Ambiente de Síntese
para
Projecto de Circuitos Digitais

Paulo Flores

Índice

1 Introdução **2**

2 Fluxo de Projecto **2**

3 Ferramentas de Síntese **5**

3.1 O VHDL Suportado 5

3.1.1 Unidades de Projecto 6

3.1.2 Objectos, Tipos de Dados e Atributos 8

3.1.3 Expressões 9

3.1.4 Instruções sequenciais 11

3.1.5 Instruções Concorrentes 11

3.2 Estilo de descrição 11

4 Descrições portáveis **13**

5 Biblioteca de Modelos **14**

6 Conclusões **14**

A Biblioteca de Modelos - Manual de Referência **16**

1 Introdução

O projecto de um circuito integrado começa geralmente com uma especificação de alto nível do seu comportamento e um conjunto de restrições que devem ser satisfeitas (por exemplo, área máxima do circuito, tempos de atraso mínimos e/ou máximos de alguns sinais). O processo de síntese pretende obter a partir de uma descrição de alto nível do circuito a sua representação "óptima", mapeada em portas lógicas de uma dada tecnologia, e que satisfaça as restrições impostas.

A descrição do circuito através de uma linguagem de descrição de *hardware* é o primeiro passo de projecto num ambiente de síntese. A utilização de uma linguagem poderosa e complexa, como o VHDL (norma do IEEE), requer ao projectista um profundo conhecimento da linguagem para aproveitar todas as sua potencialidades. Adicionalmente, o projectista deve ter em consideração quais as restrições impostas pela ferramenta de síntese, de forma a garantir que a descrição é sintetizável e que o circuito resultante tem o desempenho desejado. Descrições não adequadas à síntese geralmente resultam em circuitos incorrectos ou não optimizados, independentemente da qualidade e capacidade da ferramenta de síntese utilizada.

Este relatório descreve o fluxo de projecto usado num ambiente de síntese. Explicam-se os passos necessários para obter uma representação do circuito no nível lógico e destaca-se a natureza iterativa/interactiva deste processo. A questão fundamental da identificação do subconjunto de construções de VHDL sintetizáveis é analisada com algum detalhe, focando-se as construções suportadas pela ferramenta de síntese da Synopsys, `Design Compiler` [Synthesis 92]. Algumas questões de portabilidade entre diferentes ferramentas são abordadas na secção 4. Antes das conclusões é apresentada uma biblioteca de modelos sintetizável. Esta foi desenvolvida tendo em conta o subconjunto de construções sintetizáveis e utilizando um estilo de descrição apropriado à síntese.

2 Fluxo de Projecto

O processo de síntese da lista de portas lógicas que implementam um circuito a partir da sua descrição numa linguagem de alto nível não se efectua num único passo. O fluxo de projecto habitualmente seguido na síntese de circuitos digitais a partir de uma descrição VHDL é composto de quatro passos (ver figura 1):

1. Obter uma descrição do circuito em VHDL. Esta descrição deve ser feita num nível de abstracção que permita libertar o projectista dos pormenores de implementação e ao mesmo tempo possa ser sintetizada pelas actuais ferramenta de síntese.

2. Validar a descrição obtida anteriormente através de um simulador de VHDL. Este passo do projecto garante que a funcionalidade desejada pelo projectista se encontra na descrição VHDL.

