SÍNTESE DE ALTO NÍVEL DE PROTOCOLOS DE COMUNICAÇÃO ESPECIFICADOS NA LINGUAGEM FORMAL ESTELLE USANDO O AMBIENTE AMICAL

Luci Pirmez (3), Aloysio Pedroza (1) (2), A. Mesquita (1)

(1) Coppe/UFRJ - Programa de Engenharia Elétrica

(2) EE/UFRJ - Departamento de Eletrônica

(3) NCE/UFRJ - Núcleo de Computação Eletrônica
Cidade Universitária, Ilha do Fundão
Caixa Postal 2324, 20001, Rio de Janeiro , RJ , Brasil
E-mail: luci@barra.nce.ufrj.br

Resumo Este artigo apresenta uma metodologia para traduzir eficientemente especificações Estelle em descrições VHDL. A metodologia proposta permite obter descrições de protocolos adequadas à síntese de alto nível. O objetivo é desenvolver estilos de descrição de máquinas de estado em VHDL, que minimizem o número de caminhos gerados pelos algoritmos de *scheduling* correspondentes, durante a síntese de alto nível. Como resultado, a área dos circuitos gerados é minimizada. Um exemplo usando um ambiente de HLS específico foi desenvolvido para verificar a eficiência da metodologia proposta.

Abstract This paper presents a methodology to efficiently translate Estelle protocols specifications into VHDL descriptions. The proposed methodology allows to generate protocol descriptions suitable for High Level Synthesis (HLS). The goal is to develop description styles for state machines in VHDL that minimizes the number of paths in the scheduling algorithms, during the HLS. As a result, the area of the synthesized circuits is optimized. An example using a specific HLS environment was developed in order to verify the efficiency of the proposed methodology.

Palvras Chaves Redes de Computadores, Protocolos, Síntese de Alto Nível, VLSI, VHDL.

1. INTRODUÇÃO

O surgimento de novas tecnologias em VLSI, o alto desempenho dos nós comutadores, a redução dos custos de memória e de processador, e os avanços das fibras ópticas que operam em velocidades da ordem de centenas de *megabits* por segundo, com baixas taxas de erro, vêm permitindo o desenvolvimento de redes cada vez mais velozes e confiáveis.

Essas mudanças na tecnologia e o surgimento de aplicações que exigem o uso de redes de computadores de alto desempenho demandam modificações no projeto de protocolos. Entretanto, a obtenção de subsistemas de comunicação de alto desempenho não depende somente da otimização de software e do emprego de paralelismo para a construção de protocolos mas também da implementação desses protocolos em hardware integrado.

Com o objetivo de reduzir o tempo de concepção dos circuitos integrados, as metodologias de projeto atuais têm procurado automatizar as etapas intermediárias do processo que vão da especificação inicial em alto nível de um dado sistema à geração das máscaras de fabricação. O processo que permite gerar a especificação física (layout) do circuito a partir de uma especificação de alto nível é conhecido como "compilação de silício".

No caso específico dos protocolos de comunicação, é mais conveniente iniciar o projeto, visando a implementação em circuito integrado, por especificações baseadas em Técnicas de Descrição Formal (FDTs) [1]. Este procedimento tem a vantagem de permitir o uso de ferramentas mais familiares ao projetista de protocolos. A linguagem Estelle [2] (Extended Transition Language) é uma das linguagens formais de especificação padronizadas pela ISO e utilizadas na descrição de sistemas distribuídos.

Tradicionalmente, os protocolos têm sido implementados inteiramente em software. A fim de atender aos requisitos de velocidade das redes atuais, tornou-se necessário recorrer à partição software/hardware de uma dada especificação. Isto é, a parte da especificação menos restritiva em termos de velocidade será destinada à implementação em software, enquanto a parte mais crítica deverá ser implementada em hardware. Para fins de implementação, cada uma dessas partes será endereçada a ambientes específicos de projeto. No projeto do hardware, os sistemas de C.A.D. (Computer Aided Design) incluem as ferramentas de síntese, simulação de circuitos e geração de layout. A linguagem de descrição de hardware mais comumente utilizada nestes ambientes é a linguagem

Publicação recomendada pelo Editor de Área de Circuitos e Dispositivos.

VHDL [3]. Uma especificação feita em uma linguagem de descrição de software, tal como Estelle, facilita a modelagem da hierarquia e da comunicação entre processos, mas não permite seu interfaceamento com ferramentas de síntese. Para resolver esta dificuldade, várias alternativas têm sido propostas. A abordagem básica consiste em traduzir uma especificação formal no nível de sistema para uma linguagem de descrição de hardware como, por exemplo, a tradução de LOTOS (Language of Temporal Ordering Specification) para VHDL [4] ou de ESTELLE para VHDL [5,6]. Outra alternativa possível envolve a criação de uma forma intermediária tal como o SOLAR [7] que viria a servir de plataforma comum para diferentes linguagens. Finalmente, um terceiro enfoque possível é criação de uma nova linguagem de especificação tal como o SpecCharts [8].

A metodologia proposta neste trabalho, para a tradução das especificações de um protocolo em Estelle para descrições na linguagem VHDL se enquadra, portanto, no primeiro tipo de abordagem. O ponto de partida desta metodologia é uma especificação Estelle correta, isto é, que tenha sido validada através de ferramentas de validação adequadas. A etapa seguinte do processo, consiste no mapeamento desta especificação diretamente na linguagem VHDL. O objetivo final é gerar um circuito integrado que implemente o protocolo projetado em um CAD Estelle. Para isto, as descrições VHDL obtidas no processo de tradução são utilizadas como entrada para as ferramentas de síntese de hardware disponíveis.

A síntese de protocolos requer ferramentas específicas de HLS (High-Level Synthesis), conhecidas como compiladores de comportamento dominado por fluxo de controle[9]. A maioria destes compiladores têm como característica indesejável comum o fato de que os resultados da compilação dependem do estilo utilizado na descrição comportamental do hardware. O resultado da HLS é uma descrição estrutural de hardware cujas características elétricas e temporais serão determinadas fundamentalmente pelo algoritmo de "scheduling" usado. Assim, dependendo do estilo de descrição de entrada, o circuito sintetizado poderá apresentar uma solução de compromisso área x custo satisfazendo ou não os requisitos originais do projeto. Um dos resultados fundamentais deste trabalho refere-se à discussão do efeito dos diferentes estilos de descrição dos protocolos de comunicação em VHDL nos resultados da HLS e, consequentemente, na complexidade do circuito obtido.

No desenvolvimento de um exemplo de HLS de um protocolo que permitisse testar o efeito dos estilos de descrição no resultado final da síntese, foi utilizado, sem perda de generalidade, o ambiente de síntese de alto nível AMICAL[10].

O AMICAL, como a maioria das ferramentas de HLS, utiliza internamente uma arquitetura alvo de forma a gerar implementações de hardware eficientes em termos de área e velocidade. Isto implica que, apenas um conjunto restrito de instruções VHDL pode ser mapeado de forma eficaz na arquitetura alvo empregada. O subconjunto de instruções sintetizáveis da linguagem VHDL aceito pelo AMICAL será definido neste trabalho. Deve ser observado, que cada ambiente particular de HLS terá seu próprio subconjunto de instruções VHDL sintetizáveis, sem que isto implique em limitações na metodologia proposta. A principal etapa do processo de HLS é o scheduling. O ambiente AMICAL utiliza o algoritmo de Dynamic Loop Scheduling (DLS), que é reconhecidamente um dos mais eficientes para a síntese de máquinas de estado dominadas por fluxo de controle [13]. Esta foi a principal razão de se usar este ambiente para a síntese de alto nível de protocolos de comunicação.

