Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação

Verificação e Prototipação de Redes Intrachip: o estudo de caso Hermes-TB

Jeronimo Cunha Bezerra

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação da Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Ney Laert Vilar Calazans

Porto Alegre
2009

Dados Internacionais de Catalogação na Publicação (CIP)

B574v Bezerra, Jeronimo Cunha
Verificação e prototipação de redes intrachip : o estudo de caso Hermes-TB / Jeronimo Cunha Bezerra. – Porto Alegre, 2009.
79 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Ney Laert Vilar Calazans.

1. Informática. 2. Arquitetura de Redes. 3. FPGA. 4. Circuitos Integrados. 5. Roteamento – Redes de Computadores. I. Calazans, Ney Laert Vilar. II. Título.

CDD 004.6

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "Verificação e Prototipação de Redes Intrachip: O estudo de caso Hermes-TB", apresentada por Jeronimo Cunha Bezerra, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas Embarcados e Sistemas Digitais, aprovada em 07/08/09 pela Comissão Examinadora:

Prof. Dr. Ney Laert Vilar Calazans– Orientador — PPGCC/PUCRS

Prof. Dr. César Augusto Missio Marcon - — FACIN/PUCRS

Prof. Dr. Fernando Gehm Moraes – — PPGCC/PUCRS

Homologada em..15../..12../..09....., conforme Ata No. 27/09.. pela Comissão Coordenadora.

Prof. Dr. Fernando Gehm Moraes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32 – sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

AGRADECIMENTOS

Agradeço primeiramente ao Grande Arquiteto do Universo, por ter permitido tamanha oportunidade, aos meus pais que possibilitaram com os seus cuidados na educação ter possibilitado que a minha memória estivesse apta a receber tão valorosos ensinos. A minha mãe, por sempre me animar. Ao meu Pai pelos cartões fidelidade TAM e por sempre me animar dizendo: “Não desista!”.

Um trabalho de Mestrado é um produto gerado a partir da boa vontade de um universo de pessoas, universo esse formado pela contribuição de trabalhos de terceiros, em grande parte pelas indicações do orientador, de seus orientandos, dos cientistas que se predispuseram a dar sugestões na medida em que o trabalho se avançava, seja através de depoimentos, seja até mesmo através de um café amigo.

Agradeço ao Divino Miranda por ter entendido e possibilitado a minha ausência quando foi necessário, e por sempre dizer palavras edificadoras quando necessário.

Agradeço a Ideraldo por me ensinar a gerenciar o meu tempo, escrever de forma lógica e por sempre estar pronto quando precisava de dicas para a dissertação.

Agradeço aos meus irmãos da UDV e JB39 que tiveram a compreensão comigo quando foi necessário me ausentar para essa qualificação.

Agradeço aos meus amigos Hildeberto, Apoena, Ricardo Quati, que sempre me lembravam que faltava pouco para a conclusão deste passo.

Sou grato às pessoas do GAPH que sempre me auxiliaram nas minhas diversas perguntas, em especial a Marco Túlio e Ismael, que auxiliaram na reta de chegada. Grato pela sua ajuda, desejo muito sucesso em vossa caminhada.

Em momentos difíceis da nossa vida os pedidos feitos em nossas preces muitas vezes são resolvidos com ajuda daquilo que digo mensageiros. Nessa pesquisa sou grato aos mensageiros, pois sem o auxílio deles levaria muito mais tempo para avançar na pesquisa. São eles os doutorandos Everton Alceu Carara, pelo auxílio na codificação, ao Edson Ifarraguirre Moreno pelas dicas. Um agradecimento especial ao Rafael Iankowski Soares, pelas dicas, sugestões, correções e pelas longas conversas, sendo um co-orientador em meu trabalho.

Gostaria de agradecer aos meus amigos mestrandos, que me deram a honrosa atribuição de representante discente, e em especial ao trio UNIC Cirano, kkk, pelos brindes de passagem, e Komatsu por me mostrar que devo aprender a “ter foco”. Não posso esquecer-me do ilustre IWBI que acompanhou toda caminhada, :D.

Devo agradecer a Sarah Milani, my Angel nesse processo de Mestrado. Sem o teu esforço e dedicação o resultado não seria o mesmo. Grato pelas correções dos textos e de minha pessoa, e que por “toda vida” sempre exista este sentimento.

À minha Amiga, Adriane Malheiros, nunca deixando desanimar, e sempre relembrando que devemos sempre evoluir, nos estudos, nos sentimentos. Grato minha querida pelo carinho.

Devo agradecer a Rosana Xavier, pela torcida e por convencer muita gente a torcer com você, pelos momentos de reflexão sobre a vida desejo que o mestre oriente sempre sua vida, sua caminhada, seu filho.

Aos meus alunos e orientandos que me enriqueceram com a possibilidade de nas minhas aulas escutarem os ensinos que na verdade eram esforços meus em entender e depois transcrever os textos que aqui estão.

Ao mestre Arruda, por sempre me lembrar que sou capaz, e fazer entender a frase: "Nossas dúvidas são traidoras e nos fazem perder o bem que às vezes poderíamos ganhar pelo medo de tentar". (William Shakespeare)

Ao meu brilhante orientador, Prof. Dr. Ney Laert Vilar Calazans que me trouxe à luz tamanho conhecimento, me ensinou a revisar, revisar, revisar, definir, corrigir. Acredito que todo esse aprendizado que o senhor me propiciou está guardado e os meus alunos agradecem por poder levar a eles uma aula mais feliz ☺. Pelas conversas de diversos assuntos. Espero poder conversar por mais vezes. Agradeço pelo senhor me fazer compreender melhor a frase “E você aprende realmente que pode suportar... que realmente é forte, e que pode ir mais longe, depois de pensar que não pode mais.” (William Shakespeare)

Desejo a todos que participaram desse processo, o que a música Desejo, de Flávia Wenceslau diz:

“Eu te desejo vida, longa vida. Desejo-te a sorte de tudo que é bom. E dias de sol pra fazer os teus planos. Nas coisas mais simples que se imaginar. Eu te desejo a paz de uma andorinha, No vôo perfeito contemplando o mar. E que a fé movedora de qualquer montanha, Te renove sempre, te faça sonhar. Eu te desejo mais que mil amigos. Coração de menino cheio de esperança, Voz de pai amigo e olhar de avô.”

E peço que o Mestre Jesus auxilie a todos em sua caminhada e que o caminho seja sempre em um clima de Luz, Paz e Amor. Assim eu desejo.

VERIFICAÇÃO E PROTOTIPAÇÃO DE REDES INTRACHIP: O ESTUDO DE CASO HERMES-TB

RESUMO

O avanço tecnológico atual do processo de construção de circuitos eletrônicos possibilita a integração de mais de um bilhão de componentes em um único circuito integrado. Um circuito integrado no estado da arte é um componente complexo constituído por numerosos módulos complexos conhecidos como núcleos de propriedade intelectual. Circuitos integrados modernos contêm dezenas ou centenas de núcleos interconectados. Cada vez mais a interconexão de núcleos se faz através de estruturas de comunicação complexas. Uma forma de organizar estas arquiteturas é construí-las sob a forma de uma rede intrachip. O uso de estruturas de comunicação total ou parcialmente regulares tende a aumentar a escalabilidade e o grau de paralelismo da comunicação em sistemas integrados complexos. Uma das características mais importantes de uma rede intrachip é a sua topologia. Este trabalho aborda a verificação e a prototipação da rede intrachip Hermes-TB. Esta rede emprega topologia do tipo toro 2D bidirecional como forma de alcançar baixa latência e alta vazão a um custo de hardware reduzido. A verificação do projeto da Hermes-TB foi obtida aqui através da execução da simulação com atrasos do projeto original, pois a proposta inicial da rede realizou a validação do projeto apenas através de simulação funcional. Por outro lado a prototipação, aqui realizada sobre plataformas baseadas em FPGAs (do inglês, *Field Programmable Gate Arrays*) validou o projeto pela primeira vez em hardware. Ao final deste trabalho pôde-se então confirmar a viabilidade de uso da rede intrachip Hermes-TB em circuitos reais.

Palavras Chave: Redes intrachip, NoCs, topologia toro, toro 2D, prototipação, FPGA.

VERIFICATION AND PROTOTYPING OF INTRACHIP NETWORKS: THE HERMES-TB CASE STUDY

ABSTRACT

The current state of electronic circuit design and fabrication processes enables the integration of more than a billion devices in a single integrated circuit. A state of the art integrated circuit is a complex component formed by several complex modules known as intellectual property cores. Modern integrated circuits contain dozens or hundreds of such cores interconnected. The interconnection of cores is growingly performed through complex communication structures. One way to organize such interconnect architectures is to build them in the form of an intrachip network. The use of totally or partially regular communication structures improves scalability and the degree of communication parallelism in complex integrated circuits. One of the most important characteristic of intrachip networks is its topology. This work approaches the verification and prototyping of the Hermes-TB intrachip network. This network employs a regular, bidirectional 2D torus topology as a means to reach low latency and high throughput communication at a reasonable hardware cost. The Hermes-TB design verification was achieved through the use of timing simulation of the original design, since the original proposal of the network employed only functional simulation as design validation method. Prototyping of Hermes-TB, on the other hand, was conducted on an FPGA-based platform, and served to validate the network design in hardware for the first time. At the end of this work, it was then possible to confirm the viability to use the Hermes-TB intrachip network in real circuits.

Keywords: Intrachip networks, NoCs, torus topology, 2D torus, prototyping, FPGA.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

2D	Duas Dimensões
ASIC	Application Specific Integrated Circuit
CBDA	Centrally Buffered, Dynamically Allocated
CI	Circuito Integrado
CLICHÉ	Chip Level Integration of Communicating Heterogeneous Elements
CMOS	Complementary Metal-Oxide Semiconductor
CMP	Chip Multiprocessor
DAMQ	Dynamically-Allocated, Multi-Queue
DCM	Digital Clock Manager
EDIF	Electronic Data Interchange Format
FCFS	First Come First Served
FIFO	First In First Out
FPGA	Field Programmable Gate Arrays
GAPH	Grupo de Apoio ao Projeto de Hardware
HDL	Hardware Description Language
HOL	Head Of Line
IP	Intellectual Property
ITRS	International Technology Roadmap for Semiconductors
LRS	Least Recently Served
LUT	Look Up Table
MPSoC	MultiProcessor System-on-Chip
NoC	Network on Chip
PLA	Programmable Logic Array
RR	Round Robin
RTL	Register Transfer Level
SAFC	Statically Allocated, Fully Connected
SAMQ	Statically Allocated, Multi-Queue
SDF	Standard Delay Format
SoC	System on a Chip
TB	Torus Bidirectional
VC	Virtual Channel
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VLSI	Very Large Scale Integration

SUMÁRIO

1 INTRODUÇÃO

O avanço atual da tecnologia de fabricação de circuitos integrados (CIs), popularmente conhecidos como *chips*, possibilita maior densidade de integração de componentes, viabilizando hoje a construção de um único CI composto de mais de um bilhão de transistores. Atualmente, o desenvolvimento destes CIs compreende a integração de múltiplos componentes, como processadores, blocos de memória e controladores de acesso a periféricos, em um único chip, criando assim um sistema completo em uma única pastilha. Tais sistemas são conhecidos como SoCs, do inglês *Systems on a Chip*. Para atender a demanda do mercado obtendo lucros oriundos da amortização do custo de projeto e redução dos prazos, é importante que os componentes de um SoC sejam reutilizáveis. Assim, o projeto de CIs deve se basear cada vez mais no reuso de componentes pré-projetados e melhorados a cada uso. Esses componentes reutilizáveis são denominados núcleos IP (do inglês, *Intellectual Property Cores*, ou *IP Cores* ou IPs). Um SoC contendo vários processadores é denominado MPSoC (do inglês, *MultiProcessor System on a Chip*) que pode ser distinguido de CMPs (do inglês, *Chip MultiProcessors*), estes últimos definidos como CIs que contêm vários processadores idênticos. Comercialmente, CMPs são hoje designados como dual *core*, *quad-core*, etc. de acordo com o número de núcleos processadores (em inglês, *cores*) que estes possuem. MPSoCs e CMPs são resultados da tecnologia de integração em muito larga escala (do inglês, *Very Large Scale Integration* ou VLSI), que viabiliza agregar vários processadores programáveis em um único CI [JER05].

Internamente, tanto CMPs quanto MPSoCs fazem uso de arquiteturas de comunicação, podendo estas ser os tradicionais barramentos intrachip, ou estruturas mais complexas, denominadas redes intrachip (do inglês, *Networks on Chip* ou NoCs). O primeiro tipo de arquitetura de comunicação consiste de um conjunto de condutores aos quais se conectam múltiplos núcleos e que possibilitam a transmissão de dados entre um par de núcleos de cada vez. Barramentos têm como principal vantagem o baixo custo e a facilidade de expansão. Como desvantagens, podem levar à eliminação ou redução do paralelismo potencial da comunicação entre dispositivos interligados, baixa escalabilidade, alto consumo de energia quando o número de núcleos cresce, e limitação da velocidade de comunicação proporcional ao aumento do número de núcleos. O segundo tipo de arquitetura de comunicação (NoCs) consiste em geral de um conjunto de elementos roteadores interconectados por canais de comunicação ponto a ponto. O conceito e suas variações são mais detalhados nos capítulos seguintes deste trabalho.

NoCs baseiam-se em conceitos oriundos das áreas de sistemas distribuídos, redes de computadores e processamento paralelo, adaptando estes conceitos aos requisitos de comunicação no interior de um SoC complexo. O uso de NoCs como mecanismo de comunicação entre núcleos IP [BEN01] tem sido empregado por apresentar vantagens quando comparado a barramentos tradicionais, com características de melhora de parâmetros tais como: eficiência energética, confiabilidade, reusabilidade, comunicação não-bloqueante e escalabilidade de largura de banda.

O objetivo desse trabalho de pesquisa é a prototipação de uma NoC com topologia toro 2D proposta em trabalho anterior [SCH07]. Outro trabalho relacionado desenvolvido no âmbito do Grupo de Pesquisa do Autor é o ambiente de teste NoCs denominado HardNoC (do inglês, *Hardware Platform to Debug the Hermes NoC*) [MOR08]. A HardNoC é utilizada neste trabalho sendo alterada para dar suporte à rede toro Hermes-TB.

1.1 Motivação

Segundo estudos do ITRS (do inglês, *International Technology Roadmap for Semiconductors*) [ITR07], com o processo de integração de bilhões de transistores e centenas de IPs em uma mesma pastilha de silício, é importante realizar pesquisas que possibilitem uma maior eficiência, baixo consumo de energia e velocidade na comunicação das informações dentro de *chips*. Observa-se essa tendência, por exemplo, ao perceber iniciativas como a do fabricante TILERA desenvolvendo o Processador Tile64 com 64 núcleos processadores [TIL07] e a de pesquisadores da INTEL, construindo um chip contendo 80 núcleos processadores de ponto flutuante [VAN07]. Ambos os projetos utilizam uma rede intrachip como arquitetura de comunicação entre núcleos processadores. Ambos também demonstram preocupação com arquiteturas de comunicação com alto grau de paralelismo.

É evidente a necessidade de desenvolvimento de novas formas de transmitir as mensagem no interior de *chips* com maior eficiência, habilitando a vasta área de pesquisas em NoCs. Neste cenário, este trabalho possibilita a evolução da pesquisa desenvolvida no Grupo de Apoio ao Projeto de Hardware (GAPH), onde se têm utilizado sobretudo a topologia malha 2D da NoC Hermes, proposta originalmente em [MEL03]. A escolha pela topologia malha 2D está fundamentada na sua simplicidade de implementação em hardware [DUA97]. Outra rede proposta pelo GAPH foi a rede Mercury com topologia toro 2D [BAS05]. Para a rede Hermes, existe o Ambiente Atlas, um software que permite gerar automaticamente redes Hermes e Mercury entre outras, a geração de tráfego e a visualização de estatísticas do tráfego [GAP08]. O trabalho realizado por Scherer [SCH07], por outro lado, propôs, projetou e implementou duas NoCs com topologia toro 2D (Hermes-TU e Hermes-TB), utilizando a infra-estrutura Hermes como base de desenvolvimento. Além de expandir a estrutura de suporte ao projeto das duas NoCs, integrando as redes propostas ao ambiente Atlas [SCH07], o mesmo trabalho apresentou uma avaliação comparativa destas contra a rede Hermes, nos aspectos de área, vazão e latência. A rede Hermes-TB apresentou excelentes resultados de desempenho, superiores ao de redes Hermes com área equivalente.

Daí percebe-se a importância e a necessidade de dar continuidade, agregando valor aos projetos do GAPH. Assim, este trabalho traz como contribuição a verificação e a prototipação da NoC Hermes-TB, através da expansão do ambiente HardNoC para dar suporte à Hermes-TB.

1.2 Objetivos

Os objetivos estratégicos deste trabalho foram o domínio do processo de projeto e validação de redes intrachip em geral e de redes com topologia toro em particular, bem como do processo de prototipação de redes intrachip em FPGAs.

Como objetivos específicos foram estabelecidos os seguintes:

- Estudo e análise do ambiente Testador HardNoC;
- Adequações para integrar a rede Hermes-TB à HardNoC.
- Simular funcionalmente e com atrasos o projeto da Hermes-TB, com a finalidade de obter resultados experimentais próximo das características operacionais reais;
- Validar a rede Hermes-TB, demonstrando sua viabilidade de implementação em hardware.

1.3 Contribuições

A primeira contribuição foi a validação por simulação com atrasos do projeto da rede Hermes-TB. A segunda e principal contribuição é disponibilizar um ambiente que permite prototipar instâncias da rede Hermes-TB em FPGAs.

1.4 Organização do Restante do Texto

O restante do texto desta dissertação está organizado seis capítulos. O Capítulo 2 introduz alguns conceitos básicos relacionados a redes intrachip em geral. O Capítulo 3 mostra alguns trabalhos relacionados importantes para esta dissertação. O Capítulo 4 aborda os experimentos necessários para demonstrar em um ambiente de simulação, a funcionalidade da rede Hermes-TB. O Capítulo 5 discute o processo de prototipação em FPGA da Hermes-TB. Finalmente o Capítulo 6 apresenta algumas conclusões do trabalho e sugere trabalhos futuros.

2 CONCEITOS BÁSICOS DE REDES INTRACHIP

Este Capítulo apresenta os principais conceitos relacionados a redes intrachip. Redes intrachip são arquiteturas de comunicação que normalmente apresentam alto grau de complexidade. Para dominar a complexidade de projeto destas arquiteturas é necessário familiaridade com uma quantidade de conceitos fundamentais de redes de comunicação. Dentre estes conceitos, destacam-se as estratégias de chaveamento, os algoritmos de roteamento, os métodos de armazenamento temporário de mensagens, entre outros. O objetivo deste Capítulo é introduzir um subconjunto destes conceitos, aqueles mais relevantes para o presente trabalho.

2.1 Roteamento: Classificações e Problemas

Um primeiro conceito importante é o de *algoritmo de roteamento*, usado para definir o caminho que um pacote deve utilizar a partir de um nodo origem até um nodo destino. Tais algoritmos podem ser classificados de acordo com diversos critérios. A Figura 1 ilustra algumas das diferentes classificações de algoritmos de roteamento, baseado nos critérios que as definem.

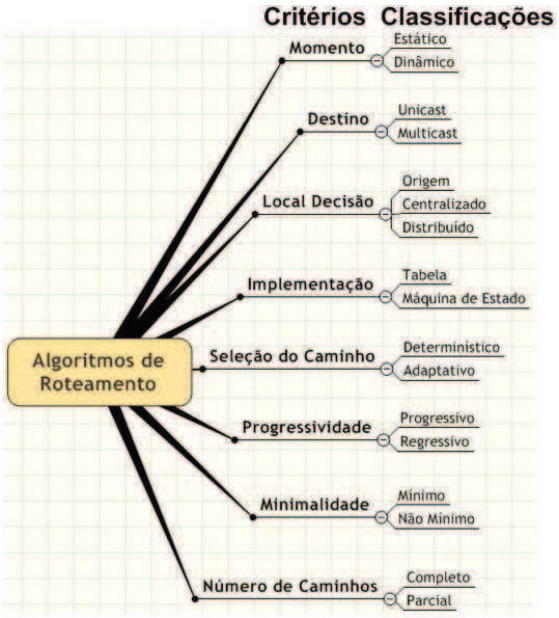

Figura 1 - Critérios utilizados para classificar algoritmos de roteamento e as classificações derivadas destes.

O conjunto amplo, embora não-exaustivo, de critérios de classificação ilustrado na Figura 1 é descrito a seguir:

- Quanto ao momento de realização do roteamento - algoritmos podem ser classificados segundo este critério em *estáticos*, quando definidos em tempo de projeto da rede ou *dinâmicos*, quando definidos durante a operação do sistema.
- Quanto à quantidade de destinos das mensagens - algoritmos podem ser classificados segundo este critério em *unicast*, quando existe apenas um destino ou *multicast* quando para pelo menos um pacote existam pelo menos dois destinos. Um caso especial importante de algoritmo *multicast* são os algoritmos *broadcast*, onde um pacote gerado em algum ponto da rede é enviado para todos os demais pontos da rede.

- Quanto ao local onde a decisão de roteamento é tomada - algoritmos podem ser classificados segundo este critério em *centralizado*, quando um nodo define o caminho de todos os pacotes, ou *origem* quando o caminho ao longo da rede é definido na origem do pacote, fora da rede. O último tipo são os algoritmos *distribuídos*, quando o caminho do pacote é definido a cada roteador por onde este passa no seu percurso ao destino.
- Quanto à forma de implementação - algoritmos podem ser classificados segundo este critério em *baseados em tabelas*, quando o trajeto é definido a partir de consultas a tabelas que indicam o caminho ou parte deste, ou *baseados em máquinas de estado*, quando o roteamento é implementado usando um algoritmo [BAR05].
- Quanto ao processo de seleção do caminho - algoritmos podem ser classificados segundo este critério em *determinísticos* quando, definidas origem e destino, o trajeto a ser realizado é sempre o mesmo. Outro tipo são os algoritmos *adaptativos*, onde o caminho da origem ao destino é determinado em tempo de execução, determinado pelas condições de tráfego, buscando o melhor caminho no instante do roteamento.
- Quanto à progressividade - algoritmos podem ser classificados segundo este critério em *progressivos*, se o pacote sempre avança reservando canais ou *regressivo*, caso o mesmo possa liberar canais reservados previamente.
- Quanto à minimalidade - algoritmos podem ser classificados segundo este critério em: *mínimos*, quando o nodo seguinte visitado por um pacote estiver sempre mais próximo do destino que o nodo anterior; ou *não mínimo*, quando se permite o afastamento temporário de um pacote do seu destino.
- Quanto ao número de caminhos - algoritmos podem ser classificados segundo este critério em *completos*, quando todos os caminhos possíveis de serem usados pelo pacote para ir de um nodo fonte a um nodo destino podem ser empregados pelo algoritmo, ou *parciais*, quando a quantidade de caminhos entre um fonte e um destino é limitada pelo algoritmo usado.

A seleção de um algoritmo de roteamento deve levar em consideração diferentes aspectos relacionados à estrutura de rede, do pacote, das interfaces de comunicação e do processo de transmissão de informação no interior da rede.

Um segundo conjunto de conceito em redes intrachip envolve a necessidade de evitar os fenômenos de *deadlock*, *livelock* e *starvation* [COU94]. Estes fenômenos podem ser definidos da seguinte maneira:

1) *Deadlock* é uma interdependência cíclica de comunicação que bloqueia indefinidamente alguns caminhos da infra-estrutura de comunicação.

2) *Livelock* é a situação em que a informação enviada nunca atinge o seu destino, circulando indefinidamente dentro da rede.

3) *Starvation* é o adiamento por períodos arbitrariamente longos do direito de acesso a um dado recurso de comunicação por uma informação transitando na rede.

2.2 Métodos e Modos de Chaveamento e Algoritmos de Chaveamento

A escolha de como se transfere pacotes entre origem e destino é o que determina o comportamento da lógica de chaveamento interna de cada roteador. É possível discernir dois métodos de transferência, denominados *chaveamento de circuito* e *chaveamento de pacotes*, definidos a seguir.

No *chaveamento de circuito* (do inglês, *circuit switching*), o trajeto do nodo origem ao destino é inicialmente determinado e reservado, criando-se uma conexão (lógica e/ou física) entre origem e destino. Somente depois de estabelecida a conexão, inicia-se o envio da mensagem.

No *chaveamento de pacotes* (do inglês, *packet switching*), a mensagem é particionada ou agregada em *pacotes*, que são transmitidos sem uma determinação prévia do trajeto do nodo de origem ao destino. Isto, porém implica a escolha de um dentre três *modos de chaveamento* que são:

Store-and-forward: Este modo se assume que cada pacote é armazenado inteiramente em um roteador antes de seguir caminho para o próximo roteador;

Virtual-cut-through: Neste modo, o roteador pode enviar um pacote adiante desde que o próximo roteador garanta a viabilidade de recebê-lo completamente;

Wormhole: Neste modo, bastante usual em redes intrachip, pacotes são divididos em partes menores denominadas *flits*, que constituem a unidade fundamental de controle de fluxo na rede. Cada *flit* pode ser enviado separadamente entre roteadores. Tipicamente, o primeiro *flit* constitui-se de um cabeçalho que sucessivamente reserva canais por onde todos os demais *flits* do pacote seguirão.

2.3 Exemplos de Algoritmos de Roteamento

Um algoritmo de roteamento simples e bem conhecido é denominado de roteamento XY [DUA97]. Este é um algoritmo determinístico de operação mais simples em redes como malha 2D. Ele é naturalmente livre de *deadlock* e *livelock*.

A vantagem deste algoritmo em redes com topologia malha é que ele garante a liberdade de *deadlock* a um baixo custo de implementação. No entanto, ele restringe fortemente a forma de utilização dos recursos da rede. Essa restrição provém do fato do pacote ter que percorrer totalmente as abscissas até chegar à coluna da ordenada em que se situa o nodo destino, seguindo então por caminhos verticais até o destino.

A Figura 2 ilustra o trajeto de um pacote de 4 *flits* que penetra uma rede 3x3 com topologia malha 2D pelo roteador do canto inferior esquerdo e tem como destino o roteador superior central.

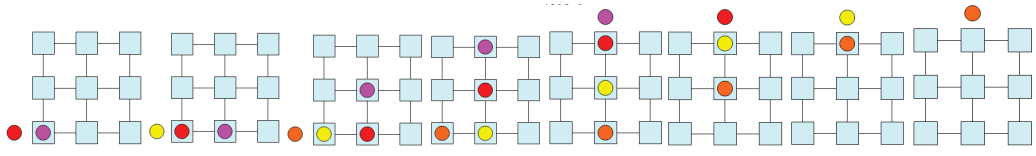

Figura 2 - Trajeto dos flits seguindo um algoritmo XY operando no modo wormhole em uma malha 3x3.

A Figura 3, por outro lado, apresenta e compara quatro algoritmos de roteamento, um determinístico, o XY explicado acima e três parcialmente adaptativos: West-first, North-last e Negative-first. Todos os algoritmos apresentados são mínimos, embora os adaptativos possuam versão não-mínima [GLA94]. A adaptatividade mesmo parcial, pode reduzir o tempo total para a entrega de um pacote individual, pois em algumas situações ele pode mudar de direção, evitando

condições de bloqueio.