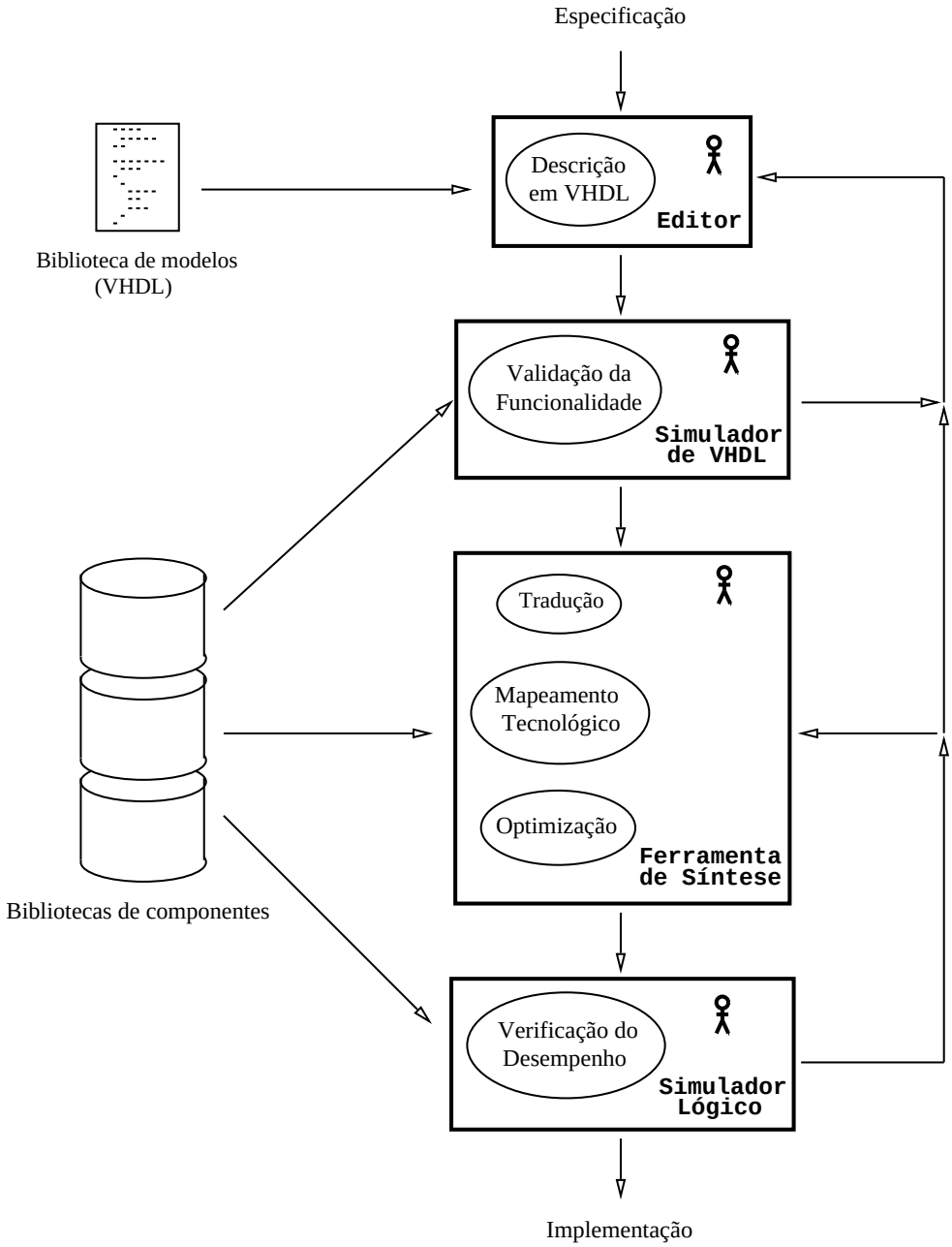

Figura 1: Fluxo de projecto num ambiente de síntese.

3. Utilizar uma ferramenta de síntese para obter uma representação do circuito ao nível lógico que satisfaça um conjunto de restrições impostas pelo projectista.

4. Verificar através de um simulador lógico que o circuito sintetizado tem as características pretendidas pelo projectista.

Sempre que é obtida uma descrição do circuito é necessário verificar a sua funcionalidade através de uma simulação. Quando os resultados da simulação não satisfizerem por completo os intentos do projectista é necessário voltar a um passo anterior, efectuar algumas alterações e repetir novamente os restantes passos. Este processo iterativo pode ser efectuado a três níveis:

- Ao nível da linguagem de descrição: enquanto a descrição do circuito não apresentar a funcionalidade desejada ou não for sintetizável será necessário reescrever o código VHDL.

- Ao nível da ferramenta de síntese: quando as alterações ao circuito são obtidas através da mudança de parâmetros na ferramenta de síntese até serem respeitadas as restrições do projectista.

- A nível global: se o resultado da simulação lógica não apresentar o desempenho ou a funcionalidade pretendida poderá ser necessário reescrever o VHDL e repetir novamente todos os passos de projecto.

