UMA METODOLOGIA PARA A IMPLEMENTAÇÃO DE SISTEMAS DISTRIBUÍDOS EM HARDWARE A PARTIR DE UMA DESCRIÇÃO FORMAL

Luci Pirmez (1) (2)
Aloysio de Castro P. Pedroza (1) (3)
Antônio Carneiro de Mesquita (1)

(1) Coppe/Ufrj - Programa de Engenharia Elétrica
Caixa Postal 68504 - 21495, Rio de Janeiro, RJ, Brasil
E-mail: luci@barra.nce.ufrj.br
(2) Núcleo de Computação Eletrônica - UFRJ
(3) EE/Ufrj-Departamento de Eletrônica

Resumo

Esse trabalho apresenta uma metodologia capaz de transformar de forma eficiente e fácil especificações formais em Estelle para descrições em VHDL (Very High Speed Integrated Circuit (VHSIC) Hardware Description Language), para que posteriormente sejam gerados os circuitos integrados correspondentes. A metodologia proposta se baseia na compilação de silício, isto é, no uso de ferramentas que permitem gerar uma máscara de fabricação de circuito integrado a partir de uma especificação de hardware abstrata em VHDL. Esse trabalho também define restrições para a obtenção de descrições eficientes em Estelle e em VHDL. Finalmente é apresentado um exemplo em que a metodologia proposta é aplicada.

Abstract

This paper shows a methodology that efficiently transforms Estelle formal specifications into a Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) description, so that later on integrated circuits can be easily created. The methodology is based on the silicon compilation technique through the use of tools that allows the creation of a manufacture mask of integrated circuit from an abstract hardware specification in VHDL. In addition, it also defines restrictions to get efficient descriptions in Estelle language and VHDL. Finally an example is shown in which the presented methodology is used.

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 vem permitindo o desenvolvimento de redes cada vez mais velozes e confiáveis. O gargalo para a obtenção de um bom desempenho não é mais o meio de transmissão mas o processamento necessário para executar os protocolos de comunicação nas estações de trabalho e nos servidores destas redes. Por essa razão, a principal fonte de erros em tais sistemas é a perda de pacotes devido à falta de área para armazenamento.

Essas mudanças na tecnologia disponível e o surgimento de aplicações multimídia demandam modificações no projeto de protocolos. Por um lado, os protocolos precisam ser leves o suficiente para permitir operações de alta velocidade e, por outro lado, algumas funções adicionais são necessárias para recuperar erros e perdas de forma eficiente, controlar o congestionamento e suportar aplicações multimídia. Esses avanços levaram ao

desenvolvimento de vários protocolos de alta velocidade como, por exemplo, os protocolos DATAKIT [3], VMTP [1], XTP [7] e o protocolo proposto por Netravali [4]. Nos trabalhos de Doshi e Johri [5] e de Doeringer et all [2] são descritas as técnicas de protocolos mais adequadas a um sistema de comunicação de alto desempenho.

A obtenção de subsistemas de comunicação de alto desempenho não se limita somente à otimização de software e ao emprego do paralelismo para a construção de protocolos (máquinas de estados paralelas) mas, também, na implementação desses protocolos em hardware VLSI. Uma implementação de uma ou mais camadas de protocolos em circuitos VLSI é denominada controlador de protocolos [22]. Um exemplo clássico de controlador é o "Protocol Engine" que implementa um protocolo de alta velocidade, o XTP. Outro exemplo de controlador é descrito por Braun, Schiller e Zitterbart [23] que apresenta a implementação de um protocolo de transporte, o Patroclos, usando componentes VLSI atribuídos somente às funções que normalmente provocam "gargalos" tais como aquelas responsáveis pela temporização, pelo gerenciamento de memória ou para o suporte de retransmissão.

Entretanto, os avanços nas tecnologias em VLSI produziram uma crise no setor de projetos. O tempo de projeto passou a ser da mesma ordem de grandeza do tempo de vida de um circuito. Com o objetivo de diminuir o tempo necessário ao projeto, os circuitos integrados são atualmente projetados de maneira (semi) automática a partir de especificações em alto nível de um dado sistema. Uma especificação de alto nível inclui qualquer especificação de hardware que não seja uma descrição exata da máscara de fabricação que o implemente. Esta forma (semi) automática de projetar circuitos integrados levou ao desenvolvimento de técnicas que permitem obter um layout de hardware para integração a partir de uma especificação de alto nível (a compilação de silício). Para reduzir ainda mais o tempo de projeto de redes de computadores, podemos projetar os circuitos integrados em VLSI a partir de especificações de sistemas utilizando níveis os mais abstratos possíveis, isto é, a partir de especificações que utilizem Técnicas de Especificação Formal (FDTs) como, por exemplo, a linguagem (Extended Transition Language) Estelle [8,10, 11] da ISO.

Neste artigo é definida uma metodologia capaz de permitir uma implementação (semi) automática de circuitos integrados para sistemas de comunicação de alta velocidade. Esta implementação é obtida a partir de especificações de protocolos eficientes feitas em Estelle usando técnicas de compilação de silício [19,21].

Com o objetivo de aplicar a metodologia proposta, foi definido um protocolo de alta velocidade baseado no protocolo de referência ABRACADABRA criado pela ISO que denominamos de ABRACADABRA_HS. Nosso protocolo reúne as características de maior parte dos protocolos de alto desempenho como o XTP e o Datakit.

A seção 2 deste artigo apresenta os conceitos introdutórios relativos à especificação formal de protocolos [14], as características principais da linguagem Estelle [8], os conceitos de compilação de silício [19] e as características principais da linguagem VHDL [18,20]. Na seção 3 é apresentada uma metodologia para o mapeamento de especificações em Estelle para VHDL. Na seção 4 é introduzido um exemplo de protocolo de alto desempenho, o protocolo ABRACADABRA_HS. Por fim, a seção 5 destacará os principais resultados já obtidos, as pespectivas e considerações finais sobre o projeto de pesquisa a ser desenvolvido.

2 - Especificação Formal e Síntese de Alto Nível de Protocolos

Os protocolos de comunicação são um conjunto de regras e procedimentos que permitem a interação entre entidades comunicantes e são, consequentemente, cruciais para o funcionamento das redes de computadores e sistemas distribuídos. A crescente utilização de sistemas distribuídos e a enorme aceitação do modelo de referência e dos protocolos padronizados pela ISO motivaram, na década passada, diversas pesquisas na área de engenharia de protocolos. Desde o início, notou-se a importância dos métodos formais para engenharia de protocolos e, em particular, na especificação de protocolos de comunicação. O termo Técnicas de Descrição Formal (FDT) foi criado no final da década de 70 para referenciar as técnicas usadas para descrever precisamente os protocolos e os serviços de comunicação de dados. Paralelamente aos esforços acadêmicos em pesquisas de FDTs, a ISO e o CCITT contribuíram com o desenvolvimento de FDTs padronizadas no fim da década de 80. A linguagem Estelle e a linguagem LOTOS foram definidas pela ISO, e a linguagem SDL pelo CCITT.

Este trabalho apresenta uma metodologia que integra um C.A.D. Estelle com um C.A.D. de microeletrônica. O C.A.D. de microeletrônica permite a síntese de alto nível a partir de uma especificação VHDL. O mapeamento de Estelle para VHDL é facilitado pois a linguagem VHDL possui estruturas similares às da linguagem Estelle.

2.1 - A linguagem Estelle

A linguagem Estelle [8,9,10] é uma Técnica de Descrição Formal desenvolvida pela ISO para a especificação de sistemas distribuídos, em particular dos serviços e dos protocolos de comunicação padronizados pela ISO.

