# Síntese Lógica do Protocolo IPv6: Resultado de uma Metodologia visando o Projeto de Protocolos em Hardware

Emerson Carli, Ricardo N. B. Lima, Aloysio C. P. Pedroza,
Luci Pirmez e Antônio C. de Mesquita

Grupo de Teleinformática e Automação (GTA)
Universidade Federal do Rio de Janeiro
COPPE/EE - Programa de Engenharia Elétrica/Depto. de Eletrônica
Caixa Postal 68504 - CEP 21945-970 - Rio de Janeiro - RJ - Brasil
e-mail: {aloysio,emerson,belem}@gta.ufrj.br

**Resumo**

Neste trabalho obtém-se a síntese lógica do protocolo IPv6 para o envio de datagramas em redes ATM, resultado da aplicação de uma metodologia que visa o projeto de protocolos em *hardware*. Os diversos passos tomados de acordo com esta metodologia são apresentados. Os resultados obtidos mostraram que a maior limitação envolvida no projeto é o tamanho dos *buffers*. Como solução, foi adotada uma partição funcional do projeto onde a memória é implementada em separado da parte de controle.

**Abstract**

*In this work the IPv6 protocol's logic schematic is obtained as a result of a methodology whose aim is the hardware implementation of protocols. The main steps of this methodology are discussed. The obtained results show that the buffer length is the major problem involved. As a solution, a project's functional partition is adopted in such a way that memory is separately implemented.*

## 1 Introdução

Com o surgimento de redes de alta velocidade e de aplicações multimídias, novos requisitos de desempenho são impostos aos sistemas de processamento de dados, refletindo-se em modificações no projeto de protocolos, que devem ser mais simples e mais eficientes. Entretanto, o alto desempenho de subsistemas de comunicação é obtido não apenas com a otimização em *software* dos protocolos mas também com sua implementação em *hardware*.

A implementação de protocolos de comunicação em *hardware*, porém, não é uma tarefa trivial. Circuitos VLSI são atualmente projetados de maneira semi-automática usando-se uma série de ferramentas integradas à um determinado ambiente de projeto. O procedimento de projeto normalmente tem como ponto de partida uma especificação do sistema em alto nível, denominada de descrição comportamental. No caso dos protocolos de comunicação, entretanto, os procedimentos de projeto tradicionais envolvem especificações do sistema em um nível de abstração maior que o normalmente empregado na descrição comportamental de circuitos. Este é o caso das linguagens de especificação derivadas das Técnicas de Descrição Formal (FDTs - *Formal Description Techniques*) tais como, por exemplo, a linguagem Estelle [1]. Como conseqüência, o projetista de protocolos quando confrontando com o problema de implementar uma dada especificação em *hardware*, deverá lidar com dois ambientes de projeto distintos: um envolvendo a especificação formal, verificação e teste do protocolo e o outro envolvendo a simulação, síntese e implementação física do circuito VLSI. Em geral, os dois ambientes são baseados em diferentes linguagens de descrição, o que requer uma interface de tradução de uma linguagem para outra. Uma linguagem de descrição de *hardware* (HDL) tal como o VHDL (*Very high speed integrated circuits Hardware Description Language*) [2] é necessária para a utilização de ambientes específicos de projeto de *hardware*, tais como as ferramentas de síntese de alto nível (HLS - *High Level Synthesis*).

O objetivo deste trabalho é aplicar uma metodologia que traduz especificações Estelle em descrições VHDL [3] com o intuito de implementar em *hardware* protocolos destinados a sistemas de comunicação de alta velocidade. Este trabalho aplica esta metodologia na nova versão do protocolo IP, oficialmente chamado de IPv6, dentro da abordagem "IP over ATM".

Este trabalho está estruturado da seguinte forma. Na seção 2 é descrita a metodologia empregada, apresentando as diversas etapas tomadas durante a implementação em *hardware* de protocolos, enfocando-se na síntese de alto nível e na síntese lógica. Na seção 3 é apresentada as principais características do protocolo utilizado neste trabalho, o protocolo IPv6, bem como os principais conceitos da abordagem "IP over ATM". A seção 4 destaca os principais resultados obtidos ao se aplicar esta metodologia. A conclusão sobre a implementação em *hardware* de protocolos é descrita na seção 5.

# 2 A Metodologia Adotada

A metodologia adotada neste trabalho para a síntese de circuitos VLSI que implementa o protocolo IPv6 é ilustrada na figura 1. O procedimento adotado tem como ponto de partida uma especificação Estelle de um protocolo corretamente compilada e simulada (1) obtida com o uso de um ambiente adequado para o desenvolvimento de protocolos em Estelle [4, 5, 6]. Em seguida, esta especificação é traduzida numa descrição VHDL através de uma estratégia de mapeamento Estelle-VHDL (2), passo este necessário para a utilização de um conjunto de ferramentas de projeto de circuitos VLSI, denominado na figura de C.A.D. de projeto de circuitos. Após este mapeamento, o comportamento de uma entidade VHDL pode ser descrito tanto de maneira comportamental como de maneira estrutural. Neste trabalho apenas a descrição comportamental foi considerada, uma vez que esta descrição é mais familiar ao projetista de protocolos além de representar, com uma maior semelhança, a especificação funcional do protocolo.

A entrada do C.A.D. de projeto de circuitos é, portanto, uma descrição comportamental em VHDL do sistema (do protocolo) (3). O C.A.D. de projeto de circuitos é composto de ferramentas padrão de projeto VLSI tais como o Synopsys [7] para a síntese lógica e o Opus [8] para a geração de *layout*. O Synopsys pode ser utilizado na simulação de descrições comportamentais e estruturais em VHDL e para a síntese lógica de descrições estruturais. Já o Opus é utilizado na geração de *layout* das estruturas VLSI, a partir do esquema lógico.