As direções permitidas nestes algoritmos são as representadas pelas linhas cheias, enquanto as linhas pontilhadas representam curvas não permitidas aos *flits* de um pacote. Com base nessa explicação detalha-se a seguir os algoritmos adaptativos propostos originalmente por Glass e Ni em [GLA94].

- Algoritmo West-First - Neste algoritmo, ilustrado na Figura 3 (quadrante superior esquerdo), curvas para Oeste são proibidas. Dessa forma, o algoritmo West-First testa se Xd <= Xo, e em seguida roteia deterministicamente, de forma semelhante ao algoritmo XY. Se Xd > Xo então este roteia de forma adaptativa nas direções Leste, Norte ou Sul.
- Algoritmo *North-Last* - Neste algoritmo, ilustrado na Figura 3 (quadrante superior direito), não é permitido realizar curvas para Norte. O algoritmo *North-Last testa se* $Y_d \geq Y_o$, e em seguida roteia deterministicamente. Se $Y_d < Y_o$ então este roteia de forma adaptativa nas direções *Oeste, Leste,* ou *Sul*.
- Algoritmo Negative-First - Neste algoritmo, ilustrado na Figura 3 (quadrante inferior direito), não são permitidas curvas para direções negativas. Dessa forma, o algoritmo, fazendo uso da abreviatura X_T para destino de X e X_S para origem de X, verifica se ($X_T \leq X_S$ e $Y_T \geq Y_S$) ou ($X_T \geq X_S$ and $Y_T \leq Y_S$), e então roteia deterministicamente como mostra a Figura no caminho 1 (endereço fonte (3,4) e endereço destino (0,7)) e no caminho 3 (endereço fonte (3,7) e endereço destino (6,5)). Qualquer outra condição permite que o roteamento seja adaptativo.

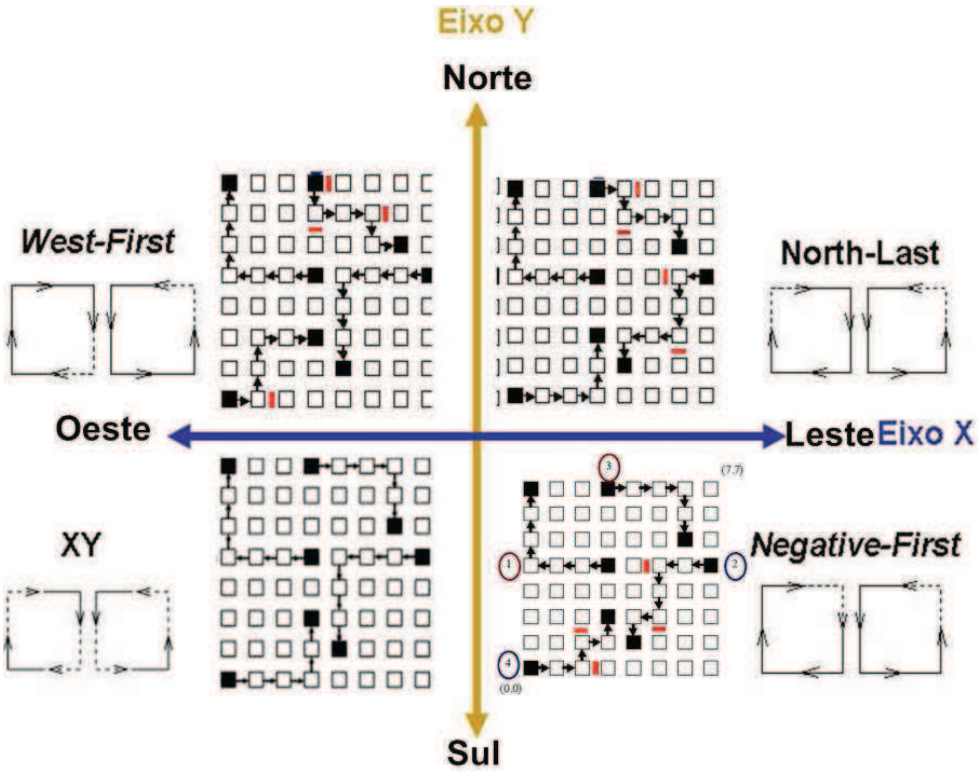

Figura 3 - Detalhamento da funcionalidade dos algoritmos de roteamento baseados no modelo de curvas de Glass e Ni. As curvas em linhas cheias representam uma curva permitida, e as linhas pontilhadas representam curvas proibidas. Barras verticais no caminho entre roteadores representam bloqueios (devido, por exemplo, a outros pacotes usando o canal ou falhas na rede).

2.4 Controle de Fluxo

Quando algum recurso está alocado para um pacote de forma a impedir a sua alocação a outro pacote, ocorre um bloqueio de recursos. Nesta situação, é necessário que exista uma política determinando a melhor maneira de lidar com o pacote que não pode ser atendido. Dentre as políticas mais usadas destacam-se algumas:

- O descarte de pacotes que não têm acesso ao recurso necessário;
- O bloqueio e armazenamento de pacote;
- O desvio do caminho seguido pelo pacote.

Essa determinação é a política do controle de fluxo, definindo quem aloca os canais e as filas no momento em que o pacote trafega na rede [NI93]. Ela também possibilita saber se o receptor está habilitado a receber dados ou não. As estratégias de controle de fluxo utilizadas aqui pressupõem o bloqueio e armazenamento de pacotes. Neste sentido duas estratégias são usadas para realizar o controle de fluxo: baseado em créditos e *handshake*:

- Controle de fluxo baseado em créditos - não deixa que pacotes sejam descartados, pois uma transmissão entre roteadores só inicia após a verificação junto ao receptor que os dados serão recebidos. Essa verificação é realizada da seguinte forma: o receptor envia ao transmissor um sinal confirmando a existência de créditos que o último possui para envio dos dados. Com esta informação, o transmissor envia dados somente quando existir crédito para o envio. Ao longo do processo de envio de dados do transmissor pode ou não ocorrer mudança no seu estado de créditos junto ao receptor, dependendo do espaço de armazenamento disponível no último.
- Controle de fluxo *handshake* - o transmissor informa ao receptor a intenção de enviar um dado através de um sinal de solicitação. O receptor recebendo o sinal de requisição verifica a existência de espaço para recebimento. Existindo, o dado é lido e armazenado, e o receptor envia o sinal de reconhecimento de recepção (*ack*) ao transmissor. Na ausência desta possibilidade de recepção, o receptor pode enviar um sinal de não reconhecimento (*nack*), fazendo com que o roteador transmissor necessite retransmitir o dado até o recebimento de um *ack*.

Uma observação quanto ao uso do controle de fluxo *handshake* é a perda de desempenho, devido ao tempo que os dados ficam armazenados nas filas até receberem o sinal de *ack,* ou quando não existe espaço em fila já que se torna necessário reenviar o dado.

2.5 Arbitragem

O fato de ocorrerem conflitos de requisições simultâneas requer a determinação de procedimentos para resolver, através do gerenciamento de acesso a recursos compartilhados, o uso dos componentes de rede pelas mensagens que nesta trafegam. Por exemplo, ao receber requisições de roteamento simultâneas, um elemento de arbitragem pode atribuir prioridade a portas de um roteador e com base nestas encaminha pacotes para uma unidade única de roteamento conforme esta escala de prioridades.

À arbitragem está relacionado ao problema de *starvation*, que pode ser amenizado e até mesmo resolvido de acordo com o critério de arbitragem adotado. Existem diversas políticas propostas, tais

como prioridade estática, prioridade dinâmica, escalonamento por idade, FCFS (*First Come First Served*), LRS (*Least Recently Served*) e RR (*Round-Robin*). As vantagens e inconvenientes de cada uma destas são amplamente discutidos na literatura. Ver por exemplo [SIL97] para uma discussão mais detalhada.

2.6 Memorização

Nas redes que utilizam chaveamento por pacotes do tipo *wormhole*, os roteadores devem armazenar os *flits* dos pacotes com destino às saídas que estejam indisponíveis, realizando o controle de fluxo para evitar a perda de dados. Para oferecer essa característica é necessário usar um esquema de memorização temporária para armazenamento dos pacotes bloqueados no roteador. Claramente não é possível disponibilizar uma capacidade de armazenamento infinita para garantir armazenamento na rede de todo e qualquer pacote em qualquer situação. Assim um processo crítico é o de dimensionar os meios de armazenamento temporário dentro dos roteadores, de forma a reduzir tanto a perda de desempenho como o desperdício de área de silício.

Existem três opções básicas de memorização:

- Memorização na entrada - a memorização temporária de dados na entrada pressupõe a existência de filas independentes em cada uma das portas de entrada do roteador. As filas podem ser implementadas de várias formas, destacando-se as estratégias FIFO (*First In First Out*), SAFC (*Statically Allocated, Fully Connected*), SAMQ (*Statically Allocated, Multi-Queue*) e DAMQ (*Dynamically-Allocated, Multi-Queue*) [CAR07]. Esta estratégia é usada, por exemplo, na NoC Hermes [MOR04].
- Memorização na saída - a memorização temporária de dados na saída implica a inserção de filas nas portas de saída do roteador. O problema desta estratégia é que cada fila deve ser capaz de receber simultaneamente dados das N entradas, implicando que a fila de saída possua N portas de entrada ou que opere a uma velocidade N vezes maior do que as entradas. O uso de armazenamento temporário de saída exige a implementação de um controle de fluxo entre a porta de entrada e de saída, aumentando assim a complexidade do roteador.
- Memorização centralizada compartilhada – a memorização centralizada compartilhada, denominado CBDA (*Centrally Buffered, Dynamically Allocated*), utiliza filas para armazenamento de pacotes de todas as portas de entrada do roteador. O espaço de memória disponível a ser utilizado é dividido de forma dinâmica entre os pacotes de diferentes entradas. Esta estratégia é usada, por exemplo, na NoC Mercury [BAS05].

O armazenamento temporário centralizado compartilhado oferece uma melhor utilização de memória do que aquelas proporcionadas pelas abordagens onde este espaço é prévia e estaticamente alocado a portas de entrada. Segundo Zeferino [ZEF03], o CBDA deve oferecer no mínimo uma largura de banda igual à soma das larguras de banda de todas as portas, fazendo com que em um roteador NxN, a fila possua $2N$ portas de acesso, de modo a permitir N acessos simultâneos de leitura e N acessos simultâneos de escrita.

Como desvantagem, pode-se citar um problema semelhante ao bloqueio HOL (do inglês, *Head of Line*) [MOR04] [ZEF03]. Este tipo de bloqueio ocorre quando uma porta de saída está em uso por uma determinada porta de entrada e ao mesmo tempo outra porta de entrada está recebendo dados e também deseja utilizar essa mesma saída. Neste caso, se o pacote que está alocando a porta de saída

encontrar um bloqueio em um roteador à frente no seu caminho, a segunda porta fica com seus dados bloqueados, talvez inutilmente. Quando isso ocorre, as outras portas de comunicação são afetadas, pois uma fila eventualmente lotada acarreta a recusa ou o bloqueio adicional de dados, criando contenção. Este problema pode ser evitado restringindo espaço alocado em cada fila para cada uma das portas de entrada.

3 TRABALHOS RELACIONADOS

A topologia de uma rede intrachip determina como são interligados os núcleos de um CI. Malha 2D e toro 2D são exemplos de topologias regulares largamente utilizadas em projetos de NoCs.

A topologia malha 2D é definida como aquela que distribui os nodos de forma simétrica em um plano cartesiano. Nesta topologia, cada nó da rede se liga a até quatro outros localizados nas direções dos pontos cardeais pelos seus lados (Norte, Sul, Leste e Oeste). Exceções são os nodos localizados na periferia da arquitetura. Por exemplo, aqueles localizados nos lados superior, inferior, direito e esquerdo terão apenas três interligações com seus três vizinhos, e aqueles localizados nos cantos superiores (esquerdo e direito) e inferiores (esquerdo e direito) terão duas conexões com os seus dois vizinhos.

A topologia toro 2D pode ser obtida a partir da malha 2D, criando conexões entre os extremos da malha 2D, de forma a criar anéis horizontais e verticais. Os anéis em uma topologia toro podem conduzir tráfego em um ou nos dois sentidos, ao contrário do que ocorre em uma rede malha 2D regular, onde os enlaces sempre devem ser bidirecionais para permitir alcançabilidade plena entre nodos da rede. Muitas das propostas de NoCs com topologia toro 2D pressupõe o uso de toros unidirecionais como a proposta de Pande et al. [PAN05].

Este Capítulo restringe-se a revisar em detalhe as duas famílias de NoCs abordadas diretamente neste trabalho, a NoC Hermes, uma variação desta, a Hermes-VC e a NoC Hermes-TB. Uma revisão do estado da arte em conceitos e propostas de NoCs pode ser encontrada em [MOR04]. No caso específico de NoCs com topologia toro, o leitor pode usar o trabalho de Scherer [SCH07] como uma revisão do estado da arte neste tipo de NOCs.

3.1 A Rede Intrachip Hermes

A rede Hermes é o projeto seminal de NoCs empreendido pelo GAPH a partir do trabalho inicial de Mello e Möller [MEL03]. Sua primeira implementação foi sob a forma de uma rede com topologia malha 2D, sem canais virtuais, com controle de fluxo do tipo *handshake*. Com referência ao modelo de referência OSI, a Hermes original implementou apenas os níveis físico e de enlace. No nível físico a rede realiza a transferência de bits através de um enlace formado por um conjunto de fios estruturados a partir da definição do tipo de controle de fluxo a utilizar entre roteadores. No nível de enlace ocorre a transmissão da mensagem, dividida conforme de acordo com os princípios ditados pela escolha do uso do modo de chaveamento *wormhole* em pacotes, e no interior deste em *flits* contendo a mensagem. Na rede Hermes, o tamanho do flit define a largura do barramento de dados entre roteadores. Ou seja, os conceitos de *flit* e *phit* [DUA97] são equivalentes. Entre roteadores, utiliza-se um controle de fluxo *handshake* tradicional e utilizam-se dois enlaces de comunicação unidirecionais para implementar um canal de comunicação bidirecional, conforme ilustrado na Figura 4 para uma instância da Hermes com *flit* de 8 bits. A ferramenta Atlas permite ao usuário escolher o tamanho do *flit* (dentre um conjunto de valores possíveis) ao gerar instâncias desta NoC .

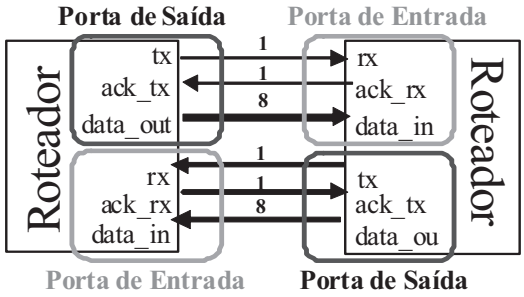

Figura 4 - Interface entre roteadores vizinhos na rede Hermes original [MEL03], ilustrada para uma instância com *flit* de 8 bits.

Os roteadores da NoC Hermes possuem entre três e cinco portas bidirecionais de comunicação uma lógica de arbitragem e uma lógica de controle que implementa o algoritmo de roteamento. As cinco portas bidirecionais são denominadas Norte (N), Sul (S), Leste (E), Oeste (W) e Local (L), esta última usada para o roteador conectar-se ao núcleo a que está vinculado. Note-se que assim a NoC Hermes pressupõe o uso de redes diretas conforme definido por Duato et al. [DUA97]. Cada porta possui uma memória de armazenamento temporária (*buffer*) de entrada associada, com uma estrutura de FIFO. As portas N, S, E, e W conectam-se aos roteadores vizinhos localizados na direção designada pelo nome da porta. A lógica de arbitragem controla o acesso à lógica de roteamento pelas portas de entrada. A lógica de roteamento computa a porta de saída a partir do algoritmo de roteamento, verifica a disponibilidade da porta de saída selecionada e realiza a conexão entre a porta de entrada e a de saída, caso isto seja possível. Depois de terminada a transmissão da mensagem a lógica de controle desfaz a conexão entre portas de entrada e de saída.

Os pacotes da NoC Hermes têm a estrutura mostrada na Figura 5. O primeiro e segundo *flits* do pacote formam o cabeçalho do pacote. O restante deste, de tamanho variável contém os dados do pacote. O primeiro *flit* dá o endereço do roteador destino, enquanto o segundo *flit* determina o tamanho do restante do pacote. Cada roteador possui um endereço único na rede, definido pelas suas coordenadas cartesianas posicionais na malha 2D que forma a rede. Assim, o endereço de cada roteador é formado por um par (abscissa, ordenadas). Uma relação importante existe entre o tamanho do *flit* e o tamanho máximo da rede. Por exemplo, se o *flit* tiver tamanho de 8 bits significa que o tamanho máximo da rede será 16 x 16 roteadores, pois metade do primeiro flit especifica a abscissa do destino e a outra metade especifica a ordenada deste.

Figura 5 - Formato do pacote da NoC Hermes original. Os dois primeiros *flits* formam o cabeçalho do pacote, e os n seguintes são a carga útil deste.

No nível de transporte ocorre o processo de envio do pacote do roteador origem ao roteador destino. Neste nível se realiza o processo de divisão/aglutinação de mensagem (ns) em pacotes (na origem) e a remontagem/divisão dos pacotes em mensagens (no destino). Maiores detalhes sobre a rede Hermes original podem ser encontrados nas referências [MEL03] e [MOR04].

3.1.1 A Rede Intrachip Hermes-VC

No trabalho de Mello [MEL06] encontra-se como primeira contribuição a inserção de canais virtuais na rede Hermes. O uso de canais virtuais tem como vantagens diminuir a contenção na rede

devido a bloqueios do tipo HOL. Este conceito pressupõe que canais físicos possam ser compartilhados por diversos pacotes em redes *wormhole*. Isto é obtido pela quebra do *buffer* de entrada em vários *buffers* (tipicamente menores) que podem servir cada uma a diferentes pacotes e um esquema de multiplexação destas FIFOs para acesso ao canal físico. O número de *buffers* de entrada associado a cada porta dá o número de canais virtuais. A inserção de canais virtuais também faz com que os algoritmos executados pelo roteador seja mais complexos. A interface de comunicação nos roteadores da NoC Hermes-VC é significativamente diferente da interface na rede Hermes original. A Figura 6 apresenta a estrutura desta interface. Note-se que se assume aqui um roteador com *flit* de n bits e **l** canais virtuais. Note-se também que a interface usa controle de fluxo baseado em créditos, em oposição ao controle de fluxo *handshake* visto antes. Em ambas, Hermes e Hermes-VC o controle de fluxo é selecionável.

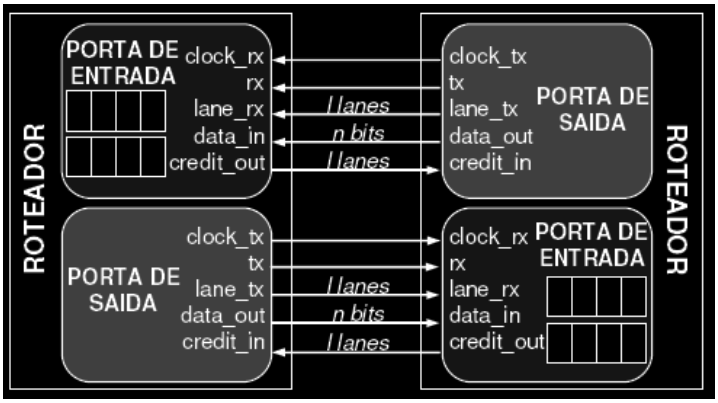

Figura 6 - Interface entre roteadores da Hermes-VC [MEL06]. O *flit* possui tamanho n e o número de canais virtuais é l.

A porta Local na rede Hermes-VC tipicamente não possui canais virtuais. Contudo a plataforma HardNoC que usa esta rede utiliza uma versão modificada, onde todos as portas possuem o mesmo número de canais virtuais.

3.2 Redes com Topologia Toro

Uma rede com topologia toro é similar às redes com topologia malha, porém elas oferecem mais opções de caminhos, pela inclusão de enlaces interligando os roteadores nas extremidades da topologia. Em redes toro 2D, ligam-se os roteadores das extremidades superior aos roteadores da extremidade inferior correspondentes, e os roteadores das extremidades direita com os roteadores das extremidades esquerda correspondentes, usando canais denominados canais de retorno (do inglês *wraparound)*. Rede toro possuem como vantagem a redução à metade de diâmetro da rede. Esse parâmetro determina o caminho mais curto entre nodos mais distantes em uma rede [DUA97]. O menor diâmetro de redes toro também propicia um aumento da largura de banda, bem como a potencial redução do tempo de armazenamento de pacotes nos *buffers* de entrada. No entanto, canais de retorno necessitam fios tipicamente mais longos que os encontrados em redes com topologia malha. Isto faz com que redes toro apresentem uma carga capacitiva tipicamente maior, gerando interferência eletromagnética potencialmente maior o que pode acarretar uma possível redução na freqüência de operação [PAN05]. Redes com esta topologia foram propostas em diversos trabalhos anteriores, tais como, por exemplo, em [MAR02], onde se realiza a prototipação de um SoC contendo uma rede que emprega algoritmo de roteamento baseado em chaveamento por pacotes usando um FPGA Virtex XCV800 da Xilinx.

Um problema importante nestas redes é que o algoritmo XY não é livre de *deadlock* em redes toro. Assim, é necessário projetar outros algoritmos para uso nestas. Como um exemplo de algoritmo de roteamento para rede toro, um trabalho seminal de Dally e Seitz [DAL86] propõe um algoritmo de roteamento livre de *deadlock* para roteamento de pacotes em uma rede toro com chaveamento *wormhole*. Este algoritmo foi implementado como parte de um CI denominado "The Torus Routing Chip", usado na implementação de máquinas paralelas com múltiplos processadores interconectados por uma rede toro 3D. Além deste algoritmo, Dally e Towles propõem, em [DAL01] o uso de uma topologia toro dobrado 2D usando canais virtuais com o mesmo algoritmo de roteamento para implementação de redes intrachip. A topologia toro dobrado não difere estruturalmente de um toro convencional, ela apenas dita uma política de posicionamento relativo dos nodos da rede para uniformizar o comprimento das conexões entre roteadores, evitando as linhas mais longas na conexões *wraparound*, conforme ilustrado na Figura 7.

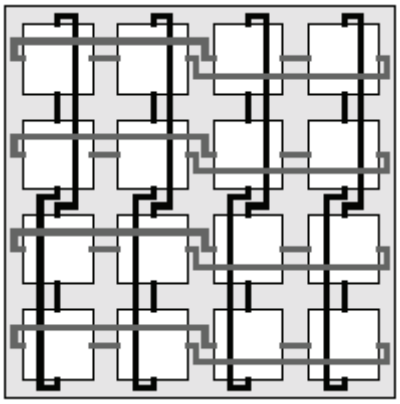

Figura 7 – Estrutura de posicionamento toro dobrado conforme proposto em [DAL01].

Segundo [BAR05], a arquitetura toro necessita de pelo menos dois canais virtuais para ser livre de *deadlock*. Em redes malha, não é necessário o uso de canais virtuais para tornar algoritmos determinísticos como XY livres de *deadlock*. Para algoritmos totalmente adaptativos ou parcialmente adaptativos são necessários três canais virtuais em redes toro e pelo menos dois para redes malha.

Uma das métricas usadas para analisar uma rede é medir a latência, definida como o tempo que decorre entre o início da injeção de um cabeçalho de pacote na rede e a chegada do pacote completo no destino [BAR05]. Em um sistema síncrono este tempo pode ser medido em ciclos de relógio.

Em [BAR05] duas topologias de rede foram comparadas, malha e toro. Observou-se que o maior problema da topologia malha é seu diâmetro, que prejudica a latência máxima dos pacotes na rede. A topologia toro habilita a redução da latência [BAR05] [DUA97]. A comparação inclui gráficos de consumo de energia versus desempenho, consumo de energia versus latência e consumo de energia versus vazão. Diferentes algoritmos de roteamento foram empregados nas comparações, mostrando conclusivamente a vantagem de se usar uma topologia toro em várias situações.

O roteamento nas redes toro é tipicamente mais complexo que em redes malha similares, pois a prevenção de *deadlock* é complicada pela existência dos canais de retorno.

Chi e Chen [CHI04] demonstram a implementação de um roteador para topologia malha ou toro. Este roteador não possui filas de entrada e saída. Os Autores sugerem uma técnica de roteamento inovadora que pode eliminar o problema de bloqueio de pacotes nas filas. Esta técnica é denominada *look-ahead* e consiste em definir o roteamento do pacote antes deste entrar no roteador seguinte. Essa inovação propicia um melhor desempenho em relação a roteadores que utilizam filas do tipo FIFO.

O trabalho realizado por Pande et al. [PAN05], contextualiza o leitor a respeito das diversas variáveis envolvidas na construção de uma NoC tendo como finalidade o uso destas em MPSoCs. Os Autores salientam a necessidade de dar atenção a escolha da topologia a ser usada como forma de minimizar o fenômeno de atrasos em fios longos. Demonstram que o atraso dos fios aumenta exponencialmente ou linearmente dependendo do uso de repetidores, que esse atraso pode exceder o limite de um ciclo de relógio. Segundo os autores, aproximadamente 80% do atraso ocorre nos fios de interconexão, principalmente em fios longos. Além disso, mostram gráficos e tabelas com valores de área, vazão, latência e consumo de energia comparando as NoCs seguintes: (i) CLICHÉ (Chip Level Integration of Communicating Heterogeneous Elements) de Kumar et al.; (ii) a rede toro 2D proposta por Dally e Towles; (iii) OCTAGON, proposta por Karim et al. A contribuição maior do trabalho é o estabelecimento de métricas que habilitam uma análise quantitativa de diferentes topologias de NOCs.

3.3 A rede Intrachip Hermes-TB

A rede Hermes-TB [SCH05] é uma NoC derivada da rede Hermes [MOR04]. A diferença principal encontra-se na topologia utilizada. A rede Hermes [MOR04] é uma rede com topologia malha 2D e a Hermes-TB emprega topologia Toro 2D. A rede Hermes-TB possui, segundo os critérios da Figura 1 as seguintes características para seu algoritmo de roteamento: dinâmico, *unicast*, distribuído, baseado em máquinas de estado e adaptativo. O seu algoritmo de roteamento é uma modificação do algoritmo *West-First*, adaptado a partir da proposta original para redes malha de Glass e Ni [GLA94] [SCH07]. Esta rede utiliza modo de chaveamento *wormhole* sem canais virtuais, controle de fluxo baseado em créditos e política de escalonamento *round robin*.

A estrutura do roteador Hermes-TB é similar ao roteador utilizado no projeto inicial da HERMES sem canal virtual, ilustrado na Figura 4. Cada roteador possui exatamente cinco portas, usa filas de entrada, controle de fluxo baseado em créditos e não emprega canais virtuais.