A iteração a nível global não seria necessária se as ferramentas de síntese garantissem a geração de circuitos funcional e temporalmente correctos. No entanto, há duas razões que justificam a simulação lógica após a síntese e portanto a necessidade deste ciclo de iteração. A primeira resulta da possibilidade de diferentes interpretações semânticas serem dadas a certas construções por parte do simulador de VHDL e da ferramenta de síntese. A segunda por, só após o mapeamento tecnológico resultante da síntese, ser possível uma simulação lógica com estimativas mais realistas dos tempos de atraso.

O fluxo de projecto apresentado para além de ser iterativo é também interactivo. Isto é, durante o processo de síntese o projectista tem que "guiar" a ferramenta de síntese para obter circuitos de qualidade aceitável. Este processo consiste na caracterização do circuito e imposição de restrições. Enquanto que a caracterização fornece à ferramenta de síntese informações sobre o ambiente onde o circuito vai ser utilizado (informação temporal e eléctrica), a imposição de restrições establece um conjunto de objectivos que devem ser atingidos pela ferramenta de síntese.

As bibliotecas que permitem representar o circuito nos vários níveis de projecto apresentam características diferentes consoante o objectivo a que se destinam. A biblioteca de modelos é escrita em VHDL, de uma forma independente da tecnologia, e destina-se a auxiliar o projectista na tarefa de descrever e testar o seu circuito. Um exemplo de uma biblioteca de modelos sintetizáveis será apresentada na secção 5. As bibliotecas de componentes têm como objectivo representar as portas lógicas existentes numa dada tecnologia, através da função que realizam e de um conjunto de parâmetros que variam consoante a

ferramenta a que se destina a biblioteca. Enquanto que, por exemplo, para as ferramentas de síntese são importantes a área de cada componente e os tempos de propagação, para os simuladores só estes últimos são necessários. A utilização de várias representações não coerentes de uma mesma biblioteca de componentes pode aumentar o número de iterações a realizar ou mesmo tornar a tarefa de síntese impossível.

3 Ferramentas de Síntese

A linguagem VHDL foi criada originalmente para a descrição e simulação de sistemas electrónicos. Por este motivo algumas das suas instruções estão intimamente ligadas à simulação e não são em geral sintetizáveis. Por exemplo, a capacidade do VHDL suportar estruturas aritméticas com números reais e de permitir a utilização de ficheiros para entrada e saída torna-o muito conveniente e sofisticado para a modelação de sistemas mas requer capacidades irrealistas para as ferramentas de síntese [Levitan 89].

Dado que a semântica do VHDL está adaptada à simulação, praticamente todos os simuladores suportam a linguagem na totalidade. Aqueles que não o fazem deixam de fora apenas um conjunto pequeno de instruções que também não são certamente suportadas para síntese. Assim, as construções de VHDL que podem ser usadas numa descrição sintetizável estão em grande parte limitas pelas ferramentas de síntese.

3.1 O VHDL Suportado

O subconjunto de instruções suportadas para síntese não se encontra ainda normalizado, apesar dos esforços que têm vindo a ser desenvolvidos pelo grupo de trabalho de síntese [Harper 92]. Assim, o subconjunto sintetizável de VHDL está dependente da ferramenta de síntese.

Do ponto de vista das ferramentas de síntese, as construções de VHDL podem ser divididas em três categorias:

- *Ignorada* - significa que a construção é permitida no código VHDL, mas ignorada para a síntese.
- *Não Suportada* - significa que a construção não é permitida numa descrição VHDL. Se construções não suportadas existirem numa descrição, o analisador identifica-as como erros e o circuito não será sintetizado.
- *Suportada* - significa que a construção pode ser usada numa descrição sintetizável. Algumas restrições podem ser impostas se a construção não for completamente suportada.

O resto desta secção é dedicada à apresentação do suporte que a ferramenta de síntese da Synopsys faz de cada construção de VHDL [CRM 92].

3.1.1 Unidades de Projecto

A linguagem VHDL define unidade de projecto como o conjunto mínimo de instruções que podem ser analisadas em separado. Cada unidade de projecto é analisada para uma biblioteca, cujo valor por defeito é denominado de `WORK`. O uso de bibliotecas e de análise separada de unidades de projecto é completamente suportada.