Em Estelle, uma especificação (specification) é composta por um conjunto de *módulos* organizados de forma hierárquica. Um módulo pode ser refinado em submódulos, definindo uma estrutura hierárquica entre eles. Um módulo é composto por dois componentes: um *cabeçalho* (module) e um *corpo* (body). No *cabeçalho* são definidos seus pontos de interação, variáveis exportadas, bem como seu atributo de classe. No componente *corpo* é definido o comportamento do módulo. A especificação do comportamento de um módulo Estelle é baseado no modelo de máquina de estados finita estendida. Esta máquina de estados é um sistema composto por um conjunto de transições cujo comportamento é descrito através de instruções em Pascal, com algumas restrições e extensões. A evolução do módulo corresponde ao disparo das transições desta máquina de estados. As transições são consideradas atômicas. A declaração de um módulo define um tipo dentro da especificação. Variáveis de um tipo módulo especificam instâncias de um mesmo módulo.

A linguagem Estelle possui algumas características, como o sistema de filas e a existência de diferentes tipos de paralelismo entre instâncias de módulos, que tornam difícil a implementação de uma especificação Estelle numa linguagem como o "C" e o "Pascal".

Courtiat em [9] introduziu uma versão simplificada e mais eficiente de Estelle, o Estelle*. Essa versão permite obter especificações de protocolo com alto nível de abstração, utilizando uma semântica de paralelismo entre instâncias de módulos consistentes com a semântica definida nos modelos frequentementee utilizados para a modelagem e validação de protocolos

de comunicação (máquina de estados comunicantes, redes de Petri). Estelle* será utilizada como um dos pontos de partida deste trabalho, e algumas restrições adicionais à linguagem Estelle para melhorar o mapeamento para VHDL são encontradas na seção 3.

2.2 - A Síntese de Alto nível

Os circuitos integrados VLSI são projetados de maneira (semi) automática a partir de especificações dos sistemas em alto nível com o objetivo de diminuir o tempo necessário do projeto. Esta linha levou ao desenvolvimento da técnica de compilação de silício, isto é, ao desenvolvimento de técnicas que permitem obter um layout de hardware para integração a partir de uma especificação de alto nível.

A especificação de um circuito integrado pode ser classificada quanto ao seu domínio e quanto ao seu nível. As diversas representações existentes podem ser discutidas de forma mais clara através do auxílio do diagrama-Y de Gajski-Kuher [19], mostrada na figura 2.1. Por convenção, cada eixo corresponde a um domínio e quanto mais longe do centro do diagrama, mais abstrato é o nível de especificação em cada eixo.

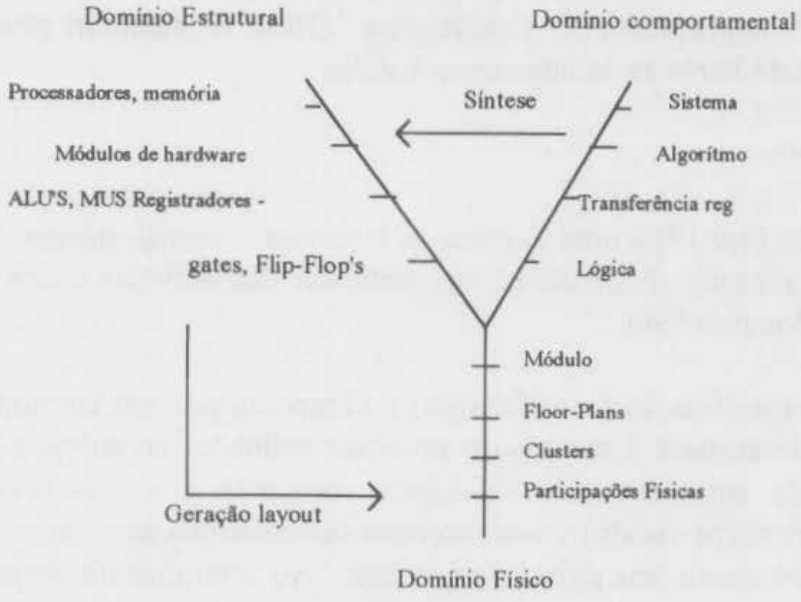

Figura 2.1 O Diagrama-y

Existem três domínios de especificação de hardware: *o comportamental, o estrutural e o físico*. No *domínio comportamental* estamos interessados no que o projeto faz e não em como ele é construído. Neste domínio, o projeto corresponde a uma ou mais caixas pretas onde são considerados apenas a sua interface com o meio externo e um conjunto de funções. Estas funções descrevem o comportamento de cada saída em termos das entradas no tempo. O *domínio estrutural* é a ponte entre o domínio comportamental e o domínio físico. Neste domínio, um sistema é especificado através de seus componentes e conexões, tais como os processadores, as memórias, transistores, flip flop e etc. No *domínio físico* é descrito à alocação de módulos de hardware na área destinada a fabricação do circuito.

Quanto à representação em nível, existem vários níveis, tais como o *arquitetural, o de Transferência de Registros (RT), o nível lógico e o nível de dispositivo.* Um nível é caracterizado pelos elementos que ele utiliza, como podemos observar na figura 2.1.

A técnica de compilação de silício de uma especificação de hardware pode ser dividida em duas etapas distintas: a síntese e a geração de layout. A síntese corresponde à tradução de uma especificação *comportamental* em uma especificação *estrutural*. O processo de síntese não é único, pois existem várias estruturas de hardware que podem corresponder a uma mesma especificação comportamental. Este trabalho utiliza o AMICAL[25] para fazer a síntese. O AMICAL permite a síntese de alto nível a partir de uma especificação em VHDL. VHDL é uma linguagem que tem-se tornado padrão para especificação de hardware nos diversos domínios do gráfico de Gajski-Kuher. Desse modo, a síntese de uma especificação em Estelle é obtida após o mapeamento de uma especificação em Estelle em descrições comportamentais do VHDL. A geração de layout é responsável pela tradução da estrutura de hardware em informação física correspondente à alocação das estruturas de hardware sobre a área do chip e a geometria dos componentes a serem fabricados e suas interconexões.

2.3 - A Linguagem VHDL

VHDL [18, 20] é uma linguagem padrão desenvolvida pelo Departamento de Defesa Americano (DoD) para descrever sistemas eletrônicos digitais. VHDL suporta o projeto, a descrição e a simulação de estruturas de hardware em diferentes níveis de abstração, do arquitetural ao lógico. Esta linguagem foi projetada para ser independente de qualquer tecnologia específica, ambiente de projeto, ou métodos de projeto, e, consequentemente, possibilitar a integração de qualquer combinação de ambiente, tecnologia, e metodologia.

VHDL é uma linguagem multidomínios e multiníveis que permite descrições estruturais e comportamentais. A descrição estrutural permite descrever as estruturas de um projeto, sua decomposição em sub-projetos e a interligação entre eles. A descrição comportamental permite a especificação de funções de um projeto utilizando estruturas familiares da linguagem de programação. Além disso, a linguagem VHDL permite que um projeto seja simulado antes de ser fabricado, de forma que os projetistas possam comparar alternativas rapidamente e testar a correção sem o atraso e o custo de um protótipo de hardware.

```
entity exemplo is
    port   (ConReq, ConRsp : in bit; ConInd ,ConCnf : out bit);
    end    exemplo;
architecture exemplo_body of exemplo is
    component conexão                          -- declaração
        port (Conreq: in bit; ConInd : out bit);
    end component;
    outras declarações
begin
    CON : conexão                              -- Instânciação
        port map (ConReq => CR, ConInd =>CI);
    outras instruções concorrentes;
end    exemplo_body;
```

Figura 2.2 - Exemplo do segmento interface e do segmento corpo de uma entidade.

Um sistema digital é normalmente projetado como um conjunto hierárquico de módulos. Cada módulo possui um conjunto de portas que constituem sua interface com o exterior. Em VHDL, uma entidade representa um módulo. A entidade pode ser usada como um componente

em um projeto ou pode ser o módulo topo do projeto, isto é, o módulo cuja hierarquia é a de nível mais elevado.