A etapa seguinte do processo, consiste na obtenção de uma descrição comportamental VHDL do sistema corretamente simulada, com o auxílio da ferramenta de simulação existente no Synopsys, o *vhdldbx* (3,4). A síntese de alto nível propriamente dita consiste na aplicação de um algoritmo de *scheduling* (5,6) à esta descrição comportamental obtida em (4). A síntese de alto nível tem como definição a tradução de uma descrição comportamental em uma descrição estrutural. Entretanto, para se obter uma descrição estrutural, a descrição comportamental do sistema, que é a entrada do processo de *scheduling*, deve ser formada por um único processo principal constituído por um conjunto de componentes (*component*). Um componente é um subsistema que é representado por uma entidade VHDL. Desta maneira, se a descrição comportamental do sistema possuir subsistemas, o algoritmo de *scheduling* deve ser empregado em todos seus subsistemas, gerando descrições comportamentais sem dependências de dados e de controle. A descrição comportamental do sistema, após se aplicar a síntese de alto nível, resultará numa descrição estrutural sem dependências.

No caso do projeto de um protocolo, especificado inicialmente em Estelle, os subsistemas existentes em uma descrição comportamental são facilmente identificados, pois os submódulos (módulos-filhos), contidos numa especificação Estelle, correspondem diretamente à subsistemas descritos por entidades VHDL. Entretanto, a especificação em Estelle do protocolo utilizado neste trabalho, o protocolo IPv6, não possui submódulos. Isto corresponde a dizer que sua descrição comportamental em VHDL equivalente também não terá subsistemas (componentes). Desta maneira, ao se aplicar a síntese de alto nível nesta descrição comportamental ter-se-á como resultado uma descrição ainda comportamental só que sem dependências de dados e de controle, e não uma descrição estrutural. O algoritmo de *scheduling* utilizado foi o algoritmo DLS (*Dynamic Loop Scheduling*) proposto por Rahmouni [9]. Novamente utiliza-se a ferramenta *vhdldbx* com o intuito de se verificar, agora, a validade dessa descrição VHDL sem dependências (7,8). Esta descrição corretamente simulada é então submetida à ferramenta de síntese lógica *design_analyzer* (9,10). O esquema lógico resultante é utilizado como descrição de entrada da ferramenta Opus para a geração do *layout* desta estrutura física (11,12).

## 2.1 A Síntese de Alto Nível

Uma etapa fundamental da síntese de alto nível é o uso de um algoritmo de *scheduling* adequado à síntese de compiladores comportamentais dominados por fluxo de controle [10]. Neste trabalho é empregado o algoritmo DLS. Este algoritmo tem como entrada uma descrição comportamental VHDL composta pelo par *entity/architecture*.

Este algoritmo é baseado nos princípios de escalonamento de caminhos (*path scheduling*) [11] mas reduz de maneira significativa o número de caminhos gerados e, conseqüentemente, o custo de computação e a complexidade do *hardware* gerado posteriormente. Este algoritmo consiste dos seguintes passos:

- geração do grafo de fluxo de controle;
- geração de caminhos;

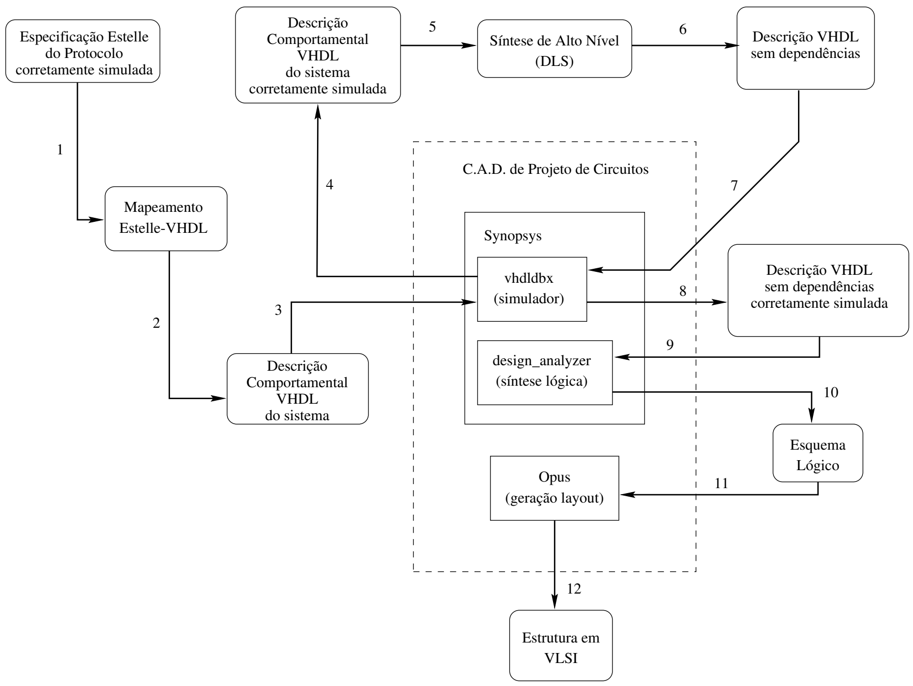



Figura 1: A Metodologia adotada na obtenção de circuitos VLSI.

- geração de estados e transições.

A principal representação usada pelo DLS é um grafo de fluxo de controle (CFG - *Control-Flow Graph*). Este grafo fornece um meio conveniente de representar as propriedades inerentes de circuitos caracterizados por fluxo de controle, que são descritos por *loops* aninhados e por construções de sincronismo tais como a construção *wait*. Este CFG contém nós, representando as operações da descrição original, e ramos, indicando os fluxos de sinal da descrição original. O CFG identifica três tipos diferentes de operações: a construção *wait*, as construções de fluxo de controle (como *if* e *while*) e expressões.

A mais importante característica deste CFG é a introdução do nó *wait*. Um nó *wait* significa sempre uma mudança de estado que será efetuada. Isto causa a interrupção incondicional da geração de caminhos, evitando desta maneira o problema da explosão de caminhos [9].

Um caminho $C = (n_1, n_2, ...., n_i)[n_s]$ é definido como um conjunto de nós, começando pelo nó $n_1$ e tendo como seu sucessor um outro caminho iniciado por $n_s$. Novos caminhos são gerados conforme se progride no CFG começando-se pelo primeiro nó. A geração de um caminho é interrompida e um novo caminho se inicia quando uma das duas condições abaixo é satisfeita:

- uma construção *WAIT* é encontrada (condição $C_1$);

- o novo nó encontrado possui dependência de dados com relação à um dos nós já existentes no caminho (condição $C_2$).