As unidades de projecto especificadas na linguagem VHDL são apresentadas nas subsecções seguintes.

Entidades

A entidade é uma abstracção de *hardware* em VHDL que pode representar um sistema completo, uma placa, um integrado, uma célula ou mesmo uma simples porta lógica. Na realidade, em cada entidade é definido o interface do bloco que representa com o ambiente onde esse bloco vai ser integrado. Na declaração de uma entidade são dados a conhecer os acessos ao interior de cada bloco sem no entanto ser descrita a sua funcionalidade. Na tabela 1 são apresentadas as restrições impostas a uma entidade destinada a ser sintetizada.

entity *entity_name* **is**
    [**generic**(*generic_declaration*)**;**]
    [**port**(*port_declaration*)**;**]
    [*entity_declarative_part*]
[**begin**
        entity_statement_part]
end [*entity_name*];

Item	Restrições
generic_declaration	Apenas do tipo inteiro
port_declaration	Valores de defeito nos acesso são ignorados
entity_declarative_part	Declarações de `alias` e especificações de interrupção não são suportadas
entity_statement_part	Ignorada

Tabela 1: Suporte da unidade entidade

Arquitecturas

A arquitectura define a funcionalidade de uma dada entidade. Utilizando o conjunto de instruções disponíveis em VHDL descreve-se na arquitectura como são obtidas as várias saídas do circuito em função das suas entradas.

Cada declaração de entidade pode ter associada um número indeterminado de arquitecturas. A "mesma" funcionalidade em cada uma delas pode ser descrita em estilos que variam desde de uma descrição algorítmica (um conjunto de instruções sequenciais dentro de um processo) até à descrição estrutural de uma lista de portas (a instanciação de um

conjunto de componentes). A tabela 2 apresenta o suporte que a ferramenta de síntese da Synopsys faz da arquitectura.

architecture *architecture_name* **of** *nome_entidade* **is**
    [*architecture_declarative_part*]
begin
    [*statements*]
end [*architecture_name*]**;**

Item	Restrições
architecture_declarative_part	Apenas subprogramas (declaração ou corpo), tipos, subtipos, constantes, sinais ou declarações de componentes são suportados.
statements	Suportado conforme apresentado nas secções 3.1.5 e 3.1.4.

Tabela 2: Suporte da unidade arquitectura

Configurações

Podendo existir várias arquitecturas para uma mesma entidade é necessário definir para cada instanciação de uma entidade qual a arquitectura envolvida.

A unidade de configuração permite definir o par entidade-arquitectura que será usado por cada componente instanciado dentro de uma arquitectura. Na tabela 3 são descritas as restrições à utilização da unidade de configuração num ambiente de síntese.

configuration *configuration_name* **of** *nome_entidade* **is**
    block_configuration
end [*configuration_name*]**;**

Item	Restrições
block_configuration	Apenas para especificar a arquitectura do nível hierárquico mais elevado. Configuração de componentes e de blocos de configuração embrincados não são suportados.

Tabela 3: Suporte da unidade de configuração

Módulos

O uso de módulos permite uma melhor organização dos circuitos a descrever através da partilha de partes de código que sejam frequentemente usadas, tais como: declarações de tipos e/ou componentes, procedimentos de conversão de tipos, etc.

Os módulos pode ser constituído por duas partes: a declaração, que indica qual a visão externa que se tem do módulo; e o corpo, onde se encontra a implementação dos

procedimentos e funções oferecidas. Conforme apresentado na tabela 4, estas duas vistas do módulo são completamente suportadas.

package *package_name* **is** – – declaracao do modulo
    [*package_declarative_part*]
end [*package_name*] **;**

package_body *package_name* **is** – – corpo do modulo
    [*package_body_declarative_part*]
end [*package_name*]**;**

Item	Restrições
package_declarative_part	Completamente suportado
package_body_declarative_part	Completamente suportado

Tabela 4: Suporte da unidade módulo

O módulo normalizado pelo IEEE, `std_logic_1164`, define um sistema de lógica de nove valores destinado à modelização de circuitos digitais. Este módulo também define os operadores lógicos e funções de conversão necessárias para a utilização de uma lógica de nove valores. O módulo `std_logic_1164` é completamente suportado à excepção das seguintes funções: `rising_edge`, `falling_edge` e `is_x`.

