2
Trabalhos Relacionados

Com o intuito de conhecer e aprofundar os conhecimentos sobre o conceito de processadores de rede são apresentados a seguir, três trabalhos relacionados ao tema, onde são ilustradas algumas perspectivas do desenvolvimento de trabalhos relacionados.

A fim de apresentar as comparações, o Capítulo está estruturado da seguinte forma: a descrição de arquitetura de processadores (ARCHC, 2001) baseada na linguagem *SystemC* (SYSTEMC, 1999) é analisada na Seção 2.1. A Seção 2.2 apresenta as principais características no uso de Reescrita Lógica utilizada na Especificação de Processadores sobre arquiteturas simples (Ayala-Rincón et al., 2003). Finalmente, a Seção 2.3 analisa uma proposta para simulação de processadores (Freitas & Martins, 2002). Ao final de cada Seção, são feitas algumas comparações de cada um desses trabalhos com o aqui proposto.

2.1.
Arch C

O ArchC (ARCHC, 2001) é um sistema que propõe uma linguagem para descrição de arquiteturas de processadores baseada na linguagem *SystemC*. (SYSTEMC, 1999). Com ArchC é possível descrever processadores tanto na forma comportamental (apenas indicando o que cada instrução faz) quanto na forma de desempenho, com precisão de ciclos (indicando o que cada instrução faz em cada ciclo do pipeline). Com base nessa descrição, as ferramentas da linguagem são capazes de gerar simuladores para os processadores.

Atualmente existem inúmeros trabalhos em desenvolvimento utilizando esta linguagem de descrição de arquiteturas de processadores, entre os quais podemos destacar:

- Simulação Compilada: técnica que permite a geração de um simulador muito mais rápido que o convencional (entre 100x e

1000x mais rápido);

- Modelagem de Múltiplos Processadores: suporte ao uso de mais de um processador e também ao mapeamento de software entre eles;
- Suporte a Processadores Super-escalares: inclusão de primitivas na linguagem e nos simuladores para dar suporte a modelos de processadores super-escalares;

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 de AC_ISA o projetista fornece todos os detalhes sobre o conjunto de instruções como: nomes das instruções, formatos, tamanhos, bem como a descrição de cada instrução e a informação requerida para decodificá-la. Na descrição de AC_ARCH o projetista lista todos os recursos da arquitetura como os módulos, estruturas de armazenamento, etc.

2.1.1. Descrição do AC_ARCH

O AC_ARCH usa a estrutura de informações sobre os recursos disponíveis na arquitetura a fim gerar automaticamente um simulador. O projetista deve fornecer tal informação na descrição (AC_ARCH), que é composta basicamente de elementos de armazenamento e declarações. O nível do detalhamento usado nesta descrição dependerá do nível da abstração que o projetista desejar para seu modelo.

Por exemplo, querer simular o conjunto de instruções da arquitetura dos MIPS, sem considerar seu pipeline. Isto torna a descrição do comportamento de cada instrução muito simples e exige também pouca informação estrutural.

Por outro lado, para modelos com ciclos de execução mais detalhados, o projetista deve fornecer ArchC com mais detalhes sobre a estrutura do processador, como mostrado na Figura 2-1.

```
AC_ARCH(mips){
ac_wordsize 32;
ac_mem MEM:256K;
ac_regbank RB:34;
ac_pipe pipe = {IF, ID, EX, MEM, WB};
ac_format Fmt_IF_ID = "%npc:32";
ac_format Fmt_ID_EX =
 "%npc:32 %data1:32 %data2:32 %imm:32:s rs:5 %rt:5 %rd:5
  %regwrite:1 %memread:1 %memwrite:1" ;
ac_format Fmt_EX_MEM =
  "%alures:32 %wdata:32 %rdest:5 %regwrite:1 %memread:1
   %memwrite:1";
ac_format Fmt_MEM_WB = "%wbdata:32 %rdest:5 regwrite:1";
ac_reg<Fmt_IF_ID> IF_ID;
ac_reg<Fmt_ID_EX> ID_EX;
ac_reg<Fmt_EX_MEM> EX_MEM;
ac_reg<Fmt_MEM_WB> MEM_WB;
ARCH_CTOR(mips) {
ac_isa("mips_isa.ac");
set_endian("big");
};
};
```

