T&D-Bench – um repositório de modelos de simulação de processadores

Daniel Westerlund, Alaor Crestani, Júlio C. Maccali, Antônio R. Nodari, Sandro N. Soares

Universidade de Caxias do Sul
Alameda João Dal Sasso, 800, Bento Gonçalves – RS. Tel./Fax: 54-452-1188
snsoares@ucs.br

Introdução

A atividade de exploração do espaço de projeto incorporou-se ao fluxo de projeto dos atuais processadores de alto desempenho e de sistemas embarcados para auxiliar os projetistas no tratamento da complexidade inerente ao processo contemporâneo de projeto de sistemas computacionais. A exploração do espaço de projeto é a rápida avaliação de alternativas para a escolha das mais adequadas, segundo as especificações de desempenho, área, potência e custo, previamente estabelecidos. Patt [PAT 03] afirma que a Arquitetura de Computadores, se ela pode ser considerada uma ciência, é uma ciência de *trade-offs*, centrada na análise das vantagens e desvantagens das escolhas disponíveis. Por isso, a exploração do espaço de projeto é importante também no contexto educacional.

Este resumo apresenta os recursos de modelagem e de simulação do framework T&D-Bench. O T&D-Bench é um framework de código aberto e independente de plataforma, que provê um conjunto de soluções de modelagem e simulação de processadores que concilia as necessidades dos ambientes educacional e de pesquisa. Destas soluções, sobressai-se um processo de modelagem simplificado e, conseqüentemente, rápido, que contribui para a exploração do espaço de projeto. Na página do T&D-Bench [SOA 04] encontram-se diversos modelos de simulação de processadores que podem ser usados no ensino de Organização e Arquitetura de Computadores. Uma análise comparativa do T&D-Bench com outros ambientes de mesmo propósito, especialmente com *Architecture Description Languages*, está fora do escopo deste trabalho, e pode ser encontrada em [SOA 04].

Recursos de Modelagem

O T&D-Bench possui os seus recursos de modelagem dispostos em três camadas: biblioteca de componentes, linguagem de definição e a classe principal do framework, chamada classe *processor*. A biblioteca de componentes é a base para a criação de novos modelos de processadores na metodologia de modelagem do T&D-Bench. Na biblioteca, há componentes como registradores, bancos de registradores, unidades funcionais e memórias. Um componente do T&D-Bench possui portas, atributos e, eventualmente, conteúdo. Ele, igualmente, executa um certo comportamento. As portas são usadas para conectar componentes entre si ou para fornecer sinais de controle. Os atributos de um componente armazenam valores que

descrevem uma instância específica daquele componente. O conteúdo de um componente armazena o seu estado interno no caso de componentes como registradores, bancos de registradores e memórias. A biblioteca pode ser estendida com a criação de novas classes, representando novos componentes, para o atendimento de necessidades do projetista. A Figura 1 mostra um trecho selecionado da especificação da organização do processador didático Neander [WEB 01]. A seleção e a configuração de componentes são feitas usando-se o comando *create* da linguagem de definição do T&D-Bench, enquanto que a interconexão entre componentes é especificada usando-se o comando *link*. O comando *create* possui o seguinte formato:

• *create <nome> <tipo> <valor1> ... <valorN>*

onde: *nome* é um texto que define o nome do componente no modelo de processador que está sendo criado; *tipo* é um valor numérico que corresponde ao tipo do componente: um registrador ou uma unidade lógica e aritmética, por exemplo; *valor1* a *valorN* são valores numéricos a serem atribuídos a atributos do componente, como tamanho (de uma memória, por exemplo) e largura em bits. Cada componente possui a sua própria lista de valores numéricos a serem fornecidos no comando *create*. Por isso, o framework exige que um método documentando o formato do comando *create* seja programado para cada novo componente incluído na biblioteca. O comando link, por sua vez, estabelece uma conexão por onde trafegam dados entre dois componentes. Ele possui o seguinte formato:

