Computação aproximada em modelo de alto nível de processadores RISC-V

Palavras-Chave: Computação Aproximada, Computação Energeticamente Eficiente, RISC-V

Autores/as:
Lucas Cunha Agustini [UNICAMP]
Prof. Dr. Lucas Francisco Wanner (orientador) [UNICAMP]

INTRODUÇÃO:

Até então, os avanços tecnológicos em microchips têm-se dado pelo refinamento dos processos utilizados para construção e redução do tamanho dos transistores que compõem o processador moderno, pois este fato permite uma redução na tensão de alimentação do circuito e uma redução no atraso de propagação, levando a frequências de operação maiores [1].

Entretanto, décadas de avanços neste campo levaram a tecnologia de fabricação atual a atingir os limites físicos dos transistores, com a possibilidade de produzir transistores com tamanhos na ordem de átomos, além de problemas com a corrente de fuga, impossibilitando uma redução maior da tensão sem um aumento correspondente na potência dissipada pelo circuito [2].

A Computação aproximada tem sido objeto de estudo há alguns anos como uma alternativa para diminuir as necessidades energéticas de um circuito [3]. Tem se notado que diversas aplicações são resilientes a resultados imprecisos em seus cálculos [4], então algumas técnicas são aplicadas com o objetivo de reduzir o consumo energético, como redução controlada de tensões de alimentação de circuitos, o que pode levar a falhas no chaveamento de transistores, a adaptação da lógica de funções de modo a gerar hardware menor, mas com a possibilidade de erros no cálculo, e a não execução de funções que pouco contribuem para o resultado final [5].

No trabalho anterior a este projeto foi desenvolvida ADeLe [6], uma linguagem de descrição de aproximações e modelagem energética para ser integrada a simuladores, provendo então uma maneira fácil e rápida de implementar e testar diversas aproximações pelo hardware e software simulado, demonstrada com um modelo de processador MIPS.

Neste trabalho, o objetivo é testar esta implementação de ADeLe ao utilizar o simulador ArchC com suporte à linguagem para implementar e testar aproximações em um modelo de processador RISC-V de modo a avaliar a portabilidade e facilidade de implementação de ADeLe para outras arquiteturas de processador.

IMPLEMENTAÇÃO:

O simulador ArchC necessita de um modelo do processador, que descreve todos os detalhes deste, como quantos registradores ele possui e o tamanho e utilidade de cada um, além

da descrição de funcionamento de todas as instruções que ele suporta. Por conta disto, foi escolhido o processador RISC-V RV32IMAFD, que já possui um modelo para o simulador ArchC, porém este modelo é antigo e não funcional.

Deste modo um modelo novo derivado deste antigo foi criado, o qual inclui instruções faltantes ou que tinham comportamento errado, que foram corrigidas e implementadas. Os procedimentos de chamadas de sistema foi completamente refeito para as versões mais recentes do ArchC, possibilitando que o programa simulado conseguisse utilizar argumentos da linha de comando e interfaces de entrada e saída diversas.

Por fim ajustes na inicialização do processador foram feitos para permitir com que o simulador iniciasse programas compilados para o processador RISC-V escolhido, sem modificações especiais para o simulador no processo de compilação. O modelo foi validado através de testes das aplicações que serão executadas para avaliação das aproximações, porém neste momento elas ainda são executadas sem estas.

Além disso também foi feito o processo de criação da descrição das aproximações utilizando a linguagem ADeLe, junto com o código que o simulador executa para aplicar a aproximação quando necessário. Por fim uma listagem das instruções e descrição de quais instruções podem receber quais aproximações foi feita para completar os passos necessários para o funcionamento do ADeLe.