Figura 2-1 - Exemplo de Descrição do AC_ARCH

2.1.2. Descrição do AC_ISA

A descrição de AC_ISA fornece à ArchC toda a informação que necessita para sintetizar automaticamente um decodificador, junto com a descrição de cada instrução na arquitetura. Esta descrição é dividida em dois arquivos: um contém a instrução e o formato das declarações; e o outro contém a descrição das instruções.

```
AC_ISA(mips){
  ac_format Type_R = "%op:6 %rs:5 %rt:5 %rd:5 0x00:5
%func:6";
  ac_format Type_I = "%op:6 %rs:5 %rt:5 %imm:16:s";
  ac_format Type_J = "%op:6 %addr:26";
  ac_instr<Type_R> add, sub, instr_and, instr_or, mult,
div;
  ac_instr<Type_R> mfhi, mflo, slt, jr;
  ac_instr<Type_R> addu, subu, multu, divu, sltu;
  ac_instr<Type_R> sll, srl;
  ac_instr<Type_I> load, store, beq, bne;
  ac_instr<Type_I> addi, andi, ori, lui, slti;
  ac_instr<Type_I> addiu, sltiu;
  ac_instr<Type_J> j, jal;
  ISA_CTOR(mips){
    load.set_asm("lw %rt, %imm(%rs)");
    load.set_decoder(op=0x23);
    store.set_asm("sw %rt, %imm(%rs)");
    store.set_decoder(op=0x2B);
    add.set_asm("add %rd, %rs, %rt");
    add.set_decoder(op=0x00, func=0x20);
    addu.set_asm("addu %rd, %rs, %rt");
    addu.set_decoder(op=0x00, func=0x21);
...
  };
};
```

Figura 2-2 - Exemplo de Descrição do AC_ISA

A Figura 2-2 mostra um exemplo de uma descrição de AC_ISA extraída de um modelo MIPS. Contém instruções, formatos, informação de decodificação e declarações da sintaxe do conjunto, ilustrando as principais características de uma descrição de ISA em ArchC. Cada instrução deve ter um formato previamente declarado associado a ela. O projetista declara uma instrução

através da palavra *ac_instr* e pode atribuir-lhe um formato usando moldes do estilo de C++.

No exemplo, o formato *Type_R* é associado à instrução *add*. Isto permite que o projetista acesse cada campo da instrução individualmente ao descrever as descrições da instrução.

Embora o ArchC seja uma poderosa ferramenta que permite a descrição de arquiteturas de processadores, nos deparamos com o problema da dependência da linguagem *SystemC*. (SYSTEMC, 1999), que dessa forma amarra a sintaxe de utilização da ferramenta em função das restrições de código possíveis de serem implementados pelo *SystemC*. Já na biblioteca proposta pelo presente trabalho fornece-se ao usuário um conjunto de classes prontas a partir das quais ele estará livre para realizar a sua implementação em função de suas necessidades, ficando limitado somente ao escopo de sua própria implementação.

2.2. Reescrita Lógica na Especificação de Processadores (ELAN)

O ELAN (Ayala-Rincón et al., 2003) é um ambiente de Reescrita Lógica na Especificação de Processadores, outra técnica importante para a simulação de processadores sobre arquiteturas simples, para verificar e testar qualquer projeto de novas tecnologias previamente, ao invés de custosas implementações em hardware.