Na seção 4 este algoritmo será aplicado ao protocolo IPv6 tornando mais claro o entendimento dos conceitos apresentados anteriormente. Partindo-se das representações dos CFGs, todos os caminhos são então calculados. O primeiro caminho inicia-se com o primeiro nó do CFG. Nós sucessivos são seqüencialmente adicionados ao caminho até que uma das condições mostradas anteriormente ($C_1$ e $C_2$) seja satisfeita. Na fase de geração de estados e transições, cada caminho gerado irá constituir uma transição. As transições em conjunto com os estados gerados irão produzirá uma máquina de estados (FSM). Transições adicionais serão geradas devido aos *loops* implícitos de todas as construções *wait*. Todos os caminhos iniciados com o mesmo nó serão colocados juntos no mesmo estado. O resultado do DLS é uma tabela de estados e transições que pode ser manipulada pela ferramenta de síntese lógica existente no Synopsys, *design_analyzer*.

## 2.2 A Síntese Lógica

Síntese é um método automático de converter uma descrição de alto nível de abstração em uma descrição de baixo nível. A síntese lógica envolve o mapeamento de uma descrição no nível RTL do domínio comportamental em uma *netlist* no nível de portas lógicas. Esta *netlist* especifica as interconexões de macrocélulas descritas ao nível de portas lógicas. Na figura 2 é mostrada a estrutura geral de um ambiente de síntese lógica.

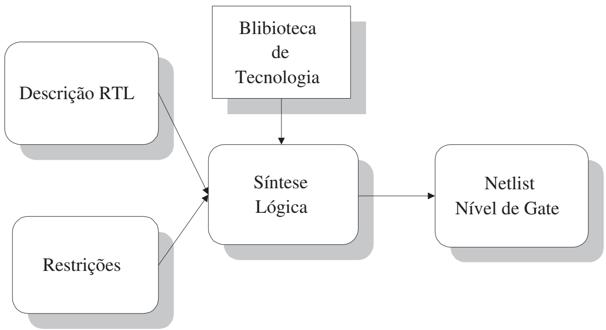



Figura 2: Estrutura geral de um ambiente de síntese lógica.

A descrição RTL é um estilo que especifica todos os registradores e a lógica combinacional entre eles. Os registradores podem ser indicados seja explicitamente através da instanciação de componentes seja implicitamente através de inferência. A lógica combinacional pode ser descrita através de equações lógicas, instruções de controle seqüenciais, subprogramas ou instruções concorrentes. No primeiro passo da síntese lógica a descrição RTL é convertida numa descrição booleana não otimizada como é mostrado na figura 3.

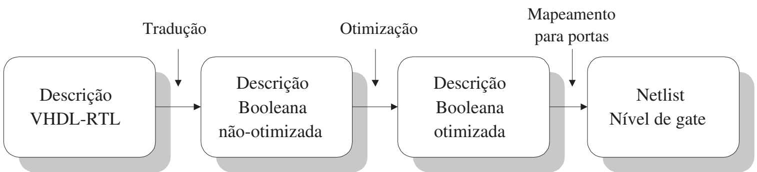



Figura 3: Etapas da síntese lógica.

A tradução consiste em converter todas as construções *if*, *case*, *loop*, atribuições condicionais de sinal e instruções de atribuição de sinal em equivalentes booleanos. *Flip-flops* e *latches* podem ser instanciados ou inferidos.

# 3 O Protocolo IPv6 e a Abordagem “IP over ATM”

## 3.1 O Protocolo IPv6

O atual protocolo IP (IPv4) está se tornando inadequado para suportar os atuais requisitos de desempenho das novas aplicações. Entretanto, o principal motivo para a atualização do atual protocolo IP é o esgotamento iminente do seu espaçamento de endereços de 32 bits.

Em resposta, pesquisas foram realizadas e chegou-se ao desenvolvimento de uma nova versão deste protocolo, oficialmente chamado de **IPv6** [12]. Resumidamente, as mudanças introduzidas para o IPv6 podem ser vistas sob 3 aspectos:

- *endereços maiores.* A mudança mais perceptível é o novo tamanho destinado aos campos de endereçamento do datagrama. O protocolo IPv6 quadruplicou o tamanho do endereçamento do IPv4 de 32 bits para 128 bits;

- *cabeçalho de formato flexível.* O protocolo IPv6 adota um formato completamente novo para seu datagrama. Ao contrário do IPv4 que usa um cabeçalho único de tamanho variável, o protocolo IPv6 utiliza um cabeçalho básico de tamanho fixo seguido ou não de cabeçalhos opcionais;

- *suporte aos tráfegos multimídias.* O protocolo IPv4 oferece pouco suporte aos tráfegos multimídias. Já o IPv6 permite que os datagramas pertencentes a um mesmo fluxo sejam “rotulados”, traduzindo desta maneira o tratamento especial que devem ter por parte dos roteadores intermediários.

O protocolo IPv6 mantém muitas das características do protocolo IPv4. Como exemplo, o protocolo IPv6 continua sendo um protocolo não orientado a conexão (tratando cada datagrama independentemente). Neste trabalho, será implementado em *hardware* algumas das funções básicas do protocolo IPv6 para o envio e a recepção de datagramas. Na transmissão de datagramas, o protocolo IPv6:

- fragmenta o datagrama se necessário, inserindo o cabeçalho de fragmentação estendido;

- envia o datagrama para o *driver* apropriado da camada enlace. No caso em questão, será considerado o protocolo AAL5 imediatamente abaixo do IPv6, de acordo com a abordagem “IP sobre ATM”.

Na recepção de datagramas, o protocolo IPv6:

- verifica se o datagrama está corretamente formatado;

- concatena o datagrama se necessário;

- passa a mensagem encapsulada para o protocolo de transporte apropriado. Neste caso, é considerado apenas o protocolo TCP.

## 3.2 A Abordagem "IP over ATM"

Este trabalho adota a abordagem "IP over ATM" na implementação do protocolo IPv6 dada sua popularidade [13, 14]. Na abordagem "IP over ATM" o mecanismo de resolução de endereços deve traduzir diretamente o endereço IP para o endereço ATM e vice-versa, fornecendo a ilusão da existência de sub-redes lógicas IP (LIS - *Logical IP Subnetworks*). LIS são grupos de estações que possuem o mesmo endereço de sub-rede IP mas que estão fisicamente dispersos ao longo da rede ATM. Assim, cada LIS age como uma rede LAN.

