Aplicação da Linguagem VHDL na Especificação Funcional de Sistemas em Ambiente de Síntese

Paulo Flores

Inesc - Apartado 10105, Lisboa PORTUGAL

Contents

1 Introdução

Este realatório descreve de uma forma breve o uso de linguagens de descrição de *hardware* num ambiente de CAD. Especial atenção será dada à nova linguagem padrão designada "VHSIC[1] Hardware Description Language" (VHDL).

As linguagens de descrição de *hardware* permitem descrever um circuito num nível mais elevado de abstração, conseguindo-se assim descrever circuitos mais complexos com maior facilidade e verificar a sua funcionalidade através de simulação. A tradução do circuito para uma descrição ao nível de portas lógicas pode ser feita de uma forma manual ou automaticamente, recorrendo por exemplo a ferramentas de síntese.

As linguagens de descrição de *hardware* superam algumas das limitações das linguagens algorítmiticas, onde a referência ao tempo não é importante. Enquanto que nas primeiras o comportamento temporal do circuito está explicito na linguagem, nas segundas este comportamento tem de alguma forma ser representado no algoritmo que desreve o circuito.

Devido à representação de um circuito ser feita de uma forma tecnologicamente independente, a escolha do fabricante pode ser adiada para uma fase posterior do projecto. Assim a reutilização de projectos anteriores torna-se possivel de uma forma mais simples com o uso de bibliotecas que podem ser partilhadas pelos diversos projectistas.

As linguagens de descrição de *hardware* têm ainda a vantagem de representar também um meio importante para a troca de informação de *hardware* entre organizações, vantagem essa que pode ter um significado mais real se a linguagem escolhida estiver largamente divulgada na comunidade de micro-electrónica.

Nas próximas secções apresentaremos sumariamente a linguagem VHDL e teceremos alguns comentários sobre o seu uso num ambiente de CAD.

2 A Linguagem VHDL

O processo de normalização do VHDL começou em 1986 com adopção do VHDL versão 7.2. Actualmente a definição da linguagem está estável e tornou-se numa norma do DoD[2]/IEEE para todos os projectos de electrónica. Hoje em dia esta linguagem "começa a ser" suportada praticamente por todos os sistemas comerciais de CAD.

[1] VHSIC - Very High Speed Integrated Circuits

[2] Departamento de Defesa dos Estados Unidos da América

2.1 Entidades de Projecto

A entidade de projecto é uma abstração de *hardware* em VHDL que pode representar um sistema completo, uma placa, um integrado, uma célula ou mesmo uma simples porta lógica. Cada entidade de projecto é formada por duas partes: a declaração da entidade e a sua arquitectura. Na primeira define-se a interface do sistema para com o exterior. Na segunda é descrita a arquitectura interna do sistema.

Cada declaração de entidade pode ter associada um número indeterminado de arquitecturas. A mesma funcionalidade em cada uma das arquitecturas pode ser obtida recorrendo aos seguintes estilos de descrição:

Descrição Estrutural - o circuito é apresentado como um conjunto de componentes interligados.

Descrição de Fluxo de Dados - representa o comportamento do circuito mas implica haver alguma estrutura. É identica a uma descrição a nível RTL[3]

Descrição Comportamental - descreve o comportamento do circuito sem qualquer informação quanto à sua estrutura. Este tipo de descrição é feita usando uma sequências de instruções idênticas às da linguagens de programação algorítmiticas.

```
entity test_circuit is
    port(I: in integer range 0 to 7;        -- Entrada de dados
         D_IN: in bit;                      -- Carregamento de dados
         X: in bit_vector(1 to 0);          -- Codigo de operacao
         O: out integer range 0 to 7;       -- Saida de dados
         OVF,                               -- Condicao de 'Overflow'
         LT, EQ, GT: out boolean;           -- Resultado da comparacao
         RESET,                             -- 'Reset' do sistema
         CLK: in BIT                        -- Relogio
    );
end test_circuit;
```

Figure 1: A descrição do interface

Uma descrição mista onde seja usada uma combinação dos três estilos acima referidos também é possivel.