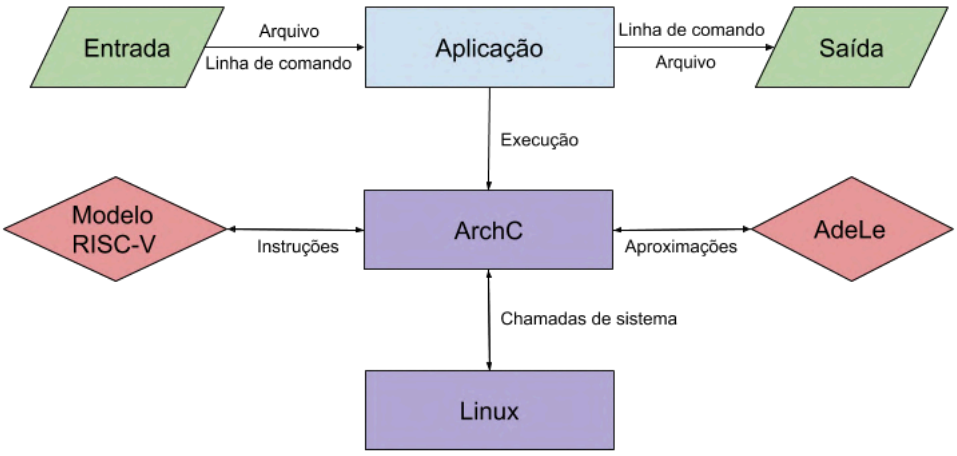

Figura 1: Arquitetura da pilha de software proposta com o ArchC.

RESULTADOS:

Com o objetivo de comparar o modelo RISC-V produzido neste trabalho com o modelo MIPS existente do trabalho anterior, alguns experimentos foram feitos, baseados nos experimentos presentes no trabalho original. Desta maneira foi decido comparar os resultados das aplicações JPEG, que utiliza multiplicações e portanto pode se beneficiar dos multiplicadores da biblioteca Evoapprox8b [7], e FFT, ou Fast Fourier Transform, uma operação matemática complexa amplamente utilizada em processamento de imagens e que pode se aproveitar do ponto flutuante de meia precisão.

Em primeiro temos na Tabela 1 a relação sinal-ruído de pico entre a imagem original escolhida e o resultado da compressão usando o algoritmo JPEG. Aqui podemos ver que os valores de ruído são bem próximos aos valores encontrados com a aplicação executando em MIPS, e a ordem entre os multiplicadores se mantém a mesma, o que mostra que as aproximações executam da mesma maneira entre as duas arquiteturas e a diferença absoluta dos números provavelmente se deve a diferenças entre as arquiteturas e os compiladores.

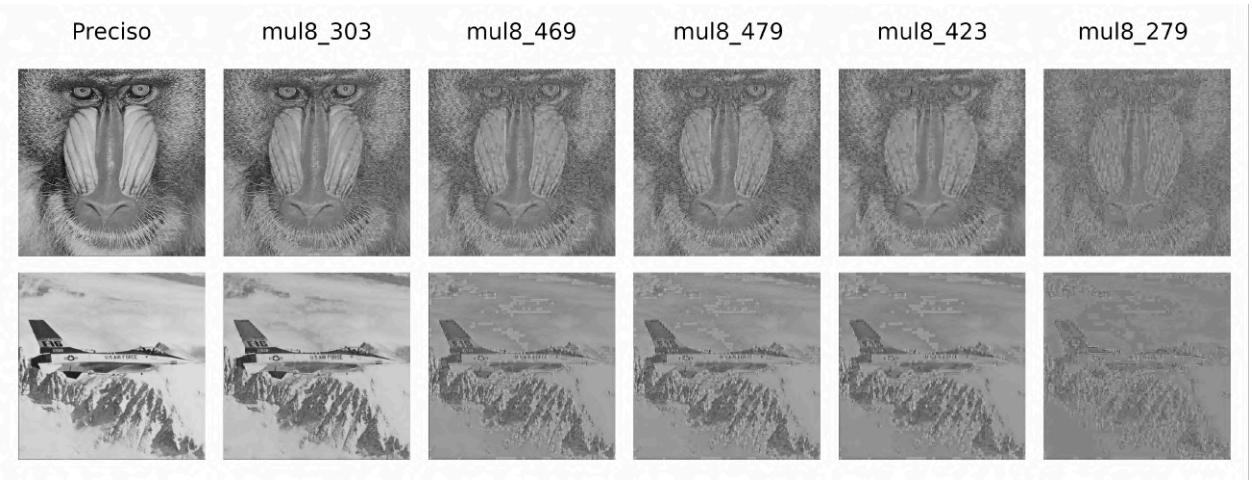

Figura 2: Imagens após compressão JPEG com cada tipo de multiplicador

	Preciso	**mul8_303**	**mul8_469**	**mul8_479**	**mul8_423**	**mul8_279**
PSNR (dB)	16,468	15,737	15,059	14,927	14,621	13,840
Tempo de execução (s)	38,030	38,200	38,210	38,280	37,900	36,900