3.3.1 O Algoritmo de Roteamento da Rede Hermes-TB

O algoritmo de roteamento utilizado na rede Hermes-TB é uma adaptação do algoritmo *west-first* de Glass e Ni [SCH07]. Conforme a Figura 1 ele pode ser classificado como:

- *Dinâmico*, quanto ao momento de realização do roteamento;
- *Unicast*, quanto à quantidade de destinos das mensagens;
- *Distribuído*, quanto ao local onde a decisão de roteamento é tomada;
- *Baseado em máquinas de estado*, quanto à forma de implementação;
- *Adaptativo*, quanto ao processo de seleção de caminhos;
- *Progressivo*, quanto à progressividade;
- *Não mínimo*, quanto à minimalidade;
- *Parcial*, quanto ao número de caminhos empregado dentre todos disponíveis.

A adaptação consistiu em inserir modificações no comportamento do algoritmo quando o pacote chega a roteadores com valores máximos de coordenada (abscissas ou ordenadas). Todo o texto desta Seção faz referência à Figura 8.

Os passos descritos representam a busca pela direção que se deve tomar após a análise do estado em que se encontra o pacote. Para isto, observa-se a posição X/Y do roteador corrente, tendo como pontos de referência:

- O valor de abscissa máximo, representado por Max(X), e o valor de ordenada máxima representado por Max(Y).
- O valor de abscissa do destino representado por D(X) e o valor de ordenada do destino, representado por D(Y).
- O valor de abscissa do roteador corrente, representado por A(X) e o valor de ordenada do roteador corrente, representado por A(Y).
- O valor médio de abscissa da rede representado por (Max(X)+1)/2 e o valor médio de ordenada, representado por (Max(Y)+1)/2.

Algoritmo de Roteamento da Rede Hermes-TB	
1	Xos<=D(X)-A(X) Yos<=D(Y)-A(Y)
2	**SE** Xos=0 **E** Yos=0 ***ENTAO***
3	**SE** free(Local) ***ENTAO*** Porta<=Local
4	**SENAO**
5	**SE** IN=Local ***ENTAO***
6	**SE** A(X)=Max(X) **E** (D(X)=0 **OU** D(X)<((Max(X)+1)/2)) ***ENTAO***
7	**SE** free(Leste) ***ENTAO*** Porta<=Leste
8	**SENAO**
9	**SE** A(X)=0 **E** D(X)=Max(X) ***ENTAO***
10	**SE** free(Oeste) ***ENTAO*** Porta<=Oeste
11	**SENAO**
12	**SE** A(Y)=Max(Y) **E** D(X)>=A(X) **E** (D(Y)=0 **OU** D(Y)<((Max(Y)+1)/2)) ***ENTAO***
13	**SE** free(Norte) ***ENTAO*** Porta<=Norte
14	**SENAO**
15	**SE** A(Y)=0 **E** D(X)>=A(X) **E** (D(Y)=Max(Y) **OU** D(Y)>((Max(Y)+1)/2)) ***ENTAO***
16	**SE** free(Sul) ***ENTAO*** Porta<=Sul
17	**FIM SE**
18	**SE** ((IN=Local) **E** (A(X)>0 **OU** A(X)<Max(X)) **E** (A(Y)>0 **OU** A(Y)<Max(Y)))
19	**OU** (IN <> Local) ***ENTAO***
20	**SE** Xos<0 **E** free(Oeste) ***ENTAO***
21	Porta<=Oeste
22	**SENAO SE** Xos>0 **E** Yos<0 **E** free(Leste) ***ENTAO***
23	Porta<=Leste
24	**SENAO SE** Xos>0 **E** Yos<0 **E** free(Sul) ***ENTAO***
25	Porta<=Sul
26	**SENAO SE** Xos>0 **E** Yos>0 **E** free(Leste) ***ENTAO***
27	Porta<=Leste
28	**SENAO SE** Xos>0 **E** Yos>0 **E** free(Norte) ***ENTAO***
29	Porta<=Norte
30	**SENAO SE** Xos>0 **E** Yos=0 **E** free(Leste) ***ENTAO***
31	Porta<=Leste
32	**SENAO SE** Xos=0 **E** Yos<0 **E** free(Sul) ***ENTAO***
33	Porta<=Sul
34	**SENAO SE** Xos=0 **E** Yos>0 **E** free(Norte) ***ENTAO***
35	Porta<=Norte
36	**SENAO** Reescalona_Pacote()
37	**SENAO** Reescalona_Pacote()
38	**FIM SE**
39	**FIM SE**

Figura 8 – O algoritmo de roteamento da rede Hermes-TB, adaptado em [SCH07] do algoritmo de roteamento *West-first* de Glass e Ni [GLA94].

O algoritmo é dividido em quatro blocos principais: o primeiro determina as distâncias em X e Y do pacote ao destino; o segundo testa se o pacote chegou ao seu destino e age de acordo; o terceiro serve, sobretudo se o pacote ingressou na rede no roteador corrente, ou seja, se o pacote acabou de ingressar na rede de comunicação; o quarto é o caso geral, onde se testam várias situações do estado do pacote e age-se de acordo. Antes de atribuir um pacote a uma porta de saída esta é testada para verificar se está livre (rotina `free(ID_porta)`).

No primeiro bloco, correspondendo à linha 1 da Figura 8, são atribuídos valores a `Xos` e `Yos`, que representam as distâncias ou deslocamentos (em inglês, *offset*) do pacote, do ponto em que este se encontra até o roteador destino. Computa-se este valor pela diferença entre destino e origem. Naturalmente, `Xos` é o valor de *offset* das abscissas enquanto `Yos` é o valor de *offset* das ordenadas.

O segundo bloco é formado pelas linhas 2 e 3 da Figura 8. Após a determinação de *offset* do pacote, busca-se saber se o destino foi alcançado. Para tanto, é necessário apenas verificar se `Xos=Yos=0`. Se isto ocorrer, encaminha-se o pacote para a porta Local e o roteamento é concluído.

O terceiro bloco do algoritmo, nas linhas 4 a 17 da Figura 8, é executado se o pacote atingiu o roteador atual pela porta Local, ou seja, se o pacote acabou de entrar na rede. Este bloco define a restrição básica para gerar do algoritmo livre de *deadlock* em redes toro [DUA97], qual seja a de que os enlaces de *wraparound* somente podem ser utilizados como primeiro enlace de um caminho na rede. O valor verdadeiro para a condição testada na linha 5 (`IN=Local`) indica que o pacote ingressou na rede no roteador corrente, pois a porta de entrada do mesmo (`IN`) é a Local. Neste caso, testam-se os quatro conjuntos de condições que podem levar a selecionar o uso de um enlace *wraparound*. Dada a restrição básica citada, sabe-se que isto só pode acontecer a partir de um roteador nos limites externos da rede. Os conjuntos de condições testadas e as ações correspondentes são:

- Linhas 6 e 7 – Se o pacote está na borda Leste da rede (`A(X)=Max(X)`) e o destino do mesmo é nodo situado na borda Oeste (`D(X)=0`), ou pelo menos na metade esquerda da rede (`D(X)<((Max(X)+1)/2)`), toma-se o enlace *wraparound* pela porta Leste;
- Linhas 8 a 10 – Caso o conjunto de condições anteriores não se verifique, se o pacote está na borda Oeste da rede (`A(X)=0`) e o destino do mesmo é nodo situado na borda Leste (`D(X)=Max(X)`), toma-se o enlace *wraparound* pela porta Oeste[1];
- Linhas 11 a 13 – Caso os conjuntos de condições anteriores não se verifiquem, se o pacote está na borda Norte da rede (`A(Y)=Max(Y)`) e o destino não está à esquerda do nodo atual (`D(X)>=A(X)`) e este destino ou está na borda Sul (`D(Y)=0`), ou pelo menos na metade inferior da rede (`D(Y)<((Max(Y)+1)/2)`), toma-se o enlace *wraparound* pela porta Norte;
- Linhas 14 a 16 – Caso os conjuntos de condições anteriores não se verifiquem, se o pacote está na borda Sul da rede (`A(Y)=0`) e o destino não está à esquerda do nodo atual (`D(X)>=A(X)`) e este destino ou está na borda Norte (`D(Y)=Max(Y)`), ou pelo menos na metade superior da rede (`D(Y)>((Max(Y)+1)/2)`), toma-se o enlace *wraparound* pela porta Sul[2].

O quarto bloco do algoritmo (linha 18 a 39 da **Figura 8**) consiste no caso geral. Observe-se as linhas 18 e 19, que estabelecem a condição para rotear algum pacote durante a execução deste bloco.

[1] Note-se que os conjuntos de condições testadas nas bordas Leste e Oeste não são totalmente simétricos.
[2] Note-se que os quatro conjuntos de condições testadas não exaurem todas as possibilidades.

O roteamento neste bloco é feito para dois tipos de enlace:

- Os que ou não são o primeiro enlace no caminho do pacote ao longo da rede, o que acontece quando o pacote não entra no roteador atual pela porta local (`IN<>Local`);
- Os que são o primeiro enlace (`IN=Local`) mas não estão em nenhuma das bordas da rede (`(A(X)>0 OU A(X)<Max(X)) E (A(Y)>0 OU A(Y)<Max(Y))`).

Se o teste acima der verdadeiro, isto garante que o pacote será roteado neste bloco. Senão, o pacote deve ser reescalonado para roteamento, pois não há porta livre que ele possa usar no momento. Caso a condição seja verdadeira e lembrando que se trata de um algoritmo do tipo *West-first*, as linhas 20 e 21 contêm o teste básico para rotear o pacote para Oeste, o que acontece sempre que o destino estiver à esquerda da posição atual do pacote. Se isto não acontecer, alcança-se as linhas 22 a 36, que tentam rotear para alguma porta que não seja a Oeste. A sequência de ações neste trecho garante certo grau de adaptatividade ao algoritmo e é a seguinte:

- Se o destino está no quadrante Sudeste em relação à posição atual do pacote (linhas 22 a 25, teste `Xos>0 E Yos<0`), tenta-se ir para Leste ou para Sul, nesta ordem, adaptativamente;
- Se o destino está no quadrante Nordeste em relação à posição atual do pacote (linhas 26 a 29, teste `Xos>0 E Yos<0`), tenta-se ir para Leste ou para Norte, nesta ordem, adaptativamente;
- Nas linhas 30 e 31, se o destino já está alinhado verticalmente com a posição atual do pacote (`Yos=0`) e encontra-se à direita desta posição, tenta-se ir para Leste;
- Nas linhas 32 e 33, se o destino já está alinhado horizontalmente com a posição atual do pacote (`Xos=0`) e encontra-se abaixo desta posição, tenta-se ir para Sul;
- Nas linhas 34 e 35, se o destino já está alinhado horizontalmente com a posição atual do pacote (`Xos=0`) e encontra-se acima desta posição, tenta-se ir para Norte.

Caso nenhuma das tentativas de roteamento da lista acima consiga atribuir o pacote a uma porta livre, o pacote é reescalonado para novo roteamento (linha 36).

3.3.2 Exemplo de funcionamento do algoritmo

O algoritmo verifica o posicionamento do pacote. Em seguida, deve selecionar uma das condições previstas. Para exemplificar mostram-se quatro situações em uma rede toro 5x5.

O primeiro exemplo implica computar o caminho tendo como origem a coordenada (4,2) e destino em (1,2). Neste caso, a melhor opção é utilizar o canal de retorno, usando como critério a quantidade de passos necessária. Usando o canal de retorno é necessário atravessar 2 segmentos (em inglês *hops*) da rede, enquanto na abordagem usando, por exemplo, uma malha 2D, seriam 3 passos.

Analisando a execução passo a passo do algoritmo tem-se o seguinte. Inicia-se verificando a posição inicial, depois se verifica a chegada ao destino. Como se parte de uma porta local e estando sobre uma borda, a condição da linha 4 é atendida. Por se estar na borda e o destino estar localizado antes do meio da matriz, a condição da linha 6 é atendida. Assim, executa-se o passo que realiza a transposição, usando o canal de retorno pela porta Leste do extremo leste da matriz. Ao utilizar a porta Leste alcança-se a porta Oeste do outro extremo, alcançando assim o destino no passo seguinte.

O segundo exemplo consiste em computar o caminho tendo como origem a coordenada (2,0) e como destino (2,4). Neste caso, a melhor opção é utilizar o canal de retorno, usando como critério a

quantidade de passos necessária. Assim, usando o canal de retorno é necessário 1 passo enquanto na abordagem usando, por exemplo, uma malha 2D, seriam 4 passos.

Analisando a execução passo a passo do algoritmo tem-se o seguinte. Inicia-se verificando a posição inicial, depois se verifica a chegada ao destino. Como se parte de uma porta local e se está na borda, a condição da linha 4 é atendida. Por se estar na borda superior e o destino estar localizado na borda inferior, a condição da linha 11 é atendida. Assim, executa-se o passo que realiza a transposição usando o canal de retorno pela porta Norte do extremo Norte da matriz. Ao utilizar a porta Norte alcança-se a porta Sul do outro extremo, chegando assim ao destino.

O terceiro exemplo consiste em computar o caminho tendo como origem a coordenada (2,2) e como destino (2,0), estando o caminho localizado entre (2,2) e (3,2) bloqueado.

Neste caso, a melhor opção é seguir de forma adaptativa na direção Sul.

Analisando a execução passo a passo do algoritmo tem-se o seguinte. Inicia-se verificando a posição inicial, e depois a chegada ao destino. Como se parte de uma porta local e se está no centro da matriz, a condição da linha 18 é atendida. Por se estar no centro e o destino estar localizado na borda inferior, a linha 31 é atendida. Assim, executa-se o passo que toma o caminho pela porta Sul e alcança a porta Norte do outro roteador, chegando ao destino no próximo passo.

O quarto exemplo consiste em executar o caminho tendo como origem a coordenada (2,2) e como destino a coordenada (4,2), estando o caminho localizado entre (2,2) e (2,3) bloqueado.

Neste caso, a melhor opção é seguir de forma adaptativa na direção Leste.

Analisando a execução passo a passo do algoritmo tem-se o seguinte. Inicia-se verificando a posição inicial, depois verificando a chegada ao destino. Como se parte de uma porta Local e se está no centro da matriz, a condição da linha 18 é atendida. Por se estar no centro e o destino estar localizado na borda Leste, a linha 25 é atendida. Assim, executa-se o passo que escolhe o caminho pela porta Leste e alcança a porta Oeste do outro roteador, chegando ao destino no próximo passo.

4 SIMULAÇÃO DO HARDWARE DA NOC HERMES-TB

Este trabalho fez uso de uma metodologia baseada no domínio de projetos anteriores do GAPH necessários a este trabalho. Neste grupo enquadram-se as pesquisas que deram origem a rede Hermes-TB, como uma evolução descrita no Capítulo 3. A implementação em VHDL utilizada nessa pesquisa é produto da evolução da rede Hermes. Um resumo da metodologia descrita aqui se encontra na Figura 9. Inicialmente, realizou-se um estudo de ferramentas, seguido da prática com estas para criar e exercitar instâncias da rede Hermes-TB. Depois, se utilizou HardNoC, um ambiente de validação em hardware de instâncias da NoC Hermes-VC [MOR08]. Neste ambiente os roteadores estão conectados a IPs geradores de tráfego sintético baseados em modelos estatísticos. Posteriormente, realizaram-se as adequações necessárias para mudar o a ambiente HardNoC para usar a Hermes TB ao invés da Hermes-VC, criando assim o ambiente HardNoC-TB. Ambos ambientes foram exercitados por simulação. Para garantir a operacionalidade da rede Hermes-TB realizou-se uma simulação com atrasos computados após a síntese do ambiente.

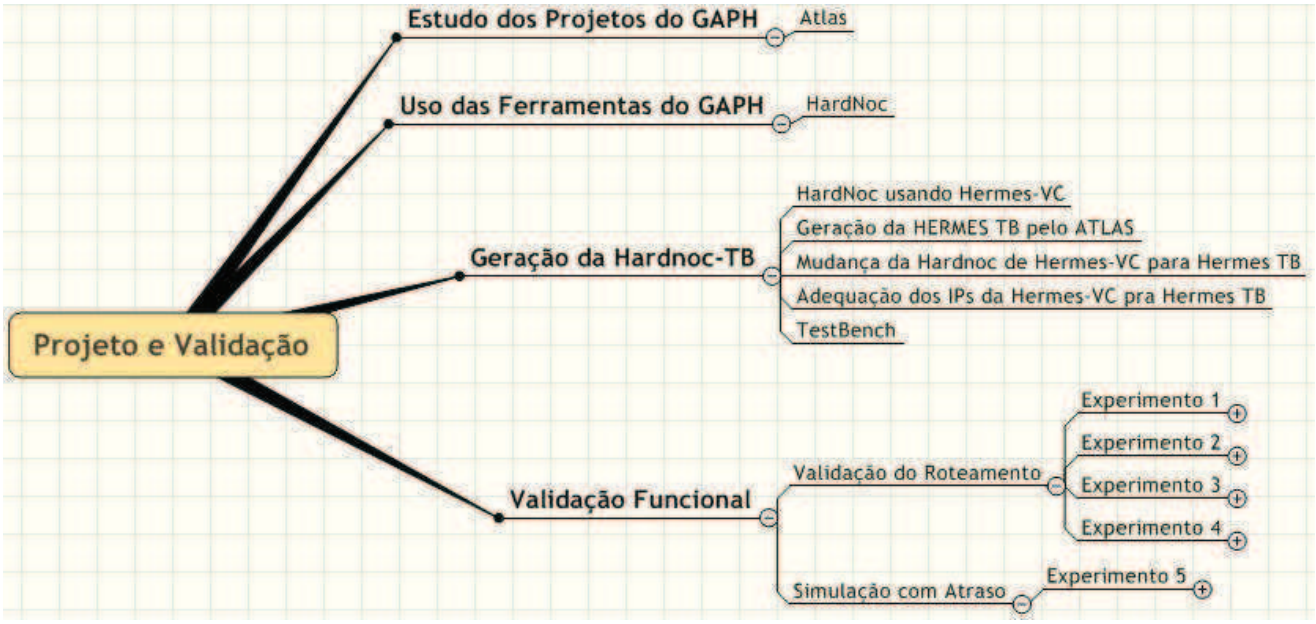

Figura 9 - Etapas realizadas no processo de validação da NoC Hermes-TB.

4.1 O ambiente de validação de NoCs HardNoC

O diagrama de blocos do ambiente HardNoC é apresentado na Figura 10. Este ambiente é composto de uma instância da rede Hermes-VC com topologia malha 2x3, onde roteadores possuem dois canais virtuais por porta. Esta rede interconecta cinco IPs testadores de rede que podem ser programados através da NoC para gerar e receber tráfego e computar e informar algumas estatísticas sobre o tráfego recebido. A interação com o ambiente é habilitada através do IP serial, usado para comunicação com um computador hospedeiro do tipo PC. Através de um software especificamente desenvolvido, é possível programar os IPs testadores com padrões de tráfego, disparar o processo de geração de tráfegos programados para todos os Testadores e capturar de volta as estatísticas de tráfego resultantes do processo de execução.

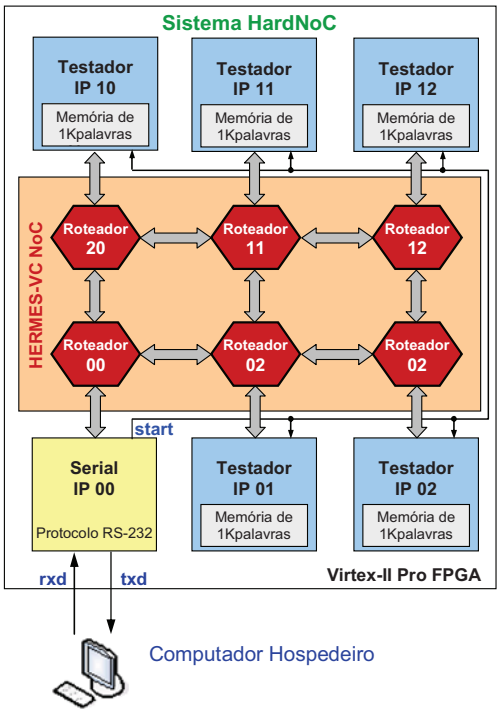

Figura 10 - Diagrama de blocos da plataforma HardNoC [MOR08].

Este ambiente de validação de NoCs em hardware foi especificamente desenvolvido para a rede Hermes-VC. Ele pode ser facilmente adaptado para instâncias desta rede com número diferente de roteadores, mas não necessariamente é simples seu emprego com outras redes. O protótipo inicial do ambiente, descrito em [MOR08] foi validado sobre uma plataforma de prototipação da empresa Digilent denominada XUP-V2PRO, contendo um FPGA Xilinx XC2VP30 da família VirtexII-Pro. Esta plataforma pode ser vista na Figura 45 do Capítulo 5.

A NoC do ambiente HardNoC pode ser gerada automaticamente a partir do uso do ambiente Atlas [GAP08]. Atlas é um ambiente desenvolvido no GAPH para habilitar a geração automatizada de descrições sintetizáveis de redes intrachip. Além desta capacidade, Atlas é útil para geração de tráfego, simulação de redes com injeção destes tráfegos e avaliação de desempenho das redes após uma simulação de tráfego.

Para geração do hardware e prototipação, empregaram-se as ferramentas comerciais ISE e Modelsim. A primeira constitui um ambiente de síntese voltado para FPGAs da Xilinx. A segunda é um ambiente de simulação disponibilizado para várias tecnologias, incluindo ASICs, FPGAs de vários fabricantes e diversas linguagens de descrição de hardware e validação, tais como VHDL, Verilog e SystemC.

4.2 Hermes-VC versus Hermes-TB e alterações no ambiente HardNoC

A rede Hermes-TB possui algumas características que diferem da rede Hermes-VC como já descrito no Capítulo 3. A rede Hermes-VC utiliza no ambiente HardNoC topologia malha 2D com dois canais virtuais, enquanto a Hermes-TB possui topologia toro e não possui canais virtuais.

Após uma análise das redes e suas interfaces externas, chegou-se a conclusão ser necessário, para adaptar o ambiente HardNoC para a Hermes-TB, trabalhar na geração e utilização dos sinais denominados `lane_tx` e `lane_rx` no IP Serial e nos IPs Testadores. As interfaces em questão são ilustradas na Figura 11. Da Figura 12 até a Figura 17 estão ilustradas as modificações conduzidas, que

transformam o código VHDL do ambiente HardNoC no ambiente HardNoC-TB. Nestas figuras, a coluna da esquerda contém o código da HardNoC enquanto a coluna direita contém o código HardNoC-TB.

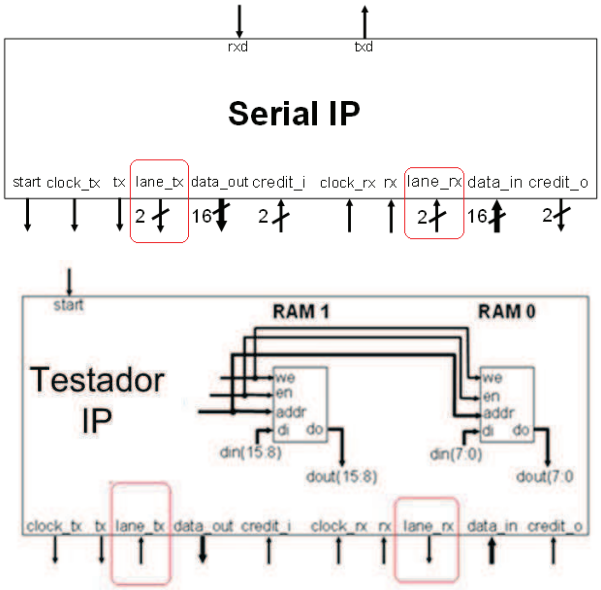

Figura 11 - Interface externa dos IPs Serial e Testadores da HardNoC.

Na Figura 12 observa-se a mudança na biblioteca principal, alterando de HermesPackage para HermesTBPackage. Essa alteração é necessária devido ao fato da biblioteca principal conter as informações de constantes, variáveis que são utilizadas nos diversos códigos que compõem o projeto. As informações contidas nesse pacote compreendem constantes que permitem definir características dos roteadores como tamanho de *flits*, tipos específicos, subtipos e funções associadas.

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.std_logic_arith.CONV_STD_LOGIC_VECTOR;
use work.HermesPackage.all;
```

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.std_logic_arith.CONV_STD_LOGIC_VECTOR;
use work.HermesTBPackage.all;
```

Figura 12 - Alteração da menção do pacote básico de definições da rede Hermes-VC para o pacote correspondente da rede Hermes-TB.

Na Figura 13 observa-se a mudança necessária para tratar a diferença entre a rede Hermes-VC que possui dois canais virtuais e a Hermes-TB que não possui canais virtuais. A solução encontrada foi inutilizar a linha que possui os sinais `Lane_rx`, `Lane_tx` e mudar o tipo utilizado para os sinais `credit_o` e `credit_i` do tipo `arrayNrot_reglane` para `regNrot`, que é definido no HermesTBPackage.

```
architecture HardNoC of HardNoC is

    signal clock, clock_rx, clock_tx: regNrot;
    signal rx, tx: regNrot;
    signal lane_rx, lane_tx: arrayNrot_regNlane;
    signal data_in, data_out : arrayNrot_regflit;
    signal credit_o, credit_i: arrayNrot_regNlane;
    signal n_reset, reset, ck, locked, start
: std_logic;

begin
```

```
architecture HardNoC of HardNoC is

    signal clock, clock_rx, clock_tx: regNrot;
    signal rx, tx: regNrot;
--  signal lane_rx, lane_tx: arrayNrot_regNlane;
    signal data_in, data_out : arrayNrot_regflit;
    signal credit_o, credit_i: regNrot;
    signal n_reset, reset, ck, locked, start
: std_logic;

begin
```

Figura 13 – Eliminação da menção a canais virtuais e adequação dos tipos utilizados.

Para garantir um sinal de relógio que atinja todos os pontos do FPGA praticamente ao mesmo

tempo, FPGAs dispões de módulos de controle de escorregamento de relógio. No caso dos FPGAs da Xilinx, estes módulos são chamados gerenciadores digitais de relógio (em inglês, Digital Clock Manager ou DCM). Na Figura 14 o sinal de relógio descrito como `ck` e o sinal de `reset` foram colocados fora do DCM, indicando a sua função fora da especificação do mapa de portas. Esta tarefa foi feita apenas para fins de validação funcional do sistema, sendo desfeita para a realização de simulação com atrasos.

Figura 14 - Remoção temporária do DCM.

A Figura 15 mostra as alterações iniciais do arquivo de descrição da NoC onde se adéqua a posição de ligação dos sinais de relógio que será utilizada na Hermes-TB. Essa distribuição espacial pode ser vista nos desenho da topologia, bem como na planta baixa obtida após a síntese. Esta alteração modifica a implementação original, pois aquela não estava de acordo com a documentação do projeto. A implementação inverte, em relação à documentação a orientação dos roteadores (no exemplo da Figura, mostra-se a inserção dos roteadores de coordenadas XY=20 e XY=21 e a remoção dos roteadores XY=02 e XY=12.

Figura 15 - Reorganização dos roteadores para adequação do projeto à documentação.

Na Figura 16 nota-se as alterações que foram necessárias no nível de instanciação da NoC Hermes-TB para eliminar o uso de canais virtuais da plataforma HardNoC original, pela eliminação do sinais `lane_rxLocal` e `lane_txLocal (others=>'0')`, mudança esta testada com sucesso na simulação.

