ALGORITMOS DE SIMULAÇÃO DE HARDWARE NO NÍVEL RT

por

Flávio Rech Wagner

RT nº 021 CPGCC-UFRGS SET/85

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Av. Osvaldo Aranha, 99
Caixa Postal 1501
90.000 - Porto Alegre - RS - Brasil
Telex (051) 2680 Tel. (0512) 21.8499

AGRADECIMENTOS

Este trabalho foi parcialmente desenvolvido durante estágio de doutoramento na Universidade de Kaiserslautern, República Federal da Alemanha, junto ao Prof. R. Hartenstein, ao qual agradeço pela motivação e apoio. Agradeço ainda ao CNPq pelo apoio fornecido a todo o desenvolvimento deste trabalho.

RESUMO

Este trabalho cria uma taxonomia para classificação de algoritmos de simulação de sistemas digitais descritos no nível de transferência entre registradores. Os algoritmos de simulação são apresentados e sua eficiência é discutida brevemente de forma qualitativa.

PALAVRAS-CHAVE: simulação de sistemas digitais, nível de transferência entre registradores.

ABSTRACT

This work presents a taxonomy for the classification of simulation algorithms for digital systems described at the register-transfer level. The simulation algorithms are presented and their efficiency is briefly discussed in a qualitative way.

KEYWORDS: digital systems simulation, register-transfer level.

SUMÁRIO

1. INTRODUÇÃO

A literatura oferece raras referências a respeito de algoritmos de simulação para sistemas digitais descritos no nível de transferência entre registradores (RT) /BAR 75/. Esta escassez se opõe frontalmente ao caso da simulação no nível de portas lógicas, onde podemos encontrar descrições bastante detalhadas dos algoritmos de simulação e da estrutura interna dos simuladores /WAG 84/.

A explicação para esta situação é simples. Simuladores para o nível RT estão de modo geral associados a linguagens de descrição de hardware /BAR 75/. O nível RT diferencia-se basicamente do nível lógico por introduzir explicitamente construções de controle para explicar o comportamento do timing dos sistemas. Enquanto no nível lógico sinais são tratados indistintamente, carreguem eles informações de dados ou de controle, no nível RT a estrutura de controle é declarada explicitamente, permitindo-se assim uma visão do paralelismo e da sequencialização das ações no sistema. Cada linguagem de descrição de hardware oferece uma estrutura de controle própria, em função de uma visão particular do nível RT. Assim p.ex.:

- a construção "next" em ISPS /BAR 81/, que separa grupos de comandos simultâneos, dá à descrição um controle global assíncrono;

- a construção "state" em DDL /DUL 68/ força a descrição de um sistema como sendo uma máquina de estados finitos, sendo a cada estado associado um grupo de comandos;

- a construção "label" em CDL /CHU 70/, que identifica grupos de comandos simultâneos, permite tanto um controle global assíncrono como síncrono (caso a condição de habilitação associada ao label seja função de um clock).

Um simulador para cada uma destas linguagens será certamente construído em função da estrutura de controle particular da linguagem. Por este motivo, de modo geral, simuladores RT são particulares e não podem ser aproveitados noutro contexto.

Este trabalho pretende apresentar diferentes algoritmos de simulação de hardware para o nível RT, que são genéricos e independentes de uma linguagem de descrição. Tal abordagem só é possível se eles não forem condicionados por nenhuma estrutura de controle particular. Como tais estruturas de controle implicam numa descrição comportamental que abstrai detalhes da estrutura real dos sistemas, este trabalho se limitará a considerar a simulação de descrições puramente estruturais.

Descrições RT para as quais são válidos os algoritmos aqui apresentados têm então as seguintes características:

- cada elemento a ser simulado é um elemento real do hardware no nível RT. Registradores, somadores, decodificadores, multiplexadores, gates básicos (AND, OR, NOT) são exemplos típicos.

- os sistemas são síncronos, i.e., existe um clock central responsável pela carga de todos os elementos com memória. Este clock pode ser de uma ou várias fases.

- não existem laços assíncronos de realimentação na lógica combinacional.

- os elementos são totalmente livres de delay, i.e., o único timing considerado é o implicado pelo clock. Elementos combinacionais reagem instantaneamente a varia

ções em suas entradas, e elementos com memória apresentam um novo valor em suas saídas imediatamente após a carga.

Estas suposições são perfeitamente válidas para o nível RT e os propósitos de projeto e simulação neste nível /HAY 78/, considerando-se as restrições impostas no âmbito deste trabalho à natureza das descrições.

Apenas registradores sensíveis à borda são considerados. Registradores "master-slave" podem ser facilmente incluídos se cada fase do clock (e portando cada passo de simulação, considerando-se um passo para cada fase) for dividida em duas sub-fases e se a cada registrador forem associadas duas posições de memória, cada uma carregada numa sub-fase. A posição "master" recebe o valor da entrada na primeira sub-fase e a posição "slave" recebe o valor da posição "master" na segunda sub-fase. Registradores "latch" implicam num tratamento especial, que no entanto em nada afeta as filosofias básicas de simulação.

2. ESTRATÉGIA GLOBAL DE SIMULAÇÃO

Em princípio hardware pode ser visto como na Fig. 1, i.e., como uma interconexão de elementos que pode assumir qualquer configuração. Ao se modelar hardware no nível RT um destes elementos será um oscilador que fornece um clock para o resto do sistema. Este clock é ligado a todos os registradores, supondo-se simplificadamente clock de uma única fase e que RAM's sejam consideradas como um conjunto de registradores isolados. Este tipo de modelo é denominado "rede com clock explícito" /WEN 79/ e o algoritmo de simulação correspondente usa o princípio de "selective trace" /WAG 84/: uma alteração num sinal provoca a avaliação dos elementos a cujas entradas ele está conectado. Como resultado destas avaliações, novos sinais podem ser alterados, provocando a avaliação de novos elementos, até que o sistema alcance uma situação de repouso, na qual nenhum elemento avaliado tem seu sinal de saída alterado. Imaginando-se simulação dirigida por eventos /WAG 84/, pode-se modelar o avanço do tempo como um evento programado pelo oscilador, que provoca a sua própria chamada no início do tempo seguinte. Este algoritmo de simulação é resumido na Fig. 2.