• *link <origem> <psaida> <destino> <pentrada>*

onde: *origem* e *destino* são os nomes dos componentes a serem conectados. *origem* é a fonte de dados para o componente *destino*; *psaida* e *pentrada* são os nomes da porta de saída do componente fonte de dados e da porta de entrada do componente destino, respectivamente.

```
// Cria o reg.Acumulador de 8 bits. Valores
// numéricos após o nome: tipo=0,não sado=0,largura=8
create ACC     0       0       8

...
// Cria a Unidade Lógica e Aritm.: tipo=8,não usado=0,largura=8
create ULA     8       0       8
// Cria registradores de 1 bit para manter flags da ULA
create N       0       0       1
create Z       0       0       1
...
// Conecta os componentes
link ULA       S1      ACC     E1
link ULA       neg     N       E1
link ULA       zero    Z       E1
...
```

Figura 1. Especificação da Organização

A ativação de microoperações, como um cálculo numa unidade funcional e a carga de um registrador, que ocorrem durante a execução de uma determinada instrução, é descrita pela execução do comportamento de componentes. A definição de seqüências de execução do comportamento de componentes, ou seqüências de microoperações, compõe unidades elementares de execução que podem ser reutilizadas para a formação do comportamento das instruções do processador. A Figura 2 mostra um trecho da especificação da execução de uma operação pela Unidade Lógica e Aritmética. Para a especificação da execução do comportamento de um componente, na linguagem de definição do T&D-Bench, são usadas duas construções:

• *<nome>.< behavior | read | write>;* e

• *<nome>.<pcontrole> = valor.*

onde: *nome* é o nome do componente cujo comportamento será executado. Ele foi definido na modelagem da organização. No primeiro formato, usado para realmente executar o comportamento de um componente: *behavior* é usado para executar o comportamento de um componente que não possui conteúdo; *read* e *write* são usados para ler ou escrever, respectivamente, o conteúdo de um componente. No segundo formato, usado para configurar uma porta de controle do componente: *pcontrole* é o nome da porta de controle que será configurada; *valor* é o valor a ser atribuído à porta de controle (atualizado durante a simulação, quando da decodificação das instruções do programa).

```
// Lê o acumulador, executa a operação na ULA, escreve
// o resultado no acumulador, atualiza os flags
// negativo e zero
ACC.read
ALU.OP = 10
ALU.behavior
ACC.write
N.write
Z.write
```

Figura 2. Especificação da Arquitetura

Aspectos temporais devem ser especificados e, posteriormente, associados a microoperações individuais descritas em unidades elementares de execução. Mais detalhes sobre a especificação de aspectos temporais podem ser encontrados em [SOA 04]. Estas especificações, descrevendo a organização, a arquitetura e aspectos temporais do processador, são feitas usando-se a linguagem de definição do T&D-Bench.

As informações do modelo provenientes da descrição com a linguagem de definição podem ser acessadas e manipuladas por um conjunto de funções especializadas, denominadas macros do T&D-Bench. O uso destas macros para a implementação de mecanismos que não são contemplados na linguagem de definição deve ser feito estendendo-se a classe *processor*, que constitui o ponto de entrada para alterações na infra-estrutura de software.

Recursos de Simulação

Após a criação do modelo do processador, o framework disponibiliza classes de visualização e de interação que podem ser, facilmente, associadas ao modelo criado e, igualmente, especializadas para contemplar necessidades específicas de cada processador. A Figura 3 apresenta os simuladores dos processadores Neander e MIPS I disponíveis no framework.

