UNIVERSIDADE DO SUL DE SANTA CATARINA

MARCOS ROBERTO DA SILVA

IMPLEMENTAÇÃO DE UM MECANISMO TDM PARA GARANTIR PREVISIBILIDADE NA COMUNICAÇÃO DA REDE SOCIN

Palhoça

2013

MARCOS ROBERTO DA SILVA

IMPLEMENTAÇÃO DE UM MECANISMO TDM PARA GARANTIR PREVISIBILIDADE NA COMUNICAÇÃO DA REDE SOCIN

Trabalho de Conclusão de Curso apresentado ao Curso de graduação em Engenharia Elétrica – Telemática da Universidade do Sul de Santa Catarina como requisito parcial à obtenção do título de Engenheiro Eletricista.

Orientadora: Prof[a]. Sheila Santisi Travessa, M. Eng.

Co-orientador: Prof. Marcelo Daniel Berejuck, MSc.

Palhoça

2013

MARCOS ROBERTO DA SILVA

IMPLEMENTAÇÃO DE UM MECANISMO TDM PARA GARANTIR PREVISIBILIDADE NA COMUNICAÇÃO DA REDE SOCIN

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do título de Engenheiro Eletricista e aprovado em sua forma final pelo Curso de Engenharia Elétrica – Telemática da Universidade do Sul de Santa Catarina.

Palhoça, 28 de junho de 2013.

_____

Prof[a]. e orientadora Sheila Santisi Travessa, M. Eng.
Universidade do Sul de Santa Catarina

_____

Prof. Marcelo Daniel Berejuck, MSc.
Universidade do Sul de Santa Catarina

_____

Prof. Douglas Rossi de Melo, MSc.
Universidade do Vale do Itajaí

_____

Marcos Roberto da Silva, Bel.
Universidade do Vale do Itajaí

MARCOS ROBERTO DA SILVA

IMPLEMENTAÇÃO DE UM MECANISMO TDM PARA GARANTIR PREVISIBILIDADE NA COMUNICAÇÃO DA REDE SOCIN

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do título de Engenheiro Eletricista e aprovado em sua forma final pelo Curso de Engenharia Elétrica – Telemática da Universidade do Sul de Santa Catarina.

Palhoça, 28 de junho de 2013.

Prof[a]. e orientadora Sheila Santisi Travessa, M. Eng.
Universidade do Sul de Santa Catarina

Prof. Marcelo Daniel Berejuck, MSc.
Universidade do Sul de Santa Catarina

Prof. Douglas Rossi de Melo, MSc.
Universidade do Vale do Itajaí

Marcos Roberto da Silva, Bel.
Universidade do Vale do Itajaí

À minha mãe e ao meu pai, Maria e Carlos, por tudo o que fizeram e representam para mim. Com eles, tenho o grande prazer de viver a vida.

AGRADECIMENTOS

Primeiramente a Deus, por colocar em minha vida pessoas que me engrandeceram como ser humano e me proporcionaram diversos ensinamentos. Sem ajuda delas eu jamais conseguiria tamanha conquista.

Em especial quero agradecer imensamente ao Marcelo Berejuck, pois ele me propiciou grandes aprendizados e contribuiu de forma significativa no meu crescimento profissional. Serei eternamente grato a ele.

Outra grande pessoa que gostaria de agradecer é ao meu homônimo Marcos Roberto, que sempre foi solícito às minhas dúvidas, independente do horário, tendo contribuído bastante no desenvolvimento deste trabalho.

Aos meus familiares, que foram a base do meu crescimento e que sempre me ofereceram apoio em todos os momentos. Amo-os de todo o meu coração.

“Só fazemos melhor aquilo que repetidamente insistimos em melhorar. A busca da excelência não deve ser um objetivo, e sim um hábito” (Aristóteles).

RESUMO

O crescente aumento na densidade dos circuitos integrados tem permitido que os fabricantes implementem múltiplos processadores de diferentes tipos em um mesmo chip. Esta arquitetura forma sistemas conhecidos como System-On-a-Chip (SoC). Devido a alta taxa de processamento requerida nos sistemas mais complexos, a interconexão dos componentes em um SoC demanda soluções que minimizem problemas de ruído e consumo de energia. Uma solução de consenso para esse problema são as Networks-on-Chip (NoCs), que são arquiteturas de comunicação intra-chip baseadas em redes chaveadas. Entre as principais vantagens dessa abordagem, pode-se destacar o fato de que essas redes proveem paralelismo na comunicação, largura de banda escalável e modularidade de projeto.

No entanto, alguns SoCs que têm características de tempo real exigem que a comunicação entre seus componentes seja realizada com garantia de entrega no prazo estabelecido. Uma das soluções apresentada para resolver este problema é o uso de multiplexação dos canais de comunicação, conhecida como TDM (em inglês Time-Division Multiplexing).

A rede SoCIN é uma NoC que apresenta algumas soluções para garantir a qualidade de serviços. Porém, nenhuma dessas soluções consegue garantir a entrega de pacotes se algum núcleo requisitar uma conexão por tempo indeterminado.

Este trabalho propõe a implementação de um mecanismo de comunicação TDM no enlace entre os roteadores da rede SoCIN.

Palavras-chave: SoC. NoC. TDM.

ABSTRACT

The growing increase in density of integrated circuits has empowered manufacturers to implement multiple processors of different types in one single chip. This architecture is known as Systems-on-a-Chip (SoC). The use of traditional bus in SoCs with high data processing rate causes noise and power consumption issues. A new conception for cores connection based on switched networks, known as Network-on-Chip (NoCs), was developed to minimize these problems. Parallel communications, scalable bandwidth and modularity are the main advantages of the NoCs technology.

However, SoCs with real-time characteristics demands communication time between its components that meet strict deadlines. A solution presented in order to fulfill real-time systems requirements is the TDM technique (acronym of Time Division Multiplexing).

The network SoCIN (System on Chip Interconnection Network) is a NoC which uses some of these solutions to assure QoS (Quality of Service), and yet packet delivering issues are found when a core requests the communication channel for a long period.

This study proposes a solution to those problems by means of the TDM technique.

Keywords: SoC.NoC. TDM.

LISTA DE ILUSTRAÇÕES

Figura 1 – Topologia ponto-a-ponto 22
Figura 2 – Topologia Estrela 23
Figura 3 – Topologia Anel 23
Figura 4 – Topologia Barramento 24
Figura 5 – Estrutura da mensagem 26
Figura 6 – Exemplo de um roteador 27
Figura 7 – Redes diretas: (a) grelha 2-D; (b) toróide 2-D; (c) hipercubo 3-D 30
Figura 8 – Redes indiretas: (a) crossbar; (b) multiestágio 31
Figura 9 – TDM abordagem de ciclo fixo 33
Figura 10 – TDM abordagem em demanda 34
Figura 11 – Arquitetura do roteador da Æthereal 37
Figura 12 – Formato do pacote 39
Figura 13 – Canais de comunicação do roteador 40
Figura 14 – Arquitetura do roteador QNoC 41
Figura 15 – Formas de ondas de transmissão: preempção no pacote Real-Time 42
Figura 16 – Estrutura de malha Hermes 3x3 42
Figura 17 – Largura de banda compartilhada no roteador Hermes-SC 43
Figura 18 – Arquitetura do roteador Hermes-FP 45
Figura 19 – Campos dos pacotes transmitidos pela Hermes-FP 46
Figura 20 – Estrutura do roteador Hermes-DP 47
Figura 21 – Campos do pacote de controle 48
Figura 22 – Campos do pacote de dados 49
Figura 23 – Arquitetura do roteador para suportar escalonamento baseado em taxas 49
Figura 24 – Topologia em grelha 2-D 50
Figura 25 – Enlace da rede SoCIN 51
Figura 26 – Formato do pacote da rede SoCIN 51
Figura 27 – Formato do cabeçalho do pacote SoCIN: (a) HLP e RIB; (b) campos do RIB 52
Figura 28 – Interfaces do roteador ParIS 53
Figura 29 – Interfaces do roteador ParIS 53
Figura 30 – Formato do pacote modificado 54
Figura 31 – Novos módulos com canais virtuais: (a) X_{in2VC}; e (b) X_{out2VC} 55
Figura 32 – Enlace da rede SoCIN-VC 56

Figura 33 – Diagrama em blocos da arquitetura proposta com *aging* no roteador ParIS 58
Figura 34 – Formato do pacote modificado para *aging* 58
Figura 35 – Rede SoCIN com a *Interface_TDM_Channel* 61
Figura 36 – Sinais do mecanismo TDM 62
Figura 37 – *Interface_TDM_Channel* 63
Figura 38 – *Clock_Manager* 64
Figura 39 – Sinais do *Clock_Manager* 65
Figura 40 – *Connection_Table* 65
Figura 41 – Sinais do *Connection_Table* 66
Figura 42 – *DEMUX* 67
Figura 43 – Sinais do *DEMUX* 69
Figura 44 – *Data_SoCIN_Dr* 70
Figura 45 – Sinais do *Data_SoCIN_Dr* 71
Figura 46 – *Read_Write_MUX* 72
Figura 47 – Sinais do *Read_Writ_MUX* 73
Figura 48 – *MUX* 74
Figura 49 – Sinais do *MUX* 75
Figura 50 – *Data_SoCIN_Dx* 76
Figura 51 – Sinais do *Data_SoCIN_Dx* 77
Figura 52 – *Interface_TDM_Channel* 78
Figura 53 – Sinais do *Interface_TDM_Channel* 79

LISTA DE GRÁFICOS

Gráfico 1 – Comparativo no consumo de silício 81

LISTA DE TABELAS

Tabela 1 – Sinais do roteador QNoC 39
Tabela 2 – Comparativo sobre o custo de silício – chaveamento de circuitos 55
Tabela 3 – Comparativo sobre o custo de silício – canais virtuais 57
Tabela 4 – Comparativo sobre o custo de silício – envelhecimento de pacotes 59
Tabela 5 – Comparativo do custo de silício 80

LISTA DE ABREVIATURAS E SIGLAS

BE	Best Effort
bop	begin of packet
bps	bit por segundo
CRT	Current Routing Table
CSIP	Currently Serviced Input Port
DMA	Direct Memory Access
E/S	Entrada e Saída
FLIT	FLow control digIT
FPGA	Field Programmable Gate Array
GT	Guaranteed Throughput
HLP	Higher Level Protocol
LAN	Local Area Network
LUT	Look Up Table
MAN	Metropolitan Area Network
NBS	Next Buffer State
NoC	Network on Chip
ParIS	Parameterized Interconnection Switch
PDH	Plesiochronous Digital Hierarchy
QNoC	Quality of service NoC
QoS	Quality of Service
RIB	Routing Information Bits
SDH	Synchronous Digital Hierarchy
SoC	System on Chip
SoCIN	System on Chip Interconnection Network
TCC	Trabalho de Conclusão de Curso
TDM	Time Division Multiplexing
TRA	Target Routing Address
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
WAN	Wide Area Networks
WCET	Worst Case Execution Time

LISTA DE SÍMBOLOS

® Marca registrada em órgão oficial de registro de patentes.

SUMÁRIO

1 INTRODUÇÃO 18
1.1 PROBLEMATIZAÇÃO 18
1.1.1 Formulação do problema 19
1.1.2 Solução proposta 19
1.2 OBJETIVO 20
1.2.1 Objetivo Geral 20
1.2.2 Objetivos Específicos 20
2 FUNDAMENTAÇÃO TEÓRICA 21
2.1 REDES DE COMPUTADORES 21
2.2 CONCEITOS DE REDES-EM-CHIP 24
2.2.1 Comunicação de uma NoC 25
2.2.2 Roteador 27
2.2.3 Enlace 28
2.2.4 Largura de banda 28
2.2.5 Vazão 28
2.2.6 Latência 29
2.2.7 Topologia 29
2.2.7.1 Redes diretas 29
2.2.7.2 Redes indiretas 30
2.3 ESCALONAMENTO DE TAREFAS EM SISTEMAS DE TEMPO REAL 31
2.3.1 Conceitos básicos 31
2.3.2 Definições de sistemas de tempo real 31
2.3.3 Fator de utilização de um processador 32
2.4 MULTIPLEXAÇÃO 33
2.4.1 Multiplexação por Divisão de Tempo (TDM) 33
2.5 HIERARQUIA DIGITAL PLESIÓCRONA 34
2.6 HIERARQUIA DIGITAL SÍNCRONA 34
3 ARQUITETURAS DE REDES-EM-CHIP 36
3.1 ÆTHEREAL 36
3.1.1 Visão geral da arquitetura 36
3.1.2 Visão geral do roteador 36
3.2 QNOC 38

3.2.1 Visão geral da arquitetura ... 38
3.2.2 Visão geral do roteador ... 39
3.3 HERMES ... 42
3.3.1 Visão geral da arquitetura ... 42
3.3.2 Visão geral do roteador com chaveamento de circuitos (CS) ... 43
3.3.3 Visão geral do roteador com Prioridades Fixas (FP) ... 44
3.3.4 Visão geral do roteador com Prioridades Dinâmicas (DP) ... 46
3.3.5 Visão geral do roteador com escalonamento baseado em taxas (RB) ... 47
3.4 REDE SOCIN ... 50
3.4.1 Topologia ... 50
3.4.2 Enlace ... 50
3.4.3 Formato do pacote ... 51
3.4.4 Interface do roteador ... 52
3.4.5 Conexões entre canais ... 53
3.4.6 Chaveamento de circuitos na SoCIN – SoCIN-CS ... 54
3.4.7 Canais virtuais na rede SoCIN – SoCIN-VC ... 55
3.4.8 Envelhecimento de pacotes na rede SoCIN – SoCIN-AG ... 57
4 DESENVOLVIMENTO ... 60
4.1 *INTERFACE_TDM_CHANNEL* ... 60
4.1.1 *Clock_Manager* ... 64
4.1.2 *Connection_Table* ... 65
4.1.3 *Router_Interface_input* ... 66
4.1.3.1 *DEMUX* ... 67
4.1.3.2 *DATA_SOCIN_Dr* ... 69
4.1.3.3 *READ_WRITE_MUX* ... 71
4.1.4 *Router_Interface_output* ... 73
4.1.4.1 *MUX* ... 73
4.1.4.2 *DATA_SOCIN_Dx* ... 75
4.1.5 *Interface_TDM_Channel* ... 77
4.1.6 Resultados alcançados ... 79
4.1.6.1 CONSUMO DE SILÍCIO UTILIZADO PELA *INTERFACE_TDM_CHANNEL* ... 80
5 CONCLUSÃO ... 82
5.1 TRABALHOS FUTUROS ... 82
REFERÊNCIAS ... 83

1 INTRODUÇÃO

A constante evolução nos processos de fabricação de circuitos tem criado possibilidades para integração de sistemas num único chip (*System-on-Chip* – SoC), os quais são construídos com blocos de silício reutilizáveis. Com as novas tecnologias de integração, é possível construir um sistema com várias dezenas a centenas de núcleos num mesmo chip. Esses sistemas terão alto poder de processamento, porém, requererão estruturas de comunicação reutilizáveis com desempenho escalável, o que não é possível de ser obtido com as arquiteturas usadas atualmente, baseadas no barramento (HEMANI et al. 2000; BENINI; DE MICHELI, 2002).

A comunicação entre os núcleos de um SoC, tipicamente, é realizada por canais ponto-a-ponto, em que canais exclusivos são responsáveis pela interconexão dos núcleos, possibilitando assim melhor desempenho. No entanto, essa solução de interconexão apresenta baixa escalabilidade, pouca flexibilidade e baixa reusabilidade. Outra técnica de comunicação é através de canais multipontos, também conhecidos por barramento, tipicamente usados na comunicação com memória. Essa estrutura permite reusabilidade, o que reduz o tempo do desenvolvimento de novos projetos. Porém, como desvantagem, podemos citar o baixo grau de paralelismo, pois somente uma transação pode ser realizada no mesmo instante de tempo. Ainda tem o problema no desempenho causado pelas capacitâncias e resistências parasitas devido à utilização de fios longos (BENINI, 2002).

Uma forma de resolver tais problemas é o uso de redes de intercomunicação chaveadas, similares as que são usadas em computadores paralelos, as quais são conhecidas como Redes intra-chip ou NoC (do inglês Network-on-Chip). Entre as principais vantagens desse sistema de comunicação estão: a largura de banda escalável, o uso de conexões ponto-a-ponto curtas e o paralelismo na comunicação. Como desvantagem, esta técnica apresenta maiores custos de silício e latência na comunicação. Esses problemas podem ser atenuados utilizando-se de grande quantidade de transistores e soluções arquiteturais (BENINI, 2002).

1.1 PROBLEMATIZAÇÃO

Os sistemas embarcados atuais demandam por um alto nível de integração com múltiplos núcleos de processamento que trocam dados entre si, como microprocessadores. Normalmente, tais sistemas são utilizados em sistemas de tempo real, no qual o tempo de pior caso de execução WCET (*Worst Case Execution Time*) é relevante (BUTTAZZO; 2005). Tais

sistemas possuem requisitos rigorosos no processamento e nos serviços de comunicação, o que torna o uso de NoC inadequado se não houver garantias temporais na comunicação da rede.

1.1.1 Formulação do problema

Diversas soluções tem sido apresentadas na literatura para lidar com o problema da previsibilidade na comunicação em redes intra-chip. Algumas delas estão baseadas no uso de canais virtuais dedicados para fluxos com prioridade na comunicação. Outras soluções baseiam-se no uso de chaveamento de circuitos, fazendo com que um caminho entre a origem e o destino na comunicação seja garantido através da reserva dos recursos de comunicação (canais e roteadores) (BENINI, 2002).

Uma terceira técnica utilizada para garantir a latência[1] na comunicação é o uso de multiplexação dos canais de comunicação no tempo. Essa técnica, também conhecida pelo acrônimo de TDM (*Time-Division Multiplexing*) é empregada por algumas redes como a rede Æthereal (GOOSSENS; 2005). Porém, as redes que adotam essa tecnologia exigem que a comunicação entre dois pontos quaisquer da rede só possa ocorrer após uma negociação de qual fatia de tempo (ou em inglês *time slots*) será usada na comunicação entre dois núcleos da rede (HANSSON; 2007). De forma similar, a conexão também deve ser informada, caso necessário, para liberar o *time slot* para outras aplicações.

1.1.2 Solução proposta

A solução que está sendo proposta neste trabalho é o uso de um mecanismo de comunicação, baseado na técnica TDM, que permita à rede SoCIN (System-on-Chip Interconnection Networks) fazer a alocação dos *time slots*.

---

[1] A latência é o tempo necessário para que um pacote atravesse a Rede de uma origem até um destino. (DALLY; 2004)

1.2 OBJETIVO

1.2.1 Objetivo Geral

O objetivo principal do trabalho é propor a utilização da técnica de multiplexação por divisão do tempo (em inglês, *Time Division Multiplexing* – TDM) no canal de comunicação da rede SoCIN, como forma de garantir previsibilidade de latência na comunicação entre dois pontos quaisquer da rede. A locação dos *time slots* deverá ser feita pelos blocos de saída da rede SoCIN, em tempo de execução. Não cabe ao escopo deste trabalho a realização das alterações necessárias na rede SoCIN para realizar a alocação dinâmica. Este trabalho apresenta apenas o mecanismo TDM devidamente adaptado para receber os comandos vindos dos roteadores da rede SoCIN.

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Entender o conceito de Redes em Chip;
- Conhecer a arquitetura da rede SoCIN;
- Analisar as técnicas de multiplexação;
- Implementar canal TDM na rede SoCIN; e
- Fazer simulações para validar o funcionamento do canal TDM.

2 FUNDAMENTAÇÃO TEÓRICA

Este trabalho buscou implementar um mecanismo de multiplexação dos canais de comunicação na rede SoCIN. Para fornecer a fundamentação teórica necessária para este trabalho, este Capítulo foi elaborado com o objetivo de apresentar os conceitos fundamentais que envolvem a redes intra-chip, apresentando seus principais elementos e termos técnicos adotados nesta área do conhecimento, e também apresentar os conceitos fundamentais de multiplexação do tempo e aspectos relacionados a sistemas de tempo real.