Este modelamento no entanto não é ótimo para sistemas que possuem as características relacionadas na introdução, pois não considera o clock como um sinal com propriedades distintas dos demais sinais (ele está associado ao avanço do tempo e está conectado apenas a uma classe de elementos, os registradores), nem considera a divisão dos elementos em duas classes com propriedades distintas (registradores e elementos combinacionais). Dividimos então os elementos nas classes M (elementos com memória de leitura/escrita) e C (elementos combinacionais e com memória apenas de leitura). O clock não precisa aparecer explicitamente como um sinal no modelo, já que a di-

visão dos elementos em duas classes implica em que todos os elementos da classe M recebem o clock. Por isto chamamos este modelo de "rede com clock implicito" /WEN 79/.

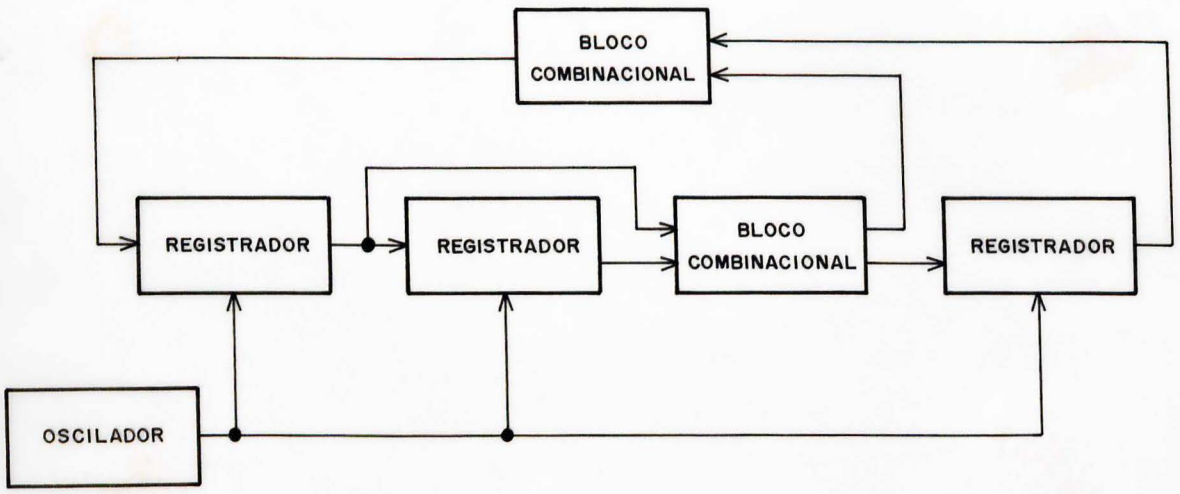


Figura I. - Hardware visto no Nível RT.

Existem dois tipos de algoritmos para a simulação da classe C, que definem também a estratégia global para o tratamento combinado das classes M e C. Na estratégia de simulação por avanço a cada passo de simulação a classe M é processada inteiramente em primeiro lugar. As conseqüências das alterações nas saídas dos registradores são então determinadas para os elementos da classe C, de tal modo que ao final do passo todos os sinais do circuito estão atualizados. Isto significa que os valores das entradas de dados para os registradores são conhecidos quando do passo seguinte, podendo eles serem carregados imediatamente. Na estratégia de simulação por recuo, o processamento das duas classes é intercalado. Os elementos da classe M são varridos: para cada elemento é avaliada apenas a parte da classe C que interessa para a sua carga, de modo que percorre-se para trás o caminho de entrada de dados de cada registrador.

```
algoritmo principal de simulação;
begin
avance o tempo até o tempo do evento mais próximo;
para cada evento neste tempo
    begin
    valor-sinal :=  novo-valor anotado no evento;
    chame  rotina de avaliação do elemento anotado no evento;
    para cada elemento conectado a este sinal
        coloque nome do elemento na lista de chamada

para cada elemento na lista de chamada
    begin
    avalie função do elemento;
    se novo valor da saída é diferente do valor anterior
    então se elemento tem delay zero
          então begin
                valor-sinal := novo-valor;
                para cada elemento conectado a este sinal
                    Coloque nome do elemento na lista de chamada
                end
          senão programe evento (elemento, novo-valor, tempo-futuro)
                   de acordo com o delay
end

rotina de avaliação do oscilador;
begin
    se saída = 1
    então begin
          saída := 0;
          programe evento (oscilador, saída = 1, tempo = tempo atual +1)
          end
    senão begin
          saída := 1;
          programe evento (oscilador, saída = 0, tempo = tempo atual +1)
          end
```

Fig. 2 - Algoritmo de Simulação para Redes com Clock Explícito

Este modelo com clock implícito pode ser facilmente expandido para um clock de várias fases. É então necessário distinguir os elementos M em subclasses M1,...,Mn, uma para cada uma das n fases do clock CP1,...,CPn. O algoritmo de simulação faz uma varredura das fases do clock: a cada passo são avaliados os elementos da classe Mi (que corresponde à fase atual CPi do clock) e da classe C.

Como geralmente associa-se uma rotina de processamento a cada elemento, usaremos de forma equivalente ao longo deste trabalho as expressões "chamar" e "carregar", para elementos da classe M, e "chamar" e "avaliar", para elementos da classe C.

Deve ficar claro que redes de clock implícito são ótimas para o modelamento de sistemas RT que se adequem perfeitamente às características expostas na introdução. Registradores carregados por sinais que não o clock central, lógica combinacional assíncrona, delays, etc., criam um problema de modelamento para as redes de clock implícito e tornam os algoritmos aqui estudados não ótimos, podendo ser preferível o uso do modelo de redes com clock explícito.

