXIX SIC

UFRGS PROPESQ

Salão UFRGS 2017

múltipla UNIVERSIDADE inovadora inspiradora

Evento	Salão UFRGS 2017: SIC - XXIX SALÃO DE INICIAÇÃO CIENTÍFICA DA UFRGS
Ano	2017
Local	Campus do Vale
Título	Otimização de Desempenho com Baixo Custo em Processadores Softcores via Reuso de Funções
Autor	PEDRO HENRIQUE EXENBERGER BECKER
Orientador	ANTONIO CARLOS SCHNEIDER BECK FILHO

Título: Otimização de Desempenho com Baixo Custo em Processadores *Softcores* via Reuso de Funções
Autor: Pedro Henrique Exenberger Becker
Orientador: Antônio Carlos Schneider Beck Filho
Instituição: Universidade Federal do Rio Grande do Sul

Processadores implementados em FPGAs através de uma descrição em linguagem de hardware, conhecidos como *softcores*, ganharam espaço já que: (i) permitem a definição de arquiteturas específicas bem como aceleradores de hardware, (ii) a descrição do hardware se perpetua para além da tecnologia atual, (iii) o FPGA pode ser reutilizado em projetos futuros, reduzindo custo e (iv) possibilita prototipação. Entretanto, processadores menos robustos (como os softcores) não conseguem garantir a execução de aplicações pesadas, como codificação de vídeo, imagem ou criptografia, em tempo plausível. É por isso que smartphones, por exemplo, fazem uso de Multiprocessor Systems-on-Chips (MPSoCs) como sua unidade de processamento principal. Nestes dispositivos, diversos módulos de hardware específicos são adotados – além do processador – para garantir a entrega de processamento.

No caso de softcores, muitas vezes é inviável alocar todos os módulos que se deseja dentro dos recursos disponíveis de um determinado FPGA. Quando isso ocorre, o trabalho que seria realizado por um acelerador de hardware deve ser resolvido por software, ao custo de performance. Para aliviar o custo de desempenho quando se precisa de uma solução em software e aumentar o espaço de exploração de projeto, este trabalho apresenta uma técnica para otimizar aplicações sem o custo de módulos de hardware dedicados.

Para tal, observamos que em projetos de FPGA, o uso percentual de Slice LUTs (que implementam as funções lógicas) e Slice Registers (que registram valores) é muito maior do que o de Block-RAMs (BRAMs, memória distribuída dentro do FPGA). A ideia é ocupar estes componentes de memória subutilizados para acelerar aplicações específicas, utilizando uma técnica de reúso na execução de programas. Esta técnica baseia-se em salvar os valores de entrada e saída de funções em uma tabela de reúso, baseada em BRAMs. A partir disto, quando a chamada de uma função se repete e os parâmetros de entrada casam com os valores salvos, é possível obter o resultado da função diretamente da tabela de reúso, ao invés de reexecutar toda a função, reduzindo o tempo de execução.

Como estudo de caso, a técnica foi analisada sobre operações em ponto-flutuante. Assim, ao invés de inserirmos uma nova unidade específica para isto no FPGA, fazemos o uso de uma biblioteca que resolve operações de ponto-flutuante convertendo-as em sucessivas operações sobre dados inteiros. A técnica suporta, atualmente, funções que resolvem as quatro operações básicas (adição, subtração, multiplicação e divisão) em tipos de dados de ponto-flutuante com precisão dupla. O esquema de reúso foi implementado sobre a descrição de hardware (VHDL) do softcore VLIW ρVEX, em sua versão 4-issue.

A partir das modificações, obteve-se uma aceleração média de 23% em 5 dispositivos testados entre Virtex 4, 5 e 7 (xc4vlx40, xc4vsx55, xc5vsx50t, xc5vsx95t, xc7vx690t) considerando uma tabela de reúso com 4096 entradas (a maior tabela que cabe em todos os modelos testados). Além disso, nosso esquema de reúso gerou acréscimo de slice LUTs de 22% e o de slice registers de apenas 4% comparado com a versão não modificada do processador. Em contrapartida, uma unidade aritmética de ponto flutuante teria demandado 141% a mais de slice LUTs e 48% a mais de slice Registers em comparação com o ρVEX não modificado.

Finalmente, estão sendo analisados os impactos de um sistema de reúso de funções em um ambiente multicore, onde uma tabela de reúso pode ser compartilhada entre aplicações que executam concorrentemente, bem como para novos nichos de aplicação (além de ponto-flutuante).