[3]Register Transfer Level

```
architecture behavioral of test_circuit is
begin
    process
        variable reg, acc: integer range 0 to 7;
        variable temp: integer range 0 to 15;
    begin
        if CLK = '1' and not CLK'stable then
            if D_IN = '1' then
                reg := I;
            else
                case X is
                    when "00" =>                -- Operacao nula
                        null;
                    when "01" =>                -- Carrega reg no acc
                        acc := reg;
                        ovf <= FALSE;
                    when "10" =>                -- Compara reg com acc
                        LT <= (reg < acc);
                        GT <= (reg > acc);
                        EQ <= (reg = acc);
                    when "11" =>                -- Soma reg e acc
                        temp := reg + acc;
                        acc := temp rem 8;
                        ovf <= (temp > 7);
                end case;
            end if;
        end if;
        if RESET = '1' and RESET'stable then
            reg := 0;
            acc := 0;
        end if;
        O <= acc;
        wait on CLK, RESET;
    end process;
end behavioral;
```

Figure 2: A descrição comportamental do circuito

Para ilustrar o conceito de entidade de projecto, usaremos como exemplo uma Unidade Lógica e Aritmética (ALU) de 3 bits (para uma descrição mais detalhada das operações desta ALU consultar [1]). Na figura 1 define-se apenas a interface, sendo a própria unidade considerada como uma "caixa preta". Na figura 2, que representa a arquitectura do circuito, está o código VHDL que implementa as funções que esta ALU desempenha.

2.2 Objectos e Tipos de Dados

Objectos em VHDL são entidades que contêm um valor de um dado tipo. Existem três classes de objectos: constantes, variáveis e sinais. Constantes têm um valor fixo que não pode ser alterado, enquanto que as variáveis têm um valor que pode ser alterado ao longo da execução do programa. Os sinais podem ser visto como a representação abstracta de um fio: têm um conjunto de valores passados, um valor presente e um conjunto de valores projectados para o futuro. Só estes ultimos valores podem ser alterados pelas instruções de atribuição.

O tipo de um objecto determina quais os valores que esse objecto pode conter. Em VHDL é permitido ao projectista declarar qualquer número de tipo de dados para caracterizar os seus objectos.

Os 4 tipos básicos escalares são: inteiros, virgula flutuante, físicos e enumerados. Tipos compostos como matrizes e agregados podem ser definidos à custa dos tipos básicos. Sub-tipos podem ser também definidos através da imposição de restrições a um outro tipo. Existem ainda os apontadores, idênticos àqueles que são usados nas linguagens vulgares de programação, e os ficheiros que permitem uma forma de comunicação do circuito com o meio exterior.

2.3 O Modelo Temporal

O modelo temporal em VHDL é bastante geral e complexo. Devido a ser uma linguagem de simulação discreta controlada por acontecimentos, assume-se que todos os sinais se propagam só numa direcção e que algum atraso está sempre envolvido.

A escala temporal pode ser analizada sobre duas perspectivas, uma que mede o tempo real de simulação e outra que representa o tempo de cálculo de um ciclo de simulação. Este último é chamado de "*delta delay*" e pode ser visto como um atraso infinitesimalmente pequeno mas diferente de zero que é incrementado cada vez que um ciclo de simulação termina, no mesmo instante de tempo real.

São os *drivers* que contêm os valores projectados para o futuro de uma onda. Por cada *driver* associado a um sinal, devido a uma instrução de atribuição, podem existir dois modelos de atraso que controlam a forma como os novos pares instante/valor vão ser colocados nesse *driver*:

Atraso Inercial - é o modelo de defeito em VHDL. Significa que para que um dado valor se repercuta na saída, ele tem que permanecer um determinado tempo mínimo na entrada.

Atraso de Transporte - este modelo é idêntico aos atrasos que sofre uma corrente electrica ao atravessar um fio, qualquer conjunto de impulsos na entrada são reflectidos na saída.

2.4 Instruções

As instruções especificam o modo de organização e de operação de um circuito. Uma vez que os dispositivos electrónicos funcionam na sua maior parte em paralelo, a VHDL é uma linguagem com grandes capacidades de concorrência. Assim o projectista tem dois niveis para especificar o seu sistema: o nível sequencial e o nivel concorrente.