Uma entidade, como ilustrada na figura 2.2, é composta por dois segmentos, uma *interface* (Entity) e um conjunto de um ou mais *corpos* (architecture). A *interface* define as características observáveis externamente, isto é, as portas de entrada/saída. Um *corpo* define a implementação de uma entidade, o que não é observável externamente.

No segmento *interface* da estrutura entidade, as declarações de portas (port) de entrada/saída, cuja classe é sinal, possibilitam que uma entidade possa trocar informações com outras entidades. No segmento *corpo* da estrutura entidade é encontrado a implementação de uma entidade. O segmento *corpo*, por sua vez, é subdividido em duas partes: a parte declarativa e a parte referente às instruções concorrentes (concurrent_statement). Sinais e componentes podem ser declarados em um *corpo*, permitindo então, a construção de uma descrição estrutural em termos de instâncias de componentes, como ilustrado na figura 2.2. Os sinais são utilizados para conectar os submódulos de um projeto. A declaração *component* permite uma entidade utilizar outras entidades descritas separadamente e armazenadas em bibliotecas. Para permitir tal descrição estrutural, o *corpo* de uma entidade deve declarar um componente e depois instânciá-lo dentro da parte referente a instruções. Mais tarde, uma especificação de configuração será usada para especificar a ligação do componente com a biblioteca da entidade em questão.

2.4 - Restrições à linguagem VHDL para a obtenção de circuitos VLSI eficientes

O processo de concepção de um projeto tem o papel principal na síntese de alto nível, desde que a maneira como um projeto é modelado e descrito tem um impacto direto na sua implementação final, ambos em termos de custo e desempenho. Dessa forma, a obtenção de uma síntese de alto nível *eficiente* dependerá de um bom "casamento" entre o modelo semântico da linguagem e o modelo arquitetural de hardware utilizado. Entretanto, caso a linguagem seja desenvolvida para atender vários projetos e aplicações, o modelo semântico da linguagem pode ser bem diferente do modelo arquitetural gerado pelas ferramentas de síntese de alto nível. Além disso, ambas, a qualidade do projeto de síntese e a complexidade das ferramentas de síntese usadas são influenciadas pelo estilo da linguagem utilizado para descrever o comportamento de entrada.

Esta discrepância entre os modelos semânticos da linguagem e os modelos arquiteturais é particularmente verdadeira na linguagem VHDL. A linguagem VHDL foi projetada para atender descrições de projeto no domínio comportamental e no estrutural. Consequentemente, essa linguagem possui uma variedade de estruturas que permite a descrição do mesmo comportamento de maneiras diferentes. Após a síntese, algumas dessas descrições geram estruturas de hardware desnecessárias. Essas estruturas ineficientes precisam ser eliminadas por uma ferramenta de síntese de alto nível para possibilitar a geração de um projeto razoável. Infelizmente, existem poucas ferramentas que facilitem a tarefa de modelagem e de descrição do projeto em síntese de alto nível.

Um dos objetivos deste trabalho é propor normas para um bom "casamento" entre o modelo da linguagem utilizado para a descrição de um protocolo de comunicação e a arquitetura de hardware gerada após a síntese e, que implementa o protocolo considerado. O

modelo FSMD (Finite State Machine with Datapath) é geral e pode ser usado de forma eficiente no projeto de protocolo de comunicação. Estruturalmente o modelo FSMD corresponde a um projeto no nível de transferência de registros (RT). Consequentemente, a descrição comportamental de um projeto de RT é baseada em uma máquina de estados, onde cada estado especifica a condição que será testada na unidade de controle, a operação que será executada no dathpath e o próximo estado do projeto. Infelizmente, a linguagem VHDL não possui estruturas embutidas que implemente o conceito de estados. Por esse motivo, pode-se modelar o comportamento de uma máquina de estados de diferentes formas e nem todos resultarão em projetos eficientes de síntese.

3 - Metodologia para as síntese de sistemas distribuídos a partir de uma especificação formal

Esta seção propõe uma metodologia capaz de transformar de forma eficiente e fácil as especificações de um protocolo em Estelle em descrições em VHDL. A partir destas descrições VHDL acopladas a outras ferramentas de síntese pode-se gerar um circuito integrado que implemente o protocolo considerado. Com esta metodologia, o tempo necessário ao projeto pode ser diminuído. Além disso, o processo de síntese deve transparente para os usuários de Estelle.

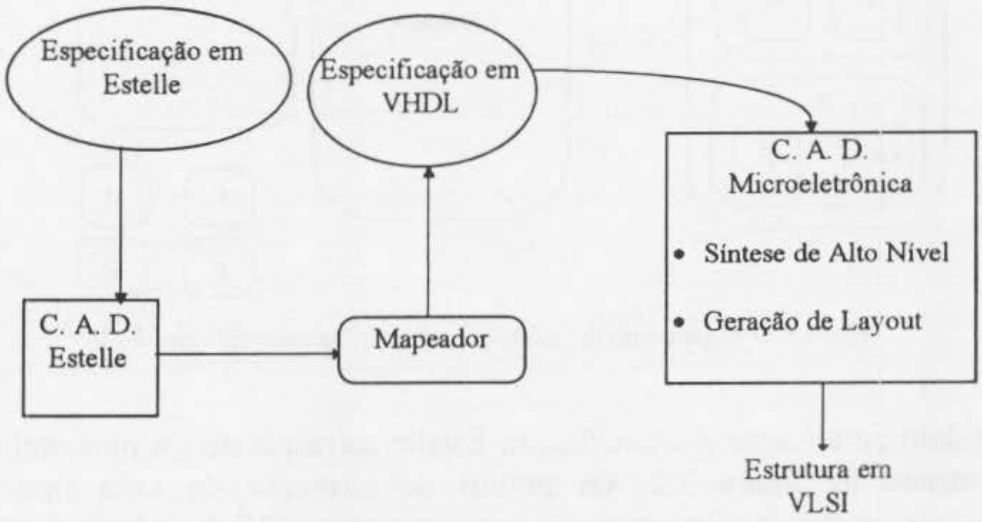

Figura 3 . 1 - Metodologia para a obtenção de circuitos VLSI a partir de uma especificação Estelle

A metodologia para a obtenção de circuitos VLSI a partir de uma especificação Estelle é mostrada na figura 3.1. A metodologia parte de um ambiente próprio para o desenvolvimento de protocolos em Estelle, um C.A.D. Estelle. Após a geração de uma especificação em Estelle corretamente simulada, esta mesma especificação é mapeada diretamente na linguagem VHDL. Esse mapeamento é feito pelo MAPEADOR. Posteriormente, a saída do MAPEADOR será utilizada como entrada para um C.A.D de microeletrônica. O C.A.D. de microeletrônica é um ambiente adequado ao projeto de circuitos integrados utilizando a síntese de alto nível. Podemos observar a independência entre os dois ambientes, o C.A.D. Estelle e o C.A.D. de microeletrônica. O MAPEADOR deve ser uma ponte entre estes ambientes.

3.1- Mapeamento de uma especificação escrita na linguagem Estelle completa para VHDL

O MAPEADOR deve fazer a ligação entre o C.A.D Estelle e o C.A.D. de microeletrônica. A entrada do MAPEADOR é um protocolo especificado em Estelle e a saída é um protocolo descrito em VHDL. O MAPEADOR é responsável por encontrar estruturas em VHDL que correspondam a estruturas em Estelle. Mas, nem sempre é possível o mapeamento direto entre estruturas preexistentes nas duas linguagens. Adaptações neste caso nem sempre geram textos VHDL compactos e eficientes.