O trabalho contém na seção II os conceitos fundamentais associados às linguagens Estelle e VHDL necessários à apresentação da metodologia de mapeamento. Na seção III, a metodologia para o mapeamento de especificações Estelle de protocolos em descrições VHDL é detalhada. Na seção IV são analisados os diferentes estilos que podem ser utilizados para descrever um protocolo em VHDL. Na seção V são examinados os efeito dos estilos de descrição de protocolos em VHDL nos resultados da síntese. Finalmente, na seção VI são apresentadas as conclusões.

2. AS LINGUAGENS ESTELLE E VHDL

A fim de facilitar a discussão do problema de mapeamento de especificações Estelle em descrições VHDL, os conceitos básicos associados a estas linguagens serão apresentados nas próximas subseções.

2.1. A LINGUAGEM ESTELLE

Uma *especificação* Estelle é composta por um conjunto de módulos organizados de forma hierárquica. Cada módulo Estelle é constituído de duas partes: o cabeçalho (*module*) e um ou mais corpos (body). A visão externa de um módulo é definida no cabeçalho que contém o seu atributo de classe, seus pontos de interação e suas variáveis exportadas. O comportamento interno, por outro lado, é descrito no corpo. A especificação do comportamento de um módulo Estelle é baseada no modelo de máquina de estados finita estendida (FSME).

Cada módulo pode ser visto como uma caixa preta que possui portas de entrada/saída, isto é, os pontos de interação. Um ponto de interação possui um conjunto de interações que podem ser enviadas ou recebidas através deste ponto. O papel deste ponto de interação determinará se a mensagem é de recepção ou de transmissão. Os canais de comunicação são objetos abstratos que especificam um conjunto de interações (mensagens). Os canais podem conectar os módulos através de suas portas. A troca de mensagem ocorre, quando duas instâncias de módulos estabelecem um "link" de comunicação entre dois pontos de interação. Uma interação recebida por uma instância de módulo em um dos seus pontos de interação é armazenada em uma fila FIFO de tamanho ilimitado associada a esse ponto de interação. Outro mecanismo que

possibilita a cooperação entre duas instâncias de módulos pode ser implementado através de variáveis partilhadas.

Um módulo pode possuir os seguintes atributos: "systemprocess", "systemactivity", "process", "activity". Um módulo com o atributo "systemprocess" ou "systemactivity" é denominado de módulo sistema. Um módulo sistema com atributo "systemprocess" pode ser estruturado em submódulos hierárquicos com atributos do tipo "activity" ou do tipo "process". Já um módulo com o atributo do tipo "system activity" pode ser estruturado em submódulos hierárquicos com atributos do tipo apenas "activity". As classes de atributos acima determinam o tipo de paralelismo que será empregado pelos módulos. Estelle possui dois tipos de paralelismo: o síncrono e o assíncrono.

Courtiat em [11] introduziu o dialeto denominado Estelle* que é uma versão simplificada de Estelle. Esta versão permite obter especificações de protocolos com alto nível de abstração, utilizando uma semântica de paralelismo entre instâncias de módulos, consistente com a semântica definida nos modelos frequentemente utilizados para a modelagem e validação de protocolos de comunicação. O tipo de atributo de módulo adotado no dialeto Estelle* e usado pela maioria dos projetistas é o *"activity"*. O paralelismo permitido entre sistemas do tipo "activity" é do tipo assíncrono, isto é, entre ações de instâncias pertencentes a sistemas distintos. É bom ressaltar que o sistema do tipo "activity" não possui paralelismo interno, implicando na execução sequencial dos seus módulos filhos.

Um corpo (*body)* em Estelle é composto por três partes: uma parte declarativa, uma parte de iniciação e uma parte para a especificação de transições. O comportamento de um módulo é descrito pelas partes de iniciação e de transição do corpo. A execução de ambas as partes é seqüencial.

A parte declarativa é composta de declarações em Pascal (funções, procedimentos, variáveis, constantes) e de declarações de objetos em Estelle como: canais, módulos, variáveis de tipo módulos, estados utilizados pela máquina de estados finita e pontos de interação internos. As variáveis de tipo módulo são utilizadas para instanciar módulos. Por exemplo, a declaração *"modvar X, Y, Z : A1"* define que X, Y e Z são variáveis de tipo módulo especificadas pelo cabeçalho do módulo denominado A1. O comportamento de cada instância de módulo é definido por um autômato de estados cujos estados de controle são dados pela enumeração de seus nomes. Por exemplo, a declaração *"state : IDLE, WAIT, OPEN, CLOSED"* declara 4 estados. Os pontos de interação internos, que são declarados da mesma forma que os pontos de interação externos, permitem a comunicação entre uma instância de módulo e suas instâncias de módulos filhos. Deve ser observado que a parte declarativa de um módulo pode incluir a declaração de outros módulos obtendo-se uma estrutura hierárquica de módulos.

A parte de iniciação de um corpo de módulo contém a iniciação de variáveis globais e de estado, a criação de instâncias de módulos e de ligações entre pontos de interação. Estas instruções são executadas sequencialmente e apenas quando a instância do módulo em questão for iniciada.

A parte das transições de um corpo de módulo é formada por um conjunto de transições onde, cada uma delas é composta por duas partes: a condição da transição e a ação da transição. A condição da transição contém uma ou mais cláusulas *"from..."*, *"when..."*, *" provided..."*, *"priority..."*, *"delay..."*. A ação da transição é composta de uma cláusula *"To..."* seguida de um bloco de transições da forma *"begin* <instruções em Pascal> *end"*.

As extensões em Pascal permitem a criação, destruição e desconexão das instâncias de módulos. As instruções de criação e conexão são *"init"*, *"connect"* e *"attach"*. As componentes para a destruição e desconexão são: "release", "disconnect" e *"detach"*.

Todos os módulos definidos em uma especificação são encapsulados em um módulo denominado de *"specification"* que representa o sistema como um todo.

Como já mencionado, apenas um subconjunto de instruções Estelle é necessário para especificar protocolos de forma eficiente. Este subconjunto, proposto por Courtiat [11], resultou no dialeto conhecido como Estelle*. Além disso, para facilitar o mapeamento Estelle-VHDL, foi necessário acrescentar novas restrições àquelas sugeridas por Courtiat. Disso resultou um novo dialeto de Estelle que, seguindo a nomenclatura proposta por Courtiat, será aqui denominado de Estelle**. As restrições impostas à linguagem Estelle que resultaram no dialeto Estelle** são:

1 Só se admite paralelismo assíncrono entre as instâncias de módulo. Para isto:

- Não existe a cláusula de prioridade associada a qualquer transição definida com a cláusula *"when"*.
- As únicas instâncias de módulo ativas são as folhas da árvore de instâncias de módulos. Uma instância de módulo é ativa se a FSME possuir pelo menos uma transição.

Como conseqüência, elimina-se a prioridade pai/filho de Estelle e o uso variáveis partilhadas.

2 Apenas as formas *"delay*(E1)" ou *"delay*(E1,E1)" da cláusula *"delay"* em Estelle serão aceitas.

3 Considerar a fila FIFO com sendo o mecanismo de comunicação utilizado. Tal mecanismo difere do utilizado por Estelle* mas é o mesmo utilizado por Estelle.

