PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Estratégias para Otimização de Desempenho em Redes Intra-Chip

Implementação e Avaliação sobre a Rede Hermes

Everton Alceu Carara

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Fernando Gehm Moraes

Porto Alegre
2008

Resumo

Os ganhos de desempenho proporcionados pelas arquiteturas paralelas não estão relacionados somente ao poder computacional dos vários elementos de processamento. A arquitetura de interconexão, responsável pela intercomunicação dos elementos de processamento, tem um papel relevante no desempenho geral do sistema. Redes intra-chip (NoCs) podem ser vistas como a principal arquitetura de interconexão responsável pelo futuro das tecnologias multiprocessadas, as quais estão rapidamente prevalecendo em SoCs. Atualmente, existem inúmeros projetos de NoCs disponíveis, os quais focam diferentes aspectos desse tipo de arquitetura de interconexão. Alguns aspectos relevantes considerados durante o projeto de NoCs são a capacidade de atingir QoS (*Quality-of-Service*), a redução de latência, a redução do consumo de energia e o mapeamento de aplicações. Este trabalho propõem diversos mecanismos para otimizar o desempenho das NoCs, contribuindo para que estas tornem-se a arquitetura de interconexão prevalente em SoCs multiprocessados modernos. Os mecanismos propostos abrangem diferentes aspectos relativos à otimização de desempenho como latência, vazão, contenção e tempo total para a transmissão de conjuntos de pacotes. As avaliações realizadas apresentam ganhos de desempenho relativos a todos mecanismos propostos, comprovando a eficiência dos mesmos.

Palavras Chave: NoCs, canais virtuais/replicados, sessão, roteamento adaptativo, *multicast, spilling*.

Abstract

Performance gains provided by multiprocessor architectures are not only related to the computational power of the several processing elements. The interconnection architecture, responsible by the communication among the several processing elements, has an important contribution in the overall performance. NoCs can be seen as the main interconnection architecture responsible by the future of the multiprocessed technologies, which are rapidly prevailing in SoCs. A considerable number of NoC designs are available, focusing on different aspects of this type of communication infrastructure. Example of relevant aspects considered during NoC design are quality-of-service achievement, the choice of synchronization method to employ between routers, latency reduction, power consumption reduction and application modules mapping. This work proposes several mechanisms to optimize NoC performance, contributing for them to become the prevalent interconnection architecture in modern multiprocessed SoCs. The proposed mechanisms include different aspects of performance optimization like latency, throughput, contention and total time to transmit sets of packets. The conducted evaluations show performance gains in all proposed mechanisms, demonstrating their efficiency.

Keywords*: NoCs, virtual/replicated channels, session, adaptive routing, multicast, spilling*.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ATM	A*synchronous Transfer Mode*
ASIC	*Application Specific Integrated Circuit*
BE	*Best Effort*
CC	Chaveamento por Circuito
CP	Chaveamento por Pacotes
CR	Canais Replicados
CS	*Connect Setup*
CV	Canais Virtuais
DyAD	*Dynamic Adaptive Deterministic*
DyXY	*Dynamic XY*
EOP	*End of Packet*
FIFO	*First-In First-Out*
FPGA	*Field-Programmable Gate Array*
GT	*Guaranteed Throughput*
HDTV	*High-Definition Television*
IP	*Intellectual Property*
LDM	*Lane Division Multiplexing*
MAC	*Media Access Control*
Mbps	*Megabits por segundo*
MOTIM	Montagem de Interfaces do Módulo TETHA
MPEG	*Moving Picture Experts Group*
MPSoC	*Multi Processor System on a Chip*
MUX	Multiplexador
NI	*Network Interface*
NoC	*Network-on-Chip*
OSI	*Open System Interconnection*
PC	Pacote/Célula
PCC	*Packet Connected Circuit*
QoS	*Quality-of-Service*
RTL	*Register Transfer Level*
SDM	*Spatial Division Multiplexing*
SoC	*System on a Chip*
TC	Tamanho da Célula
TCP	*Transmission Control Protocol*
TDM	*Time Division Multiplexing*
TETHA	Circuito para Transporte de Dados Ethernet sobre Redes de Alta Velocidade
TI	Taxa de Injeção
VHSIC	*Very High Speed Integrated Circuit*
VHDL	*VHSIC Hardware Description Language*

Sumário

1 INTRODUÇÃO

Desde a introdução do conceito de *System-on-Chip* (SoC) nos anos 90, as soluções para a sua estrutura de comunicação têm sido caracterizadas geralmente por combinações *ad hoc* de barramentos e canais ponto-a-ponto [LAH01]. Barramentos empregam os conceitos de compartilhamento do meio físico, permitindo interconectar de forma simples os núcleos do sistema. Entretanto, em um sistema com muitos núcleos, o barramento rapidamente se torna o gargalo da comunicação. Para barramentos multi-mestre, o problema da arbitragem não é trivial. Estruturas como *crossbars* contornam algumas limitações de barramentos, mas também sofrem com a falta de escalabilidade. Canais dedicados ponto-a-ponto são ótimos em termos de largura de banda, latência e consumo de energia. Contudo, o número de canais necessários cresce de maneira exponencial, à medida que novos núcleos são adicionados ao sistema.

Um novo conceito para a comunicação entre os núcleos de um SoC é baseado em redes, conhecido como *Network-on-Chip* (NoC) [GUE00][DAL01][BEN02][JAN03]. NoCs são vistas mais como um conceito unificado do que como uma nova alternativa de estrutura de comunicação. Entre os pesquisadores, existem duas idéias amplamente defendidas: (*i*) NoC é um subconjunto de SoC e (*ii*) NoC é uma extensão de SoC. No primeiro ponto de vista, a NoC é definida estritamente como uma estrutura de comunicação para encaminhar pacotes. No segundo ponto de vista, a NoC é definida mais amplamente, incluindo também questões relacionadas à aplicação, arquitetura do sistema e seu impacto na comunicação [BJE06].

NoCs são vistas por muitos pesquisadores e projetistas como uma resposta ao problema da escalabilidade, encontrado nas arquiteturas baseadas em barramentos. A premissa básica da evolução proporcionada pelas NoCs é fundamentalmente simples: a interconexão intra-chip deve ser projetada utilizando os mesmos princípios que direcionam o desenvolvimento das redes de computadores, as quais têm demonstrado escalabilidade, melhora exponencial de desempenho, robustez e confiabilidade em muitos anos de rápida evolução. A literatura relacionada a NoCs tem prosperado nos últimos anos, com muitas contribuições relativas a desenvolvimento, análise e implementação [BEN05].

Latência e potência atualmente são considerados pelos pesquisadores os principais desafios na área de NoCs. Esforços de pesquisa nessas questões visam potencializar essas redes de maneira a torná-las a principal estrutura de interconexão em SoCs multiprocessados. Minimizar a latência em NoCs é crucial, visto que elas provavelmente substituirão as estruturas de interconexão baseadas em barramentos (barramentos simples, segmentados ou hierárquicos), as quais apesar de apresentarem baixa latência, não são escaláveis e a contenção posterga significativamente o início de uma dada transmissão. Redes com baixa latência facilitam o trabalho dos programadores e projetistas de sistema, pois favorecem a exploração do paralelismo e a intercomunicação entre os núcleos de processamento [DAL06].

Inovações nas arquiteturas de roteadores são fundamentais para a redução da latência, procurando-se manter baixo os consumos de área de silício e potência. Diversos segmentos arquiteturais são citados em [DAL06]. Exemplos destes compreendem:

- utilizar a abundância de espaço disponível para conexões nas tecnologias submicrônicas para replicar canais físicos;
- utilizar um modo de chaveamento híbrido pacotes/circuito, visto que ambos apresentam vantagens e desvantagens dependendo do tipo de tráfego na rede;
- reduzir o número de *hops* na transmissão das mensagens, dado que a transmissão dos dados nas redes usualmente é feita na forma de um *pipeline* (topologias irregulares)*;*
- exploração de topologias além da tradicional malha.

Para auxiliar na pesquisa relativa tanto à redução de latência quanto à potência, a comunidade de pesquisa precisa desenvolver um conjunto de *benchmarks* e metodologias de avaliação, as quais possibilitem avaliações realistas das abordagens propostas e uma comparação uniforme entre as várias propostas. Dessa maneira será possível fazer a comparação direta de resultados além de facilitar a troca de informações entre os pesquisadores.

1.1 Projeto TETHA

As propostas apresentadas na presente Dissertação vêm sendo implementadas no contexto do projeto TETHA [TET06]. O principal objetivo desse projeto é desenvolver IPs para auxiliar a implementação de módulos de hardware, visando o transporte de tráfego IP em redes de alta velocidade que empregam Ethernet nos níveis mais baixos da pilha de protocolos.

1.1.1 Arquitetura MOTIM

Trata-se de uma arquitetura reusável para a implementação de *switches* Ethernet com baixa latência e alta vazão. A estrutura de interconexão utilizada é uma NoC, o que garante sua escalabilidade. As propostas apresentadas na presente Dissertação foram implementadas e validadas (simulação/prototipação) nesta arquitetura. A arquitetura MOTIM vem sendo desenvolvida no contexto do projeto THETA. A Figura 1 ilustra, a título de exemplo, a arquitetura MOTIM conectada a 4 sub-redes.

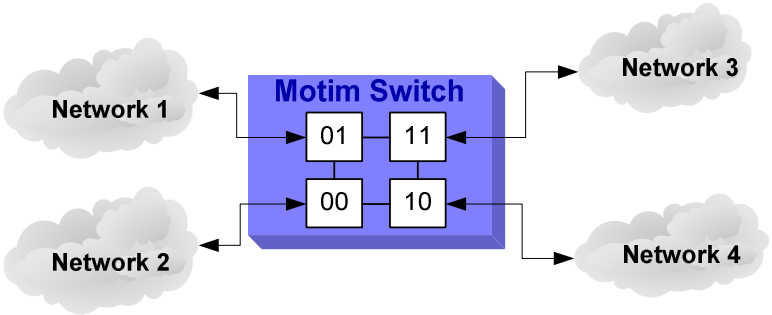


Figura 1 – Instância reduzida da arquitetura MOTIM, conectada a 4 sub-redes.

A arquitetura MOTIM foi projetada para conectar até 24 sub-redes. A arquitetura é composta por 4 módulos: MAC Ethernet, Pacote/Célula (PC), *Network Interface* (NI), e *Network-on-Chip* (NoC). A Figura 2 apresenta a arquitetura MOTIM, mostrando como os módulos se interconectam.

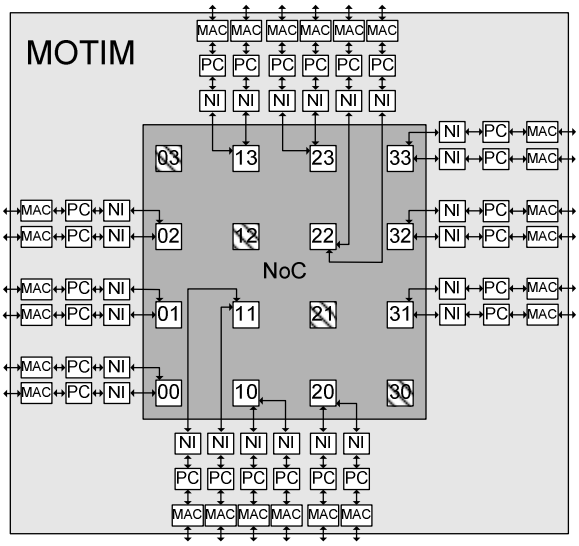


Figura 2 – Arquitetura MOTIM completa.

O módulo MAC Ethernet foi adaptado do módulo disponível em [NGU01]. Sua função é receber e transmitir pacotes Ethernet 100 Mbps. O módulo PC tem por função fragmentar o pacote Ethernet em células na transmissão (PC→NI) e remontar na recepção (NI→PC).

O módulo NI realiza a interface com a NoC e executa o roteamento dos pacotes Ethernet, definindo para qual porta da NoC um dado pacote Ethernet deve ser enviado. Na transmissão dos dados (NI→NoC), o módulo NI armazena as células, realiza o roteamento para um roteador destino e transmite a célula. Na recepção (NoC→NI) o módulo NI repassa as células para o módulo PC e armazena internamente a relação endereço do roteador origem – endereço Ethernet em uma tabela de roteamento.