3.1.2 Objectos, Tipos de Dados e Atributos

Objectos em VHDL são elementos que contêm valores de um dado tipo. Existem três classes de objectos: constantes, variáveis e sinais.

Objectos	Restrições
Constantes	Constantes "prorrogadas" não são suportadas
Variáveis	Valores iniciais não são suportados
Sinais	Declarações tipo `register` e `bus` não são suportadas. As funções de resolução suportadas correspondem apenas ao tipo *wired* e alta-impedância. Valores iniciais não são suportados.

Tabela 5: Objectos suportados

O tipo de um objecto determina quais os valores que esse objecto pode conter. Os quatro tipos básicos escalares são: inteiros, vírgula flutuante, físicos e enumerados. Tipos compostos como matrizes e agregados podem ser definidos à custa dos tipos básicos. Os apontadores, idênticos àqueles que são usados nas linguagens vulgares de programação, e ficheiros, permitem o acesso a objectos de um dado tipo. Subtipos podem ser também definidos através da imposição de restrições a um outro tipo. Os tipos definidos na

linguagem VHDL encontram-se na tabela 6 com as restrições impostas pela ferramenta de síntese.

Tipos	Restrições
Inteiros	Aritmética de precisão infinita não é suportada. Os dados do tipo inteiro são convertidos para vectores com o comprimento mínimo para acomodar todos os valores do intervalo (codificação binária para intervalos positivos, em complemento para 2 para intervalos com valores negativos).
Vírgula Flutuante	Não suportado.
Físicos	Ignorado.
Enumerados	Completamente suportado.
Matrizes	Suportados apenas para índices inteiros. Matrizes de dimensão superior a um não são suportadas, mas matrizes de matrizes são suportadas.
Agregados	Completamente suportados .
Apontadores	Não suportados.
Ficheiros	Não suportados.
Tipos de dados incompletos	Não suportados.

Tabela 6: Tipos de dados suportados

Os atributos são características (valores, funções, etc) que podem estar associadas a determinados elementos que constituem o VHDL. Para além dos atributos pré-definidos na linguagem o utilizador pode definir os seus próprios atributos.

Muitas ferramentas de síntese permitem a definição atributos com especial significado para a ferramenta. Apesar da caracterização do circuito e a imposição de restrições poder ser feita através da utilização desses atributos, este método deve ser evitado no sentido de obter uma descrição portável[1] Devido ao VHDL ter sido desenvolvido para simulação, existem atributos que não têm significado num ambiente de síntese. Os atributos suportados pela ferramenta de síntese encontram-se na tabela 7.

3.1.3 Expressões

As expressões realizam cálculos aritméticos ou lógicos por aplicação de operadores a um ou mais operandos. Enquanto que os operadores especificam o cálculo a realizar, o operandos fornecem os dados.

Operadores

Um operador em VHDL é caracterizado por: nome, função, número e tipo de operadores e o tipo do resultado. É possível definir novos operadores para qualquer tipo de

[1] A caracterização e imposição de restrições no circuito pode, e deve, ser feita na ferramenta de síntese (interactivamente ou através de um *script*).

Classe de Atributos	Restrições
Atributos pré-definidos	Apenas os seguintes atributos são suportados: `base, left, high, low, range, reverse_range` and `length`.
Atributos definidos pelo utilizador	Não suportados.
Atributos para síntese	São suportados para caracterizar componentes, acessos de entrada e saída, ou para indicar restrições ao circuito.

Tabela 7: Atributos suportados

operandos e tipo de resultado, ou mesmo redefinir os operadores pré-definidos na linguagem. O operadores pré-definidos em VHDL encontram-se na tabela 8 juntamente com as restrições impostas pela ferramenta de síntese.