2.2. A LINGUAGEM VHDL

VHDL é uma linguagem de descrição de hardware de uso geral permitindo descrições comportamentais e estruturais em diversos níveis de abstração. Uma descrição VHDL é composta por um conjunto de entidades especificadas separadamente. Uma entidade pode ser usada como um componente em um projeto ou pode ser o módulo de maior nível hierárquico em um projeto.

Uma descrição VHDL é constituída no mínimo de uma entidade, *"entity"*, e de uma ou mais arquiteturas, *"architecture"*. A entidade especifica a interface de um sistema ou subsistema com o exterior através da declaração de portas de entrada e saída. O comportamento interno do sistema, por seu turno, é especificado na arquitetura.

Em uma entidade, as declarações de portas de entrada/saída, cuja classe é sinal, permitem que uma entidade troque informações com outras entidades. O sinal é a única forma possível de troca de informações entre entidades podendo ser dos seguintes tipos: entrada, *"in"*; saída, *"out"* ou entrada/saída, *"inout"*.

A uma mesma entidade podem corresponder várias arquiteturas, cada uma correspondendo a um domínio específico de descrição, comportamental ou estrutural. A estrutura de uma arquitetura pode ser dividida em duas partes: uma parte declarativa e uma parte que descreve o comportamento.

Na parte declarativa de uma arquitetura, são declarados os sinais e componentes. A declaração de um componente, *"component"*, permitirá a utilização de entidades, descritas separadamente e armazenadas em bibliotecas, no interior de uma outra entidade. Para isto, o componente deve ser inicialmente declarado na arquitetura e instanciado na descrição do comportamento. A ligação entre uma instância de componente e a entidade correspondente na biblioteca é feita através de uma estrutura de configuração, *"configuration"*.

Na especificação do comportamento são usadas em geral estruturas do tipo bloco, *"block"*, ou processo, *"process"*.

Blocos são estruturas com sua própria interface e podem estar conectados a outros blocos por meio de sinais. No interior de um bloco, as instruções são executadas de forma concorrente.

No interior de um processo, as instruções são executadas de forma sequencial sendo esta característica sua principal diferença em relação a uma estrutura "block". Por outro lado, processos entre si são executados de forma concorrente. Um processo é ativado durante a fase de

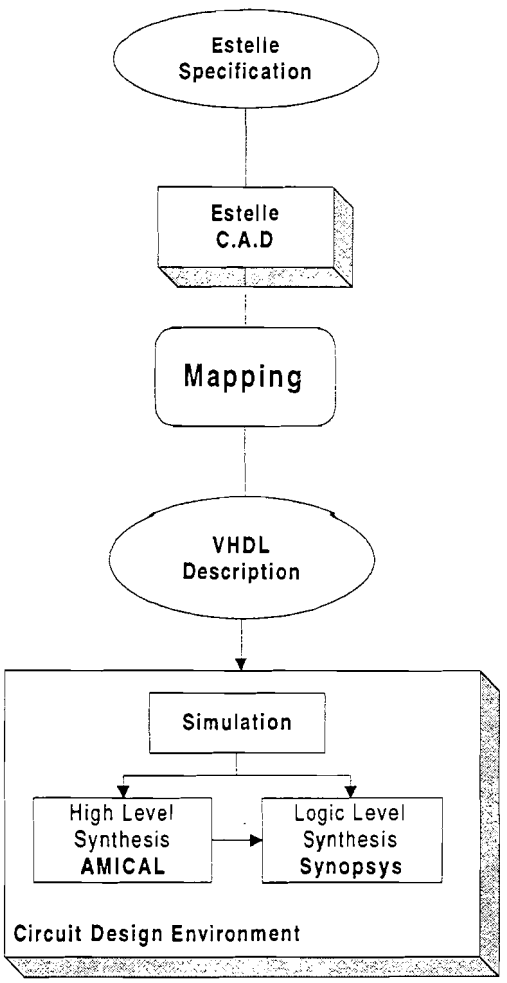

Figura 1 - Metodologia para a implementação de protocolos em hardware integrado.

iniciação da simulação. Após a execução de um processo o ciclo é repetido a partir da primeira instrução. Um processo pode ser suspenso quando executa uma instrução de *"wait"*. A lista dos sinais da instrução de *"wait"* determinará a suspensão ou o retorno da execução. Se esta lista é declarada como lista de parâmetros da instrução *"process"*, assume-se que o processo tem uma instrução *"wait"* implícita cuja condição será testada após a execução da última instrução do processo. Caso contrário, a instrução *"wait"* deverá ser explicitada no corpo do processo.

3. METODOLOGIA DE SÍNTESE DE ALTO NÍVEL DE PROTOCOLOS

Os passos envolvidos na implementação da metodologia proposta neste trabalho estão ilustrados na Figura 1. O objetivo é gerar um circuito integrado que implemente um protocolo projetado em um ambiente familiar ao projetista de protocolos, denominado CAD Estelle [15], na Figura 1. Após a geração de uma especificação em Estelle correta, testada por simulação, esta especificação é mapeada na linguagem VHDL, que é a linguagem geralmente utilizada nos ambientes de projeto de circuitos integrados. Neste trabalho, apenas a etapa inicial do projeto de hardware, constituída pela síntese de alto nível, será abordada.

Como pode ser observado na Figura 1, o elo fundamental do processo é o mapeamento ou tradução de uma especificação Estelle em uma descrição VHDL. Os conceitos e as restrições, decorrentes de limitações dos ambientes de HLS, envolvidos na definição deste mapeamento serão discutidos nas próximas seções.

3.1. MAPEAMENTO DE INSTRUÇÕES ESTELLE EM INSTRUÇÕES VHDL

Como foi visto anteriormente, a unidade básica de uma especificação Estelle, o módulo, é constituída de duas partes: o cabeçalho e o corpo. O cabeçalho *(module)* corresponderá a uma entidade VHDL *(entity)* e cada corpo *(body)* a uma arquitetura VHDL *(architecture)*, como é ilustrado na Figura 2.(1), (2) e (3). O módulo de mais alto nível hierárquico em Estelle, *"specification"*, não possui uma visão externa e, consequentemente, a entidade correspondente em VHDL será vazia.

A hierarquia implícita entre modulos Estelle é obtida em VHDL através da declaração *"component"*, como é mostrado na Figura 2.(4). Cada declaração de um *"component"* em VHDL cria uma entidade cuja descrição será armazenada na biblioteca de trabalho do VHDL. A instrução de instânciação de componentes do VHDL é responsável pela criação de instâncias de entidades VHDL, que equivale à função *"init"* do Estelle, e pela conexão das portas de componentes com sinais reais ou com as portas de outras entidades, funções *"connect"* e *"attach"* do Estelle. Isto é mostrado na Figura 2.(5). A ligação de uma instância de componente com a respectiva entidade VHDL armazenada na biblioteca de trabalho é obtida através do uso de uma declaração de configuração, *"configuration"*, como é mostrado na Figura 2.(6).