A NoC tem a função de transporte de dados. Internamente, a rede é composta por 16 roteadores em topologia malha 4x4, havendo duas portas locais por roteador para conexão a módulos externos. Externamente, utilizam-se 24 portas para conexão a entrada/saída de dados (pacotes Ethernet) e 8 portas para conexão a outros módulos, como processadores para controle e supervisão do sistema.

1.2 Objetivos

O objetivo *estratégico* da presente dissertação é dominar a tecnologia de redes intra-chip e mecanismos utilizados para otimizar seu desempenho. Alguns dos mecanismos implementados são mapeamentos diretos de abordagens projetadas para outras áreas como multicomputadores e redes de computadores, visto que essas áreas podem ser consideradas paternas em relação a área de redes intra-chip. Mesmo aplicando-se técnicas oriundas de multicomputadores e redes de computadores, deve-se observar as restrições inerentes a sistemas embarcados, como consumo de potência e área de silício consumida.

Dentre os objetivos *específicos* enumeram-se:

- *Replicação de canais físicos*;

- Implementação de *chaveamento por circuito*;
- Adição de um *controle de sessão* sobre o chaveamento por circuito;
- Aplicação da *adaptatividade em função do tráfego* sobre um algoritmo de roteamento adaptativo utilizado como estudo de caso;
- Implementação de um *algoritmo de multicast/broadcast* livre de *deadlock*;
- Implementação de um módulo *spilling*;
- *Avaliação* das implementações.

1.2.1 Organização do Documento

Cada capítulo da dissertação apresenta uma proposta para otimização de desempenho. Eles estão estruturados na forma de artigos, na sua maioria apresentado trabalhos relacionados, descrição, validação, avaliação e conclusão. Os assuntos tratados em cada capítulo são listados a seguir.

O Capítulo 2 apresenta a replicação de canais físicos como uma alternativa para canais virtuais. O principal benefício proporcionado por essa abordagem é o aumento da largura de banda do roteador e um conseqüente ganho de desempenho em situações de fluxos concorrendo por caminhos em comum.

O Capítulo 3 propõe um mecanismo de controle de sessão associado ao chaveamento por circuito. A partir dessa proposta, a largura de banda dos canais físicos é maximizada através de transmissões em rajada na taxa do canal. O controle de sessão eleva o rendimento dos recursos alocados eliminando a ociosidade da conexão, a qual é comum quando o chaveamento por circuito é empregado.

O Capítulo 4 apresenta um mecanismo de adaptatividade do roteamento em função do tráfego, o qual pode ser aplicado a qualquer algoritmo de roteamento adaptativo. Através desse mecanismo o algoritmo de roteamento adaptativo toma decisões baseado em informações locais de congestionamento (roteadores vizinhos) e desvia de áreas congestionadas. Como estudo de caso é apresentado o algoritmo de roteamento Hamiltoniano.

O Capítulo 5 apresenta uma implementação livre de *deadlock* do algoritmo de *multicast dual-path*, o qual foi proposto originalmente para multicomputadores. A partir desse algoritmo evita-se a transmissão de múltiplas cópias de uma mesma mensagem para diferentes destinos.

O Capítulo 6 propõem uma nova solução para reduzir a contenção em NoCs, através de um módulo intermediário entre origem e destino chamado de *spilling*. Sempre que uma origem não conseguir enviar pacotes para um determinado destino, por falta de um caminho disponível, ela pode tentar transmitir seus pacotes para o módulo *spilling*. Esse módulo armazena temporariamente os pacotes e se encarrega de retransmiti-los ao correspondente destino quando houver disponibilidade de caminho.

A conclusão global da dissertação e os trabalhos futuros são apresentados no Capítulo 7.

2 REPLICAÇÃO DE CANAIS FÍSICOS

NoCs podem ser modeladas como um grafo do tipo $G=<R,E>$, onde o conjunto de vértices R é o conjunto de roteadores, e o conjunto E representa seus enlaces de comunicação bidirecionais. Cada enlace usualmente contém dois canais físicos unidirecionais, os quais possibilitam a comunicação entre roteadores vizinhos, de maneira simultânea nos dois sentidos. A largura dos canais físicos (*phit*) é comumente parametrizável nos projetos de NoCs e fator determinante na largura de banda máxima do roteador.

Considerando as tecnologias atuais, um tamanho de *phit* igual a 32 ou 64 bits subutiliza a quantidade de conexões que pode ser usada para conectar roteadores vizinhos. Tomando como exemplo uma tecnologia de 90 nm, 140 nm *wire pitch* e um roteador com área igual a 0,1 mm^2 [LEI06], cada roteador poderia ser conectado ao seu vizinho através de 2258 conexões (Figura 3), considerando o uso de somente uma camada de metal. Conseqüentemente, esse cenário favorece a replicação ao invés da multiplexação dos canais físicos.

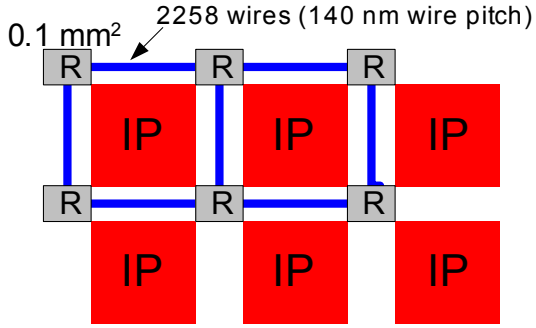


Figura 3 – Número de conexões disponíveis para interconectar roteadores (R), na tecnologia de 90 nm.

Logo, uma forma simples de se obter maior desempenho em NoCs, é utilizar canais físicos replicados ao invés de uma técnica de multiplexação, como canais virtuais por exemplo. O objetivo de usar canais virtuais ou replicados é reduzir o congestionamento quando diferentes fluxos competem pelo mesmo caminho dentro da rede. Ambas técnicas também podem ser utilizadas para eliminar *deadlock* em algoritmos de roteamento adaptativos e *multicast* [BOP98].

2.1 Trabalhos Relacionados

Esta seção apresenta alguns trabalhos relacionados à alocação de canais físicos para a comunicação intra-chip entre elementos de processamento e roteadores, e entre roteadores vizinhos. Como será mostrado a seguir, a transmissão simultânea de diferentes fluxos entre elementos vizinhos pode ser realizada através da multiplexação ou replicação do meio físico. A partir da multiplexação, os diferentes fluxos compartilham o mesmo meio físico entre elementos conectados. Na replicação, os canais físicos que conectam os elementos são replicados e cada fluxo aloca um canal físico para a comunicação.

2.1.1 Multiplexação dos Canais Físicos

Os canais físicos podem ser multiplexados de maneira espacial ou temporal, permitindo assim o uso do mesmo canal físico por diferentes fluxos. Essas técnicas aumentam o desempenho da NoC pois visam maximizar a largura de banda dos canais físicos.

2.1.1.1 Multiplexação Temporal

O mecanismo de multiplexação mais comum entre NoCs é o *Time Division Multiplexing* (TDM). O TDM compartilha os canais físicos dividindo-os em canais lógicos (ou canais virtuais) [DAL92] [MUL04] [MEL05]. Nesse esquema, o tempo é dividido em períodos iguais chamados de fatias de tempo. Durante uma fatia de tempo, a largura de banda disponível é exclusivamente dedicada a um fluxo. O TDM reduz o congestionamento e conseqüentemente melhora o desempenho da NoC. A Figura 4 apresenta um exemplo de TDM onde um canal físico é multiplexado em quatro canais virtuais. Observe que o tráfego D aloca o canal físico por 1 fatia de tempo; em seguida o tráfego C aloca o canal por 1 fatia de tempo também; depois o tráfego B aloca o canal por 2 fatias de tempo e por último o tráfego A aloca o canal por 4 fatias de tempo.

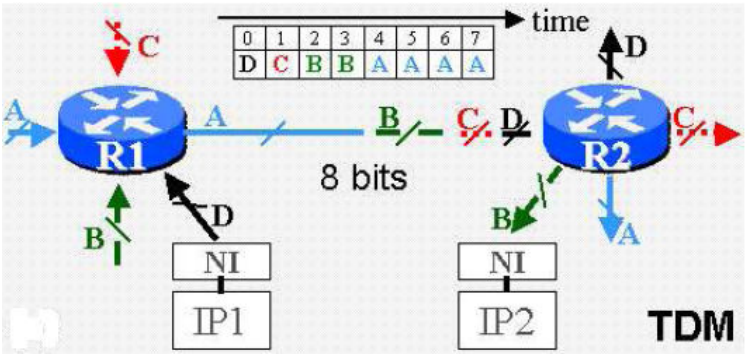


Figura 4 - Multiplexação temporal do canal físico em quatro canais virtuais [LER05].

Alguns mecanismos de QoS utilizam essa técnica para garantir requisitos de vazão, alocando tantas fatias de tempo quantos forem necessários. O principal problema do TDM é que a configuração dos chaveamentos do roteador tem de ser atualizada a cada fatia de tempo, pois o canal físico deve ser alocado para outro fluxo (fluxo A, B, C ou D, no caso da Figura 4). Para isso, os roteadores têm de armazenar essas configurações em memórias locais, resultando em um alto consumo de área e energia [LER05]. Algumas NoCs que utilizam TDM são: Æthereal [GOO05], Nostrum [MIL04] e HERMES [MEL05].

2.1.1.2 Multiplexação Espacial

Alternativas para o TDM têm sido propostas, como é o caso do *Spatial Division Multiplexing* (SDM) [LER05] e o *Lane Division Multiplexing* (LDM) [WOL05]. No SDM, ao invés de toda a largura de banda de um canal ser alocada por uma fatia de tempo, os bits do canal são alocados individualmente, conforme for a largura de banda exigida, por todo tempo de vida da conexão (como no chaveamento por circuito). A Figura 5 ilustra um exemplo de multiplexação espacial. Observe que os quatro tráfegos utilizam o canal físico simultaneamente. Dessa maneira, os

chaveamentos são realizados apenas uma vez (durante a alocação) e desfeitos ao final da transmissão. Antes de serem transmitidos, os pacotes são serializados na origem, e quando chegam ao destino são deserializados. Os resultados apresentados por Leroy et al. em [LER05] mostram, através de um estudo de caso (aplicação de vídeo), um ganho de 8% no consumo de energia e 24% no consumo de área, usando SDM ao invés de TDM. No entanto, o SDM aumentou 37% o caminho crítico, conseqüentemente reduzindo a freqüência máxima de operação.

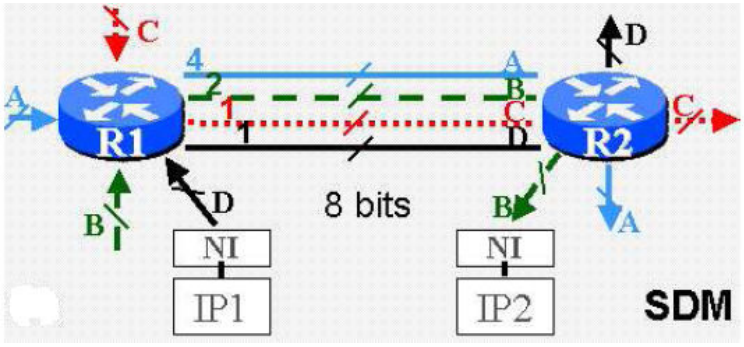


Figura 5 – Multiplexação espacial do canal físicos [LER05].

Em [WOL05], Wolkotte et al. empregam uma abordagem semelhante ao SDM, chamada de *Lane Division Multiplexing* (LDM). A largura e o número de *lanes* são parâmetros ajustáveis do projeto. Eles devem ser ajustados durante o projeto do SoC, visando atingir os requisitos de largura de banda exigidos pela aplicação. A Figura 6 ilustra a arquitetura do LDM. Através do módulo *Data Coverter* os dados são serializados antes de entrar na rede e deserializados quando saem da rede.

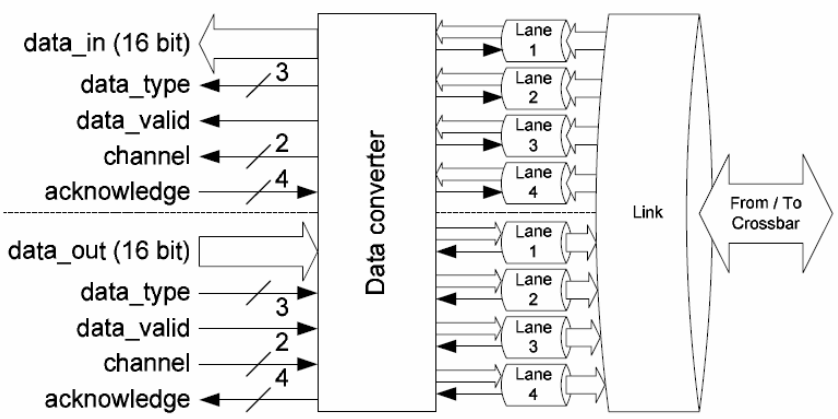


Figura 6 – Canal físico multiplexado em quatro *lanes* de tamanho fixo [WOL05].