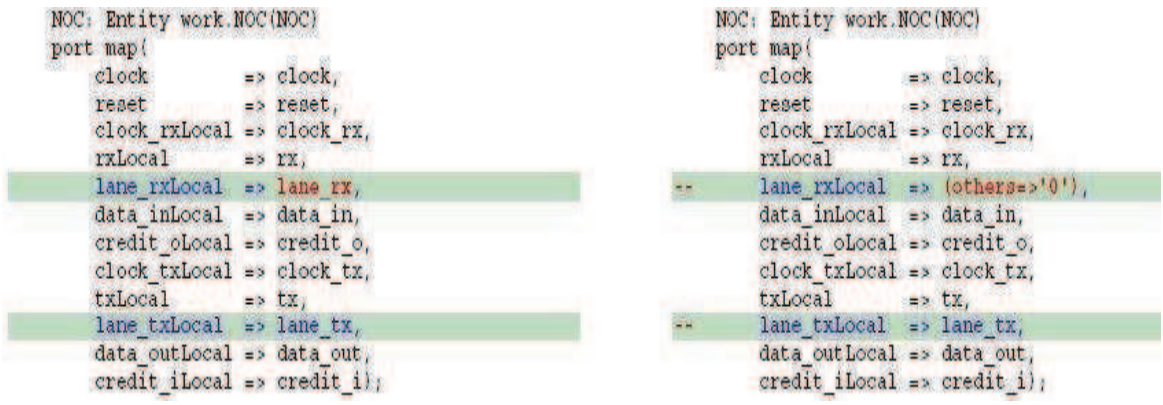

Figura 16 - Eliminação dos canais virtuais em nível de instanciação da NoC.

A Figura 17 mostra a mudança mais significativa, qual seja a maneira escolhida de resolver o

problema de a rede original possuir dois canais virtuais e da rede Hermes-TB não fazer uso desses canais. Mostram-se aqui apenas as alterações realizadas no IP Serial (IP 00), existindo mudanças similares em todos os IPs e nos próprios roteadores. Em um primeiro momento, observou-se a linha que continha o sinal denominado `lane_tx`, depois no sinal `credit_i(L1)`, que representa o sinal de crédito para o primeiro canal virtual. Notou-se ser necessário realizar a ligação do mesmo de forma explícita com o sinal credit_o do Roteador 00, enquanto que o sinal `credit_i(L2)`, desnecessário na Hermes TB, foi aterrado.

```
IP0000 : Entity work.Serial
port map(
    clock    => clock(N0000),
    reset    => reset,
    address  => addressN0000,
    start    => start,
    ---------------- Interface Serial ------------
    rxd      => rxd,
    txd      => txd,
    ------------------ Interface NoC --------------
    clock_tx => clock_rx(N0000),
    tx       => rx(N0000),
    lane_tx  => lane_rx(N0000),
    data_out => data_in(N0000),
    credit_i => credit_o(N0000),

    clock_rx => clock_tx(N0000),
    rx       => tx(N0000),
    lane_rx  => lane_tx(N0000),
    data_in  => data_out(N0000),
    credit_o => credit_i(N0000));
```

```
IP0000 : Entity work.Serial
port map(
    clock    => clock(N0000),
    reset    => reset,
    address  => addressN0000,
    start    => start,
    ---------------- Interface Serial ------------
    rxd      => rxd,
    txd      => txd,
    ------------------ Interface NoC --------------
    clock_tx => clock_rx(N0000),
    tx       => rx(N0000),
--  lane_tx  => lane_rx(N0000),
    data_out => data_in(N0000),
    credit_i(L1) => credit_o(N0000),
    credit_i(L2) => '0',
    clock_rx => clock_tx(N0000),
    rx       => tx(N0000),
    lane_rx  => (others=>'0'),
    data_in  => data_out(N0000),
    credit_o(L1) => credit_i(N0000),
    credit_o(L2) => '0');
```

Figura 17 – Explicitação do uso do canal 1 pelo IP Serial.

Após as alterações esboçadas nesta Seção, obteve-se um versão da HardNoC-TB, pronta para simulação funcional.

4.3 Simulação da HardNoC-TB

Com intuito de validar a rede Hermes-TB em um ambiente funcional, além das modificações realizadas na NoC, utilizam-se os IPs desenvolvido pelo GAPH. A idéia é inicialmente gerar simulações sem levar em conta atrasos nos fios, depois realizar simulações com atraso, buscando uma situação mais próxima da realidade, para só então partir-se para a prototipação da HardNoC-TB.

Os experimentos deste Capítulo na realidade validam o roteamento da rede Hermes-TB, testando canais de retorno, e enviando pacotes aos 5 IPs Testadores a partir do IP Serial.

Quando se trabalha com simuladores como o Modelsim é possível utilizar *scripts* com o objetivo de agilizar o processo de simulação, visto que este requer diversos passos. A Figura 18 mostra os principais blocos que compõem o script usado para realizar a simulação sem atraso:

- Definição da biblioteca de trabalho utilizada no projeto (marcado com 1 na Figura);
- Mapeamento de elementos necessários ao projeto, como a área local, bem como bibliotecas importantes no processo de simulação. Neste exemplo existe a chamada de duas bibliotecas. A primeira, de nome UNISIM é utilizada em simulação funcional (sem atraso) de componentes específicos de FPGAs da Xilinx. No caso da simulação de temporização (com atrasos) deve-se usar a SIMPRIM. Esta contém a descrição VITAL, contendo informações precisas de atraso dos componentes Xilinx (marcado com 2 na Figura);

- Comandos de compilação dos fontes VHDL, descrevendo os elementos que compõem o projeto e que serão utilizados no processo de simulação (marcado com 3 na Figura);
- Comando que realiza o disparo da simulação (marcado com 4 na Figura);
- Comando **do** realiza o disparo de outro *script*. **N**este caso, dispara-se a execução de um *script* que possui sobretudo o detalhamento dos sinais a serem visualizados. O comando **`run`** indica o tempo da simulação. Ao executar este o Modelsim realiza a simulação e gera as formas de onda definidas no *script* W_Routers_IPs.do (marcado com 5 na Figura).

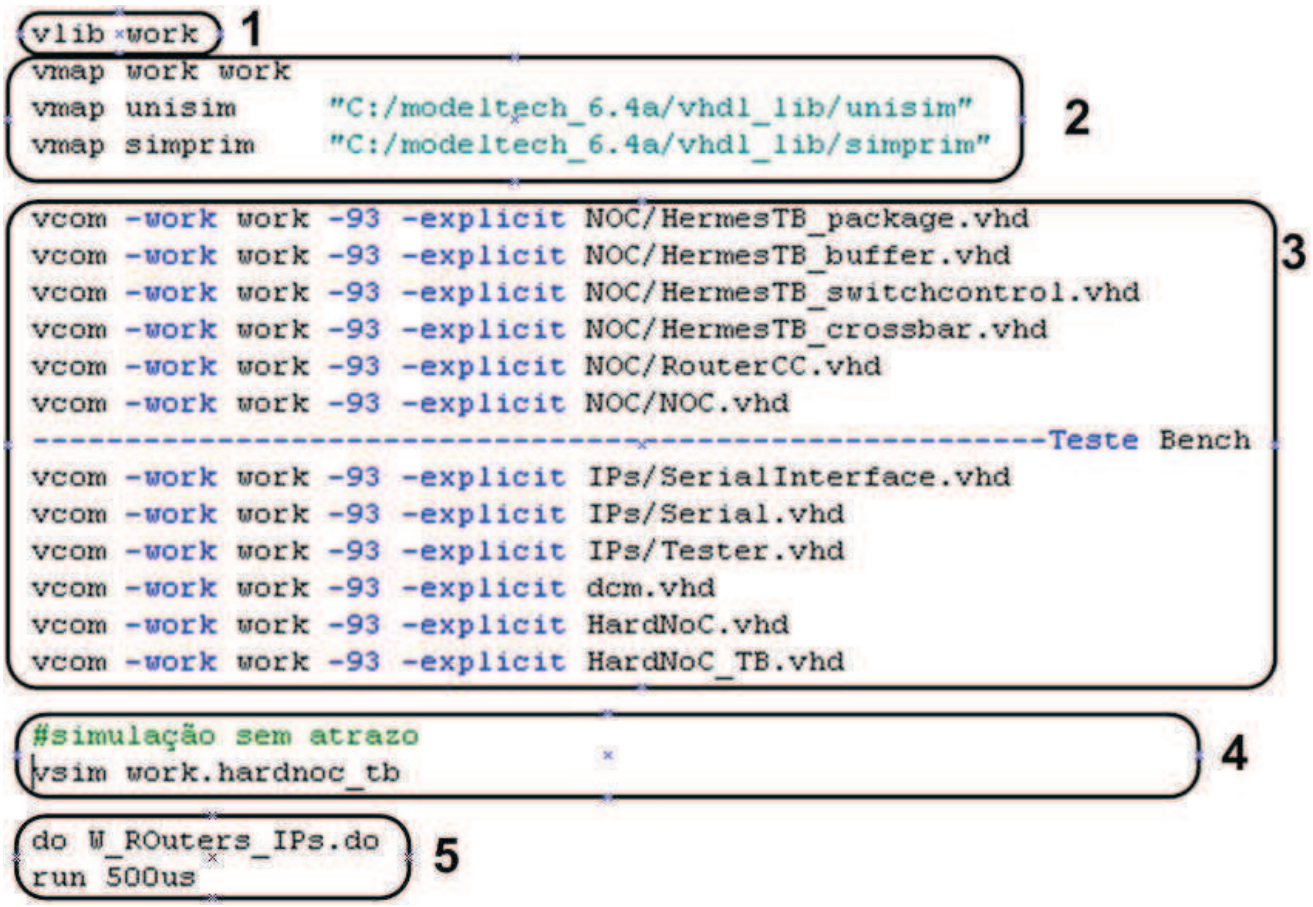

Figura 18 - *Script* Modelsim usado para simulação sem atraso.

A forma usada para transmitir os dados através da rede no ambiente HardNoC é determinada em [MOR08]. O funcionamento dos pacotes dentro da rede é realizado da seguinte forma.

Existem algumas formas do pacote trafegar pelo ambiente HardNoC, de acordo com a posição que ele se encontra. Descreve-se a seguir os formatos de pacote usados nos diferentes pontos do ambiente.

4.3.1 Pacotes Oriundos do Hospedeiro para o IP Serial

Os pacotes gerados pelo software denominado SerialSoftware e enviados de um PC hospedeiro para a plataforma HardNoC têm a estrutura ilustrada na Figura 19.

Figura 19 – Formatos de pacotes recebidos do Hospedeiro pelo IP Serial. Todos os dados são de 8 bits.

Existem três tipos de mensagens oriundas do hospedeiro para o IP Serial.

A primeira é o comando de leitura. Nesta, o primeiro dado contém o código do comando (0, que identifica uma operação de leitura); o segundo dado indica o IP destino do pacote (de onde se quer ler dados), o terceiro especifica o número de palavras a ser lido do IP, e o quarto e quinto indicam o endereço inicial de leitura dos dados no IP.

O segundo tipo de mensagem é o comando de escrita representada no primeiro dado pelo numero 1. O segundo, o terceiro, o quarto e o quinto dados têm interpretação similar as do comando de leitura. Do sexto ao nono dados está o dado a ser escrito (32 bits).

O último tipo de mensagem é um comando de inicialização do processo de geração de tráfego e armazenamento de estatísticas nos IPs testadores. Este comando é de apenas um byte, sem nenhuma outra informação. Com o auxílio destes três comandos cria-se o fluxo de dados do mundo exterior para a HardNoC. Cada um dos pacotes chega ao IP Serial onde é decodificado, gerando pacotes no formato da rede ou outras ações implícitas no pacote proveniente do hospedeiro.

4.3.2 Pacotes Trocados entre IPs da NoC Hermes-TB

O IP serial executa a formatação dos dados de e para a NoC usando dois comandos, conforme descrito na Figura 20. Os IPs testadores também podem ler e escrever dados em outros IPs da NoC e usam estes mesmos formatos.

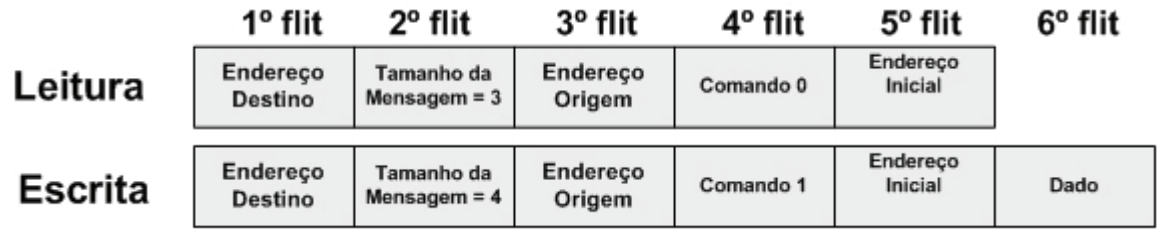

Figura 20 – Formato dos pacotes enviados para a NoC pelo IP Serial ou pelos IPs Testadores. Cada *flit* é **de 16 bits.**

O primeiro destes comandos é de leitura. Nele, o primeiro *flit* contém o endereço de destino, o segundo *flit* possui o tamanho da mensagem (sempre igual a 3), o terceiro *flit* especifica o endereço de origem (no caso, será o endereço do IP que envia a mensagem), o quarto *flit* contém o código 0, que indica tratar-se de comando de leitura e o quinto *flit* contém o endereço do dado a ser lido.

O segundo comando é o comando de escrita onde o primeiro *flit* contém o endereço de destino, o segundo o tamanho da mensagem, o terceiro especifica o endereço de origem (no caso, será o endereço do IP que envia a mensagem), o quarto contém o identificador de comando de escrita (1), o quinto possui o endereço de escrita e o sexto possui o dado a ser escrito.

Para cada um dos casos, o pacote é recebido e tratado conforme as máquinas de estado do IP

que recebe o pacote, podendo este ser o IP Serial ou um dos IP Testadores. Obviamente, os roteadores da rede manipulam este tipo de pacotes, que é compatível com o formato de pacote da Hermes-TB.

4.3.3 Visão Geral da Simulação da HardNoC-TB

A Figura 21 ilustra o diagrama de forma em termos gerais, contendo todos os principais objetos necessários aos experimentos de validação da HardNoC-TB. Neste diagrama existe uma divisão dos sinais em 4 grupos distintos: Os sinais de `clock` e `reset`, os sinais do IP serial, os sinais de roteadores, e os sinais de IPs Testadores.

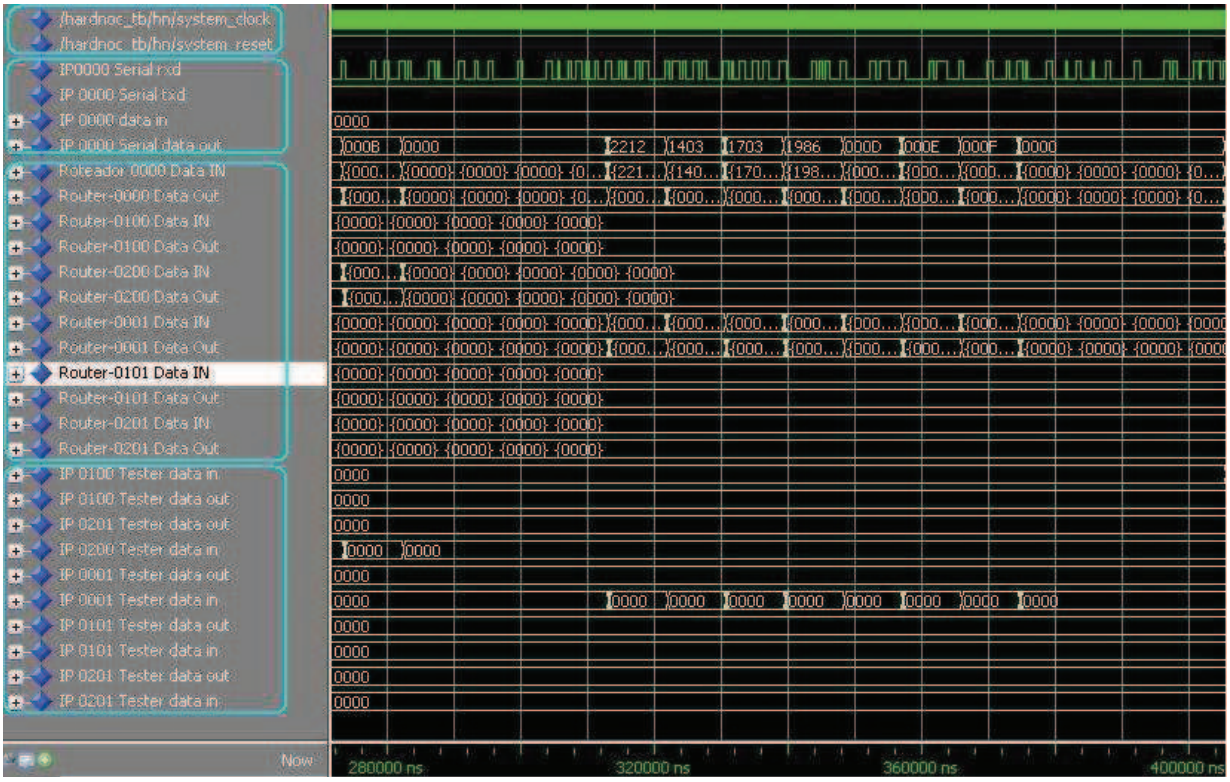

Figura 21 - Diagrama de formas de onda geral da simulação, enfatizando sinais a serem analisados, neste caso apenas os sinais principais da serial, dos roteadores e dos testadores.

No primeiro grupo, existem os sinais de relógio e de reset como uma referência aos ciclos necessários para realizar uma operação.

No segundo grupo existem sinais do IP 00 que é o IP serial: `rxd, txd`, `data in` e `data out`. Estes sinais são importantes nos experimentos porque a partir deles comprova-se o funcionamento da comunicação com o hospedeiro, observação feita da seguinte maneira: o sinal entra pela linha indicada como data in e seguindo o ciclo de relógio ela sai em data out. Os sinais `rxd` e `txd` indicam o momento em que ocorre a recepção e transmissão, respectivamente.

O terceiro grupo é composto de linhas com a representação dos roteadores com seus respectivos sinais de entrada e saída. Nos experimentos individuais esses sinais serão tratados em mais detalhes.

O quarto grupo é composto de sinais de entrada e saída dos IPs testadores.

4.3.4 Experimento 1: Validação da primeira mensagem

Utilizam-se aqui figuras para ilustrar a operação dos experimentos, indicando os caminhos produzidos pela aplicação do algoritmo de roteamento. A Figura 22 mostra o detalhamento da

estrutura interna e o esboço das interfaces dos diferentes módulos que compõem a HardNoC-TB. Os módulos são:

- O roteador e suas portas de entrada e saída (a);
- A interface de IP Testador com a porta Local de seu Roteador (b);
- Ligação da porta local dos roteadores com o IP Serial (c).

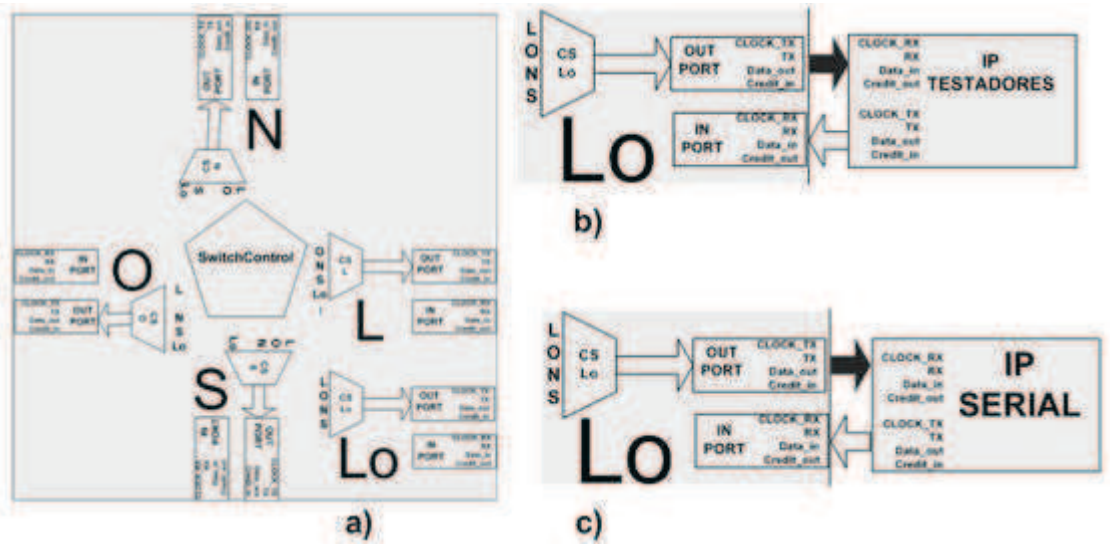

Figura 22 – Situações de roteamento local: (a) Roteador de rede Hermes-TB, com as cinco portas e seus buffers; (b) Detalhe de conexão da porta local dos roteadores com seus respectivos sinais de ligação ao IP Testador; (c) Detalhe de conexão da porta local dos roteadores com os seus respectivos sinais de ligação ao IP Serial.

Para realizar a simulação foi usado o software Modelsim, e escrito um *testbench* para a NoC. O arquivo utilizado nos experimentos denomina-se HardNoC_TB.vhd.

O componente denominado IP Serial é responsável pelo recebimento dos comandos pelo hospedeiro e pela decodificação dos mesmos e posterior envio destes a algum componente denominado IP Testador. Assim, o *testbench* possui capacidade de ler arquivos contendo comandos, com o objetivo de simular a operação do hospedeiro de comunicando com o IP Serial. Estes comandos são armazenados em um arquivo de entrada denominado **`in_tb.txt`**, declarado conforme ilustra o trecho VHDL do testbench da **Figura 23** e com um conteúdo exemplificado na Figura 24.

```
architecture HardNoC_TB of HardNoC_TB is
file ARQ_IN: TEXT open READ_MODE is "in_tb.txt";
file ARQ_OUT: TEXT open WRITE_MODE is "out_tb.txt";
```

Figura 23 – Visão parcial do código do *testbench* com a declaração dos arquivos de entrada e saída de testes.

O *testbench* é responsável pela produção e leitura de todos os estímulos necessários para a execução da simulação.

O trecho do arquivo de entrada da Figura 24 mostra o início de uma mensagem do hospedeiro para a HardNoC-TB, que envia oito palavras de dados a serem localizadas em um IP Testador. Estas devem percorrer o caminho que tem origem no IP Serial até o IP Testador de número 21. Estas palavras são 1203, 1601, 2009, 0377, 0001, 0002, 0003 e 0004, todas com tamanho de 16 bits. A **Figura 25** mostra uma forma de onda onde pacotes saem pela porta do IP Serial em direção à porta Local do Roteador 00. Nesta Figura observa-se apenas o final de cada pacote, identificado pelo valor

de seu último *flit*. O último dos pacotes apresenta um problema derivado da visualização com Modelsim, visto que no lugar do dado 0004 surge um valor 0000. Ao executar uma ampliação na forma de onda percebe-se que a simulação opera de forma correta, conforme esperado. Este problema se repete em múltiplas figuras similares a seguir.

```
55 -- Sinal de sincronismo

01 -- Comando de Escrita
21 -- Destino IP 21
08 -- Numero de Palavras
00 -- Parte Alta
00 -- Parte Baixa (Endereço Inicial)
12 -- Parte Alta
03 -- Parte Baixa (Palavra = 0)
16 -- Parte Alta
01 -- Parte Baixa (Palavra = 1)
20 -- Parte Alta
09 -- Parte Baixa (Palavra = 2)
03 -- Parte Alta
77 -- Parte Baixa (Palavra = 3)
00 -- Parte Alta
01 -- Parte Baixa (Palavra = 4)
00 -- Parte Alta
02 -- Parte Baixa (Palavra = 5)
00 -- Parte Alta
03 -- Parte Baixa (Palavra = 6)
00 -- Parte Alta
04 -- Parte Baixa (Palavra = 7)
```

Figura 24 – Conteúdo parcial do arquivo de entrada de nome `in_tb.txt` com envio de sinal do hospedeiro para a HardNoC-TB usado na simulação pelo *testbench* denominado HardNoC_TB.vhd

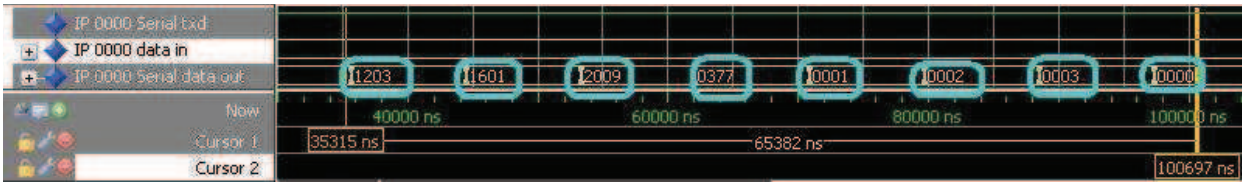

Figura 25 – Forma de onda revelando o fluxo de pacotes que entram na NoC a partir da saída do IP Serial (sinal DataOut). O valor do último pacote não aparece correto, mas uma ampliação desta onda revela o comportamento esperado.

O hardware do IP Serial monta os pacotes no formato esperado pela Hermes-TB. Segundo [MOR08], cada pacote que circula na rede da HardNoC (e da mesma forma da HardNoC-TB) especifica uma dentre quatro possíveis operações: Leitura, Escrita, Retorno de Leitura e Dado. A estrutura dos pacotes para efetuar estas operações é detalhada na Figura 26. Vale a pena salientar que os dois primeiros tipos de pacote são uma repetição da Figura 20.

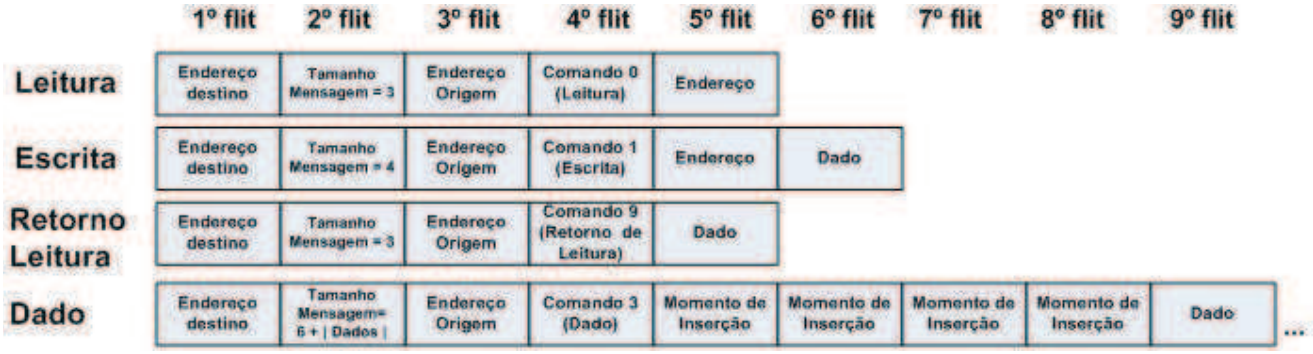

Figura 26 – Os quatro formatos de pacotes que circulam no interior da HardNoC-TB.

