Título do Artigo: Síntese de Alto Nível de Protocolos para a Abordagem IP sobre ATM

Responsável: Ricardo Neimo Belem Lima
E-mail: belem@gta.ufrj.br

Síntese de Alto Nível de Protocolos para a Abordagem IP sobre ATM

Ricardo Neimo Belem Lima
belem@gta.ufrj.br

Emerson Carli
emerson@gta.ufrj.br

Aloysio C. P. Pedroza
aloysio@gta.ufrj.br

Luci Pirmez
luci@nce.ufrj.br

Antônio C. de Mesquita
mesquita@lpc.ufrj.br

Grupo de Teleinformática e Automação (GTA)
Universidade Federal do Rio de Janeiro
COPPE/EE - Programa de Engenharia Elétrica/Departamento de Eletrônica
Caixa Postal 68504 - CEP 21945-970 - Rio de Janeiro - RJ - Brasil
FAX: + 55 21 2906626

Resumo

A abordagem IP sobre ATM surgiu como uma possibilidade de aumento no desempenho e na eficiência das redes de computadores tradicionais. Entretanto, os protocolos utilizados continuam sendo os mesmos da época em que haviam baixas taxas de transmissão no meio. Neste trabalho é proposta a implementação em *hardware* de dois protocolos envolvidos nesta abordagem, IPv6 e AAL5, como uma solução de contornar a limitação de velocidade no processamento destes protocolos. O ponto de partida desta implementação é uma descrição comportamental dos protocolos em VHDL, obtendo-se em seguida através da síntese de alto nível uma descrição estrutural implementável em um circuito integrado VLSI.

Abstract

The IP over ATM approach has emerged as a possibility to increase the performance and the efficiency of traditional computer networks. However, the protocols involved are the same ones used in the past. In this paper the hardware implementation of two protocols involved in this approach, IPv6 and AAL5, is proposed in order to overcome the data processing bottleneck imposed by these protocols. The starting point of this implementation is a VHDL behavioural description of these protocols, which is translated by high-level synthesis to a structural description suitable to be mapped in a VLSI integrated circuit.

1 Introdução

A Internet pode ser vista como uma rede virtual construída pela interconexão de redes heterogêneas (locais, metropolitanas e de longa distância). A arquitetura Internet, constituída pela família de protocolos TCP/IP, é baseada principalmente em um serviço de transporte orientado a conexão, fornecido pelo *Transmission Control Protocol* (TCP) [1], e em um serviço de rede não orientado a conexão (datagrama não confiável), fornecido pelo *Internet Protocol* (IP) [2].

Atualmente, as redes locais (LANs) começam a se mostrar limitadas em termos de desempenho devido ao crescente número de estações interconectadas e ao surgimento de novas aplicações (as chamadas aplicações multimídias) que exigem grande banda passante.

Uma resposta à tentativa de implementação de redes de alta velocidade capaz de fornecer uma alta banda passante e de suportar, de maneira integrada, todos os tipos de tráfegos (voz, vídeo e dados) é o modo de transferência assíncrono (ATM). Esta tecnologia possui as seguintes vantagens: maior flexibilidade, independência da taxa de transmissão e comutação de todos os tráfegos em uma estrutura única.

A simples implantação de novas LANs baseadas na tecnologia ATM mostra-se, no entanto, inviável dada a enorme base instalada. Os custos de mudança para estas novas redes integralmente baseadas na tecnologia ATM seriam muito grandes, o que inviabiliza a introdução da nova tecnologia a curto prazo. Como conseqüência, é necessário uma migração suave para as redes ATM, com soluções intermediárias, permitindo que as aplicações existentes possam continuar sendo utilizadas.

Existem basicamente dois caminhos possíveis na implementação da emulação de LANs sobre redes ATM. Um deles consiste em se implementar o protocolo IP diretamente sobre a tecnologia ATM, sendo comumente chamado de "IP over ATM". Este é o caminho que vem sendo adotado pelo grupo do IETF (*Internet Engineering Task Force*) responsável pela RFC 1577 [3] e que será estudado neste trabalho, dada sua popularidade [4, 5, 6]. O outro caminho resulta da solução proposta pelo ATM Forum, denominado "LAN Emulation" (LANE) [7], que consiste na implementação de uma camada emulando os serviços da subcamada MAC (*Medium Access Control*) das LANs tradicionais sobre a camada de adaptação ATM (AAL - *ATM Adaptation Layer*).

Mesmo alcançando uma alta banda passante e, por conseguinte, viabilizando as aplicações multimídias, as propostas descritas acima não levaram em conta que os protocolos continuam sendo os mesmos da época em que haviam baixas taxas de transmissão. Assim, o ponto crucial na obtenção de um sistema de comunicação que permita uma alta vazão passou a ser a quantidade de processamento necessária para executar as funções inerentes à comunicação e não mais a taxa de transmissão no meio.

Uma possível solução para contornar o gargalo imposto pelo processamento dos protocolos é implementá-los em hardware. Objetivando atender a implementação em hardware de protocolos de alta velocidade, este trabalho utiliza uma linguagem de descrição de hardware denominada VHDL (*Very high speed integrated circuits Hardware Description Language*) [8, 9], linguagem padrão desenvolvida pelo Departamento de Defesa Americano (DoD) para a descrição e a simulação de estruturas de hardware em diferentes níveis de abstração e domínios de descrição, indo do ***comportamental*** ao ***estrutural***.