3. SIMULAÇÃO DA CLASSE M

Existem dois problemas a serem considerados na simulação da classe M: a) quais elementos devem ser carregados com um novo valor, dentre aqueles pertencentes à classe (ou sub-classe, caso tenhamos clock com várias fases); b) para os elementos que devem ser carregados, que seqüência de carga deve ser adotada.

3.1 Seqüência de carga

Do ponto de vista dos elementos da classe M que devem ser carregados no passo atual, o sistema pode ser visto como na Fig. 3, i.e., como vários conjuntos disjuntos G1,...,Gn de elementos a serem considerados, separados entre si por conjuntos H1,...,Hm de elementos da classe C ou de outros elementos da classe M, não ativos neste passo. Qualquer seqüência de grupos G pode ser adotada, pois os novos valores carregados nos registradores contidos num grupo Gi num passo não afetam as entradas dos resgistradores contidos noutro grupo Gj ainda neste mesmo passo.

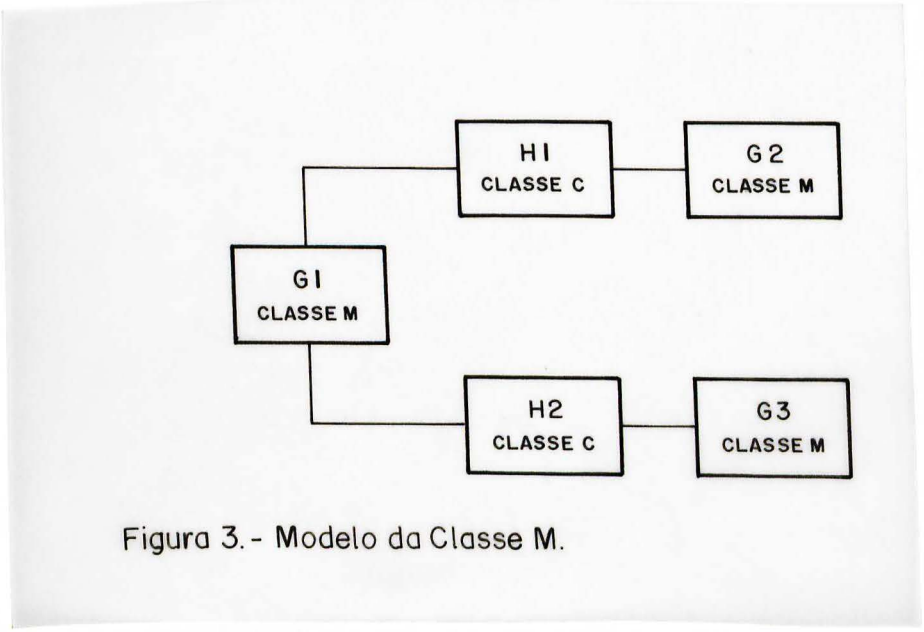


Figura 3.- Modelo da Classe M.

O problema da seqüência de carga reside dentro de um grupo G. Dois algoritmos são possíveis para a determinação desta seqüência. Na estratégia "master-slave", pode-se ter qualquer tipo de estrutura de conexão entre os elementos. Para a estratégia de ordem reversa de alimentação, é necessário modelar cada bloco como na fig. 4, dando a ele uma estrutura em forma de árvore, o que é possível se laços internos ao bloco forem quebrados introduzindo-se elementos "frios" da classe C, cuja única função é transportar um valor da entrada para a saída.

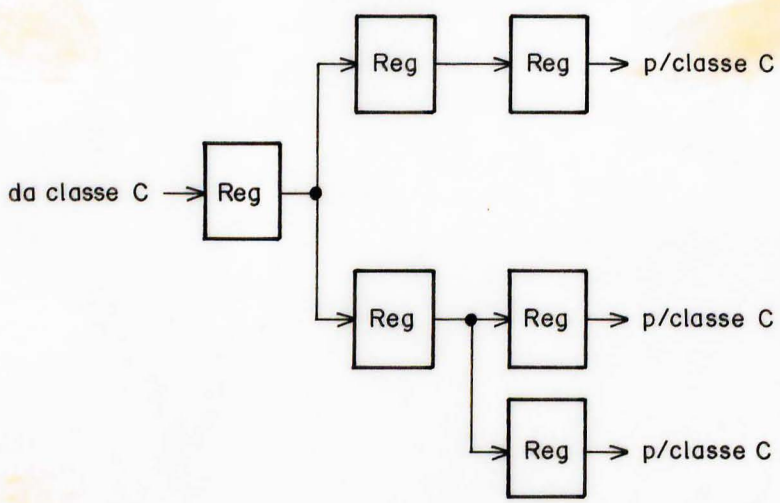


Figura 4.- Modelo Interno de um Bloco da Classe M.

Na estratégia "master-slave" (ver Fig. 5), os registradores sensíveis à borda são tratados como se fossem do tipo master-slave. O passo de simulação é dividido em dois instantes: no primeiro instante os valores das entradas de dados para os registradores são tomados das saídas dos registradores que os alimentam e armazenados numa variável temporária; no segundo instante estes valores são copiados das variáveis temporárias para as saídas dos registradores. Este procedimento garante que qualquer ordem de carga pode ser adotada para os elementos da classe M, obtendo-se sempre o mesmo resultado. Note-se que se os registradores já forem do tipo master-slave, então este procedimento terá sido obrigatoriamente adotado, e não se precisa ter preocupação em relação à ordem de carga.

```
("passo 1 da simulação")
para cada  registrador habilitado
   copie valor da entrada de dados para a variável intermediária

("passo 2 da simulação")
begin
para cada registrador habilitado
   begin
   copie valor da variável intermediária para a saída;
   se saída foi alterada
   então  para cada elemento conectado à saída
             coloque nome do elemento na lista de chamada
   end;

para cada elemento na lista de chamada
   propague alteração pelo bloco combinacional, até que ele alcance  um
   estado de repouso
end
```

Figura 5 - Simulação da classe M: estratégia "master-slave"

