MODELAGEM FUNCIONAL DO MODO THUMB DO PROCESSADOR ARM

Paulo F. Kuss, Max R. de O. Schultz[*],
Olinto J. V. Furtado, Luis F. Friedrich, Luiz C. V. dos Santos

Departamento de Informática e Estatística
Universidade Federal de Santa Catarina
Florianópolis, SC, Brasil

{pkuss, max, olinto, fernando, santos}@inf.ufsc.br

ABSTRACT

Platform-based SoC design is the methodological response to the high non-recurring engineering costs of deep submicron VLSI technologies. Given a chosen platform, design space exploration is crucial to fulfill not only functional requirements, but also real-time, low-power and compact code constraints. Exploration of alternative processors requires CPU models in distinct abstraction levels. This work describes a functional model for the THUMB mode, the highly-used 16-bit instruction set of the most popular embedded processor, the ARM CPU. The model is manually written in the ArchC architecture description language (ADL). An executable model is automatically generated by the tool `acsim`. Well-known benchmark programs were run on the executable model for the sake of validation. Experimental results give evidence of the model's correctness and robustness.

1. INTRODUÇÃO

Os requisitos cada vez mais complexos das aplicações de sistemas embarcados, a redução no ciclo de desenvolvimento dos produtos eletrônicos, a conseqüente necessidade de se aumentar o ganho de produtividade e os altos custos de engenharia não recorrente das tecnologias VLSI contemporâneas são fatores que determinaram a adoção do projeto baseado em plataforma como paradigma para a concepção de *Systems-on-Chip* (SoCs) [8].

Uma vez escolhida uma plataforma adequada para um domínio de aplicação, a exploração de soluções alternativas é crucial não somente para atender os requisitos funcionais, mas também para satisfazer as restrições de tempo real, tamanho de código e consumo de potência.

A exploração de CPUs alternativas requer um acervo de modelos de processadores. Ademais, para garantir o necessário ganho de produtividade, é preciso que tais modelos estejam disponíveis em diferentes níveis de abstração. Modelos puramente funcionais ou funcionais com precisão de ciclos são usados durante a exploração do espaço de projeto devido ao menor tempo de simulação resultante, enquanto que modelos no nível RT (*Register-Transfer Level*) são preferencialmente usados nas etapas finais do fluxo de projeto. Embora resultem em um tempo de simulação muito maior, modelos no nível RT são mais adequados para a síntese.

Este trabalho propõe um modelo funcional para o conjunto de instruções do modo THUMB do processador embarcado com maior volume de venda dos últimos cinco anos [6], a CPU ARM.

Este artigo está organizado da seguinte forma: a Seção 2 revisa os trabalhos correlatos; a Seção 3 descreve o modelo do THUMB e suas principais características; a Seção 4 apresenta os resultados experimentais e as conclusões e perspectivas de trabalhos futuros são apresentadas na Seção 5.

2. TRABALHOS CORRELATOS

SystemC [12] é uma biblioteca de classes que estende a linguagem C++ para permitir a descrição de elementos típicos de hardware, tais como módulos, canais, portas e sinais. Assim, o projetista pode descrever de forma integrada o hardware e o software do sistema em uma única linguagem, o que facilita sua co-verificação.

Ademais, SystemC admite modelos em diferentes níveis e estilos de descrição, tais como: RT, comportamental, funcional, etc.

Em princípio, um modelo funcional para uma CPU pode ser descrito diretamente em SystemC [12] e, então,

[*] Amparado pelo Programa Nacional de Microeletrônica/CNPq, Proc. nº 132874/2005-9.

compilado para gerar um modelo executável. Entretanto, para maior ganho de produtividade, uma abordagem mais eficiente é a automação da geração do modelo. Para isso, costuma-se utilizar as assim-chamadas linguagens de descrição de arquiteturas (ADLs), concebidas especialmente para descrever CPUs [4] [11]. Uma ADL é uma linguagem cujas primitivas permitem a descrição do conjunto de instruções de uma CPU e, possivelmente, de algumas características e parâmetros de sua organização (comprimento da palavra de instrução, códigos operacionais, bancos de registradores, memórias, etc.).