A descrição comportamental permite a especificação das funções executadas por um projeto, sem se referir à sua estrutura interna, utilizando estruturas familiares da linguagem de programação. No domínio estrutural, as estruturas que implementam um dado comportamento são descritos, bem como sua decomposição em subprojetos e a interligação entre eles.

Usualmente, o projeto de um protocolo em hardware VLSI parte de uma descrição comportamental pois esta descrição tem a vantagem de ser semelhante à especificação funcional do protocolo. Contudo, sua descrição estrutural também é de suma importância pois é utilizada como entrada pela maioria das ferramentas de síntese lógica, tal como o Synopsys [10].

Portanto, partindo de uma descrição comportamental, há a necessidade da tradução desta descrição em uma descrição estrutural. A essa tradução dá-se o nome de síntese de alto nível (HLS - *High Level Synthesis*).

Este trabalho visa a implementação em hardware VLSI de dois protocolos pertencentes à abordagem IP sobre ATM [11]: o protocolo IPv6 e o protocolo AAL5. A metodologia

utilizada neste trabalho é composta dos seguintes passos. Primeiramente, descreve-se de maneira comportamental em VHDL ambos os protocolos. Em seguida, realiza-se a síntese de alto nível, utilizando o algoritmo DLS (*Dynamic Loop Scheduling*) para obter suas respectivas descrições estruturais. De posse de tais descrições, executa-se a síntese lógica utilizando o software Synopsys e, por fim, a geração do *layout* físico destes protocolos são alcançados com a utilização da ferramenta Opus [12].

Este artigo está estruturado da seguinte forma. A seção 2 aborda as principais características de ambos os protocolos bem como as características da estratégia que os envolve (IP sobre ATM). A seção 3 apresenta as diversas etapas tomadas durante a implementação em hardware dos protocolos. Na seção 4 são analisados os resultados obtidos. As considerações finais encontram-se na seção 5.

2 Os Protocolos IPv6 e AAL5 e a Abordagem IP sobre ATM

2.1 O Protocolo IPv6

Com a mudança natural da Internet, 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** [13, 14, 15]. Resumidamente, as mudanças introduzidas para o IPv6 podem ser vistas sob 3 aspectos:

- *endereços maiores.* O novo tamanho destinado aos campos de endereçamento do datagrama é a mudança mais perceptível. O protocolo IPv6 quadruplicou o tamanho do endereçamento do IPv4 de 32 bits para 128bits;
- *cabeçalho de formato flexível.* O protocolo IPv6 utiliza 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 não oferece nenhuma assitência nessa área. 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;
- 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 possui a versão correta;
- concatena o datagrama se necessário;
- passa a mensagem encapsulada para o protocolo de transporte apropriado. Neste caso, é considerado apenas o protocolo TCP.

2.2 O Protocolo AAL5

O aparecimento da tecnologia ATM (*Asynchronous Transfer Mode*) constitui uma possibilidade de crescimento do desempenho atingido pelas redes locais, tornando possível a transmissão eficiente de informação multimídia como voz e vídeo. Assim, o ATM proporciona uma melhoria nas aplicações já existentes com garantia de qualidade de serviço (QoS - *Quality of Service*). O ATM é, por natureza própria, orientado a conexão. A transmissão de dados ocorre em canais virtuais, estabelecidos previamente.

Dentre as três camadas presentes nesta tecnologia - Física, ATM e AAL - a camada AAL é a responsável em adaptar toda a atual arquitetura de mais alto nível a este novo modelo. Ela tem como função efetuar as adaptações necessárias para compatibilizar e oferecer os serviços desejados pelas camadas superiores, utilizando tecnologia ATM.

Existem várias classes de serviços suportados por esta tecnologia e cada uma é provida de uma camada de adaptação diferente. Isto quer dizer que o seu propósito é acrescentar funcionalidades para cada camada de adaptação ATM, de modo a diferenciar as classes de serviços exigidas pelas camadas superiores.

Dentre essas classes de serviço e suas respectivas camadas de adaptação, a AAL5 [16] é uma das que foram elaboradas para serem usadas em serviços de dados orientados a conexão ou não. A transferência de dados é feita em operação assegurada ou não assegurada, suportando deteção de erros sem fazer recuperação de pacotes perdidos. Sua elaboração teve o propósito de reduzir o processamento de protocolo e o *overhead* de transmissão, diminuindo significativamente a complexidade de implementação, o que vem de encontro com as novas características que os protocolos de redes de alta velocidade atuais devem possuir.

O protocolo AAL5 é composto por duas subcamadas: a subcamada **SAR** (*Segmentation and Reassembly sublayer*) e a subcamada **CPCS** (*Common Part Convergence Sublayer*). A subcamada CPCS adiciona cabeçalhos aos quadros do usuário e assegura integridade no nível de quadros. A subcamada SAR é responsável pela fragmentação da informação de modo a poder se acomodar nas células ATM e pela remontagem da informação a partir das células recebidas.