Alguns resultados comparativos apresentados pelo autor, comparando um roteador LDM com chaveamento por circuito contra um TDM com chaveamento por pacotes, mostram menor consumo de energia, menor área e maior vazão. Como desvantagens do LDM associado ao chaveamento por circuito, o autor cita a falta de flexibilidade no projeto do roteador (*lanes* de tamanho fixo) e a falta de suporte a tráfegos do tipo *best-effort* (BE).

2.1.2 Replicação dos Canais Físicos

Em [BOU06] é proposta uma NoC com alta vazão baseada em uma topologia árvore gorda. Tomando vantagem do contexto intra-chip, os roteadores têm os canais físicos replicados de maneira que existam mais portas de saída do que de entrada. Tendo como estratégia a eliminação da contenção, a latência é reduzida e a vazão é maximizada. Não existindo contenção, elimina-se também a necessidade de *buffers* internos, os quais são movidos para fora da NoC, no lado dos IPs (Figura 7c).

Em uma árvore gorda, o pacote é roteado para cima até atingir um roteador que tenha um canal físico de descida que leve em direção ao destino. Esse roteador é chamado de topo. Visto que a contenção ocorre somente na descida, os canais físicos são replicados no sentido roteador → IP. A Figura 7 ilustra: (a) árvore gorda regular; (b) árvore gorda com os canais físicos replicados e (c) a arquitetura de *buffers* entre roteador e IP.

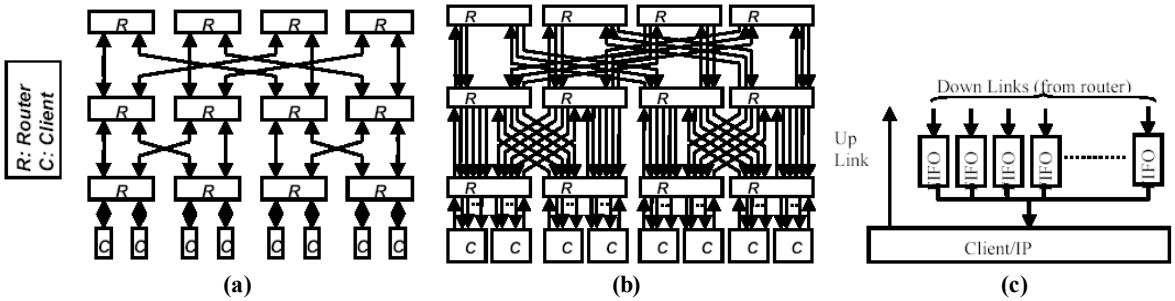


Figura 7 - (a) árvore gorda regular; (b) árvore gorda com os canais físicos replicados; (c) arquitetura de *buffers* entre roteador e IP [BOU06].

Em [SET06a], Sethuraman et al. propõem reduzir o consumo de área em NoCs reduzindo o número de roteadores. O objetivo é atingido através da replicação da porta local, dessa maneira torna-se possível interconectar mais IPs utilizando menos roteadores. A Figura 8a ilustra um roteador com quatro portas locais além das tradicionais *North*, *South*, *East* e *West*, características das topologias malha. Para identificar a porta local destino do pacote, além da identificação do roteador, é necessário um campo extra no cabeçalho (Figura 8b).

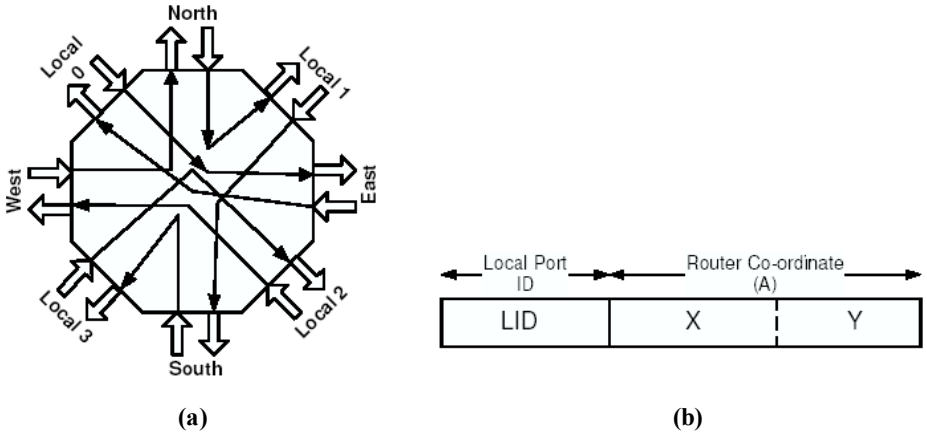


Figura 8 – (a) roteador com porta local replicada; (b) cabeçalho dos pacotes [SET06a].

A arquitetura proposta apresenta ganhos em termos de área, consumo de energia e desempenho. Intuitivamente, um roteador com *n* portas locais seria a melhor opção. No entanto algumas questões como: aumento do caminho crítico, concorrência pelos recursos do roteador e restrições de E/S limitam o número máximo de portas locais.

A PNoC [HIL05] tem como principal atrativo a flexibilidade arquitetural. Sua topologia pode ser personalizada, dependendo da maneira como os roteadores são interligados no sistema. A interconectividade entre os IPs e roteadores é flexível e definida pelo projetista do sistema. Durante o projeto é possível construir uma variedade de arquiteturas de rede, cada uma delas visando atingir os requisitos de uma determinada aplicação. A Figura 9 ilustra duas diferentes arquiteturas, ambas com canais físicos replicados. Os roteadores são parametrizáveis em relação ao número de portas e à largura dos canais físicos.

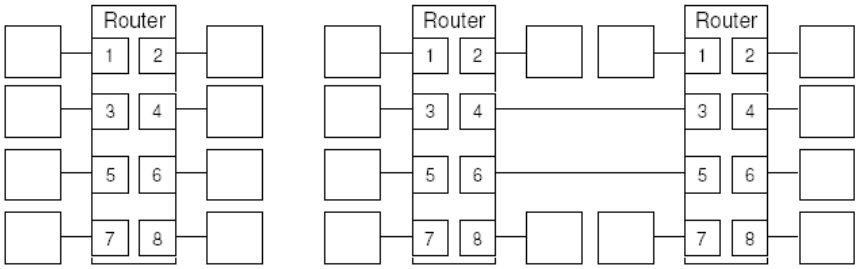


Figura 9 – Duas diferentes arquiteturas de rede [HIL05].

A arquitetura de roteador proposta em [JUN07] replica os canais físicos no eixo Y, criando dois caminhos verticais disjuntos (N1/S1 – *North1*/*South1* e N2/S2 – *North2*/*South2*), visando a prevenção de *deadlock* em algoritmos adaptativos. A Figura 10a ilustra a arquitetura da NoC e a Figura 10b a arquitetura do roteador. Pacotes que chegam no roteador pela direção *West* deslocam-se no eixo Y utilizando os canais N1 e S1, enquanto que os pacotes que chegam pela direção *East* deslocam-se no eixo Y utilizando os canais N2 e S2. Observe também na Figura 10b, que a porta local (INT) possui dois canais de injeção (*IntL-in* e *IntR-in*), os quais são usados dependendo da posição do destino em relação a origem. Outra característica interessante dessa arquitetura é a largura dos canais físicos, que é igual a 64 bits.

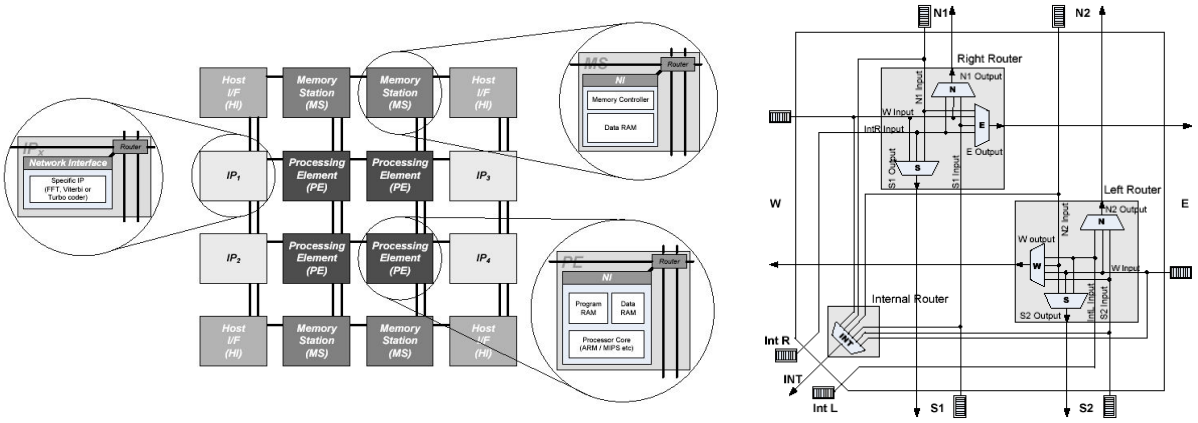


Figura 10 – (a) arquitetura da NoC; (b) arquitetura do roteador.

2.2 Descrição da Arquitetura Implementada

Esta seção descreve o projeto da replicação dos canais físicos do principal componente da infra-estrutura HERMES [MOR04], o roteador. A infra-estrutura HERMES possibilita a geração de redes intra-chip com topologia malha, chaveamento por pacotes *wormhole* e baixa sobrecarga de área. Diz-se infra-estrutura porque não se trata de uma única rede intra-chip. Existe um conjunto de parâmetros que podem ser definidos pelo usuário, tais como: (*i*) controle de fluxo; (*ii*) dimensões da malha; (*iii*) largura dos canais físicos; (*iv*) profundidade dos *buffers*; e (*v*) algoritmo de roteamento. O número de portas dos roteadores variam conforme sua posição na malha (Figura 11a). O roteador central possui cinco portas bidirecionais (*North*, *South*, *East*, *West* e *Local*). A Figura 11b apresenta a arquitetura do roteador central. Os principais componentes são: (*i*) *Switch Control*, responsável pela arbitragem e roteamento; (*ii*) *Crossbar*, responsável pela conexão das portas de entrada às portas de saída; (*iii*) *Input Buffers* para o armazenamento temporário de *flits*.

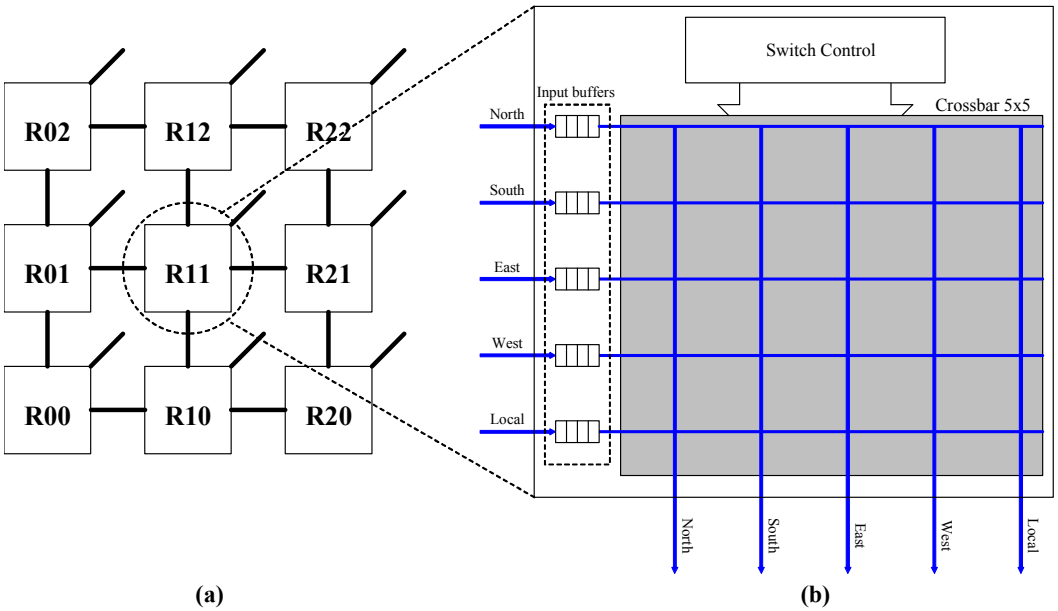


Figura 11 – (a) malha 3x3; (b) roteador central da infra-estrutura HERMES.