Classe	Operadores	Restrições
Lógicos	*and*, *or*, *nand*, *nor*, *xor*	Completamente suportados
Relacionais	=, / =, <, <=, >, >=	Completamente suportados
Adição	+, −, &	Completamente suportados
Sinal	+, −	Completamente suportados
Multiplicativos	∗, /, *mod*, *rem*	O operador ∗ é completamente suportado. Os operadores /, *mod* e *rem* são suportados apenas quando ambos os operadores são constantes ou o operando da direita é uma potência "calculável[1]" de 2.
Vários	∗∗, *abs*, *not*	O operador ∗∗ é suportado apenas quando ambos os operandos são constantes ou quando o operador da esquerda é uma potência "calculável[1]" de 2. Os operadores *abs* e *not* são completamente suportados.

[1] O seu valor pode ser determinado estaticamente durante a análise do código.

Tabela 8: Operadores suportados

Operandos

Como já foi mencionado, os operandos fornecem os dados que vão ser utilizados pelos operadores. Os operandos numa expressão podem incluir identificadores, nomes, literais, agregados, chamadas a subprogramas, expressões qualificadas, conversão de tipos, apontadores ou outras expressões dentro de parênteses. Os operandos suportados pela ferramenta de síntese são apresentados na tabela9.

Operandos	Restrições
Identificadores	Completamente suportados.
Nomes seleccionados	Completamente suportados.
Nomes indexados	Completamente suportados.
Pedaços de nome	Pedaços nulos não são suportados.
Atributos	Completamente suportados para os atributos definidos na secção 3.1.2.
Literais	Literais nulos não são suportados. Literais físicos são ignorados.
Agregados	Suportados apenas os do tipo matriz.
Chamadas a subprogramas	Funções de conversão nos acessos de entrada não são suportadas.
Expressões qualificadas	Completamente suportadas.
Conversão de tipos	Completamente suportadas.
Apontadores	Não suportados.
Expressões estáticas	Completamente suportadas.
Expressões universais	Expressões em virgula flutuante não são suportadas. A precião é limitada a 32 *bits*; todos os resultados intermedios são convertidos para inteiros.

Tabela 9: Operandos suportados

3.1.4 Instruções sequenciais

As instruções especificam o modo de organização e de operação de um circuito. As instruções sequenciais são usadas quando se pretende descrever um circuito utilizando um estilo de descrição comportamental. O algoritmo é executado passo a passo, tal como nas linguagens de programação de alto nível. Todos as instruções sequenciais estão encapsuladas dentro de processos, subprogramas ou funções, que são usadas em contextos concorrentes. A tabela 10 lista as instruções concorrentes de VHDL e as restrições impostas pela ferramenta de síntese.

3.1.5 Instruções Concorrentes

Dado que os dispositivos electrónicos funcionam na sua maior parte em paralelo, o VHDL inclui capacidades de concorrência para modelar este comportamento. Ao contrário das instruções sequenciais, que são executadas uma após outra, as instruções concorrentes são executadas continuamente, em paralelo e sem qualquer ordem pré-definida de execução. O suporte das instruções concorrentes é apresentado na tabela 11.

3.2 Estilo de descrição

O estilo de descrição a adoptar num ambiente de síntese tem como objectivo reduzir o número de iterações necessárias para obter uma implementação final do circuito com o desempenho e funcionalidade desejada. No estado actual da tecnologia de síntese, a

Instruções	Restrições
wait	Apenas suportada nas seguintes construções: **wait until** sinal = *valor_lógico* **;** **wait until** sinal**' event and** sinal = *valor_logico* **;** **wait until not** sinal**'stable and** sinal = *valor_logico* **;** onde *valor_logico* é '0' or '1' para sincronização com o flanco descendente ou ascendente do relógio, respectivamente. A instruções de `wait` não pode ser usada dentro de subprogramas ou ciclos.
assertion	Ignorada.
Atribuições a sinais	Atribuições com múltiplos elementos na forma de ondas não são suportados. A palavra chave `transport` e a opção `after` são ignoradas.
Atribuições a variáveis	Completamente suportadas.
Chamadas a subprogramas	Conversão de tipos nos parâmetros formais não é suportada.
if	Completamente suportado.
case	Completamente suportado.
loop	O corpo do ciclo tem que conter pelo menos uma instrução `wait`.
for	O intervalo de iteração tem que ser "calculável", e corpo não pode conter nenhuma instrução `wait`.
while	O corpo do ciclo tem que conter pelo menos uma instrução `wait`.
next	Completamente suportada.
exit	Completamente suportada.
return	Completamente suportada.
null	Completamente suportada.