O mecanismo interno de fila FIFO com tamanho ilimitado associado a cada ponto de interação da linguagem Estelle não tem correspondente em VHDL. Uma solução possível para este problema consiste em criar duas entidades VHDL, denominadas QUEUE_ENTITY e HIERARCHY_QUEUE_ENTITY. A entidade QUEUE_ENTITY será responsável apenas pela gerência de filas de um módulo Estelle e a entidade

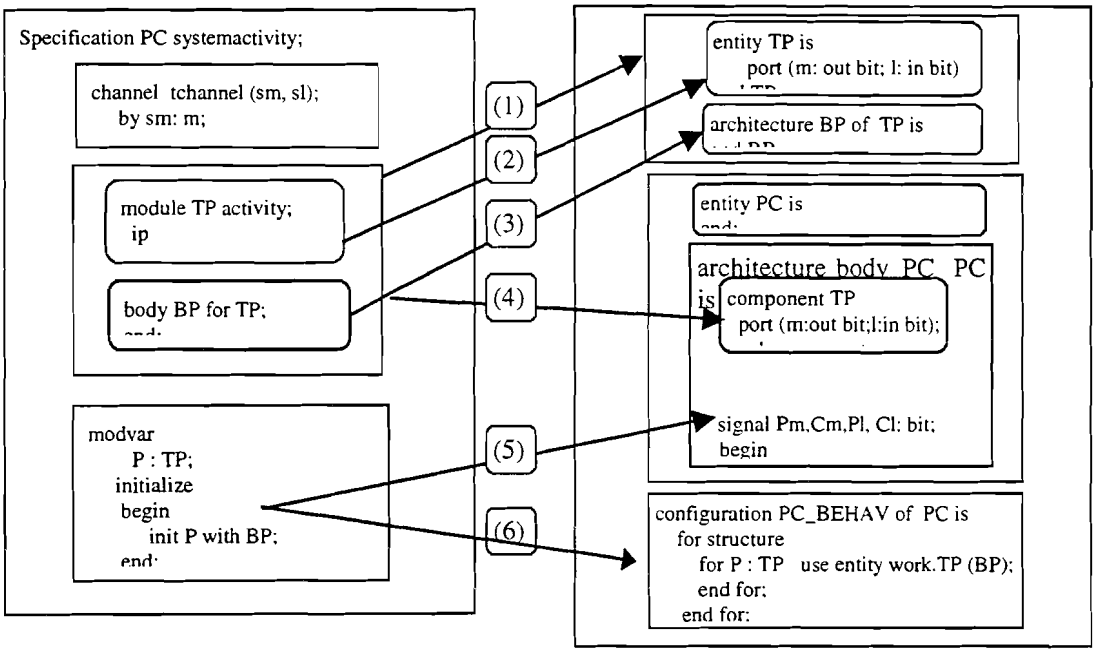

Figura 2 - Mapeamento de Estelle para VHDL.

HIERARCHY_QUEUE_ENTITY implementará a semântica de um módulo "*activity*" em Estelle.

A comunicação entre entidades VHDL é implementada através de sinais declarados especificamente para este fim, como ilustrado na Figura 3. Assim, para representar uma interação entre dois módulos Estelle, dois sinais devem ser declarados em cada uma das entidades VHDL correspondentes. A entidade QUEUE_ENTITY, responsável pelo gerenciamento das filas de um módulo Estelle, possuirá as mesmas portas que a entidade VHDL que implementa este módulo. Além disso, será criada na entidade que implementa um módulo Estelle e sua correspondente QUEUE_ENTITY, uma porta adicional denominada de *próximo*. Esta porta permitirá que a entidade correspondente a um módulo Estelle solicite à sua entidade QUEUE_ENTITY a próxima mensagem .

Será criada uma entidade HIERARCHY_QUEUE_ENTITY para cada grupo de entidades que correspondem aos módulos filhos de mesma hierarquia de um módulo Estelle. Devido às restrições impostas a Estelle discutidas na seção II.2, que resultaram na definição do dialeto Estelle**, definido anteriormente, a HIERARCHY_QUEUE_ENTITY será responsável apenas pelo encaminhamento da interação à entidade responsável pelo seu tratamento.

O mecanismo de criação/destruição dinâmica de módulos, existente em Estelle, não será considerado neste trabalho devido à impossibilidade de sua implementação em hardware.

O cabeçalho de um módulo contém *pontos de interação*. As interações pertencentes a cada um dos *pontos de interação* de cada módulo corresponderão diretamente às portas de uma entidade VHDL, como ilustrado na Figura 3. O tipo a ser atribuído a um sinal em VHDL depende do papel do ponto de interação correspondente em Estelle. O sinal será do tipo in/out se o papel do ponto de interação for recepção/transmissão.

O corpo de um módulo em Estelle é composto por três partes: declarações, iniciação e transições.

As regras de mapeamento que se aplicam às declarações são as seguintes:

as declarações usuais do Pascal podem ser mapeadas diretamente em declarações VHDL. Porém, como VHDL não permite declarar variáveis na parte declarativa da arquitetura, uma solução é colocar as declarações de variáveis na parte declarativa da instrução "*process*".

as declarações de objetos específicos são mapeadas segundo os procedimentos abaixo:

como visto, um módulo Estelle mapeia em uma par entidade/arquitetura VHDL,

o comportamento interno de cada módulo Estelle é modelado por uma FSME. Como não existe tal conceito na linguagem VHDL, será necessário criar um procedimento específico para a implementação de FSMEs em VHDL. Tal estrutura será discutida mais adiante.

as interações pertencentes a pontos de interação internos também são mapeadas em sinais.

As regras de mapeamento que se aplicam à parte de iniciação de um corpo de módulo são as seguintes:

as instruções de iniciação de variáveis em Estelle e em VHDL são semelhantes, obedecendo à sintaxe geral "*var:*=valor".

a instrução de iniciação de estado, "*to* <valor>", será mapeada na instrução "ESTADO <= valor" onde o sinal ESTADO indica o estado corrente da FSME.

as instruções "*attach*" e "*connect*" em Estelle são modeladas em VHDL através do mecanismo de instanciação de componentes.

As regras de mapeamento que se aplicam à parte de *transições* de um corpo de módulo são as seguintes:

O conjunto de estruturas VHDL resultante do mapeamento de cláusulas Estelle é incluído em um "*process*". As cláusulas de Estelle "*from* state", "*when* interação-ponto.interação", "*provided* expressão" , "to estado" e "*output* interação-ponto.interação(msg)" são convertidas diretamente em VHDL para "*when* valor do estado => ... da estrutura case", "*wait until* condição", "*if* expressão *then* ..." , "*state* := valor do estado" e "interação <= mensagem". As instruções em Estelle "*procedure* nome(lista de parâmetros)" e "*function* nome(lista de parâmetros): tipo do parâmetro de retorno" são convertidas diretamente nas declarações correspondentes em VHDL.

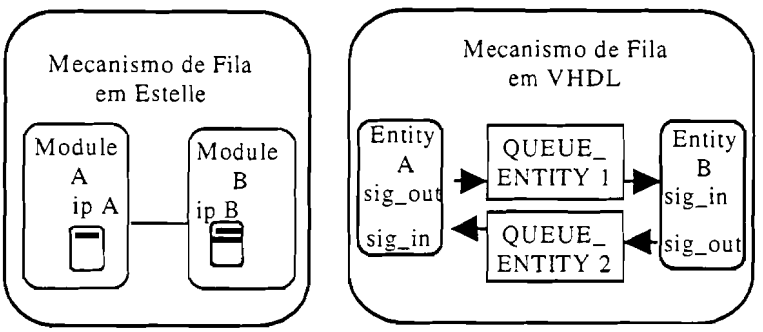

Figura 3 - Mecanismo de Filas em Estelle e em VHDL.