O procedimento anterior é custoso, pois exige o desdobramento do passo em dois sub-passos, a existência de uma variável adicional para cada registrador e duas atribuições a variáveis para cada registrador. Um resultado correto para os valores dos registradores também é obtido se eles forem chamados uma única vez, desde que na "ordem reserva de alimentação", ou seja, um dado registrador deve ser sempre chamado antes daquele que o alimenta (fornece o seu dado de entrada). Pensando-se na estrutura em forma de árvore, isto significa chamar, para cada nodo, em primeiro lugar os elementos correspondentes aos ramos e depois o correspondente ao nodo. Esta ordem é sempre a mesma para um dado sistema a ser simulado, de modo que ela

pode ser estabelecida previamente à simulação. O algoritmo de simulação correspondente é apresentado na Fig. 6.

```
para cada bloco G_i de registradores
begin
   carregue registradores do último nível; ("nível das folhas da
                                            árvore")
   carregue registradores do penúltimo nível; ("nível dos pais dos
                                               nodos-folha")
      .
      .
      .
   carregue registradores do segundo nível;  ("nível dos filhos do
                                              nodo-raíz")
   carregue registrador do primeiro nível  ("nível da raíz da árvore")
end
```

Figura 6 - Simulação da classe M: estratégia segundo a ordem reversa de alimentação

3.2 Determinação dos elementos a serem carregados

É comum em descrições RT associar-se uma condição de habilitação a cada registrador. Esta condição é um sinal que vem dos elementos da classe C. Assim, dos resgistradores que recebem a fase atual do clock, alguns podem estar desabilitados e não devem ser carregados no passo corrente. Duas estratégias existem para a determinação dos registradores habilitados.

Na estratégia de "varredura completa" (ver Fig.7) todos os registradores que recebem a fase atual do clock são pesquisados. A condição de cada um é interrogada, e caso o registrador esteja habilitado ocorre a carga. Ca-

so esta estratégia esteja sendo usada em conjunto com o método de simulação por recuo da classe C, então a condição será avaliada apenas quando da sua interrogação. Caso ela esteja sendo usada em conjunto com o método de simulação por avanço, a condição já terá sido avaliada previamente no passo anterior e necessita agora apenas ser interrogada.

```
para cada registrador
begin
   avalie condição de habilitação; ("só se for usado o método de
      simulação por recuo para a classe C")
   se condição de habilitação = verdadeira
   então coloque registrador na lista dos registradores a serem
         carregados
end
```

Figura 7 - Determinação dos registradores a serem carregados: estratégia de varredura completa

Na estratégia de "avaliação seletiva" (ver Fig. 8), os registradores com condição habilitada são conhecidos previamente. Como resultado da avaliação da classe C no passo anterior, os sinais de habilitação foram calculados, e os registradores que tiveram suas condições habilitadas foram colocados numa lista de chamada, que é percorrida então no passo seguinte. Claramente, apenas os registradores contidos nesta lista precisam ser carregados. Esta estratégia obviamente só é compatível com um algoritmo de simulação por avanço da classe C, pois com um algoritmo de simulação por recuo os sinais de habilitação não são conhecidos previamente ao passo no qual os registradores são carregados, e assim todos os registradores precisam ser interrogados. Nas Figs. 7 e 8, imagina-se que está sendo construída uma lista de registradores a serem carregados por um

dos algoritmos mostrados nas figs. 5 e 6. Estas descrições estão supondo clock de uma única fase, e deveriam obviamente ser adaptadas para clock de várias fases.

```
("simulação da classe C no tempo anterior")
   .
   .
   .
para cada elemento conectado a um sinal que se alterou
   se (elemento é registrador) e (sinal é entrada de habilitação) e
      (registrador está habilitado)
   então coloque elemento na lista de registradores habilitados
   .
   .
   .
("simulação da classe M no tempo atual")
para cada registrador na lista de habilitados
   coloque registrador na lista dos registradores a serem carregados
```

Figura 8 - Determinação dos registradores a serem carregados: estratégia de avaliação seletiva

Do ponto de vista da classe M, esta segunda estratégia é claramente mais eficiente. No entanto, como se verá na seção seguinte, o método de simulação por recuo pode ser mais eficiente para a classe C. Uma maior eficiência global de uma das duas possíveis combinações de métodos para as classes M e C dependerá muito do sistema a ser simulado, especialmente da quantidade de elementos em cada classe.

4. SIMULAÇÃO DA CLASSE C

4.1 Simulação por avanço

4.1.1 Chamada fixa (Fig. 9)

Descrição do método: todos os elementos da classe C são avaliados a cada passo. Os elementos estão organizados numa lista de chamada, de tal modo que uma única varredura desta lista permite a atualização de todos os sinais gerados pelos elementos. Isto é possível sob duas condições: a) que não existam laços de realimentação dentro da classe, o que deve ser verdade, pois caso contrário se teria um circuito seqüencial assíncrono, contrariando as premissas deste trabalho; b) que a lista esteja organizada segundo a ordem direta de alimentação dos elementos, ou seja, se um elemento i é alimentado por um elemento j, então j precede i na lista.

```
begin
avalie elemento 1;
avalie elemento 2;
  .
  .
  .
avalie elemento n
("cada elemento i é alimentado apenas por elementos j < i")
end
```

Figura 9 - Simulação da classe C por avanço: chamada fixa

Avaliação do método: Este método é tão mais eficiente quanto mais simples forem as rotinas de avaliação dos elementos da classe C. A lista de chamada é organiza

da previamente à simulação, de modo que o algoritmo de simulação é extremamente elementar, devendo apenas chamar as rotinas que correspondem aos elementos na ordem em que estes aparecem na lista. Se as rotinas são muito simples, é preferível avaliar todos os elementos do que adotar algum algoritmo que determine o sub-conjunto de elementos potencialmente ativos neste passo, pois a execução deste algoritmo certamente representaria uma sobrecarga não compensável pela não avaliação de parte dos elementos.

