Avaliação do Pipeline no Processador MIPS_Robot

Vicente S. Cruz, Henrique C. Freitas, Philippe O. A. Navaux

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
{vscruz,hcfreitas,navaux}@inf.ufrgs.br

Resumo. *Com a crescente presença dos robôs na realização de diversas tarefas, fica claro a necessidade de aumentar o seu desempenho. Este artigo apresenta a avaliação da inclusão da técnica Pipeline no processador Mips_Robot, um chip de propósitos gerais desenvolvido para auxiliar na área da robótica. Os resultados iniciais mostram a viabilidade desta técnica para aumento do desempenho. Entre os trabalhos futuros ressaltamos a avaliação do protótipo para verificação da arquitetura proposta, além dos limites físicos impostos.*

1. Introdução

A tecnologia usada para implementar robôs vem crescendo consideravelmente, e é notável o crescente uso desses robôs no auxílio de diversas atividades da humanidade, como por exemplo a medicina, a indústria automobilística e a exploração de terrenos de difícil acesso ao homem, como o planeta Marte. Dessa forma, a obtenção de maior desempenho por parte deles aumentaria a produtividade no trabalho em que ele está dando suporte. Tendo essa meta em foco, incluiu-se a técnica de *Pipeline* no MIPS_Robot, um processador de propósitos gerais projetado para auxiliar na área da robótica [Cruz 2007]. A esse novo processador foi dado o nome de MIPS_Robot_Pipeline.

Para realizar essa atividade, utilizou-se a linguagem de descrição de arquiteturas ArchC [Rigo 2004] com o intuito de projetar um simulador do MIPS_Robot_Pipeline, em nível de arquitetura de conjunto de instruções, para realizar uma comparação de desempenho entre esse novo processador e o MIPS_Robot. Essa comparação foi realizada através da execução de uma aplicação e, através da comparação dos dados obtidos pelos dois processadores na realização dessa tarefa, conclui-se que é interessante incluir o *Pipeline* no *chip*.

2. Projeto do Processador

Este projeto visou a extensão do MIPS_Robot através da inclusão de um *Pipeline*, e o trabalho apresentado aqui aborda os métodos usados para incluir essa técnica. Dessa forma, para se obter mais informações sobre o projeto do processador, tal como a análise matemática e o estudo em robótica realizados para projetar o conjunto de instruções, podem ser encontrados em [Cruz 2007].

Técnicas de *Pipeline* são comumente usadas nos processadores atuais para obter um maior ganho de desempenho. De posse desse conhecimento, e visando obter uma maior produtividade do MIPS_Robot, é que decidiu-se incluir essa técnica no *chip*. Maiores detalhes sobre a técnica em si, tal como suas restrições, encontram-se em [Patterson 2005].

Inicialmente a extensão foi bastante semelhante à inclusão do *Pipeline* no processador MIPS [Patterson 2005], ou seja, dividiu-se o caminho de dados em cinco estágios e, entre cada estágio, inseriu-se um registrador para armazenar os valores intermediários. A diferença em relação ao MIPS é que, devido ao fato do MIPS_Robot utilizar dados vetoriais, esses registradores intermediários foram modificados para permitir a inclusão desse tipo de dado.

Além disso, existem dois novos tipos de instruções: instruções normais e instruções vetoriais. As primeiras, manipulam dados isolados de 32 bits, e as últimas manipulam dados vetoriais, que são vetores de três posições de 32 bits. Dessa forma, foi necessário incluir novos sinais de controle que identificam o tipo de instrução que está sendo executado no i-ésimo estágio do *Pipeline* para que, assim, a unidade de controle possa tomar a decisão mais adequada naquele instante.