As funções do protocolo AAL5 que serão implementadas neste trabalho são:

- adição de uma cauda (*trailer*) a fim de assegurar integridade na transmissão;
- segmentação da informação;
- concatenação da informação;
- verificação da validade da informação a partir da ánalise da cauda;
- entrega da informação correta ao usuário.

2.3 A Abordagem IP sobre ATM

Na abordagem IP sobre ATM é possível que, protocolos do tipo IP, utilizados nas redes locais e metropolitanas, sejam mapeados diretamente em conexões ATM, permitindo a interconexão dessas redes.

Nessa configuração, o mecanismo de resolução de endereços deve traduzir diretamente o endereço IP para o endereço ATM e vice-versa, o que fornece a ilusão da existência de sub-redes lógicas IP (LIS - *Logical IP Subnetworks*) [17], ou seja, 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.

As estações existentes numa LIS estabelecem canais virtuais entre si para enviar datagramas IP. Pelo fato de cada LIS ser, de maneira conceitual, uma rede LAN, o protocolo IP aplica as regras tradicionais a cada uma delas. Como exemplo, todas as estações de uma mesma LIS compartilham um único prefixo do endereço IP (*netid*) e este prefixo difere dos prefixos usados pelas outras sub-redes LIS. Além disso, uma estação de uma LIS não pode se comunicar diretamente com uma estação de uma outra LIS. Ao invés disso, toda a comunicação entre sub-redes LIS deve ser realizada através de um roteador pertencente a várias sub-redes LIS.

Duas AALs foram definidas para protocolos que trabalham com pacotes como é o caso do IP: 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" [18], que define o valor de 9.180 octetos como a MTU do protocolo IP.

Uma característica importante numa sub-rede LIS é o formato de encapsulamento que deve ser obedecido pelo pacote IP. A figura 1 mostra o encapsulamento dos pacotes IPv6 para formar uma AAL_SDU, conhecido como encapsulamento LLC/SNAP.

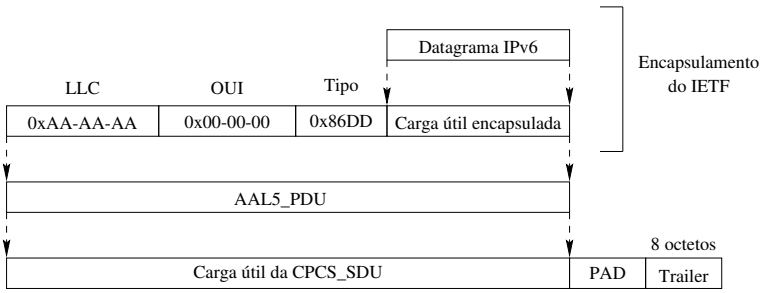


Figura 1: Encapsulamento LLC/SNAP dos pacotes IPv6.

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 IP. Quando uma estação (A) deseja enviar um datagrama para outra estação (B), o protocolo IP em (A) executa o algoritmo de roteamento tradicional para determinar o endereço apropriado do próximo nó (N) para o qual o datagrama será enviado. De posse deste endereço, a estação (A) o passa juntamente com o datagrama IP 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 [19] para estabelecer o canal virtual necessário. Para tanto, a

estação (A) entra em contato com um servidor existente na sua LIS de modo a traduzir o endereço IP de (N) no seu respectivo endereço ATM. Após a criação do canal virtual, a estação (A) o adiciona à sua tabela.

Neste trabalho, durante a implementação em *hardware* de ambos os protocolos envolvidos nesta abordagem (IPv6 e AAL5), 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.

3 A Metodologia Adotada

A metodologia adotada neste trabalho para a síntese de circuitos VLSI que implementam os protocolos IPv6 e AAL5 é ilustrada pelo diagrama da figura 2. O procedimento adotado utiliza um conjunto de ferramentas de projeto de circuitos VLSI, denominado na figura de C.A.D. de Microeletrônica, cuja interface com o projetista de protocolos é uma descrição comportamental em VHDL do sistema. O C.A.D. de Microeletrônica é composto de ferramentas padrão de projeto VLSI tais como o Synopsys para a síntese lógica e o Opus para geração de *layout*.