O objetivo da replicação dos canais físicos é prover baixa latência aumentando a largura de banda do roteador. A partir do roteador HERMES, com cinco portas bidirecionais, um novo roteador é criado contendo dez portas bidirecionais (Figura 12). Os canais físicos foram replicados em todas as direções, permitindo dois fluxos simultâneos no mesmo sentido entre roteadores vizinhos. Para um melhor aproveitamento da largura de banda, a porta local também é replicada. Dessa maneira, a porta local pode receber *n* fluxos distintos simultaneamente, onde *n* representa o grau de replicação. Essa característica permite a conexão de *n* IPs no mesmo roteador, reduzindo assim o número de roteadores da NoC e a área total do SoC [SET06a]. É possível também variar a largura de banda da comunicação entre IP e rede, atribuindo mais ou menos conexões de cada IP ao mesmo roteador.

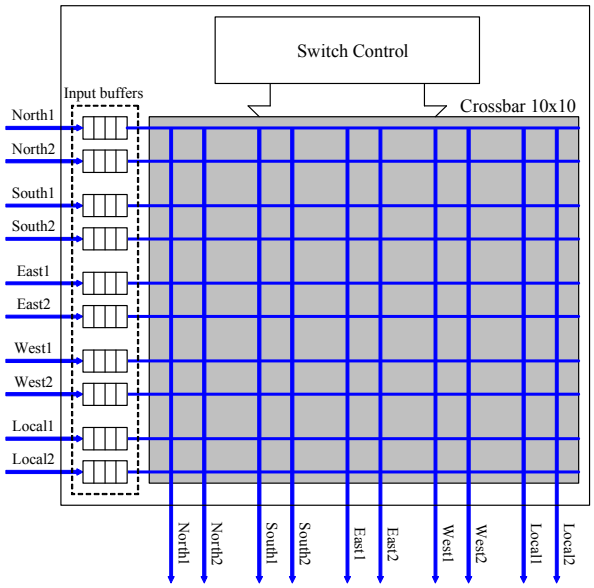


Figura 12 – Roteador HERMES com canais físicos replicados.

Como se pode observar na Figura 12 em relação à Figura 11, o tamanho/quantidade dos componentes que compõem a arquitetura do roteador foram duplicados, o que implica um custo de área na mesma proporção. Não há prioridade nem ordenamento na alocação dos canais físicos em uma mesma direção, de maneira que o primeiro canal livre é alocado. Juntamente com os canais físicos, todos os sinais que compõem a interface das portas de comunicação foram replicados, como mostra a Figura 13. Os sinais correspondentes à interface de saída são: (1) *data_out*: dado a ser transmitido; (2) *tx*: indica disponibilidade de dados no barramento *data_out*; (3) *eop_out*: indica que o dado no barramento *data_out* corresponde ao último *flit* do pacote; (4) *credit_i*: indica que há espaço para o armazenamento de *flits* no *buffer* do roteador vizinho.

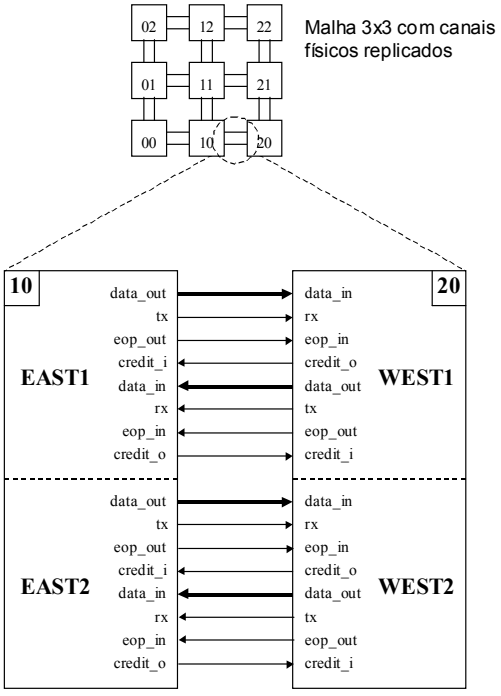


Figura 13 – Interface entre roteadores vizinhos.

O tamanho do *flit* na infra-estrutura HERMES é parametrizável. O tamanho máximo de pacote suportado pela infra-estrutura é igual a $2^{(\text{largura do } flit \text{ em bits})}$-1 *flits*. O primeiro e o segundo *flit* de um pacote são informações de cabeçalho, sendo respectivamente o endereço do roteador destino e o número de *flits* do corpo do pacote. Nessa nova arquitetura de roteador, inseriu-se o sinal *eop* (*end-of-packet*) à interface das portas de comunicação do roteador HERMES. Através desse sinal elimina-se a limitação quanto ao tamanho máximo de pacote e não há a necessidade de um *flit* no cabeçalho indicando o tamanho do corpo do pacote. Eliminam-se também os contadores de *flits* nos *buffers* de entrada, os quais indicam quantos *flits* do pacote foram transmitidos. O cabeçalho do pacote passa ter apenas um *flit* com os campos descritos na Figura 14, considerando-se um tamanho de *flit* igual a 8 bits.

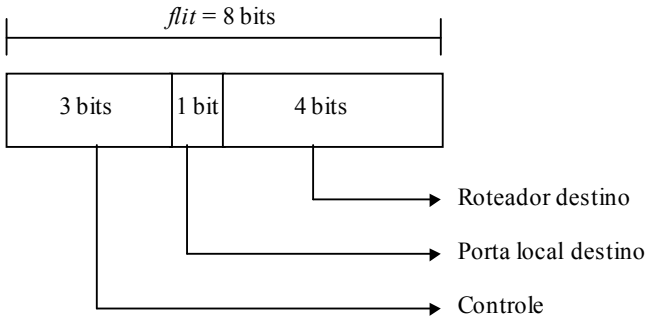


Figura 14 – Cabeçalho dos pacotes do roteador com canais físicos replicados.

O campo *Controle* não é usado na implementação de canais replicados, no entanto será utilizado nas demais implementações desse trabalho para diferenciação entre modos de chaveamento (por pacotes ou por circuito) e *multicast*. As demais características da infra-estrutura HERMES como modo de chaveamento (pacotes/*wormhole*), arbitragem (*round-robin*), controle de fluxo (baseado em créditos) e roteamento (XY) não foram alteradas.

2.3 Validação do Roteador com Canais Físicos Replicados

O roteador com canais físicos replicados foi descrito em VHDL e validado por simulação funcional e prototipação em FPGA. A Figura 15 ilustra um cenário no qual os IPs IP1 e IP2 enviam simultaneamente pacotes para os IPs IP3 e IP4, respectivamente.

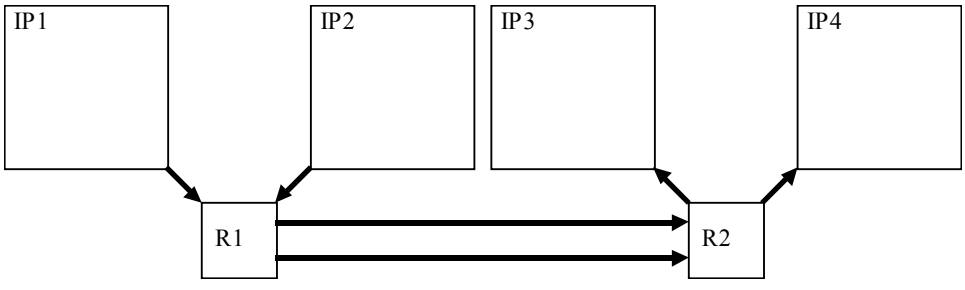


Figura 15 – Dois fluxos simultâneos na direção *EAST*.

Na Figura 16 é apresentada a simulação funcional correspondente. São mostradas as formas de onda correspondentes às interfaces de saída EAST1 e EAST2 do roteador R1, e as

interfaces de entrada dos IPs IP3 e IP4. Os passos da simulação são descritos a seguir, onde a numeração tem correspondência com a Figura 16.

1. Roteador R1 inicia a transmissão de dados pela porta *EAST1*.
2. Roteador R1 inicia transmissão de dados pela porta *EAST2*. A partir desse momento existem dois fluxos simultâneos na mesma direção.
3. IP3 começa a receber o pacote.
4. IP4 começa a receber o pacote. A partir desse momento ambos IPs estão recebendo pacotes simultaneamente.
5. Roteador R1 sinaliza que o último *flit* do pacote está sendo enviado pela porta *EAST1*.
6. Roteador R1 sinaliza que o último *flit* do pacote está sendo enviado pela porta *EAST2*.
7. IP3 recebe o último *flit* do pacote.
8. IP4 recebe o último *flit* do pacote.

Apesar dos IPs IP1 e IP2 injetarem simultaneamente os pacotes na rede, ele aparecem defasados nas portas de saída *EAST1* e *EAST2* devido ao fato de que a arbitragem e o roteamento no roteador R1 são centralizados e compartilhados pelas portas de entrada.

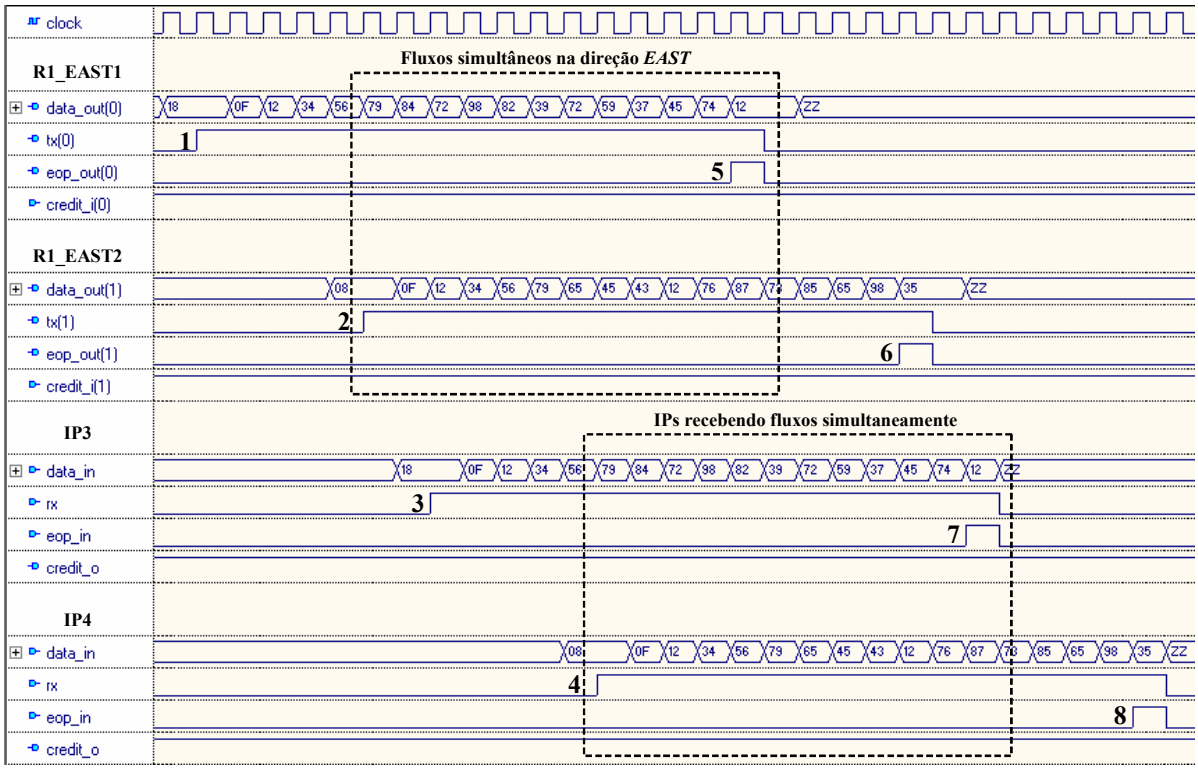


Figura 16 – Simulação funcional correspondente ao cenário da Figura 15.