Tabela 10: Suporte das instruções sequenciais

qualidade final do circuito sintetizado, em termos de área ou velocidade, depende tanto da capacidade da ferramenta usada, como da forma e qualidade da descrição de VHDL que é fornecida [Carlson 91].

Presentemente as ferramentas comerciais de síntese apenas aceitam descrições no nível de abstracção RTL ou inferior, apesar da linguagem VHDL permite descrições em níveis de abstracção superiores. Por esta razão, o projectista tem que ter a estrutura do seu circuito representada ao nível RTL, isto é, constituída pelo(s) fluxo(s) de dados, com elementos de funções bem definidas (registos, somadores, comparadores, etc) e o(s) respectivo(s) controlador(es).

Um estudo detalhado sobre os estilo de descrição a adoptar num ambiente síntese foi realizado neste projecto. As suas conclusões já foram apresentadas no relatório [Flores 92], pelo que não irão ser repetidas.

Instruções	Restrições
block	Acessos e genéricos em instruções de bloco e blocos com guarda não são suportados.
process	Lista de sensibilidades é ignorada.
Chamada a Subprogramas	Coversão de tipos nos parâmetros formais não são suportadas.
assertion	Ignoradas.
Atribuições a sinais	Atribuições com múltiplos elementos na forma de ondas não são suportados. A palavra chave `transport` e a opção `after` são ignoradas.
Instanciação de componentes	Conversão de tipos na especificação dos acessos de um componente não é suportados.
generate	Completamente suportada.

Tabela 11: Suporte das instruções concorrentes

4 Descrições portáveis

Para garantir a portabilidade de uma descrição VHDL, entre várias ferramentas de síntese, é necessário usar um subconjunto de construções comum e um estilo de descrição compatível. Apesar destas restrições não poderem ser satisfeitas para todas as ferramentas, existem algumas construções que não devem de ser usadas numa descrição portável, tais como:

- Atributos específicos que são reconhecidos apenas por uma ferramenta de síntese. Como se referiu anteriormente estes atributos permitem a caracterização ou a imposição de restrições ao circuito. O módulo de síntese que se encontra a ser desenvolvido pelo grupo de trabalho de síntese do IEEE foca este aspecto, definindo um conjunto de atributos que serão normalizados pelo IEEE.

- Comentários sintéticos[2] que permitem incluir na descrição VHDL comandos para a ferramenta de síntese. Um exemplo de um comentário sintético suportado pela maioria das ferramentas de síntese, mas de forma diferente, é aquele que permite o cancelamento ou re-inicialização da análise de partes de código.

- Funções ou procedimentos descritos em módulos que estão integrados na ferramenta. Para este módulos apenas existem as suas declaração, o corpo não tem representação em VHDL pois este está incorporado na ferramenta.

Por vezes, para obter melhores resultados da ferramenta de síntese poderá ser necessário usar algumas destas construções, à custa de ser obter uma descrição não portável.

Nas descrições que utilizam lógica de vários valores (0, 1, Z, etc) deve ser utilizado o módulo normalizado `std_logic_1164`. Este módulo, que define uma lógica de nove

[2]Comentários cuja primeira palavra é especial e serve para indicar que o resto da linha deve ser interpretada pela ferramenta.

valores e um conjunto de funções para a sua manipulação, é suportado pela maioria das ferramentas de síntese.

5 Biblioteca de Modelos

A biblioteca de modelos (introduzida na secção 2) destina-se a auxiliar o projectista a obter uma descrição funcionalmente correcta e sintetizável do seu circuito. Esta biblioteca pode conter descrições de circuitos de dois tipos: sintetizáveis e não sintetizáveis. Enquanto que as primeiras podem ser utilizadas para a descrição de circuitos mais complexos (destinados as serem sintetizados), as segundas permitem verificar a funcionalidade dessas descrições (e eventualamente do circuito sintetizado) de uma forma mais simples. Por exemplo, o projecto de um circuito de interface a um microprocessador, pode ser mais facilmente realizado se houver uma descrição precisa, em VHDL, desse mesmo microprocessador.