A partir de uma descrição em ADL da CPU, seu modelo executável pode ser gerado automaticamente. Para este trabalho, escolhemos a ADL denominada ArchC [11], por ser uma linguagem de domínio público e por ser capaz de gerar automaticamente descrições em SystemC [12], a partir das quais modelos executáveis são obtidos através de mera compilação.

A partir de uma descrição ArchC, vários geradores de ferramentas estão disponíveis [11]: gerador de simuladores interpretados (`acsim`), gerador de simuladores compilados (`accsim`), gerador de montadores (`acasm`), gerador de linkeditores (`aclink`).

Há relatos de utilização da ADL EXPRESSION [4] para a geração de compilador, simulador e montador para o processador ARM [5].

Um modelo comportamental com precisão de ciclos do processador ARM é descrito em [10] com o uso de *Abstract State Machines*, a partir das quais gera-se um modelo executável do processador para simulação.

Em [7] foi proposto um modelo genérico de representação para acomodar as variações de conjuntos de instruções e formatos binários da família de processadores ARM. Dada a descrição de uma arquitetura ARM alvo, obtida através da especialização do modelo genérico, é possível gerar-se automaticamente um simulador funcional daquela arquitetura. Ou seja, trata-se de uma ferramenta redirecionável para geração de modelos funcionais, mas dedicada à família de arquiteturas ARM.

Embora essa abordagem permita alguma exploração de alternativas, ela se restringe à família de CPUs ARM. Para fins de uma exploração mais genérica de CPUs alternativas, é preciso que um modelo do ARM (THUMB) seja elaborado no mesmo *framework* onde modelos de outras CPUs estejam disponíveis para efeito comparativo. Essa é uma razão para a adoção de uma ADL como *framework* para construção de modelos.

Este trabalho contribui com um novo modelo para o repositório da ADL ArchC: o modelo do modo THUMB do processador ARM. Um modelo funcional do ARM5 está sendo simultaneamente desenvolvido pela UNICAMP [11]. Os modelos, depois de devidamente validados, serão integrados em um único modelo.

3. DESCRIÇÃO DO MODELO THUMB/ARM

O conjunto de instruções THUMB existe para atender à demanda de compactação de código em sistemas embarcados. Ele pode ser visto como uma forma comprimida de um subconjunto de instruções do ARM [2].

O hardware do processador ARM não executa as instruções THUMB diretamente. Estas são dinamicamente descomprimidas e mapeadas para instruções ARM, que são as instruções efetivamente executadas [9].

Processadores ARM (com suporte apropriado) interpretam as instruções segundo os formatos de 16 e 32 bits, caracterizando os modos de operação THUMB e ARM, respectivamente. A distinção é feita através de um bit do registrador de status (o bit T do registrador CPSR).

Se a CPU estiver operando em modo THUMB, pode-se comandar uma transição explícita para o modo ARM, através da instrução *Branch and Exchange* (`BX`). Ademais, uma transição implícita para o modo ARM ocorre sempre que houver exceções, pois estas são sempre tratadas no modo ARM.

O programador pode acessar diretamente oito registradores de uso geral no modo THUMB (`R0 a R7`), assim como o *program counter* (`PC`), o *stack pointer* (SP), o *link register* (LR), e o registrador de status (CPSR). Os registradores R8 a R16 não fazem parte do conjunto de registradores padrão no modo THUMB, mas podem ser usados para armazenamento temporário. O modo THUMB possui 36 instruções codificadas em 19 formatos distintos [2].

3.1. O desenvolvimento do modelo

Uma descrição de arquitetura em ArchC é dividida em duas partes: a descrição do conjunto de instruções da arquitetura (`AC_ISA`) e a descrição dos elementos da arquitetura (`AC_ARCH`). Na descrição `AC_ISA`, o projetista fornece informações sobre o conjunto de instruções, tais como mnemônicos, formatos, tamanhos, e códigos para decodificação. Na descrição `AC_ARCH` o projetista descreve os componentes principais da arquitetura, como os módulos e estruturas de armazenamento.

A partir destas duas descrições, o pré-processador ArchC (`acpp`) gera o esqueleto do simulador da arquitetura, que contém as assinaturas dos métodos que especificam o comportamento de cada instrução. Em seguida, define-se o comportamento comum a todas as instruções e o comportamento específico de cada formato de instrução [11].