Esta Seção está organizada da seguinte forma: a Subseção 2.1 apresenta os conceitos básicos de redes de computadores como referência para o estudo de redes intra-chip; a Subseção 2.2 trata de conceitos de redes de interconexão, que servirão de base teórica para as NoCs; a Subseção 2.3 é abordado o conceito sobre escalonamento para sistemas de tempo real; a Subseção 2.4 é uma revisão da técnica de multiplexação no tempo (TDM), pois esse conceito é aplicado no desenvolvimento do trabalho. As subseções 2.5 e 2.6 apresentam conceitos de sincronismos digitais, sendo elas: a Plesiócrona e Síncrona, respectivamente.

2.1 REDES DE COMPUTADORES

Uma rede de computadores pode ser vista como um conjunto de computadores, enlaces de comunicação e diversos periféricos compartilhando vários recursos, tais como impressoras, modems, arquivos de dados gravados, entre outros (DANTAS, 2002, p.3).

A rede de computadores pode ser definida pela abrangência física, ou seja, pela distância em que os recursos estão interligados. Quando esta distância é de poucos quilômetros, em torno de 25, a rede é denominada de LAN (*Local Area Network*). Essas redes surgiram para viabilizar a troca e o compartilhamento de informações e recursos de hardware e software, preservando as várias estações de processamentos e permitindo a integração em ambientes de trabalhos cooperativos (SOARES et al., 1995, p.11) e (DANTAS, 2002, p.15).

Quando a distância de ligação entre os vários módulos processadores abrange uma região metropolitana de uma determinada cidade, não são mais consideradas redes locais e sim uma rede metropolitana, ou MAN (*Metropolitan Area Network*). A definição do termo ‘rede metropolitana’ surgiu através do aparecimento do padrão IEEE 802.6 (SOARES et al., 1995, p.12). Uma MAN, de certa forma, é uma versão ampliada de uma LAN, pois as tecnologias utilizadas por ambas são basicamente as mesmas (TANENBAUM, 1997, p.10).

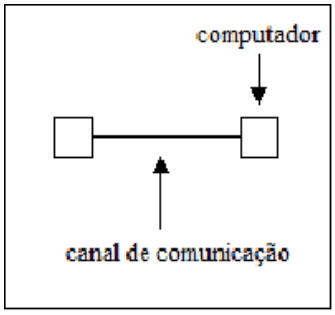
Já em âmbito global, a rede é classificada como WAN (*Wide Area Networks*). As WANs são redes distribuídas geograficamente que englobam uma ampla área, desde país até continente. Nessas redes a taxa de erros são mais altas do que as já citadas e a informação é destinada através de roteamento (DANTAS, 2002, p.15).

Na concepção de Comer (1999), "cada rede é classificada em uma categoria de acordo com sua topologia ou forma geral." Algumas dessas topologias são descritas a seguir.

- Ponto a ponto
- Estrela
- Anel
- Barramento

A topologia com interconexão ponto-a-ponto é a forma mais comum, na qual há dois pontos, um de transmissão e outro de recepção, que são interligados de maneira que possam trocar informações diretamente (SOUSA, 1999, p.23). Essa rede baseia-se em uma comunicação compartilhada, ou seja, os computadores se alternam no uso do meio para enviar pacotes (COMER, 1999, p.83). A Figura 1 ilustra uma topologia ponto-a-ponto.

Figura 1 – Topologia ponto-a-ponto

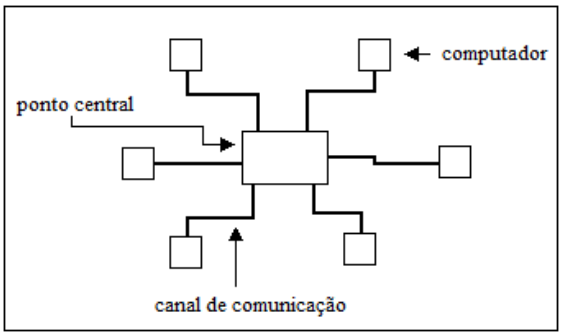


Fonte: Próprio autor.

A topologia é classificada como estrela se todos os equipamentos de uma rede estão conectados a um ponto central, que pode ser um computador de grande porte chamado de mainframe ou host central. (SOUSA, 1999, p. 25). Na configuração em estrela não há necessidade de roteamento, pois todas as mensagens se concentram num ponto central. O gerenciamento das comunicações por esse ponto pode ser tanto por chaveamento de pacotes quanto por chaveamento de circuitos. Nesse caso, os pacotes são enviados de um ponto de

origem ao ponto central, que os retransmite ao ponto destino no instante apropriado (SOARES et al., 1995, p.26). Essa topologia é representada pela Figura 2.

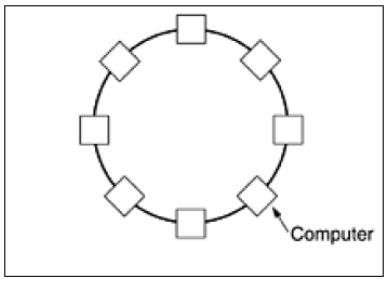
Figura 2 – Topologia Estrela



Fonte: Próprio autor.

Na arquitetura em anel, as estações estão conectadas através de um caminho fechado. Nessas redes, em teoria, a transmissão e recebimento dos dados podem ocorrer em qualquer direção. Uma mensagem ao entrar no anel fica circulando até encontrar o destino ou até voltar ao ponto de origem (SOARES et al., 1995). A Figura 3 ilustra a topologia em anel.

Figura 3 – Topologia Anel

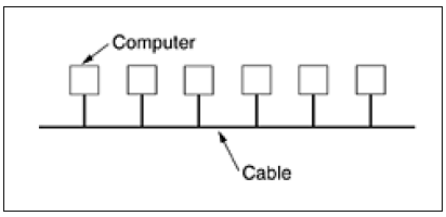


Fonte: Tanenbaum (1997, p. 11).

Em uma rede de barramento, representada pela Figura 4, a interconexão dos computadores é estabelecida ao longo de um cabo. Quando um computador envia uma mensagem nessa rede, todos os que estão conectados a esse cabo receberão a mensagem (COMER, 1999, p.86). Todas as máquinas podem realizar uma transmissão, portanto, qualquer uma pode desempenhar o papel de mestre. No momento da transmissão as outras

máquinas não poderão enviar nenhum tipo de mensagem. Para evitar conflito, é necessário criar algum tipo de arbítrio quando duas ou mais máquinas quiserem realizar uma transmissão simultaneamente. Os computadores de uma rede, padrão IEEE 802.3, conhecida como Ethernet, podem estabelecer uma transmissão no instante em que quiserem, pois, caso houver uma colisão de dois ou mais pacotes, cada computador aguardará um tempo aleatório para então fazer uma nova tentativa (TANENBAUM, 1997).

Figura 4 – Topologia Barramento



Fonte: Tanenbaum (1997, p. 11).

2.2 CONCEITOS DE REDES-EM-CHIP

As redes-em-chip, em inglês *Network-on-Chip* (NoC), são estruturas de interconexão que fazem ligação de vários núcleos dentro de um SoC. Essas redes baseiam-se no conceito de redes de computadores (BENINI, 2002).

A NoC pode ser definido como sendo vários roteadores interligados entre si por um canal de comunicação ponto-a-ponto na qual é possível estabelecer trocas de informações. Uma rede-em-chip é composta basicamente por roteadores e enlaces (*links*). Os enlaces são os canais que ligam os roteadores entre si e aos nodos[2] do computador paralelo, enquanto que os roteadores ficam estabelecendo o caminho necessário à transferência de dados pela rede. Esses dados são transferidos em forma de mensagens, que por sua vez, podem ser divididas em unidades menores, denominadas pacotes (ZEFERINO, 2003).

Segundo Zeferino (2003, p.29), uma rede de interconexão se caracteriza pela sua topologia e por um conjunto de mecanismos responsáveis em definir a maneira da qual

[2] Um nodo pode ser um processador, um módulo de memória ou um computador com processador mais a memória local e uma interface de rede (ZEFERINO, 2003, p.29).

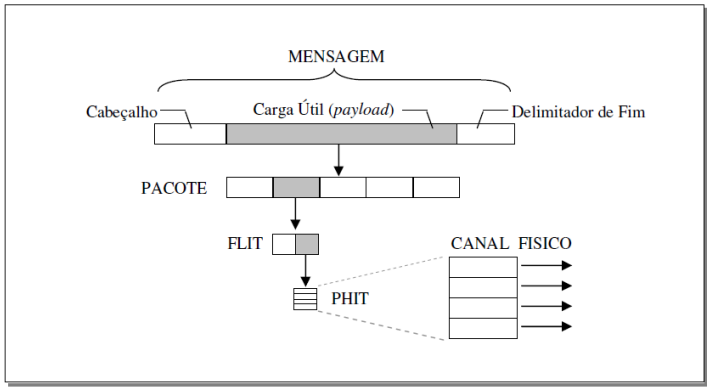
ocorrerá à transferência de mensagens pela rede. O desempenho da rede está relacionado diretamente ao tipo de topologia e dos mecanismos de comunicação utilizados. A largura de banda, a vazão e a latência são métricas para avaliar o desempenho. Nas Subseções seguintes serão descritos os aspectos citados nesse parágrafo.

2.2.1 Comunicação de uma NoC

Para Benini e De Micheli (2002), em uma NoC, uma comunicação entre núcleos é estabelecida através de mensagens. A estrutura dessas mensagens é dividida em três partes: o cabeçalho (*header*), os dados da mensagem ou carga útil (*payload*) e o delimitador de fim (*trailer*). No cabeçalho estão as informações, tanto a que informa início da mensagem, quanto à de roteamento e controle, utilizadas pelos roteadores para trafegar a mensagem até o nodo de destino. Já o delimitador de fim, contém informações usadas para a detecção de erros e também de indicação do fim da mensagem. Conforme já citado na seção 2, a mensagem pode ser dividida em partes menores, denominados pacotes, que contém a mesma estrutura da mensagem, ou seja, cabeçalho, carga útil e delimitador de fim.

O pacote é formado por uma sequência de palavras que possui um tamanho igual à largura física do enlace, a qual é denominada PHIT (acrônimo do inglês, *PHysical unIT*). Um PHIT é definido pelo número de bits transmitidos simultaneamente, no caso de um enlace serial o tamanho de um PHIT é igual a 1 bit e em enlaces paralelos possui um tamanho de *n* bits. Além do que, um pacote pode ser dividido em FLITs (acrônimo do inglês, *FLow control unIT*), do qual o tamanho é múltiplo do PHIT, conforme ilustra a Figura 5 (CORRÊA, 2007).

Figura 5 – Estrutura da mensagem



Fonte: Corrêa (2007, p. 32).

Para que haja a comunicação entre os nodos é necessário garantir que o pacote seja entregue ao seu destino, porém, existem três situações que podem impedir que isso aconteça, são elas: *deadlock*, *livelock* e *starvation* (DUATO et al., 1997). Para Zeferino (2003, p.35), o *deadlock* é o problema mais difícil de ser resolvido.

O *deadlock* é definido como sendo uma dependência cíclica entre as solicitações de acesso aos recursos, tanto de comunicação, quanto de armazenamento. Isso ocorre na rede toda vez que cada pacote solicita o uso de recursos que já estão ocupados por outro pacote. Uma das maneiras para prevenir o *deadlock* é usando um chaveamento por circuito que reserva todos os recursos das redes na transmissão do pacote entre os nodos de origem e de destino. Quando se trata de uma rede que aloca os recursos à medida que o pacote vai avançando, para evitar o *deadlock*, limita-se o número de voltas que o algoritmo de roteamento pode realizar na rede, permitindo assim, uma melhor utilização da rede. Outra estratégia é a de recuperação, entretanto, é recomendada somente em situações em que o *deadlock* não é frequente (DUATO et al., 1997). Contudo, para esse caso, são usadas ferramentas de detecção dos *deadlocks* e também de realocação dos recursos entre os pacotes. Sendo que, os pacotes que não possuem mais os recursos já alocados podem ser descartados ou até mesmo terem suas rotas redefinidas (ZEFERINO, 2003).

Já a situação conhecida por *livelock*, acontece pelo fato do pacote ficar trafegando permanentemente na rede, pois os canais de acesso ao nodo destinatário encontram-se sempre indisponíveis. Em redes contendo algoritmos de roteamentos que não utilizam os caminhos mais curtos e são tolerantes a falhas, é comum que aconteça esse tipo de problema. No entanto, para evitar o *livelock* basta apenas permitir o caminho mais curto até o nodo de

destino. A prevenção do *livelock* pode ser através da limitação do número de operação de desvio através de caminhos que não são os mais curtos (ZEFERINO, 2003, p.35).

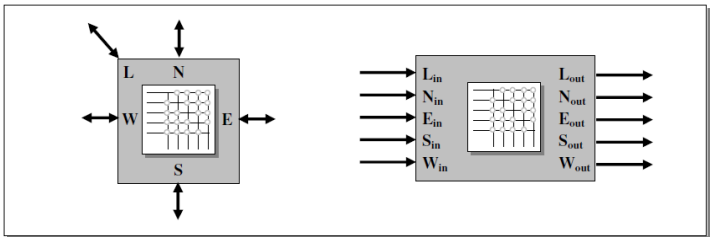
No roteador pode existir dois ou mais *buffers* de entradas com pacotes destinados a uma mesma saída. Ao ocorrer isso, uma política arbitrária deve ser utilizada para selecionar qual dos *buffers* de entrada será conectado à saída. A arbitragem pode ser feita através de prioridades estáticas ou dinâmicas. Dependendo da arbitragem estabelecida, um pacote no *buffer* de entrada pode ficar bloqueado permanentemente, caso concorra pela saída com outros pacotes que tenham a preferência. Esse problema é denominado *starvation*, porém, utilizando-se um mecanismo de arbitragem adequado, poderá ser evitado. Se a maioria dos pacotes prioritários ocupar toda a banda, uma solução será reservar uma parte da banda para os pacotes de menor prioridade (ZEFERINO, 2003, p.35).

2.2.2 Roteador

Para De Freitas (2009, p.29), o roteador é o elemento fundamental para interconexão da rede, pois é o responsável pela definição de rotas, controle de fluxo e garantia de entrega dos pacotes de dados, determinando assim, a qualidade de serviço.

Conforme Zeferino (2003, p.30), um roteador é composto por um núcleo de chaveamento (*crossbar*), uma lógica para controlar o roteamento e arbitragem e portas de entradas e saídas que servem para se comunicar com outros roteadores e/ou com os nodos do computador paralelo. Nessas portas são incluídos canais de entrada e saída que podem ter uma pequena memória para armazenamento temporário de dados (*buffer* ou fila). Entretanto, as portas ainda têm controladores de enlace para a implantação do protocolo físico de comunicação que servem para regular o tráfego das informações. A Figura 6 representa um exemplo de roteador (CORRÊA, 2007, p.31).

Figura 6 – Exemplo de um roteador



Fonte: Corrêa (2007).

2.2.3 Enlace

Os enlaces são canais ponto-a-ponto unidirecionais e assíncronos, que fazem a ligação entre dois roteadores ou entre o nodo do computador paralelo e um roteador, podendo até possuir um ou dois canais físicos de comunicação. Sua implementação é feita por um cabo elétrico ou óptico. Normalmente, os enlaces em redes de interconexão são *full-duplex*, formados por dois canais unidirecionais opostos, possibilitando a transferência simultânea dos dados nas duas direções do enlace (ZEFERINO, 2003, p.30) e (DUATO, 2003).

2.2.4 Largura de banda

A largura de banda, ou em inglês *bandwidth*, é a taxa máxima que a rede de interconexão consegue trafegar as informações assim que uma mensagem entra na rede. O cálculo da largura de banda é contado através dos bits do cabeçalho, da carga útil e do delimitador de fim da mensagem. A unidade de medida para a largura de banda é bps (bit por segundo) (HENNESSY; PATTERSON, 2003).

2.2.5 Vazão

A vazão, ou em inglês *throughput*, é definido por Dally e Towles (2004) como sendo o número de mensagens que a rede entrega na unidade de tempo.

Para Hennessy e Patterson (2003) é definida como sendo a largura de banda da rede para uma dada aplicação.

Entretanto, as duas definições não são muito claras, pois seguindo o que foi definido no primeiro parágrafo, o valor varia se o tamanho das mensagens varia, já no segundo parágrafo, a vazão pode ser confundida com a largura de banda máxima da rede.

Na definição de Duato (2003), é apresentado um conceito mais nítido. A vazão é o tráfego máximo aceito pela rede, em outras palavras, é a quantidade máxima de informação entregue na unidade de tempo. O seu valor pode ser normalizado, dividindo-o pelo tamanho das mensagens e pelo da rede, com isso, a vazão fica independente do tamanho das mensagens e da rede. Sua medida é em bits por nodo por ciclo de relógio (ou por microssegundos).

2.2.6 Latência

A latência é o tempo que a mensagem leva da origem até o destino, considerando desde o início do envio até o recebimento completo da mensagem. Em alguns casos, a latência individual de cada mensagem é de pouca importância, ainda mais quando são usadas cargas sintéticas (DUATO, 2003). Nessa situação, é mais significativo para avaliação da rede o valor médio da latência, ou também denominado, latência média. Contudo, poderá haver um impacto importante no desempenho de algumas aplicações, caso algumas mensagens experimentam latências muito maiores que o valor médio.

Os fatores que influenciam a latência de uma rede com carga são: O tempo gasto pelos nodos para colocar e retirar a mensagem da rede, o tempo consumido para propagar a mensagem pelos enlaces entre os nodos, fonte e destino, o tempo despendido pelo roteador para determinar a rota a ser utilizada na propagação da mensagem e os períodos de tempo em que a mensagem é impedida de avançar devido ao congestionamento da rede (ZEFERINO, 2003).

2.2.7 Topologia

De acordo com Dally e Towles (2004), as redes de interconexão são formadas por um arranjo de nodos e canais compartilhados, sendo que, a topologia é a forma em que esses nodos e canais estão arranjados.

Para Zeferino (2003, p.32), as redes de interconexão, quanto à topologia, podem ser agrupadas em duas classes: as redes diretas e as indiretas.

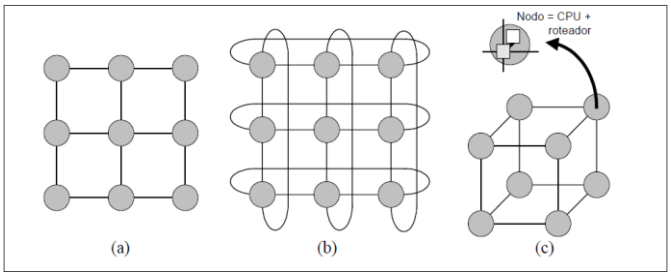
2.2.7.1 Redes diretas

As redes diretas são caracterizadas pela associação de cada roteador a um processador, formando um único elemento chamado nodo. Cada nodo é ligado ponto-a-ponto a outros nodos vizinhos. Uma mensagem trocada entre dois nodos que não são vizinhos deve passar por nodos intermediários. Uma mensagem ao ser recebida por um nodo que não é o destinatário deve ser encaminhada para algum nodo vizinho para que a mensagem prossiga. Os processadores dos nodos intermediários não se envolvem nessa comunicação, somente os roteadores. Para tanto, um algoritmo de roteamento é utilizado pelo roteador com o propósito

de decidir qual dos nodos vizinhos a mensagem será destinada (CORRÊA, 2007, p.34) e (ZEFERINO, 2003,p.33).

Tratando-se de conectividade, a rede direta ideal é aquela completamente conectada. A figura 7 mostra algumas topologias de redes diretas. Conforme é representado, o processador e o roteador compõe um nodo, além de outros componentes, como memória e entrada-e-saída (E/S).

Figura 7 – Redes diretas: (a) grelha 2-D; (b) toróide 2-D; (c) hipercubo 3-D



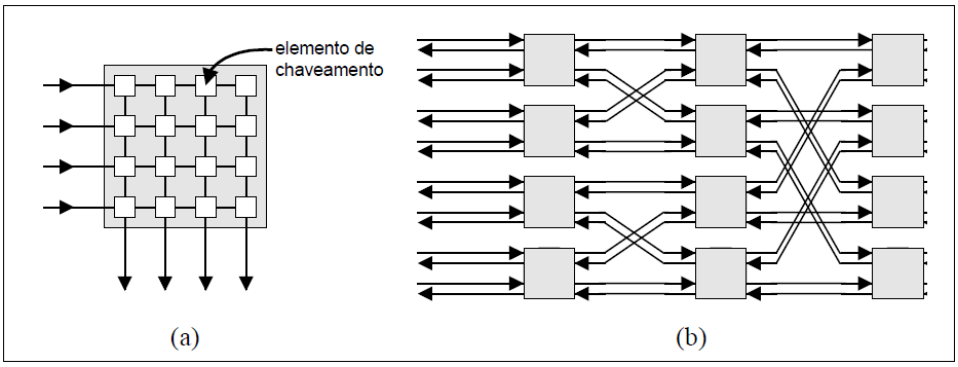
Fonte: Zeferino (2003, p. 33).