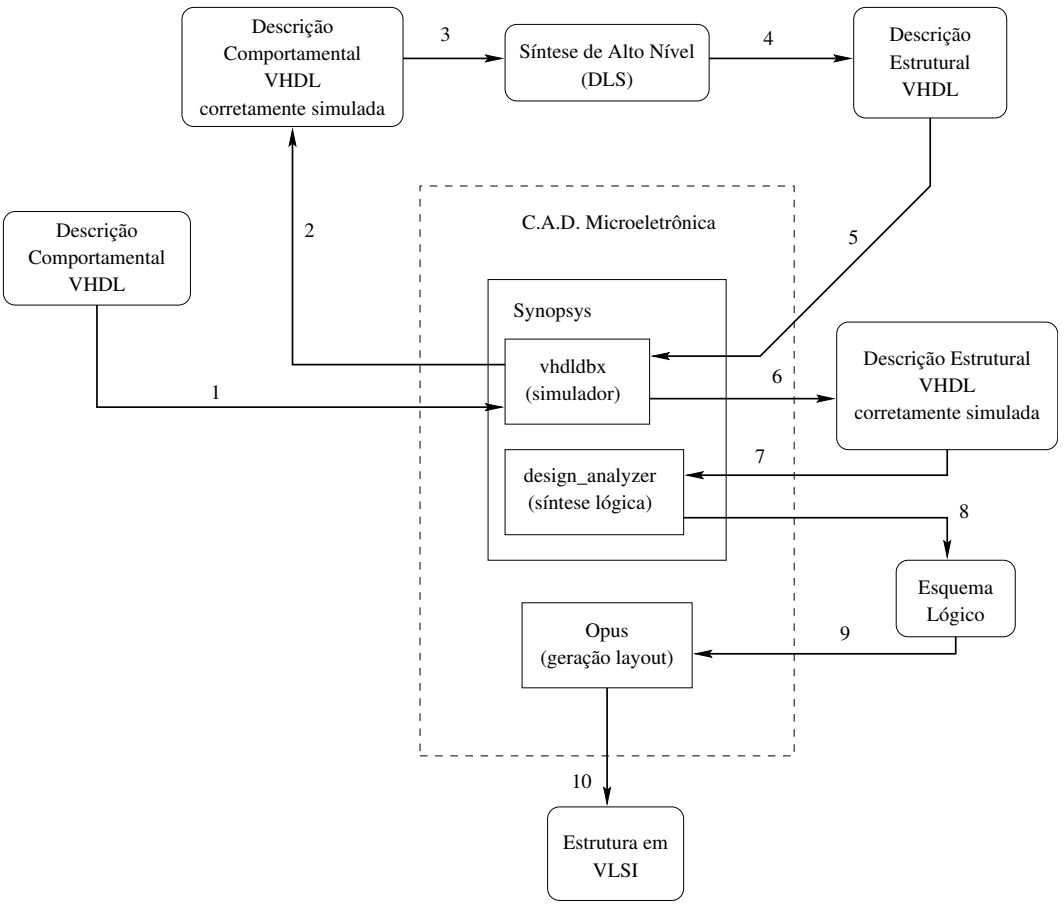


Figura 2: A abordagem adotada para a obtenção de estruturas VLSI.

O Synopsys pode ser utilizado na simulação de descrições comportamentais e estruturais em VHDL e para síntese lógica de descrições estruturais. Já o Opus é utilizado na geração de *layout* das estruturas VLSI, a partir do esquema lógico.

A primeira etapa do processo, chamada aqui de síntese da máquina de estados estendida (FSME), consiste na obtenção de uma descrição comportamental VHDL do protocolo, corretamente simulada (1,2). A síntese propriamente dita consiste na aplicação de um algoritmo de *"scheduling"* (3,4) à descrição comportamental de forma a produzir uma descrição de máquina de estados, que já se pode considerar como uma descrição estrutural dado que a partir desta descrição já é possível a síntese lógica no Synopsys. Como algoritmo de scheduling foi utilizado o algoritmo DLS (*Dynamic Loop Scheduling*) proposto por Rahmouni [20]. Novamente faz-se uso do simulador com o intuito de se verificar, agora, a validade das descrições estruturais obtidas (5,6).

Estas descrições corretamente simuladas são então submetidas à ferramenta de síntese lógica (7,8). Os esquemas lógicos resultantes são utilizados como descrição de entrada da ferramenta Opus para a geração do *layout* destas estruturas físicas (9,10).

3.1 A Descrição Comportamental VHDL

O componente básico de uma especificação em VHDL é uma entidade. Uma entidade é composta por duas partes: uma interface (*Entity*) e um conjunto de uma ou mais arquiteturas (*architecture*). A interface define as características observáveis externamente, através das portas de entrada ou de saída, enquanto uma arquitetura define a implementação desta entidade.

A declaração *port* define todas as portas que uma entidade irá possuir. As portas são sinais que podem ser de entrada ou de saída, possibilitando que uma entidade troque informações com outras entidades VHDL.

Uma vez que uma entidade teve sua interface especificada pela declaração *entity*, sua implementação pode ser descrita por diferentes arquiteturas. Como exemplo, uma arquitetura pode simplesmente descrever o comportamento entrada/saída da entidade enquanto outras podem incluir informações estruturais específicas através de um conjunto de componentes dispostos hierarquicamente. Em outras palavras, uma arquitetura pode ser descrita tanto de maneira comportamental como de maneira estrutural.

Quando descrito estruturalmente, os sub-módulos de uma arquitetura podem ser descritos como blocos. Um bloco é uma unidade de módulo dentro da descrição estrutural, com sua própria interface, conectada a outros blocos ou portas por meio de sinais. Por outro lado, quando descrito de maneira comportamental, uma arquitetura é implementada através da instrução *process*. Um *process* é formado por um conjunto de instruções seqüenciais que é executado sempre que um sinal (pertencente ao *process*) muda de valor, ou seja, de forma concorrente.

A figura 3 ilustra, em parte, a entidade do protocolo IPv6. Deve-se notar que a arquitetura do protocolo foi descrita de maneira comportamental, pois como dito anteriormente, tal descrição permite que os projetistas de protocolos os descrevam de modo mais claro e fácil.