Com o ELAN é possível especificar processadores simples através de sistemas de reescrita de termos. A reescrita lógica pode ser aplicada a outros propósitos nesse contexto, como no caso da simulação e avaliação do desempenho de características importantes dos processadores idealizados. O ELAN é suficientemente versátil para permitir implementações de fácil modificação, que tornam viável uma análise de componentes relacionados com o projeto de processadores, como por exemplo, o tamanho e o controle de buffers de reordenação e mecanismos de predição utilizados por processadores.

```
[Loadc] Sys(m,Proc(ia,rf,prog)) =>
 Sys(m,Proc(ia+1,insertRF(rf,r,v),prog))
  where instIa :=() selectinst(prog,ia)
  if isinstLoadc(instIa)
  where r :=() nameofLoadc(instIa)
  where v :=() valueofLoadc(instIa)            end
[Loadpc] Sys(m,Proc(ia,rf,prog)) =>
 Sys(m,Proc(ia+1,insertRF(rf,r,ia),prog))
  where instIa :=() selectinst(prog,ia)
  if isinstLoadpc(instIa)
  where r :=() nameofLoadpc(instIa)            end
[Op] Sys(m,Proc(ia,rf,prog)) =>
 Sys(m,Proc(ia+1,insertRF(rf,r,v),prog))
  where instIa :=() selectinst(prog,ia)
  if isinstOp(instIa)
  where r1 :=() reg1ofOp(instIa)
  where r2 :=() reg2ofOp(instIa)
  where r :=() nameofOp(instIa)
  where v:=() valueofOp(r1,r2,rf)              end
[Jz] Sys(m,Proc(ia,rf,prog)) =>
 Sys(m,Proc(nia,rf,prog))
  where instIa :=() selectinst(prog,ia)
  if isinstJz(instIa)
  where r1:=() reg1ofJz(instIa)
  where r2:=() reg2ofJz(instIa)
  choose try where nia:=()ia+1
              if valueofReg(r1,rf)!=0
         try where nia:=()valueofReg(r2,rf)
              if valueofReg(r1,rf)==0
  end                                          end
[Load] Sys(m,Proc(ia,rf,prog)) =>
 Sys(m,Proc(ia+1,insertRF(rf,r0,v0),prog))
  where inst :=() selectinst(prog,ia)
  if isinstLoad(inst)
  where r0 :=() nameofLoad(inst)
  where v0 :=() getMem(inst,rf,m)              end
[Store] Sys(m,Proc(ia,rf,prog)) =>
 Sys(insertMEM(m,valueofReg(rA,rf),
 valueofReg(rB,rf)),Proc(ia+1,rf,prog))
  where inst :=() selectinst(prog,ia)
  if isinstStore(inst)
  where rA :=() nameofStoreR1(inst)
  where rB :=() nameofStoreR2(inst)            end
```

Figura 2-3 - Exemplo de Especificação de Um Processador Básico com ELAN

O ambiente de Reescrita Lógica na Especificação de Processadores permite, como o próprio nome diz, a simulação de processadores sobre arquiteturas simples através de reescrita de termos. Essa reescrita lógica é completamente dependente do conjunto de instruções do processador alvo da implementação o que exige o conhecimento específico de determinado conjunto de instruções (deste processador), bem como limita a reescrita em função da restrição deste conjunto de instruções.

No caso da biblioteca aqui fornecida, buscou-se algo semelhante, mas sempre deixando em aberto a possibilidade do usuário definir, ele próprio, todo o conjunto de instruções que ele julgasse necessário, sem as limitações apontadas.

2.3. Simulador NPSIM

O Network Processor Simulator (NPSIM) (Freitas & Martins, 2002) foi desenvolvido com o objetivo de simular e verificar o funcionamento de um processador de rede, chamado RCNP (ReconFigurable CISC Network Processor). A principal característica desse processador de rede é o seu suporte à comunicação de dados, estabelecendo e efetivando a comunicação entre os diversos equipamentos e dispositivos de rede. O RCNP foi desenvolvido para validação de conceitos relacionados à micro-arquitetura de rede. Além ser um software de simulação funcional, o NPSIM também pode ser utilizado como software de aprendizado, que é o seu foco principal.

Com o RCNP é possível escrever e executar diversos algoritmos escritos em linguagem assembly e visualizar a execução e os resultados através dos registradores, pilhas e matrizes, representados pelos diversos componentes existentes no compilador C++ Builder 5.0. Também e possível estudar o comportamento de um processador, escrever algoritmos de roteamento e oferecer aos desenvolvedores, a possibilidade de escrever compiladores para o processador virtual (simulador) de rede, tal como uma máquina virtual. A interface gráfica é dividida em sete partes:

- Memória, Registradores e Botões de Acesso Rápido:
- Montador (janela de edição do programa assembly)
- Buffers Permanentes (Buffers reconfiguráveis da porta de entrada)
- Buffers Temporários (oito Buffers fixos de entrada)
- Pacotes de Entrada (janela de edição de pacotes)
- Seletor de Conexão (matriz de comutação entre entrada e saída)
- Registradores de DMA (acesso direto à memória)