Podemos apontar alguns aspectos não encontrados na linguagem VHDL mas presentes em Estelle como, entre outros: o sistema de filas, variáveis partilhadas, a abrangência da cláusula de prioridade. Nessa seção é apresentada uma possível solução para um mapeamento de Estelle para VHDL.

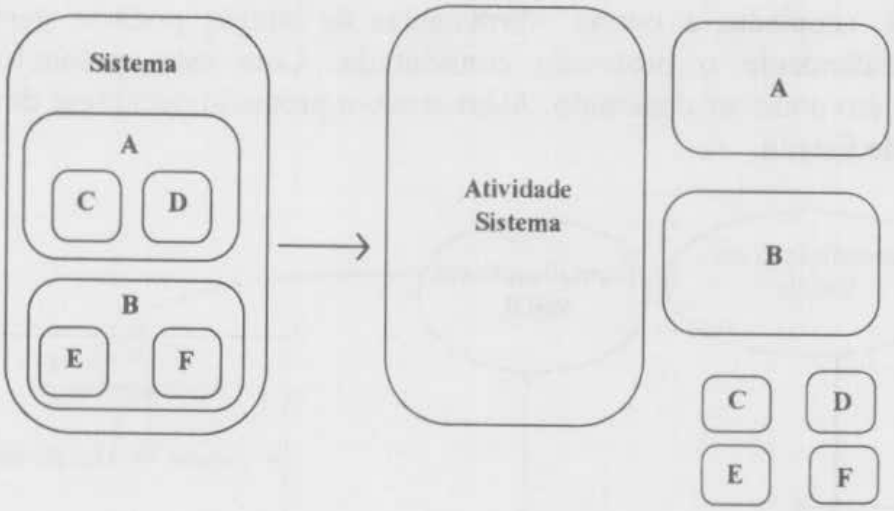

Figura 3.2 - Mapeamento dos módulos em Estelle para entidades em VHDL

Cada módulo pertencente à especificação Estelle corresponderá a uma entidade VHDL, conforme mostrado na figura 3.2. Os pontos de interação de cada tipo de módulo corresponderão às portas da interface de uma entidade VHDL. Dependendo do papel assumido pelo ponto de interação, as portas em VHDL serão do tipo de entrada (**in**), do tipo saída (**out**) ou entrada e saída (**inout**). As declarações dos tipos de dados e dos tipos de objetos (constantes e variáveis) em VHDL são mapeadas nas declarações correspondentes em Estelle (declarações em Pascal).

O mecanismo de filas fifo embutido na linguagem Estelle não pode ser mapeado diretamente em VHDL. O mecanismo de filas pode ser implementado por uma entidade, denominada QUEUE_ENTITY, responsável apenas pelo gerenciamento das filas de um módulo em Estelle, observe a figura 3.3. Além disso, uma outra entidade deve ser criada, denominada HIERARCHY_QUEUE_ENTITY, para gerenciar as filas dos módulos de mesma hierarquia. Para cada canal de comunicação pertencente a um módulo Estelle, a entidade responsável pela implementação de filas desse módulo deve gerenciar duas filas fifos. A primeira guarda os tipos de interações recebidas/transmitidas e a outra armazena o conteúdo da mesma.

As variáveis exportadas são um outro aspecto encontrado em Estelle e não existente em VHDL. Caso haja uma declaracão de uma variável partilhada (exportada) no cabeçalho de um módulo, duas portas, uma de entrada e a outra de saída, devem ser criadas nas entidades em VHDL que correspondem a esse módulo e ao seu módulo pai. Em VHDL, as entidades são independentes de tal forma que a comunicação entre entidades só é permitida através de portas de entrada e saída. Além disso, cria-se uma outra entidade, que será responsável por garantir a exclusão mútua. Nessa entidade também são criados pares de portas (entrada e saída), uma para cada entidade que tem acesso a essas declarações. Todas as atribuições (consultas) na (da) variável partilhada devem ser trocadas por cláusulas de "output" (cláusulas de "when") nas entidades em questão.

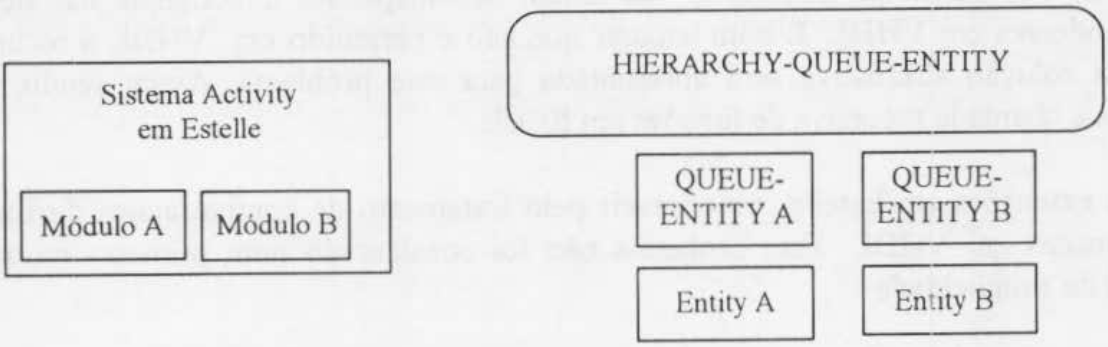

Figura 3.3 - Entidades que simulam o sistema de filas

A linguagem Estelle possui quatro tipos diferentes de módulos correspondentes às diferentes maneiras de execução: atividade sistema, atividade, processo sistema e processo. Não existe estrutura correspondente em VHDL para os tipos de paralelismo de Estelle. As instâncias de módulos do tipo processo executam segundo um paralelismo síncrono enquanto as instâncias de módulos do tipo atividade executam de forma não determinística. O mapeamento do tipo *process* ou do tipo *activity* de Estelle para VHDL não pode ser obtido de forma direta. O funcionamento de um sistema do tipo *activity*, é implementado em VHDL na entidade HIERARCHY_QUEUE_ENTITY. Esta entidade deve selecionar apenas uma instância de módulo para execução. Essa arquitetura é repetitiva ao longo dos níveis hierárquicos sucessivos de uma especificação Estelle. Um sistema do tipo "process" em Estelle não pode ser mapeado diretamente em entidades cujos corpos possuam a instrução "process" uma vez que o paralelismo em VHDL é assíncrono. A solução para um sistema do tipo "***process***" é implementar na entidade HIERARCHY_QUEUE_ENTITY de VHDL o mecanismo responsável pela seleção de um grupo de instâncias, que não possuem conflitos de prioridade entre pai e filhos, prontas para a execução. Além disso, a entidade HIERARCHY_QUEUE_ENTITY tem que garantir que a próxima seleção só ocorrerá após o término da execução de todas as instâncias selecionadas inicialmente.

A cláusula "**delay**" em Estelle é mais genérica do que a correspondente em VHDL. A solução para o mapeamento é restringir a utilização desta cláusula em Estelle e em VHDL colocar o atraso na primeira instrução do procedimento que trata a execução da transição correspondente. Em Estelle, a cláusula "**delay**" será usada somente na forma "delay(E1)" ou "delay(E1,E1)".

A utilização da cláusula "**priority**" em Estelle é mais genérica do que a cláusula similar em VHDL. Como o objetivo é obter um hardware composto por circuitos lógicos simples e

eficientes, a implementação de tal gerenciamento, que é sem dúvida pesada, resultaria em um hardware não tão eficiente. Desse modo, não é considerado o mapeamento desta cláusula neste trabalho.

As cláusulas "**from** *estado*", "**when** *ponto_interação.interação*", "**provided** *expressão*", "**to** *estado*", e "**output** *ponto_interação.interação(msg)"* de Estelle são mapeadas em VHDL respectivamente nas instruções "**when** state => .... da estrutura **case**", "**wait on** *interação* **until** *condição"*, "**if** *expressão* **then** ..." , state := *estado e* " interação <= msg".