2.4 Avaliação de Desempenho e Consumo de Área

Esta seção tem por finalidade comparar, em termos de área, latência e vazão, uma NoC com arquitetura baseada em canais replicados contra uma com arquitetura baseada em canais

virtuais. A arquitetura baseada em canais virtuais possui dois canais lógicos em cada direção e a arquitetura baseada em canais físicos possui dois canais físicos em cada direção. A Figura 17a ilustra uma típica arquitetura de roteador com os canais físicos divididos em dois canais lógicos. Observe a presença de demultiplexadores nas portas de entrada e multiplexadores nas portas de saída, os quais aumentam o consumo de área do roteador. Apesar de não estarem explícitos na Figura 17a, existem também circuitos extras responsáveis pela multiplexação temporal dos canais físicos, os quais efetivamente controlam os multiplexadores. Na Figura 17b tem-se uma arquitetura equivalente empregando a replicação dos canais físicos. Observe a eliminação dos demultiplexadores e multiplexadores.

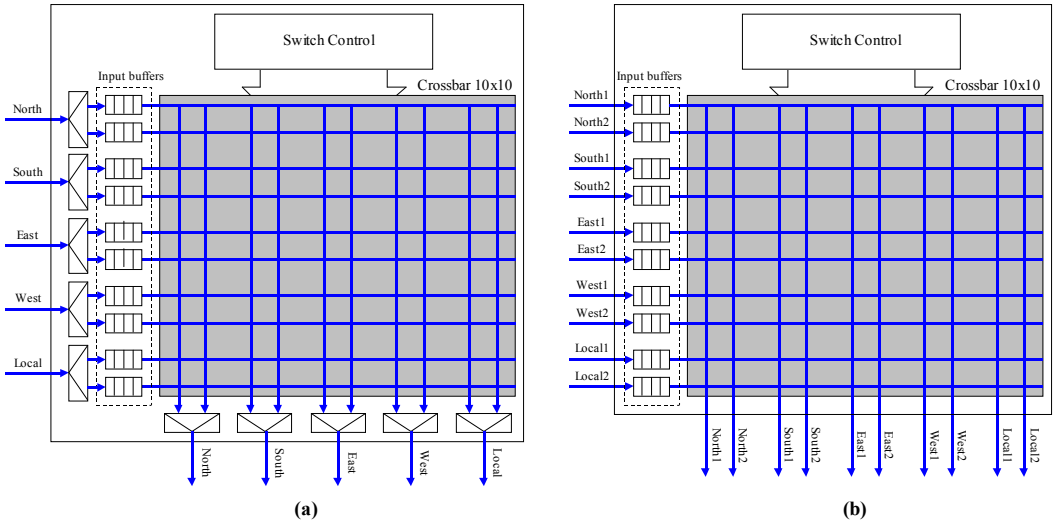


Figura 17 – Comparação entre arquiteturas. (a) arquitetura baseada em canais virtuais; (b) arquitetura baseada em canais replicados.

A complexidade do *Switch Control* em ambas abordagens é similar, visto que sua principal função é controlar o *crossbar* interno, o qual tem a mesma dimensão em ambas arquiteturas. Os *buffers* de entrada (*input buffers*) em ambas abordagens têm a mesma profundidade, consumindo a mesma área de silício. A replicação dos canais físicos dobra a largura de banda do roteador, quando comparada com a multiplexação dos canais físicos (canais virtuais), devido a duplicação no seu número de portas.

Ambas NoCs foram descritas em VHDL sintetisável, derivando a infra-estrutura HERMES. A Tabela 1 mostra as características comuns a ambas NoCs. A arquitetura com canais virtuais tem um *buffer* de dezesseis posições (*flits*) para cada canal lógico. A arquitetura com canais replicados tem um *buffer* de dezesseis posições (*flits*) para cada canal físico.

Tabela 1 – Características comuns a ambas NoCs.

Tamanho de Flit/phit	8 bits
Controle de fluxo	Baseado em créditos
Topologia da NoC	Malha 4x4
Algoritmo de roteamento	XY determinístico
Modo de chaveamento	pacotes/*wormhole*

A Figura 18 ilustra o cenário de tráfego usado para avaliar a latência e a vazão. Este cenário é justificado pela quantidade de fluxos concorrendo pelo mesmo caminho. Linhas indicam o caminho dos pacotes da origem até o destino. Elipses indicam os caminhos compartilhados por mais de um fluxo.

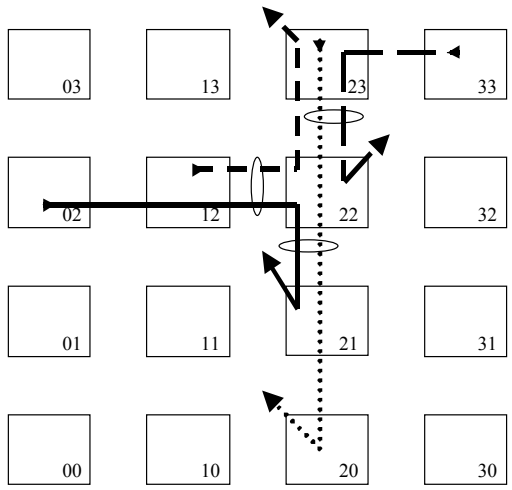


Figura 18 – Distribuição espacial de tráfego para a avaliação de latência e vazão, comparando canais virtuais com canais replicados.

Cada par origem-destino na Figura 18 transmite 500 pacotes de 257 *flits* (2 *flits* de cabeçalho e 255 de *payload*). A NoC com canais virtuais não possui o sinal *sideband eop* (Seção 2.2), por isso o tamanho máximo de pacotes suportado, considerando um tamanho de *flit* igual a 8 bits, é 257 *flits*. Esse sinal é uma característica incorporada juntamente com a replicação de canais físicos. Os pacotes são injetados na rede na taxa do canal. A Tabela 2 apresenta a latência média (em ciclos de *clock*) para transmitir um pacote e a vazão média por pacote. A latência apresentada inclui a latência da rede, proporcional ao número de *hops*, e a latência do pacote, proporcional ao tamanho do pacote. Ela é definida como o tempo entre a entrada do primeiro *flit* do pacote no roteador origem e a saída do último *flit* do pacote no roteador destino. A vazão é relativa à largura de banda do canal (%). Ela é definida como a razão entre o tamanho do pacote (257 bytes) e o tempo (em ciclos de *clock*) transcorrido entre a chegada (IP destino) do primeiro e do último *flit* do pacote, multiplicada por 100.

Tabela 2 – Valores médios de latência (ciclos de *clock*) e vazão.

Origem	Destino	Latência com Canais Virtuais	Vazão com Canais Virtuais	Latência com Canais Replicados	Vazão com Canais Replicados
02	21	580	50,8%	305	100%
12	23	546	51,2%	290	100%
33	22	556	51,8%	302	100%
23	20	570	51,2%	290	100%

Para o cenário da Figura 18, os canais replicados reduziram em média 47,3% a latência média e dobraram a vazão, quando comparado aos canais virtuais. Esse resultado é esperado, visto que a largura de banda interna da NoC foi duplicada. Quando não há congestionamento, ambas

abordagens apresentam valores de latência e vazão iguais.

A Tabela 3 apresenta a área de FPGA consumida. Para um único roteador de cinco portas, uma redução de área de 12% pode ser observada quando a replicação de canais físicos é utilizada. Para uma NoC malha 4x4 a redução é de 15%.

Tabela 3 – Resultados de área para Canais Virtuais (CV) e Canais Replicados (CR), usando o dispositivoVirtex 2VP30.

Recurso	Roteador de 5 portas		NoC malha 4x4		Disponibilidade
	CV	CR	CV	CR	
Slices	861	758	10538	8904	13696
LUTs	1722	1515	21075	17808	27392
Flip Flops	455	398	5866	5057	29060

A Tabela 4 apresenta o consumo de área para ASIC (0,35µm), considerando o número de *gates* equivalentes e "blocos de memória de 16x16 bits" para implementar os *buffers* de entrada. Para um único roteador e a NoC malha 4x4, uma redução de área de 4% e 6,4% respectivamente, é observada quando a replicação de canais físicos é utilizada.

Tabela 4 – Resultados de área para Canais Virtuais (CV) e Canais Replicados (CR), usando uma biblioteca ASIC.

Recurso	Roteador de 5 portas		Noc malha 4x4	
	CV	CR	CV	CR
Gates equivalentes	6709	6416	83952	78759
Blocos de memória de 16x16 bits	5	5	64	64

A redução de área obtida a partir da replicação dos canais físicos ao invés da multiplexação deve-se à eliminação dos demultiplexadores de entrada, multiplexadores de saída e a lógica de TDM responsável pelo controle destes. Os ganhos de área obtidos em FPGA são maiores porque os multiplexadores são implementados utilizando LUTs, enquanto que em ASIC eles são implementados a partir de portas lógicas.

2.5 Conclusões do Capítulo

Baseado na grande disponibilidade de área para conexões dentro dos chips, este capítulo propôs a replicação de canais físicos visando a redução de latência. Uma replicação de grau 2 foi aplicada a todos canais físicos (*North*, *South*, *East*, *West* e *Local*) do roteador HERMES, dobrando sua largura de banda total. Dispondo de uma alta largura de banda, o congestionamento na NoC reduz, consequentemente diminuindo a latência total.

Os resultados comparativos apresentaram ganhos significativos de desempenho e área, demonstrando a efetividade da proposta e apontando uma alternativa ao uso de canais virtuais. Canais virtuais foram introduzidos por Dally e Seitz [DAL87], visando resolver o problema de *deadlock* em redes que implementam *wormhole* e não visando desempenho, apesar de contribuírem para uma melhor utilização dos canais físicos. Essa técnica é uma herança de redes de

computadores, onde há uma limitação significativamente maior no número de conexões que ligam dois elementos que se comunicam. Neste caso a multiplexação do meio físico torna-se a técnica mais adequada. Até o presente momento, a proposta da replicação de canais físicos como alternativa ao uso de canais virtuais em topologias malha não foi apresentada por nenhum autor, podendo assim ser considerada como uma contribuição deste trabalho para a área de NoCs.

3 CHAVEAMENTO POR CIRCUITO E NÍVEL DE SESSÃO

A maioria das NoCs propostas empregam pilhas de protocolo semelhantes ao modelo de referência OSI [DEH06][DAY83]. Os três níveis mais baixos (físico, enlace e rede) são comumente implementados em hardware. O nível físico provê as definições elétricas do meio para conectar os roteadores entre si ou roteadores e elementos de processamento. O nível de enlace é responsável pelo transporte confiável dos pacotes entre elementos que se comunicam, a partir de estratégias de controle de fluxo como *handshake* ou baseado em créditos. O nível de rede é responsável pela determinação do caminho entre origem e destino (algoritmos de roteamento) e endereçamento lógico. O quarto nível, transporte/sessão, não é usualmente integrado em NoCs. Ele controla a conexão fim-a-fim, monta/desmonta mensagens e trata de erros fim-a-fim. A Tabela 5 apresenta uma comparação entre redes de computadores e NoCs, no que diz respeito a implementação dos níveis de protocolo.

Tabela 5 – Comparação de implementações dos níveis de protocolo [DEH06].

<table>
<tr><th rowspan="2">Níveis de protocolo</th><th colspan="2">Implementação</th></tr>
<tr><th>Redes de Computadores</th><th>NoCs</th></tr>
<tr><td>Aplicação</td><td>Software</td><td rowspan="3">Hardware/Software</td></tr>
<tr><td>Apresentação</td><td>Software</td></tr>
<tr><td>Sessão</td><td>Software</td></tr>
<tr><td>Transporte</td><td>Software</td><td rowspan="2">Hardware</td></tr>
<tr><td>Rede</td><td>Hardware/Software</td></tr>
<tr><td>Enlace</td><td>Hardware/Software</td><td>Hardware</td></tr>
<tr><td>Físico</td><td>Hardware</td><td>Hardware</td></tr>
</table>

Este capítulo propõe a adição de um nível de sessão sobre o chaveamento por circuito, incluindo o controle de sessões simultâneas por roteador. Visto que a NoC tem uma largura de banda significativamente maior que as taxas de transmissão das aplicações, foi desenvolvido um método de transmissão, através da associação do chaveamento por circuito associado ao nível de sessão, que maximiza a utilização dos recursos da rede. Considere por exemplo, um roteador de 16 bits operando a 200MHz: a largura de banda disponível por canal é 3,2Gbps. Em contrapartida, a taxa de transmissão de uma aplicação que requer uma alta largura de banda, como um fluxo HDTV, é 15 Mbps (MPEG2 – vídeo compactado). Essa diferença de taxas (injeção e transmissão), entre as aplicações e a NoC, justifica o método proposto.