A cláusula "*delay*" em Estelle é mais genérica do que a cláusula similar em VHDL. A solução para este problema é restringir o uso desta cláusula em Estelle para "*delay* (T)"ou "*delay (T,T)*". A tradução desta cláusula gera em VHDL uma entidade denominada RELÓGIO_ENTITY_TYPE. Esta entidade RELÓGIO_ENTITY_TYPE possui três portas: *Count, Period e Finished.* A porta *Period* contém o valor do tempo de espera de uma mensagem. A porta *Count* informa quando a contagem deve ser iniciada e a porta *Finished informa o fim da contagem. A entidade* RELÓGIO_ENTITY_TYPE inicia a contagem quando uma entidade executando uma FSME solicita à sua entidade gerenciadora de filas uma próxima mensagem e esta última não possui nenhuma. Se o valor da contagem é maior que o especificado em *Period*, a entidade RELÓGIO_ENTITY_TYPE envia um sinal de término de contagem através do sinal *Finished* à entidade que executa a FSME. Ao receber o sinal *Finished*, esta última entidade executa a sequência de instruções associada à condição de fim de contagem.

3.2. ESTRUTURAÇÃO DE UMA *ENTIDADE VHDL*

Na seção precedente foram descritas as regras que permitem o mapeamento de instruções Estelle em VHDL. Estas regras não indicam porém, a estrutura a ser seguida em uma entidade VHDL para a implementação do conceito de máquina de estados. Para isto, as etapas necessárias são as seguintes:

Inicialmente são determinados os estados e as transições pertencentes ao corpo de cada tipo de módulo Estelle. Estes estados corresponderão aos estados da FSME de cada entidade VHDL. Esta máquina de estados constitui a arquitetura da entidade VHDL. As ações correspondentes ao disparo das transições serão executadas de acordo com as interações recebidas pela máquina de estados. Estas interações coincidem com os sinais responsáveis pela evolução da máquina de estados construída em VHDL.

Como a maioria das ferramentas de HLS utiliza como entrada descrições comportamentais do sistema a ser projetado, a arquitetura correspondente à entidade VHDL que implementa uma FSME será uma descrição comportamental.

As regras para a implementação de uma FMSE em VHDL são detalhadas a seguir:

define-se uma variável *state* para armazenar o estado corrente FSME em VHDL. O conjunto dos diferentes estados de um módulo Estelle dará origem a uma estrutura "*case*" em VHDL.

as transições correspondentes a cada estado são agrupadas por interação recebida. A instrução "*wait until* condição_interação" suspenderá ou continuará a execução até que a cláusula condição_interação seja avaliada como verdadeira. A instrução "*if ..then ...else ...*" é colocada logo após a instrução "*wait until* condição_interação" prevendo a possibilidade de mais de uma condição_interação ser avaliada como verdadeira, em um mesmo estado.

A máquina de estado resultante é inserida dentro da estrutura "*process*" do VHDL permitindo, assim, que a máquina de estados seja executada repetidamente.

3.3. O AMBIENTE DE SÍNTESE DE ALTO NÍVEL AMICAL

A fim de permitir a aplicação da metodologia proposta à síntese de alto nível de um protocolo de comunicação, utilizou-se sem perda de generalidade, a ferramenta AMICAL[10]. O AMICAL, como a maioria das ferramentas de HLS, utiliza internamente uma arquitetura alvo de forma a gerar implementações de hardware eficientes em termos de área e velocidade. Isto implica, por outro lado, que apenas um conjunto restrito de instruções VHDL pode ser mapeado de forma eficaz na arquitetura alvo empregada. Deve ser observado, que cada ambiente particular de HLS terá seu próprio subconjunto de instruções VHDL sintetizáveis.

Por princípio, a HLS se refere ao mapeamento de uma especificação comportamental em uma estrutural acima do nível de transferência de registradores [9]. Como conseqüência, as ferramentas de HLS aceitam, geralmente, como entrada apenas descrições comportamentais. No caso particular do AMICAL, o ambiente requer, além de uma especificação comportamental em VHDL, uma biblioteca de unidades funcionais.

A descrição comportamental aceita pelo AMICAL é formada por um único processo que pode conter subsistemas complexos utilizando chamadas a funções e/ou procedimentos. Entretanto, para cada procedimento ou função utilizada, a biblioteca deve conter no mínimo uma unidade funcional capaz de executar a operação correspondente. Durante os diferentes passos envolvidos na HLS, as unidades funcionais são usadas como caixas pretas que podem executar procedimentos.

Os principais conceitos utilizados na metodologia de projeto estruturado são os de sistema e componente. Um sistema é um projeto completo constituído de componentes. O conceito abstrato de componente no nível comportamental foi empregado para permitir a sua reutilização. A metodologia de projeto estruturado para VLSI consiste de três passos:

Análise do sistema e seu particionamento em subsistemas;

Síntese de cada subsistema resultante;

Abstração de cada subsistema sintetizável para ser usado como componente durante a síntese do sistema de maior nível hierárquico.

Um projeto estruturado para síntese é caracterizado por um sistema especificado por uma descrição comportamental e por um conjunto de componentes. Assim, um sistema será composto pela interconexão de componentes. Um componente será representado em VHDL por um par entidade/arquitetura capaz de executar um conjunto de operações que, no ambiente de síntese de alto nível AMICAL, são ativadas por chamadas a funções ou procedimentos. Um componente também atua como uma caixa preta que permite fazer a ligação do domínio comportamental ao domínio estrutural. As operações executadas por um componente comportamental podem ser tão simples quanto as operações pré-definidas, +, -, *, ou tão complexas quanto operações de entrada e saída com *handshaking* ou acesso a memória com funções de endereçamento complexas. Um componente pode ser o resultado de um projeto produzido em outro ambiente de projeto ou pode ser um subsistema sintetizado nas etapas anteriores do projeto.

Como foi visto, o primeiro passo da metodologia de projeto estruturado é a análise do sistema e seu particionamento em subsistemas. No caso do projeto de um protocolo, especificado inicialmente em Estelle, o processo de partição do sistema é simplificado pois os submódulos, contidos numa especificação Estelle, correspondem diretamente a subsistemas descritos por pares entidade/arquitetura VHDL. A partir deste momento, cada subsistema será mapeado em um par entidade/arquitetura VHDL segundo a estratégia de mapeamento descrita nas subseções III.1 e III.2.

Para que uma descrição VHDL seja aceita pelo AMICAL, ela deve satisfazer as seguintes restrições:

uma entidade VHDL só pode aceitar declarações de sinais de I/O.

os seguintes tipos: *physical, floating point, enumeration, arrays e records,* não são aceitos. Uma solução para *array* é criar uma unidade funcional que implemente as operações de leitura e escrita em uma área de memória de tamanho fixo cujo tipo seja escalar.

os operadores de adição e multiplicação são implementados por unidades funcionais. A instrução *"wait on"* não é aceita e em seu lugar deve-se utilizar a instrução *"wait until ..."*.

a instrução *"wait for* tempo" é ignorada. Uma solução é usar a instrução *"wait until ..."*, acrescentar um sinal de término de contagem na lista de sensibilidade da instrução *"wait ..."* e aguardar que a entidade RELÓGIO_ENTITY_TYPE envie um sinal informando o término de contagem.

o uso de estruturas *"block"* não é aceita.

cada arquitetura só pode conter uma única estrutura concorrente, a estrutura *"process"*.

para ativar a execução de uma unidade funcional são utilizadas as instruções *"function ..."* e *"procedure ..."* do VHDL.

instruções de atribuição de sinal com expressão de tempo, por exemplo, *"s<='1' after 20 ns"*, não são aceitas. Uma solução é utilizar a instrução de atribuição de sinal sem expressão de tempo e aguardar que a entidade RELÓGIO_ENTITY_TYPE envie um sinal informando o término de contagem.