4.1.2 "Selective Trace" com seqüência dinâmica de chamada (Fig. 10)

Descrição do método: A cada passo são avaliados apenas os elementos potencialmente ativos, ou seja, aqueles cujas entradas variaram e portanto podem experimentar uma variação em suas saídas. Para isto é necessário que se tenha a informação sobre as interconexões dos elementos. Tendo a saída de um elemento i variado, é necessário identificar os elementos j,k,... cujas entradas estão conectadas à saída de i. Estes elementos serão a seguir avaliados. Não havendo realimentações, este processo certamente estabilizará, atingindo-se elementos cujas saídas não variam como resultado de variações em suas entradas, ou sinais que servem de entradas de dados ou habilitação para registradores, que não são avaliados neste passo em conseqüência destas variações. A lista de chamada dos elementos é montada dinamicamente durante o passo, de acordo com as variações e as interconexões.

Avaliação do método: o processamento da informação de interconexão dos elementos representa obviamente um custo em tempo. O método só se justifica se os elementos ativos representarem uma pequena parcela do total de elementos, o que depende fortemente do tipo de sistema a ser simulado, e se as rotinas de avaliação dos ele

mentos forem complexas.

```
("lista de avaliação inicializada com elementos cujas entradas estão
 conectadas a saídas de registradores que foram alteradas")
para cada elemento na lista de avaliação
   begin
   avalie função do elemento;
   se novo valor da saída é diferente do valor anterior
   então para cada elemento conectado a esta saída
         se elemento não é registrador
         então coloque elemento na lista de avaliação
         senão ...
end
```

Figura 10 - Simulação da classe C por avanço: "Selective Trace" com seqüência dinâmica de chamada

4.1.3 "Selective Trace" com seqüência estática de chamada (Fig. 11)

Descrição do método: ele é uma combinação do primeiro método (geração de uma lista fixa de chamada dos elementos de acordo com a ordem de alimentação) com o princípio do "Selective Trace". A cada elemento da lista é agora associado um flag, que é ligado quando uma entrada para o elemento varia, o que pode ocorrer durante o próprio processamento da classe C, caso o sinal que variou seja saída de um elemento da classe C, ou durante o anterior processamento da classe M, caso o elemento receba diretamente um sinal de saída de um registrador. Obviamente é necessária a informação de interconexão entre os elementos, assim como no segundo método. Um flag é desligado imediata-

mente após a execução da rotina correspondente ao elemento.

```
("cada elemento i na lista de chamada é alimentado apenas por
 elementos j < i")
("flag(i) = verdadeiro no início se entrada de i foi alterada por
 ação de um registrador")
begin
i = 0;
faça
    begin
    se  flag(i) = verdadeiro
    então begin
          avalie função do elemento;
          se novo valor da saída é diferente do valor anterior
          então para cada elemento k conectado a esta saída
                  se elemento não é registrador
                  então flag(k) := verdadeiro
                  senão ... ;
          flag(i) := 0
          end ;
    i := i+1
    end;
até que i=n
end
```

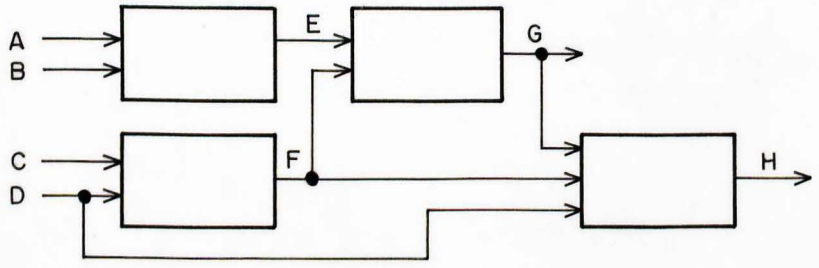
Figura 11 - Simulação da classe C por avanço: "Selective Trace" com seqüência estática de chamada

Avaliação do método: Em relação ao método de avaliação fixa, este terceiro método apresenta a vantagem de só serem chamados os elementos pontencialmente ativos a cada passo, tendo a desvantagem da sobrecarga devido ao pro

cessamento das informações de interconexão. Ele é seguramente mais rápido do que o método de "Selective Trace" com seqüência dinâmica de chamada, pois a única diferença entre ambos está na geração da lista de chamada. Naquele segundo método ela é gerada dinamicamente de acordo com as informações de interconexão, enquanto neste terceiro método ela existe permanentemente, bastando-se ligar e desligar os flags dos elementos ativos.

4.1.4 Instanciação

Para cada um dos três métodos é possível realizar-se simulação instanciada ou não-instanciada. Simulação instanciada significa que as informações de interconexão são armazenadas em tabelas, como na Fig. 12. Com isto utiliza-se rotinas genéricas de avaliação para os elementos de mesmo tipo (p.ex., uma única rotina para um multiplexador genérico), que não dependem das interconexões específicas do sistema sendo simulado. Esta rotina é então instanciada a cada chamada, ou seja, os nomes das entradas e saídas são passados a ela como parâmetros. Um supervisor da simulação se encarrega de, a partir do nome do elemento a ser processado, encontrar nas tabelas o seu tipo (determinando assim a rotina de avaliação) e os nomes de suas entradas e saídas, realizando a instanciação da rotina. A necessidade deste supervisor representa uma sobrecarga em tempo de simulação que é compensada quando de uma alteração no sistema sendo simulado, pois apenas as tabelas precisam ser refeitas, permanecendo válidas as rotinas e o supervisor. Esta vantagem ocorre obviamente em tempo de compilação, o que é importante durante um projeto num ambiente interativo de simulação, onde alterações freqüentes são efetuadas na descrição do sistema.



a) exemplo de sistema RT.

NOME DO ELEMENTO	TIPO DO ELEMENTO	VALOR DA SAÍDA	LISTA DE Fan-In	LISTA DE Fan-Out
⋮				
D	-	-	-	F, H
E	Tipo 1	Ve	A, B	G
F	Tipo 2	Vf	C, D	G, H
G	Tipo 3	Vg	E, F	H
H	Tipo 4	Vh	D,F,G	-

b) estrutura de dados para o exemplo de (a).