A base do método proposto de transmissão é ajustar as taxas da aplicação às taxas dos canais da NoC. Isso é feito através do agrupamento dos dados em um *buffer* na origem e em seguida transmitindo-os pela NoC em rajada. A Figura 19 ilustra a transmissão de dados de uma aplicação a uma taxa inferior a do canal da NoC, e o correspondente agrupamento antes da

transmissão. Na abordagem proposta, as mensagens (dados da aplicação como pacote Ethernet ou bloco de cache) são divididas em pacotes de tamanho fixo chamados de células, semelhante a células ATM. O *buffer* na origem garante a transmissão de células na taxa do canal, evitando o tempo ocioso entre *flits* e maximizando o uso da largura de banda do canal.

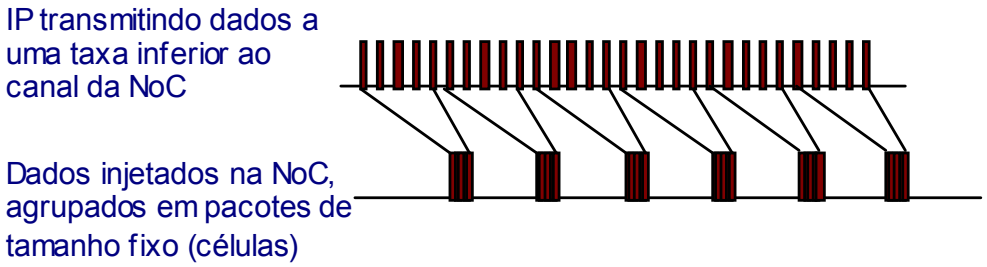


Figura 19 – Agrupamento dos dados em células para o ajuste das taxas entre NoC e aplicação.

3.1 Trabalhos Relacionados

O chaveamento por pacotes (baseado em *wormhole*) é o modo de chaveamento mais comumente empregado em NoCs [BJE06]. Ele freqüentemente proporciona alta largura de banda, visto que muitos pacotes podem estar sendo simultaneamente transmitidos em um dado instante de tempo [HIL06]. Entretanto, geralmente esse modo requer controle de congestionamento e processamento de pacotes, implicando *buffers* internos aos roteadores para o armazenamento temporário de pacotes que esperam pela disponibilidade de recursos de roteamento. O dimensionamento adequado desses *buffers* é um parâmetro fundamental para otimizar o desempenho da rede. *Buffers* rasos aumentam o congestionamento da rede (rápida saturação), enquanto *buffers* profundos aumentam o consumo em área de silício. Esse modo de chaveamento se adapta bem a serviços do tipo *best-effort* [JAN03] e canais virtuais, sendo mais adequado para tráfegos que podem se beneficiar de pacotes curtos e freqüentes. HERMES [MEL05], Xpipes [BEN04], MANGO [BJE05] e SoCIN [ZEF03] são exemplos de NoCs que empregam o chaveamento por pacotes (*wormhole*).

O chaveamento por circuito proporciona garantias de vazão e limites de latência, visto que uma caminho exclusivo é alocado para a transferência de dados entre os IPs origem e destino. Em relação ao armazenamento temporário (*buffers*), tipicamente é utilizado um registrador ao invés de uma fila, pois uma vez estabelecido o circuito, a NoC funciona como um *pipeline*. As desvantagens do chaveamento por circuito são a subutilização da largura de banda dos canais, quando o tráfego é transmitido a taxas baixas, e a latência para o estabelecimento do circuito, o qual depende do tráfego no caminho durante o estabelecimento do circuito. Esse modo de chaveamento é mais eficiente para o tráfego de pacotes longos com altas taxas de transmissão, que necessitem de garantias de vazão e latência. Algumas NoCs que implementam o chaveamento por circuito são: PNoC [HIL06], Æthereal [GOO05], SoCBUS [WIK03] e Octagon [KAR02]. A NoC Æthereal emprega o chaveamento por circuito somente para tráfegos com requisitos de *QoS*, enquanto que para os demais tráfegos o chaveamento por pacotes é empregado. A Tabela 6 resume as vantagens e

desvantagens dos chaveamentos por circuito e pacotes.

Tabela 6 – Vantagens e desvantagens dos modos de chaveamento apresentados (circuito x pacotes).

Modo de chaveamento	Vantagens	Desvantagens
Circuito	- Garantias de vazão e latência - Registradores ao invés de filas (FIFO *buffers*)	- Reserva de caminho estática - Possível perda de largura de banda
Pacotes (*wormhole*)	- Compartilha os recursos (canais físicos) da NoC, possibilitando vários fluxos simultâneos (canais virtuais)	- Durante tráfego intenso, os flits podem bloquear muitos roteadores - Perda de largura de banda quando a taxa de transmissão do iniciador do tráfego é menor que a do canal

A NoC SoCBUS apresenta um chaveamento por circuito híbrido chamado de *packet connected circuit* (PCC). Nesse chaveamento, um pacote de estabelecimento de conexão é enviado do IP origem até o IP destino utilizando chaveamento por pacotes. Os recursos do caminho pelo qual o pacote de estabelecimento de conexão percorre, em direção ao destino, vão sendo reservados. Ao atingir o IP destino, uma confirmação (*Ack*) é enviada de volta à origem (pelo mesmo caminho) e os recursos previamente reservados tornam-se então alocados. Quando a confirmação atinge o IP origem, a conexão está estabelecida e os dados podem ser transmitidos. Depois de encerrada a transmissão dos dados, o IP origem envia para o IP destino uma requisição de cancelamento e os recursos vão sendo desalocados na medida em que a requisição avança pelo caminho. Se o estabelecimento de conexão é bloqueado em algum roteador, esse roteador retorna para o IP origem uma não-confirmação (*nAck*). Ao receber a não-confirmação, o IP origem tenta estabelecer a conexão mais tarde. O protocolo para a transmissão de dados em é ilustrado na Figura 20. O estabelecimento de conexão utilizado no método proposto neste capítulo é semelhante a esse, a principal diferença está na geração do sinal de não-confirmação, o qual é gerado somente pelo roteador destino.

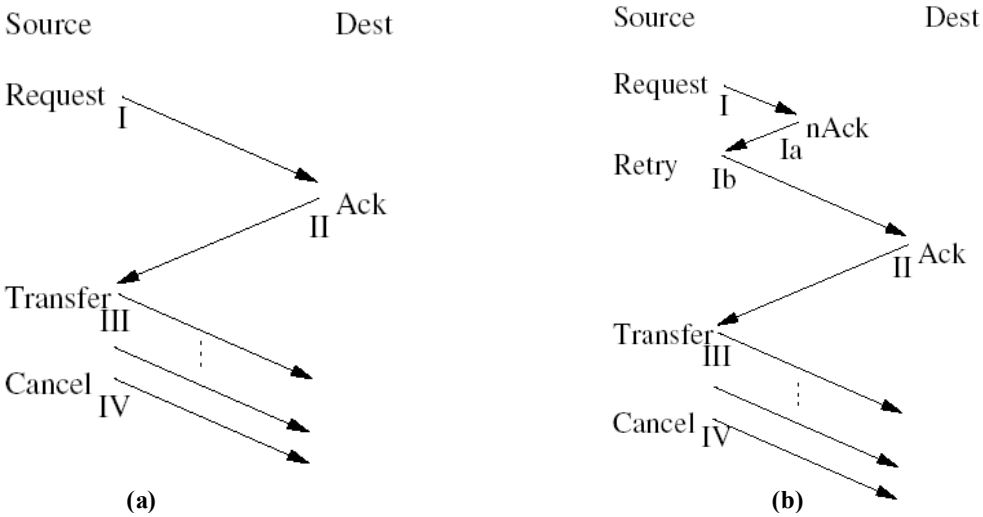


Figura 20 – Protocolo para a transmissão de dados na NoC SoCBUS. (a) estabelecimento de conexão em uma tentativa; (b) estabelecimento de conexão em duas tentativas [WIK03].

Na arquitetura de roteador apresentada em [NAR05], é proposta uma abordagem de chaveamento semelhante ao da NoC SoCBUS. Para o estabelecimento da conexão é utilizado um pacote chamado de *conection setup* (CS), o qual é enviado pelo IP origem ao IP destino com a finalidade de alocar um caminho para a comunicação. Ao receber o pacote CS, o IP destino responde ao IP origem com outro pacote CS, contendo a confirmação e parâmetros da conexão (unidirecional ou bidirecional, por exemplo). Se a conexão é bidirecional, então o pacote CS de resposta aloca o caminho no sentido contrário. Após receber o pacote CS de resposta, o IP origem pode iniciar a transmissão dos dados. Uma vez estabelecida a conexão entre origem e destino, os recursos alocados permanecem indisponíveis para outros fluxos até a conexão ser encerrada. Tipicamente a transferência de dados entre IPs é feita em rajadas. Conseqüentemente, durante o tempo ocioso entre as rajadas, os recursos alocados para a conexão são subtilizados. Para resolver essa questão o autor propõe uma nova técnica. Quando a conexão estiver ociosa por muito tempo, os recursos alocados tornam-se temporariamente disponíveis para outros fluxos, mesmo que a conexão não tenha sido encerrada. Se durante esse tempo uma nova rajada estiver pronta para ser transmitida, a transmissão dos outros fluxos é paralizada e a conexão volta a ser utilizada.

3.2 Descrição do Método de Transmissão e Arquitetura

A transmissão das mensagens pode ser orientada a conexão (chaveamento por circuito) ou sem conexão (chaveamento por pacotes). Usando o chaveamento por pacotes, as células podem ser bloqueadas dentro da rede, aumentando a latência. O maior benefício do uso de células surge quando o chaveamento por circuito é associado ao nível de sessão. O objetivo é reduzir o tempo de alocação de recursos internos a NoC, quando IPs transmitem dados a uma taxa menor que a suportada pelos canais. O chaveamento por circuito permite uma negociação entre origem e destino antes de iniciar a transmissão de dados. A partir de uma *conexão física* (circuito) entre origem e destino, é possível estabelecer uma *sessão* para então iniciar a transmissão de uma célula. O método proposto emprega as definições detalhas a seguir.

Definição 1: *Conexão física*. Corresponde ao estabelecimento de um circuito entre os IPs origem e destino, *para cada* célula da mensagem.

Definição 2: *Sessão*. Corresponde à reserva da porta local no roteador destino (IP destino) para todas as células oriundas do mesmo IP origem. A sessão é estabelecida pela primeira *conexão física* e é liberada pela última. Esta reserva é necessária para evitar a mistura de células recebidas de diferentes origens, o que impossibilitaria a remontagem da mensagem.

Para cada célula da mensagem, o IP origem envia ao IP destino um pacote de estabelecimento de conexão, o qual vai alocando os recursos para a *conexão física*. A posição da célula dentro da mensagem é indicada no pacote de estabelecimento de conexão. A célula pode ser:

(*i*) primeira; (*ii*) intermediária; (*iii*) última/única. O pacote de estabelecimento de conexão é composto por dois *flits* e tem o formato ilustrado na Figura 21, considerando-se um tamanho de *flit* igual a 8 bits.

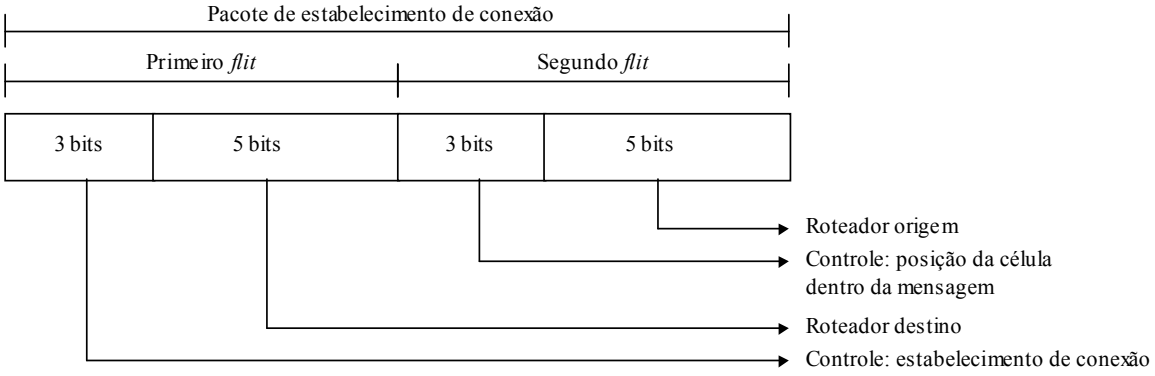


Figura 21 – Pacote de estabelecimento de conexão.