2.2.7.2 Redes indiretas

Nas redes indiretas, os roteadores e processadores não formam um elemento único como nas redes diretas. Os nodos (agora sendo processadores, módulos de memória ou computadores completos) têm uma interface para uma rede de computadores. É atribuído a cada roteador um conjunto de portas bidirecionais que possibilitam fazer ligações com outros roteadores e/ou com nodos da máquina.

Dentre algumas, duas topologias clássicas de redes indiretas se destacam, são elas: o *crossbar* e as redes multiestágio, mostradas na Figura 8.

Figura 8 – Redes indiretas: (a) crossbar; (b) multiestágio



Fonte: Zeferino (2003, p. 34).

2.3 ESCALONAMENTO DE TAREFAS EM SISTEMAS DE TEMPO REAL

2.3.1 Conceitos básicos

Segundo Buttazzo (2005), quando um único processador tem que executar um conjunto de tarefas que podem se sobrepor no tempo, ele deve escolher qual tarefa executar num determinado tempo de acordo com critérios pré-definidos, chamados de "política de escalonamento". O conjunto de regras que, a qualquer instante, determina a ordem na qual as tarefas são executadas é chamado de "algoritmo de escalonamento". O autor afirma que uma tarefa que pode ser executada em um processador pode estar em dois estados possíveis: estar sendo executada pelo processador ou aguardando pelo processador, se outras tarefas estiverem sendo executada. A tarefa que está sendo executada pelo processador é chamada de "tarefa em execução", enquanto que as tarefas que estão aguardando pelo processador são chamadas de "tarefas prontas", e estas tarefas prontas ficam aguardando por sua execução em uma fila denominada de "fila de prontos".

2.3.2 Definições de sistemas de tempo real

Os sistemas de tempo real são caracterizados por atividades computacionais com restrições temporais rigorosas que precisam ser atendidas para que se possa ter o comportamento esperado no sistema. Uma restrição temporal típica é o "prazo de execução", usualmente chamado de *deadline*, o qual representa o tempo antes do qual a tarefa deve ser

concluída para evitar prejuízos ao sistema. Se um *deadline* é especificado em relação ao tempo de chegada, ele é chamado de "*deadline* relativo", enquanto que se ele for especificado em relação ao tempo "zero", ele é chamado de "*deadline* absoluto" (BUTTAZZO; 2005).

A maioria dos autores define duas classes de tarefas de tempo real que dependem das consequências causadas pela perda de *deadline*. São chamados de sistemas *hard real-time* e sistemas *soft real-time*. Uma tarefa é dita como sendo *hard* se a ocorrência de sua finalização for posterior aos seus *deadlines* e podem causar consequências catastróficas ao sistema (LIU; 2000). Por sua vez, uma tarefa é dita como sendo *soft* se a perda de seu *deadline* causa apenas degradação do desempenho do sistema, sem colocá-lo em perigo (LIU; 2000).

2.3.3 Fator de utilização de um processador

O fator de utilização de um processador provê uma medida da carga computacional em um processador devido a um conjunto de tarefas periódicas. Este fator foi definido formalmente por (LIU, LAYLAND; 1973) da seguinte forma:

Dado um conjunto de tarefas, Γ de *n* tarefas periódicas, então o fator de utilização *U* é uma fração do tempo que o processador disponibiliza para a execução do conjunto de tarefas. Como $\frac{C_i}{T_i}$ é a fração de tempo que o processador disponibiliza para a execução de uma tarefa τ_i, então o fator de utilização para n tarefas é dado pela seguinte expressão:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Embora a utilização do processador possa ser aumentada pelo tempo de computação das tarefas ou pela redução dos seus períodos de execução, existe um valor máximo U abaixo do qual Γ é escalonável e acima do qual Γ não é escalonável (BUTTAZZO; 2005).

2.4 MULTIPLEXAÇÃO

Para Dantas (2002, p.42), a ligação física entre os dispositivos em uma rede de comunicação é conhecida como canal. O objetivo básico da multiplexação é o de compartilhar o mesmo meio de transmissão (canal) entre vários usuários (SOUSA, 1999, p.266).

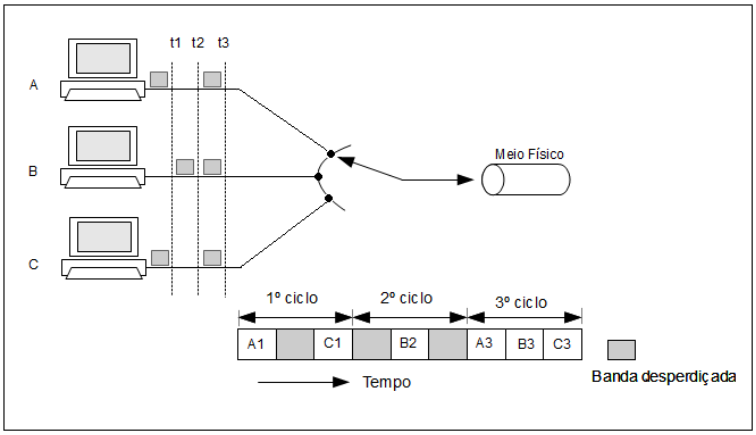
2.4.1 Multiplexação por Divisão de Tempo (TDM)

Na técnica de multiplexação por divisão do tempo (*Time Division Multiplexing* - TDM), vários sinais ou mensagens podem trafegar por um único caminho físico, intercalando-se porções de cada sinal no tempo (SOARES et al., 1995, p.66). De acordo com Dantas (2002), a multiplexação TDM trata dados digitais através de duas técnicas:

- Síncrona: nesse tipo de abordagem TDM cada usuário possui acesso ao canal de maneira precisa em termos de intervalos de tempo;
- Assíncrona: para esta abordagem (em demanda) cada usuário faz um acesso aleatório ao canal, que ao adquirir o canal, apenas ele poderá fazer uso do meio para tempo necessário do uso do canal. É importante saber que alguns mecanismos conhecido por *jabber* existem para que um computador não tenha acesso infinito à rede.

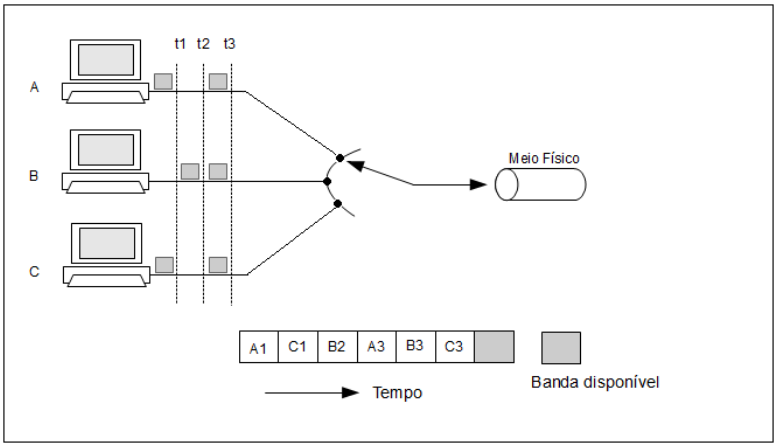
As Figuras 9 e 10 ilustram o método de TDM para as abordagens em ciclo fixo e em demanda, respectivamente. Ambas referem-se a uma conexão entre computadores e um link TDM.

Figura 9 – TDM abordagem de ciclo fixo



Fonte: Dantas (2002).

Figura 10 – TDM abordagem em demanda



Fonte: Dantas (2002).

2.5 HIERARQUIA DIGITAL PLESIÓCRONA

A Hierarquia Digital Plesiócrona (*Plesiochronous Digital Hierarchy* – PDH) foi concebida para uma arquitetura de multiplexação assíncrona. Cada canal multiplexado opera de maneira plesiócrona, ou seja, com um relógio que não é sincronizado com os relógios dos outros canais.

As redes digitais que utilizam o padrão PDH surgiram na área de telecomunicações para atender aos sistemas de telefonia em que sinais de alto nível são obtidos de múltiplos sinais de baixo nível adicionados a ‘bits de enchimento’ com o propósito de compatibilizar as diferentes velocidades de transmissão (PINHEIRO, 2004).

2.6 HIERARQUIA DIGITAL SÍNCRONA

O padrão da Hierarquia Digital Síncrona (*Synchronous Digital Hierarchy* – SDH) é baseado em uma arquitetura de multiplexação síncrona. Nesse caso, cada canal opera com um relógio sincronizado com os relógios dos outros canais e é sincronizado com o equipamento *multiplex* através de um processo de justificação de bit e encapsulamento da informação. O sinal SDH transporta também os diferentes tipos de sinais PDH, através do

frame padronizado chamado STM-N (*Syncronous Transport*), usado tanto para sinais elétricos quanto para sinais ópticos.

Cada *frame* SDH possui um cabeçalho (*header*) com informações de controle, gerenciamento do quadro e de seus diversos canais multiplexados (normalmente chamados de tributários) e a carga útil (*payload*) com os tributários propriamente ditos. O *frame* SDH tem um tamanho padrão para cada hierarquia.

As redes SDH formam um sistema síncrono no qual todos os relógios de seus equipamentos possuem, em média, a mesma frequência. O relógio de cada equipamento, chamado de relógio secundário ou escravo, pode ser rastreado até o relógio mestre, garantindo assim, a distribuição e qualidade do sinal de sincronismo (FILHO, 2002).

3 ARQUITETURAS DE REDES-EM-CHIP

Este capítulo apresenta algumas NoCs que oferecem qualidade de serviços em comunicação. Na Subseção 3.1 é feita uma abordagem conceitual da rede Æthereal, na Subseção 3.2 da rede Qnoc, na Subseção 3.3 da rede Hermes e por último, na Subseção 3.4 é apresentada a rede SoCIN.

3.1 ÆTHEREAL

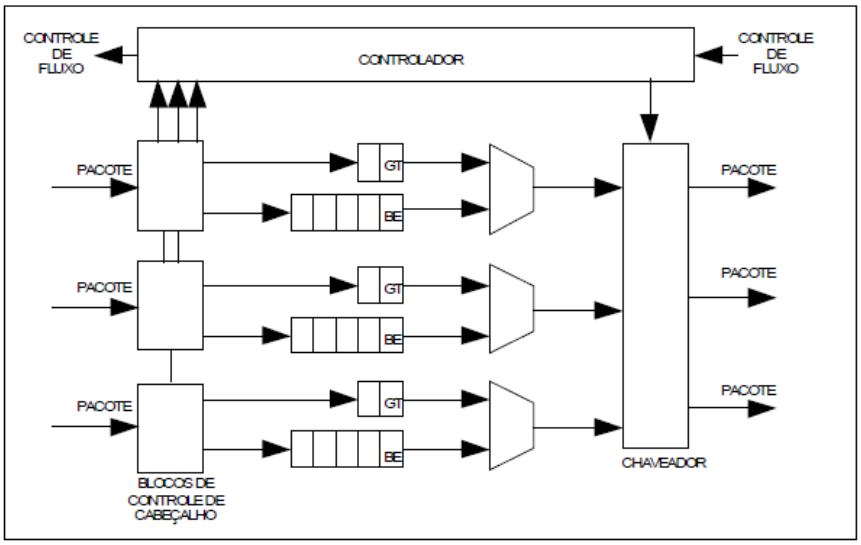
3.1.1 Visão geral da arquitetura

A rede Æthereal, desenvolvida por Goossens et al. (2005), foi concebida tanto com serviços de melhor esforço (em inglês, *Best Effort* - BE) quanto com serviços de vazão garantida (em inglês, *Guaranteed Throughput* - GT). Essa rede utiliza uma técnica de roteamento livre de contenção baseada na técnica TDM, que provê um chaveamento de circuitos, podendo assim, implementar serviços garantidos. Isso requer uma noção lógica de sincronismo na qual todos os roteadores e interfaces de rede estão em *time slots* (ou fatias de tempo) específicos. Os circuitos são criados através da reserva de time slots consecutivos em roteadores consecutivos, ou seja, se um circuito é estabelecido do roteador R para um roteador R' e o *time slot* S é reservado no roteador R, então o *time slot* (S+1) deve ser reservado no roteador R'. Com isso, uma informação recebida em um *time slot* será enviado ao roteador seguinte no próximo *time slot*. Assim, as informações são transportadas pela rede sem contenção, garantindo dessa maneira, a vazão e a latência.

3.1.2 Visão geral do roteador

De modo a minimizar o custo da implementação, o roteador da rede Æthereal usa o chaveamento *wormhole*, no qual a capacidade do *buffer* pode ser menor que o tamanho de um pacote. Para reduzir a capacidade de fila, e por consequência minimizar a área de implementação, são usadas filas de entrada, denominadas filas GT para fluxos com garantia de vazão e filas BE para fluxos com garantias de melhor esforço. A arquitetura do roteador da Æthereal é mostrada na Figura 11.

Figura 11 – Arquitetura do roteador da Æthereal



Fonte: Dielissen et al. (2003).

Foi definido, por Dielissen et al. (2003), o roteamento na origem como esquema de endereçamento, pois julgou-se ser uma implementação com maior grau de independência e de menor custo. Nesse esquema de roteamento, o caminho em que o pacote irá seguir até seu destino está dentro do cabeçalho do pacote. Na Æthereal, esse caminho é uma lista com endereços de destino em que cada roteador no caminho remove o primeiro elemento para o seu próprio uso.

Conforme pode ser observado na Figura 11, cada roteador tem um elemento de controle e um caminho de dados. Neste, as mensagens de entrada, do roteador e das interfaces de rede, são analisadas pela Unidade de Análise de Cabeçalho (*Header Parsing Unit*). A unidade remove o primeiro elemento da lista para o caminho, envia o *flit* já analisado para a fila BE ou GT e informa ao controlador que está disponível um novo pacote para ser transmitido. Então, o controlador faz o escalonamento dos *flits* para o próximo ciclo. Depois de concluído o escalonamento de todos os *flits* presentes na fila de GT, o controlador escalona os slots restantes para os fluxos BE. Se houver conflito, por exemplo, dois fluxo BE endereçando o mesmo destinatário, é aplicado um esquema de arbitragem *round-robin*.

Foi implementado um esquema de detecção de nível de canal para evitar transbordo na fila de entrada de fluxos BE. Ao ser inicializado, cada roteaor tem uma quantidade de espaço livre nas suas conexões com roteadores ou interfaces de rede. Sempre que um *flit* é enviado a outro roteador, o contador de espaço livre do destinatário

correspondente é decrementado. O roteador ao escalonar um *flit* para o próximo slot sinaliza ao seu antecessor que o contador de espaço livre pode ser incrementado.

3.2 QNoC

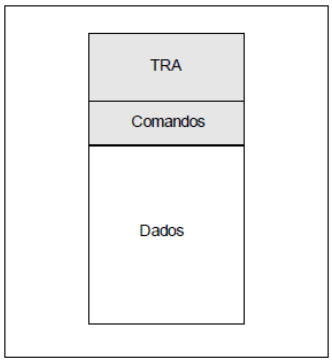
3.2.1 Visão geral da arquitetura

A rede QNoC (*Quality-of-service* NoC) foi apresentada por Bolotin et al. (2003). Foi projetada para prover QoS baseando-se em diferenciação de serviços, sendo quatro níveis de serviço para diferentes classes de tráfego, cada qual com um nível de prioridade diferente:

- Sinalização: Nível de serviço com maior prioridade na rede. É reservado para sinais de controle e interrupções e sua estrutura é constituída por 2 *flits*;
- Tempo real: Esse nível garante largura de banda e latência para aplicações de tempo real;
- Leitura/Escrita: Corresponde a acessos curtos a memórias e registradores; e
- Transferência de bloco: Nível de serviço com menor prioridade, usado para transferência de mensagens extensas e blocos grandes de dados, tal como preenchimento de setores de memória *cache* e transferência DMA (*Direct Memory Access*).

Cada pacote da rede é constituído por três campos, são eles: o campo de informações de roteamento ou TRA (*Target Routing Address*), que contém o endereço necessário para o roteamento do pacote, o campo de dados e o campo de comandos, que por sua vez, especifica o tipo de operação ao qual ser referem os dados transportados no campo de dados. Os campos que formam a estrutura do pacote são ilustrados na Figura 12.

Figura 12 – Formato do pacote



Fonte: Bolotin et al. (2003).

Os pacotes são divididos em vários *flits* e transmitidos através de sinais específicos do roteador, chamados de Dados de Saída (*Data*_o), conforme é mostrado na tabela 1. Os *flits* são classificados em três tipos:

- FP (*Full Packet*): primeiro *flit* do pacote;
- EP (*End of Packet*): Último *flit* do pacote; e
- BDY (*Body*): Não é o último *flit* do pacote.

Tabela 1 – Sinais do roteador QNoC

Sinais	Largura	Descrição
Clk	1	Relógio que sincroniza a transmissão de flits.
Data_o	Parametrizável	Dados saindo do roteador
Type	2	Tipo do flit: 00: idle 01: EP 10: BDY 11: FP
SL	2	Nível (classe) de serviço do flit
Buffer_Credit_SL	4	Indica se existe espaço disponível no *buffer* para cada nível de serviço.
Buffer_Credit_Valid	1	Indica se *Buffer_Credit_SL* transporta uma informação de crédito válido.

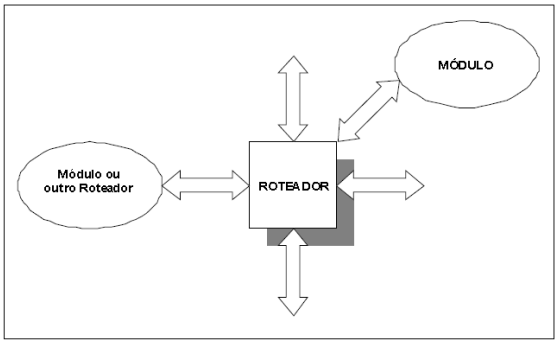
Fonte: Bolotin et al. (2003).

3.2.2 Visão geral do roteador

Os roteadores da rede se conectam em até cinco canais de comunicação, como mostra a Figura 13. Sendo um para conexão do núcleo local e os outros quatros para roteadores vizinhos. O roteador encaminha os pacotes recebidos nas portas de entrada para as

portas de saída. Os dados são recebidos em *flits*, sendo que cada *flit* que chega é primeiro armazenado em *buffers* de entrada.

Figura 13 – Canais de comunicação do roteador



Fonte: Bolotin et al. (2003).

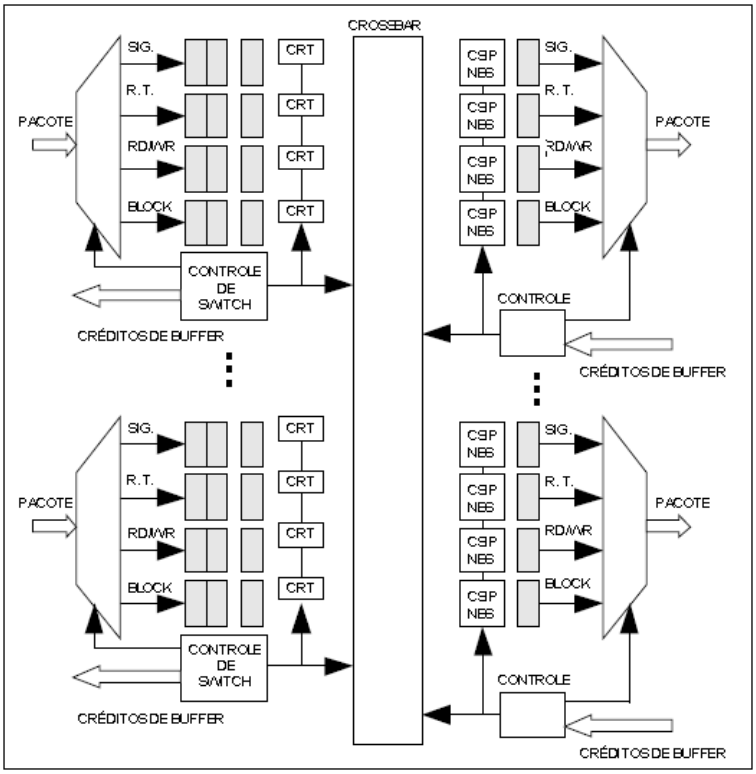
Cada um dos quatros níveis de serviço tem *buffers* separados, como mostra a Figura 14. Os *buffers* consideravelmente pequenos são alocados para cada nível de serviço, capaz de armazenar alguns poucos *flits*. O algoritmo XY é usado para o roteamento dos *flits* que é executado sempre que um novo pacote chega nos *buffers* de entrada. Enquanto o último *flit* do pacote não é recebido, processado e encaminhado ao seu destino, as informações de roteamento de cada nível de serviço para cada porta de entrada são armazenadas na tabela de roteamento CRT (*Current Routing Table*), de acordo com a Figura 14. Ao ser enviado um *flit* de uma porta de entrada para uma porta de saída, é disponibilizado um *buffer* e é enviado para o roteador um crédito de *buffer* (*buffer_credit*_SL), conforme Tabela 1.