A forma de onda da Figura 27 demonstra como se realiza o processo de envio do primeiro pacote (um comando de escrita), que tem no primeiro *flit* o valor 0021, que corresponde ao destino deste pacote; no segundo *flit*, tem-se o tamanho que já é pré-definido conforme o padrão na Figura 26. No terceiro *flit* aparece o endereço de origem. Nesse caso ele se origina do IP Serial denominado IP 00. No quinto *flit* tem-se a posição deste pacote na mensagem. Como se trata do primeiro, ele possui valor 0000; no sexto *flit* aparece o dado que corresponde à carga útil do pacote.

Observe-se na Figura 28 o quinto *flit*, que identifica a posição deste pacote na mensagem. Como ele é o penúltimo de oito, este campo vale 0006. No sexto *flit* aparece o dado que representa a carga útil do pacote.

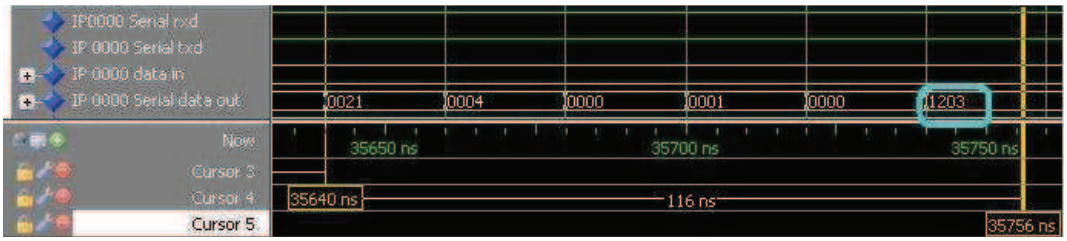

Figura 27 - Detalhe do primeiro pacote enviado do roteador XY=00 (IP Serial) ao roteador XY=21 (Testador).

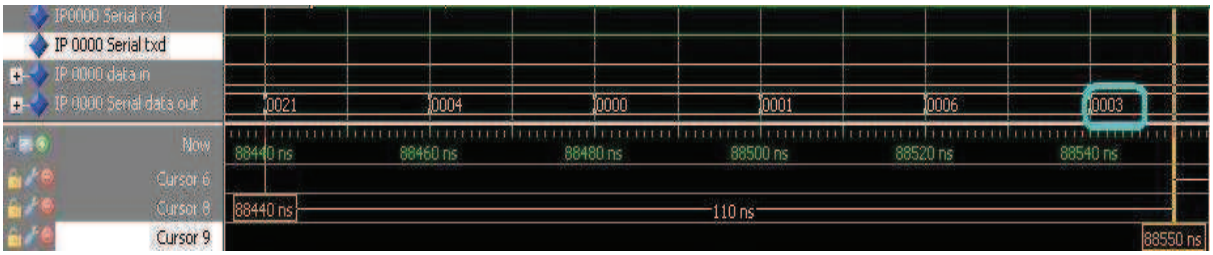

Figura 28 - Detalhe do penúltimo pacote enviado do roteador XY=00 ao roteador XY=21.

4.3.5 Experimento 2: Validação do Roteamento da Porta Local do IP 00 ao IP 21

Para ilustrar a validação funcional e a operação do modelo VHDL da HardNoC-TB é necessário observar o algoritmo de roteamento da HardNoC-TB 3X2. Neste experimento, os pacotes têm como origem o IP Serial (IP 00) e como destino o IP Testador 21. Inicialmente, a seqüência de pacotes é enviada pelo usuário a partir do hospedeiro pela porta serial, nesse caso através do arquivo de extensão `txt` que simula a alimentação de dados. Depois, os pacotes provenientes do hospedeiro são empacotados no formato da NoC Hermes-TB pelo IP Serial e enviados à rede. Neste meio são encaminhados aos roteadores e pelos canais Norte, Sul, Leste e Oeste são repassados a outros roteadores até a chegada ao destino onde serão encaminhados à porta Local.

Para garantir facilidade de referência, as formas de onda utilizadas são todas semelhantes entre

si, mudando os objetos, mas mantendo a estrutura geral de sinais apresentada na Figura 21, conforme o caminho a ser utilizado.

No Experimento 2, cujo fluxo é ilustrado na Figura 29, o pacote deve chegar ao IP Serial, denominado IP 00, pela sua porta In, conectada ao hospedeiro, sendo encaminhado em direção à porta Out do mesmo IP, a partir de onde atinge o Roteador 00 pela porta Local. No roteador será encaminhado para a porta Oeste, em direção à porta Leste do Roteador 20, pelo canal *wraparound*. Chegando ao roteador 20, será encaminhado à porta Norte do mesmo, sendo recebido pela porta Sul do roteador 21, onde, por ser este o roteador conectado ao IP Testador destino a mensagem segue para a porta Local do mesmo. A Figura 30 detalha a simulação deste processo.

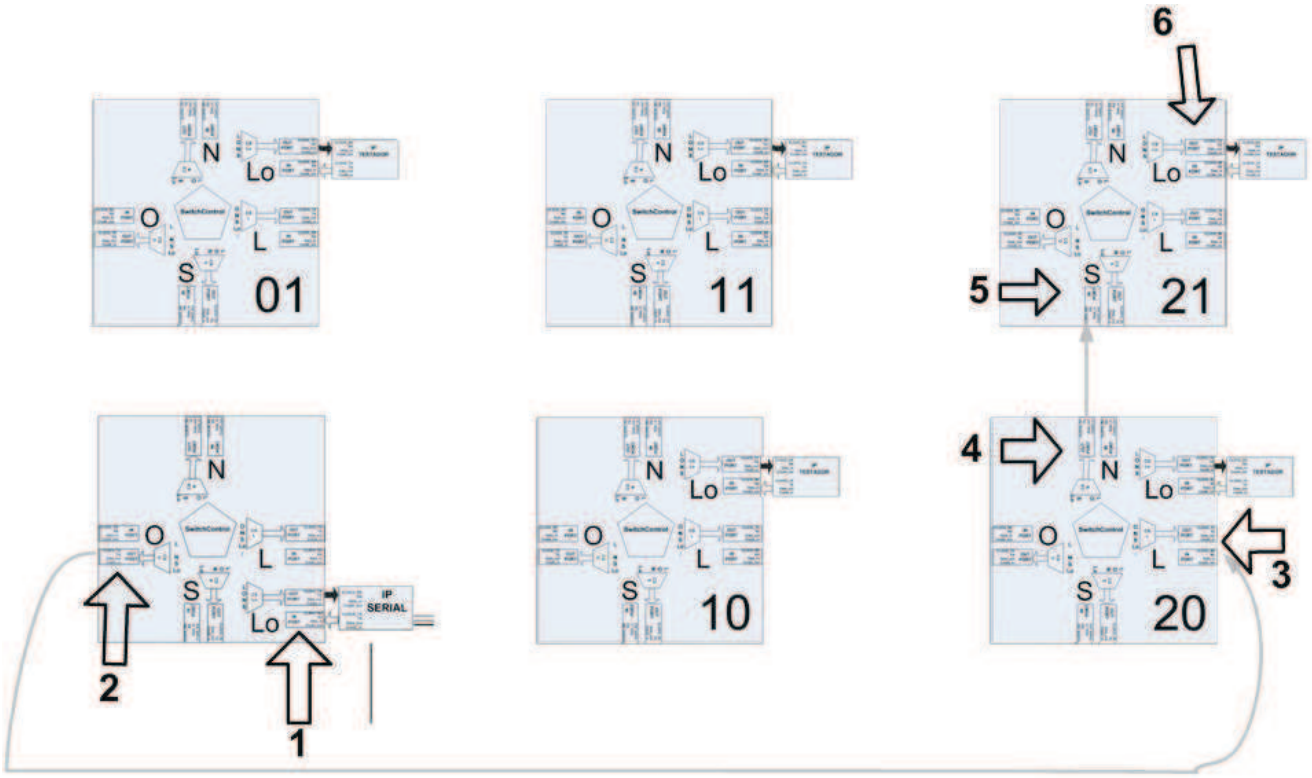

Figura 29 - Caminho realizado do IP 00 ao IP 21 passando pelos Roteadores 00, depois Roteador 20, chegando ao Roteador 21.

A partir desse momento explica-se o esquema da Figura 29, e detalhada em formas de onda na Figura 30. Observa-se que os passos realizados são numerados da mesma forma no desenho e na forma de onda. O detalhamento dos passos é o seguinte:

- 1 – O IP Serial (IP 00) Recebe a sequência de oito dados para transmitir a outro IP da NoC. Os dados de cada pacote (carga útil) são, em hexadecimal: 1203, 1601, 2009, 0377, 0001, 0002, 0003, 0004. Cada um dos dados gera um pacote para a rede, que penetram esta pelo Roteador 00, via porta de entrada 4 (Local) deste. Neste roteador, o algoritmo de roteamento é executado, a tabela de chaveamento é atualizada, realizando a ligação entre a porta Local do Roteador 00 com a porta de saída 1 (Oeste) do mesmo Roteador, conforme se pode acompanhar na forma de onda da Figura 30 com a ligação da porta de entrada 4 (Local) à porta de saída 1 (Oeste) no Roteador 00.
- 2 – A porta de saída Oeste do Roteador 00 recebe os pacotes da porta de entrada Local do mesmo Roteador. O Roteador 00 interage com a porta de entrada Leste do Roteador 20

conforme a Figura 30, onde se nota a ligação da porta de saída Oeste do Roteador 00 com a porta de entrada Leste (0) do Roteador 20.

- 3 – O Roteador 20 recebe o pacote oriundo da porta de saída Oeste do Roteador 00. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada, realizando a ligação entre a porta de entrada 0 (Leste) e a porta de saída 2 (Norte) do mesmo Roteador.
- 4 – A porta de saída Norte do Roteador 20 recebe o pacote oriundo da sua porta de entrada Leste. O Roteador 20 interage com a porta de entrada Sul do Roteador 21 conforme a **Figura 30**, onde se nota a ligação da porta de saída 2 (Norte) do Roteador 20 com a porta de entrada 3 (Sul) do Roteador 21.

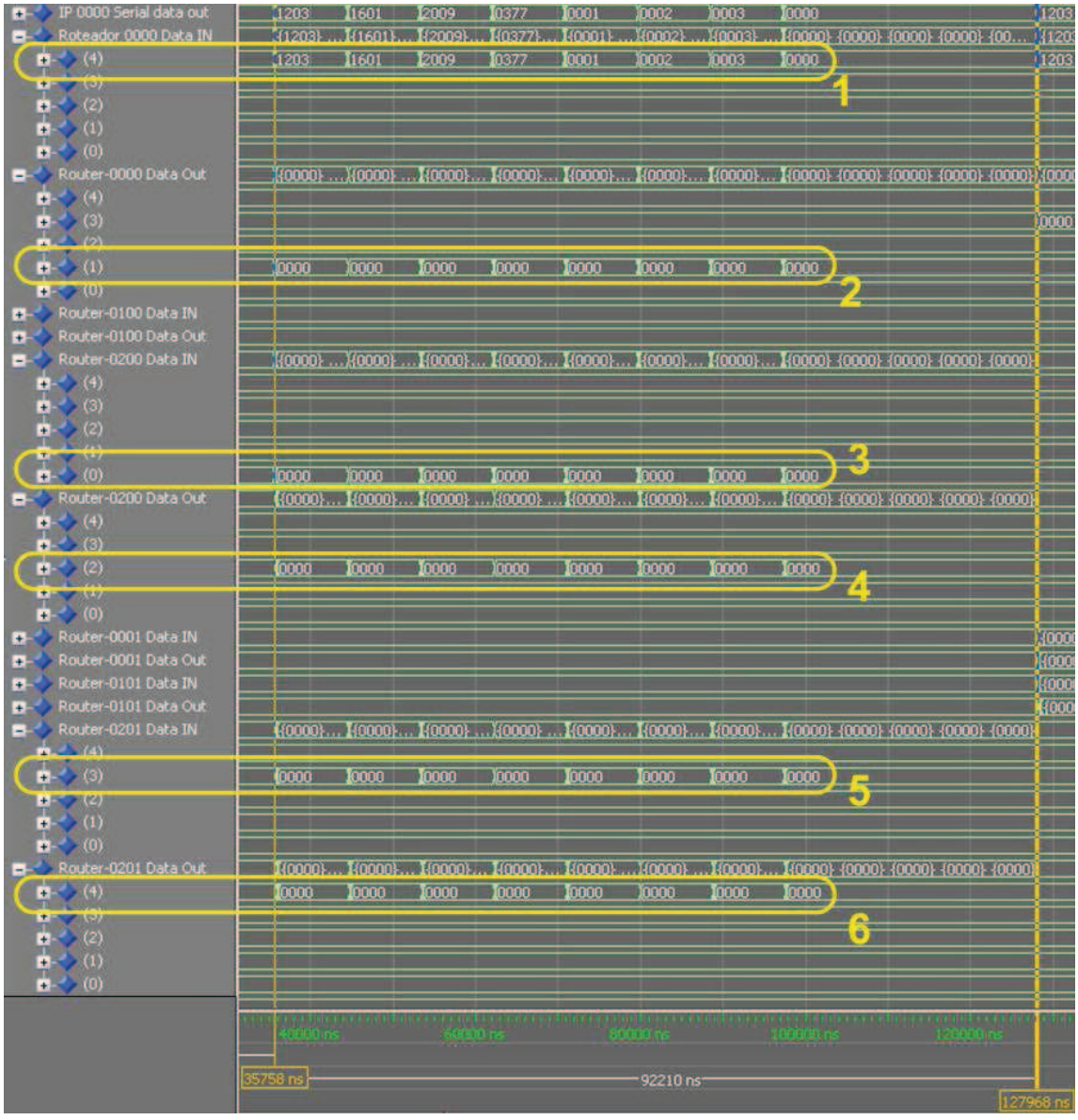

Figura 30 - Forma de onda que mostra em uma simulação sem atraso o processo de transmissão e recepção para percorrer o caminho do IP 00 ao IP 21, passando pelo Roteador 00, depois o Roteador 20, chegando finalmente ao Roteador 21. Os sinais mostrados para os pontos 2-6 não mostram o valor esperado como ocorre em 1, mas isto é devido a falhas de detalhamento da visualização e não a erros de simulação.

- 5 – A porta de entrada 3 (Sul) do Roteador 21 recebe os pacotes oriundos da porta Norte do Roteador 20. O algoritmo de roteamento é executado e a tabela de chaveamento é atualizada,

realizando a ligação entre a porta de entrada 3 (Sul) do Roteador 21 e a porta de saída 4 (Local) do mesmo Roteador, como se pode visualizar na forma de onda da Figura 30.

- 6 – A porta Local do Roteador 21 recebe o pacote oriundo da porta Sul do mesmo roteador. Nesse instante, um a um os *flits* saem da rede e seguem para a memória do IP Testador 21. O IP Testador fica aguardando o sinal de **`Start`** para após isto começar a operar, injetando o padrão de tráfego programado em si para o interior da NOC.

4.3.6 Experimento 3: Validação do Roteamento da Porta Local do IP 00 ao IP 11

Neste experimento, um pacote deve chegar ao IP 00, pela porta In, seguindo em direção a porta out, do mesmo IP, sendo direcionado ao Roteador 00 pela porta Local. Este roteador encaminhará o pacote para a porta Sul, em direção à porta Norte do Roteador 01. Chegando pelo canal de retorno a este roteador o pacote será re-encaminhado à porta Leste do mesmo sendo recebido pela porta Oeste do Roteador 11. Uma vez que este é o destino, o pacote segue pela porta local do mesmo. Esta operação é detalhada em simulação no diagrama de formas de onda da Figura 32. Observa-se que os passos realizados são numerados da mesma forma no desenho e na forma de onda. O detalhamento dos passos é o seguinte:

- 1 – Recebe-se a seqüência de oito pacotes contendo as seguintes cargas úteis (em hexadecimal): 1203, 1975, 0505, 1977, 0005, 0006, 0007, 0008. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada, realizando a ligação entre a porta de entrada 4 (Local) do Roteador 00 com a porta de saída 3 (Sul) do mesmo roteador.
- 2 – No Roteador 00, a porta de saída Sul recebe o pacote da porta de entrada Local. A porta de saída Sul, pela topologia está conectada à porta de entrada 2 (Norte) do Roteador 01.

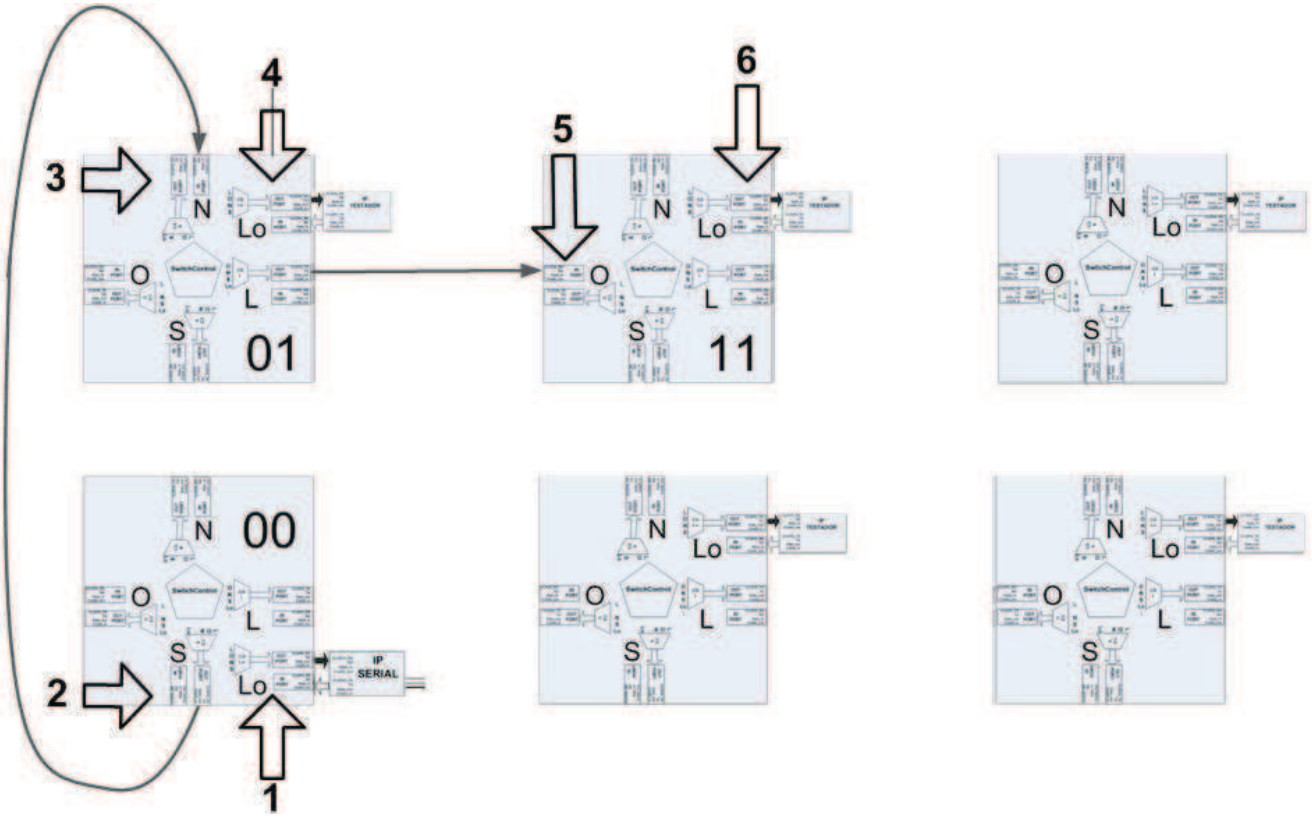

Figura 31 - Caminho realizado do IP 00 ao IP 11 passando pelo Roteador 00, depois pelo Roteador 01, atingindo o Roteador 11.

- 3 – O Roteador 01 recebe o pacote oriundo da porta de saída Sul do Roteador 00. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada, realizando-se a ligação entre a porta de entrada 2 (Norte) e a porta de saída 0 (Leste) no Roteador 01.
- 4 – No Roteador 01 ainda, a porta Leste recebe o pacote oriundo da porta Norte do mesmo roteador. A porta de saída Leste, pela topologia está conectada à porta de entrada 1 (Oeste) do Roteador 11.
- 5 – No Roteador 11, a porta de entrada 1 (Oeste) recebe o pacote oriundo da porta de saída 0 (Leste) do Roteador 01. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada realizando a ligação entre a porta de entrada 1 (Oeste) do Roteador 11 e a porta de saída 4 (Local) do mesmo roteador.
- 6 – No Roteador 11 a porta de saída 4 (Local) recebe o pacote oriundo da porta de entrada 1 (Oeste) do mesmo roteador. Nesse instante, gradativamente os pacotes atingem o IP Testador 11, são interpretados e os dados são armazenados na memória do IP 11, neste caso um IP Testador. O IP Testador fica a partir daí aguardando o sinal de **`Start`** para então simular o tráfego dentro da NOC.

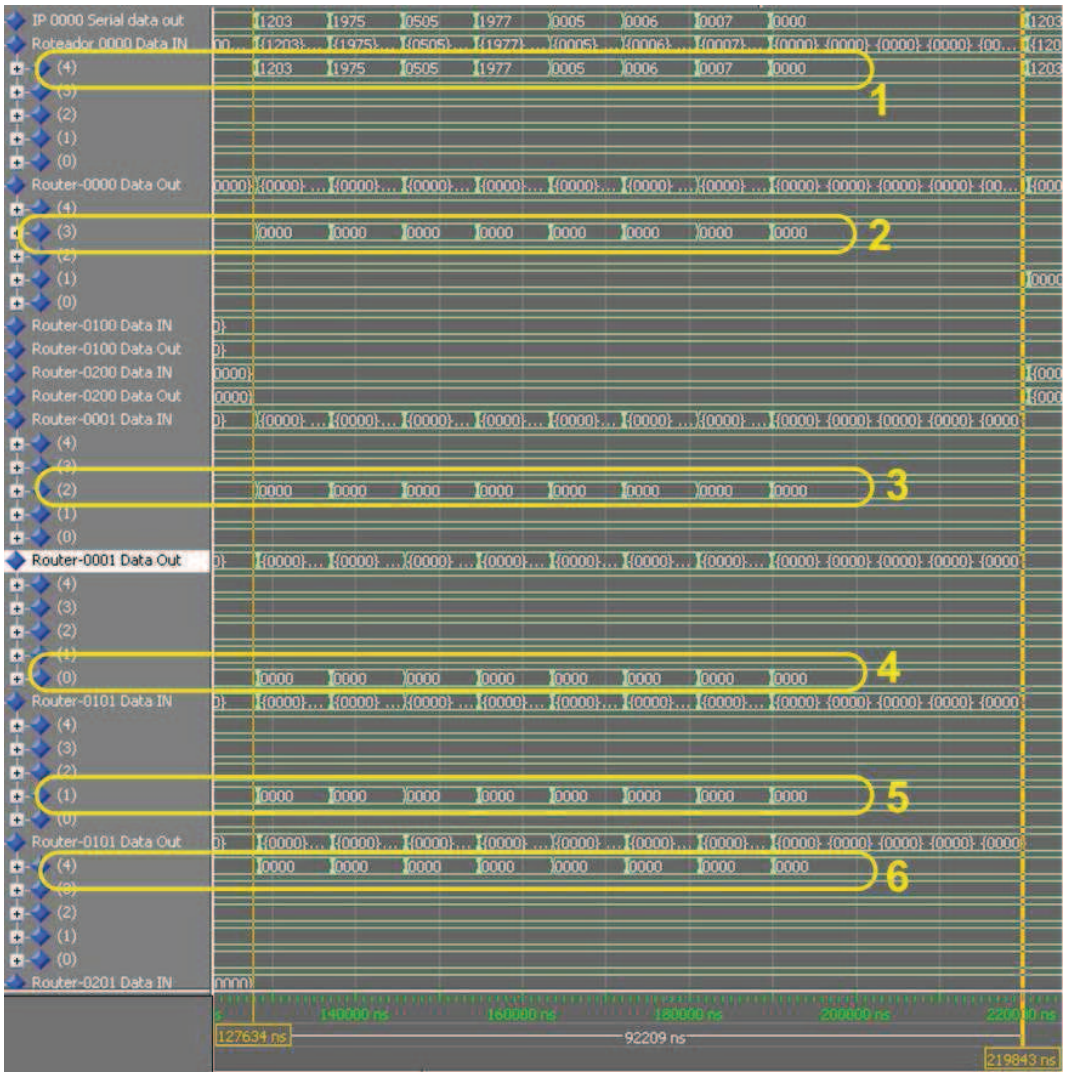

Figura 32 - Forma de onda demonstrando em uma simulação sem atraso o processo de transmissão de dados do IP 00 ao IP 11, passando pelo Roteador 00, depois pelo Roteador 10, chegando ao Roteador 11. Os sinais mostrados para os pontos 2-6 não mostram o valor esperado como ocorre em 1, mas isto é devido a falhas de detalhamento da visualização e não a erros de simulação.