Quando o pacote de estabelecimento de conexão chega ao roteador destino, este propaga de volta para o IP origem um sinal de reconhecimento (*ack*), concluindo o estabelecimento da *conexão física*. Em seguida a célula é transmitida em rajada, um *flit* a cada ciclo de *clock*. A *conexão física* vai sendo desfeita conforme o último *flit* da célula corrente vai sendo transmitido, através do sinal *sideband eop* (Seção 2.2). Note que os pacotes de estabelecimento de conexão podem encontrar bloqueios, aumentando o tempo de estabelecimento da *conexão física*.

Uma *sessão pode ser* estabelecida quando o pacote de estabelecimento de conexão indicando primeira ou única célula chega ao roteador destino. Se a porta local do roteador destino não está reservada, o primeiro sinal de reconhecimento estabelece tanto a *conexão física* quanto a *sessão*. Se uma *sessão* já está estabelecida, então um sinal de não-reconhecimento (*nack*) é propagado de volta para o IP origem indicando que, mesmo existindo um caminho livre na rede, a porta local do roteador destino já está reservada para outro IP origem. O sinal de não-reconhecimento libera todos os recursos alocados entre os IPs origem e destino. Ao receber um sinal de não-reconhecimento, o IP origem tenta estabelecer a *sessão* novamente depois de um certo tempo. Nessa implementação, este tempo é igual ao número de ciclos para uma célula atravessar um canal com a rede a vazia, o que é igual ao número de *flits* da célula. A *sessão* permanece ativa até o recebimento da última célula da mensagem. O método para transmitir mensagens com tamanhos variados pode ser resumido como mostra a Figura 22. Os passos indicados na Figura 22 são descritos a seguir.

1. O IP origem envia um pacote de estabelecimento de conexão indicando primeira célula da mensagem.
2. O IP destino gera um sinal de reconhecimento (*ack*), indicando que o IP origem pode transmitir a célula. Nesse passo, a *conexão física* e a *sessão* foram estabelecidas.
3. Ao receber o sinal de reconhecimento, o IP origem começa a transmitir a célula em rajada. A conexão física é desfeita automaticamente pelo último *flit* da mensagem.

4. O IP origem envia um pacote de estabelecimento de conexão indicando agora que trata-se de uma célula intermediária da mensagem.
5. O IP destino gera um sinal de reconhecimento (*ack*), indicando que o IP origem pode transmitir a célula.
6. Igual ao passo 3.
7. O IP origem envia um pacote de estabelecimento de conexão indicando que a última célula da mensagem será transmitida e que a *sessão* pode ser encerrada ao fim da transmissão.
8. Igual ao passo 5.
9. Igual ao passo 3.

Os passos de 4 a 6 repetem-se várias vezes para mensagens com mais de três células. Para mensagens com duas células, aplicam-se somente os passos 1, 2, 3, 7, 8 e 9. No caso de mensagens curtas, com apenas uma célula, aplicam-se somente os passos 7, 8 e 9.

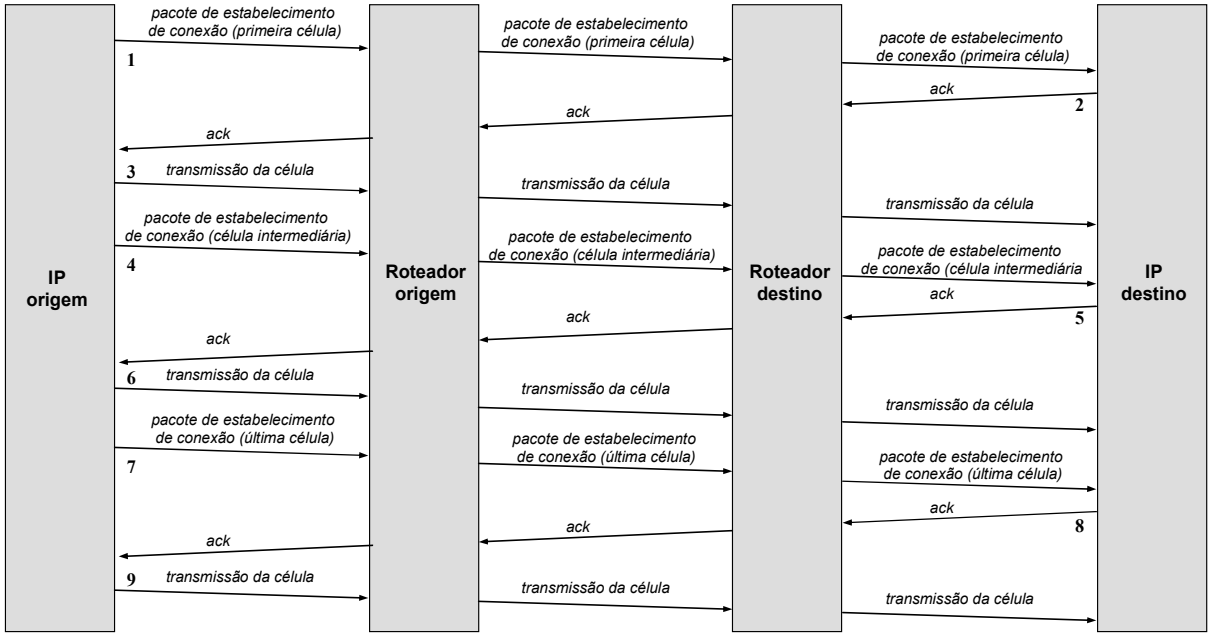


Figura 22 – Transmissão de mensagens.

O gargalo da abordagem proposta surge quando múltiplas origens tentam simultaneamente transmitir mensagens para um mesmo destino. A solução para esse problema é incluir no IP destino *buffers de sessão*. Cada *buffer de sessão* armazena células de uma mensagem, oriundas de uma mesma origem. A Figura 23 ilustra o uso de um *buffer* na origem e *buffers de sessão* no destino. Os *buffers de sessão* são colocados fora da rede (no lado do IP), na interface de rede ou dentro do próprio IP. Usando *buffers de sessão*, o IP destino pode receber *k* mensagens simultâneas, sendo *k* o número de *buffers de sessão*. Os *buffers de sessão* devem ser dimensionados para armazenar pelo menos uma mensagem de tamanho máximo.

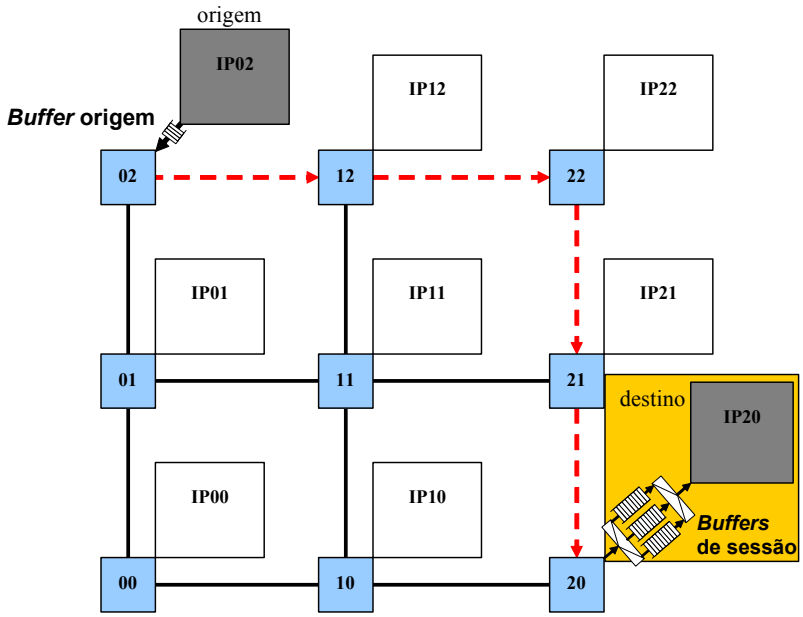


Figura 23 – Inclusão de *buffers* de sessão no IP20 para permitir múltiplas sessões por roteador.

O uso de múltiplas sessões simultâneas implica o uso de tabelas de sessão nos roteadores, as quais associam as sessões do roteador a IPs origem. Sempre que um roteador recebe um pacote de estabelecimento de conexão, a tabela de sessão é consultada para verificar se existe alguma sessão livre ou se já existe uma sessão alocada para a origem do pacote. A Figura 24a ilustra a interface entre a porta local do roteador e o IP, considerando os *buffers* de sessão colocados no lado do IP. Os sinais correspondentes à interface de saída do roteador são: (1) *data_out*: dado a ser transmitido; (2) *tx*: indica disponibilidade de dados no barramento *data_out*; (3) *eop_out*: indica que o dado no barramento *data_out* corresponde ao último *flit* do pacote/célula; (4) *session_out*: indica a qual sessão pertence a célula que está sendo transmitida em *data_out*. Esse sinal é utilizado para controlar o demultiplexador da Figura 23, ou seja, selecionar o *buffer* de sessão para o armazenamento da célula; (5) *ack_in*: indica que a conexão física com o IP destino foi estabelecida e o IP origem pode iniciar a transmissão da célula; (6) *credit_i*: indica que há espaço para o armazenamento de *flits* no *buffer* do IP.

A Figura 24b ilustra a interface entre roteadores vizinhos. Os sinais correspondentes à interface de saída são: (1) *data_out*: dado a ser transmitido; (2) *tx*: indica disponibilidade de dados no barramento *data_out*; (3) *eop_out*: indica que o dado no barramento *data_out* corresponde ao último *flit* do pacote/célula; (4) *ack_in*: propagação do sinal de reconhecimento (*ack*) gerado pelo *IP destino*. (5) *nack_in*: propagação do sinal de não-reconhecimento (*nack*) gerado pelo *roteador destino*; (6) *credit_i*: indica que há espaço para o armazenamento de *flits* no *buffer* do roteador vizinho. O sinal de não-reconhecimento é gerado pelo roteador destino porque ele mantém a tabela de sessões. Se ao receber um pacote de estabelecimento de conexão, o roteador verificar que não há sessão livre, o sinal de *nack* é gerado. A Figura 25 mostra o estabelecimento de *sessão* em duas tentativas, considerando que na primeira tentativa não havia uma *sessão* livre.

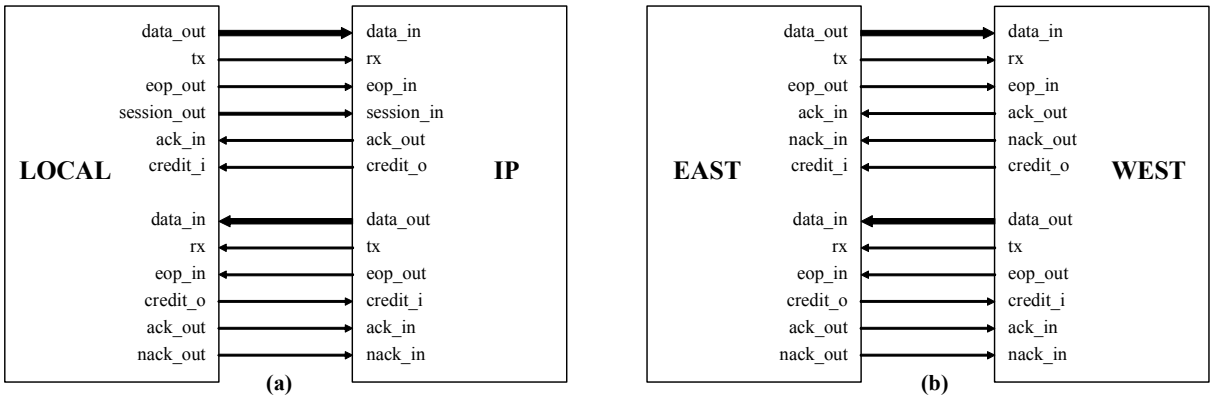


Figura 24 – (a) interface entre a porta local do roteador e o IP; (b) interface entre roteadores vizinhos.

