Implementação superescalar para o FemtoJava

Luiz S. Laurino, Flávio R. Wagner, Philippe O. A. Navaux

Instituto de Informática
Universidade Federal do Rio Grande do Sul - UFRGS
Av. Bento Gonçalves, 9500. Porto Alegre-RS
{lslaurino, flavio, navaux}@inf.ufrgs.br

Introdução

O FemtoJava[ITO 2001] é um processador desenvolvido na UFRGS e que atende restrições de área e potência, sendo voltado especificamente para sistemas embarcados. É capaz de processar *bytecodes* Java de forma nativa, possui conjunto reduzido de instruções, arquitetura *Harvard* e tamanho pequeno, dentre outras características.

A versão Pipeline do FemtoJava pode não ter um desempenho satisfatório para certas aplicações de sistemas embarcados, como multimídia, por exemplo [BEC 2004]. Dessa forma, para atender tais necessidades, foi desenvolvida uma versão VLIW. Como já se sabe, arquiteturas VLIW são muito dependentes do compilador e qualquer modificação em sua organização exige alterações no compilador também.

É apresentada neste trabalho uma extensão da versão Pipeline do FemtoJava, de modo a se ter dois fluxos de execução simultâneos no pipeline. Além disso, é utilizada a mesma estratégia para detecção de paralelismo proposta pelo FemtoJava VLIW, que leva em conta o fato do FemtoJava ser um processador de pilha [BEC 2004].

O processador superescalar

Para dotar o processador da capacidade de executar duas instruções ao mesmo tempo, um novo estágio foi criado no pipeline (até então de cinco estágios), aqui chamado de unidade de despacho. É nesse estágio onde as instruções são analisadas em busca de paralelismo e quanto à ocorrência de conflitos por uma mesma unidade funcional (UF).

Analisar paralelismo em processadores RISC convencionais corresponde à avaliação de dependências de dados e conflitos de UFs entre instruções, enquanto que em máquinas de pilha deve ser analisado, também, o uso da pilha de operandos, visto que as instruções têm como entradas e saídas os dados desta pilha. Dessa forma, uma seqüência de instruções caracteriza um fluxo de operação e o paralelismo é obtido entre fluxos.

Com a inclusão deste novo estágio, várias modificações tiveram de ser feitas nos outros estágios do pipeline, a fim de torná-los compatíveis com o novo modelo. Entretanto, apenas a unidade de despacho será detalhada nesta apresentação.

É no estágio da unidade de despacho que o paralelismo entre instruções é analisado. Optou-se por utilizar uma fila de despacho centralizada (a fila está associada a todas as UFs existentes na unidade de execução), devido à simplicidade de sua implementação em relação a uma janela de instruções distribuída.

	Superescalar	**VLIW**	**Pipeline**
Área	7530LCs	5693LCs*	3106LCs
Frequência	20.73 MHz	–	30.63 MHz

Tabela 1: Comparação dos resultados em área e frequência dos processadores

A execução fora de ordem pode ser feita apenas no despacho e finalização de instruções de fluxos diferentes. Esta restrição é imposta devido à seqüencialidade existente entre instruções de um mesmo fluxo, onde o despacho e a finalização precisam ser feitos em ordem.

É importante observar que um dos limitantes de desempenho em arquiteturas superescalares é o tamanho da janela de instruções utilizada para realizar a análise de paralelismo. Fazendo uma comparação com o processador VLIW, este tem uma janela de infinitas posições, já que analisa estaticamente o código da aplicação, enquanto que nesta implementação superescalar a janela limitou-se a 4 posições.

Resultados e considerações finais

Resultados preliminares foram obtidos através da prototipação do FemtoJava Superescalar em FPGA, utilizando o Quartus II 2.2 Web Edition. Como pode ser observado na Tabela 1, o FemtoJava Superescalar apresentou a maior área dentre as diversas versões do processador (7530 LCs), enquanto que uma estimativa [BEC 2004] para o VLIW atingiu 5693 LCs. Com relação à freqüência de operação, houve uma diferença significativa entre os *clocks* do Pipeline e do Superescalar, sendo que o caminho crítico ficou na unidade de despacho. Não há como se realizar uma comparação direta com o VLIW neste quesito porque ainda não existe uma versão em VHDL para esta versão.

A relação entre a área utilizada e o número de instruções despachadas em paralelo não parece vantajosa, visto a quantidade excessiva de área que cada posição no buffer de instruções da unidade de despacho consome. Portanto, devido às restrições de área, devemos ter um buffer bastante reduzido. Na continuidade deste trabalho, serão feitas análises mais precisas do desempenho e do consumo de potência do FemtoJava Superescalar na execução de aplicações embarcadas típicas.

Referências

[BEC 2004] BECK, Antonio C. S. Uso da técnica VLIW para aumento de performance e redução do consumo de potência em sistemas embarcados baseados em Java. Dissertação de Mestrado, PPGC-UFRGS. 2004.

[ITO 2001] ITO, Sergio, CARRO, Luigi, e JACOBI, Ricardo. Sashimi and FemtoJava: Making Java Work for Microcontroller Applications. IEEE Design & Test, Set-Out. 2001.

[WAG 2004] WAGNER, F. R. *Notas de aula*. 2004.