Figura 12.- Estrutura de Dados para Simulação Instanciada.

Na simulação não-instanciada, existe uma rotina para cada elemento no sistema sob modelamento. Esta rotina conhece os nomes dos sinais de entrada para o elemento e os nomes dos elementos ao quais está conectada a saída do elemento (estes últimos não são necessários para o método de chamada fixa). A lista de chamada contém os nomes das rotinas (ou as próprias rotinas, no caso de chamada fixa), e não os nomes dos elementos, como na simulação instanciada. É necessário um supervisor de simulação bem mais simples, cuja única tarefa consiste em chamar as rotinas constantes na lista. As desvantagens da simulação não-instanciada em relação à instanciada são duas: a) exige muito mais memória; b) quando uma alteração é feita na descrição do sistema, é exigido maior esforço de recompilação. Em compensação, simulação não-instanciada é muito mais rápida, pela grande simplificação do supervisor de simulação que co

ordena as interações entre os elementos. Esta estratégia obviamente só é válida para rotinas bastante simples, pois em caso contrário o gasto de memória pode tornar-se proibitivo.

4.1.5 A duplicação na avaliação dos elementos

No método de "Selective Trace" com seqüencia dinâmica de chamada, pode ocorrer que um mesmo elemento seja avaliado duas ou mais vezes num mesmo passo, o que evidentemente redunda numa perda de eficiência para a simulação. Nos métodos de chamada fixa e "Selective Trace" com seqüência estática, a lista de chamada é construída de modo que certamente uma única avaliação de cada elemento pode ocorrer em cada passo.

No método de "Selective Trace" com seqüência dinâmica, um elemento alimentado por dois sinais que foram alterados no passo atual será colocado duas vezes na lista de chamada, e portanto será chamado duas vezes, e não só ele, mas todos os elementos por ele alimentados, caso sua saída seja alterada em conseqüência das alterações nas entradas. Obviamente esta duplicação pode ser evitada se: a) a lista de chamada for administrada como uma fila, e não como uma pilha; b) a cada elemento for associado um flag que indique se ele já está na lista. Dependendo do tipo de topologia do sistema sendo simulado e da complexidade das rotinas de avaliação dos elementos, este tratamento para evitar duplicações pode representar uma sobrecarga não compensável, de modo que é preferível permitir as duplicações.

A administração da lista como uma pilha (o que é mais simples caso não se deseje evitar as duplicações) pode resultar em pulsos de duração zero na saída dos elementos.

A fig. 13 exemplifica isto. O sinal C tem a princípio o valor 1, e após as duas transições de entrada, ambas ocorridas neste passo, mantém o valor 1. A pilha contém a princípio A e B. A conseqüência da transição A:1→0 é a avaliação de C (com B ainda igual a 0), o que resulta em C=0. Esta alteração irá provocar a chamada dos elementos "fan-out" de C (i.e., os elementos ligados à saída de C) de forma desnecessária, pois a posterior transição B:0→1 traz C de volta para 1, causa novamente a chamada de seus "fan-outs" e cancela os possíveis efeitos de C:1→0 dentro da classe C, já que por definição não existem dentro dela laços de realimentação. Do ponto de vista da classe C, portanto, esta duplicação apenas causa perda de eficiência na simulação e a passagem do sistema por um falso estado intermediário. Para a classe M, no entanto, os efeitos podem ser duradouros, caso a alteração em C cause a habilitação de um registrador e esteja se usando avaliação seletiva no processamento da classe M. Neste caso, é necessário um tratamento especial que evite este erro.

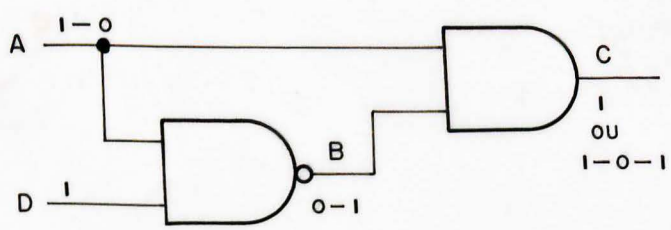


Figura 13. - Exemplo de duplicação na avaliação dos elementos da classe C.

4.2 Simulação por recuo

Descrição do método: Nesta estratégia, são avaliados apenas os elementos da classe C cujos sinais de saída são necessários à determinação do valor de entrada para um

registrador atualmente considerado. Do ponto de vista deste registrador, a classe C pode ser vista como na Fig. 14, i.e., como uma árvore cujos nodos são os elementos da classe C e cujas folhas são elementos da classe M, entradas primitivas (E) e constantes (K). Na realidade, alguns destes nodos podem ser um mesmo, de modo que a estrutura é um grafo, e não uma árvore. As implicações disto serão consideradas adiante.

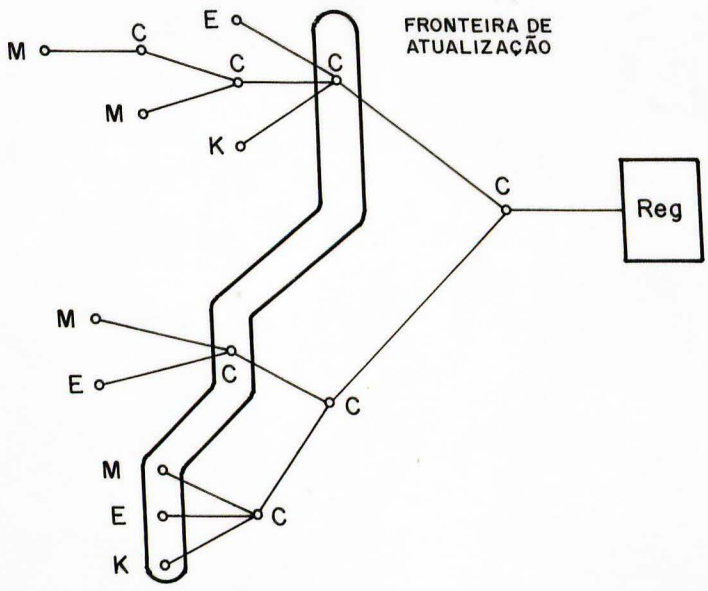


Figura 14.- Classe C vista pelo algoritimo de simulação por recuo.