3.2. A estrutura do modelo

Não seria prático mostrar a descrição completa do modelo neste artigo. Portanto, as figuras desta seção são versões condensadas da descrição real para fins de visualização de como se organiza seu conteúdo.

A Figura 1 mostra a estrutura das instruções, enquanto que a Figura 2 ilustra seu comportamento.

```
AC_ISA(thumbv1){

/*-- Section A - Type Declarations --*/
/*-- Data processing instructions --*/
/* Add/subtract register */
  ac_format Type_ASR = "%rdASR:3 %rnASR:3
     %rmASR:3 %opcASR:1 %funcASR:6";

/*-- Section B - Type with Instructions --*/
/*-- Data processing instructions --*/
  ac_instr<Type_ASR> mov2;

/*-- Section C - Registers Declarations --*/
  ac_asm_map reg {
    "$"[0..17] = [0..17];
    "$r"[0..12] = [0..12];
    "$sp" = 13;
    "$lr" = 14;
    "$pc" = 15;
    "$cpsr" = 16;
    "$spsr" = 17;
  }

/*-- Section D - Assembly Sintax and
Operacional Codes --*/
ISA_CTOR(thumbv1){
/* Add/subtract register */
  mov2.set_asm("mov %reg, %reg", rdASR,
     rnASR);
  mov2.set_decoder(opcASR=0x00, funcASR=0x07,
     rmASR=0x00);
```

Figura 1 – Modelo do THUMB: estrutura

```
/* ASR Instructions behavior methods - Mov */
void ac_behavior( mov2 ) {
  dbg_printf("mov r%d, r%d\n", rdASR, rnASR);
  dbg_printf("mov2 %#x, %#x\n", RB[rdASR],
     RB[rnASR]);

  RB[rdASR] = RB[rnASR];
  flags.N = getBit(RB[rdASR],31);
  flags.Z = ((RB[rdASR] == 0) ? true :
     false);

  dbg_printf("Result: %#x\nFlags: N=%d,
     Z=%d\n",RB[rdASR],flags.N,flags.Z);
};
```

Figura 2 – Modelo do THUMB: comportamento

A descrição da Figura 1 consiste essencialmente de três seções, que descrevem diferentes aspectos da arquitetura do conjunto de instruções. Apenas um dos dezenove formatos é ilustrado, o formato `ASR`, denominado *add-substract-register* (*Section A*). Cada instrução é associada com seu respectivo formato (*Section B*), cada registrador com seu nome simbólico ou número respectivo (*Section C*) e a cada instrução é atribuída uma sintaxe assembly e um código operacional (*Section D*).

O comportamento da instrução `MOV` no formato `ASR (mov2)` é ilustrado na Figura 2.

3.3. Limitação do modelo

A descrição desenvolvida e o respectivo modelo executável permitem a simulação de qualquer programa que contenha apenas instruções THUMB. Assim, o modelo não suporta programas que façam transições do modo THUMB para o modo ARM. Esta limitação será eliminada quando este modelo for integrado ao modelo do ARM5, atualmente em desenvolvimento.

4. RESULTADOS EXPERIMENTAIS

4.1. Configuração experimental

Os experimentos foram executados em um computador com CPU Intel® Pentium 4 (1.8 GHz), com 256 MB de memória principal. O sistema operacional utilizado foi o Mandrake GNU/Linux. Para a geração do simulador, utilizou-se a ADL ArchC, versão 1.5.1. Utilizou-se também o *cross-compiler* `ARM GCC` versão 3.3.1 [1] que, através dos parâmetros `-EL -mthumb`, determina a configuração *little endian* e força a geração de código somente com instruções THUMB, tanto no formato *assembly* (para inspeção da sintaxe das instruções geradas) quanto no formato binário (para interpretação pelo simulador gerado pela ferramenta `acsim` do pacote ArchC [11]).

Há que se fazer uma ressalva. Como o modelo proposto representa somente o modo THUMB (a ser oportunamente integrado com trabalho correlato focado no modo ARM), há uma dificuldade operacional em se modelar o término de um programa. Na prática, a CPU é transicionada do modo THUMB para o modo ARM, através da instrução `BX` (*Branch and Exchange*). Assim, utilizamos uma solução provisória: a instrução `BX` é codificada para forçar o término da simulação, sem perda de generalidade para a modelagem do modo THUMB. Obviamente, esta limitação será removida quando os modelos de ambos os modos forem integrados.