Cada porta de saída de um roteador, por meio de um canal de comunicação, está conectada a uma porta de entrada do roteador seguinte. A porta de saída mantém a informação do espaço de armazenamento disponível em *buffer* para cada nível de serviço na próxima porta de entrada. Essa função é denominada crédito e é armazenada na unidade NBS (*Next Buffer State*). Ao transmitir um *flit* o número de créditos é decrementado e ao receber um crédito de *buffer*, do roteador seguinte, o número de créditos é incrementado. Havendo espaço livre para transmissão, são escalonados os *flits* de um *buffer* na porta de entrada pelo canal de saída que os transmite para rede.

O estado presente do escalonador *roud-robin* é armazenado no bloco CSIP (*Currently Serviced Input Port*). Esse número é incrementado quando a transmissão de um

pacote completo for finalizada ou se não tiver mais nenhum pacote do mesmo nível para ser transmitido de uma porta de entrada em particular.

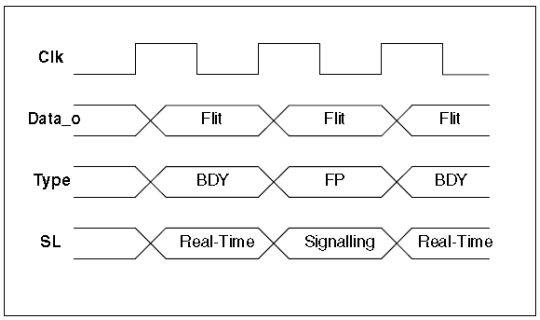
Figura 14 – Arquitetura do roteador QNoC



Fonte: Bolotin et al. (2003).

A regra de escalonamento na rede QNoC, determina que um *flit* pode ser transmitido por uma porta de saída, se o *buffer* da porta de entrada do próximo roteador estiver espaço disponível para receber dados e não haver nenhum pacote com serviço de maior prioridade para essa porta de saída. Caso um pacote de maior prioridade surgir em um dos canais de entrada, a transmissão do pacote é imediatamente interrompida para que seja enviado o pacote de maior prioridade. A transmissão do pacote de menor prioridade só será continuada e finalizada após o envio de todos os pacotes de maior prioridade. A Figura 15 mostra a interrupção na transmissão do pacote *Real-time* para o envio de outro com maior prioridade, nesse caso o pacote *Signalling*, que após concluído volta a transmitir o pacote *Real-time*, esse processo é conhecido por preempção.

Figura 15 – Formas de ondas de transmissão: preempção no pacote Real-Time



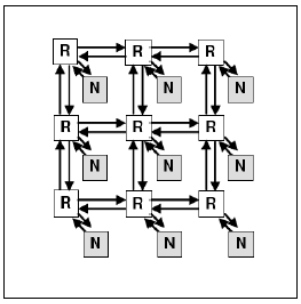
Fonte: Bolotin et al. (2003).

3.3 HERMES

3.3.1 Visão geral da arquitetura

Proposta por Moraes et al. (2004), na topologia da rede Hermes cada roteador tem um número diferente de portas, dependendo de sua posição em relação aos limites da rede, como mostrado na Figura 16. Para esse exemplo, o roteador que está no centro possui todas as cinco portas, já os que estão no canto da rede têm três portas. São representados na Figura 16 os roteadores e núcleos, denominados de R e N respectivamente.

Figura 16 – Estrutura de malha Hermes 3x3



Fonte: Mello (2006).

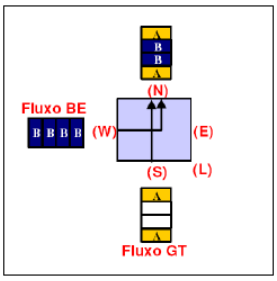
A rede Hermes foi concebida para atuar como uma rede de melhor esforço (*Best effort* - BE). Depois foram desenvolvidas quatro variações da rede Hermes com mecanismo de QoS (MELLO, 2006):

- Rede Hermes com chaveamento por circuitos (Hermes-CS);
- Rede Hermes com escalonamento baseado em prioridades fixas (Hermes-FP);
- Rede Hermes com escalonamento baseado em prioridades dinâmicas (Hermes-DP); e
- Rede Hermes com escalonamento baseado em taxas (Hermes-RB);

3.3.2 Visão geral do roteador com chaveamento de circuitos (CS)

A rede Hermes-CS diferencia fluxos através do estabelecimento de conexão. Aos fluxos com requisitos de QoS é oferecido serviço com vazão garantida (*Guaranteed Throughput* -GT), para os sem requisito de QoS é oferecido serviço BE. Essa rede utiliza dois canais virtuais, um transporta dados utilizando chaveamento por circuito, o outro utilizando chaveamento por pacote. A prioridade dos fluxos GT são maiores que as dos fluxos BE, com garantia de latência fim-a-fim. Fluxos BE podem trafegar no canal se nele não houver fluxo GT, ou seja, se o canal estiver livre, com isso o fluxo GT não é penalizado. Conforme é mostrado na Figura 17, na porta N o fluxo BE utiliza a largura de banda que o fluxo GT não ocupou.

Figura 17 – Largura de banda compartilhada no roteador Hermes-SC



Fonte: Mello (2006).

Para transmissão de dados de um fluxo GT é necessário estabelecer primeiro a conexão. Uma conexão entre os núcleos, origem e destino, requer a reserva do canal virtual ao longo do caminho entre esses roteadores. No entanto, são evitadas outras conexões por essa

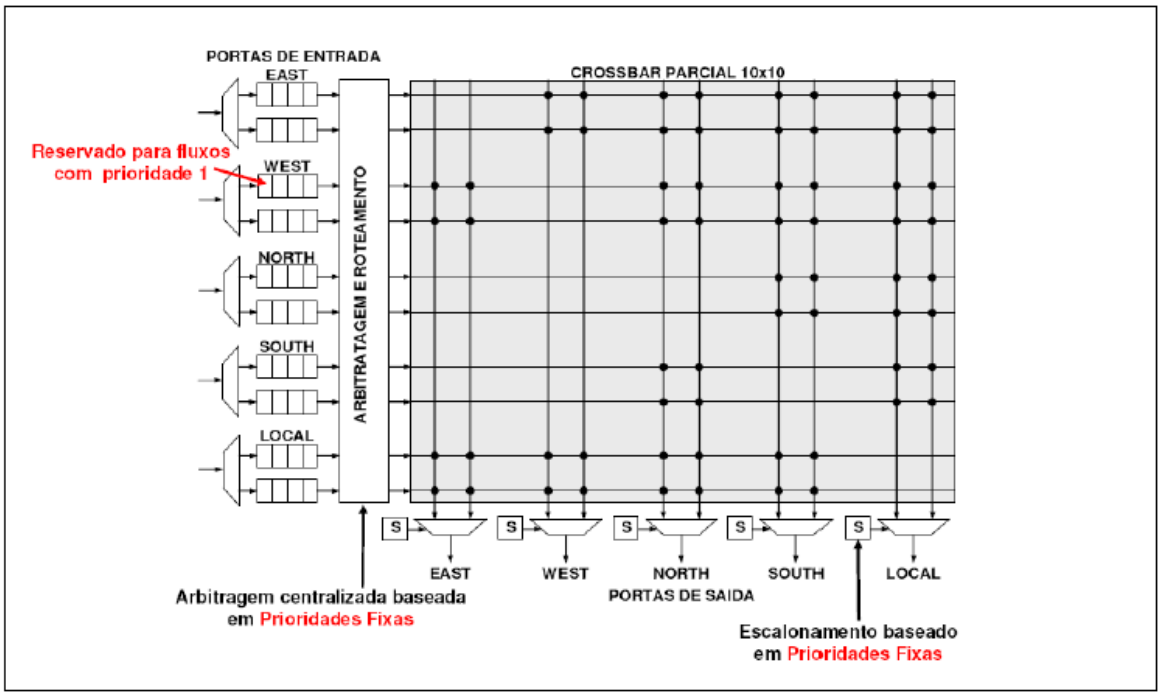
reserva de caminho. Os passos usados pelo protocolo para transmitir dados dos fluxos GT são descritos abaixo:

1. Solicita-se a criação de uma conexão com o núcleo origem enviando um pacote BE de controle, através do canal virtual para fluxos BE, para o núcleo destino;
2. Quando chega o pacote de controle no núcleo destino é ativado um sinal de aceitação (ACK). Esse sinal é propagado até o núcleo origem em *r* ciclos de relógio, no qual *r* é o número de roteadores no trajeto da conexão (incluem-se os roteadores de origem e destino);
3. No momento em que o sinal ACK chega ao núcleo origem, a conexão é estabelecida e os dados já podem ser transmitidos pelo canal virtual para fluxos GT;
4. Não havendo mais dados a ser transmitidos pelo núcleo de origem, é enviado por ele um pacote BE de controle, através do canal virtual para fluxos BE, solicitando a remoção da conexão; e
5. Chegando o pacote no núcleo de destino, é desativado o sinal de ACK e essa informação se propaga até a origem em *r* ciclos de relógio. A conexão é removida assim que o sinal ACK chega no núcleo origem.

3.3.3 Visão geral do roteador com Prioridades Fixas (FP)

A rede Hermes-FP trata a diferenciação dos fluxos de acordo com suas necessidades de desempenho, utilizando mecanismo de alocação de recursos baseado em prioridades fixas, de maneira similar à rede QNoC. É ilustrado na Figura 18 o roteador para esta rede.

Figura 18 – Arquitetura do roteador Hermes-FP



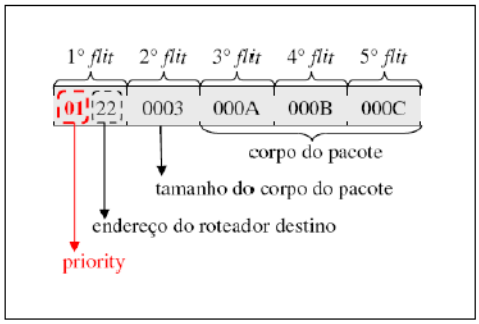
Fonte: Mello (2006).

Nesta rede é associada uma paridade estática a cada canal virtual que é servido de acordo com essa prioridade. Sendo que, a prioridade de cada canal virtual é dada por seu índice, conforme Equação 1. Assim essa rede é capaz de diferenciar até *n* fluxos, em que *n* é o número de canais virtuais por canal físico.

$$\text{Prioridade de } C_i = i - 1 \qquad (1)$$

Para que sejam diferenciados os fluxos, um novo campo denominado *priority* é inserido na parte alta do primeiro *flit* do pacote, como mostra a Figura 19. Através desse campo determina-se qual canal virtual é usado para transmissão do pacote. Por exemplo, são transmitidos pacotes com prioridade 1 pelo canal C2. Um valor entre zero e (*n*-1) deve ser atribuído ao campo *priority* pelo usuário, em que zero é atribuído aos pacotes de menor prioridade e (*n*-1) aos de maior prioridade. O campo *priority* é verificado apenas no roteador origem. Nos outros roteadores são transmitidos os pacotes usando o mesmo canal virtual alocado pelo roteador origem.

Figura 19 – Campos dos pacotes transmitidos pela Hermes-FP



Fonte: Mello (2006).

Atribuir prioridades fixas aos canais virtuais exige uma política de arbitragem e escalonamento do roteador. O pacote atribuído ao canal virtual com prioridade maior é entregue primeiro, não importando se os outros pacotes já estão esperando há mais tempo. Portanto, a transmissão dos dados dos canais virtuais com prioridade menor fica dependendo da carga dos canais virtuais com prioridade maior que pode variar dinamicamente. Por esse motivo, não podem ser determinados a todos os pacotes os limites de latência fim-a-fim, somente para os pacotes transmitidos pelo canal virtual com prioridade mais alta. Para o caso de pacotes com a mesma prioridade competirem pelos mesmos recursos, o mecanismo de prioridades fixas não prove garantias rígidas a nenhum dos pacotes.

3.3.4 Visão geral do roteador com Prioridades Dinâmicas (DP)

Na rede Hermes-DP é utilizado um mecanismo de alocação de recursos com base em prioridades dinâmicas para fornecer serviços diferenciados aos fluxos. Esse mecanismo, assim como a rede Hermes-FP, não precisa de alterações na interface do roteador.

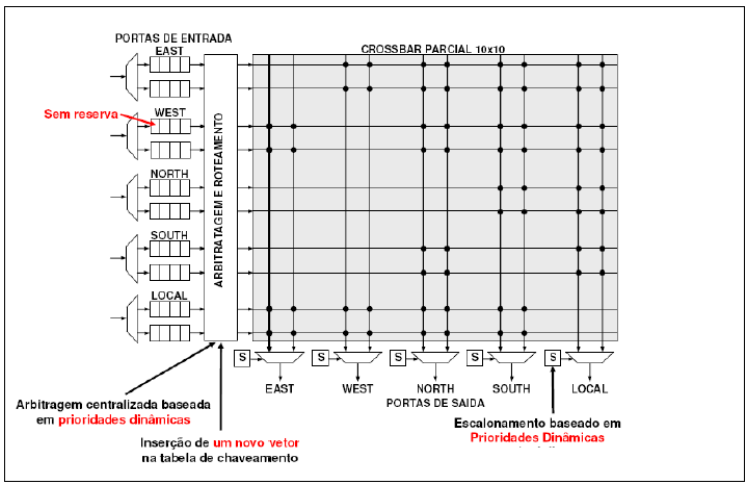
Nesta rede, a prioridade do canal virtual é variável, pois muda de acordo com a prioridade do pacote que o mesmo transmite. Isso permite:

- Que a transmissão de pacotes seja por qualquer canal livre;
- Que os pacotes sejam transmitidos por canais virtuais diferentes em roteadores diferentes; e
- Que pacotes de mesma prioridade sejam transmitidos através de canais virtuais diferentes em um mesmo roteador.

Nessa estrutura, o campo *priority* também está no cabeçalho do pacote. Entretanto, nesse campo, o usuário pode atribuir um valor entre zero e ($2t$ – 1), sendo t a largura em bits de um *flit*.

A alocação de recursos com base em prioridades dinâmicas necessita de modificações na política de arbitragem, no roteamento e no escalonamento do roteador, como é mostrado na Figura 20.

Figura 20 – Estrutura do roteador Hermes-DP



Fonte: Mello (2006).

No escalonamento, verifica-se o vetor *priority* para determinar qual dos canais virtuais tem o pacote com maior prioridade. Caso haja empate, um dos seguintes critérios é adotado: (i) *Round-robin*, em que x pacotes com maior prioridade recebem $1/x$ da largura de banda do canal físico; ou (ii) é entregue 100% da largura de banda do canal físico a um dos pacotes com maior prioridade. Esse critério que o escalonador utiliza é parametrizável e deve ser definida na fase de projeto. Pode acontecer o empate, pois pacotes de igual prioridade podem ser transmitidos por canais virtuais diferentes, sendo que isso não é possível na rede Hermes-FP.

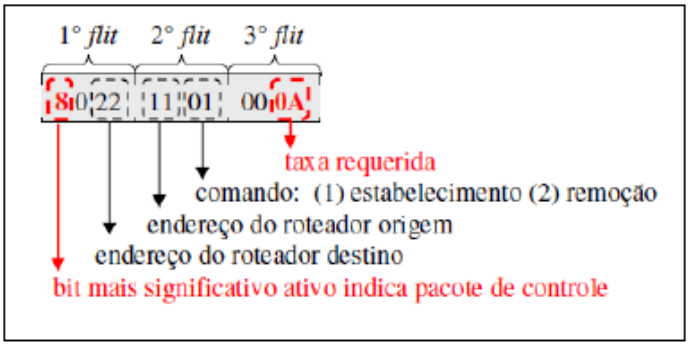
3.3.5 Visão geral do roteador com escalonamento baseado em taxas (RB)

Nesta rede, foi proposta por Mello (2006), a utilização de mecanismos de alocação de recurso com base em taxas, chamado Hermes-RB. Com esse mecanismo define-

se a prioridade de cada fluxo dinamicamente, conforme a taxa requerida e a taxa usada pelo mesmo. A política de escalonamento que se baseia em taxas, constitui duas etapas: controle de admissão seguido por um escalonamento dinâmico.

O processo de admissão determina se a rede pode comportar um novo fluxo sem colocar em risco as garantias de desempenho dadas a outros fluxo QoS. O processo de admissão inicia pelo envio de um pacote de controle do roteador origem para o roteador destino, contendo a taxa requeria pelo núcleo, conforme representado na figura 21. O fluxo QoS só é admitido, caso todos os roteadores no caminho até o destino transmitirem a taxa requerida. Após um pacote de controle ao chegar no destino, é propagado até o roteador origem um sinal de confirmação (ACK).

Figura 21 – Campos do pacote de controle



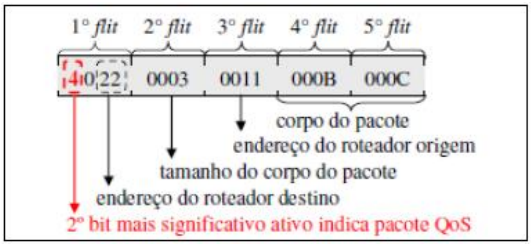
Fonte: Mello (2006).

Ao ser admitido um fluxo QoS, uma conexão virtual é estabelecida entre o roteador origem e o roteador destino. Essa conexão está relacionada a uma linha na tabela de fluxos, mostrado na Figura 23, de cada roteador no caminho da conexão. Cada linha da tabela de fluxos identifica um fluxo QoS utilizando os seguintes campos: roteador origem, roteador destino, taxa requerida e taxa usada. A quantidade de fluxos QoS simultâneos admitidos pelos roteadores é determinada pela profundidade da tabela de fluxos. Depois de completada a transmissão, é removida a conexão virtual pelo roteador origem pelo envio de outro pacote de controle.

Estabelecido o caminho virtual, o roteador origem pode iniciar o envio dos pacotes de dados do fluxo QoS, conforme representado na Figura 22. Os pacotes de dados ao chegarem numa porta de entrada de um roteador são armazenados nos *buffers* de entrada,

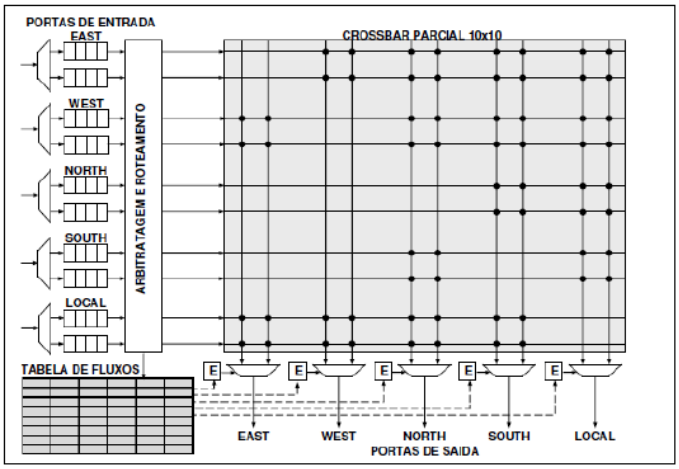
depois arbitrados e por fim roteados para uma saída (Figura 23). Para casos de pacotes roteados para a mesma porta de saída a entrega é feita de acordo com a política de escalonamento implementada.