Como em todos os métodos aqui apresentados, elementos da classe M estão permanentemente atualizados. Entradas primárias e constantes estão por definição também sempre atualizadas. O método consiste então em percorrer para trás a árvore a partir da entrada de dados de um registrador, à procura das folhas, ou seja, dos nodos com valores certamente atuais neste passo. Obviamente a avaliação dos elementos será feita voltando-se desde as folhas até a entrada do registrador, como se exige para hardware.

Sendo

s o nome de um nodo,

srci,..., srcn os nomes de seus ramos,

valor (s) o valor do sinal de saída de s, armazenado num array,

op (valor (src1),..., valor (srcn)) a rotina que executa a função do elemento correspondente ao nodo,

pode-se escrever a função recursiva mostrada na Fig. 15 para a avaliação do sinal de entrada para um registrador.

```
função inteira novo-valor (s);
begin
   se (s é registrador) ou (s é contante) ou (s é entrada primária)
   então novo-valor := valor (s)
   senão begin
         valor (s) := op (novo-valor (src1),..., novo-valor (srcn));
         novo-valor := valor (s)
         end
end
```

Figura 15 - Simulação da classe C por recuo

Avaliação do método: A vantagem do método está baseada em duas suposições: a) o usuário não tem o objetivo de conhecer a cada passo os valores de todos os elementos do sistema, ou seja, apenas as seqüências de estados por que passam os registradores interessam como resultado da simulação; b) a lógica combinacional de entrada para um registrador passa por valores intermediários que não chegam a ser utilizados para a carga deste registrador, sendo portanto desnecessária a sua determinação. A Fig. 16 ilustra este ponto. O conteúdo de R1 é alterado na fase CP1, causando alterações na lógica combinacional. Estes valores

não precisam no entanto ser calculados, o que ocorreria com o método de simulação por avanço pois em CP2 o conteúdo de R2 é alterado, causando novas alterações na lógica combinacional. Estes novos valores são então utilizados na carga de R3 em CP3. É de se esperar portanto uma maior eficiência do método por recuo na simulação de sistemas com clock de várias fases. A desvantagem básica do método está em não utilizar o princípio do "Selective Trace", pois não há como se saber se as folhas da árvore avaliada pela rotina recursiva foram alteradas desde a última avaliação ou não, significando isto que todos os nodos da árvore serão necessariamente avaliados, tenham ou não sido alterados. Uma maior eficiência de um método (por recuo ou por avanço) sobre o outro parece depender fortemente da topologia do sistema em estudo. Apenas uma análise concreta baseada em resultados obtidos de exemplos reais e significativos deve poder fornecer resultados conclusivos, o que foge ao âmbito deste trabalho.

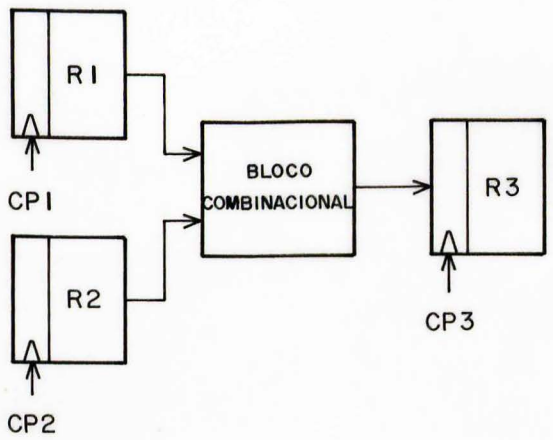


Figura 16.

Instanciação: O caráter recursivo da avaliação exige a existência de uma rotina genérica de avaliação op (...) para cada tipo de elemento da classe C, que possa receber os valores das entradas valor (srci),..., valor (srcn) como parâmetros. É portanto impossível evitar-se esta ins-

tanciação.

Duplicação na avaliação: O fato da estrutura ser na realidade um grafo e não uma árvore significa que um mesmo elemento aparece em mais de um caminho durante a avaliação do sinal de entrada para um registrador, ou então aparece na lógica de entrada de mais de um registrador, o que levaria a uma duplicação na avaliação deste elemento. Esta duplicação é no entanto de natureza bem diferente daquela existente no método "Selective Trace" com seqüência dinâmica de chamada. Lá a duplicação poderia levar a pulsos de duração zero por não se ter todos os sinais de entrada para um elemento com valores já atuais quando de sua avaliação. No caso do método de simulação por recuo no entanto, o caráter recursivo da avaliação garante que todas as entradas para um elemento estão atualizadas quando de sua avaliação. Isto significa que a duplicação resulta apenas em perda de eficiência. Ela pode ser facilmente evitada associando-se uma marca de tempo a cada elemento da classe C, que contenha sempre o tempo no qual este elemento foi pela última vez avaliado. Uma simples comparação desta marca com o tempo atual é suficiente para evitar a continuação da busca recursiva para além do nodo atualizando. Determina-se assim uma fronteira de atualização como na Fig. 14. Pode-se dizer então que o método consiste na procura desta fronteira de atualização, que no caso da avaliação do primeiro sinal dentro do passo atual certamente coincidirá com as folhas da árvore.

5. CONCLUSÕES

Este trabalho apresentou diferentes algoritmos de simulação de sistemas digitais síncronos descritos no nível RT. Considerações foram feitas a respeito da eficiência da simulação obtida com cada algoritmo, mostrando-se que, embora o nível RT seja uma abstração do nível de portas lógicas, algoritmos de simulação para este último /WAG 84/ não são eficientes para o nível RT.