Duas AALs foram definidas para protocolos que trabalham com datagramas como é o caso do protocolo IPv6: AAL3/4 e AAL5. A AAL5 é, em geral, utilizada por melhor atender às características das redes locais: transferências de dados assíncronos, vazões variáveis, serviço não orientado a conexão e maior eficiência. A AAL5 pode suportar Unidade de Dados de Serviço (AAL_SDU) com tamanho de até 64k octetos. Entretanto, o impacto da transmissão de pacotes deste porte pode ser significativo quando os pacotes entram em enlaces com MTUs menores. Para atender este propósito, o *Networking Workgroup* produziu um documento, intitulado "Default IP MTU for use over ATM AAL5" [15], que define o valor de 9.180 octetos como a MTU para o protocolo IP.

Uma característica importante numa sub-rede LIS é o formato do encapsulamento para o envio de datagramas IPv6 em redes ATM. A figura 4 mostra o encapsulamento dos datagramas IPv6 para formar uma AAL_SDU, conhecido como encapsulamento LLC/SNAP.

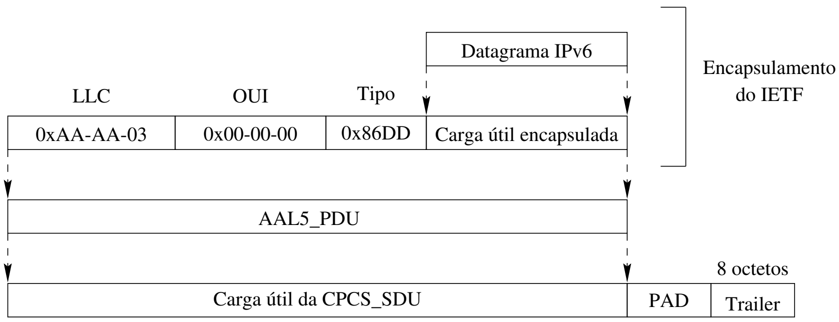



Figura 4: Encapsulamento LLC/SNAP dos datagramas IPv6.

Tal encapsulamento se faz necessário pois o protocolo AAL5 não possui em seu *trailer* um campo que defina o tipo de protocolo que está sendo carregado (*type field*). No caso específico do protocolo IPv6, o campo que define o tipo do protocolo (*tipo* na figura 4) possui o valor hexadecimal 0x86DD.

Analogamente às outras tecnologias de rede, a tecnologia ATM atribui para cada estação conectada um endereço físico que deve ser usado no estabelecimento de um canal virtual. O gerenciamento de canais virtuais ocorre na interface de rede abaixo do protocolo IPv6. Quando uma estação (A) deseja enviar um datagrama para outra estação (B), o protocolo IPv6 em (A) executa o algoritmo de roteamento tradicional para determinar o endereço apropriado do próximo nó (N), que também é um endereço IPv6, para o qual o datagrama será enviado. De posse deste endereço, a estação (A) o passa juntamente com o datagrama IPv6 para a interface de rede, que examina sua tabela de canais virtuais abertos. Neste ponto, dois caminhos podem ser tomados pela interface de rede. Se existir um canal virtual para (N), este endereço é mapeado no par VPI/VCI do canal virtual e a estação utiliza o protocolo AAL5 para enviar o datagrama. Caso contrário, antes da estação (A) enviar o datagrama, esta estação deve exercer a sinalização ATM [16] para estabelecer o canal virtual necessário. Para tanto, a estação entra em contato com um servidor existente na sua LIS, chamado de "ATMARP Server", de modo a traduzir o endereço IPv6 de (N) no seu respectivo endereço ATM. Após a criação do canal virtual, a estação (A) o adiciona à sua tabela e o envia para a estação destino.

Neste trabalho será considerado que os canais virtuais utilizados para a transferência de informações entre duas estações já estão estabelecidos, ou seja, que o processo de sinalização ATM necessário para criá-los já foi realizado. Será enfocado principalmente a transferência de um datagrama IPv6 para o protocolo AAL5, utilizando-se o encapsulamento LLC/SNAP.

# 4 Resultados Obtidos

As principais etapas da metodologia empregada neste trabalho para a implementação em *hardware* do protocolo IPv6 são mostrados nas seções subseqüentes. Na subseção 4.1 é mostrada a descrição comportamental em VHDL resultante do mapeamento Estelle-VHDL, enfocando o mecanismo de sincronização necessário para a comunicação entre entidades VHDL. A descrição sem dependências obtida através do algoritmo de *scheduling* DLS é apresentada na subseção 4.2. Finalmente, na subseção 4.3 é apresentada o resultado da síntese lógica pela ferramenta Synopsys aplicada à descrição sem dependências do protocolo.

## 4.1 A Descrição Comportamental do Protocolo IPv6

A descrição comportamental em VHDL da especificação em Estelle do protocolo IPv6 é obtida após o mapeamento Estelle-VHDL. A figura 5 ilustra, em parte, a entidade do protocolo IPv6. Deve-se lembrar que a arquitetura do protocolo foi descrita de maneira comportamental permitindo, assim, que os projetistas de protocolos os descrevam de modo mais abstrato.