Figura 22 – Campos do pacote de dados



Fonte: Mello (2006).

Figura 23 – Arquitetura do roteador para suportar escalonamento baseado em taxas



Fonte: Mello (2006).

Na política de escalonamento implementada, os fluxos BE só são transmitidos se não houver fluxo QoS requerendo o canal físico. Quando há uma competição com mais de dois fluxos, o que tem maior prioridade é escalonado primeiro. A prioridade de um fluxo QoS é a diferença entre as taxas requerida e atualmente usada pelo fluxo QoS. Sendo que essa prioridade é atualizada periodicamente conforme equação 2.

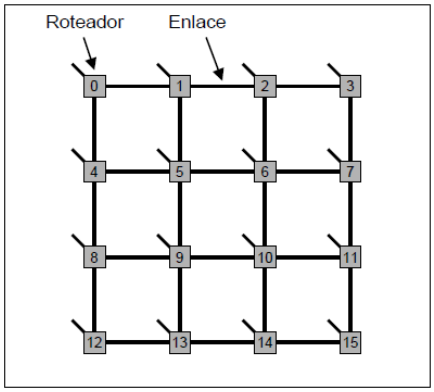
$$Prioridade = taxa\ requerida - taxa\ usada \qquad (2)$$

3.4 REDE SOCIN

3.4.1 Topologia

A topologia de referência usada na rede SoCIN é a grelha 2-D, mostrada na Figura 24. Os roteadores são representados na figura pelos blocos numerados, os enlaces pelas linhas que interligam os roteadores. No canto superior esquerdo da caixinha há uma linha que representa um enlace terminal para conexão de um núcleo à rede. A escolha dessa topologia foi devido à facilidade de implementação com as tecnologias de fabricação corrente (ZEFERINO, 2003, p.116).

Figura 24 – Topologia em grelha 2-D



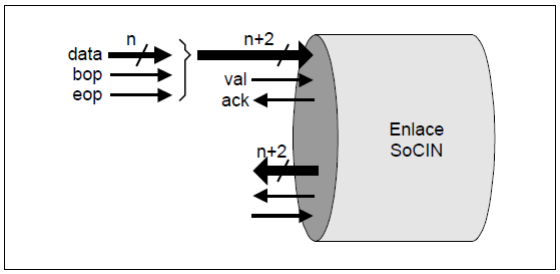
Fonte: Zeferino (2003, p. 116).

3.4.2 Enlace

Na rede SoCIN, os enlaces possuem dois canais unidirecionais *simplex* (Figura 25). Cada um é formado por *n* bits de dados e dois bits de banda lateral, o que implica uma largura física do canal (*phit*) de valor *n*+2. Os dois bits da banda lateral são os sinais marcadores de início do pacote (bop – *begin-of-packet*) e de fim de pacote (eop – *end-of-packet*) que servem para enquadramento do pacote. O parâmetro *n* é definido em função dos requisitos do sistema necessário para o transporte dos dados do pacote, podendo incluir sinais adicionais a serem manipulados por um protocolo de mais alto nível. A implementação de sinalizadores de paridade e de erro de paridade são exemplos desses sinais. Cada canal

também inclui, além dos *n*+2 bits, os sinais *val* e *ack*, sendo que eles não são contabilizados no cálculo do *phit* da rede, pois não atravessam os roteadores e não são armazenados em seus *buffers* (ZEFERINO, 2003, p.118).

Figura 25 – Enlace da rede SoCIN

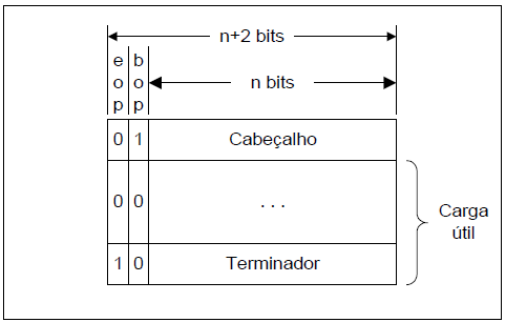


Fonte: Zeferino (2003, p. 118).

3.4.3 Formato do pacote

Na rede SoCIN o pacote é formado por vários *flits* (cada *flit* tem o tamanho de um *phit*), dos quais, o primeiro é o cabeçalho, o último é o terminador do pacote e os outros são as informações ou carga útil. Um *flit* contém *n*+2 bits, sendo que, *n* bits são dos dados e os dois, que estão sendo somados, são das indicações do início e fim do pacote, esses são para enquadramento e são denominados de *bop* (*begin-of-packet*) e *eop* (*end-of-packet*) (ZEFERINO, 2003). O formato do pacote da rede SoCIN é representado pela Figura 26.

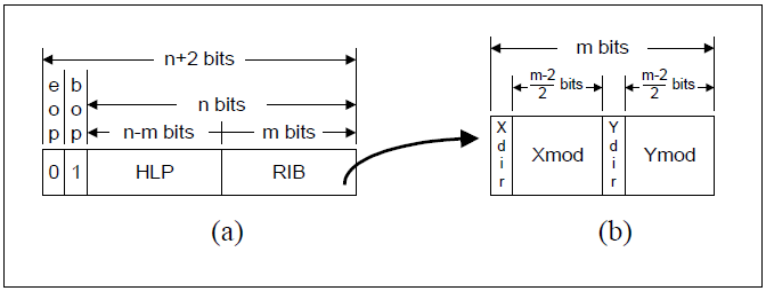
Figura 26 – Formato do pacote da rede SoCIN



Fonte: Zeferino (2003).

No cabeçalho da mensagem, dos *n* bits que o compõe, *m* bits são reservados à informação de roteamento, que pode ser configurado em função do tamanho da rede a ser sintetizada, entretanto, *m* deve ser menor ou igual a *n*. Na rede SoCIN, esses bits são chamados de RIB (*Routing Information Bits*) e incluem campos relacionados ao deslocamento nas direções X e Y. Dos m bits do RIB, a quantidade (*m*-2)/2 é reservada aos campos Xmod e a mesma quantidade é reservada para Ymod. Os dois bits que sobraram são reservados para os campos Xdir e Ydir, conforme representado na Figura 27. O restante dos bits do cabeçalho é reservado ao protocolo de alto nível (HLP – *High Level Protocol*) dos adaptadores de comunicação; o roteador não manipula esses bits, que podem ser utilizados na identificação do emissor do pacote, identificação da *thread* que enviou o pacote em um processador *multithreaded* ou para implantar protocolos de detecção e sinalização de erro de paridade (ZEFERINO, 2003).

Figura 27 – Formato do cabeçalho do pacote SoCIN: (a) HLP e RIB; (b) campos do RIB

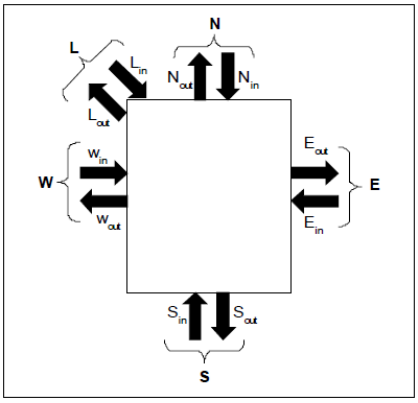


Fonte: Zeferino (2003).

3.4.4 Interface do roteador

Um dos elementos constituintes na rede SoCIN é o roteador ParIS (*Parameterizable Interconnect Switch*). Sua interface externa tem até cinco portas de comunicação bidirecionais identificadas por L (*Local* ou local), N (*North* ou norte), E (*East* ou leste), S (*South* ou sul) e W (*West* ou oeste), conforme é mostrado na Figura 28. Em cada porta existe dois canais unidirecionais, um de entrada e outro de saída, indicados pelos sufixos *in* e *out*, respectivamente. Por exemplo, na porta *East*, o canal de entrada é identificado por E*in* e a porta de saída por E*out*. A porta Local serve para interligar com um core ou um subsistema a rede, já as outras são para conexões inter-roteadores.

Figura 28 – Interfaces do roteador ParIS

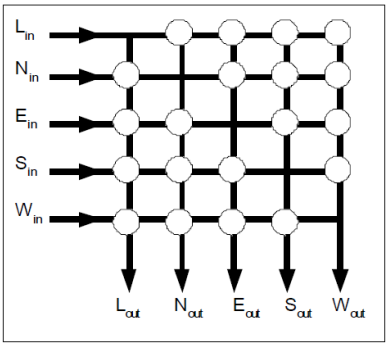


Fonte: Zeferino et al. (2004).

3.4.5 Conexões entre canais

As conexões entre os canais de entradas e saídas são estabelecidas através de uma matriz de chaveamento, como mostra a Figura 29. Os canais das entradas são representados pelas linhas e os das saídas pelas colunas. Os círculos que estão no cruzamento das linhas com as colunas, indicam a possibilidade de conexão entre os canais, quando da sua ausência indica que a conexão não é possível. Sendo assim, se um pacote chegar pelo canal de entrada L*in* não retornará pelo canal de saída L*out*, o mesmo ocorre para as outras portas. Isso porque é proibido um pacote retornar pelo canal de saída da mesma porta pelo qual entrou no roteador.

Figura 29 – Interfaces do roteador ParIS



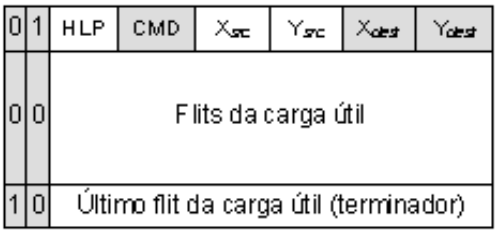
Fonte: Zeferino et al. (2004).

3.4.6 Chaveamento de circuitos na SoCIN – SoCIN-CS

De acordo com Berejuck (2009), o roteador ParIS tem por objetivo oferecer o chaveamento por circuito e por pacote de modo simultâneo, permitindo assim, que alguns fluxos de comunicação utilizem um ou outro modo de chaveamento.

Para implementar o chaveamento por circuito, foram selecionados dois bits no cabeçalho, extraídos do campo HLP (*Higher Level Protocol*). Para indicar o tipo de pacote sob o ponto de vista de chaveamento por circuito. O campo de dois bits, chamado CMD, permite identificar se o pacote é regular (00) ou se é um pacote de alocação (01), de liberação (10) ou de confirmação de circuito (11). A Figura 30 apresenta o cabeçalho modificado.

Figura 30 – Formato do pacote modificado



Fonte: Berejuck (2009).

No modo chaveamento por pacote, a comunicação entre dois nodos (origem e destino) é feita para que cada pacote aloque e libere canais de comunicações da rede, de roteador em roteador.

No modo chaveamento por circuitos, o estabelecimento e o cancelamento de um circuito são feitos através de pacotes com CMD igual a 01 (*Aloccate*) e a 10 (*Release*), respectivamente.

O custo de silício dessa implementação foi menor do que a versão original do roteador ParIS. Os valores são apresentados na tabela 2.

Tabela 2 – Comparativo sobre o custo de silício – chaveamento de circuitos

Recurso	Versão original do roteador ParIS	Versão com chaveamento de circuitos
# SLICES	832	817
# Flip-flops	636	628
# LUTs	1087	1058

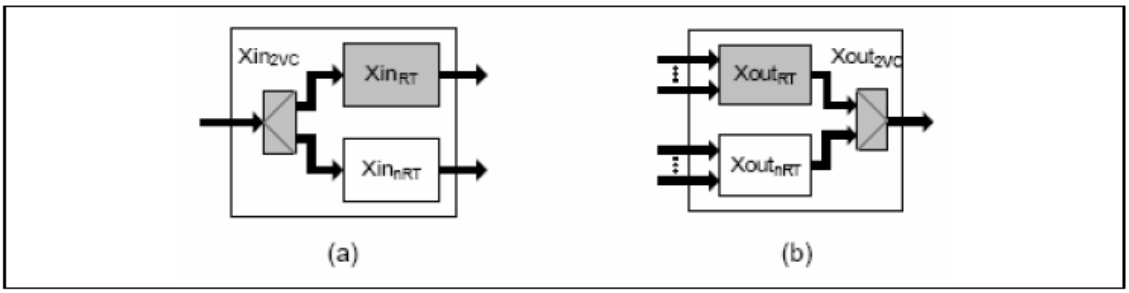
Fonte: Berejuck (2009).

3.4.7 Canais virtuais na rede SoCIN – SoCIN-VC

A técnica de canais virtuais na rede SoCIN permite que mecanismo de alocação de recursos sejam utilizados pela rede, possibilitando a diferenciação do tratamento de fluxos (BEREJUCK, 2009).

Nessa arquitetura foram incluídos dois circuitos de canais virtuais, sendo um para fluxos de melhor esforço (denominado nRT) e outro para fluxos com garantia de vazão (chamado de RT). A implementação dos canais virtuais é ilustrada pela Figura 31, a qual foi baseada na replicação dos blocos X_{in} e X_{out}, aqui representados por X_{in2VC} e X_{out2VC}. Na figura, observa-se que o bloco X_{in2VC} e X_{out2VC} são compostos por três blocos, o primeiro possui dois X_{in} e um bloco seletor, enquanto o segundo possui dois X_{out} e também um bloco seletor. Os novos blocos inseridos no roteador ParIS estão preenchidos com a cor cinza na Figura 31.

Figura 31 – Novos módulos com canais virtuais: (a) X_{in2VC}; e (b) X_{out2VC}

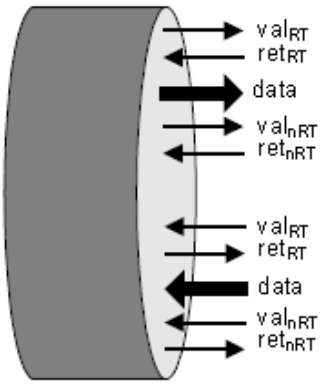


Fonte: Berejuck (2009).

Segundo Berejuck (2009), ao ser recebido um pacote no bloco X_{in2VC} de um roteador, o mesmo é encaminhado ao bloco X_{in} do canal selecionado (RT ou nRT). No canal de saída do roteador, quando existirem pacotes nos blocos X_{outRT} e X_{outnRT}, o primeiro irá sempre escalonado antes do segundo. No caso de haver um pacote nRT sendo encaminhado pelo canal de saída, na presença de um pacote RT, o primeiro será temporariamente preemptado pelo segundo.

O enlace original da rede SoCIN inclui dois canais físicos em sentidos contrários em que cada um é constituído por um canal de dados e dois sinais de controle de fluxo, o *val* e o *ret*. Para permitir e controlar a multiplexação de cada canal físico de dados por dois canais virtuais, foi necessário replicar dois sinais de controle de fluxo, no qual cada par de sinais (val_{RT} e ret_{RT}; e val_{nRT} e ret_{nRT}) foi alocado para cada canal virtual, conforme mostra a Figura 32 (BEREJUCK, 2009).

Figura 32 – Enlace da rede SoCIN-VC



Fonte: Berejuck (2009).

A Tabela 3 indica qual foi o custo de silício para implementar canais virtuais no roteador ParIS.

Tabela 3 – Comparativo sobre o custo de silício – canais virtuais

Recurso	Versão original do roteador ParIS	Versão com Canais Virtuais
# SLICES	832	1743
# Flip-flops	636	1441
# LUTs	1087	2173

Fonte: Berejuck (2009).

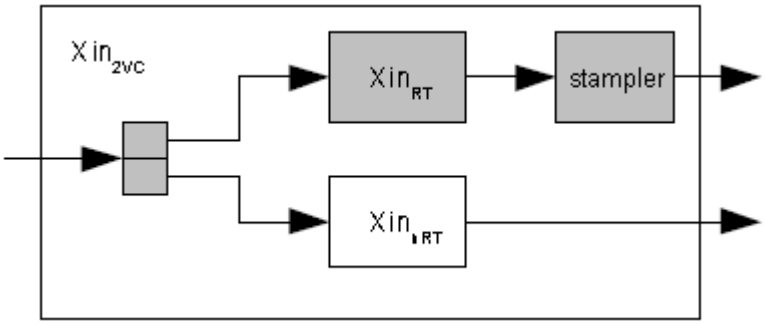
3.4.8 Envelhecimento de pacotes na rede SoCIN – SoCIN-AG

Na técnica de análise do envelhecimento (em inglês *aging*) de pacotes na rede SoCIN com priorização de fluxos, um pacote ao ser injetado na rede já sai com um nível de prioridade inicial, que é incrementado conforme seu tempo de permanência na rede, especialmente em situações de contenção. Dessa maneira, o pacote passa a ter mais oportunidades ao competir com pacotes da mesma classe injetados mais recentemente, diminuindo a latência para chegar ao destinatário (BEREJUCK, 2009).

De acordo com Berejuck (2009), a arquitetura da SoCIN-AG é baseada no roteador ParRIS com dois canais virtuais, um deles para fluxo de melhor esforço (nRT) e outro para fluxos com garantias de vazão (RT). A este, é adicionado um mecanismo de envelhecimento de pacotes e ao nRT de escalonamento baseado na idade.

Quando é criado um pacote RT é dado a ele um selo de idade (*age*) com valor inicial igual a 0 (zero). Este selo é incrementado periodicamente após chegar ao canal RT de um roteador e ficar aguardando para ser liberado. O incremento do selo (*stamp*) é feito pelo circuito *Stampler* adicionado ao bloco X_{inRT}, como mostrado na Figura 33.

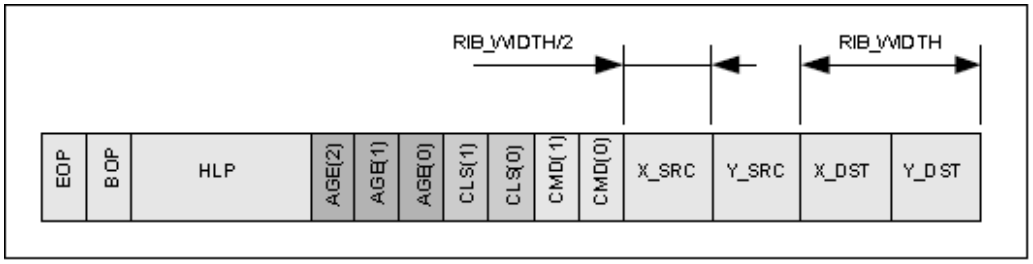
Figura 33 – Diagrama em blocos da arquitetura proposta com *aging* no roteador ParIS



Fonte: Berejuck (2009).

De modo equivalente ao que foi feito na técnica de chaveamento de circuitos, foram extraídos alguns bits do campo HLP para tratamento de *aging*: o campo AGE com três bits e o campo CLS de dois bits, ilustrados na Figura 34. O campo AGE é tratado nos novos blocos X_{in} e X_{out} do roteador ParIS, alterados para esta finalidade, e o bloco CLS é avaliado somente no novo bloco X_{out}.

Figura 34 – Formato do pacote modificado para *aging*



Fonte: Berejuck (2009).

O custo de silício para implementar essa arquitetura na rede SoCIN é apresentada na Tabela 4.

Tabela 4 – Comparativo sobre o custo de silício – envelhecimento de pacotes

Recurso	Versão original do roteador ParIS	Versão com *Aging*
# SLICES	832	1743
# Flip-flops	636	1441
# LUTs	1087	2173

Fonte: Berejuck (2009).

4 DESENVOLVIMENTO

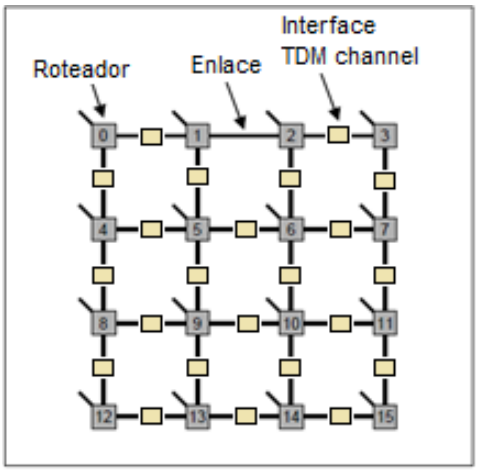
O trabalho teve como objetivo desenvolver um canal de comunicação na rede SoCIN, utilizando o conceito de TDM, para garantir na comunicação a previsibilidade de latência e também o cumprimento do prazo no envio dos pacotes (*deadline*), principalmente para sistemas de tempo real. Foi proposto um mecanismo, denominado *Interface_TDM_channel*, para ser implementado no enlace da rede SoCIN.