4.3.7 Experimento 4: Validação do roteamento do IP 00 aos IPs 20, 01 e 10

Neste experimento validam-se três caminhos distintos da NoC Hermes-TB, percorridos de forma consecutiva. Todos envolvem apenas dois roteadores entre origem e destino e todos partem do IP 00 (o IP Serial). O esquema geral do experimento é detalhado na Figura 33 e as formas de onda para cada pacote são apresentadas na Figura 34, Figura 35 e na Figura 36. O procedimento pode ser detalhado como segue:

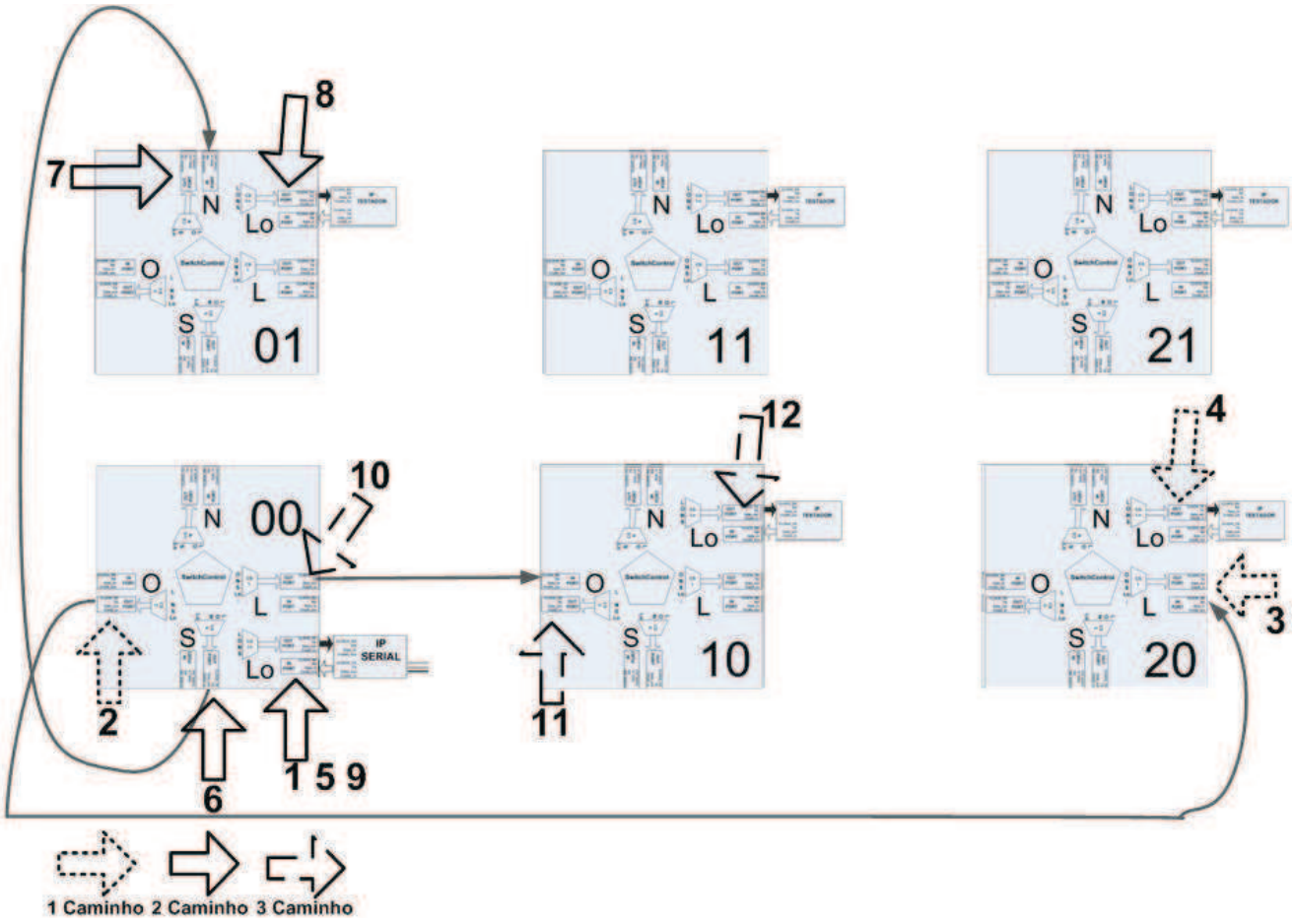

Figura 33 – Três caminhos distintos exercitados de forma consecutiva por três conjuntos de pacotes do IP 00 a três IPs distintos, o 20, o 10 e o 01. Nos três casos, a fonte dos pacotes é o Roteador 00 e cada caminho passa apenas por dois roteadores.

- 1 – O IP Serial (IP 00) recebe a seqüência de oito dados (em hexadecimal: 1203, 1975, 2304, 1007, 0009, 000A, 000B, 000C) e cria um pacote para enviar cada um deles através da NoC. Para cada pacote que entra no Roteador 00, o algoritmo de roteamento é executado, a tabela de chaveamento é atualizada, realizando a ligação entre a porta de entrada 4 (Local) do Roteador 00 e a porta de saída 1 (Oeste) do mesmo roteador.
- 2 – O Roteador 00 recebe na sua porta de saída 1 (Oeste) os pacotes da porta de entrada 4 (Local) do mesmo roteador.
- 3 – O Roteador 20 recebe na sua porta de entrada 0 (Leste) os pacotes oriundos da porta de saída 1 (Oeste) do Roteador 00. O algoritmo de roteamento é executado e a tabela de chaveamento é atualizada, realizando a ligação entre a porta de entrada 0 (Leste) com a porta de saída 4 (Local) do mesmo roteador.

- 4 – O Roteador 20 recebe na porta de saída 4 (Local) os pacotes oriundos da porta de entrada 0 (Leste) do mesmo roteador. Nesse instante, gradativamente os pacotes chegam e são armazenados na memória do IP 20, neste caso um IP Testador. O IP Testador fica então aguardando o sinal de Start para iniciar o processo de geração de tráfego programado nele para injeção na NoC.

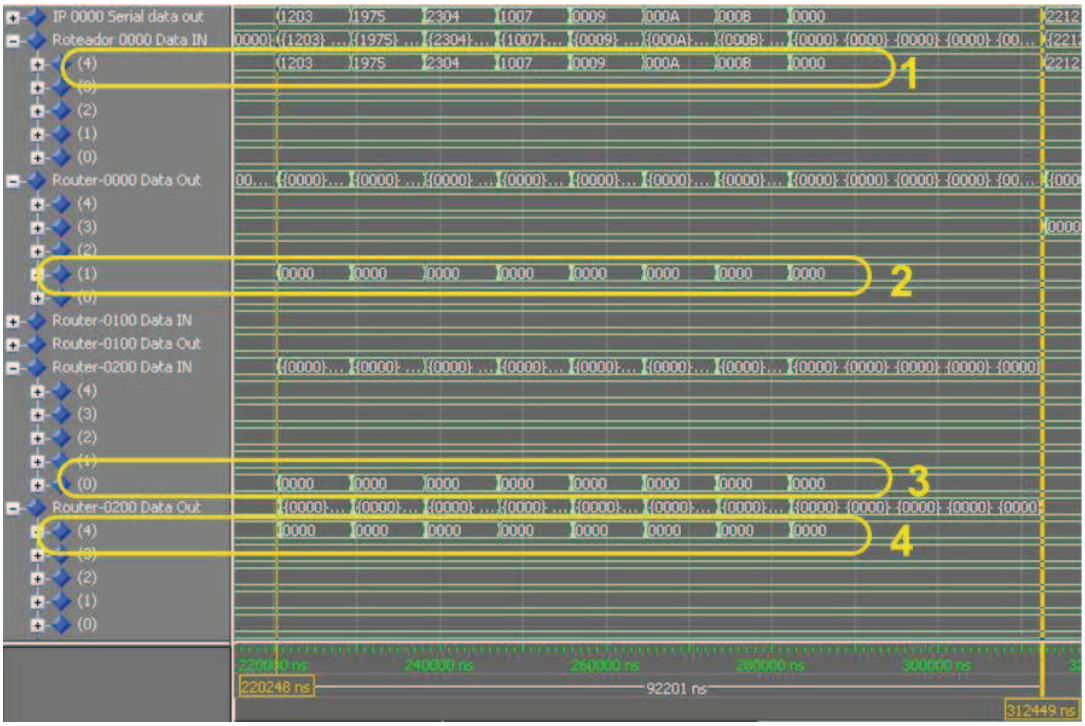

Figura 34 - Forma de onda mostrando uma simulação sem atraso de um processo de transmissão de pacotes do IP 00 ao IP 20. Os sinais mostrados para os pontos 2-4 não mostram o valor esperado como ocorre em 1, mas isto é devido a falhas de detalhamento da visualização e não a erros de simulação.

- 5 – O IP Serial (00) recebe a seqüência de oito dados (em hexadecimal: 2212, 1403, 1703, 1986, 000D, 000E, 000F e 0010) e cria um pacote para enviar cada um deles através da NoC. Para cada pacote que entra no Roteador 00, o algoritmo de roteamento é executado, a tabela de chaveamento é atualizada, realizando a ligação entre a porta de entrada 4 (Local) do Roteador 00 e a porta de saída 3 (Sul) do mesmo roteador.
- 6 – O Roteador 01 recebe na sua porta de saída 3 (Sul) os pacotes da porta Local do mesmo roteador.
- 7 – O Roteador 01 recebe os pacotes oriundos da porta 3 (Sul) do Roteador 00, pela sua porta 2 (Norte). O algoritmo de roteamento é executado e a tabela de chaveamento é atualizada, realizando a ligação entre a porta de entrada 2 (Norte) com a porta de saída 4 (Local) do mesmo roteador.
- 8 – O Roteador 01 recebe na porta de saída 4 (Local) os pacotes oriundos da porta de entrada 2 (Norte) do mesmo roteador. Nesse instante, gradativamente os pacotes chegam e são armazenados na memória do IP 01, neste caso um IP Testador. O IP Testador fica então aguardando o sinal de Start para iniciar o processo de geração de tráfego programado nele para injeção na NoC.

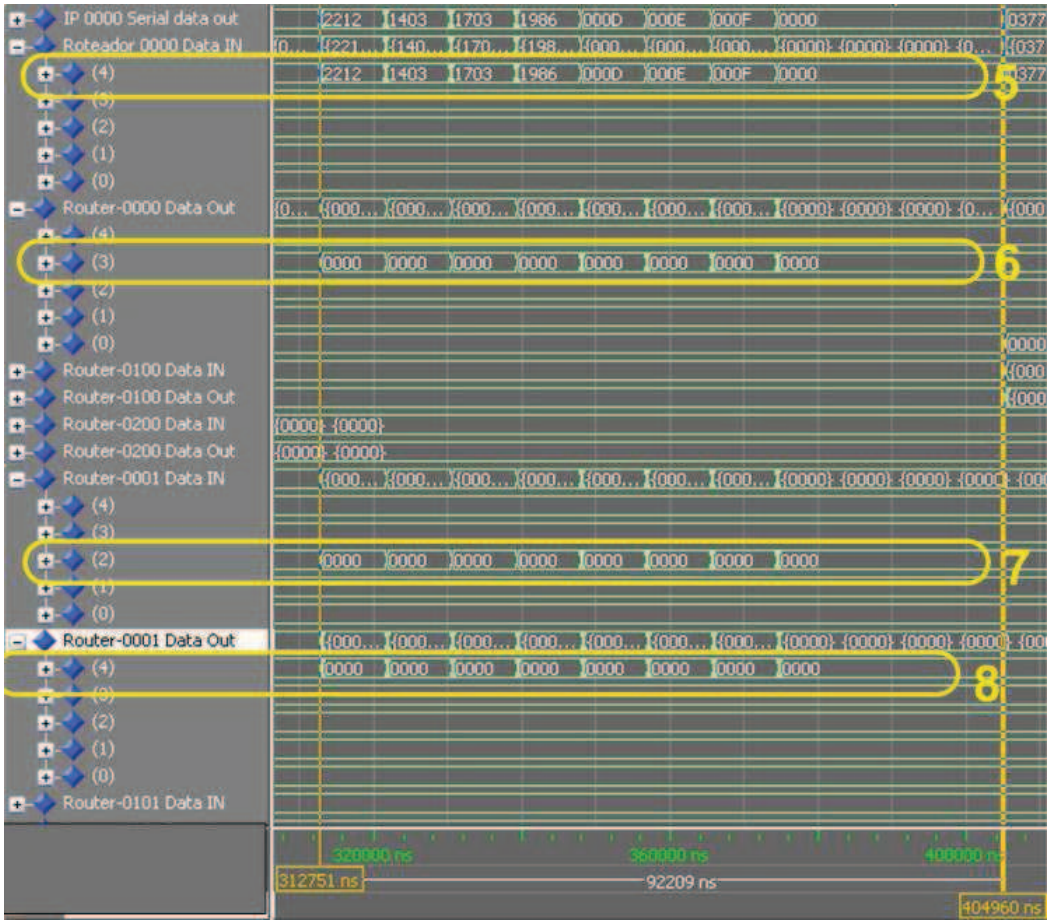

Figura 35 - Forma de onda mostrando uma simulação sem atraso de um processo de transmissão de pacotes do IP 00 ao IP 01. Os sinais mostrados para os pontos 6-8 não mostram o valor esperado como ocorre em 5, mas isto é devido a falhas de detalhamento da visualização e não a erros de simulação.

- 9 – O IP Serial (00) recebe a seqüência de oito dados (em hexadecimal: 0377, 1203, 1601, 2009, 2601, 2000, 0011 e 0000) e cria um pacote para enviar cada um deles através da NoC. Para cada pacote que entra no Roteador 00, o algoritmo de roteamento é executado, a tabela de chaveamento é atualizada, realizando a ligação entre a porta de entrada 4 (Local) do Roteador 00 e a porta de saída 0 (Leste) do mesmo roteador.
- 10 – O Roteador 10 recebe na sua porta de saída 0 (Leste) os pacotes da porta Local do mesmo roteador.
- 11 – O Roteador 10 recebe os pacotes oriundos da porta 0 (Leste) do Roteador 00, pela sua porta 1 (Oeste). O algoritmo de roteamento é executado e a tabela de chaveamento é atualizada, realizando a ligação entre a porta de entrada 1 (Oeste) com a porta de saída 4 (Local) do mesmo roteador.
- 12 – O Roteador 10 recebe na porta de saída 4 (Local) os pacotes oriundos da porta de entrada 1 (Oeste) do mesmo roteador. Nesse instante, gradativamente os pacotes chegam e são armazenados na memória do IP 10, neste caso um IP Testador. O IP Testador fica então aguardando o sinal de Start para iniciar o processo de geração de tráfego programado nele para injeção na NoC.

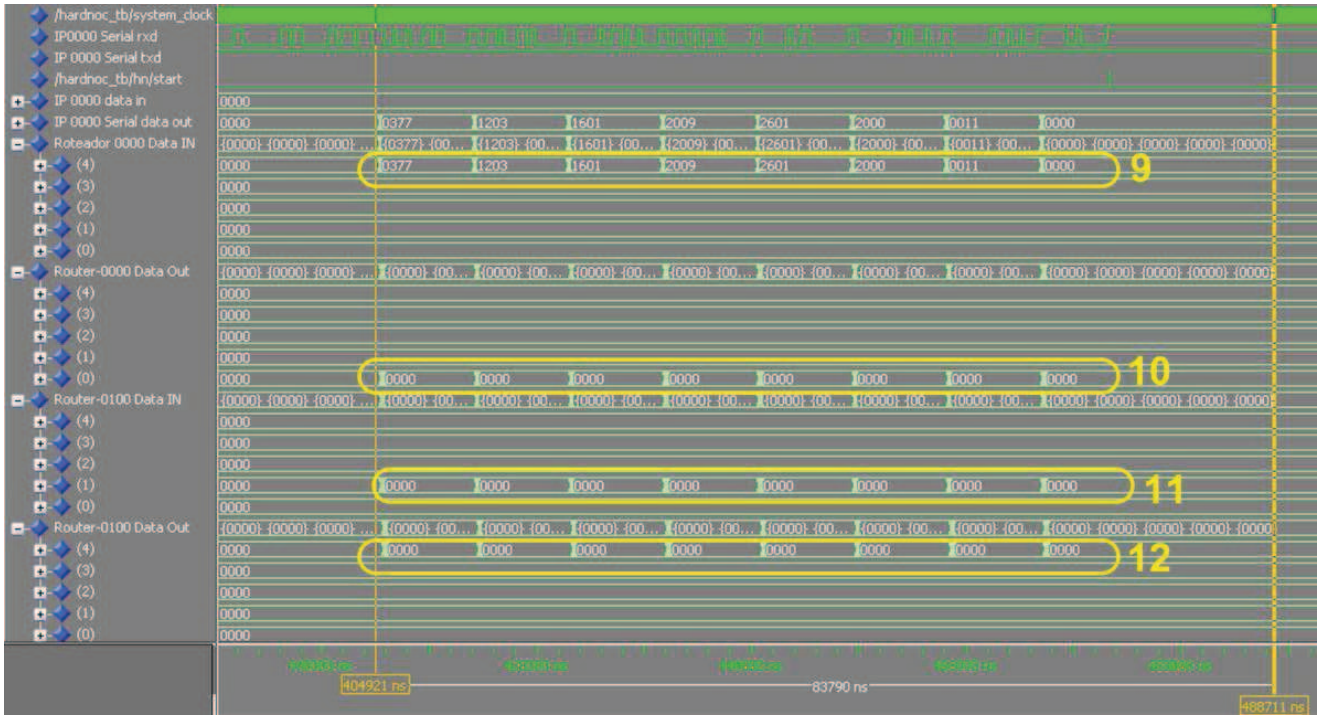

Figura 36 - Forma de onda mostrando uma simulação sem atraso de um processo de transmissão de pacotes do IP 00 ao IP 10. Os sinais mostrados para os pontos 10-12 não mostram o valor esperado como ocorre em 9, mas isto é devido a falhas de detalhamento da visualização e não a erros de simulação.

4.3.8 Experimento 5: Simulação com Atraso

Simulações como as descritas nos Experimentos 1 a 4, realizadas a partir da especificação em nível de registradores (em inglês, *Register Transfer Level*, ou RTL) do sistema são capazes de verificar a funcionalidade sem considerar a temporização do eventos no sistema de forma precisa. À medida que se avança pelas etapas de síntese e implementação, simulações mais precisas do comportamento real do circuito podem ser obtidas, tendo em vista que estas etapas incorporam informações sobre a tecnologia usada, posição de módulos e roteamento, permitindo estimar os atrasos associados à lógica e conexões.

Fazendo uso do simulador Modelsim adequadamente parametrizado, pode-se observar a diferença do comportamento de simulações com e sem atraso em um FPGA. Na Figura 37 mostra-se um comportamento síncrono controlado por um relógio. O sinal de relógio do sistema (system_clock) transiciona, habilitando que um dado passe para a saída do IP Serial, ao mesmo tempo em que entra na porta de entrada Local (4) do Roteador 00. Isso leva a reflexão que nesta simulação ideal o processo ocorre instantaneamente.

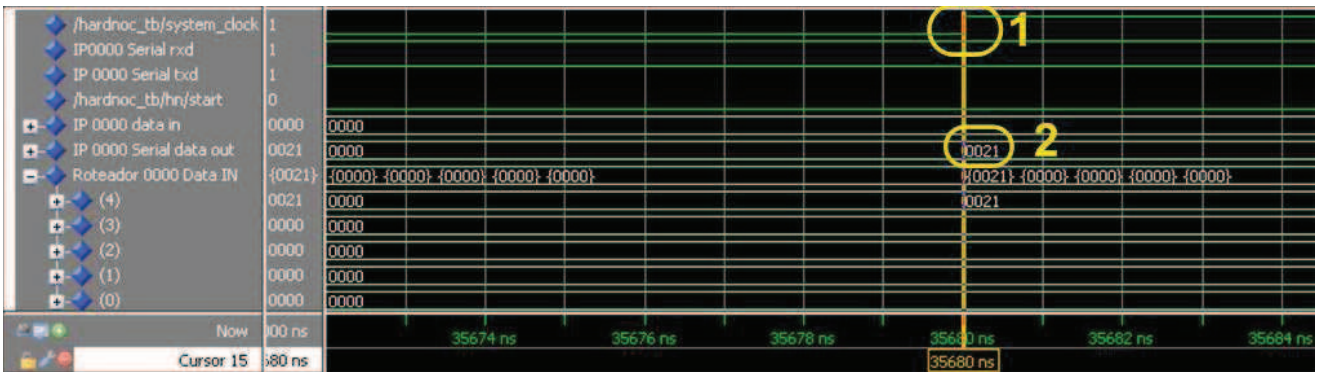

Figura 37 - Forma de onda representando uma simulação sem atraso.

Para realizar um processo de simulação com atrasos, deve-se proceder a mudanças na forma como é feita a compilação de arquivos e na preparação para gerar formas de onda. Ilustra-se este processo a seguir. Supondo que a simulação funcional do sistema já alcançou sucesso, procede-se à síntese do sistema HardNoC-TB, usando por exemplo a ferramenta XST do ambiente ISE da Xilinx, seguido de uma síntese física. Neste processo, parametriza-se a síntese para gerar um arquivo com os dados de temporização do sistema como um todo, associado ao componente FPGA específico (no caso, usa-se o FPGA XC2VP30-FF896-7). Este arquivo recebe o nome de HardNoC_timesim.sdf, e contém uma descrição VHDL no nível de portas lógicas do sistema completo, anotado com os atrasos de componentes e fios do FPGA escolhido. Os atrasos são sempre anotados em triplas de valores, correspondendo, para cada fio e componente a valores de atraso mínimo, típico e máximo. Na simulação escolhe-se um destes tipos de atraso para usar.

A **Figura 38** mostra um exemplo de script para realizar uma simulação da HardNoC-TB com atrasos utilizando o Modelsim. Apenas três arquivos necessitam ser compilados: o pacote básico com as definições internas da rede Hermes-TB; o arquivo gerado no processo de síntese, contendo toda a estrutura da HardNoC-TB e os atrasos associados; e o arquivo de testbench da HardNoC-TB, denominado HardNoC_TB. vhd.

A explicação dos passos de simulação são muito similares aos passos usados na simulação com atraso da Figura 18, alterando-se apenas o passo marcado com o indicador 4 em ambas as Figuras. Os passos iniciais (1-3) da Figura 38 são idênticos aos da Figura 18, sendo por isso omitidos na primeira. O passo 4 do script de simulação com atrasos tem a seguinte funcionalidade:

- 4 - Comando que realiza a chamada para simulação, que conta com algumas opções específicas em relação ao comando de simulação da simulação com atrasos: primeiro, a opção –t 1ps estabelece o limite de resolução de tempo explicitamente em 1ps; a opção –sdfmax estabelece que se deva usar valores máximos de atraso obtidos a partir da leitura do arquivo HardNoC_timesim.sdf.

```
vcom -work work -93 -explicit NOC/HermesTB_package.vhd
vcom -work work -93 -explicit HardNoC_timesim.vhd
vcom -work work -93 -explicit HardNoC_TB.vhd                    3

#simulação com atrazo
vsim -t 1ps -sdfmax "/HN=HardNoC_timesim.sdf" -lib work hardnoc_tb  4

do W_ROuters_IPs.do                                             5
run 500us
```

Figura 38 - Script para realizar simulação com atrasos.

O arquivo com extensão SDF contém as informações dos componentes e fios com os referentes atrasos, neste projeto especifico. SDF é uma sigla que representa um formato textual padrão de representação lógica de um circuitos com atrasos anotados. A sigla significa *Standard Delay Format*. No arquivo HardNoC_timesim.sdf está a descrição de toda estrutura de hardware, contendo os IPs e Roteadores, bem como o comportamento de atraso de fios e componentes eletrônicos do FPGA. Esse arquivo descreve um circuito de forma muito mais aproximada de um ambiente com as características físicas do circuito que se vai construir (no caso, configurando o FPGA). Contudo, seu uso destina-se a simulação. Referindo-se à forma de onda da Figura 39, nota-se que o sinal que respondia

instantaneamente à transição do sinal de relógio do sistema na Figura 37 possui em verdade um atraso de 2,408ns em relação àquele.

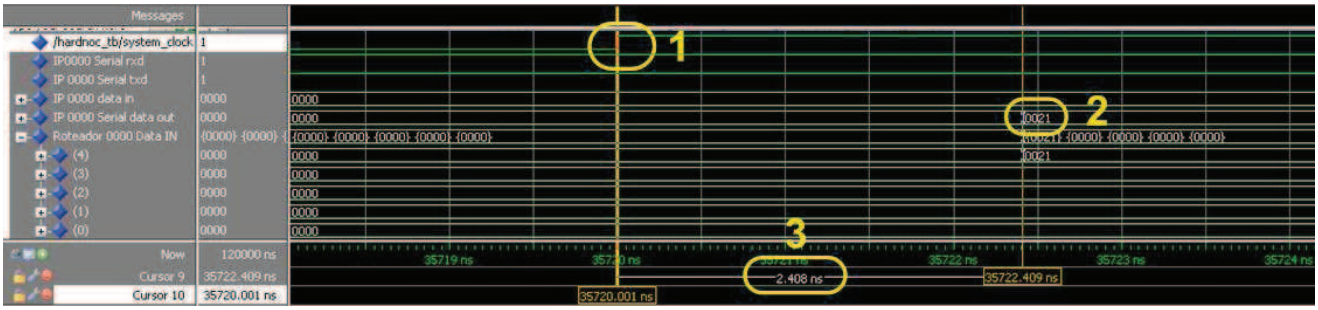

Figura 39 - Forma de onda em uma simulação com atraso, mostrando que a transição instantânea da Figura 37 na realidade demora 2,408 ns para ocorrer.

Outro exemplo, relacionado a pacotes da NoC Hermes-TB aparece ilustrado na Figura 40. No círculo 1 observa-se o tempo de 63000 ns em que o valor 0377 está na porta de saída Local (4) do Roteador 21. No círculo 2 observa-se a subida do sinal de relógio. Foi marcada de forma precisa a posição onde se observa o valor 0377 através do círculo 3 localizando o sexto *flit* de um pacote contido na terceira posição da mensagem enviada do IP Serial (IP 00) ao IP Testador 21.

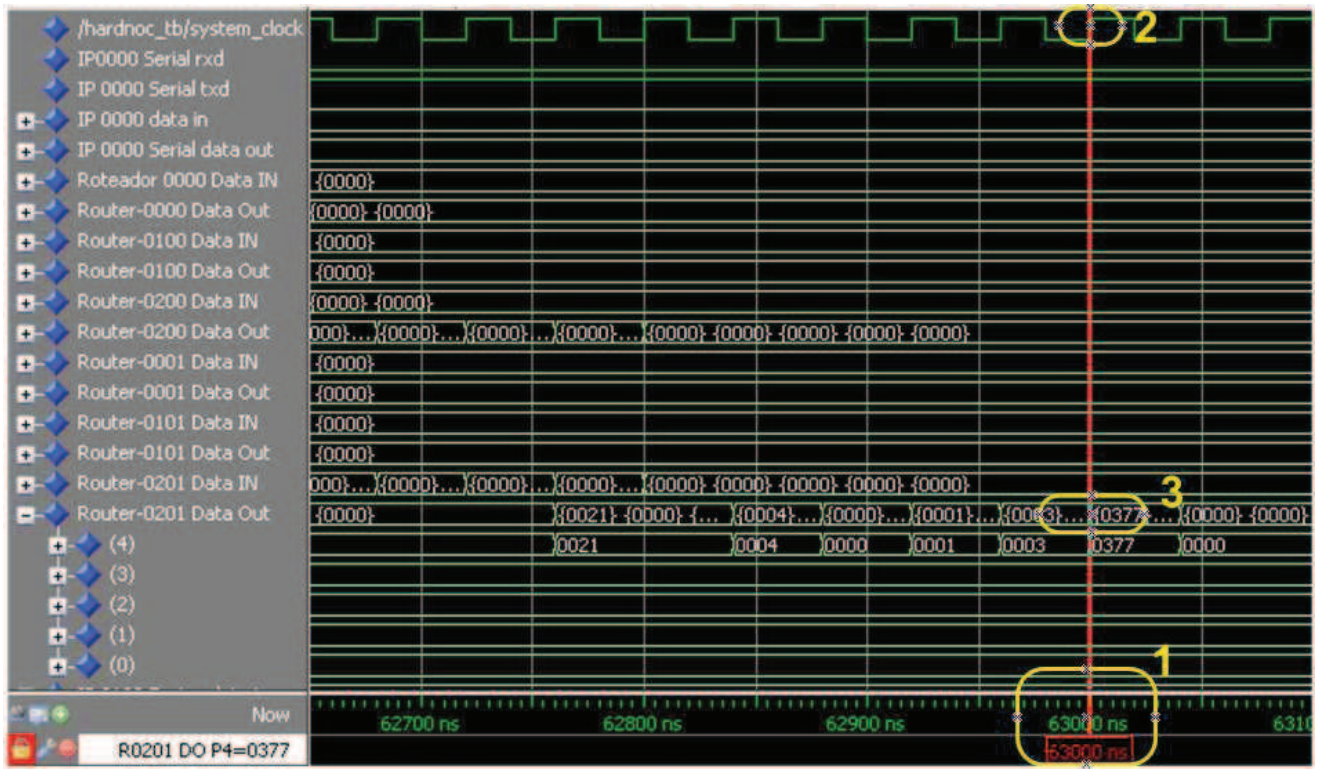

Figura 40 - Forma de onda em uma simulação sem atraso identificando o dado 0377 hexadecimal, saindo do Roteador 21 pela Porta Local (4).