Para que a troca de informações entre entidades VHDL seja eficaz é necessário que haja uma sincronização entre as entidades, de modo que uma entidade não envie dados a uma outra entidade sem que esta última esteja preparada para recebê-los. Esta sincronização é estabelecida através de pontos de sincronismo (sinais de controle) existentes em ambas entidades. Para que uma entidade `A` envie uma mensagem a uma entidade `B`, é necessário que `A` envie um sinal a `B` informando a existência de transmissão de uma mensagem. Por exemplo, quando uma entidade IPv6 deseja enviar uma mensagem para uma entidade AAL5, a instrução `AAL5_Unitdata_Req_cod <= '1'` é usada. Entretanto, a execução da instrução de envio deste sinal não habilita a entidade emissora de enviar o sinal. É necessário que a entidade receptora esteja preparada para recebê-la. Após o envio do sinal informando a existência de uma mensagem a ser transmitida, a entidade emissora suspende seu processo até receber um sinal de confirmação. A espera pelo sinal de recepção é exemplificado pela linha de código `WAIT UNTIL aal5preparado = '1'`. Vale lembrar que este sincronismo acontece em ambos os sentidos de comunicação das duas entidades envolvidas. O mecanismo de sincronização descrito acima realiza-se a nível de mensagem. No caso de protocolos, o envio de uma mensagem pode ser representada por uma primitiva de serviço. Uma primitiva de serviço pode ter vários parâmetros. Dentre eles, o parâmetro

```
entity IPv6_entity_type is
        port (start: in bit;
              clk: in bit;
              AAL5_Unitdata_Req_cod: out bit;
              AAL5_Unitdata_Req_version: out bit_4;
              AAL5_Unitdata_Req_priority: out bit_4;
              ipv6habilitado: out bit;
              aal5preparado: in bit;
              IPv6_Data_Req_cod: in bit;
end IPv6_entity_type;
architecture IPv6_entity_body of IPv6_entity_type is
    begin
        process
            begin
                AAL5_Unitdata_Req_cod <= '0';
                WAIT UNTIL (start = '1' and clk = '1');
                MEF: LOOP
                IF (IPv6_Data_Req_cod = '1') THEN
                 LermsgUserIPv6;
                 AAL5_Unitdata_Req_cod <= '1';
                 WAIT UNTIL aal5preparado = '1';
                 EnviarmsgIPv6aal5;
                ELSIF (AAL5_Unitadata_Ind_cod = '1') THEN
                .
                ENDIF;
                WAIT UNTIL (clk = '1');
                IPv6_Data_Req_cod <= '0';
                END LOOP MEF;
            end process;
end IPv6_entity_body;
```

Figura 5: Uma parte da descrição comportamental protocolo IPv6.

de dados (SDU) merece uma atenção especial, visto que este parâmetro é de tamanho variável. O sincronismo necessário para o envio e recepção de dados entre entidades de uma mesma estação, análogo ao sincronismo de mensagens, deve ser definido.

Este mecanismo de sincronização utilizado na troca de dados pode ser melhor visualizado na figura 6. Antes da entidade IPv6 receber o primeiro dado, esta habilita a entidade AAL5 através do sinal `ipv6habilitado` e suspende seu processo através do código `WAIT UNTIL ready_validin2 = '1'`. A entidade AAL5 então envia o primeiro dado, o sinal `varind_validout3` e suspende seu processo. Este sinal `varind_validout3` reabilita a entidade IPv6 a continuar a execução do processo, efetuando o tratamento do dado, enviando o sinal `ready_ackin2` e suspendendo novamente o seu processo. Este procedimento se repete até que o último dado seja enviado.

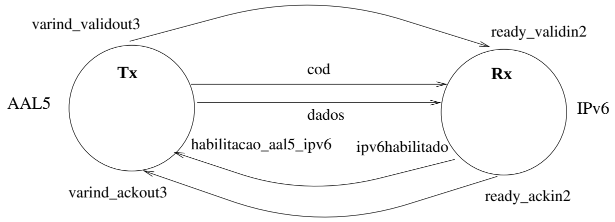



Figura 6: Diagrama simbólico do mecanismo de sincronização.

A visão global do diagrama em blocos dos protocolos IPv6 e AAL5, com os sinais de controle necessários para efetuar a comunicação entre eles é ilustrada na figura 7.

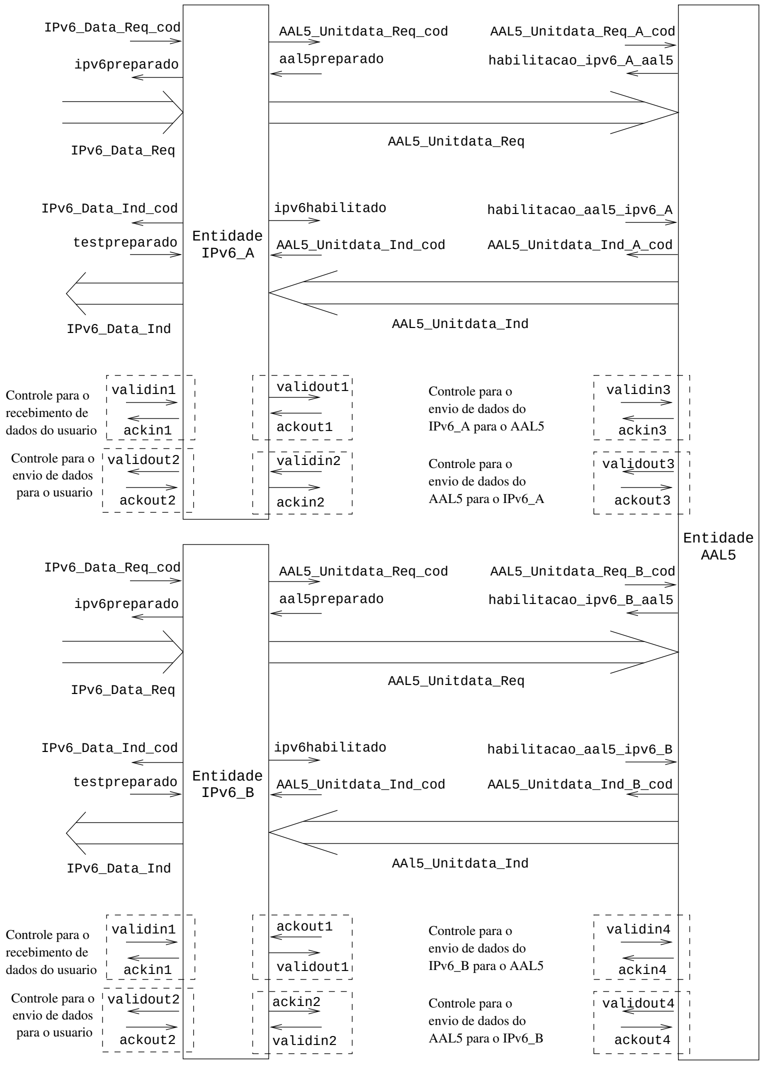



Figura 7: Sinais de controle trocados entre os protocolos IPv6 e AAL5.