A aplicação do conjunto de restrições listado acima à linguagem VHDL resultou na versão denominada VHDL**. Note que, VHDL** é o subconjunto de instruções VHDL aceitas na descrição de entrada do AMICAL e o dialeto Estelle** é o subconjunto de instruções Estelle que mapeia diretamente em instruções VHDL**.

O resultado da HLS é uma descrição estrutural de hardware cujas características elétricas e temporais serão determinadas fundamentalmente pelo algoritmo de "scheduling" usado. Assim, dependendo do estilo de descrição de entrada, o circuito sintetizado poderá satisfazer ou não as especificações originais do projeto tais como o compromisso área x custo. Este problema será abordado na próxima seção.

4. ESTILOS DE DESCRIÇÕES DE UM PROTOCOLO EM VHDL

A linguagem VHDL permite descrições de hardware comportamentais e estruturais possuindo uma grande variedade de estruturas e construções alternativas para especificar uma mesma funcionalidade do circuito. Duas descrições serão ditas semanticamente equivalentes quando reproduzirem o mesmo comportamento durante a simulação. Apesar de semanticamente equivalentes, estas descrições, quando submetidas ao processo de HLS, podem gerar estruturas de hardware diferentes do ponto de vista da área ocupada pelo circuito e velocidade de operação.

Como conseqüência, o projetista pode recorrer a uma grande variedade de estilos de descrição que diferem significativamente nos tipos de instruções e na ordem em que elas são utilizadas. Este problema é conhecido como variância sintática[9].

Os diferentes estilos de descrição de protocolos em VHDL podem ser representados por um modelo genérico de protocolo constituído por uma máquina de estados realizando operações de *handshaking* com outra máquina de estados. Esse modelo tem duas etapas: a fase de iniciação e a fase de processamento do protocolo. A fase de processamento do protocolo consiste de um *loop* que espera por sinais de entrada em cada interação, processa esse dado de entrada e transmite o sinal para a saída. Logo, a fase de processamento do protocolo é subdividida em 3 etapas: entrada, tratamento e saída de dados. Este modelo será utilizado para analisar os vários estilos de descrição e o efeito da variância sintática nos resultados da HLS. Esta análise se concentrará nos resultados do *scheduling* que é a etapa da HLS mais afetada. O algoritmo DLS (Dynamic Loop Scheduling) [13] será utilizado como base para a análise dos resultados da HLS

Tabela 1 - Tabela de Transição do Primeiro Estilo de Descrição.

Transição	*Estado*	*Operação*	*próximo*	*Condição*
1	S1	3,7	S2	state = statex
2	S1	3,15	S4	state /= statex
3	S2	9,10,11	S3	inputx = '1'
4	S2	---	S2	inputx /= '1'
5	S3	13,7	S2	state = statex
6	S3	13,15	S4	state /= statex
7	S4	17,18,19	S5	inputy = '1'
8	S4	---	S4	inputy /= '1'
9	S5	21,7	S2	state = statex
10	S5	21,15	S4	state /= statex

para os diferentes estilos de descrição [12]. O algoritmo DLS foi escolhido por ser considerado o mais eficiente dos algoritmos de *scheduling* comumente utilizados .

O primeiro estilo de descrição distribui as instruções "*wait until ...*" nos ramos da instrução "*case ...*". Na Figura 4 são mostrados a descrição VHDL e o grafo de fluxo de controle (CFG) correspondente ao primeiro estilo. Na tabela 1 é apresentada a tabela de estados resultante do *scheduling desta descrição feito pelo algoritmo DLS. É* interessante observar para referência futura que no algoritmo DLS a geração de um novo caminho é iniciada na raiz e quando uma instrução "*wait until ...*" é encontrada.

No segundo estilo de descrição, a instrução "*wait until* condição_entrada" é posta fora da instrução "*case ...*". É importante observar que neste exemplo existe somente uma possibilidade para começar um novo caminho e isto ocorre devido à existência de uma única instrução "*wait until* condição_entrada ". No terceiro estilo de descrição, a instrução "*If ... then ... else ...*" é colocada no interior de uma instrução " *process* " em lugar de uma instrução "*case ...*". *Na instrução "If ... then ... else ..." o estado* corrente da FSME é utilizado como variável mais externa. Além disso, a instrução "*wait until* condição_entrada" que no segundo estilo era colocada antes da instrução "*case ...*" é colocada agora antes da instrução "*if ... then ... else ...*" mais externa.

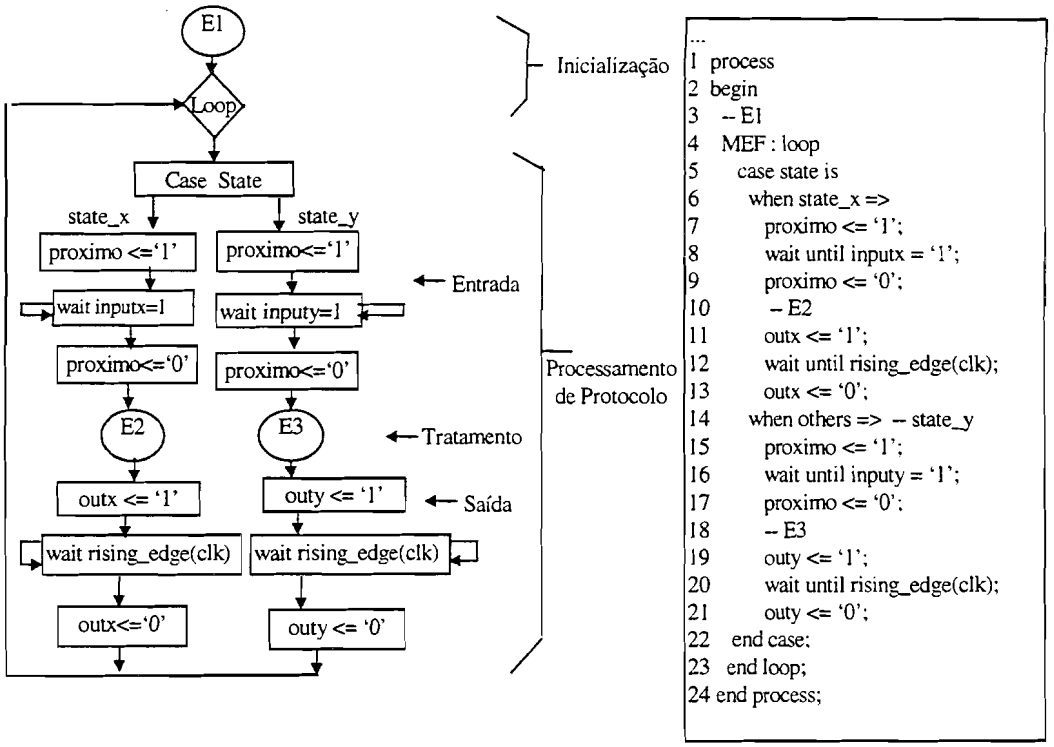

Figura 4 - CFG e a descrição VHDL referente ao Primeiro Estilo de Descrição.