As instruções "**procedure** name **(lista** de parâmetros formais) e "**function** name (lista de parâmetros formais)**:** tipo-de-retorno" de Estelle são mapeadas diretamente nas declarações correspondentes em VHDL. É bom lembrar que não é permitido em VHDL a recursividade. Nenhuma solução alternativa será apresentada para este problema. Assim sendo, não será permitida a chamada recursiva de funções em Estelle.

As extensões em Estelle responsáveis pelo tratamento de configurações dinâmicas, não são mapeadas em VHDL. Este problema não foi considerado num primeiro momento por questões de simplicidade.

Os módulos em Estelle são especificados de forma aninhada enquanto que em VHDL as entidades (módulos) são especificadas separadamente. A hierarquia entre módulos é obtida através de uma estrutura *component.* A estrutura *component* de VHDL é responsável pela criação das instâncias de módulos (função "init" do Estelle) e pelo estabelecimento dos links de comunicação (função "connect e attach" do Estelle). Na figura 3.4 está representado um sistema atividade formado por duas instâncias de módulo atividade (A e B). Cada instância é formada por outra instância de hierarquia imeditamente inferior (C, D, E e F). O módulo principal de Estelle denominado "specification" é mapeado em uma entidade em VHDL.

A metodologia geral proposta para o mapeamento protocolos descritos em estelle para VHDL é dividida em quatro etapas, como mostrado na figura 3.4. Inicialmente são identificados todos os tipos de módulos em Estelle e seu relacionamento hierárquico. A partir desta identificação, cada tipo de módulo é implementado como uma entidade em VHDL. A implementação de cada módulo é feita de maneira independente dos demais e a sua hierarquia é obtida através da estrutura *component*. Em seguida, criam-se entidades gerenciadoras de filas, uma para cada entidade criada e uma para cada grupo de entidades filhas.

Em seguida, é proposta uma metodologia para estruturar cada módulo em Estelle. Cada módulo é implementado de maneira independente dos demais como entidades em VHDL. A metodologia proposta é formada de uma sequência de etapas. Inicialmente são determinados os estados e as transições pertencentes ao corpo de cada tipo de módulo. Os estados de cada tipo de módulo em Estelle corresponderão aos estados em VHDL da máquina de estado. Esta máquina de estados implementa o corpo de uma entidade VHDL correspondente a esse módulo em Estelle. As transições corresponderão aos disparos de ações. Estas ações serão executadas por esta máquina de estado de acordo com as interações recebidas (papéis dos canais de comunicação), isto é, os sinais recebidos. Para cada estado, determine suas transições. Finalmente, obtenham-se as interações associadas a cada estado e transições. Estas interações coincidem com os sinais responsáveis pela evolução da máquina de estados construída em VHDL.

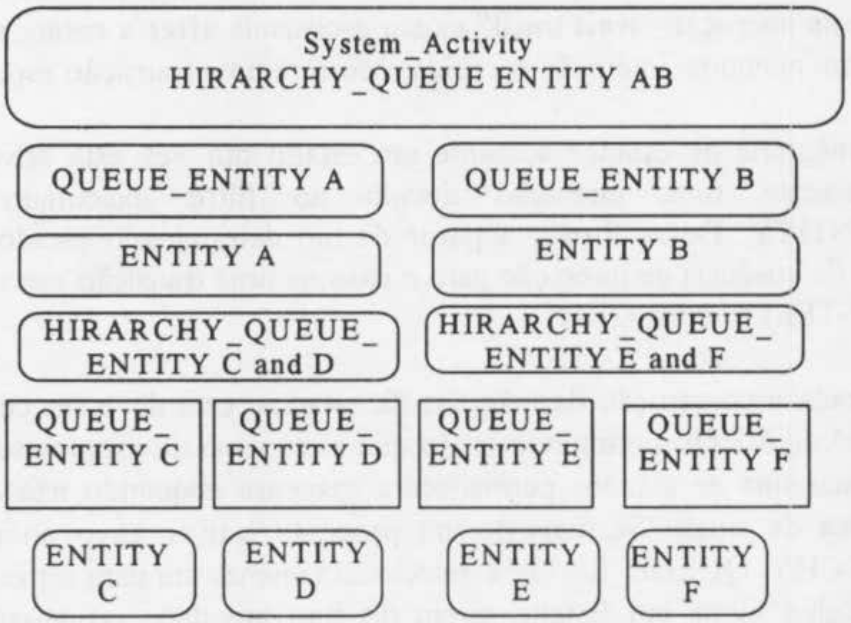

Figure 3.4 Mapeamento dos módulos em Estelle para entidades em VHDL

Neste trabalho, o corpo de uma entidade será construído apenas segundo uma descrição comportamental. Essa descrição será montada diretamente a partir da descrição da máquina de estados em Estelle. Uma variável (ESTADO) é definida com o estado atual da máquina de estados em VHDL. Neste ponto podemos escolher uma dentre duas opções: transições agrupadas a partir dos estados da máquina e transições agrupadas a partir das interações recebidas pelo módulo. Neste artigo será detalhada a primeira opção:

1 - O conjunto dos diferentes estados de um módulo dará origem a uma estrutura Case do VHDL.

2 - Para cada estado, transições associadas com este estado são agrupadas por interação recebida. A instrução "**wait until** *interações*" suspenderá ou retornará a execução devido a mudança de valores desses sinais (interações). Se mais de uma interação for recebida para um mesmo estado então, a estrutura **if** do VHDL será colocada logo após a instrução "**wait until** *interações*".

```
case ESTADO of
  when estado-1 =>
    wait until interação_1 or ... after x ns;
      if interação_1 then
        -- transições;
      else -- transições espontâneas;
      end if;
  ...
  when others -- outros procedimentos;
end case;
```

3 - As transições espontâneas não estão associadas à recepção de qualquer interação. Estas transições são executadas quando nenhuma interação é recebida em um estado.

Transições espontâneas são tratadas pela cláusula **else** da estrutura **if** obtida no ítem anterior. Se na instrução "**wait until**" existir a cláusula **after x** então, após o intervalo x de tempo com nenhuma interação recebida, ocorrerá uma transição espontânea.

4 - Numa máquina de estados somente um estado por vez está ativo. Além disso, é ativada somente uma interação devido ao filtro executado pela entidade QUEUE_ENTITY. Dessa forma, a partir de um determinado estado e da interação recebida ou da ausência de interação para o caso de uma transição espontânea, uma dada transição ESTELLE é disparada.

Após finalizada a construção da máquina de estados, esta deve ser colocada dentro da estrutura "process" do VHDL permitindo então que a máquina de estados seja continuamente executada. Esta máquina de estados permanecerá suspensa enquando não ocorrer nenhuma interação (mudança de sinal). Se mais de um processo estiver ativo ao mesmo tempo, a entidade HIERARCHY_QUEUE_ENTITY selecionará apenas um para ser executado em caso de todos os módulos filhos em Estelle serem do tipo atividade ou atividade sistema, ou selecionará um conjunto de processos que estão prontos para serem executados em caso dos módulos filhos serem do tipo processo ou processo sistema.

As especificações em Estelle e em VHDL da entidade TCONSUMIDOR do protocolo PRODUTOR-CONSUMIDOR são apresentadas nas figuras 3.5 e 3.6.

*3.2 - Restrições à linguagem Estelle: Estelle**

O mapeamento de uma especificação escrita na linguagem Estelle completa para descrições VHDL pode levar a implementações VLSI ineficientes. Alguns mecanismos encontrados em Estelle geram estruturas, algumas vezes, dificeis de implementar em sistema com um único processador e até mesmo em sistema com multiprocessadores. Outros mecanismos geram problemas como a colisão de primitivas de serviços, conflitos na alocação de extremidade de uma conexão na interface entre dois níveis adjacentes de protocolo. Portanto, para a obtenção de implementações eficentes deve-se restringir a linguagem Estelle.