Nesta figura, a comunicação entre duas entidades do protocolo IPv6 é realizada através da entidade do protocolo AAL5, que possui dois conjuntos de sinais para cada sentido da comunicação. Quando a entidade IPv6_A deseja enviar dados à sua entidade par IPv6_B, a entidade AAL5 recebe os dados pelo conjunto de sinais da família `AAL5_Unitdata_Req_A` e os envia para a entidade IPv6_B pelos sinais da família `AAL5_Unitdata_Ind_B`. No sentido oposto da comunicação, ou seja, quando a entidade IPv6_B deseja enviar dados à sua entidade par IPv6_A, os dados são recebidos pela entidade AAL5 através do conjunto de sinais da família `AAL5_Unitdata_Req_B` e enviados à entidade IPv6_A através do conjunto de sinais da família `AAL5_Unitdata_Ind_A`.

A forma de comunicação entre duas entidades VHDL de uma mesma estação (neste caso entre as entidades IPv6_A e AAL5 e entre as entidades IPv6_B e AAL5) difere da comunicação entre dois módulos Estelle. Em Estelle, a comunicação entre dois módulos se faz através de um *canal* entre dois pontos de interação. Este canal permite que os dois módulos troquem mensagens em ambas direções. Entretanto, em VHDL, a única forma de comunicação permitida entre entidades é feita através de sinais, que são unidirecionais. Por isso, foi necessário a criação dos quatro conjuntos de sinais na entidade AAL5, dois conjuntos para a comunicação com cada entidade IPv6 (`AAL5_Unitdata_Req_A` e `AAL5_Unitdata_Ind_A` na comunicação com a entidade IPv6_A e `AAL5_Unitdata_Req_B` e `AAL5_Unitdata_Ind_B` na comunicação com a entidade IPv6_B). Para possibilitar a simulação da comunicação entre as entidades IPv6 é necessária a intermediação da entidade AAL5, devido ao encapsulamento LLC/SNAP que deve ser obedecido pelo protocolo IPv6 no envio de dados ao protocolo AAL5. Este encapsulamento se faz necessário somente quando uma entidade IPv6 deseja enviar dados à entidade AAL5. Tal encapsulamento não é repassado para a entidade receptora IPv6. Isto resulta em sinais adicionais relativos ao envio de datagramas que não possuem correspondência aos sinais de entrada da entidade receptora IPv6. Por este motivo, foi necessária a inclusão da entidade AAL5 na comunicação entre duas entidades IPv6.

Após o mapeamento da especificação em Estelle do protocolo IPv6 para uma descrição comportamental em VHDL foram realizadas simulações que incluíram troca de mensagens entre entidades IPv6. Tais simulações enfocaram principalmente a transferência de datagramas IPv6 através de uma entidade AAL5, obedecendo-se o encapsulamento LLC/SNAP, realizadas com o auxílio da ferramenta de simulação existente no Synopsys, o *vhdldbx*. Desta maneira, mostrou-se que a descrição comportamental em VHDL do protocolo está de acordo com a especificação funcional do protocolo e que o mapeamento Estelle-VHDL foi corretamente efetuado.

## 4.2 O DLS Aplicado ao Protocolo IPv6

A descrição comportamental do protocolo IPv6 corretamente simulada envolve a transferência de datagramas completos e fragmentados entre duas entidades pares (`IPv6_A` e `IPv6_B`). Entretanto, visando um primeiro estudo da complexidade de *hardware* gerado por esta descrição, este trabalho irá considerar apenas a transferência de datagramas completos entre as entidades pares durante os passos restantes da metodologia empregada, indo desde a obtenção de uma descrição sem dependências do protocolo IPv6 até a realização da síntese lógica desta descrição pela ferramenta Synopsys. Com este propósito, uma nova descrição comportamental em VHDL do protocolo IPv6 foi descrita, sem os procedimentos relativos ao algoritmo de fragmentação, sendo corretamente simulada com o auxílio da ferramenta de simulação *vhdldbx*.

Apesar do protocolo AAL5 suportar Unidades de Dados de Serviço (AAL_SDU) com tamanho de até 64k octetos, foi definido o valor de 9180 octetos como a MTU *default* para a emissão de datagramas IP em redes ATM. A fim de possibilitar a síntese lógica de uma descrição sem dependências do protocolo IPv6 que contivesse *buffers* com tamanhos de 9180 octetos, adotou-se a partição funcional do sistema mostrada na figura 8. A separação dos *buffers* do protocolo IPv6 em uma outra entidade (*entity*) requer a inclusão de novos sinais para a comunicação entre as entidades envolvidas, bem como novos mecanismos de sincronização, de forma a tornar a descrição comportamental do protocolo IPv6 funcionalmente equivalente à descrição usada na simulação. Vale lembrar que esta descrição usada na simulação envolve apenas a transferência de datagramas IPv6 completos.

O objetivo da partição funcional é evitar que a ferramenta de síntese lógica considere o *buffer* como uma máquina de estados. O Synopsys converte a descrição VHDL original em uma descrição lógica interna a ser otimizada. No caso de um *buffer* o número de

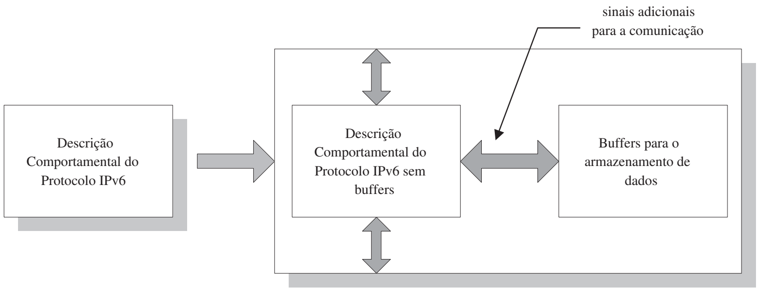



Figura 8: Divisão em duas entidades da descrição do protocolo IPv6.

variáveis torna-se impraticável.

O algoritmo de *scheduling* DLS foi utilizado após a obtenção da descrição comportamental do protocolo IPv6 corretamente simulada envolvendo a transferência de datagramas IPv6 completos. O *scheduling* é o passo principal no processo de síntese de alto nível (HLS). A utilização deste algoritmo resultou em uma descrição sem dependências do protocolo passível de ser sintetizável pela ferramenta Synopsys [7], ferramenta adequada ao projeto de circuitos VLSI.