A simulação com atraso demonstra um atraso acumulativo de 46,224ns na Figura 41 em relação à Figura 40. Como a frequência de relógio é de 25 MHz e o período, por consequência é de 40ns, esta situação indica um potencial problema de temporização do sistema, onde atrasos combinacionais que ultrapassam o valor do período de relógio estão presentes. O circuito provavelmente jamais funcionaria nesta freqüência devido a este problema, impossível de detectar em simulações sem atraso. Uma solução para este problema é reduzir a frequência de operação para 20MHz (50 ns de período).

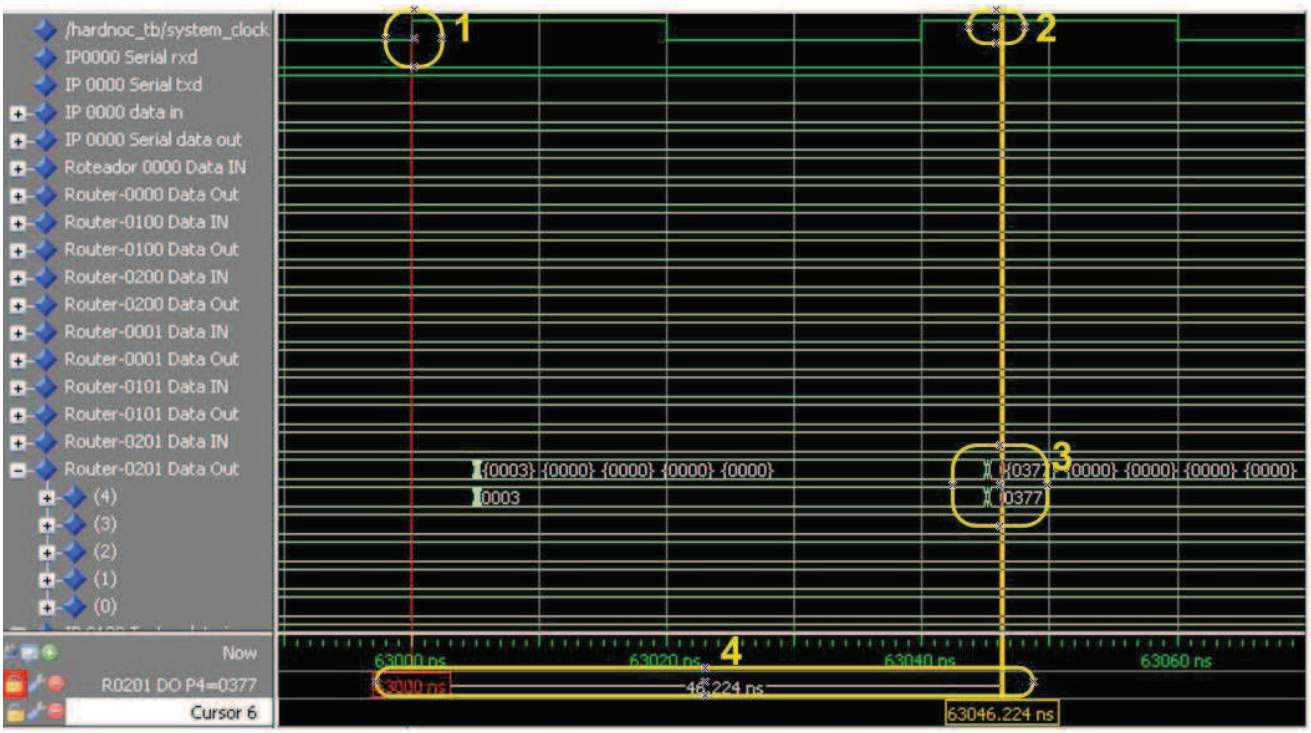

Figura 41 - Forma de onda em uma simulação expondo atraso do dado 0377 de 46.224 ns, superior ao período do relógio do sistema (40ns).

5 PROTOTIPAÇÃO DA REDE HERMES-TB

FPGAs foram introduzidos em meados dos anos 80, como uma nova tecnologia para implementação de lógica digital. Estes dispositivos representam uma evolução natural a partir dos seus antecessores, os PLAs (do inglês, *Programmable Logic Arrays*), projetados para construir lógica de dois níveis [NAS07]. FPGAs, que podem hoje comportar circuitos de até algumas dezenas de milhões de portas, permitem a implementação de grandes quantidades de lógica digital com estruturas combinacionais e sequenciais. FPGAs abriram caminho para uma revolução no domínio das tecnologias de lógica programável, com aplicações nos mais variados campos da eletrônica digital, dando suporte à construção de sistemas completos (ou quase) em seu interior.

Uma característica importante de FPGAs é sua capacidade de serem configurados e reconfigurados no campo. Desta forma, a funcionalidade do dispositivo não é definida durante sua fabricação e sim pelo projetista da aplicação final, através do preenchimento de uma memória de controle. O que o fabricante fornece é uma matriz de blocos lógicos básicos programáveis, capazes de armazenar bits de informação e realizar funções lógicas elementares, cercados por um conjunto de rotas e conexões reconfiguráveis que permitem a interligação destes blocos para implementação das funções desejadas [NAS07].

Uma aplicação pode ser especificada de forma gráfica com esquemáticos e diagramas de estados e diagramas de blocos. Contudo, para permitir a especificação de comportamentos complexos são utilizadas mais comumente descrições textuais em linguagens de descrição de hardware (do inglês, *Hardware Description Languages* ou HDLs). Exemplos destas linguagens são VHDL e Verilog. Atualmente, a utilização a linguagem SystemC vem ganhado adeptos no meio acadêmico e industrial, devido à capacidade potencial de integrar em uma mesma especificação descrições de componentes de software e de hardware de uma aplicação em um modelo único executável, bem como a habilitação de descrições de nível de abstração mais alto, útil na modelagem de sistemas altamente complexos [NAS07].

O processo de construção de uma aplicação em FPGA compreende quatro etapas importantes, e cada uma destas contém diversas etapas menores. As quatro principais são: síntese lógica, síntese física, geração do arquivo de configuração e configuração do dispositivo. Estas etapas são mais bem detalhadas a seguir.

A primeira etapa, denominada síntese lógica compreende gerar uma descrição de portas lógicas que corresponde em termos de funcionalidade ao comportamento implementado pelos arquivos em linguagem de descrição de hardware, que são a entrada desta etapa. Esta descrição de portas lógicas é denominada de *netlist*. A *netlist* descreve a interconexão de componentes digitais primitivos (como portas lógicas, *flip-flops* e registradores) obtidos de uma biblioteca abstrata de portas e macro células. Durante esta etapa são aplicados métodos de otimização de circuitos digitais para produzir uma implementação eficiente da aplicação. Um exemplo de formato de *netlist* de circuitos é o formato EDIF (do inglês *Electronic Data Interchange Format*).

A segunda etapa, denominada síntese física, gera uma rede de circuitos digitais específicos, que é dependente de tecnologia. Nessa etapa ocorre o mapeamento tecnológico, que tem por função mapear os componentes da *netlist* em blocos básicos do FPGA. A síntese física utiliza como menor unidade lógica configurável os chamados *slices* do FPGA. O *slice* é um módulo configurável,

formado por duas LUTs (do inglês, *Look Up Tables*) com uma saída e quatro entradas (no caso dos FPGAs da família VirtexII-Pro usados neste trabalho). LUTs pode ser configuradas como tabelas-verdade de até 4 variáveis Booleanas, ou como registradores de deslocamento ou memórias de porta simples ou dupla com capacidade de armazenar 16 bits. Além de LUTs, os *slices* possuem recursos como dois flip-flops tipo D, vários multiplexadores e demultiplexadores configuráveis, e lógica de propagação rápida de vai-um para facilitar implementação de circuitos aritméticos básicos. A LUT é o elemento básico em FPGAs da Xilinx, e tais elementos são modelados pela Xilinx na biblioteca UNISIM.

Após o mapeamento tecnológico, a aplicação passa a ser descrita como uma *netlist* de blocos lógicos do FPGA com as especificações de suas configurações internas. A ferramenta de posicionamento então determina a posição que cada bloco deve ocupar no FPGA, de forma a facilitar o processo de roteamento das conexões e minimizar o atraso associado. Isto permite a obtenção de freqüências de sinal de relógio adequadas para a operação do circuito. Após a execução do posicionador, a etapa de roteamento implementa todas as conexões entre blocos e pinos de entrada e saída utilizando os recursos de roteamento reconfiguráveis do FPGA.

A terceira etapa, geração do arquivo de configuração, é realizada após o roteamento. É gerada uma saída com uma descrição de todas as configurações necessárias do FPGA, em um formato binário específico do fabricante do componente. Esta configuração constitui-se em um *bitstream* para configuração do FPGA [NAS07].

A quarta etapa denominada configuração do dispositivo é o processo no qual o arquivo já compilado é enviado ao dispositivo FPGA para receber os sinais elétrico.

O processo de prototipação da rede Hermes-TB baseou-se em alguns trabalhos anteriores realizados no GAPH. O modelo VHDL da HardNoC-TB é composto por múltiplos níveis hierárquicos de entidades. Aqui a discussão se limita aos três primeiros níveis desta hierarquia de projeto. A primeira representa o nível mais alto de abstração, e corresponde à interface externa do sistema HardNoC-TB, composta por sinais de entrada e saída. Estes podem ser observados na Figura 42, compreendendo os sinais **`rxd`**, **`txd`**, **`system_clock`** e **`system_reset`**.

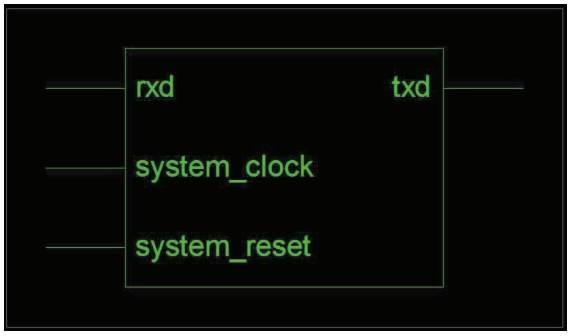

Figura 42 - A interface externa do sistema HardNoC-TB.

No segundo nível de hierarquia, mostrado na Figura 43, têm-se o diagrama de blocos principal do sistema. Este diagrama mostra os blocos fundamentais do sistema interligados por conexões na forma de barramentos e sinais individuais de controle. Estes são a NoC (vista como um bloco monolítico) o IP Serial (IP0000) e os IPs Testadores (os demais).

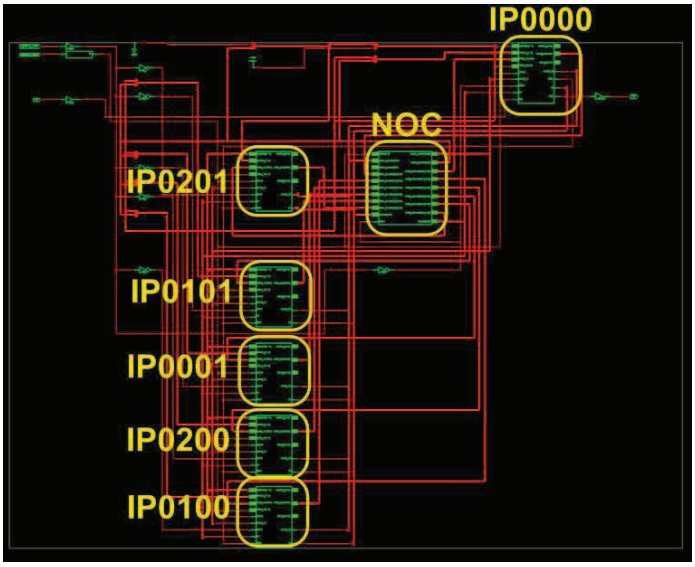

Figura 43 - Diagrama de blocos no primeiro nível da hierarquia da HardNoC-TB, contendo a NoC como um bloco monolítico e os IPs Serial (IP0000) e Testadores (demais).

A entidade VHDL HardNoC-TB consiste em uma descrição estruturada fundamentada no diagrama de blocos ilustrado na Figura 42. O detalhamento da entidade consiste em uma arquitetura VHDL descrita graficamente pelo diagrama da Figura 43. Esse diagrama apresenta apenas as instâncias das entidades IP Serial e 5 de IPs Testadores. A Figura 44 ilustra os detalhes da arquitetura da NOC Hermes-TB, que contém todo os roteadores desta. Estes esquemas anunciam ao leitor a complexidade do sistema em prototipação, seguindo desde a descrição em VHDL, até os detalhes de cada bloco componente de hardware desta.

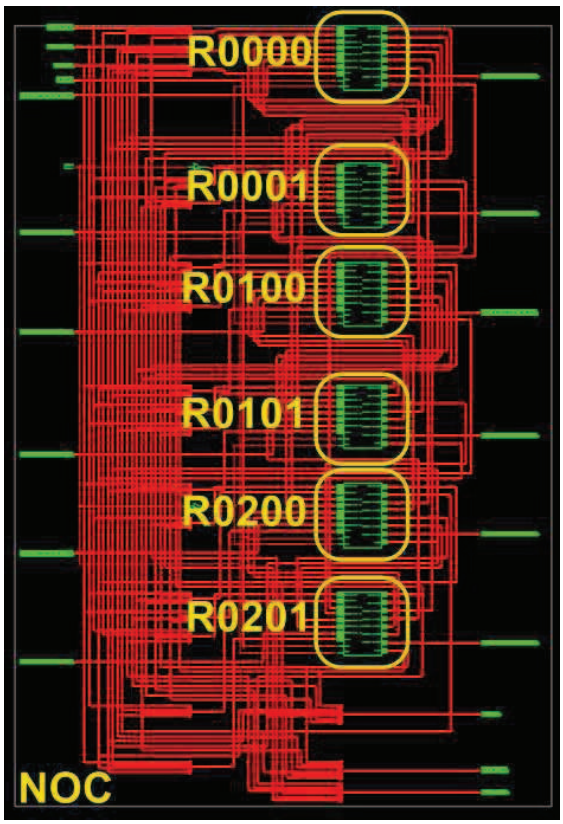

Figura 44 - Diagrama de esquemáticos da estrutura interna da NoC com seus respectivos roteadores.

Mesmo observada sem maiores detalhes, a Figura 44 revela a interligação entre os roteadores,

da NoC. Note-se que a Figura 44 detalha o módulo denominado NoC (Figura 43).

O sistema HardNoC-TB completo descrito por estes diagramas foi prototipado sobre a plataforma XUP-V2PRO da empresa Digilent, Inc [DIG09], cuja fotografia aparece na Figura 45. A interação com o computador hospedeiro se dá usando um cabo serial conectado à interface serial RS-232 da placa (mostrada no canto inferior direito da Figura 45) e ao conector equivalente do lado do hospedeiro.

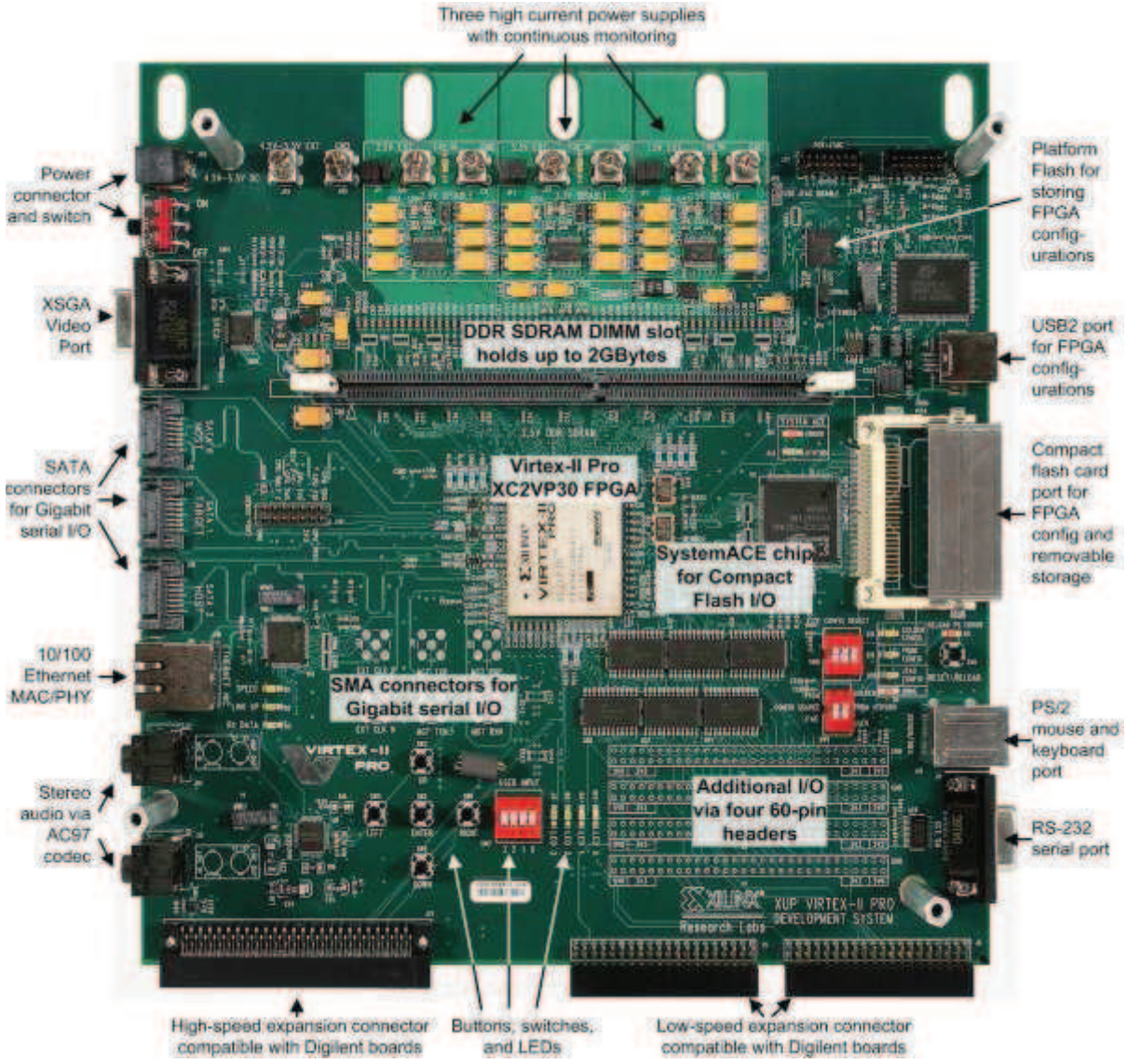

Figura 45 - Plataforma de prototipação usada nos experimentos, XUP-V2PRO da empresa Digilent [DIG09], contendo um FPGA Xilinx XC2VP30 da família VirtexII-Pro.

5.1 Análise dos Sinais da Hermes-TB no Protótipo

Através do uso da plataforma XUP-V2PRO consegui-se prototipar o ambiente HardNoC-TB e explorar seu uso mediante emprego da ferramenta ChipScope. Pôde-se verificar que a comunicação do IP Serial para programar Testadores e verificar seu conteúdo opera corretamente. Entretanto, o disparo da execução dos processos de geração de tráfego e coleta de estatísticas na HardNoC-TB a partir da geração do comando Start não operou corretamente. No início deste trabalho o fato da plataforma original (HardNoC) na qual se baseou a HardNoC-TB ter operado de maneira correta e completa foi o fator decisivo na sua escolha. Contudo, o processo de repetição da prototipação da

HardNoC revelou que esta apresenta o mesmo problema da HardNoC-TB, resultante provavelmente da perda de informações sobre o processo de prototipação do projeto. A documentação de base original da HardNoC foi então revisada para identificar os erros nesta. Vários problemas foram encontrados e corrigidos. Contudo, atualmente o ambiente HardNoC ainda está em processo de depuração, e este processo atingiu o mesmo ponto em que se congelou a versão atual da HardNoC-TB. Assim, acredita-se que este problema seja independente da NoC utilizada.

Com a finalidade de se analisar os sinais internos no FPGA utilizou-se o software Chipscope Pro 9.2i da Xilinx. Chipscope possibilita a monitoração do sistema prototipado, em tempo real, pela inserção de módulos de captura de dados do sistema em tempo real e módulos de transmissão dos mesmos através do cabo de programação da plataforma.

O processo de criação da interface Chipscope com finalidade de leitura em tempo real dos dados foi implementado na ferramenta ISE 10.1, transferindo um arquivo binário, usando um cabo de programação USB. A Figura 46 mostra os pontos da simulação que puderam ser observados operando corretamente usando o Chipscope.

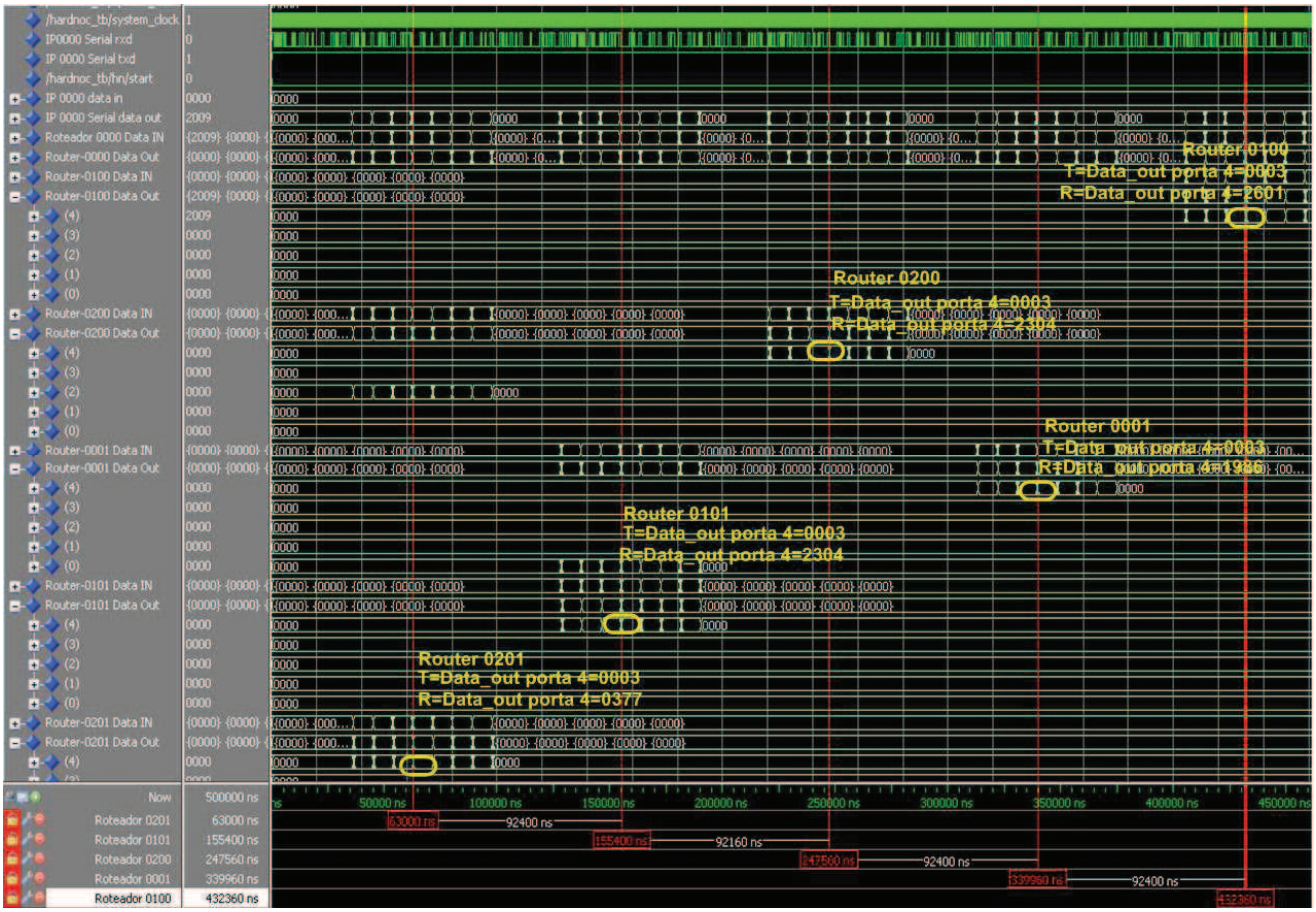

Figura 46 - Pontos da simulação a serem observados através do Chipscope na HardNoC-TB prototipada.

5.2 Falhas Encontradas na HardNoc-TB

Observou-se durante o processo de depuração da HardNoC-TB que a operação da rede toro bidirecional segue o comportamento esperado. Contudo, ao realizar os testes utilizando IPs Serial e IPs Testadores ocorrem alguns problemas, como descritos a seguir:

- Primeiramente, verificou-se que sem o DCM não é possível alcançar um funcionamento correto da HardNoC-TB. O DCM é um módulo complexo, que conforme o *datasheet* do FPGA Xilinx XC2VP30 com speed grade -7 necessita de pelo menos 5 microssegundos (ou seja, 2500 ciclos de relógio) depois do sinal de inicialização do sistema (**`reset`**) ser ativado, para que o DCM ative o seu sinal de saída LOCKED, indicando que o sistema está apto a operar. Só depois disto os sinais do sistema podem ser observados. Pode-se considerar como irrelevante qualquer sinal antes deste tempo. Nas simulações, esse sinal não sofre esse atraso, o LOCKED é ativado imediatamente depois do reset. Nos testes, o sinal LOCKED estava com valor desconhecido ou inválido (X). Esse sinal é utilizado em todo sistema e na ausência do mesmo é impossível interpretar corretamente os resultados de simulação. A solução para este problema foi a re-execução do processo de compilação das bibliotecas SIMPRIM e UNISIM. Observou-se que estas duas bibliotecas não podem coexistir ao mesmo tempo, pois isto causa uma geração incorreta do arquivo SDF da simulação com atraso e depois um arquivo de configuração binário incorreto para a prototipação. Personalizando o script de síntese, e reinstalando as bibliotecas a operação do sinal LOCKED do DCM foi corrigida.
- Resolvidos os problemas referentes às bibliotecas, e depois de migrar o ambiente para a nova versão do ambiente ISE (da 9.2 para a 10.1), a simulação revelou que o sinal LOCKED novamente apresentou problemas. O problema se configurou na forma de transitórios antes de o DCM começar a operar em si. Com isto, quando ocorre este transitório o sinal de reset, gerado a partir deste também apresenta transitório. Isto faz com que a HardNoC comece a operar antes do momento correto, o que gera problemas sérios na simulação conforme ilustra a Figura 47. Não foi possível ainda descobrir o motivo da formação deste transitório, mas reduzindo a freqüência de operação de 50MHz para 25 MHz, os transitórios desaparecem. A ferramenta ISE, no entanto, informa que a freqüência de operação máxima é de 140 MHz.
- Resolvendo o problema dos transitórios, foi detectado ainda outro problema. Esse ocorre ao se fazer a simulação com atraso. O tempo de recebimento da informação não está sendo o suficiente para que a informação possa ser enviada por completo. Com isto, ao longo da operação ocorrem falhas de temporização. O acúmulo destas falhas faz com que a HardNoC "trave" a partir de certo ponto da operação.