Os tempos de simulação associados aos *benchmarks* foram obtidos através de estatísticas geradas a partir do conjunto de ferramentas de ArchC, que são expressos em unidades de tempo de SystemC (*default time units*).

4.2. Validação do modelo

Inicialmente, cada instrução foi testada e depurada isoladamente. Um programa *assembly* contendo todas as instruções do THUMB foi manualmente desenvolvido, montado e simulado para fins de depuração preliminar do modelo.

Em seguida, uma validação mais ampla foi realizada através de experimentos baseados em programas extraídos do *benchmark* do Projeto Dalton [3]. Cada programa do *benchmark* foi compilado conforme descrito na Seção 4.1. O código objeto resultante foi alimentado no simulador gerado pela ferramenta `acsim` [11], a partir da descrição funcional do modo THUMB.

A Tabela 1 apresenta os resultados obtidos para cada um dos programas do *benchmark*.

Tabela 1 – Resultados para os programas do *benchmark*

Programa	Instruções executadas	Tempo de simulação	Tamanho de código (bytes)
Negcnt	168	0,12	34489
Gcd	242	0,21	34515
Int2bin	189	0,14	34483
Cast	96	0,06	34568
Divmul	304	0,23	34562
Fib	602	0,57	34681
Sort	2388	2,16	36558
Xram	4244	4,13	34586
Sqroot	1379	1,23	133435

Os resultados esperados foram obtidos para todos os programas testados do *benchmark* escolhido.

5. CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho contribui com um importante insumo para o projeto ArchC: a modelagem das instruções mais compactas da mais popular CPU embarcada. O modelo do THUMB mostrou-se robusto em face da variedade de experimentos a que foi submetido com sucesso.

Entretanto, o modelo puramente funcional permite apenas avaliar a adequação aos requisitos funcionais e ao tamanho de código. O modelo precisa ser estendido para capturar o *timing* das instruções, viabilizando estimativas de desempenho. Um modelo com precisão de ciclos será objeto de trabalho futuro.

A integração do modelo descrito com o modelo da CPU ARM5 – atualmente em desenvolvimento – será feita oportunamente.

REFERÊNCIAS

[1] ARM Linux Toolchains. Disponível em http://ftp.arm.linux.org.uk/pub/armlinux/toolchain/, Nov. 2005.

[2] ARM, The Architecture for the Digital World. Disponível em http://www.arm.com, Nov. 2005.

[3] Dalton Project. Disponível em http://www.cs.ucr.edu/~dalton/i8051/i8051syn/.

[4] Halambi, A., et al., "EXPRESSION: A language for architecture exploration through compiler/simulator retargetability," in Proc. of the DATE Conference, Mar. 1999.

[5] Kejariwal, A., et al., "HDLGen: Architecture Description Language driven HDL – Generation for Pipelined Processors," CECS Technical Report, University of California, Irvine, pp. 03-04, 2003.

[6] Patterson, D., Hennessy, J., *Computer Organization and Design: The Hardware/Software Interface*, 3rd ed. Morgan Kaufmann Publishers, pp. 7-8, 2004.

[7] Reshadi, M., et al. "An efficient retargetable framework for instruction-set simulation," Proc. Int. Conference on Hardware/Software Co-design and System Synthesis, pp. 13-18, 2003.

[8] Sangiovanni-Vincentelli, A., Martin, G. "Platform-based design and software design methodology for embedded systems". **IEEE Design & Test of Computers**, v.18, pp. 23-33, November-December 2001.

[9] Sloss, A., et al. *ARM System Developer's Guide - Designing and Optimizing System Software.* Morgan Kaufmann Publishers, 2004.

[10] Teich, J., et al. "Description and simulation of microprocessor instruction sets using ASMs". Lecture Notes in Computer Science, n. 1912, Springer-Verlag, pp. 266-286, October 2000.

[11] The ArchC Architectural Description Language. Disponível em http://www.archc.org.

[12] The Open SystemC Initiative. Disponível em http://www.systemc.org.