Conforme visto na seção 2.1, a principal representação usada pelo DLS é um grafo de fluxo de controle (CFG - *Control-Flow Graph*). Este grafo fornece um meio conveniente de representar as propriedades inerentes de circuitos caracterizados por fluxo de controle, que são descritos por *loops* aninhados e por construções de sincronismo tais como a construção *wait*. Este CFG contém nós, representando as operações da descrição original, e ramos, indicando os fluxos de sinal da descrição original. Vale lembrar que a mais importante característica deste CFG é a introdução do nó *wait*, significando sempre uma mudança de estado que será efetuada. Isto causa a interrupção incondicional da geração de caminhos.

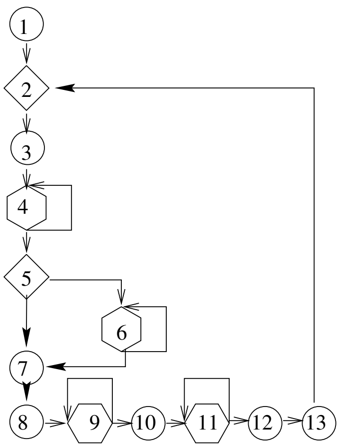
A tradução da descrição comportamental do protocolo IPv6 para seu CFG foi feita de acordo com o método descrito em [9]. Como exemplo para ilustrar este método, as figuras 9(a) e 9(b) mostram a representação em VHDL do procedimento `EnviarmsgIPv6aal5` e seu respectivo CFG. Este procedimento é executado no envio dos dados de uma entidade IPv6 para a entidade AAL5.

O procedimento `EnviarmsgIPv6aal5` contém treze operações, indicadas por números entre parênteses, quatro construções *wait*, duas construções de fluxo de controle (*if* e *while*) e sete expressões. Cada construção *wait* (os nós 4, 6, 9 e 11) contém um *loop* implícito representado por um fluxo que sai deste nó e entra novamente neste nó. Este fluxo é seguido quando a expressão booleana da construção *wait* for falsa.

Partindo-se da representação do CFG do protocolo IPv6, foram realizados os dois passos seguintes do algoritmo DLS para se chegar à descrição sem dependências do protocolo: a geração de caminhos e a geração de estados e transições. Durante a geração de caminhos, o primeiro caminho inicia-se com o primeiro nó do CFG. Os nós sucessivos são seqüencialmente adicionados ao caminho até que uma das condições mostradas anteriormente ($C_1$ e $C_2$) seja satisfeita. Na fase de geração de estados e transições, cada caminho gerado irá constituir uma transição. As transições em conjunto com os estados gerados irão produzir uma máquina finita de estados (FSM). Transições adicionais são geradas devido aos *loops* implícitos de todas as construções *wait*. Todos os caminhos iniciados com o mesmo nó serão colocados juntos no mesmo estado.

```
Procedure EnviarmsgIPv6aal5 is
begin
  i := 0; (1)
  enviandomsgaal5 : WHILE (i < sdu_usu) LOOP (2)
  ipv6_envia_aal5_mem_cod <= '1'; (3)
  WAIT UNTIL mem_autorizando = '1'; (4)
  IF (varind_ackout1 /= '0') THEN (5)
    WAIT UNTIL varind_ackout1 = '0'; (6)
  END IF;
  AAL5_Unitdata_Req_dados <= ipv6_envia_aal5_mem; (7)
  ipv6_envia_aal5_mem_cod <= '0'; (8)
  WAIT UNTIL mem_autorizando = '0'; (9)
  varind_validout1 <= '1'; (10)
  WAIT UNTIL (varind_ackout1 = '1'); (11)
  varind_validout1 <= '0'; (12)
  i := i + 1; (13)
  END LOOP enviandomsgaal5;
end EnviarmsgIPv6aal5;
```

(a)




(b)

Figura 9: Procedimento EnviarmsgIPv6aal5 (a) e seu respectivo CFG (b).

Apesar da máquina de estados da descrição comportamental do protocolo IPv6 possuir apenas um único estado, com a utilização do algoritmo DLS este estado é "quebrado" em estados intermediários, adicionados devido às condições encontradas ($C_1$ e $C_2$) ao longo de seu CFG. A descrição comportamental do protocolo IPv6, envolvendo apenas a transferência de datagramas completos, resultou numa descrição sem dependências de dados e de controle, com 21 estados.

Após a obtenção da descrição sem dependências do protocolo IPv6, foram realizadas as mesmas simulações, envolvendo troca de mensagens entre as entidades IPv6 e AAL5, que foram realizadas para a descrição comportamental em VHDL do protocolo IPv6. Novamente utilizou-se a ferramenta *vhdldbx* com o intuito de se verificar, agora, a validade dessa descrição VHDL sem dependências.

## 4.3 A Síntese Lógica do Protocolo IPv6

A descrição sem dependências do protocolo IPv6 foi simulada com a mesma seqüência de troca de mensagens das descrições comportamentais, ou seja, troca de mensagens entre as entidades IPv6 e AAL5. Conseqüentemente, a descrição sem dependências foi validada. Após o término da validação da descrição sem dependências do protocolo IPv6 é realizada a síntese lógica. A figura 10(a) apresenta o esquema lógico gerado pelo Synopsys da descrição sem dependências do protocolo IPv6 sem a existência dos *buffers*. A figura 10(b) detalha uma pequena parte do esquema lógico gerado. Para a descrição sem dependências do protocolo IPv6 envolvendo apenas a transferência de datagramas completos, obteve-se uma estimativa de área de 12,76 $mm^2$. Este valor foi obtido utilizando uma biblioteca de células padrão (*standard cells*) ES2 na tecnologia 0.7 $\mu$m.