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 ela 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 envie uma mensagem, é necessário que antes ocorra o envio de um sinal indicando a presença de uma mensagem a ser recebida pela outra entidade comunicante. Este sinal é exemplificado pela linha de código `AAL5_Unitdata_Req_cod <= '1'`, quando uma entidade IPv6 deseja enviar uma mensagem para uma entidade AAL5. Entretanto, o simples envio deste sinal não habilita a entidade emissora a enviá-la. É necessário que a entidade receptora esteja preparada para recebê-la. Assim, a

```
entity IPv6_entity_type is
       port (start: in bit;
             clk: in bit;
             AAL5_Unitdata_Req_cod: out bit;
             AAL5_Unitdata_Req_dados: out character;
             varind_ackout1: in bit;
             varind_validout1: out bit;
             ipv6habilitado: out bit;
             aal5preparado: in bit;
             IPv6_Data_Req_cod: in bit;
             IPv6_Data_Req_sdu: in character;
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
                ipv6habilitado <= '1';
                Lermsgaal5IPv6;
                IPv6_Data_Ind_cod <= '1';
                WAIT UNTIL testpreparado = '1';
                EnviarmsgIPv6User;
               ENDIF;
               WAIT UNTIL (clk = '1');
               IPv6_Data_Req_cod <= '0';
               END LOOP MEF;
           end process;
end IPv6_entity_body;
```

Figura 3: Parte do código VHDL do protocolo IPv6.

```
procedure Lermsgaal5IPv6 is
      begin
          i := 0;
          lendomsgaal5: WHILE (i < aal5_pdu) LOOP
          WAIT UNTIL (ready_validin2 = '1');
          buffer_aal5ipv6_pdu(i) := AAL5_Unitdata_Ind_dados;
          ready_ackin2 <= '1';
          WAIT UNTIL (ready_validin2 = '0');
          ready_ackin2 <= '0';
          i := i + 1;
          END LOOP lendomsgaal5;
      end Lermsgaal5IPv6;
```

Figura 4: O procedimento Lermsgaal5IPv6.

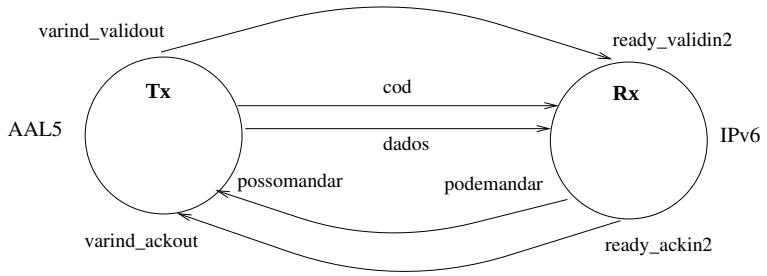


Figura 5: Diagrama simbólico do mecanismo de sincronização.

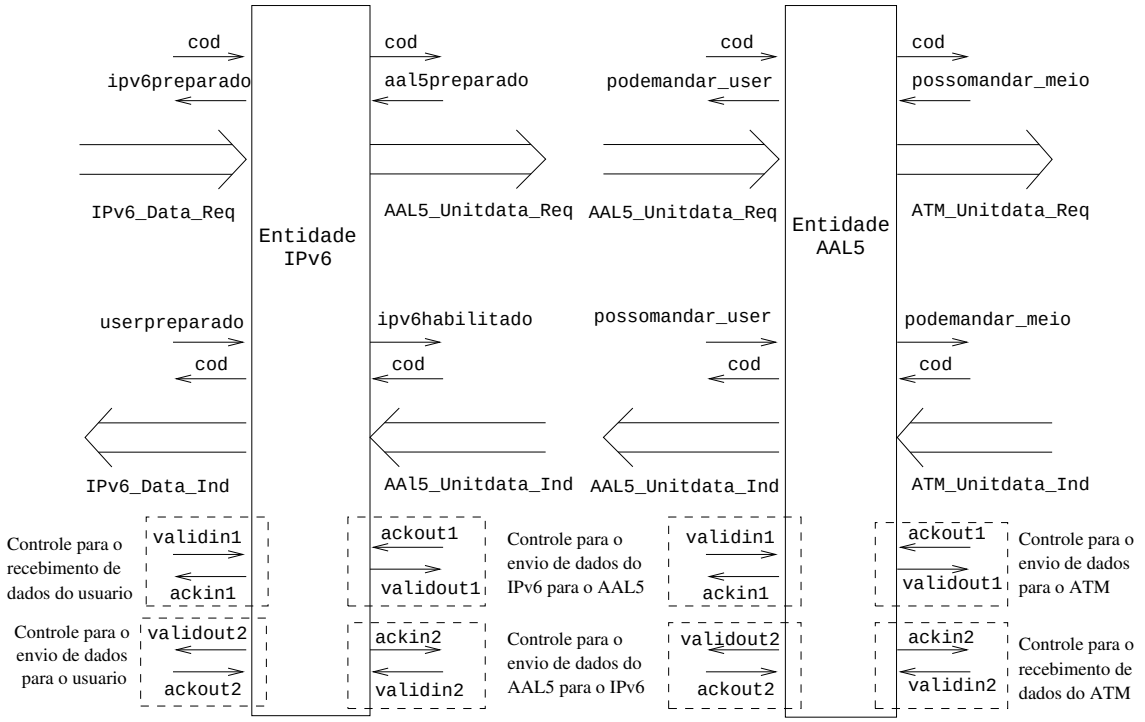


Figura 6: Diagrama em blocos referente à comunicação entre os protocolos IPv6 e AAL5.