A solução baseia-se em estruturas criadas em linguagem VHDL (*VHSIC Hardware Description Language*) utilizando a ferramenta ISE® da Xilinx®. Nas subseções seguintes serão apresentadas as estruturas que compõe a *Interface_TDM_channel*. Na Subseção 4.1.6 é mostrado o resultado alcançado através das simulações.

4.1 *INTERFACE_TDM_CHANNEL*

Conforme já visto, na rede SoCIN os enlaces possuem dois canais unidirecionais *simplex*. Dessa forma, a transmissão de um determinado pacote se fará através de um único canal. Sendo assim, durante essa transmissão, o canal ficará totalmente ocupado, o que impedirá o envio de um próximo pacote. Tomem-se como exemplo dois pacotes genéricos A e B; se o pacote A estiver utilizando um determinado canal de comunicação e o pacote B requisitar o mesmo canal, então este último só poderá ser enviado após a liberação do canal pelo pacote A. Na busca de um proveito melhor deste canal, foi idealizada para rede SoCIN um enlace com mecanismo TDM, o que permitirá enviar até quatro pacotes, praticamente simultâneos, nesse mesmo canal. Para alcançar tal objetivo, foi criada a *Interface_TDM_Channel* que ficará disposta entre dois roteadores, como pode ser visto na Figura 35.

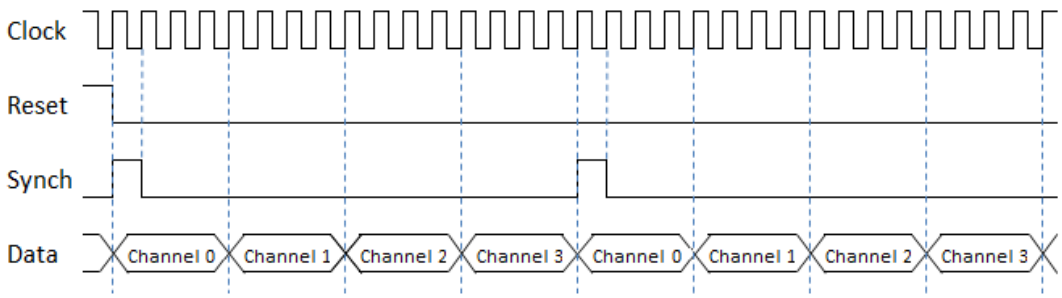
Figura 35 – Rede SoCIN com a *Interface_TDM_Channel*



Fonte: Próprio autor.

O uso de um mecanismo TDM no enlace da rede SoCIN, possibilitará o compartilhamento do canal para transmitir vários pacotes alocados no tempo. Por questão de simplicidade, iniciou-se a especificação deste TDM com apenas quatro canais. Futuramente, deverá ser empregado o uso de estruturas em linguagem VHDL para permitir que o projetista possa alterar a quantidade de canais dos links TDM antes da síntese da rede. Os sinais da Figura 36 mostram de que forma o canal será compartilhado. Esse mecanismo tem a função de compartilhar um único canal em quatro canais com o mesmo intervalo de tempo. Para tanto, será formado pelo sinal de *clock*, *synch* (*synchronism*, em português sincronismo), *data* (dados) e *reset*. Os sinais de *clock* e *reset* provirão da própria rede SoCIN. O sinal *synch*, gerado pela Interface, acontecerá a cada dezesseis pulsos de *clock* e permanecerá em nível lógico alto durante um ciclo, ele indica o início do canal 0 (zero). Já o sinal *data* são os dados transmitidos nos canais. Cada canal ficará habilitado durante quatro ciclos de *clock*, caso houver dados para serem enviados nesse canal, deverão ser transmitidos nesse intervalo de tempo.

Figura 36 – Sinais do mecanismo TDM



Fonte: Próprio autor.

Com a proposta de conseguir tornar o canal da rede SoCIN multiplexado no tempo, idealizou-se a criação da *Interface_TDM_Channel*, conforme ilustra a Figura 37. A interface é composta por quatro blocos principais, são eles: *Router_Interface_input* (Interface de entrada para o roteador origem), *Route_Interface_output* (Interface de saída para o roteador destino), *Connection_Table* (Tabela de conexão) e *Clock_Manager* (Gerenciador de *Clock*). Os blocos *Router_Interface_input*, *Router_Interface_output* e *Connection_Table* são duplicados, pois existem no enlace dois canais de sentidos contrários entre si.

Supõe-se uma comunicação entre dois roteadores vizinhos da Figura 35, o roteador 6 e o 10 por exemplo. Num sentido, o roteador 6 é origem e o 10 é destino, noutro o contrário. Sendo assim, para o conjunto formado pelos blocos tracejados da Figura 37, é estabelecida a comunicação entre os roteadores 6 (origem) e 10 (destino). Para o sentido contrário, roteador 10 (origem) e 6 (destino), a comunicação estabelecer-se-á através do conjunto formado pelos blocos superiores da Figura 37.

A função do bloco *Router_Interface_input* é receber os *flits* enviados pelo roteador origem e consultar o *Connection Table* para saber se o canal no qual pretende-se transmitir está disponível. Caso sim, sendo o primeiro *flit*, identificado pelo *bop*, será enviado um comando ao *Connection_Table* para ocupar o canal, que ficará nesse estado até o recebimento do último *flit*, identificado através do *eop*. Após recebimento deste, essa interface irá aguardar a confirmação de recebimento vindo do roteador destino para desocupar o canal. Para cada canal será realizado esse mecanismo.

O bloco *Router_Interface_output*, encaminha os *flits* e comandos para o roteador de destino, sendo também responsável em repassar ao *Router_Interface_input* as confirmações de recebimento de cada *flit* informada pelo roteador destino.

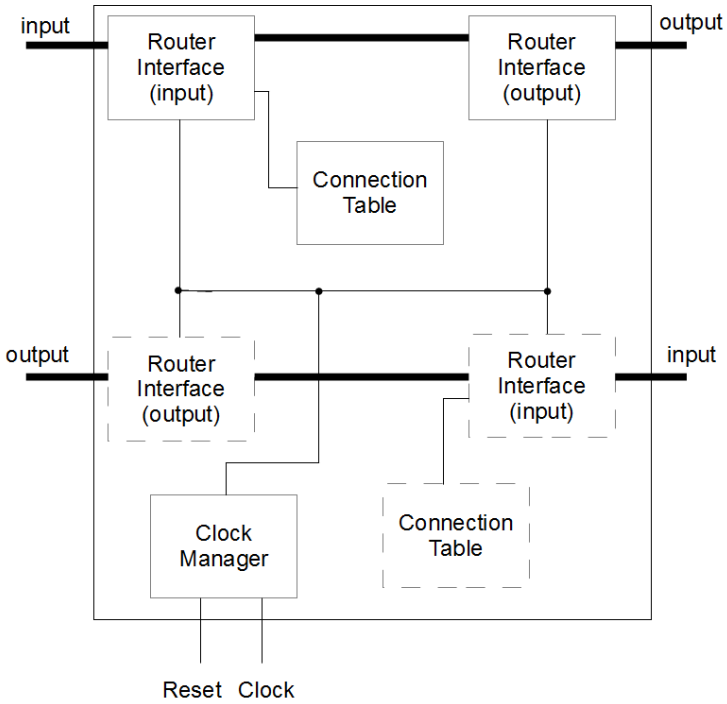
O *Connection_Table* tem como finalidade indicar ao *Router_Interface_input* os estados de cada canal, se estão ocupados ou disponíveis.

O *Clock_Manager* é o propulsor do sincronismo para toda a *Interface_TDM_Channel*.

De maneira resumida, a Interface percebe que o roteador origem deseja enviar os *flits* quando o sinal *val* está com nível lógico 1. A próxima etapa da Interface é disponibilizar o *flit* e informar ao roteador destino que tem um *flit* para ser enviado. Após a confirmação de recebimento do *flit* pelo roteador destino, através do sinal *ret* igual a 1, a Interface envia a confirmação de recebimento para o roteador origem.

Nas próximas Subseções será apresentado o funcionamento de cada bloco pertencente à Interface.

Figura 37 – *Interface_TDM_Channel*

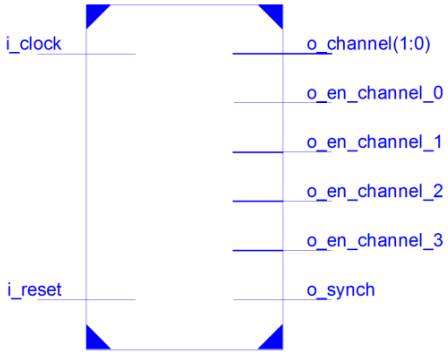


Fonte: Próprio autor.

4.1.1 *Clock_Manager*

Esse componente é o responsável pelo sincronismo da *Interface_TDM_Channel*. O *Clock_Manager* é apresentado na Figura 38. Nele, têm-se as entradas de *clock* e *reset* e as saídas *channel* (canal), *en_channel*_0 (*enable_channel*_0, em português habilita_canal_0), *en_channel*_1, *en_channel*_2, *en_channel*_3 e *synch* (sincronismo).

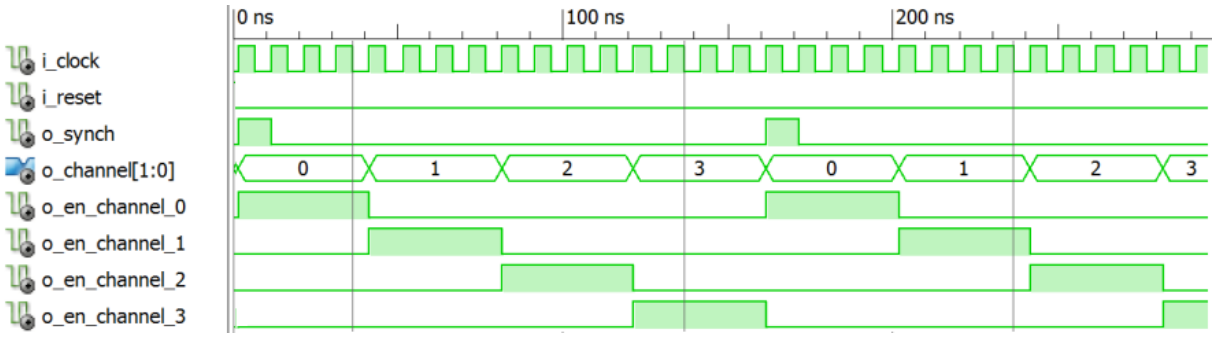
Figura 38 – *Clock_Manager*



Fonte: Próprio autor.

Os sinais do *Clock_Manager* podem ser observados através da Figura 39. Cada canal, representado pelo sinal *o_channel*, tem duração de quatro ciclos de *clock*. O sinal *o_en_channel_0* corresponde ao *enable* do canal 0 (zero), já *o_en_channel_1* corresponde ao *enable* do canal 1 e assim sucessivamente, sendo que, cada um fica ativo (nível lógico 1) durante os quatro ciclos de *clock* do canal ao qual está associado, por exemplo: O *o_en_channel_0* permanece com o nível lógico 1 enquanto o canal 0 (zero) está presente. O sinal *synch* indica início do canal 0 (zero).

Figura 39 – Sinais do *Clock_Manager*

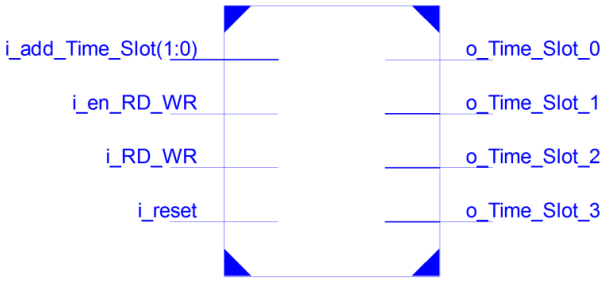


Fonte: Próprio autor.

4.1.2 *Connection_Table*

O componente *Connection_Table*, representado pela Figura 40, indica para o *Router_Interface_input* o estado em que se encontra cada canal, se disponível ou ocupado. Sua estrutura é composta pelos sinais de entrada: *i_add_Time_Slot*, *i_en_RD_WR*, *i_RD_WR* e *i_reset*, e pelos sinais de saída: *o_Time_Slot_0*, *o_Time_Slot_1*, *o_Time_Slot_2* e *o_Time_Slot_3*.

Figura 40 – *Connection_Table*

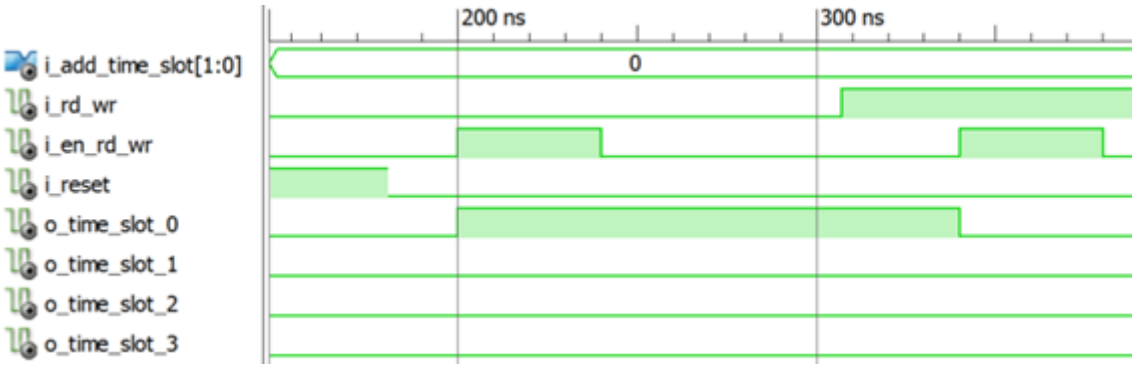


Fonte: Próprio autor.

O funcionamento desse componente é representado pelos sinais da Figura 41. O *Connection_Table* é sensível ao sinal *i_en_RD_WR*. Os sinais de saída são os indicadores de canais ocupados ou livres. Sempre que é necessário alterar seu estado lógico, é realizado um

comando através do *i_en_rd_wr*. O comando é proveniente do *Router_Interface_input*. Inicialmente todas as saídas estão com seus níveis iguais a 0 (zero), ou seja, canais disponíveis. Assim que for requisitada a ocupação de um dos canais, o *Router_Interface_input* gera um pulso no sinal *i_en_rd_wr* e coloca *o i_rd_wr* para 0 (zero), condição de ocupação. Para deixar novamente o canal disponível, é necessário gerar um pulso no sinal *i_en_rd_wr* e colocar o *i_rd_wr* para 1. O sinal *i_add_time_slot* é a referência do componente para saber o canal que está presente no momento de uma solicitação. A Figura 41 exemplifica uma situação de ocupação e desocupação do canal 0 (zero). Conforme já descrito, ao gerar o pulso no sinal *i_en_rd_wr* e colocando o *i_rd_wr* em nível lógico 0 (zero) será ativada uma das saídas. Como o canal presente no momento é o 0 (zero) a saída ativada será *o_time_slot_0*. A mesma permanecerá ativa, nível lógico 1, até que uma nova solicitação de desocupação seja feita.

Figura 41 – Sinais do *Connection_Table*



Fonte: Próprio autor.

4.1.3 *Router_Interface_input*

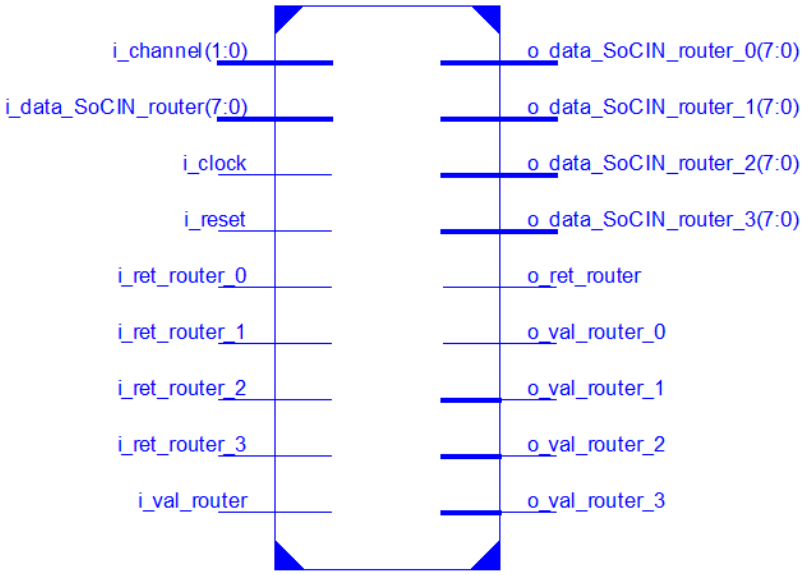
O *Router_Interface_input* é conectado diretamente ao roteador de origem da rede SoCIN. Esse componente monitora constantemente o sinal *val* enviado por esse roteador. Quando o roteador origem quer enviar um determinado pacote, faz uma requisição ativando o sinal *val* em nível lógico 1. Dessa maneira, o *Router_Interface_input* percebe uma mudança de nível no sinal *val* (de 0 (zero) para 1), então a interface verifica em qual canal foi realizada a requisição e também se o mesmo está disponível. Caso estiver ocupado, mesmo que os outros estejam disponíveis, a alocação do canal não será possível. Para trabalho futuro, o

projetista poderá implementar, através da linguagem VHDL, uma melhoria de alocação dinâmica para esta interface, ou seja, se no momento da requisição houver canais disponíveis, um desses será alocado para a transmissão do pacote, conseguindo assim, uma melhor usabilidade. A estrutura desse componente é formada por três blocos, o *DEMUX*, *Data_SoCIN_Dr* e *Read_Write_MUX*. A seguir serão apresentados cada um deles.

4.1.3.1 *DEMUX*

O *DEMUX*, representado pela Figura 42, funciona como um seletor de canais. O sinal de entrada *i_channel* informa para o bloco qual canal está presente em cada momento, se é o canal 0, 1, 2 ou 3. Os sinais *i_data_SoCIN_router* e *i_val_router* contêm os dados enviados pelo roteador origem da rede SoCIN. Os sinais *i_ret_router_0* até *i_ret_router_3* recebem as informações do *Router_Interface_input*, já o *i_clock* e *i_reset* são provenientes do *Clock_Manager*. Os sinais de saída *o_data_SoCIN_router_0* até *o_data_SoCIN_router_3* e *o_val_router_0* até *o_val_router_3* correspondem aos dados de cada canal, por exemplo: *o_data_SoCIN_router_1* e *o_val_router_1* são sinais pertencentes ao canal 1, *o_data_SoCIN_router_2* e *o_val_router_2* ao canal 2, e assim para os outros. O sinal *o_ret_router* é conectado ao roteador origem.

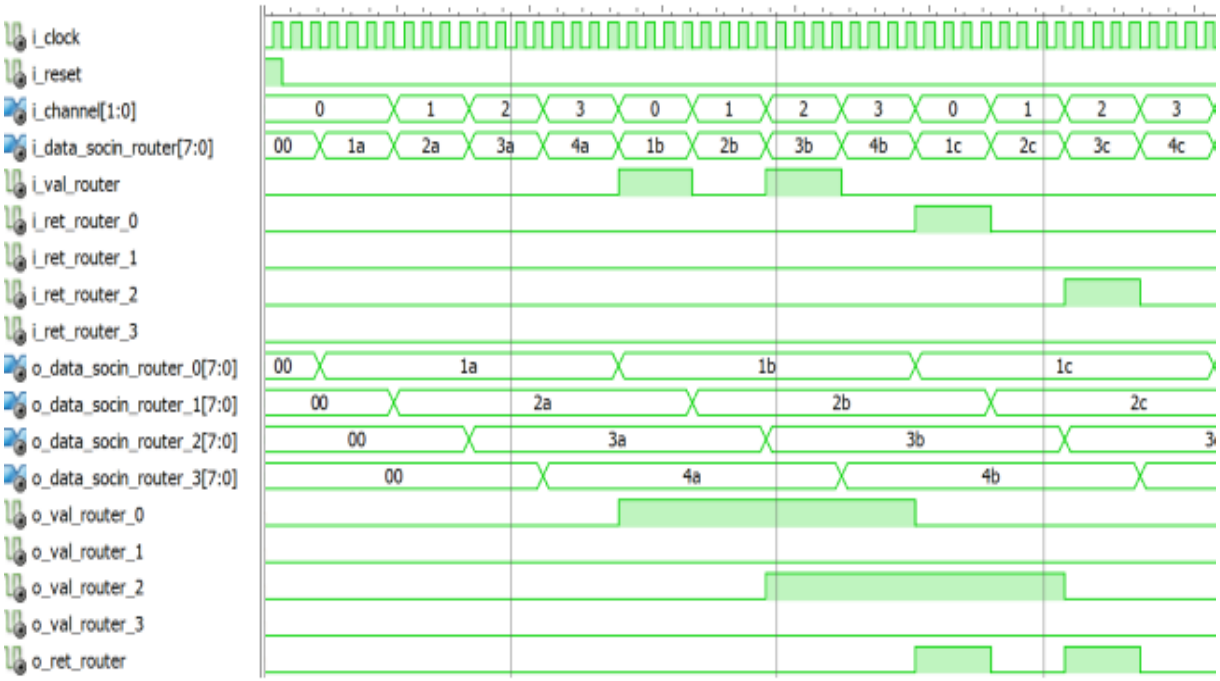
Figura 42 – *DEMUX*



Fonte: Próprio autor.