O trabalho não considerou outras conseqüencias advindas dos algoritmos de simulação apresentados, tais como aqueles resultantes do seu uso num ambiente altamente interativo de projeto. Num tal ambiente, é importante que pequenas modificações possam ser feitas na descrição do sistema digital sendo projetado, sem que isto implique numa completa recompilação do simulador ou das estruturas de dados que ele utiliza. Esta recompilação se faz necessária p.ex. se forem usadas a estratégia de chamada fixa ou de "selective trace" com seqüência estática de chamada para a simulação da classe C. Sob este aspecto, embora a estratégia de "selective trace" com seqüência dinâmica de chamada possa ser mais ineficiente do ponto de vista da simulação, ela permite facilmente alterações na descrição do sistema, através da modificação de poucos registros nas tabelas descritivas da estrutura.

Considerações semelhantes devem ser feitas quando da comparação entre a estratégia "master-slave" e a estratégia da ordem reversa de alimentação para a determinação da seqüência da carga dos elementos M, pois esta última exige recompilação do programa de simulação quando de uma alteração na estrutura do sistema.

Simuladores de sistemas assíncronos podem certamente aproveitar muitas das técnicas aqui apresentadas, especialmente no tratamento da classe C. No entanto, pos-

sivelmente eles exigirão: a) a introdução de mecanismos que permitam especificar a duração de atividades, como forma de se determinar a velocidade relativa de processos, e b) a introdução de mecanismos de sincronização entre processos. Tais exigências darão ao simulador uma estratégia global de simulação possivelmente orientada a processos e eventos.

BIBLIOGRAFIA

/BAR 75/ BARBACCI, M.R. "A Comparison of Register Transfer Languages for Describing Computers and Digital Systems", IEEE Transactions on Computers, Vol C-24, Feb. 1975, pp. 137-150

/BAR 81/ BARBACCI, M.R. "Instruction Set Processor Specifications (ISPS); The Notation and Its Applications", IEEE Transactions on Computers, Vol. C-30, Jan. 1981, pp 24-40

/CHU 70/ CHU, Y. Introduction to Computer Organization, Prentice-Hall, Englewood-Cliffs, 1970

/DUL 68/ DULEY, J.R. e D.L. DIETMEYER. "A Digital System Design Language (DDL)", IEEE Transactions on Computers, Vol. C-17, Sep. 1968, pp 850-861

/HAY 78/ HAYES, J.P. Computer Architecture and Organization. McGraw-Hill, Inc. 1978

/WAG 84/ WAGNER, F.R. Gate-level Simulation: A tutorial. CPGCC-UFRGS. Out. 1984 Relatório Técnico nº 012

/WEN 79/ WENDT, S. Blockorientiertes interaktives Simulationssystem BORIS. Universität Kaiserslautern, Fachbereich Elektrotechnik, Informationsschrift. Dez. 1979

RELATÓRIOS TÉCNICOS MAIS RECENTES

RT-003: PRONDZYNSKI, P. R., FREITAS, C. M. D. S.,
TODESCO, A. R. W.
Sistema de Exibicao Grafica: utilizacao dos dispositivos.
Porto Alegre, CPGCC/UFRGS, nov/83.

RT-004: LASCHUK, A. PRONDZYNSKI, P. R., CALAZANS, N. L. V.,
ALBECHE, K. S.
Sistema de Exibicao Grafica: extensao do barramento para o HP2100.
Porto Alegre, CPGCC/UFRGS, out/83.

RT-005: PRONDZYNSKI, P. r.
Arquiteturas para videos graficos com varredura fixa.
Porto alegre, CPGCC/UFRGS, nov/83.

RT-006: WAGNER, F. R.
Modelamento de processos digitais com redes de instancias
Porto Alegre, CPGCC/UFRGS, mar/84.

RT-007: COSTA, A. C. R.
Caracterizacao dos conhecimentos e da arquitetura de um sistema especialista em projeto logico de circuitos digitais.
Porto Alegre, CPGCC/UFRGS, jun/84.

RT-008: FREITAS, C. M. D. S.
Programacao grafica interativa com o PGE/UFRGS.
Porto Alegre, CPGCC/UFRGS, jun/84.

RT-009: FREITAS, C. M. D. S.
Descricao do pacote grafico PGE/UFRGS.
Porto Alegre, CPGCC/UFRGS, jul/84.

RT-010: SAYAO, M. & TOSCANI, S. S.
Sistema multiprogramavel HP2100S -
manual de referencia.
Porto Alegre, CPGCC/UFRGS, out/84.

RT-011: SAYAO, M. & TOSCANI, S. S.
Sistema multiprogramavel HP2100S -
manual de usuario.
Porto Alegre, CPGCC/UFRGS, out/84.

RT-012: WAGNER, F. R.
Basic techniques for gate level simulation - a tutorial.
Porto Alegre, CPGCC/UFRGS, out/84.

RT-013: WAGNER, F. R.
Hazard detection in logic simulation.
Porto Alegre, CPGCC/UFRGS, nov/84.

RT-014: COSTA, A. C. R.
Clause machines.
Porto Alegre, CPGCC/UFRGS, jan/85.

RT-015: COSTA, A. C. R.
Especificacao das tarefas do sistema especialista em projeto logico de circuitos digitais.
Porto Alegre, CPGCC/UFRGS, jan/85.

RT-016: TOSCANI, L. V. & outros.
Laboratorio de Matematica Computacional - manual do usuario.
Porto Alegre, CPGCC/UFRGS, dez/84.

RT-017: TOSCANI, L. V. & outros.
Laboratorio de Matematica Computacional - manual de programas.
Porto Alegre, CPGCC/UFRGS, dez/84.

RT-018: COSTA, A. C. R.
Introducao aos sistemas especialistas e descricao informal do sistema muPROSPECTOR.
Porto Alegre, CPGCC/UFRGS, mai/85.

RT-019: FRIEDRICH, L. F. & COSTA, A. C. R.
Descricao da implementacao do montador MC68000 escrito em Pascal Sequencial.
Porto Alegre, CPGCC/UFRGS, jun/85.

RT-020: COSTA, A. C. R.
Processando linguagens naturais em PROLOG - Parte 1: Formalismo gramatical basico.
Porto Alegre, CPGCC/UFRGS, jul/85.