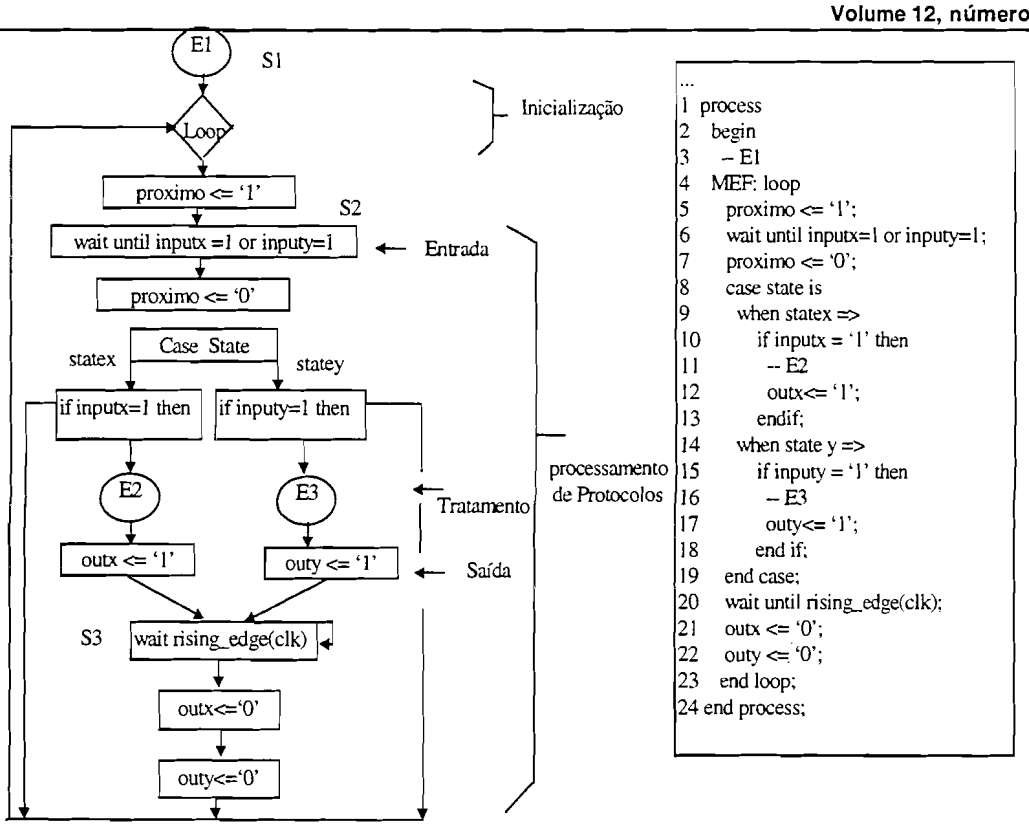

Figura 5 - CFG e a descrição VHDL referente ao Quinto Estilo de Descrição.

O quarto estilo emprega também dentro da instrução " *process* " a instrução *"If ... then ... else ..."* mas com a entrada *(input)* sendo utilizada como condição de teste mais externa. O quinto estilo de descrição é uma variação do segundo com uma única diferença: os diferentes sinais de controle de saída recebem o valor zero imediatamente após da instrução *"case ..."*. A Figura 5 apresenta a descrição VHDL e o grafo de fluxo de controle (CFG) resultante do quinto estilo. A tabela 2 apresenta a tabela de estados resultante do *scheduling* desta descrição obtida com o uso do algoritmo DLS.

A título de exemplo, serão apresentados e discutidos, na próxima seção, os resultados gerados pelo algoritmo DLS para os diferentes estilos de descrição da FSME correspondente ao módulo SINAL pertencente a um caso particular de protocolo, o protocolo ABRACADABRA_HS definido em Pirmez [14].

5. O EFEITO DOS ESTILOS DE DESCRIÇÃO NOS RESULTADOS DA SÍNTESE DE PROTOCOLOS

Os diferentes estilos que podem ser utilizados para descrever um protocolo em VHDL são discutidos com a ajuda de um exemplo, o protocolo ABRACADABRA_HS. O ABRACADABRA-HS é um protocolo de alto desempenho baseado no protocolo de referência ABRACADABRA da ISO. Trata-se de um protocolo orientado a conexão que permite duas conexões bidirecionais simultâneas entre duas entidades A e B. Uma das conexões é utilizada para a troca de mensagens de sinalização o que inclui, as mensagens de estabelecimento e término de conexão e as mensagens de controle. A outra conexão é utilizada somente para a troca de dados. As mensagens de controle são trocadas

Tabela 2 - Tabela de Transição do Quinto Estilo de Descrição.

Transição	*Estado*	*Operação*	*Próximo*	*Condição*
1	S1	3,5	S1	---
2	S2	7,11,12	S2	state = statex and inputx =1
3	S2	7	S2	state = statex and inputy =1
4	S2	7,16,17	S2	(state /= statex and inputy =1
5	S2	7	S2	state /= statex and inputx =1
6	S2	---	S2	(inputx /=1) or (inputy /= 1)
7	S3	21,22,5	S2	---

Tabela 3 -Resultados do algoritmo DLS.

Estilo	*Caminhos*	*Estados*	*Operações*	*Loops*	*Wait*	*Ifs*	*Case*	*Linhas*	*Área*
Primeiro	614	88	400	12	54	34	1	653	206.183
Segundo	138	70	406	12	45	42	1	622	46.473
Terceiro	136	69	413	12	44	51	0	625	45.904
Quarto	137	69	451	12	44	52	1	616	46.237
Quinto	104	36	325	12	11	42	1	557	35.641

periodicamente pelo receptor e transmissor para a atualização das variáveis de estado e para a confirmação de mensagens.

Os resultados da Tabela 3 foram obtidos usando o algoritmo DLS para efetuar o scheduling dos estilos de descrição propostos neste trabalho. Os parâmetros de comparação utilizados foram o número de caminhos, o número de estados gerados por cada descrição, o número de operações, o número de *Loops*, o número de *waits*, o número de *ifs*, o número de *cases*, o número de linhas de código da descrição de entrada em VHDL e a área estimada do controlador em número de transistores.

As cinco descrições utilizam estruturas de controle complexas tais como, instruções de "*wait* ..." com várias condições, chamadas de rotinas, *loops* e ninhos de *ifs*. Deve-se notar na Tabela 3 que as grandezas relevantes no que diz respeito à área do circuito sintetizado são : números de caminhos, números de estados e números de oprações resultantes do processo de scheduling. Neste caso, o quinto estilo de descrição apresenta uma nítida vantagem em relação aos demais. Isto se deve ao número reduzido de instruções "*wait until* condição_entrada" e "*wait until* rising_edge (clk)" que este estilo contém. Devido à estrutura do algoritmo DLS, que gera um novo caminho cada vez que encontra uma instrução "wait ..." ou uma dependência de dados, o número de caminhos gerados pelo algoritmo para o Quinto estilo é menor que o número de caminhos gerados para os demais estilos.

Pode ser observado também que devido à redução significativa do número de instruções "*wait until* condição_entrada" no Segundo, Terceiro e Quarto estilos, estas descrições fornecem resultados equivalentes.

Conclui-se que o principal responsável pela redução da área dos controladores gerados pelo algoritmo DLS é a localização das instruções "*wait* ..." na descrição comportamental da FSME. Em todos os estilos, exceto o primeiro, é usada uma instrução "*wait* ..." testando múltiplas condições. Ao contrário, o primeiro estilo contém várias instruções "*wait* ...", cada uma testando uma única condição de entrada. Consequentemente, o número de caminhos gerados pelo algoritmo DLS para este primeiro estilo aumenta consideravelmente, o que resulta numa área aproximadamente quatro vezes maior.