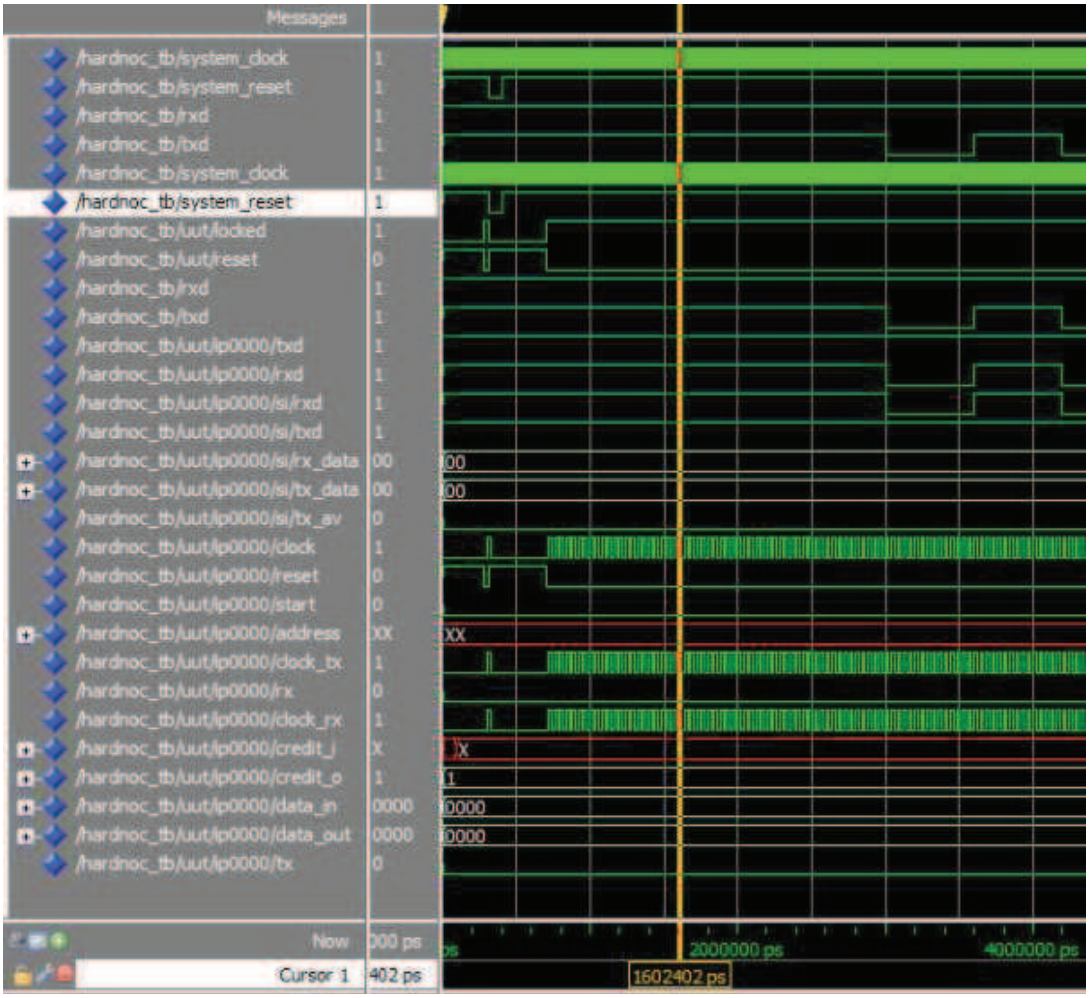

Figura 47 - Pontos de observação obtidos na simulação demonstrando o início da possível falha.

5.3 Distribuição do espaço utilizado no FPGA

A ferramenta ISE de desenvolvimento de projeto para FPGAs da Xilinx possibilita a restrições de posicionamento através do uso de *macros*. A restrição de posicionamento é importante quando se busca a hierarquia dos sinais, e para reduzir o atraso do circuito. Na Figura **48** pode-se observar o arquivo UCF e a maneira de se delimitar a área a ser ocupada por cada elemento na Hermes-TB.

```
1  ################################################
2  # VirtexII Pro VP30 FF896                      #
3  ################################################
4  #PACE: Start of Constraints generated by PACE
5  #PACE: Start of PACE I/O Pin Assignments
6  #net system_clock tnm_net = system_clock;
7  #timespec ts_periodo = period system_clock 20 ns;
8  #PACE: Start of PACE Prohibit Constraints
9  #PACE: End of Constraints generated by PACE
10 # Start of Constraints extracted by Floorplanner from the Design
11 INST "IP0201" AREA_GROUP = "AG_IP0201" ;
12 AREA_GROUP "AG_IP0201" RANGE = SLICE_X64Y159:SLICE_X91Y118 ; 1
13 AREA_GROUP "AG_IP0201" RANGE = RAMB16_X5Y15:RAMB16_X5Y19, RAMB16_X6Y15:RAMB16_X6Y18, RAMB16_X7Y15:RAMB16_X7Y19 ;
14 NET "txd" LOC = "AE7" ;
15 NET "system_reset" LOC = "AG5" ;
16 NET "system_clock" LOC = "AJ15" ;
17 NET "rxd" LOC = "AJ8" ;
18 INST "IP0200" AREA_GROUP = "AG_IP0200" ;
19 AREA_GROUP "AG_IP0200" RANGE = SLICE_X64Y41:SLICE_X91Y0 ; 2
20 AREA_GROUP "AG_IP0200" RANGE = RAMB16_X5Y0:RAMB16_X5Y4, RAMB16_X6Y1:RAMB16_X6Y4, RAMB16_X7Y0:RAMB16_X7Y4 ;
21 INST "IP0101" AREA_GROUP = "AG_IP0101" ;
22 AREA_GROUP "AG_IP0101" RANGE = SLICE_X28Y159:SLICE_X63Y118 ; 3
23 AREA_GROUP "AG_IP0101" RANGE = RAMB16_X2Y15:RAMB16_X2Y19, RAMB16_X3Y15:RAMB16_X3Y18, RAMB16_X4Y15:RAMB16_X4Y18 ;
24 INST "IP0100" AREA_GROUP = "AG_IP0100" ;
25 AREA_GROUP "AG_IP0100" RANGE = SLICE_X28Y41:SLICE_X63Y0 ; 4
26 AREA_GROUP "AG_IP0100" RANGE = RAMB16_X2Y0:RAMB16_X2Y4, RAMB16_X3Y1:RAMB16_X3Y4, RAMB16_X4Y1:RAMB16_X4Y4 ;
27 INST "IP0001" AREA_GROUP = "AG_IP0001" ;
28 AREA_GROUP "AG_IP0001" RANGE = SLICE_X0Y159:SLICE_X27Y118 ; 5
29 AREA_GROUP "AG_IP0001" RANGE = RAMB16_X0Y15:RAMB16_X0Y19, RAMB16_X1Y15:RAMB16_X1Y18 ;
30 INST "IP0000" AREA_GROUP = "AG_IP0000" ;
31 AREA_GROUP "AG_IP0000" RANGE = SLICE_X0Y41:SLICE_X27Y0 ; 6
```

Figura 48 – Arquivo UCF, definindo a área a ser ocupada conforme a Hierarquia.

Observando as referencias da Figura 48 tem-se a relação de áreas indicadas conforme o eixo x e y do FPGA na Figura 49. Esta Figura ilustra a estrutura do arquivo UCF.

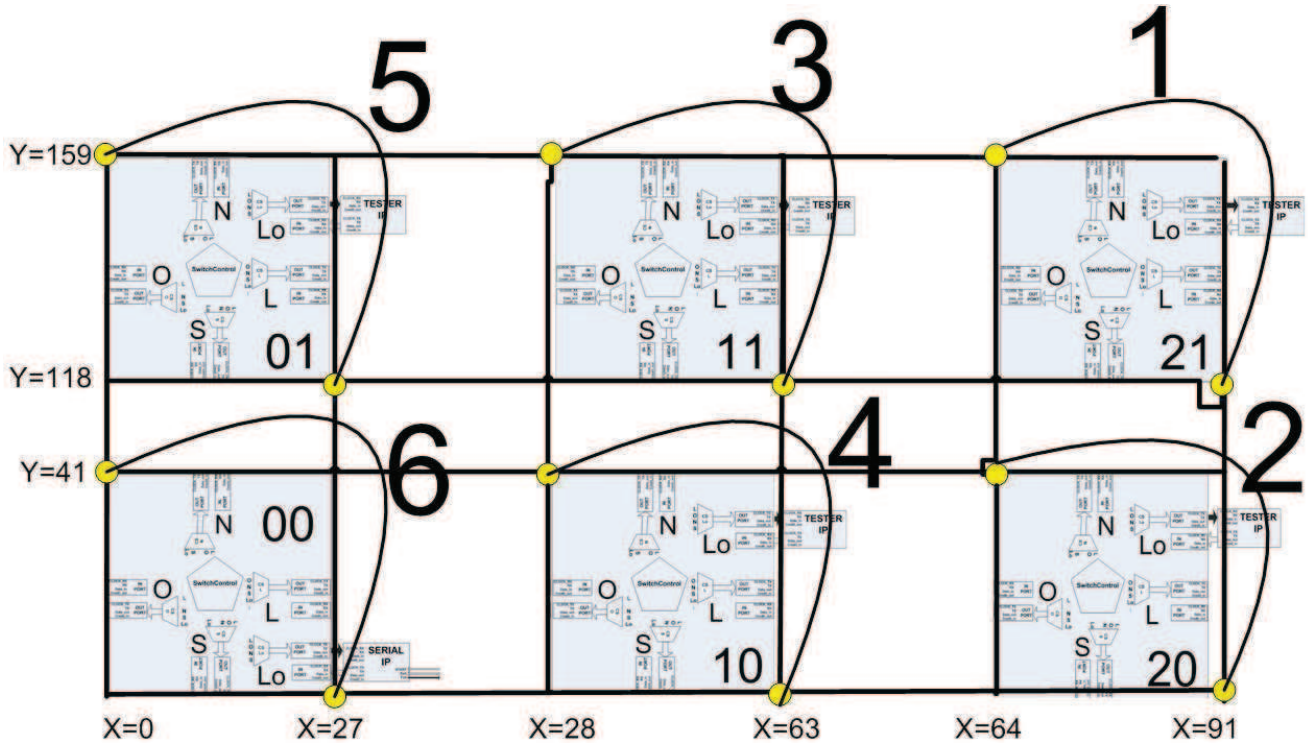

Figura 49 – Alocação de espaço conforme a proposta visual.

A Figura 50 representa a alocação real conforme gerada pela ferramenta ISE, da disposição espacial da HardNoc-TB prototipada em FPGA, conforme orientações do arquivo UCF.

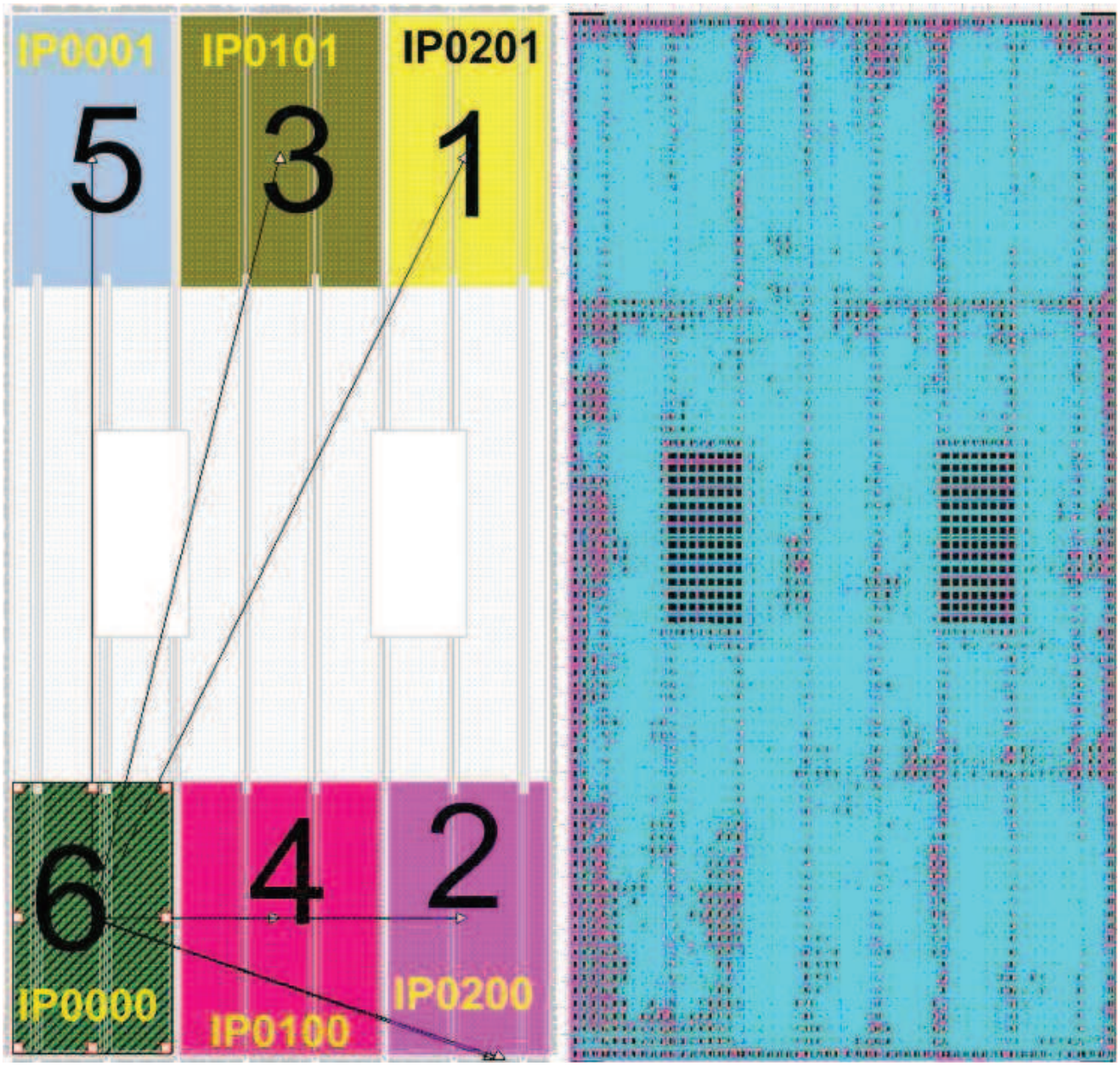

Figura 50 – Área ocupada no FPGA conforme informações do na UCF e a área ocupada no FPGA.

5.4 Ocupação de Área

A Tabela 1 ilustra as diferenças de área entre os sistemas HardNoC e HardNoC-TB.

Tabela 1 – Comparação de área entre a HardNoC e a HardNoC-TB.

	HardNoC				HardNoC-TB		
Utilização Lógica	Usado	Disponível	Utilizado		Usado	Disponível	Utilizado
Número de flip-flops de slices	7803	27392	28%		6125	27392	22%
Número de LUTs de 4 entradas	20095	27392	73%		15659	27392	57%
Distribuição Lógica	Usado	Disponível	Utilizado		Usado	Disponível	Utilizado
Número de slices ocupados	12124	13696	89%		9547	13696	70%
Número total de LUTs de 4 entradas	20095	27392	73%		17156	27392	63%

No processo de prototipação uma vantagem descoberta em relação à rede Hermes-VC foi que a rede Hermes-TB ocupa menos área. Como a única diferença entre ambos é a NoC utilizada, conclui-

se que a Hermes-TB ocupa menos área que a Hermes-VC. Mesmo oferecendo menor latência e maior vazão, a rede toro ocupa menos espaço mesmo que os roteadores sejam todos de 5 portas. Um fator que explica essa redução é o fato dos canais virtuais não existirem na rede toro, reduzindo sobremaneira a área do roteador.

6 CONCLUSÕES E TRABALHOS FUTUROS

Esta dissertação apresentou o processo de prototipação da rede Hermes-TB através do desenvolvimento da plataforma HardNoC-TB, uma evolução da HardNoC original que utiliza a mesma estrutura com topologia malha 2D. A implementação da HardNoC-TB, alvo deste trabalho, foi validada através de simulações com e sem atraso, e parcialmente em um protótipo.

A HardNoC-TB possui vantagens em relação a HardNoC original, no que tange área ocupada obtendo um ganho de cerca de 8%. O ideal é o desenvolvimento da HardNoc-TB totalmente funcional na prototipação, com fluxos oriundos do hospedeiro externo através do IP Serial e a NoC entregando aos IPs Testadores as informações, seguido do processo de geração de tráfego e troca de mensagens resultantes entre os IPs Testadores. Até agora, o processo de programação dos IPs testadores com padrões de tráfego através da NoC foi plenamente validado. A operação após gerar o sinal **`Start`**, é deixada como trabalho futuro.

Dentre outros trabalhos futuros possíveis enumeram-se três como sugestão:

- Prototipar a HardNoC-TB usando um a dimensão maior, como por exemplo uma NoC 5X5 ou 6X6, visando realizar experimentos sob condições de uso mais típicas de NoCs, onde o número de núcleos conectados pela NoC é significativo. Claro que para tanto é necessário trocar a plataforma de prototipação e o FPGA utilizado.
- Validar o projeto HardNoC-TB através do emprego de técnicas de verificação de projeto como análise de cobertura de código e asserções, visando aumentar a robustez do projeto.
- Adaptar o ambiente MPSoC-H, proposto originalmente em [CAR05] para operar com a rede Hermes-TB no lugar da rede Hermes e comparar a execução neste ambiente mais realista do ponto de vista de aplicações. Este ambiente é similar ao ambiente HardNoC, mas substitui os IPs Testadores da HardNoC por processadores programáveis parcialmente compatíveis com a arquitetura MIPS-I, denominados MR4. Instâncias prototipadas do ambiente MPSoC-H estão disponíveis no GAPH.

REFERÊNCIAS BIBLIOGRÁFICAS

[ACC03] Accellera. «Open Verification Library». Users Reference Manual, Proposed Standard, 2003, 45 p.

[BAR05] Bartic, T.; Mignolet, J.-Y.; Nollet, V.; Marescaux, T.; Verkest, D.; Vernalde, S.; Lauwereins, R. «Topology adaptive network-on-chip design and implementation». In: IEE Proceedings on Computer and Digital Tecniques, vol. 152-4, Jul. 2005, pp. 467-472.

[BAS05] Bastos, E. «MERCURY : Uma rede Intra-chip com Topologia Toro 2D e roteamento adaptativo». Dissertação de Mestrado, 2005, Porto Alegre, PPGCC/PUCRS, 148p.

[BEN01] Benini, L.; De Micheli, G. «Powering Networks on Chip». In: International Symposium on System Synthesis (ISSS´01), 2001, pp. 33-38.

[BEN02] Benini, L.; De Micheli, G. «Network on Chips: A New SoC Paradigm». IEEE Computer, vol. 35-1, Jan. 2002, pp. 70-78.

[CAR07] Carara, E. «Uma Exploração Arquitetural de Redes Intrachip com Topologia Malha e Modo de Chaveamento Wormhole». Dissertação de Mestrado, PPGCC-FACIN-PUCRS, Porto Alegre, Dec. 2007, 65 p.

[CAR05] Carara, E.; Moraes, F. «MPSoC-H - Implementação e Avaliação de Sistema MPSoC Utilizando a Rede Hermes». Relatório Técnico TR 051, PPGCC-FACIN-PUCRS, Porto Alegre, Dec. 2005, 73 p.

[CHI04] Chi, H-C.; Chen, J-H. «Design And Implementation of A Routing Switch for On-Chip Interconnection Networks». In: IEEE Asia-Pacific Conference on Advanced System Integrated Circuits 2004, pp. 392- 395.

[COU94] Coulouris, G.; Dollimore, J.; Kindberg, T. «Distributed Systems: Concepts and Design». Addison Wesley, 1994, 636 p.

[DAL86] Dally, W.; Seitz, C. «The torus routing chip». Journal of Parallel and Distributed Computing, vol. 1-3, 1986, pp. 187-196.

[DAL01] Dally, W.; Towles, B. «Route Packets, Not Wires: On-Chip Interconnection Networks». In: Design Automation Conference (DAC´01), Jun. 2001, pp. 684-689.

[DIG09] Digilent, Inc. «Virtex-II Pro Development System: Curriculum on a Chip». Capturado em: http://www.digilentinc.com/Data/Products/XUPV2P/XUPV2P-top.jpg, 7 de Janeiro de 2009.

[DUA97] Duato, J. et al. «Interconnection Networks». Los Alamitos, California : IEEE Computer Society Press, 1997, 515 p.

[FOS04] Foster, H.; Krolnik, A.; Lacey, D. «Assertion-Based Design». Kluwer Academic Publishers, 2004.

[GAP08] GAPH - Grupo de Apoio ao Projeto de Hardware. «Atlas - An Environment for NoC Generation and Evaluation». Capturado em http://www.inf.pucrs. br/~gaph/AtlasHtml/AtlasIndex_us.html, 17 de Dezembro de 2008.

[GAP08a] GAPH - Grupo de Apoio ao Projeto de Hardware. «Serial Software». Disponível em http://www.inf.pucrs. br/~gaph/homepage/download/software/SerialSoftware.zip.

[GLA94] Glass, C.; Ni, L. «The Turn Model for Adaptive Routing». Journal of the Association for Computing Machinery, vol. 41-5, 1994, pp. 874-902.

[HAR04] Tayan, O.; Harle, D. «A Manhattan Street Network Implementation for Network on Chip». In: IEEE International Conference on Information and Communication Technologies: From Theory To Applications, Apr. 2004, pp. 683-684.

[HOL03] Hölzenspies, P.; Schepers, E.; Bach, W.; Jonker, M.; Sikkes, B.; Smit, G.; Havinga, P. «A Communication Model Based on an n-Dimensional Torus Architecture Using Deadlock-Free Wormhole Routing». In: Euromicro Symposium on Systems Design (DSD´2003), 2003, pp. 166-172.

[ITR07] International Technology Roadmap for Semiconductors. «Executive Summary». 2007 Edition, 2007, 102p.

[JAH07] Jahre, M. «Improving the Performance of Parallel Applications in Chip Multiprocessors with Architectural Techniques». Master Thesis in Computer Science, Norwegian University of Science and Technology, Jul. 2007.

[JER05] Jerraya, A.; Tenhunen, H.; Wolf, W. «Multiprocessor systems-on-chips». IEEE Computer, vol. 38-7, 2005, pp. 113-149.

[LU06] Lu, Z.; Zhong, M.; Jantsch, A. «Evaluation of on-chip networks using deflection routing». In: ACM Great Lakes Symposium on VLSI (GLSVLSI), 2006, pp. 296-301.

[MAR02] Marescaux, T.; Bartic, A.; Verkest, D.; Vernalde, S.; Lauwereins, R. «Interconnection Networks Enable Fine-Grain Dynamic Multi-Tasking on FPGAs». In: International Conference on Field-Programmable Logic and Applications (FPL´02), 2002, pp 795-805.

[MEL03] Mello, A. V.; Möller, L. H. «Arquitetura Multiprocessada em SoCs:Estudo de Diferentes Topologias de Conexão». Trabalho de Conclusão do Curso de Bacharelado em Ciência da Computação, PPGCC-FACIN-PUCRS, 2003, 120p.

[MEL05] Mello, A.; Tedesco, L.; Calazans, N.; Moraes, F. «Virtual channels in network on chip: Implementation and evaluation on Hermes NoC ». In: Symposium on Integrated Circuit and System Design (SBCCI), 2005, pp. 178-183.

[MEL06] Mello, A, V; «Qualidade de Serviço em rede Intra-Chip Implementação e avaliação sobre a rede Hermes », Dissertação de Mestrado, PPGCC-FACIN-PUCRS, Porto Alegre, Mar. 2006, 138 p.

[MIR07] Mirza-Aghatabar, M.; Koohi, S.; Hessabi, S.; Pedram, M. «An Empirical Investigation of Mesh and Torus NoC Topologies under Different Routing Algorithms and Traffic Models». In: Euromicro Symposium on Systems Design (DSD´07), 2007, pp. 19-26.

[MOR04] Moraes, F.; Calazans, N.; Mello, A.; Möller, L.; Ost, L. «Hermes: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip». Integration the VLSI Journal, vol. 38-1, Oct. 2004, pp. 69-93.

[MOR08] Moraes, F.; Mello, A.; Abreu, P.; Calazans, N. «HardNoC - Hardware Platform to Debug the Hermes NoC». GAPH Application Note 02 (GAPP02, v2.0), PPGCC-FACIN-PUCRS, Porto Alegre, Oct. 2008, 15 p.

[NAS07] Nascimento, P.S.B.; Lima, M.E.; Sant'ana, R.E.; Lisboa. E.B.; Silva, S.M.; Seixas, J.L.; «Sistemas Reconfiguráveis: Novo Paradigma para o Desenvolvimento de Aplicações de Computação Massiva de Dados». Centro de Informática-UFPE, Recife, PE, 2007.

[NI93] Ni, M.; McKinley, P. « A Survey of Wormhole Routing Techniques in Direct Networks ». IEEE Computer, vol. 26-2, 1993, pp. 62-76.

[PAN05] Pande, P.; Grecu, C.; Jones, M.; Ivanov, A.; Saleh, R.; «Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures». IEEE Transactions on Computers, vol 54-8, Aug. 2005, pp. 1025–1040.

[PON08] Pontes, J.; Moreira, M.; Soares,R.; Calazans, N.; «Hermes-GLP: A GALS Network on Chip Router with Power Control Techniques». IEEE Computer Society Annual Symposium on VLSI, 2008, pp. 347–352.

[SCH07] Scherer, C. «Redes Intrachip com Topologia Toro e Modo de Chaveamento Wormhole: Projeto, Geração e Avaliação». Dissertação de Mestrado, PPGCC-FACIN-PUCRS, Porto Alegre, Mar. 2007, 101 p.

[SIL97] Silberschatz, A. and Galvin, P. «Operating Systems Concepts». New York, Addison Wesley, 5th ed., 1997, 888 p.

[TIL07] Tilera Corporation. «TILE64™ Processor». Product Brief, Santa Clara, CA, EUA, Aug. 2007, 2p.

[VAN07] Vangal, S.; Howard, J.; Ruhl, G.; Dighe, S.; Wilson, H.; Tschanz, J.; Finan, D.; Iyer, P.; Singh, A.; Jacob, T.; Jain, S.; Venkataraman, S.; Hoskote, Y.; Borkar, N. «An 80-Tile 1.28TFLOPS

Network-on-Chip in 65nm CMOS». In: IEEE International Solid-State Circuits Conference (ISSCC), Feb. 2007, pp. 5-7.

[VAS07] Vasconcelos, A. «Adaptando um Processo de Desenvolvimento de Software para Análise de Cobertura de Código». In: II Encontro Brasileiro de Teste de Software (IBTS´07), 2007.

[WOL04] Wolf, W. «The Future of Multiprocessor Systems-on-Chip». In: Design Automation Conference (DAC´04), 2004, pp. 681-685.

[ZEF03] Zeferino, C. «Redes-em-Chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho». Tese de Doutorado, PPGC-II-UFRGS, Porto Alegre, Apr. 2003, 212 p.