2.4.1 Instruções Sequenciais

Todos as instruções sequenciais estão encapsuladas dentro de processos, subprogramas ou funções, que são usadas em contextos concorrentes. As instruções sequenciais especificam o algoritmo passo a passo e tal como acontece em qualquer outra linguagens de alto nível existem construções do tipo if, case, ciclos, procedimentos, atribuição a variáveis, etc que têm a sua própria sintaxe. Sendo a VHDL no entanto uma linguagem de descrição de *hardware* possuiu algumas construções próprias:

Atribuição Sequencial a Sinais - são onde os futuros valores para os sinais são "propostos". Como veremos em secções seguintes estes valores poderão não vir a ser os futuros valores que os sinais irão tomar.

Instrução de "Espera" - suspende a execução do algoritmo e espera por um conjunto de condições. Essas condições podem ser a sensibilidade à variação de sinais, uma condição lógica dos seus valores, o expirar de um determinado tempo ou uma mistura das três condições.

Instruções Sequenciais de "Defesa" - verificam se uma determinada condição é verificada, caso contrário reportam um erro.

As instruções sequenciais são usadas quando se pretende descrever um circuito indicando apenas o seu comportamento, ou seja sem dar qualquer informação quanto à sua estrutura.

2.4.2 Instruções Concorrentes

As instruções concorrentes são executadas assincronamente, isto é, nada pode ser assumido quanto à sua ordem de execução. São elas que permitem descrever um circuito sob a forma estrutural ou de fluxo de dados:

- Descrição Estrutural

  Instrução Bloco - permite que um conjunto de instruções concorrentes sejam agrupadas numa unidade lógica para descrever uma parte do circuito.

  Instruções de Instanciação de Componentes - faz a chamada a um outro componente existente na biblioteca e associa sinais ao seus portos de interface.

  Instrução de Geração - fornece um mecanismo iterativo e/ou condicional para a criação de parte do circuito.

- Descrição de Fluxo de Dados

  Instrução Concorrente de Atribuição a Sinais - cria um novo *driver* por cada atribuição, onde os novos valores propostos para o sinal são guardados. Por cada sinal existe um conjunto associado de 'drivers' que, se estiverem a tentar impôr valores diferentes num mesmo instante vão chamar uma função de resolução para decidir qual o valor que o sinal vai ter de facto.

  Instrução de Processo - permite de uma forma independente e sequencial descrever o comportamento de uma parte do circuito.

  Chamadas Concorrentes de Procedimentos - permitem que estes sejam executados de uma foma concorrente.

  Instruções Concorrentes de "Defesa" - têm o mesmo significado das instruções sequenciais de "defesa" mas usadas num meio concorrente.

3 Aplicações de VHDL

Apesar de a VHDL ter sido desenvolvida como uma linguagem de simulação, hoje em dia muitos sistemas de CAD aceitam-na como forma de descrever os seus circuitos. No entanto devido à sua complexidade algumas restrições são feitas e portanto cada ferramenta apenas aceita um subconjunto da linguagem.

3.1 Simuladores

Como seria de esperar, os simuladores foram as primeiras ferramentas a fazerem uso de VHDL, e apesar de actualmente existir no mercado um vasto conjunto de opções, são poucos ainda os que aceitam todas as capacidades da linguagem.

Basicamente existem duas aproximações na forma como os simuladores de VHDL são desenvolvidos. Num caso é produzido código intermédio para posteriormente ser interpretado, provavelmente por simuladores já existentes, resultando assim numa simulação mais lenta mas com um ciclo de eliminação de erros mais rápido. No outro caso é gerado código C, a patir de VHDL, que depois vai ser compilado, resultando assim numa simulação mais rápida. No futuro os simuladores compilarão directamente de VHDL para código máquina executável.

3.2 Ferramentas de Síntese

As ferramentas de síntese estão a começar a suportar VHDL, embora alguns problemas possam surgir no uso desta linguagem num ambiente de síntese [2]. Dificuldades de como modelar dispositivos a baixo nível, compreender se a informação temporal faz parte da especificação ou do modelo para a simulação, definir qual a equivalência entre os níveis lógicos e os níveis eléctricos são apenas alguns dos problemas que poderão surgir.