A definição de um conjunto de restrições à linguagem Estelle visando a obtenção de um mapeamento eficiente em VHDL resultou numa versão denominada de Estelle**. Esta versão inclui alguns requisitos elaborados por Courtiat para Estelle* e outros que facilitam o mapeamento para a linguagem VHDL. As restrições impostas a Estelle são portanto as seguintes:

1. Em configurações estática entre instâncias de módulos, considerar o conjunto abaixo de condições suficientes que impõem algumas restrições na utilização de alguns aspectos da linguagem Estelle. Este requisito torna possível que o paralelismo entre as instâncias de módulo, cujo relacionamento não é da forma ascendente/descendente, possa ser representados de forma *assíncrona*. É razoável não utilizar configurações dinâmicas na definição de arquiteturas em hardware.

```
Specification PRODUTOR_CONSUMIDOR;
  default individual queue ; timescale seconds;
  type  string = array [0..90] of char;
  const Maxdados = 5;
  type MSG_TYPE = array [0 ..MaxDados]
        of char;

channel  tipo_canal (envia_msg, envia_lib);
  by envia_msg:
     msg (Mensagem ; MSG_TYPE);
  by envia_lib:
     lib (Ack : bit);

module TPRODUTOR systemactivity;
  ip  INTERFACE : tipo_canal (envia_msg);
end;

body BPRODUTOR for TPRODUTOR;
   var    mens : MSG_TYPE;
   state EP;
   ...
   initialize to EP
      begin
      end;
   trans
      from EP
         to same
         when INTERFACE.lib
         name TP:
          begin
             criar_msg (mens);
             output INTERFACE.msg(mens);
          end;
end (* BPRODUTOR*);

module TCONSUMIDOR systemactivity;
   ip INTERFACE : tipo_canal (envia_lib);

body BCONSUMIDOR for
         TCONSUMIDOR;
   var    mens : MSG_TYPE;
   var    ac : bit;

   procedure criar_lib (var lib : bit);
   begin
      lib := 1;
   end;
   state S0, S1;

   trans
   from S0
      to S1
      name TC1:
      begin
         criar_lib(ac);
         output INTERFACE.lib(ac);
      end;
   from S1
      to S1
      when INTERFACE.msg(mens);
      name TC2:
      begin
         -- consumir mensagem
         criar_lib(ac);
         output INTERFACE.lib(ac);
      end;
end (* BCONSUMIDOR *);

modvar
   PRODUTOR : TPRODUTOR;
   CONSUMIDOR : TCONSUMIDOR;

   initialize
   begin
      init PRODUTOR with
            BPRODUTOR;
      init CONSUMIDOR with
            BCONSUMIDOR;
      connect PRODUTOR.interface to
         CONSUMIDOR.INTERFACE;
   end;
end (* PRODUTOR_CONSUMIDOR *);
```

Figura 3.5 - Especificação em Estelle do protocolo Produtor_Consumidor

```
package PROD_CONS_PACK is
   constant time_scale : time := 1 sec;
   type string is array (0 to 90) of character;
   constant MaxDados : integer := 5;
   type MSG_TYPE is array (0 to MaxDados)
       of character;
end PROD_CONS_PACK;

use work.PROD_CONS_PACK.all;
entity TPRODUTOR is
    port (msg : out MSG_TYPE;
       lib : in bit);
end TPRODUTOR;

architecture BPRODUTOR of
       TPRODUTOR is
   signal  mens : MSG_TYPE;
   type    ESTADO is (EP);
begin
process
   variable state : ESTADO :=EP;
  ...
   begin
       if  state = EP then
         wait until lib = '1'; --recebe liberacao
         criar_msg (mens); -- produz msg
         msg<=mens;   -- envio da mensagem
         state :=EP;
       end if;
   end process;
end    BPRODUTOR;

use work.PROD_CONS_PACK.all;
entity TCONSUMIDOR is
      port (msg : in MSG_TYPE;
            lib : out bit);
end T_CONSUMIDOR;
architecture BCONSUMIDOR of
   TCONSUMIDOR is
   signal mens : MSG_TYPE;
   type ESTADO is (S0,S1);
begin
process
  variable state : ESTADO :=S0;
  ...
 begin
   case state is
        when   S0 =>
       -- liberação inicial do produtor
            lib <= '1', '0' after 1 sec;
        when S1  -- recepção de msg
            wait until msg(0) ='1';
            mens <=msg;
            consome(msg);
            lib <= '1', '0' after 1 sec;  --libera
      end case;
   end process;
end BCONSUMIDOR;

use work.PROD_CONS_PACK.all;
entity PRODUTOR_CONSUMIDOR is
end PRODUTOR_CONSUMIDOR;

architecture structure of
    PRODUTOR_CONSUMIDOR is
    component   TPRODUTOR
      port (msg : out MSG_TYPE;
            lib : in bit);
    end component;
    component   TCONSUMIDOR
      port (msg : in MSG_TYPE;
            lib : out bit);
     end component;
    signal Pmsg, Cmsg :   MSG_TYPE;
    signal Plib,Clib : bit;
    begin
       P: TPRODUTOR
           port map (msg => Pmsg,
                     lib =>Plib);
       C: TCONSUMIDOR
           port map (msg => Cmsg,
                     lib =>Clib);
        Cmsg <= Pmsg;
        Plib <= Clib;
    end structure;
configuration PROD_CONS_BEHAVIOUR of
    PRODUTOR_CONSUMIDOR is
    for structure
       for P : TPRODUTOR
          use entity work.TPRODUTOR
             (BPRODUTOR);
       end for;
       for C : TCONSUMIDOR
          use entity work.TCONSUMIDOR
             (BCONSUMIDOR);
       end for;
    end for;
end PROD_CONS_BEHAVIOUR;
```

Figura 3.6 - Especificação em VHDL do protocolo Produtor_Consumidor

Condições suficientes para o caso onde a configuração é estática:

a) - Não existe a cláusula de prioridade associada a qualquer transição definida com a cláusula "**when**".
b) - As únicas instâncias de módulo ativas são as folhas da árvore de instâncias de módulos.

2. O mecanismo de comunicação utilizado é o rendez-vouz.
3. Se nós restrigirmos o uso da cláusula "**delay**" em Estelle para cláusulas da forma "delay(E1)" ou "delay(E1,E1)", a cláusula "delay" poderá ser mapeada diretamente em VHDL utilizando a opção ***for P*** da estrutura **wait.**
4. Retirar a prioridade pai/filho de Estelle.
5. Ao invés de utilizar variáveis partilhadas entre módulos pai/filho, passar as informações através de pontos de interação.

O uso destas restrições garantirão um mapeamento mais eficiente em VHDL e, por conseguinte, circuitos VLSI mais simples. Estas restrições evitarão a implementação dos seguintes mecanismos: de gerenciamento de filas, de gerenciamento de módulos do tipo processo sistema ou do tipo processo, de prioridade e de controle do paralelismo síncrono entre instâncias de módulos.