entidade emissora suspende seu processo até receber um sinal de confirmação. Este sinal é exemplificado pela linha de código `WAIT UNTIL aal5preparado = '1'`. Vale lembrar que este sincronismo acontece em ambos os sentidos de comunicação entre duas entidades.

Deve ficar claro que o mecanismo de sincronização descrito acima realiza-se a nível de mensagem. No caso de protocolos, uma mensagem é descrita pela sua primitiva de serviço. Uma primitiva de serviço pode ter vários parâmetros. Dentre eles, o parâmetro de dados merece uma atenção especial, visto que este parâmetro é de tamanho variável. Como conseqüência, um sincronismo de envio e recepção de dados entre entidades, análogo ao sincronismo de mensagens, deve ser estabelecido.

Tal sincronismo pode ser exemplificado pelo procedimento `Lermsgaal5IPv6`, relativo ao envio de dados do protocolo AAL5 para o protocolo IPv6 que é mostrado na figura 4. Este mecanismo de sincronização utilizado na troca de dados pode ser melhor visualizado na figura 5. Antes da entidade IPv6 receber o primeiro dado, ela habilita a entidade AAL5 através do sinal ‘‘`podemandar`’’ e suspende seu processo através do código `WAIT UNTIL ready_validin2 = '1'`. O AAL5 então envia o primeiro dado e o sinal `varind_validout` e suspende seu processo. Este sinal reabilita o IPv6 a continuar o processo, efetuando

o tratamento do dado, enviando o sinal `ready_ackin2` e novamente suspendendo o seu processo. Este procedimento se repete até que o último dado seja enviado.

Numa visão global, a figura 6 mostra o diagrama em blocos dos protocolos IPv6 e AAL5, com os sinais de controle necessários para efetuar a comunicação entre eles.

3.2 A Síntese de Alto Nível

Para a síntese de alto nível, este trabalho utiliza um algoritmo de escalonamento (*scheduling*) chamado **DLS** [21]. 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"*) [22, 23] 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;
- 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* sempre significa 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 [21].

Este grafo pode ser definido como

$$G = (N, F), \tag{1}$$

onde N é o conjunto de nós e F é o conjunto de ramos definindo a precedência entre os nós. O conjunto de nós é definido por:

$$N = N_w \cup N_f \cup N_e, \tag{2}$$

onde N_w é o conjunto de construções *wait*, N_f o conjunto de construções de fluxo de controle e N_e o conjunto de expressões.

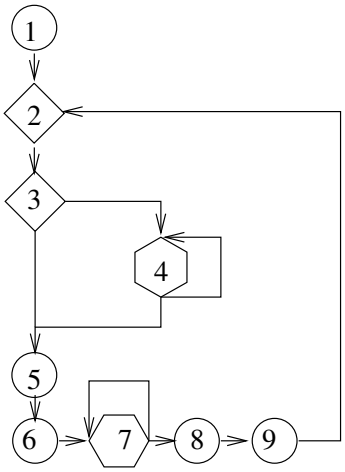
Um caminho C = (n_1, n_2,...., n_i)[n_s] é definido como um conjunto de nós, começando pelo nó n_1 e tendo como seu sucessor um outro caminho iniciado por n_s. Novos caminhos são gerados conforme se progride no CFG começando-se pelo primeiro nó. A geração de um caminho é interrompida e um novo caminho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

A tradução de descrições comportamentais em VHDL para o CFG é feita de forma simples. As figuras 7(a) e 7(b) mostram a representação em VHDL do procedimento `EnviarmsgIPv6aal5` e seu respectivo CFG. O procedimento contém nove operações, indicadas por números em parênteses, duas construções *wait*, duas construções de fluxo de controle (*if* e *while*) e cinco expressões.

```
Procedure EnviarmsgIPv6aal5 is
begin
   i := 0  (1);
   enviandomsgaal5: WHILE (i<sdu_usu) LOOP (2)
   IF (varind_ackout1 /= '0') THEN (3)
     WAIT UNTIL varind_ackout1 = '0'; (4)
   END IF;
   AAL5_Unitdata_Req_dados <= buffer_usuipv6_sdu(i); (5)
   varind_validout1 <= '1'; (6)
   WAIT UNTIL (varind_ackout1 = '1'); (7)
   varind_validout1 <= '0'; (8)
   i := i + 1; (9)
   END LOOP enviandomsgaal5;
end Enviarmsgaal5IPv6;
```

(a)



(b)

Figura 7: Procedimento EnviarmsgIPv6aal5 (a) e seu respectivo CFG (b).