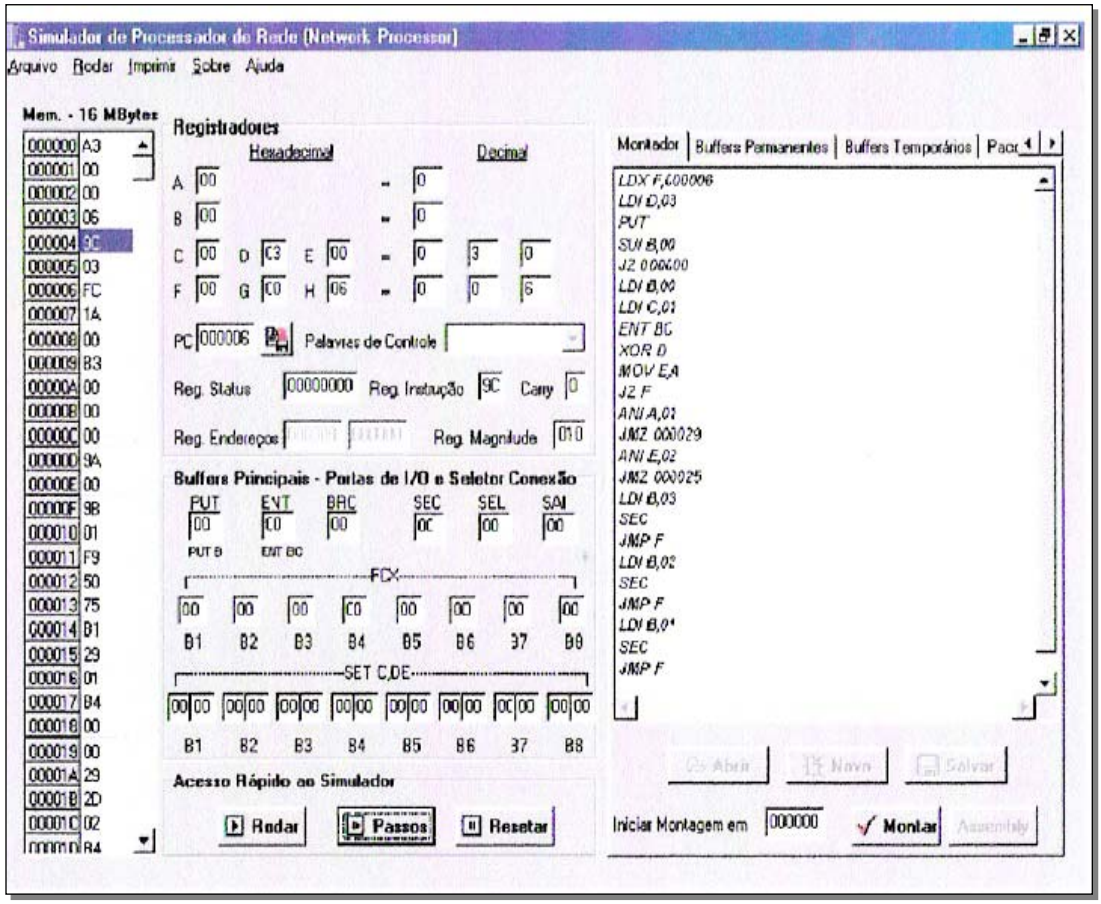

Figura 2-4 - Interface Principal do NPSIM

Apenas a primeira interface é fixa. Nela está contida a modelagem da memória, registradores e os botões de execução, todos sempre à esquerda na tela do simulador, para que o usuário possa visualizar os resultados que são comuns a todas as outras interfaces. O NPSIM embora permita simular o funcionamento de um processador de rede, foi desenvolvido para ser executado a partir de uma arquitetura proprietária, chamada RCNP (Reconfigurable CISC Network Processor). Apesar de ser um software de simulação, sua implementação foi focada no aprendizado, sendo possível escrever e executar diversos algoritmos escritos em linguagem assembly. Isto torna sua implementação muito parecida com a proposta do presente trabalho, já que ele também permite escrever compiladores para um processador virtual de rede. A grande diferença reside no fato de que, no NPSIM, apenas se pode implementar simulações baseadas no conjunto de instruções de processadores de rede do CISCO, ao contrário do que é proposto neste trabalho, onde a simulação, embora mais complexa de ser implementada, pode ser realizada sobre o processador que o usuário projetista puder criar a partir da biblioteca fornecida.