Em linhas gerais, a simulação de um modelo do T&D-Bench inicia com a carga de um programa em Assembly, por parte do usuário, e a sua execução ciclo a ciclo, instrução a instrução, ou por um número definido de ciclos ou instruções. A simulação pode ser acompanhada de diversas formas:

• para cada tempo simulado, o simulador atualiza o estado dos componentes estruturais do processador, conforme o comportamento que eles desempenham ao executar uma dada instrução. Isso pode ser visualizado na aba Caminho de Dados, onde se encontra o diagrama de blocos do processador;

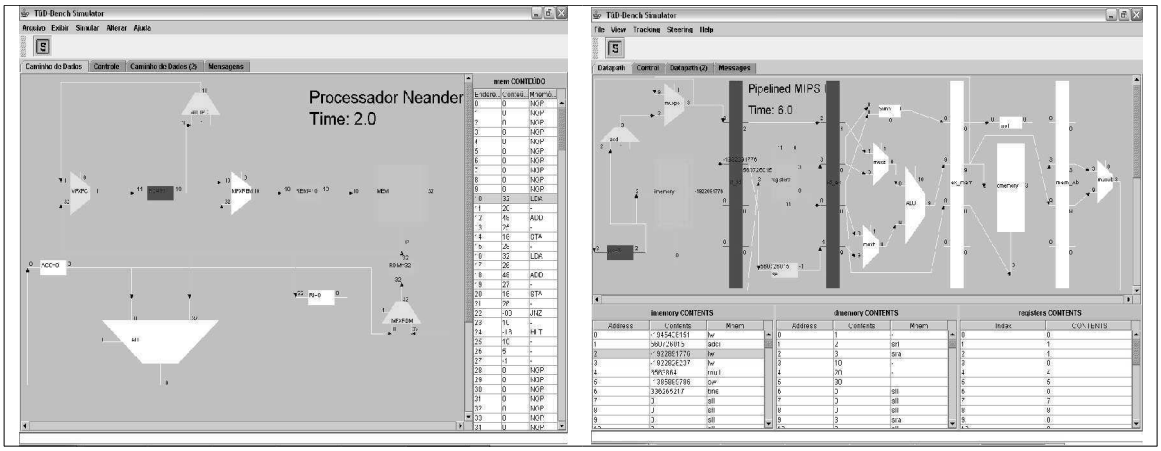

Figura 3. Alguns dos simuladores disponíveis

• as instruções e os dados do programa podem ser visualizados em painéis que mostram o conteúdo das memórias do processador, ainda na aba Caminho de Dados;

• na aba Controle, é possível visualizar o estágio atual de execução de uma instrução e os detalhes desta instrução, como *opcode* e demais campos;

• é possível, também, a qualquer tempo, verificar o estado de um componente estrutural específico (entradas, saídas, atributos e conteúdo) na aba Caminho de Dados (2). Seleciona-se o componente e o seu estado é apresentado.

Conclusão e Trabalhos Futuros

A metodologia de modelagem do T&D-Bench foi validada com a modelagem de tipos diferentes de processadores: Neander, FemtoJava (máquina de pilha), e MIPS (em versões monociclo, multiciclo, com pipeline e superescalar). Estes modelos vem sendo usados no ensino, com parecer favorável dos estudantes. Os modelos do Neander e do MIPS I estão disponíveis na página do projeto e prontos para serem usados, conforme descrito acima. A tradução da organização dos processadores para descrições em VHDL e a criação de modelos de multiprocessadores estão planejados como trabalhos futuros.

Referências

[PAT 03] PATT, Y.N. Teaching and Teaching Computer Architecture: Two Very Different Topics (Some Opinions about Each). In: WORKSHOP ON COMPUTER ARCHITECTURE EDUCATION, 2003, San Diego, Califórnia. **Proceedings...** San Diego: [s.n.], 2003. p. 1-4.

[SOA 04] SOARES, S. N.; WAGNER, F. R. Design Space Exploration using T&D-Bench. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, 16., 2004, Foz do Iguaçu, PR. **Proceedings...** Foz do Iguaçu: [s.n.], 2004. p. 40-47. Disponível em: < *http://www.ucs.br/carvi/cent/dpei/snsoares/TDBench*> Acesso em: 10 nov. 2005.

[WEB 01] WEBER, R. F. **Fundamentos de Arquitetura de Computadores**. 2.ed. Porto Alegre: Instituto de Informática da UFRGS: Sagra Luzzatto, 2001. 299 p. (Série Livros Didáticos, n. 8).