A Figura 43 demonstra o funcionamento do *DEMUX*. É importante informar que no final de cada nome dado aos sinais, consta o número do canal, como exemplo: o sinal *o_data_SoCIN_router_3* indica que ele pertence ao canal 3. Conforme pode ser observado na Figura 43, para cada canal tem uma informação contida no sinal *i_data_SoCIN_router*, no canal 0 (zero) tem a informação "1a", no 1 a informação "2a", no 2 tem "3a" e assim por diante. Pode-se reparar que a informação que aparece no sinal de saída *o_data_SoCIN_router_0* é o mesmo que está no sinal *i_data_SoCIN_router* durante o intervalo de tempo destinado ao canal 0, já a informação que aparece no sinal de saída *o_data_SoCIN_router_1* é igual ao que está no *i_data_SoCIN_router* durante o intervalo de tempo destinado ao canal 1, o mesmo se aplica para os canais 2 e 3. Isso ocorre pois, como já citado acima, o *DEMUX* funciona como um seletor de canais, ou seja, no intervalo de tempo pertencente a cada canal, a saída que receberá o que está na entrada será aquela correspondente ao canal de momento. Exemplo: o sinal *o_val_router_2* só receberá a informação que está *no i_val_router* quando o intervalo de tempo pertencer ao canal 2. De acordo com a Figura 43, o *i_val_router* tem seu valor igual a 1 em dois momentos, um no canal 0 (zero) e outro no canal 2. Quando no canal 0 (zero), a saída *o_val_router_0* recebe esse valor, já no caso do *o_val_router_2*, ele recebe o valor lógico 1 no intervalo de tempo que é destinado ao canal 2. Em ambos os casos, os valores voltam a ser iguais a 0 (zero) no próximo intervalo de tempo de cada canal. Para o sinal *o_ret_router* acontece diferente, no intervalo de tempo alocado para o canal 0 (zero), ele receberá a informação retida no sinal de entrada *i_ret_router_0*, no intervalo de tempo do canal 1 será recebido a informação presente no *i_ret_router_1*, da mesma maneira para os canais 2 e 3.

Figura 43 – Sinais do *DEMUX*

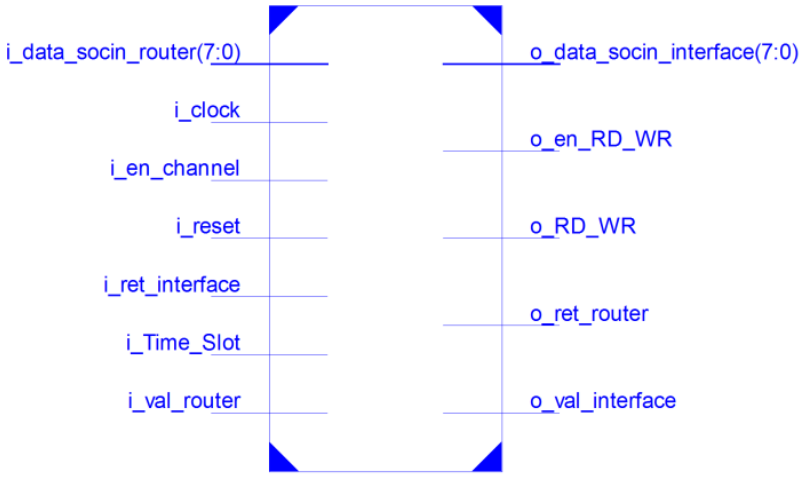


Fonte: Próprio autor.

4.1.3.2 *DATA_SOCIN_Dr*

Para a *interface_TDM_channel* serão utilizados, para cada um dos sentidos no enlace de comunicação, quatro blocos *Data_SoCIN_Dr* (Figura 44), sendo um para cada canal. Esse componente serve como um registrador de cada canal e tem em sua entrada os seguintes sinais: *i_data_SoCIN_router*, *i_clock*, *i_en_channel*, *i_reset*, *i_ret_interface*, *i_Time_Slot* e *i_val_router*. Já sua interface de saída é composta pelos sinais: *o_data_SoCIN_router*, *o_en_RD_WR*, *o_RD_WR*, *o_ret_router* e *o_val_interface*.

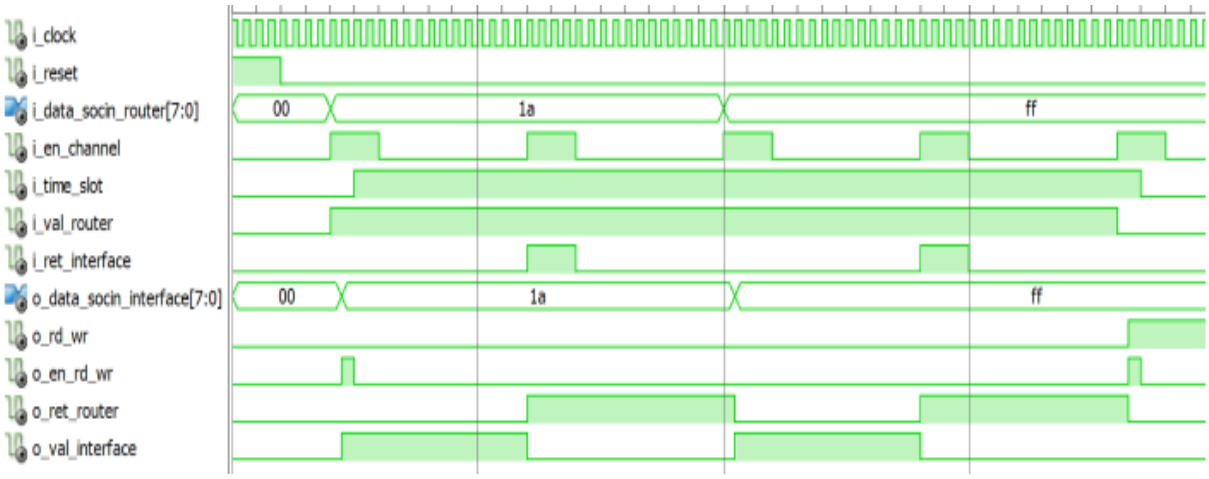
Figura 44 – *Data_SoCIN_Dr*



Fonte: Próprio autor.

Os sinais da Figura 45 mostram de que maneira funcionará o bloco *Data_SoCIN_Dr*. Para explicar o funcionamento da forma mais simples, foi feita uma simulação considerando o envio de dois *flits*, um *bop* (1a) e outro *eop* (ff). Todos os eventos são realizados somente quando o *i_en_channel* estiver com seu nível lógico igual a 1. O envio do primeiro *flit* (1a), presente no *i_data_SoCIN_router*, foi realizado no primeiro pulso do *i_en_channel* com *o i_val_router* em nível lógico 1. Para que o envio do primeiro *flit* seja possível, o *i_time_slot* deve estar com o valor igual a 0 (zero). Após isso, é feita a transferência do *flit* (1a) para o sinal de saída *o_data_socin_interface*, gerado um pulso no *o_en_rd_wr*, colocado *o_rd_wr* em nível lógico 0 (zero) e ativado o *o_val_interface* para nível lógico igual a 1, que fica nesse estado até a confirmação de recebimento do último *flit* enviado, indicado pelo sinal de entrada *i_ret_interface*. Recebida essa confirmação, é gerado um pulso em *o_ret_router* e aguardado o próximo *flit*. Quando é enviado o segundo e último *flit* (ff), todas as etapas se repetem até o recebimento de confirmação, ou seja, *i_ret_interface* igual a 1. Depois dela, é gerado um pulso no *o_en_rd_wr* e colocado o *o_rd_wr* para nível lógico 1, com isso, é finalizado o envio deste pacote. O *i_clock* e *i_reset* são oriundos do *Clock_Manager*.

Figura 45 – Sinais do *Data_SoCIN_Dr*

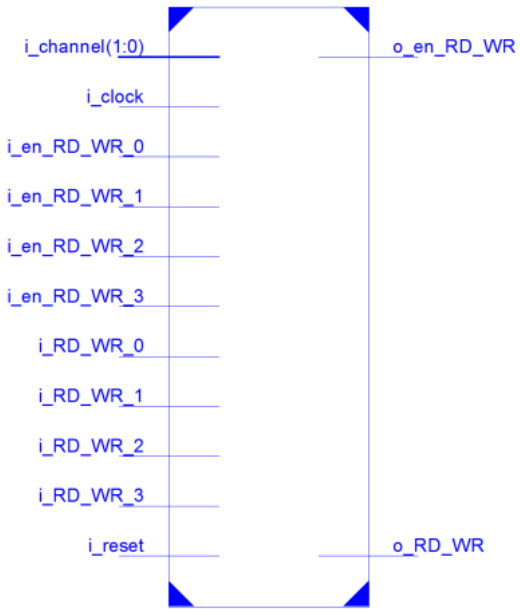


Fonte: Próprio autor.

4.1.3.3 *READ_WRITE_MUX*

O componente *Read_Write_MUX* (Figura 46) faz a multiplexação dos sinais que estão na sua entrada, que são: *i_en_RD_WR_0* até *i_en_RD_WR_3* e *i_RD_WR_0* até *i_RD_WR_0*. O sinal *i_channel* funciona como seletor dos sinais. O *i_clock* e *i_reset* são originários do *Clock_Manager*. Os sinais da saída *o_en_RD_WR* e *o_RD_WR* determinam para o *Router_Interface_output* a ocupação ou não de cada canal.

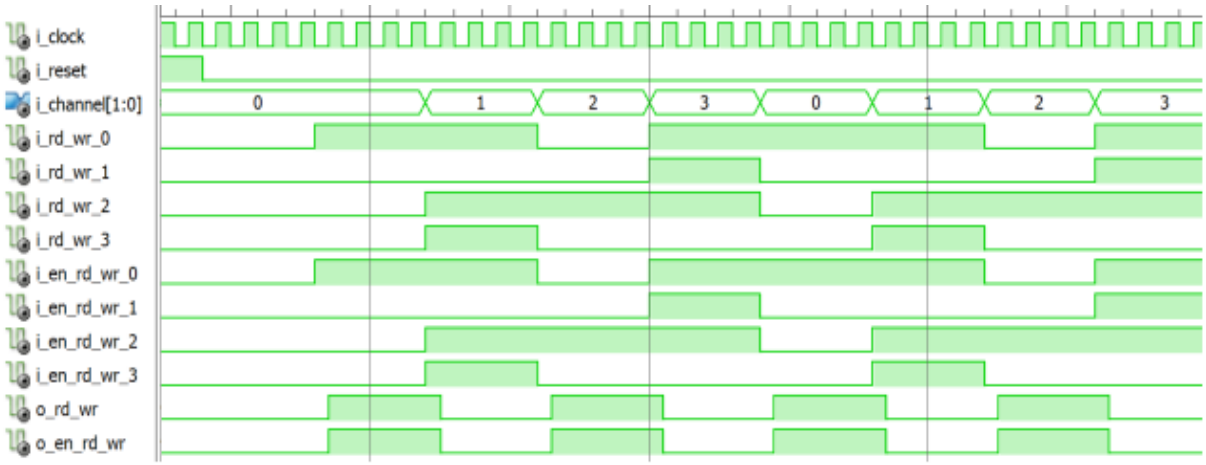
Figura 46 – *Read_Write_MUX*



Fonte: Próprio autor.

Na Figura 47 é mostrada a simulação de funcionamento do bloco *Read_Write_MUX*. Como pode ser observado, no canal 0 (zero), identificado pelo sinal *i_channel*, os valores do *o_rd_wr* e *o_en_rd_wr*, são iguais a 1. Esses valores vêm dos sinais de entrada *i_rd_wr_0* e *i_en_rd_wr_0*. Quando o *i_channel* indica canal 0 (zero), os dois sinais da saída recebem os valores que estão nos sinais de entrada correspondentes ao canal 0 (zero), por exemplo: o sinal *o_rd_wr* receberá o valor de *i_rd_wr_0* e o *o_en_rd_wr* receberá o valor de *i_en_rd_wr_0*. Já, quando no canal 1, os dois sinais da saída recebem os valores dos sinais *i_rd_wr_1* e o *i_en_rd_wr_1*. No canal 2, os sinais *o_rd_wr* e *o_en_rd_wr*, recebem os valores de *i_rd_wr_2* e *i_en_rd_wr_2*, respectivamente. Portanto, no canal 3, eles recebem os valores dos sinais de entrada *i_rd_wr_3* e *i_en_rd_wr_3*.

Figura 47 – Sinais do *Read_Writ_MUX*



Fonte: Próprio autor.

4.1.4 *Router_Interface_output*

O *Router_Interface_output* é conectado diretamente com o roteador destino da rede SoCIN. Este componente, além de informar ao roteador de destino que há um pacote a ser enviado, também monitora as respostas de recebimento de cada *flit* enviada por esse roteador. A estrutura desse componente possui dois blocos: o *MUX* e o *Data_SoCIN_Dx*.

4.1.4.1 *MUX*

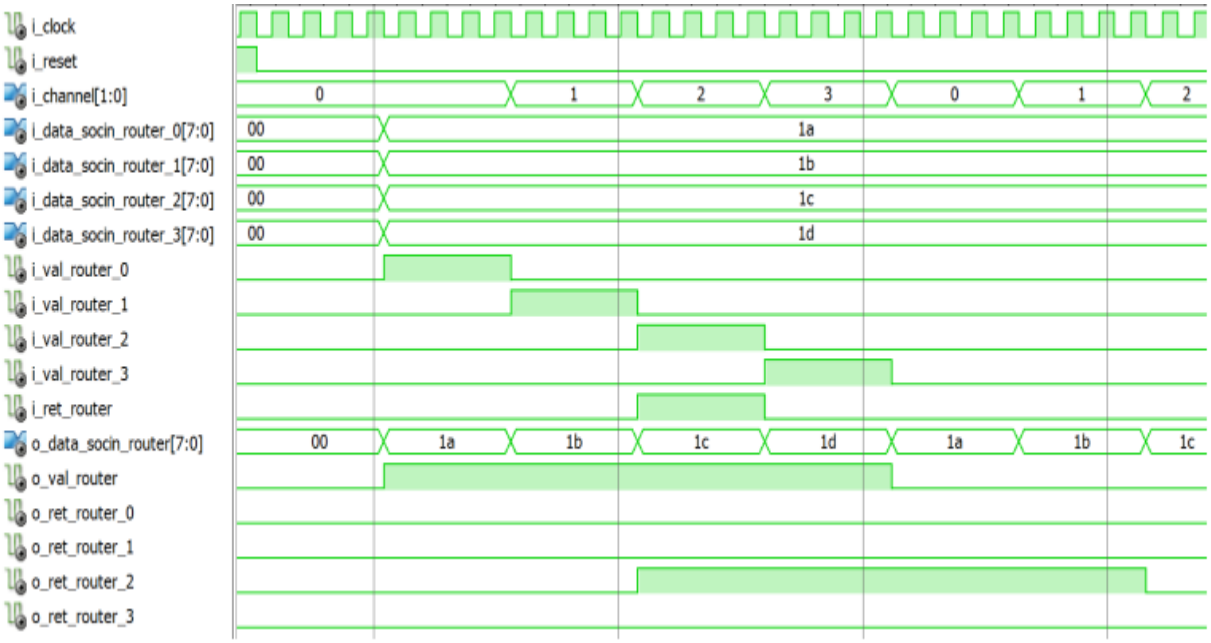
O *MUX*, mostrado na Figura 48, tem como principal função, selecionar os dados de cada canal que estão dentro da *Interface*_TDM_*channel* e transferi-los novamente para a rede SoCIN através do *o_data_SoCIN_router* a fim de entregá-los para o roteador destino. O *i_channel* informa para o bloco o canal de momento, se é o canal 0 (zero), 1, 2 ou 3. Os sinais *i_data_SoCIN_router_0* até *i_data_SoCIN_router_3* e *i_val_router_0* até *i_val_router_3* contêm os dados oriundos do roteador origem. O sinal *i_ret_router* recebe as confirmações de recebimento de cada *flit* enviadas pelo roteador destino. O *i_clock* e *i_reset* vêm do *Clock_Manager*. Os sinais de saída *o_ret_router_0* até *o_ret_router_3* recebem as confirmações de resposta, vindas do *i_ret_router* aos canais correspondentes, *o_ret_router_0* para o canal 0 (zero), *o_ret_router_1* para o canal 1 e assim sucessivamente. O *o_val_router* indica ao roteador destino quando tem algum *flit* a ser enviado.

Figura 48 – *MUX*

Fonte: Próprio autor.

Por meio da Figura 49 pode-se observar o funcionamento do *MUX*. Os sinais de entrada *i_data_socin_router_0*, *i_data_socin_router_1*, *i_data_socin_router_2* e *i_data_socin_router_3* possuem as informações “1a”, “1b”, “1c” e “1d”, respectivamente. Elas são disponibilizadas na saída *o_data_socin_router* a cada intervalo de tempo. Quando *i_channel* indica o intervalo do canal 0 (zero), as saída *o_data_socin_router* e *o_val_router* recebem os valores dos sinais de entrada *i_data_socin_router_0* e *i_val_router_0*, respectivamente. Quando no canal 1, recebem os valores dos sinais *i_data_socin_router_1* e *i_val_router_1*, tendo a mesma lógica para os canais 2 e 3. No caso das saídas *o_ret_router_0* até *o_ret_router_3*, ocorre uma demultiplexação do sinal *i_ret_router*, exemplo: se o intervalo de tempo é válido para o canal 2, a saída que receberá o valor do *i_ret_router* é a *o_ret_router_2*.

Figura 49 – Sinais do *MUX*

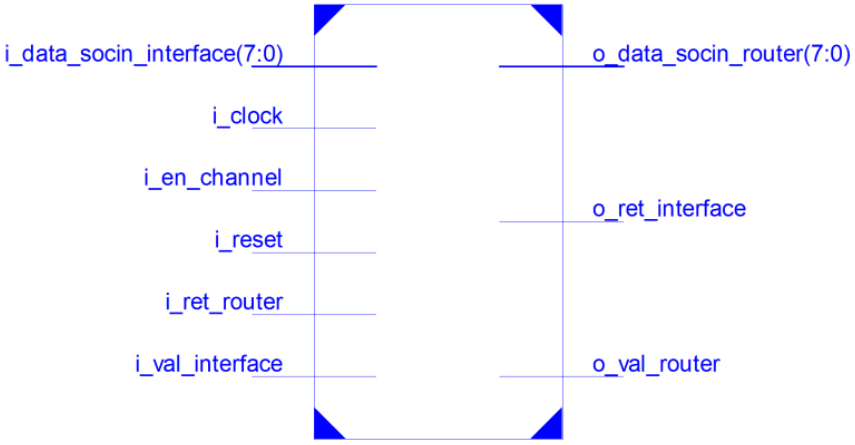


Fonte: Próprio autor.

4.1.4.2 *DATA_SOCIN_Dx*

A *interface_TDM_channel* utiliza, para cada um dos sentidos no enlace de comunicação, quatro blocos *Data_SoCIN_Dx* (Figura 50), um para cada canal. Esse componente, semelhante ao *Data_SoCIN_Dr*, também funciona como um registrador de cada canal e possui em sua entrada, os sinais: *i_data_SoCIN_interface*, *Data_SoCIN_Dr*, *i_clock*, *i_en_channel*, *i_reset*, *i_ret_router* e *i_val_interface*. Já na saída têm os sinais: *o_data_SoCIN_router*, *o_ret_interface* e *o_val_router*.