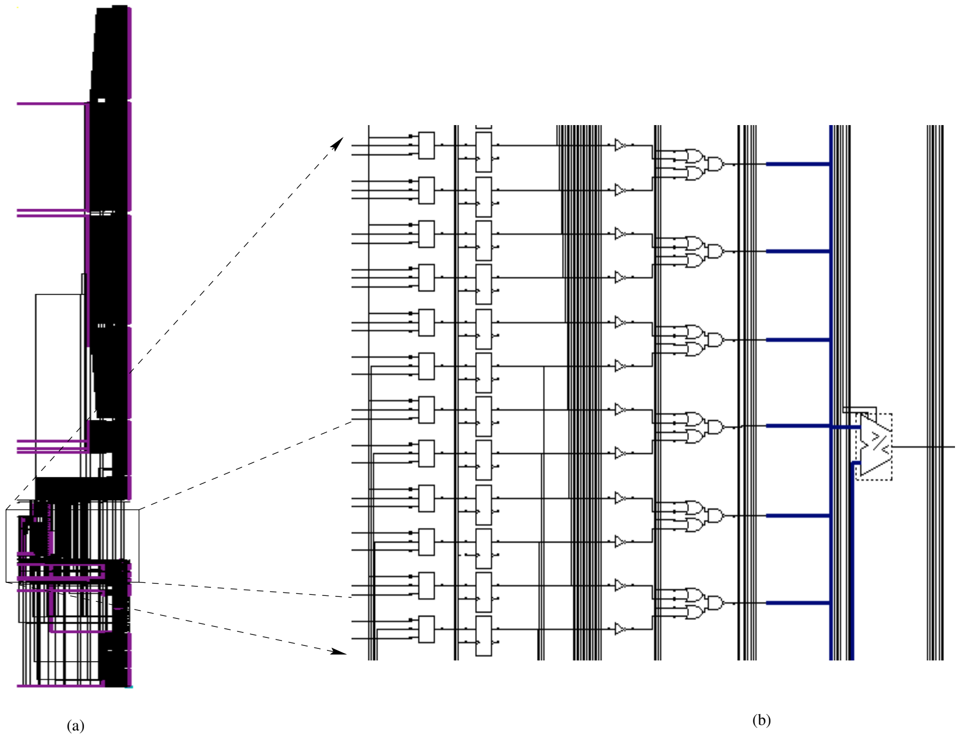



Figura 10: Esquema lógico do protocolo IPv6(a) e um detalhe do circuito sintetizado(b).

# 5 Conclusão

Neste trabalho foi apresentada uma metodologia para a implementação em *hardware*, de maneira semi-automática, de protocolos destinados a sistemas de comunicação de alta velocidade. Esta metodologia é baseada no uso do mapeamento Estelle/VHDL [3] e no uso da ferramenta de síntese lógica Synopsys [7]. Esta metodologia foi aplicada na nova versão do protocolo IP, oficialmente chamado de IPv6 dentro da abordagem “IP over ATM”. Tal escolha foi estimulada pelo fato deste protocolo estar relacionado com as mudanças que se fazem necessárias para se atingir os novos requisitos de desempenho impostos pelas novas aplicações.

Com este trabalho, mostrou-se que é possível a implementação em *hardware* de um protocolo de comunicação a partir de uma descrição formal de alto nível. Como um primeiro estudo, este trabalho optou por especificar em Estelle apenas um subgrupo do protocolo IPv6, de modo a se analisar a complexidade do *hardware* gerado ao final da metodologia empregada.

Os resultados obtidos mostraram que a maior limitação envolvida no projeto é o tamanho dos *buffers* para o armazenamento de dados que no caso do protocolo IPv6 chega a 9180 octetos. Para isto, foi adotada uma partição funcional do projeto onde a memória é implementada em separado da parte de controle. A síntese lógica do controle é efetuada pelo Synopsys mostrando que a metodologia proposta pode ser implementada sem o recurso a ferramentas de síntese de alto nível. Isto porque protocolos podem em geral ser modelados de forma eficiente por máquinas de estados finitas, dispensando a descrição RTL na síntese lógica.

# Referências


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

[2] IEEE Std 1076-1987, *IEEE Standard VHDL Language Reference Manual*, março 1988.

[3] L. Pirmez, A. Pedroza e A. C. Mesquita, "High level synthesis of protocols described by a formal description technique", in *IX IFIP International Conference on VLSI'97*, Gramado, RS, Brazil, 1997.

[4] Cruz, R., "*Um Compilador para a Linguagem Estelle*", Tese de Mestrado, Universidade Federal do Rio de Janeiro, PEE/COPPE/UFRJ, Rio de Janeiro, Brasil, 1991.

[5] Soares da Silva, G., "*Geração Automática de Sequências de Testes para Protocolos de Comunicação*", Tese de Mestrado, Universidade Federal do Rio de Janeiro, PEE/COPPE/UFRJ, Rio de Janeiro, Brasil, 1992.

[6] Resende Loureiro, F., "*Um Simulador para a Linguagem Estelle*", Tese de Mestrado, Universidade Federal do Rio de Janeiro, PEE/COPPE/UFRJ, Rio de Janeiro, Brasil, 1995.

[7] Synopsys, Inc., *Synopsys Online Documentation, v1997.01*, 1997.

[8] Cadence Design Systems, Inc., *Cadence Online Library*, 1994.

[9] O'Brien, K., Rahmouni, M., Jerraya, A. A., "DLS: A Scheduling Algorithm for High-Level Synthesis in VHDL", in *EDAC'93*, Paris, Franca, 1993.

[10] Gajski, Daniel D. et al., *High-Level Synthesis - Introduction to Chip and System Design.* University of California, Irvine: Kluver Academic Publishers, 1992.

[11] Camposano, R., Bergamaschi, R. A., "Synthesis Using Path-Based Scheduling: Algoritms And Exercises", in *27th DAC*, Orlando, pp. 450–455, Jun 1990.

[12] Stallings, W., "IPv6: The New Internet Protocol", *IEEE Communications Magazine*, Jul 1996.

[13] Chao, H. J., Ghosal, D. et al., "IP over ATM Local Area Networks", *IEEE Communications Magazine*, Ago 1994.

[14] Biagioni, E., Cooper, E. et al., "Designing a Pratical ATM LAN", *IEEE Network Magazine*, Mar 1993.

[15] Atkinson, R., *Default IP MTU for use over ATM AAL5.* RFC 1626, Naval Research Laboratory, Mai 1994.

[16] Perez, M., Liaw, F. et al., *ATM Signaling Support for IP over ATM.* RFC 1755, ISI, FORE System, Inc., Motorola Codex, Ascom Timeplex, Inc., Fev 1995.