No entanto, é possível sintetizar circuitos a partir de uma descrição de VHDL, mas algumas restrições são impostas. Só um sub-comjunto da linguagem é usado, o que significa que algumas construções próprias para simulação são ignoradas ou mesmo não suportadas. Há mesmo ferramentas que impôem restrições mais severas, necessitando de informação estrutural sobre o circuito, exigindo por isso uma descrição a nível RTL. Além disso, os resultados da síntese dependem em grande parte da forma como a descrição é feita.

3.3 Outras Ferramentas de CAD

Ferramentas de teste e de captação de esquemáticos já começaram a aceitar VHDL, mas, mais uma vez apenas sub-conjuntos da linguagem são suportados. Assim para cada ferramenta existe um "VHDL prórprio".

Analizadores, que apenas lêem uma descrição VHDL, verificam a sua sintaxe e semântica, e colocam essa descrição num biblioteca de projecto, tornam mais fácil a interface, entre velhas e novas ferramentas, para com o VHDL. Com o uso de atributos, potencialidade que a linguagem oferece, podem-se associar valores a quaisquer dados do projecto facilitando assim também a interface com as ferramentas.

4 Conclusões

No futuro os projectos em electrónica começarão a ser descritos num nível de abstração elevado, recorrendo por isso cada vez mais ao uso de linguagens de *hardware*.

Ao contrário de outras linguagens de descirção de *hardware*, que em geral são proprietárias de determinados fabricantes, a VHDL é público, o que garante a sua independencia face às políticas de cada companhia. Mais ainda, a própria definição da linguagem envolveu a participação dos utilizadores, o que garante que a VHDL virá a ser de facto uma norma usada na prática.

A adopção da VHDL como uma norma para a descrição de circuitos electrónicos por parte do DoD/IEEE trouxe enormes vantagens para a comunidade de CAD. Primeiro, do ponto de vista do projectista só será necessário aprender uma única linguagem para "todas as fases" do projecto (simulações, síntese, etc). Em segundo lugar, permite de uma forma normalizada, a partilha de informação entre diferentes grupos de projecto, usando diferentes níveis de abstração. Devido ainda à possibilidade do uso de diferentes estilos de descrição, não é necessário haver um compromisso prévio na escolha duma dada arquitectura aquando do projecto dum sistema. Mesmo os vendedores podem distribuir o comportamento dos seus dispositivos sem fornecer qualquer tipo de informação confidencial sobre a sua construção. Finalmente, é possível a documentação de sistemas digitais de uma forma tecnologicamente independente.

Decidir qual o melhor conjunto de construções de VHDL e o estilo de descrição a adoptar num ambiente de CAD, depende das ferramentas envolvidas e ainda é um assunto de investigação actual. No entanto, a inexistência de um subconjunto de VHDL comum a todas as ferramentas de CAD é a sua principal limitação.

References

[1] Ana Teresa Sousa e José Carlos Monteiro. O processo de síntese lógica: Breve descrição. Relatório jnict, INESC, 1991.

[2] Michel Crastes and Alain Fonkoua. Hdls and logic synthesis. Technical report, INPG/91/ECIP2/WP2/SYN.D, 1991.

[3] R. Lipsett, C. Schaefer, and C. Usserty. *VHDL: Hardware Description and Design*. Kluwer Academic publishers, 1989.

[4] IEEE Std 1076-1987. *IEEE Standard VHDL Language Reference Manual*. IEEE, 1989.

[5] CAD Languages Systems. Inc. *VHDL Training Seminar*, 1990.

[6] Moe Shahdad. An Overview of VHDL Language and Technology. In *Design Automation Conference*, pages 320–326, July 1986.

[7] T. Dillinger, K. McCarthy, T. Mosher, D. Neumann, and R. Schmidt. A Logic Synthesis System for VHDL Design Description. In *IEEE International Conference on Computer-Aided Design*, pages 66–69, 1989.

[8] Larry M. Augustin. Timing Models in VAL/VHDL. In *IEEE International Conference on Computer-Aided Design*, pages 122–125, 1989.

[9] Steve Carlson. *Introduction to HDL-Based Design Using VHDL*. Synopsys, Inc., 1990.