Devido à descrição/modelação de circuitos não sintetizáveis estar fora do âmbito deste trabalho, a biblioteca de modelos apresentada é constituída apenas por blocos de *hardware* descritos em VHDL sintetizável.

O desenvolvimento dos blocos desta biblioteca de modelos foi feita tendo em conta a metodologia de projecto apresentada. Ou seja, na descrição tecnologicamente independente de cada bloco da biblioteca, foi tomada em consideração o subconjunto de construções sintetizáveis de VHDL e utilizado um estilo de descrição apropriado.

A utilização da construção de VHDL `generic` e de alguns pinos de "programação" permitiu a definição de blocos parametrizáveis. Assim, é possível usar o mesmo bloco da biblioteca (ou seja, a mesma descrição VHDL), com parâmetros diferentes, para obter implementações diferentes. Por exemplo, é possível parametrizar um contador, quando é feita a sua instanciação, de forma a definir os valores de inicialização síncrona e assíncrona, a quantidade incrementada e/ou decrementada por ciclo de relógio, o valor máximo e/ou mínimo de contagem, etc. A utilização de uma ferramenta de síntese permite mapear estes blocos numa tecnologia específica, ao mesmo tempo que realiza uma optimização de cada bloco instanciado, de acordo com a sua parametrização e com o "ambiente" onde este está integrado.

No apêndice A apresenta-se o manual de referência da biblioteca de modelos desenvolvida (ICBlocks). Para cada bloco existe uma "folha de catalgo" que contém: uma descrição a funcionalidade do bloco, os seus parâmetros e lista de pinos, o código VHDL e a implementação, por defeito, numa tecnologia genérica. Em alguns casos são também dados exemplos de aplicação.

6 Conclusões

Neste relatório apresentou-se o fluxo de projecto geralmente utilizado num ambiente de síntese. O conhecimento do subconjunto sintetizável de VHDL e a utilização de um estilo

de descrição apropriado permite obter mais rapidamente a representação do circuito no nível lógico, através da redução do tempo despendido em cada passo de projecto e/ou do número de iterações necessárias. Um levantamento das construções de VHDL suportadas pela ferramenta de síntese escolhida no projecto foi apresentado, salientando-se também a importância da sua utilização de acordo com um estilo próprio para síntese.

Algumas considerações sobre a portabilidade do código VHDL foram apresentadas. A utilização do módulo norma `sdt_logic_1164` é o primeiro passo para uma descrição portável, evitando-se assim utilização de construções específicas das ferramentas. Especial atenção deve ser dado ao trabalho do grupo de síntese no desenvolvimento de um módulo norma tendo conta este problema.

Foi ainda apresentada uma biblioteca de modelos sintetizável. Esta, foi descrita de acordo com a metodologia apresentada e a sua utilização permite reduzir o tempo de desenvolvimento de um circuito num ambiente de síntese.

Referências

[Carlson 91] Steve Carlson. *Application of VHDL to Circuit Design*, chapter Modeling Style Issues for Synthesis. Kluwer Academic Publishers, 1991. Ed. R. Harr and A. Stanculescu.

[CRM 92] Synopsys, Inc. *VHDL Compiler Reference Manual*, November 1992. Version 3.0.

[Flores 92] Paulo Flores. Metodologias de Síntese em VHDL. Technical report, Projecto JNICT PMCT/856/90, Outubro 1992.

[Harper 92] P. Harper and K. Scott. Towards A Standard VHDL Synthesis Package. In *Proceedings of European Design Automation Conference / Proceedings of Euro-VHDL*, September 1992.

[Levitan 89] S. Levitan, A. Martello, R. Owens, and M. Irwin. Using VHDL as a Language for Synthesis of CMOS VLSI Circuits. In *Proceedings of the Ninth IFIP Symposium on Computer Hardware Description Languages and their Applications*, June 1989.

[Synthesis 92] Synopsys, Inc. *Design Compiler Reference Manual*, December 1992. Version 3.0.

A Biblioteca de Modelos - Manual de Referência