Por fim, foi preciso reprojetar a unidade de *forwarding* para que ela também pudesse fazer o adiantamento de dados vetoriais. Para implementá-la, foi necessário levar em consideração todos os tipos de dados e instruções que o processador manipula. Por exemplo, suponha que a instrução mais recentemente lida da memória seja uma instrução de translação. Esta é uma instrução vetorial que manipula dados em ponto flutuante, requer dois dados vetoriais como operandos e retorna um dado vetorial como resposta. Isso faz a unidade de *forwarding* verificar diversas situações. Primeiramente é testado se a instrução posterior a essa, que está um estágio a frente no *Pipeline*, é uma instrução de ponto-flutuante e, se for, verifica-se se ela é uma instrução vetorial. Nesse caso a unidade de *forwarding* pode tanto realizar o adiantamento do vetor para o estágio em que a instrução de translação se encontra, caso a instrução mais a frente também seja vetorial, quanto realizar o adiantamento do dado único, caso ele faça parte de um dos operandos da instrução. Caso a última condição seja verdadeira, a unidade de *forwarding* ainda vai verificar, nos estágios posteriores do *Pipeline*, se existe mais algum dado isolado que possa ser adiantado, ou se o restante dos dados do vetor é buscado diretamente do banco de registradores. A Figura 1 mostra uma parte do caminho de dados do MIPS_Robot_Pipeline com as modificações realizadas. Nota-se, na figura, a inclusão dos dados vetoriais nos registradores do *Pipeline* (ArrayData1 e ArrayData2 em ID/EX, ALUArrayAnswerData em EX/MEM e RBArrayWriteBackData em MEM/WB), um sinal de controle na ALU (ArrayInst) para decidir se o dado a ser processado é vetorial ou não, e a unidade de *forwarding* recebendo os dados para a verificação de um possível adiantamento deles.

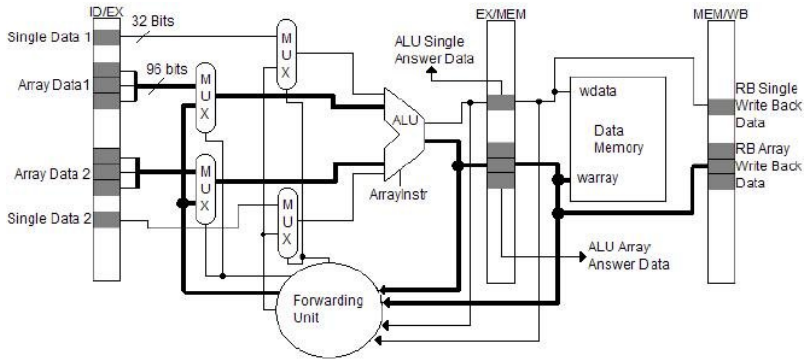

Figura 1. Caminho de Dados do MIPS_Robot_Pipeline

3. Aplicação

Uma das atividades realizadas por robôs consiste em explorar terrenos inacessíveis ao homem e coletar amostras desse ambiente. Temos como um exemplo dessa aplicação o sonda de Marte, que está explorando as terras desse planeta e enviando os dados para a NASA [NASA 2007]. Assim, uma tarefa bastante comum a esse sonda consiste em esticar e rotacionar um braço mecânico de tal forma a posicioná-lo sobre um objeto, capturar esse objeto e finalmente, rotacionar e encolher o braço para guardar essa amostra.

A aplicação implementada nesse trabalho consiste em realizar os cálculos necessários para determinar a posição em que o robô deve posicionar seu braço sobre um objeto do solo para que possa pegá-lo. Supondo que o braço esteja trabalhando em um sistema de coordenadas cartesianas (com eixos X, Y e Z), ele parte de um ponto inicial [2 3 4], translada 3 unidades na direção dos três eixos (com vetor de translação igual a [3 3 3]), rotaciona 3 radianos em torno do eixo X e, por fim, rotaciona 2 radianos em torno do eixo Z.

Para realizar essa aplicação na simulação, foi levado em consideração apenas a arquitetura do conjunto de instruções dos processadores, sendo que esse simulador estipulou que MIPS_Robot rodasse sob uma freqüência de 50MHz. Como incluiu-se cinco estágios nesse processador, supos-se que, com isso, o seu ciclo de clock seja divido por cinco, e com isso espera-se que o MIPS_Robot_Pipeline tenha um desempenho aproximadamente cinco vezes mais rápido comparado ao MIPS_Robot. A Figura 2 abaixo mostra o tempo de execução que cada *chip* levou para realizar a tarefa.