Tabela 1: Relação sinal-ruído de pico entre a imagem JPEG e a imagem original do babuíno

Na mesma tabela podemos comparar o tempo de execução de cada cenário, e podemos ver que as aproximações da biblioteca Evoapprox8b não introduzem grandes diferenças no tempo de execução, sendo insignificativos os efeitos na performance do simulador.

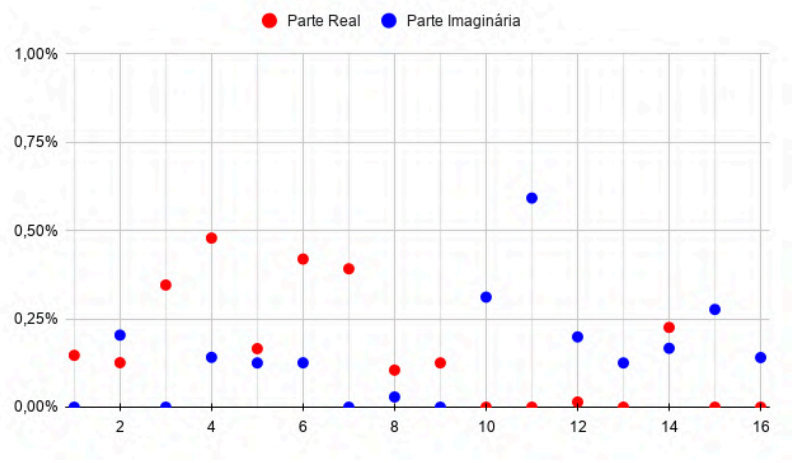

Figura 3: Erro da FFT com meia precisão em relação à precisão simples

Para a transformada de Fourier, foi executada para 16 entradas aleatórias, o que resulta em 16 saídas complexas. Na Figura 3 temos o erro relativo de cada saída complexa utilizando a aproximação de ponto flutuante de meia precisão em relação a saída esperada, quando executado com precisão simples.

Primeiramente pode-se notar que a escala do gráfico está

limitada a 1%, o que demonstra o baixo efeito que esta aproximação tem no resultado final da execução. Também vemos que vários resultados não possuem erro nenhum, ou seja, a execução com aproximação levou ao mesmo resultado final, porém retendo os ganhos de energia esperados de executar com aproximações.

EXTENSÃO DO PROJETO:

Como extensão deste trabalho, decidiu-se utilizar as aproximações discutidas anteriormente em hardware real, ao invés de simulado. Para chegar neste objetivo, foi decidido primeiro conseguir executar as aplicações no Spike, o simulador oficial da arquitetura RISC-V, para então avançar para uma FPGA, executando em hardware real finalmente.

O motivo para utilizar o Spike inicialmente é que ele tenta simular um processador exatamente, o que significa que ele não implementa facilidades que o ArchC possui, como chamadas de sistema e integração com o GDB, o GNU Debugger. Por conta disto, as ferramentas utilizadas para executar e resolver problemas com aplicações no Spike são as mesmas que em uma FPGA, o que permite um desenvolvimento mais rápido, num ambiente simulado e contido, mas que resulta numa infra-estrutura e código preparados para o hardware real.

Como o Spike não possui simulação de um kernel, com chamadas de sistema para lidar com entrada e saída, foi necessário pensar numa estratégia de comunicação com o programa RISC-V. Para isso decidimos utilizar o Open On-Chip Debugger, ou OpenOCD. Este permite se comunicar com a porta de debug presente no processador, e converte esta interface para que seja possível acessa-la utilizando o GDB.