4 - Exemplo de um Protocolo de Alto Desempenho : ABRACADABRA-HS

O protocolo ABRACADABRA_HS é baseado no protocolo de referência ABRACADABRA da ISO e foi criado com o objetivo de demonstrar a metodologia proposta. Trata-se de um protocolo orientado a conexão que reúne características de maior parte dos protocolos de alto desempenho. Seu projeto engloba mecanismos para os serviços de:

- *Gerenciamento de Conexão:* o esquema "two-way handshake" utilizado para a operação de estabelecimento de conexão é baseado no do protocolo Datakit [3] e o esquema "three-way handshake" utilizado para a operação de término de conexão é baseado no do protocolo XTP [7]. A operação de sinalização utiliza o esquema "out_of_band" semelhante ao do protocolo Datakit;
- *Confirmação:* o serviço de confirmação utiliza um esquema de confirmação que é independente do transmissor e é baseado no do protocolo OSI/TP4;
- *Controle de Fluxo:* os dois esquemas de controle de fluxo usados são: janela e o de controle de taxa. Este esquemas são baseados no do protocolo XTP;
- *Manipulação de Erros:* a deteção de erros é obtida através do número de sequência. O número de sequência é baseada em bytes. Esse esquema é encontrado no protocolo XTP. A notificação de erros é obtida através de um Reject não seletivo. O protocolo XTP utiliza um Reject seletivo mas, como o objetivo é obter um protocolo simples e didático, optou-se pelo uso de um reject não selectivo.

A especificação deste protocolo em Estelle está organizada em três camadas de protocolos: uma camada corresponde ao protocolo propriamente dito, isto é, o ABRACADABRA_HS, e as outras correspondem ao usuário e ao meio utilizado pelo ABRACADABRA_HS.

```
var_comp = record
   First_Received_PDU: boolean;
   E, R : integer;
end;
channel ABRA_interface (user, provider);
   by user: Conreq; ConRsp; DisReq;
      DatReq(SDU:SDU_tipo);
   by provider : ConInd; ConCnf; DisInd;
      DatInd(SDU:SDU_tipo);
...
module TSINAL_ENTITY activity;
   ip  UCEP : ABRA_interface(provider);
       MCEP : MEDIUM_interface(user);
   ...
 end;
body BSINAL_ENTITY for
   TSINAL_ENTITY;
   var  PDU_aux: pdu_type;
   state
      CLOSED, WFCC,CONTROL,
      WFDCC, CLOSING, ERROR,
      WFUR, WFDTCOMP_mcep,
      WFDTCOMP_sinc,FRP_TRUE_wfdcc,
      FRP_TRUE_closed;
trans
from CLOSED to WFCC
   when UCEP.ConReq
      name A1mandaCR:
      begin
         BuildCR(PDU_aux);
         output MCEP.UnitReq (PDU_aux);
      end;
from WFCC
   when MCEP.UnitInd
      provided (Code (PDU) = CC) or
                              (Code(PDU) = CR)
         to CONTROL
         name A2recCCCR:
         begin
         end;
      provided (Code(PDU) = DR)
         to FRP_TRUE_wfdcc  (* WFDCC
         name A3recDRmandaDC:
         begin
         end;
      provided otherwise to WFCC
         name A4naorecuCRCCDR:
         begin
         end;
   ...
end;  (* de BSINAL_ENTITY *)
```

figura 4.2 - Especificação em Estelle de parte do protocolo ABRACADABRA-HS

O protocolo ABRACADABRA-HS, conforme ilustrado na figura 4.1, é estruturado em cinco módulos denominados *Sinalização, Dados, Região_Crítica, Geração_Ctrl e Controle_taxa*. O módulo *Sinalização* é responsável pelo gerenciamento de mensagens de sinalização, que inclui as mensagens utilizadas no procedimento de estabelecimento/término de associação e as mensagens de controle. O módulo *Dados* é responsável somente pelo gerenciamento das mensagens de dados. O módulo *Região_Crítica* é responsável por garantir a exclusão mútua no acesso pelos módulos de Dados e de Sinalização de determinadas variáveis. O módulo *Geração_Ctrl*, quando ativo, é responsável pela geração periódica de um sinal ao módulo de sinalização. O módulo *Controle_taxa* é responsável pela geração periódica de um sinal ao módulo *Dados*.

Estes módulos são conectados via canais internos. Dois destes canais, o UCEP e o MCEP, são externos e conectam, respectivamente, o módulo *Usuário* e o módulo *Meio*. Existem seis canais internos que são: SINC_DATA, SINC_CTRL, SINCBEG_TAXA, SINC_TAXA, ACESSO_SINAL e ACESSO_DATA. O canal **SINC_DATA** conecta o módulo *Sinalização* com o módulo *Dados*. O canal **SINC_CTRL** conecta o módulo

Sincronização com o *Geração_Ctrl.* O canal **SINCBEG_TAXA** conecta o módulo *Sinalização* com o módulo *Controle_taxa*. O canal ***SINC_TAXA*** conecta o módulo *Dados* com o módulo *Controle_taxa.* O canal ***ACESSO_SINAL*** conecta o módulo *Sinalização* com o *Região_Crítica.* O canal ***ACESSO_DATA*** conecta o módulo *Dados* com o *Região_Crítica.*