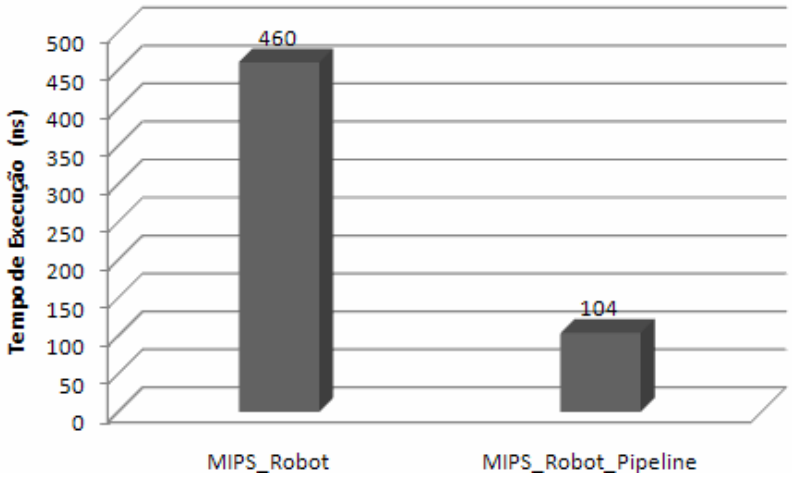

Figura 2. Tempo de execução dos processadores

O tempo de execução do MIPS_Robot foi de 460 ns e ele realizou 23 instruções, enquanto que o tempo de execução do MIPS_Robot_Pipeline foi de 104 ns, com a computação de 26 instruções. Apesar de ser mais rápido, o MIPS_Robot_Pipeline não atingiu o ganho de cinco vezes mais velocidade em relação ao primeiro. Isso se explica pelo fato do MIPS_Robot_Pipeline ter executado três instruções a mais que não fazem nada, ou seja, devido à aplicação, existiam bolhas dentro do *Pipeline* que o impediam de executar em desempenho máximo.

4. Conclusões e trabalhos futuros

Conforme os resultados mostram, constatou-se que em nível de conjunto de instruções é bem interessante incluir o *Pipeline* no processador, pois obteve-se um ganho de desempenho quase cinco vezes maior. No entanto, a velocidade do MIPS_Robot_Pipeline foi apenas estimado com base nas modificações realizadas em cima do MIPS_Robot, ou seja, o verdadeiro ciclo de *clock* do novo processador só será obtido através da implementação de um protótipo.

Como tarefas futuras planeja-se avaliar o desempenho desse processador em outras aplicações reais. Além disso, será construído o protótipo do *chip* para verificar a verdadeira velocidade dele, além de obter dados em um nível mais físico, como a quantidade de área ocupada e o consumo de potência. Tendo em mãos esses resultados, poderá se chegar a um resultado mais concreto se o ciclo de *clock* se alterará, ou não. Essas informações serão obtidas usando-se a linguagem de descrição de hardware VHDL (VHSIC *Hardware Description Language)* [Ashenden 2004] para que seja feita uma prototipação desse *chip* em um dispositivo FPGA (*Field Programable Gate Array*) [Xilinx 2007].

5. Referências

Ashenden, P. J. (2004), VHDL Tutorial, http://www.tutground.net/Files/VHDL_TUTORIAL.pdf.

Cruz, V. S., Freitas, H. C, Navaux, P. O. A. (2007) “Proposta de Expansão do Conjunto de Instruções do MIPS para Robótica”, Concurso de Trabalhos de Iniciação Científica em Arquitetura de Computadores e Computação de Alto Desempenho, Brasil.

NASA. (2007), Mars Exploration. http://mars.jpl.nasa.gov/.

Patterson, D. A., Hennessy, J. L. (2005), Organização e Projeto de Computadores: A Interface Hardware/Software, Editora Campus, 3ª edição.

Rigo, S., et al (2004) “ArchC: A SystemC-Based Architecture Description Language ”, International Symposium on Computer on Computer Architecture and High Performance Processing, pp.66-73, October 2004.

Xilinx (2007), Silicon Devices, http://www.xilinx.com/products/silicon_solutions/index.htm, December.