Tendo em visto a discurssão precedente, o procedimento a ser adotado para a descrição de FSMEs em VHDL, de acordo com o Quinto estilo, é o seguinte:

Uma FSME deve ser descrita por um processo em VHDL composto por dois *loops*: um externo que modela a reinicialização do processo e um interno que modela a FSME do protocolo.

A instrução "*wait until* condição_entrada " é posta no início do *loop* mais interno onde os sinais de controle são testados. O uso de múltiplas condições de entrada em uma única instrução "*wait until* condição_entrada" permite uma redução significativa na complexidade da tabela de transição, através da redução do número de caminhos gerados.

Sinais de exceções (reset, situações de erros) são criados e acrescentados na lista de sensibilidade da instrução "*wait* ...", para evitar duplicações no código. Os sinais de exceções são verificados imediatamente após a instrução "*wait*".

A FSME é construída utilizando a instrução "*case* ..." ou a construção "*if ... then ... elsif ... endif*". Em ambas as soluções, a escolha entre testar inicialmente o estado ou as condições de entrada depende do processamento ser mais dependente do estado ou das condições de entrada.

Após o uso da instrução "*case* ..." ou da construção "*if ... then ... elsif ... endif* ", os diferentes sinais de controle recebem o valor zero.

6. CONCLUSÃO

Este trabalho apresenta uma proposta de mapeamento de especificações Estelle de protocolos de comunicação em descrições VHDL adequadas às ferramentas de HLS.

A metodologia proposta foi testada com o auxílio de uma ferramenta específica de HLS, o ambiente AMICAL, sem que isto implicasse em maiores restrições à metodologia de mapeamento. De fato, as conclusões obtidas se aplicam a outros ambientes de HLS e se referem as precauções que devem ser tomadas na descrição de FSMEs em VHDL. Como foi observado ao longo do trabalho, a localização das instrucões do tipo "*wait* ..." no interior da descrição de uma FSME em VHDL, tem um efeito considerável no

resultado da etapa de *scheduling*, que é fundamental em todo o processo de HLS.

Como consequência são sugeridos diferentes estilos de descrição de protocolos em VHDL. Estes estilos se diferenciam exatamente pela estruturação dos processos VHDL que descrevem as FSMEs que modelam um protocolo de comunicação. Esta estruturação é feita segundo as regras sugeridas neste trabalho.

Como perspectivas de trabalhos futuros podemos citar a formulação e o desenvolvimento de metodologias para a solução dos problemas relacionados à implementação de protocolos de comunicação sob a pespectiva do projeto integrado (co-design) *hardware-software*. Os principais problemas envolvidos nesta área referem-se ao estabelecimento de critérios para o particionamento da especificação/implementação de protocolos parte em software e parte em hardware e a obtenção de um modelo para a avaliação das diferentes alternativas de partição hardware/software.

REFERÊNCIAS

[1] Chanson et al, *On Tools Supporting the Use of Formal Description Techniques in Protocol Development*. Computer Networks and ISDN Systems, v.25, p.723-39, North-Holland, 1993.

[2] Budkowski, S. and Dembinski, P. *An Introduction to Estelle : A Specification Language for Distributed Systems*. Computer Networks and ISDN Systems v.14, n.1, p. 3-23, North-Holland, 1987.

[3] Ashenden, P. J., *The VHDL Cookbook*. Dept. Computer Science, Univ. of Adelaide, 1990.

[4] Kloos, C. D. et all, *VHDL Generation from a Timed Extension of the Formal Description Technique LOTOS within the FORMAT project*, Microprocessing and Microprogramming, v. 38, p. 589-596, North-Holland, 1993.

[5] Pirmez L., Carneiro F., Pedroza A., Mesquita C. *A Methodology to Developed Integrated Circuits from Estelle Specifications*. In: Proceedings of the 38th Midwest Symposium on Circuits and Systems, Rio de Janeiro, Brasil, p. 19-22, Agosto, 1995.

[6] Pirmez L., Pedroza A.,. Mesquita A, Jerraya A., *Mapeamento de Estelle em VHDL e Estilos de Descrição de protocolos de Comunicação para Síntese de Alto Nível*, 14° Simpósio Brasileiro de Redes de Computadores (SBRC), Fortaleza, Ceará,. p 253-272, Maio, 1996.

[7] Jerraya, A. and O'Brien, K. *SOLAR: an intermediate format for system-level modeling and synthesis*. Chapter 7 in Codesign computer aided software/hardware engineering , p. 145-175, IEEE Press, 1994.

[8] Vahid, F. et al. *SpecCharts: A language for System Level Design*. Proc. VHDL'91, IFIP, p.145-155, Abril 1991.

[9] Gajski D. et al., *Introduction to High_Level Synthesis*, IEEE Design and Test of Computers, p. 44-54, 1994.

[10] Park I., O'Brien K., Jerraya A.A., *AMICAL: Architectural Synthesis Based on VHDL*, in Synthesis for Control Dominated Circuits, p. 219-234, North-Holland, 1993.

[11] Courtiat, J., *Estelle* A Powerful Dialect of Estelle for OSI Protocol Description*, In: Proceeding of the VIII International Symposium on Protocol Specification, Testing and Verification, p. 171-186, France, North-Holland, 1988.

[12] Pirmez L. et al, " Analysis of different protocol descriptions styles in VHDL to High-Level Synthesis", In: Proceedings of the EURODAC'96, Geneva, Switzerland, p. 490-495, Setembro, 1996.

[13] Rahmouni M., O'Brien K. and Jerraya A. A., A Loop-based Scheduling Algorithm For Hardware Description Languages, Parallel Processing Letters, v. 4, n. 3, p. 351-364, World Scientific Publishers, 1994.

[14] Pirmez L., Pedroza A., Mesquita C. *A Methodology to the Implementation of Distributed System in Hardware from a Formal Description*, In: Proceedings of the Fifteenth IFIP WG. 6.1 International Symposium on Protocol Specification, Testing and Verification, Warsow, Poland, p. 419-434, Chapman&Hall, Junho, 1995.

[15].Pedroza, A. et al. *Um sistema de Auxílio ao Projeto de Protocolos de Comunicação para Redes de Computadores*, Seminário Franco-Brasileiro em Sistemas Distribuídos, Florianópolis, Santa Catarina, p. 80-87, Setembro, 1989

Antonio Mesquita é Engenheiro Eletricista pela PUC-MG, Mestre pela COPPE-UFRJ, e Docteur d'État pela Université Paul Sabatier de Toulouse, França. É Professor Associado da COPPE-UFRJ. Suas áreas de interesse são teoria de circuitos e sistemas, projeto e circuitos em VLSI, e processamento de sinais.

Aloysio Pedroza é Engenheiro Eletricista pela UFRJ, Mestre pela COPPE-UFRJ, e Doutor pela Université Paul Sabatier de Toulouse, França. É Professor Associado da UFRJ. Suas áreas de interesse são especificação formal, e verificação e implementação de protocolos.

Luci Pirmez é Engenheira Eletricista pela UFRJ, Mestre pela COPPE-UFRJ, e Doutora pela COPPE-UFRJ. Seus interesses de pesquisas são em especificação formal, e verificação, implementação e síntese de protocolos.