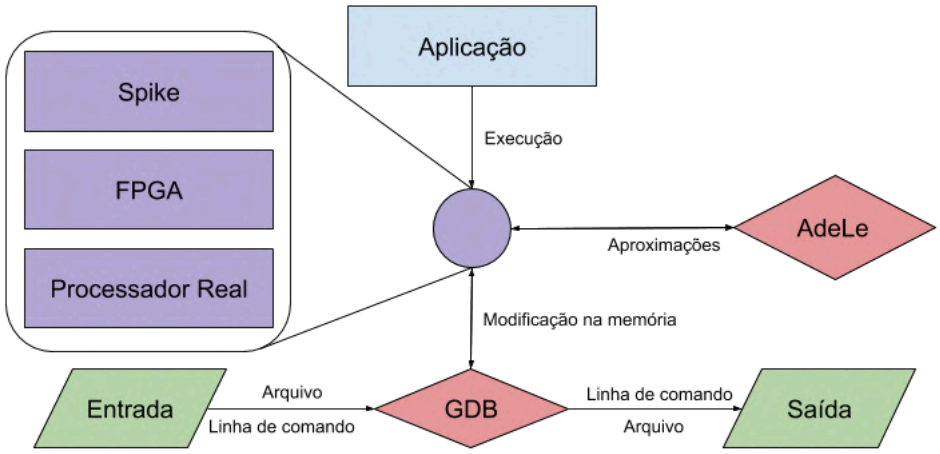

Figura 4: Arquitetura da pilha de software proposta com o Spike.

Com o GDB, é possível acessar variáveis do programa em memória e desta maneira se comunicar com o programa em execução, utilizando uma variável dedicada a entrada e saída para transferir a entrada, uma imagem crua por exemplo, e depois extrair a saída, a imagem comprimida.

Após termos uma maneira de lidar com a entrada e saída de cada programa, percebeu-se também a necessidade de termos uma biblioteca padrão de C e C++, já que até então estávamos

testando programas básicos, compilados para rodar diretamente no Spike, sem nenhuma biblioteca ou dependência externa, já que o objetivo era executar de maneira totalmente embarcada. Porém ao tentar utilizar os programas utilizados para benchmark dos modelos em ArchC, percebeu-se que muitos eram fortemente acoplados as bibliotecas padrões de C.

Por conta desta necessidade, foi necessário utilizar o PK, ou Proxy Kernel. Este é um ambiente de execução de aplicações RISC-V mínimo, mas que nos permite utilizar as bibliotecas padrões de C e C++, assim conseguimos compilar e executar algumas aplicações dentro do ambiente do Spike em conjunto com o GDB.

CONCLUSÕES:

Este trabalho concluiu a construção de um modelo de processador RISC-V para o ArchC, com suporte a aproximações através da linguagem ADeLe, consolidando o trabalho realizado, anteriormente com MIPS, como uma opção para desenvolvimento e teste rápido de aproximações de maneira portátil em diversas arquiteturas.

Com isso temos um modelo para o ArchC do processador RV32G funcionando totalmente, capaz de executar aplicações com aproximações do ADeLe sem necessidade de modificações no processo de compilação das aplicações.

Também foi possível ir além do objetivo proposto e avançar o estado da arte, implementando métodos novos para trazer o ADeLe, e suas vantagens na implementação de aproximações, para hardware real, aumentando ainda mais as possibilidades e utilidade da linguagem.

BIBLIOGRAFIA:

[1] J. Koomey, S. Berard, M. Sanchez, and H. Wong, “Implications of historical trends in the electrical efficiency of computing,” *IEEE Annals of the History of Computing*, vol. 33, no. 3, pp. 46–54, 2011.

[2] T. Austin et al., “Leakage current: Moore’s law meets static power; computer,” *IEEE Service Center*, vol. 36, no. 12, 2003.

[3] S. Mittal, “A survey of techniques for approximate computing,” *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, p. 62, 2016.

[4] J. Y. F. Tong, D. Nagle, and R. A. Rutenbar, “Reducing power by optimizing the necessary precision/ range of floating-point arithmetic,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 273–286, 2000.

[5] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar, “Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency,” in *Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 555–560.

[6] I. B. Felzmann, M. M. Susin, L. Duenha, R. Azevedo, and L. F. Wanner, “Adele: Rapid architectural simulation for approximate hardware,” in *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 2018, pp. 9–16.

[7] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, “Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods,” in *Proceedings of the Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2017, pp. 258–261.