MO401 1s2006 - Trabalho 1: Resumo do Artigo [KBG+04]

Tony Minoru Tamura Lopes - ra017502

24 de Abril de 2006

Em [KBG+04] é apresentada a arquitetura de processadores *Vector-Thread*(VT). Essa arquitetura tenta explorar extensivamente o paralelismo e a localidade, características fundamentais para se melhorar o desempenho com o aumento do número de transistores, quando se lida com atrasos e dissipação de calor nos circuitos. Para tal, ela une os modelos de execução vetorial e *multithread*, fornecendo uma ISA onde se pode utilizar o paralelismo e a localidade explicitamente, sem necessitar de uma área considerável no circuito para extrair o paralelismo dinâmicamente e realizar comunicação de longo prazo. Essa arquitetura visa, dessa forma, especialmente o domínio dos processadores embarcados.

O modelo de programação do VT utiliza de um processador de controle e um vetor de processadores virtuais (VPs). Os VPs possuem conjuntos de registradores próprios e executam grupos de instruções "RISC-like" empacotadas em um bloco atômico de instruções (AIBs). Não havendo um PC automático, o processador de controle utiliza de comandos *vector-fetch* para distribuir AIBS para todos os VPs, executando código em paralelo. Já para código de threads independentes, cada VP executa comandos *thread-fetch* e adquire suas próprias AIBs.

Esse mecanismo possibilita diversas formas de mapeamento de aplicações no VT, sendo a execução de *loops*, a principal. Nela cada iteração é executada por um VP e o código comum pode ser executado pelo processador de controle. Para permitir a execução de loops com depedência de dados, os VPs são conectados em uma topologia de anel unidirecional e comandos de recebimento e envio para os VP adjacentes, podem ser utilizados. Em contraste com as arquiteturas VLIW, só há a necessidade de um *vector-fetch* para a AIB da iteração, sendo os atrasos, pelas dependências entre os VPs, calculados dinâmicamente.

No caso de loops internos, os comandos de *thread-fetch* podem ser utilizados para controlar o fluxo interno em um VP. Essa habilidade de realizar *vector-fetches* e *thread-fetches* permite à arquitetura VT explorar o paralelismo e a localidade.

[KBG+04] apresenta também, um protótipo de processador chamado SCALE que implementa os conceitos da arquitetura VT, visando sistemas embarcados.

Para se obter menor área e consumo de energia, o SCALE utiliza baias de execução para os VPs com múltiplos *clusters*, onde cada um deles possue somente um subconjunto de funções e registradores. A execução atômica dos AIBs permite ao SCALE expor registradores internos dos *clusters*, utilizados pela ALU, ao programador. Há também os registradores privados que mantém seu valor entre AIBs e registradores compartilhados, utilizados para armazenar dados comuns aos VPs. O desacoplamento dos *clusters* e a comunicação entre *clusters* de baias adjacentes, permite a execução de diversos VPs simultaneamente em uma mesma baia.

O processador protótipo SCALE usado para testes possuía um processador escalar de controle MIPS, 4 baias de execução cada uma com 4 *clusters*, suportando até 128 processadores virtuais. A área estimada para sua implementação, incluindo memória cache, é de 10mm^2. A freqüência de *clock* escolhida foi de 400mhz, considerando o consumo, complexidade de implementação e desempenho.

Os testes foram feitos usando o *benchmark* EEMBC, que abrange diversos domínios de software. O código para o SCALE foi produzido diretamente em *assembly* e foram utilizados códigos otimizados para outros processadores de sistemas embarcados usados na comparação. Analisou-se, nos testes, o mapeamento de paralelismo do SCALE, o aproveitamento da localidade e sua eficiência. Foi verificado que o SCALE obteve resultados competitivos, se comparado a outros processadores mais complexos, e que sua eficiência aumenta ao acrescentar novas baias de execução.

Por fim, a nova arquitetura VT se mostrou interessante, através da sua implementação satisfatória pelo SCALE, como uma opção de design menos complexa para processadores embarcados. O grande ganho neste aspecto está na economia de energia, sem perda de desempenho. A utilização da arquitetura VT para outros fins será vista em trabalhos futuros.

Referências

[KBG+04] R. Krashinsky, C. Batten, S. Gerding, M. Hampton, B. Pharris, J. Casper, and K. Asanović. The vector-thread architecture. In *31st International Symposium on Computer Architecture*, Munich, Germany, June 2004.