A construção *wait* (os nós 4 e 7) 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 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 que produzirá uma máquina finita de estados (FSM - *Finite State Machine*). 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, conseqüentemente, uma tabela de estados e transições que pode ser manipulada pela ferramenta de síntese lógica existente no Synopsys, o *design_analyzer*.

4 Resultados

Os protocolos IPv6 e AAL5 foram descritos inicialmente de maneira comportamental em VHDL. A partir disto, foram feitas simulações num ambiente C.A.D. de Microeletrônica que incluíram troca de mensagens entre as entidades, mostrando que os protocolos estão de acordo com suas respectivas especificações funcionais.

Em seguida, utilizou-se o algoritmo DLS a fim de se obter as descrições estruturais dos respectivos protocolos. Neste caso particular, estas descrições contém uma única máquina de estados cuja tabela de estados e transições é gerada pelo algoritmo DLS. A descrição comportamental do protocolo IPv6 resultou numa máquina de estados com 13 estados.

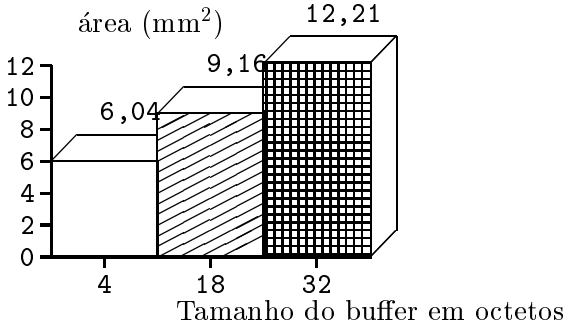


Figura 8: Tamanho do buffer × Hardware sintetizado do submódulo SAR.

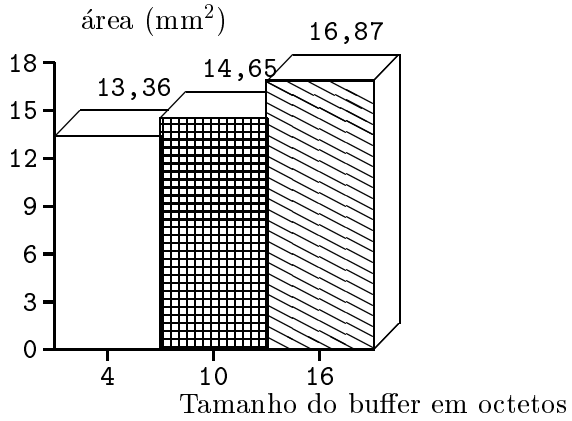


Figura 9: Tamanho do buffer × Hardware sintetizado do protocolo IPv6.

Para o protocolo AAL5 foi obtida uma máquina de estados com 29 estados.

As descrições estruturais dos protocolos foram então simuladas com a mesma seqüência de troca de mensagens das descrições comportamentais, verificando-se a validade da utilização do algoritmo.

A fim de ilustrar alguns aspectos relevantes da síntese de protocolos, foram feitas algumas experiências relativas ao armazenamento de dados no protocolo IPv6 e no submódulo SAR do protocolo AAL5, responsável pela segmentação e remontagem da informação. Estas experiências são relevantes devido à complexidade do *hardware* gerado.

Com o objetivo inicial de se analisar a viabilidade da geração do esquema lógico pelo Synopsys, sua complexidade e sua área, as primeiras experiências foram realizadas com pequenos tamanhos de buffer para o armazenamento de dados. A figura 8 ilustra alguns valores tomados para o submódulo SAR. Observa-se que partindo de um buffer com tamanho de 4 octetos obteve-se uma estimativa de área de 6,04 mm^2 e com um buffer de 32 octetos obteve-se uma estimativa de 12,21 mm^2. Estes valores foram obtidos utilizando uma biblioteca de células padrão (*standard cells*) ES2 na tecnologia 0.7 μm. Os valores para o caso do protocolo IPv6 são ilustrados na figura 9.

A título de ilustração, a figura 10 apresenta uma pequena parte do esquema lógico gerado pelo Synopsys a partir da descrição estrutural do protocolo IPv6.

No atual estágio de implementação, obteve-se o *layout* físico do submódulo SAR com buffer de armazenamento de 32 octetos.

5 Considerações Finais

Atualmente, as redes locais (LANs) começam a se mostrar limitadas em termos de desempenho devido ao crescente número de estações interconectadas e ao surgimento de