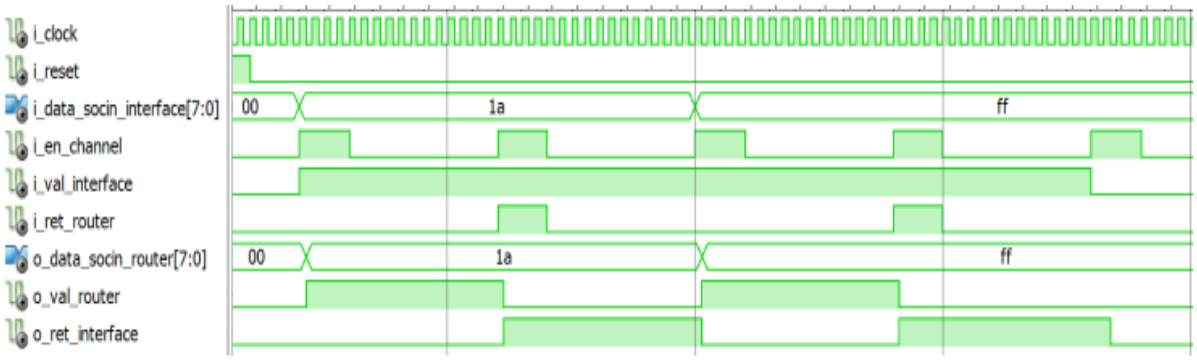
Figura 50 – *Data_SoCIN_Dx*



Fonte: Próprio autor.

Na simulação mostrada na Figura 51 é possível saber de que maneira funcionará o bloco *Data_SoCIN_Dx*. Assim como no *Data_SoCIN_Dr*, foi realizada uma simulação com o envio de dois *flits*, um *bop* (1a) e outro *eop* (ff). Os eventos desse componente acontecem nos instantes em que o *i_en_channel* tem o valor igual a 1. O envio do primeiro *flit* (1a) do *i_data_SoCIN_interface* para o *o_data_SoCIN_router* ocorre no momento em que o *i_en_channel* e *o i_val_interface* são iguais a 1. Simultaneamente ao recebimento da informação (1a) pelo sinal *o_data_SoCIN_router* é colocado o *o_val_interface* em nível lógico 1, que fica nesse estado até a confirmação de recebimento do *flit* enviado, informado pelo sinal de entrada *i_ret_router*. Após essa confirmação, é gerado um pulso em *o_ret_router* e aguardado o envio do próximo. Quando é enviado o segundo e último *flit* (ff), todas as etapas se repetem e é finalizado o envio deste pacote. O *i_clock* e *i_reset*, são provenientes do *Clock_Manager*.

Figura 51 – Sinais do *Data_SoCIN_Dx*

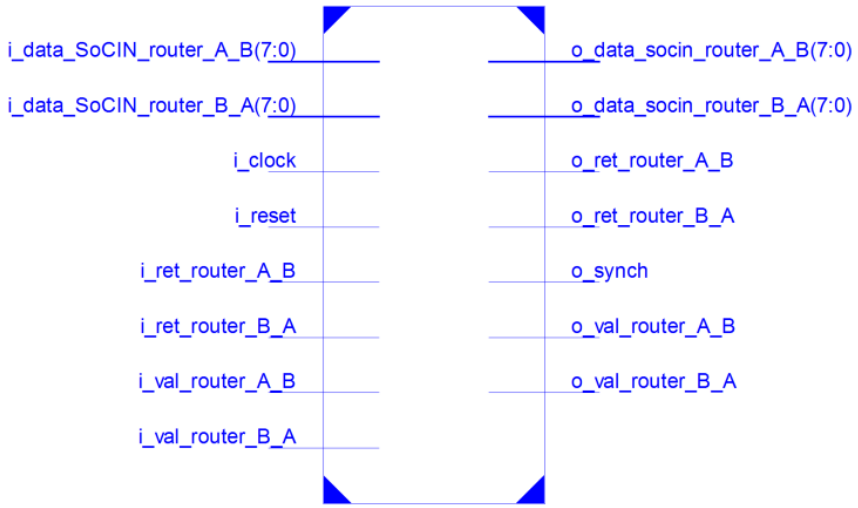


Fonte: Próprio autor.

4.1.5 *Interface_TDM_Channel*

O componente *interface_TDM_channel* é composto por todos os blocos mostrado nesse capítulo. Sua estrutura é apresentada na Figura 52. Este bloco é implementado no canal de comunicação do enlace da rede SoCIN e fará interface entre dois roteadores e também entre o roteador e seu núcleo local. Um dos canais será interligado através dos sinais *i_data_SoCIN_router_A_B*, *i_ret_router_A_B*, *i_val_router_A_B*, *o_val_router_A_B*, *o_data_SoCIN_router_A_B* e *o_ret_router_A_B*, o outro, no sentido contrário, pelos *i_data_SoCIN_router_B_A*, *i_ret_router_B_A*, *i_val_router_B_A*, *o_val_router_B_A*, *o_data_SoCIN_router_B_A* e *o_ret_router_B_A*. Os sinais *i_clock*, *i_reset* e *o_synch* são interligados com o *clock* e *reset* geral do sistema.

Figura 52 – *Interface_TDM_Channel*

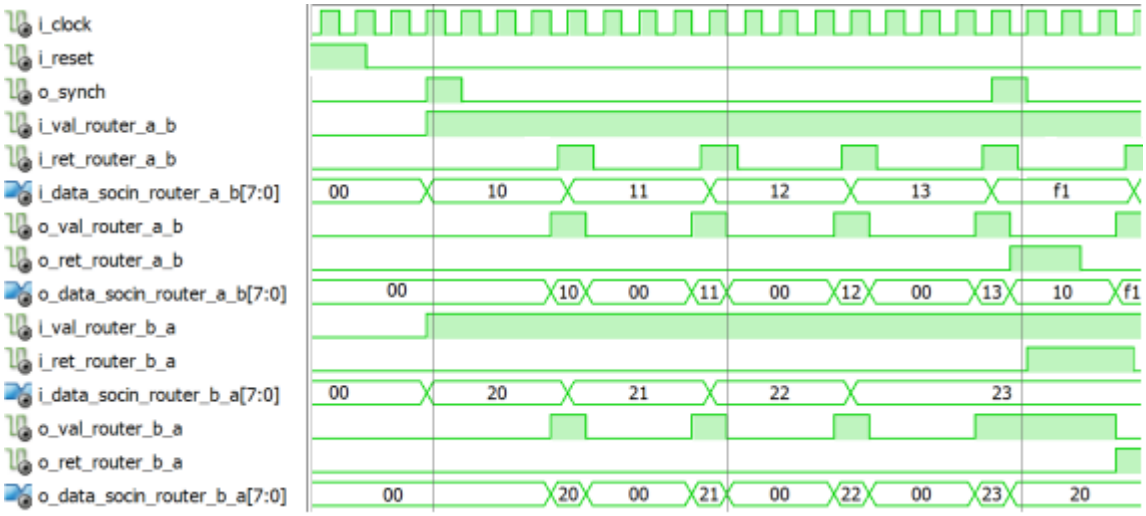


Fonte: Próprio autor.

É apresentada na Figura 53 uma simulação de comunicação entre dois roteadores genéricos (A e B) utilizando a *Interface_TDM_Channel* no enlace. Para fazer a simulação foram transmitidos *flits* nos dois canais, no sentido A para B e vice-versa. O pulso do sinal *o_synch* indica, em ambos os sentidos, o início do canal 0 (zero). A solicitação para o envio do pacote do roteador A para o roteador B é feita quando o *i_val_router_a_b* recebe nível lógico 1. Para enviar os *flits* de A para B, utilizaram-se os sinais *i_data_socin_router_a_b* e o_data_socin_router_a_b, no qual foram transmitidas as informações “10”, “11”, “12” e “13” (valores hexadecimal), nos canais 0 (zero), 1, 2 e 3, respectivamente. No instante em que a informação é disponibilizada no *o_data_socin_router_a_b* o sinal *o_val_router_a_b* vai para nível lógico 1. Praticamente ao mesmo tempo é enviada a confirmação de recebimento do *flit* pelo roteador destino (B) através do *i_ret_router_a_b*. A próxima etapa é informar ao roteador origem (A) a confirmação do recebimento do *flit* pelo roteador B. Isso ocorre no segundo pulso do o_synch, quando o *o_ret_router_a_b* é ativado em nível lógico 1. Simultaneamente, acontece o mesmo processo de envio dos *flits* (“20”, “21”, “22” e “23”) do roteador B para o roteador A. Nesse caso, os sinais utilizados foram i*_data_socin_router_b_a* e *o_data_socin_router_b_a*. Após a solicitação, feita quando o sinal *i_val_router_b_a* é igual a 1, o sinal *o_data_socin_router_b_a* recebe a informação do *i_data_socin_router_b_a* e o *o_val_router_b_a* é ativado em nível lógico 1. Depois disso, aguarda-se a confirmação de recebimento do *flit* vinda do roteador destino (A) através do *i_ret_router_b_a*. Recebida a

confirmação, *i_ret_router_b_a* com nível lógico 1, é informado ao roteador origem (B) ativando o *o_ret_router_b_a* em nível lógico 1.

Figura 53 – Sinais do *Interface_TDM_Channel*



Fonte: Próprio autor.

4.1.6 Resultados alcançados

Para implementar o bloco *Interface_TDM_Channel* utilizou-se a ferramenta ISE® versão 14.4 e o FPGA XC6VLX75T-3FF484, ambos da empresa Xilinx®. O objetivo é criar um mecanismo no canal de comunicação da rede SoCIN que garanta previsibilidade de latência e também cumprir com a entrega dos pacotes num prazo aceitável para sistemas de tempo real. Para isso foi necessário criar um bloco em linguagem VHDL, denominado *Interface_TDM_Channel*, que para ser implementado demandará um certo consumo de silício do FPGA.

Para ter uma noção do consumo requerido por esse bloco, na Subseção seguinte, será apresentado um comparativo entre o consumo dele com o custo de silício de cada mecanismo implementado na comunicação da rede SoCIN apresentados no capítulo 3.

4.1.6.1 CONSUMO DE SILÍCIO UTILIZADO PELA *INTERFACE_TDM_CHANNEL*

O custo de silício mostrado no Gráfico 1, considera a implementação do bloco *Interface*_TDM_*channel* nos cinco canais de comunicação do roteador ParIS. Os canais são: *Local*, *North*, *East*, *South* e *West*. Esses canais interligam o roteador com o nodo local e com os roteadores vizinhos. A tabela 5 apresenta um comparativo do consumo de silício do *Interface*_TDM_*channel* com outros mecanismos utilizados na rede SoCIN para prover QoS, que foram mostrados no Capítulo 3, e também com a versão original do roteador ParIS.

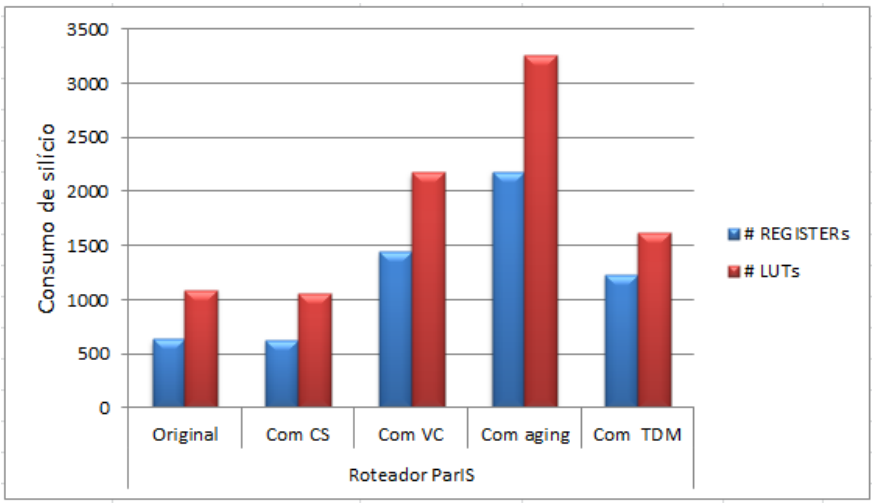
Tabela 5 – Comparativo do custo de silício

Recurso	*Roteador ParIS*				
	Original	Com CS	Com VC	Com *aging*	Com TDM
# REGISTERs	636	628	1441	2180	1230
# LUTs	1087	1058	2173	3261	1620,5

Fonte: Próprio autor.

Conforme pode ser observado na Tabela 5, a implementação do mecanismo TDM consome mais silício que o mecanismo com chaveamento de circuitos (CS), porém, menos que os mecanismos com circuitos de canais (VC) e com envelhecimento (*aging*). Apesar de ter apresentado um custo de silício superior ao apresentado pela solução baseada em chaveamento de circuitos, a solução TDM ainda mostrou-se econômica, frente às outras soluções baseadas em Aging e Canais Virtuais. Além disso, o uso da técnica TDM oferece garantia de previsibilidade de latência no envio de pacotes, que é um ponto crítico para o cumprimento dos limites de *deadline* em sistemas de tempo real. É importante salientar que o aumento de registradores em relação à versão com chaveamento de circuitos deve-se à necessidade de armazenar *flits* enquanto é aguardado a confirmação do recebimento dos *flits* pelo roteador ParIS destino. O gráfico 1 representa os valores da Tabela 5.

Gráfico 1 – Comparativo no consumo de silício



Fonte: Próprio autor.

5 CONCLUSÃO

Este trabalho teve como objetivo o desenvolvimento de uma interface TDM para ser utilizada na interconexão de roteadores da rede SoCIN. A sua principal contribuição foi a definição de uma interface com baixo consumo de silício, quando comparada com outras soluções que provêm Qualidade de Serviço implementadas nessa Rede.

Por meio das simulações, baseada na transmissão de *flits* entre dois roteadores, pode-se observar o comportamento do mecanismo no enlace de comunicação da rede SoCIN. Apesar de ser apenas uma simulação, entende-se que os experimentos realizados foram suficientes para comprovar a eficiência na transferência de dados.

Com a *Interface_TDM_channel* será possível o envio de até quatro pacotes simultâneos. Caso haja necessidade de transmitir mais que isso, ou seja, mais que quatro, deverão ser seguidos os critérios de prioridades de acordo com os mecanismos provedores de qualidade de serviço da rede.

Entende-se, baseado nos resultados, que os objetivos propostos no início desse trabalho foram atingidos e que o mecanismo deverá gerar trabalhos futuros, conforme é apresentado na Subseção que segue.

5.1 TRABALHOS FUTUROS

Pelo menos dois trabalhos poderão ser desenvolvidos a partir dos resultados obtidos neste TCC. O primeiro deles é realizar uma melhoria no *Interface*_TDM_*Channel* para permitir que o projetista do sistema possa definir a quantidade de canais TDM que deverão ser gerados na síntese da Rede.

O segundo trabalho deverá ser a alteração nas interfaces de entrada e saída da Rede SoCIN para que o roteador possa fazer uso do mecanismo proposto nesta interface de comunicação TDM e encaminhar dados nos *time slots* definidos pelo projetista do sistema.

REFERÊNCIAS

BENINI, Luca; DE MICHELI, Giovanni. Networks on chips: A New SoC Paradigm, **IEEE Computer,** [S.1.], v.35, n.1, p.70-78, Jan 2002.

BEREJUCK, Marcelo D. **Rede Intra-Chip com Qualidade de Serviços para uso em Telecomunicações**. São José, 2009. Dissertação (Mestrado em Computação Aplicada) - Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí.

BOLOTIN, E. et al. QNoC: QoS architecture and design process for network on chip. **Journal of Systems Architecture,** v. 49, 2003.

BUTTAZZO, Giorgio C. **Hard Real-Time Computing Systems: Predictable Scheduling Algorithms an Applications.** 2. ed.: EUA: Springer, 2005.

COMER, Douglas E. **Redes de Computadores e Internet:** Abrange Transmissão de Dados, Ligação Inter-redes e Web. 2 ed. Porto Alegre: Bookman, 2001.

CORRÊA, Edgard De F. **Redes-em-Chip para Sistemas Embarcados Visando a Otimização de Medidas de Qualidade de Serviço para Aplicações de Tempo Real.** 2007. 131 f. Tese (Doutorado) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2007.

DALLY, William. J.; TOWLES, Brian. **Principles and practices of interconnection networks.** San Francisco: Morgan Kaufmann, 2004.

DANTAS, Mario. **Tecnologias de Redes de Comunicação e Computadores:** Inclui: Tabela Colorida de Atribuição de Faixas de Frequência no Brasil, Bluetooth, WAP, Cluster e Grid Computing. Rio de Janeiro: Axcel Books, 2002.

DE FREITAS, Henrique C. **Arquitetura de NoC Programável Baseada em Múltiplos Clusters de Cores para Suporte a Padrões de Comunicação coletiva.** 2009. 125 f. Tese (Doutorado) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2009.

DIELISSEN, John; GOOSSENS, Kees; RADULESCU, Andrei; RIJPKEMA, Edwin. Concepts and implementation of the Philips Network-on-Chip. In: IP-SOC, 2003, Grenoble. **Proceedings…** [S.l. s.n.], 2003.

DUATO, J. et al. **Interconnection networks:** An Engineering Approach. Los Alamitos, CA: IEEE Computer Society Press, 1997.

DUATO, J. **Interconnection networks:** an engineering approach. Revised printing. Amsterdam: Morgan Kaufmann, 2003.

FILHO, Huber B. **Padrões de canalização em sistemas de transmissão digitais: TDM**. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialtdm/default.asp>. Acesso em: 30 jan. 2013.

GOOSSENS, Kees; DIELISSEN, John; RADULESCU, Andrei. Æthereal Network on Chip: concepts, architectures, and implementations. **IEEE Design & Test of Computers,** v. 22, n. 5, p. 414-421, 2005.

Hansson, A. and Goossens, K., Tradeoffs in the configuration of a network on chip for Multiple use-cases, In The 1st ACM/IEEE International Symposium on Networks-on-Chip, 2007.

HEMANI, A. *et al*., Networks on Chip: an architecture for billion transistor area. NORCHIP, 2000, [S.l.] **Proceedings...** [S.l.: s.n.], 2000, p. 166-173.

HENNESSY, J. L.; PATTERSON, D. A. **Arquiteturas de computadores:** uma abordagem quantitativa. 3 ed. Rio de Janeiro: Campus, 2003.

LIU, C. L.; LAYLAND, J. W. **Scheduling algorithms for multiprogramming in a hard-realtime environment.** Journal of the Association for Computing Machinery, 20(1), 1973.

LIU, Jane W. S. **Real-time Systems.** 1.ed. [S.l.]: Prentice Hall, 2000.

MELLO, Aline V. **Qualidade de serviço em Rede Intra-chip:** implementação e avaliação sobre a rede Hermes. 2006. 129f. Dissertação (Mestrado) – Pontifícia Universidade Católica do Rio Grande do Sul. Programa de Pós Graduação em Computação, Porto Alegre, 2006.

MORAES, F. et al. HERMES: an infrastructure for low area overhead packet-switching networks on chip. **Integration - The VLSI Journal,** London, v. 38, n. 1, Oct. 2004.

PINHEIRO, José M. S. **Redes Digitais Plesiócronas**. Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_redes_digitais_plesiocronas.php>. Acesso em: 30 jan. 2013.

SOARES, Luiz Fernando G. et al. **Redes de computadores:** Das LANs, MANs e WANs às Redes ATM. 2 ed. Rio de Janeiro: Campus, 1995.

SOUSA, Lindeberg Barros. **Redes de Computadores:** Dados, Voz e Imagem. 4.ed. São Paulo: Érica, 1999.

TANENBAUM, Andrew S. **Redes de computadores.** 3 ed. Rio de Janeiro: Campus, 1997.

ZEFERINO, Cesar Albenes. **Redes-em-Chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho.** 2003. 242f. Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós Graduação em Computação, Porto Alegre, 2003.

ZEFERINO, Cesar Albenes; SANTO, Frederico G. M. E.; SUSIN, Altamiro A. ParIS: a parameterizable interconnect switch for networks-on-chip. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN – SBCCI, 17., 2004. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 2004.