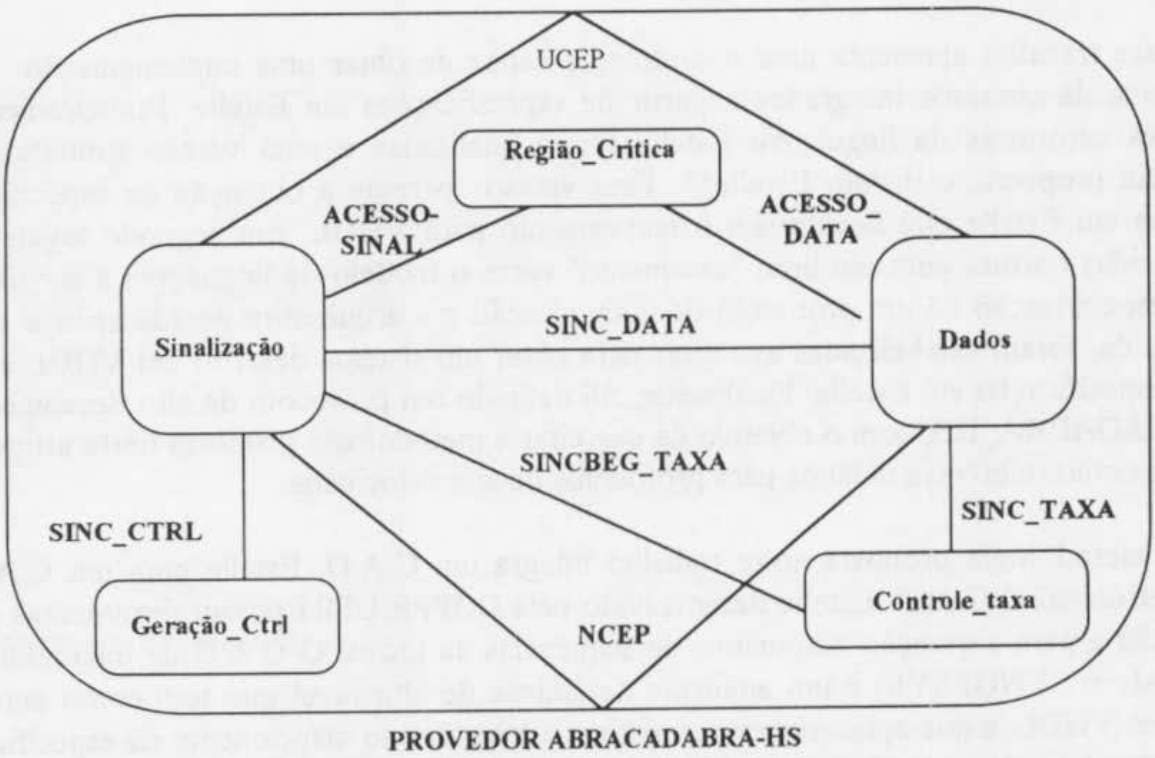

Figura 4.1 - Arquitetura do Provedor ABRACADABRA-HS

```
package ABRA_HS_PACK is
   signal E: integer;  -- Num. de seq. de emissao
   signal R: integer;  -- Num. de seq. de recepcao
   signal First_Received_PDU: boolean;
end ABRA_HS_PACK;
use work.ABRA_HS_PACK.all;
entity TSINAL_ENTITY is
   port(ConReq, ConRsp, DisReq: in bit;
        DatReq: in SDU_tipo;
        ConInd, ConCnf, DisInd: out bit;
        DatInd: out SDU_TIPO; ... );
end TSINAL_ENTITY;
architecture BSINAL_ENTITY of
          TSINAL_ENTITY is
   type ESTADO is (CLOSED,WFCC,CONTROL,
      WFDCC, CLOSING, ERROR, WFUR,
      WFDTCOMP_mcep, WFDTCOMP_sinc,
      FRP_TRUE_wfdcc, FRP_TRUE_closed);
   begin
     process
       variable state: ESTADO := CLOSED;
       variable PDU_aux, PDU: PDU_type;
       begin
       case state is
         when CLOSED =>
           wait on ConReq;
           if ConReq = '1' then  -- A1mandaCR
             BuildCR(PDU_aux);
             output_MCEP_UnitReq(PDU_aux);
             state := WFCC;
           elsif UnitInd.codigo /= CR then ...
           end if;
         when WFCC =>
           wait on UnitInd;
           if Code(PDU)=CC or Code(PDU)=CR then
             state:=CONTROL;
             ...
           elsif Code(PDU)=DR then
             state:= FRP_TRUE_wfdcc;
             ...
           else
             state := WFCC;
           end if;
       end case;
     end process;
   end BSINAL_ENTITY;
```

Figura 4. 3 - Especificação em VHDL de parte do protocolo ABRACADABRA-HS

Na figura 4.2 e 4.3 são mostradas, respectivamente, uma especificação em Estelle e uma especificação em VHDL de um trecho do protocolo ABRACADABRA_HS aplicando a metodologia de mapeamento.

5 - Consideracões Finais

Esse trabalho apresenta uma metodologia capaz de obter uma implementação (semi) automática de circuitos integrados a partir de especificações em Estelle. Primeiramente, as diferentes estruturas da linguagem Estelle foram analisadas e uma versão simplificada de Estelle foi proposta, o dialeto Estelle**. Essa versão permite a obtenção de especificações eficientes em Estelle que facilitariam o mapeamento para VHDL. Em segundo lugar, foram estabelecidas normas para um bom "casamento" entre o modelo da linguagem a ser utilizada para a especificação de um protocolo de comunicação e a arquitetura gerada após a síntese. Em seguida, foram estabelecidas as regras para obter um sistema descrito em VHDL a partir de sua especificação em Estelle. Finalmente, foi definido um protocolo de alto desempenho, o ABRACADABRA_HS, com o objetivo de exercitar a metodologia proposta neste artigo e ser utilizado como referência didática para protocolos de alta velocidade.

A metodologia proposta neste trabalho integra um C.A.D. Estelle com um C.A.D de microeletrônica. O C.A.D. Estelle desenvolvido pela COPPE/UFRJ possui ferramentas para a verificação e para a geração automática de sequências de testes. O C.A.D de microeletrônica (AMICAL + SYNOPSYS) é um ambiente de síntese de alto nível que tem como entrada a linguagem VHDL, e que apresenta características adequadas ao mapeamento de especificações em Estelle. A implementação em hardware VLSI do protocolo de alto desempenho proposto, o ABRACADABRA_HS, será obtida com o uso dos recursos do laboratório do Institut National Polytechnique de Grenoble na França disponíveis a partir de um acordo de cooperação tecnológica entre a UFRJ e o INPG.

6 - Referências bibliográficas

.[1] D.Cheriton , C. Willianson, "VMTP as Transport Layer for High-Performace Distributed Systems", IEEE Communications Magazine, june 1989, pp 37-44.

.[2] W. Doeringer et all, "A Survey of Light-Weight Transport Protocols for High-Speed Networks", IEEE transactions on Communications, vol. 388, NO. 11, november 1990, pp 2025-2038.

.[3] A. Fraser, W. Marshall, "Data Transport in a Byte Stream Network", IEEE Journal on Selected Areas in Communications, Vol. 7, NO.7, September 1989.

.[4] .A. Netravali et all, "Design and Implementation of a High-Speed transport Protocol", IEEE transaction ons Communications, Vol. 38, No. 11, November 1990, pp 2010-2024.

.[5] B. Doshi, P. Johri, "Communications Protocols for High Speed Packet Networks", Computer Networks and ISDN Systems", 24(1992) ppp 243-273.

.[6] Robin Milner, "Communication and Concurrency", Prentice-Hall, 1989.

.[7] Strayer W. et all, "XTP: The Xpress Transfer Protocol", Addison-Wesley Publishing Company, Inc.

.[8] S. Budkowski and P. Dembinski, "An Introduction to Estelle : A Specification Language for Distributed Systems", Computer Network and ISDN System 14 (1987) 3-23, North-Holland

.[9] J. P. Courtiar, "Estelle* : A powerful Dialect of Estelle for OSI Protocol description", International Workshop on Protocol Specification verification and testing, Toulouse-Moissac, France, 1988

.[10] Courtiar, J. P. et all "Estelle: Un language ISO pour les Algoritmed Distribués et protocoles", Rapports de Recherche Iria Rennes NO. 595, 1986

.[11] ISO/TC97/SC21/WG16-1, "Estelle : a formal Description Technique Based on a Extended State transiction Model", DP9074,1987

.[12] King P. , "Formalization of Protocol Engineering Concepts", IEEE transactions on Computers, vol. 40, NO. 4, April 1991,pp 387 - 403

.[13] Vissers C. et all, "Formal Description Techniques", Proceeding of IEEE, Vol. 71. NO. 12, December 1983, pp 1356 - 1364

.[14] Chanson et all, "On tools supporting the use of formal description techniques in protocol development", Computer Networks and ISDN Systems 25 (1993) 723-739

.[15] G. T'Hooft, "Formal Description Techniques: Communication Tolls for Data Communication Specialists", Computer Networks and ISDN Systems, 14 (1987) 311 - 321

.[16] Bochmann G. e Sunshine C., "Formal Methods in Communication Protocol Design", IEEE transactions on communications, Vol. 28, NO. 4 , April 1980, 624 - 631

.[17] IEEE Std 1076 - 1987, IEEE Standard VHDL Language Reference Manual, March 31,1988

.[18] Peter J. Ashenden, "The VHDL Cookbook", Dept. Computer Science of University of Adelaide, 1990.

.[19] Daniel D. Gajski et all, "High - Level Synthesis : Introduction to Chip and System Design", Kluwer Academic Publishers, 1992

.[20] Z. Navabi, "A high-level language for Design and Modeling of Hardware", J. System Software, 1992 : 18 : 5-18

.[21] R. L. Land, "Síntese de Alto nível de Protocolos de Rede Especificados na Linguagem Estelle", Tese de M. Sc., Engenharia Elétrica da Coppe/Ufrj, 1994

.[22] Krishnakumar, A. S. and Sabnani K.K. ,"VLSI Implementation of Communication Protocols - A survey", IEEE Journal on select Areas in Communications, Vol 7, No 7 : 1082 - 1090, 1989, North Holland

.[23] Braun T. , Shiller J. et Zitterbart M., "Implementation of Transport Protocols using Paralelism and VLSI Components",

.[24] C. D. Kloos et all, "VHDL generation from a timed extension of the formal description technique LOTOS within the FORMAT project", Microprocessing and Microprogramming 38 (1993) 589-596, North-Holland.[25] Park, O'Brien K., Jerraya A. A., Tutorial - AMICAL :Interative Architectural Synthesis Based on VHDL", Internal Report, IMAG/TIM3, Março de 1982

.[25] Park, O'Brien K., Jerraya A. A., Tutorial - AMICAL :Interative Architectural Synthesis Based on VHDL", Internal Report, IMAG/TIM3, Março de 1982