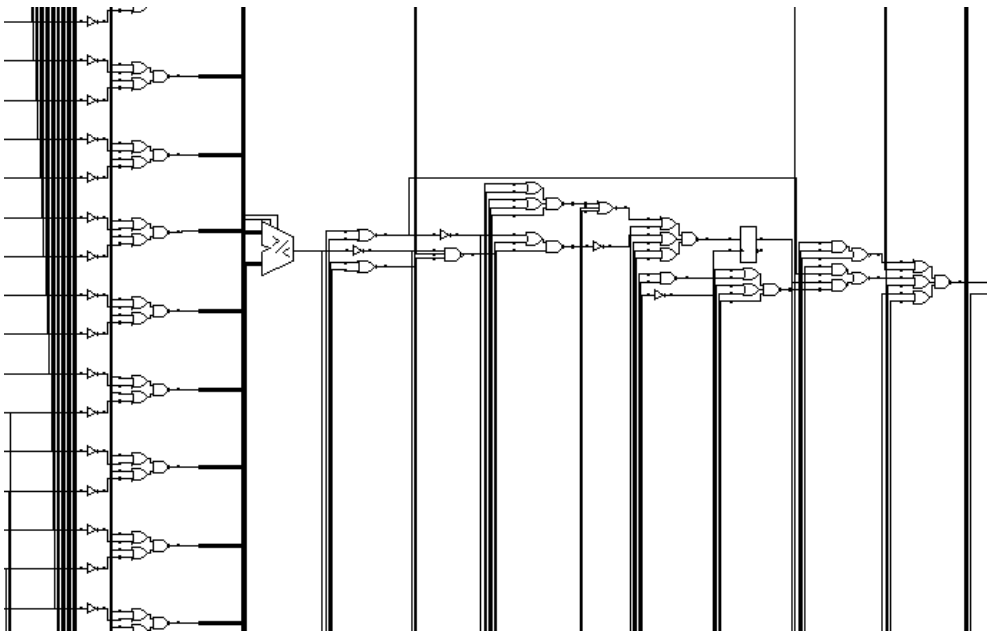


Figura 10: Parte do esquema lógico do protocolo IPv6.

novas aplicações que exigem grande banda passante.

A tecnologia ATM é capaz de fornecer uma alta banda passante e de suportar, de maneira integrada, diferentes tipos de tráfego (voz, vídeo e dados).

A simples implantação de novas LANs baseadas na tecnologia ATM mostra-se, no entanto, inviável dada a enorme base instalada. Como conseqüência, é necessário uma migração suave para as redes ATM, com soluções intermediárias, que permitam que as aplicações existentes possam continuar sendo utilizadas.

A abordagem IP sobre ATM é um dos caminhos na implementação da emulação de LANs sobre redes ATM, possibitando então, um aumento no desempenho e na eficiência das redes locais. Entretanto, os protocolos envolvidos nesta abordagem continuam sendo os mesmos da época em que haviam baixas taxas de transmissão no meio. Assim, o processamento necessário para executar as funções inerentes à comunicação é agora o ponto crucial na obtenção de um melhor desempenho nas redes.

A solução proposta neste trabalho foi a implementação dos protocolos envolvidos nesta abordagem como uma solução de contornar a limitação de velocidade no processamento destes protocolos. Neste sentido, foram implementados dois protocolos em *hardware* VLSI: o protocolo IPv6 e o protocolo AAL5.

Para se obter uma primeira estimativa do custo da implementação em termos de área versus o tamanho do buffer responsável pelo armazenamento de dados nos respectivos protocolos, foram analisados alguns resultados obtidos em experiências envolvendo diversos tamanhos de buffer. A análise dos resultados obtidos foi apresentada na seção anterior.

6 Agradecimentos

Este trabalho recebeu apoio financeiro da UFRJ, FUJB, PROTEM-CC, CNPq e CAPES.

Referências

[1] Postel, J. (ed), *Transmission Control Protocol.* RFC 793, Set 1981.

[2] Postel, J. (ed), *Internet Protocol - DARPA Internet Program Protocol Specification.* RFC 791, USC/Information Sciences Institute, Set 1981.

[3] Laubach, M., *Classical IP and ARP over ATM.* RFC 1577, Hewlett-Packard Laboratories, Jan 1994.

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

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

[6] Villela, D. A. M., Costa, L. H. M. K., *Implementando Redes Locais em ATM.* Relatório Técnico do Grupo de Teleinformática e Automação da COPPE/UFRJ, Jun 1997.

[7] The ATM Forum, LAN Emulation SWG Drafting Group, *LAN Emulation Over ATM Specification, Version 1.0*, Jan 1995.

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

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

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

[11] Carli, E., Lima, R. N. B. et al., "Implementação em Hardware de Protocolos para Redes de Alta Velocidade a partir de uma Descrição Formal", in *II Simpósio Franco-Brasileiro de Sistemas Informáticos Distribuídos, SFBSID'97*, Fortaleza, CE, pp. 211–220, Nov 1997.

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

[13] Bradner, S., Mankin, A., *The Recommendation for the IP Next Generation Protocol.* RFC 1752, Harvard University, ISI, Jan 1995.

[14] Deering, S., Hinden, R., *Internet Protocol Version 6 (IPv6) Specification.* RFC 1883, Xerox PARC, Ipsilon Networks, Dec 1995.

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

[16] ITU-T Recommendation I.363, *B-ISDN ATM adaptation layer (AAL) specification*, 1993.

[17] Comer, D. E., *Internetworking with TCP/IP*, vol. 1. New Jersey: Prentice Hall, 3 ed., 1995.

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

[19] 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.

[20] Rahmouni, M., O'Brien, K., "A Loop-Based Scheduling Scheduling Algorithm for Hardware Description Languages", *Journal of Parallel Processing Letters*, vol. 4, no. 3, no. 3, pp. 351–364, 1994.

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

[22] Camposano, R., "Path-Based Scheduling for Synthesis", *IEEE Transactions on CAD*, vol. 10, pp. 85–93, Jan 1991.

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