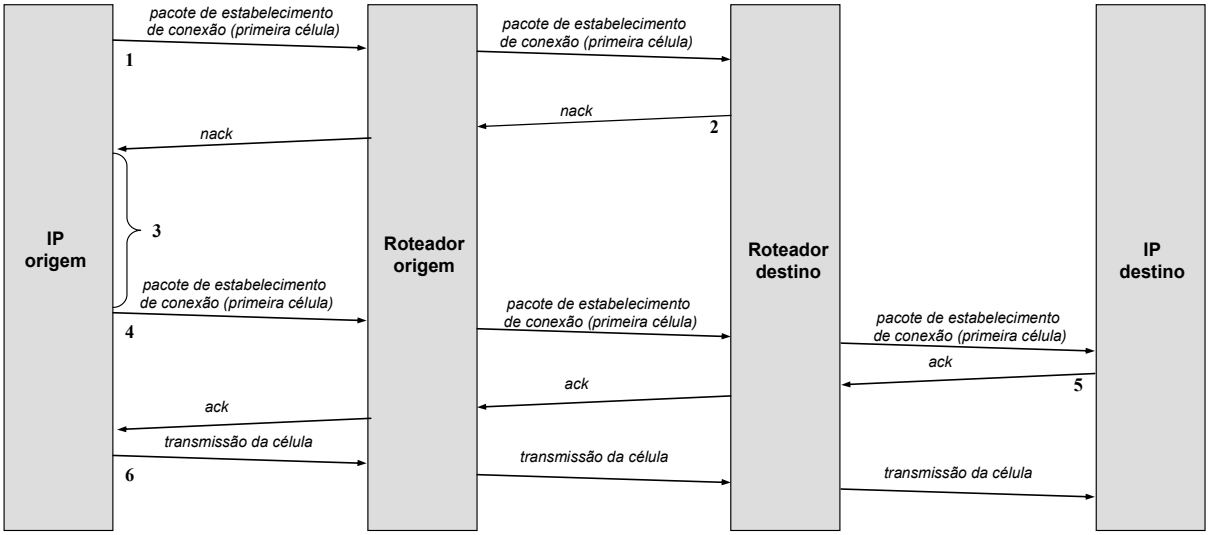


Figura 25 – Estabelecimento de *sessão* em duas tentativas.

1. O IP origem envia um pacote de estabelecimento de conexão indicando primeira célula da mensagem.
2. Ao consultar a tabela de sessões, o roteador destino verifica que não há nenhuma sessão livre e gera um sinal de não-reconhecimento (*nack*).
3. Ao receber o sinal de não-reconhecimento (*nack*), o IP origem espera um tempo e tenta estabelecer a conexão mais tarde.
4. O IP origem tenta estabelecer a *sessão* novamente.
5. Dessa vez havia uma *sessão* livre e o IP destino gera um sinal de reconhecimento (*ack*) estabelecendo a *conexão física* e a *sessão*.
6. Ao receber o sinal de reconhecimento, o IP origem começa a transmitir a célula em rajada.

O chaveamento por pacotes foi mantido, além do chaveamento por células, tornando a rede mais flexível. A partir do campo *controle* do *flit* de cabeçalho é possível especificar que o pacote deve ser transmitido usando chaveamento por pacotes. Pacotes de controle, como o pacote de estabelecimento conexão, podem ser transmitidos utilizando chaveamento por pacotes.

3.3 Validação do Chaveamento por Circuito e Controle de Sessão

O chaveamento por circuito e o controle de sessão foram descritos em VHDL e validados por simulação funcional e prototipação em FPGA. A Figura 26 ilustra um cenário no qual dois IPs enviam simultaneamente mensagens para um único IP destino. Nesta validação, o IP destino suporta até quatro sessões simultâneas.

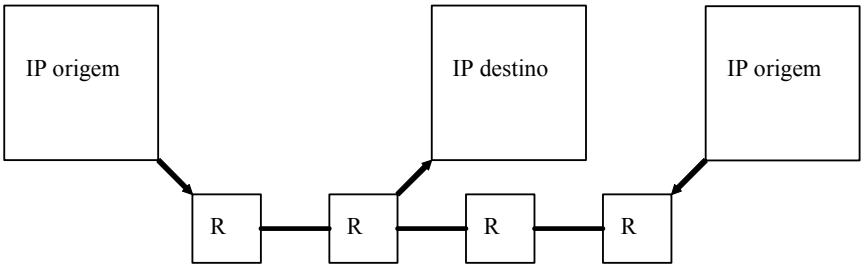


Figura 26 – Dois IPs enviando mensagens para o mesmo IP destino.

A Figura 27 e a Figura 28 apresentam a simulação funcional correspondente ao envio da primeira célula de cada IP origem. São mostradas as formas de onda correspondentes às interfaces de saída dos IPs origem e a interface de entrada do IP destino. A descrição dos passos da simulação segue às figuras.

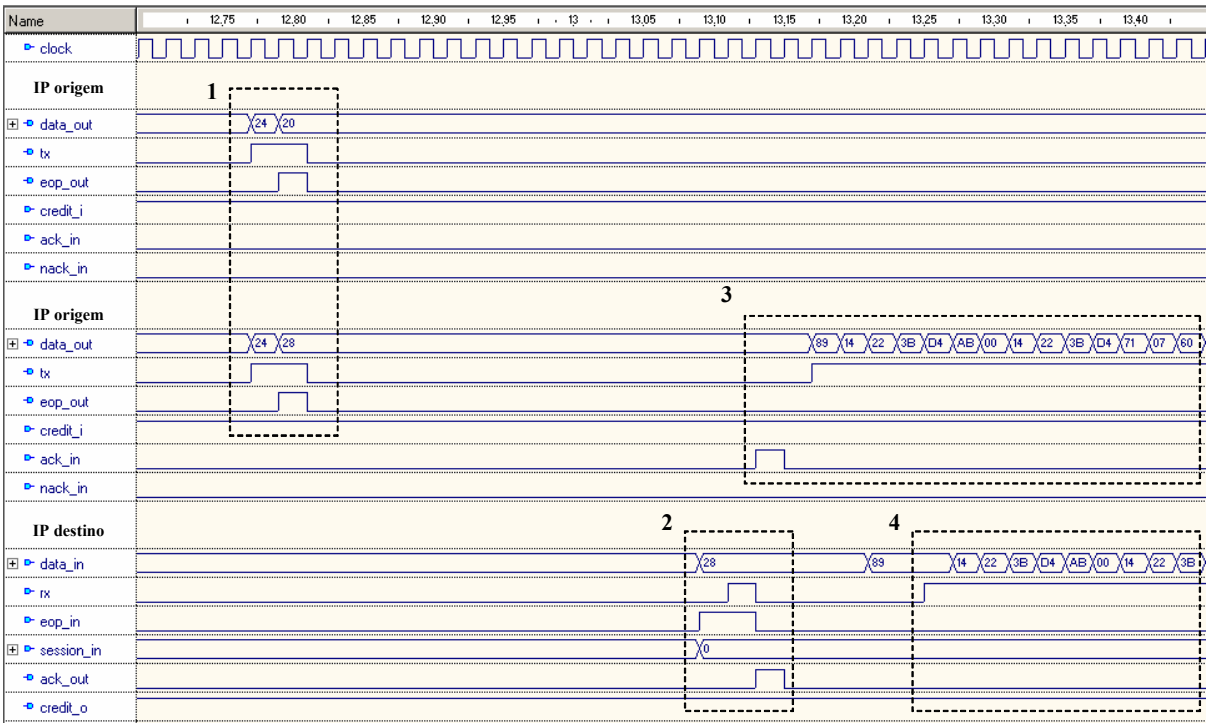


Figura 27 – Simulação funcional correspondente ao cenário da Figura 26 (parte 1).

1. Os IPs origem enviam o pacote de estabelecimento de conexão (Figura 21) para o mesmo IP destino.
2. O IP destino recebe o primeiro pacote de estabelecimento de conexão. Através do sinal *ack_out* é indicado que a *conexão física* e a *sessão* foram estabelecidas e o IP origem pode começar a

transmitir a célula. Observe que o IP destino recebe somente o segundo *flit* do pacote de estabelecimento de conexão. Isso ocorre porque o roteador destino remove o primeiro *flit* para poder analisar o segundo, o qual contém a origem do pacote. Em seguida ele consulta a tabela de sessão para verificar se existe uma sessão livre ou se já existe uma sessão alocada para a origem do pacote. No caso da simulação apresentada, todas as sessões estavam livres e a sessão 0 foi alocada (*session_in = 0*).

3. Ao receber o sinal de reconhecimento (*ack_in = '1'*), o IP origem começa a transmitir a célula em rajada. Observe que o sinal de reconhecimento foi recebido no mesmo ciclo de *clock* em que foi gerado pelo IP destino. Isso ocorre porque o sinal *ack* é transmitido combinacionalmente pela *conexão física*. A simulação apresentada não inclui o atraso das portas lógicas. No entanto, em simulações com atraso é possível verificar que esse atraso é inferior a um ciclo de *clock* (100MHz).
4. O IP destino começa a receber a célula.

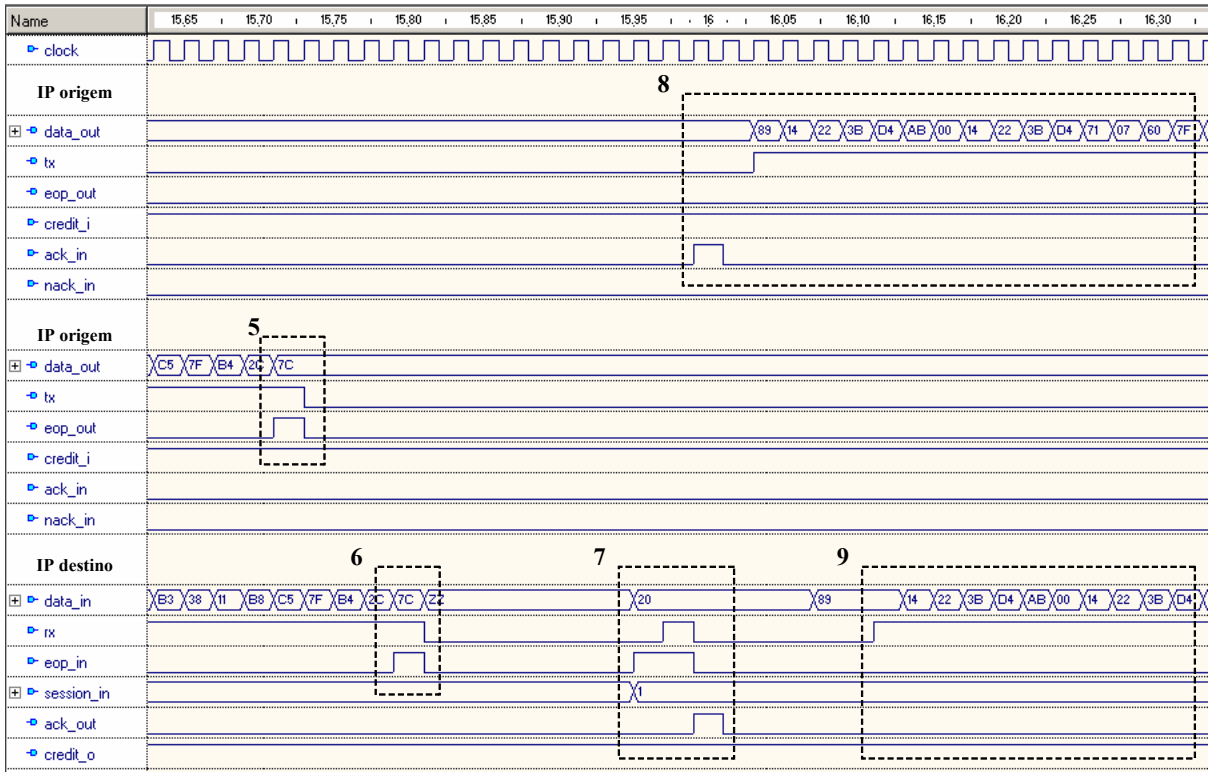


Figura 28 - Simulação funcional correspondente ao cenário da Figura 26 (parte 2).

5. O último *flit* da célula é enviado, desfazendo a *conexão física*.
6. O IP destino recebe o último *flit*. A sessão 0 continua ativa até que seja recebida a última célula da mensagem.
7. O IP destino recebe o segundo pacote de estabelecimento de conexão. Visto que a sessão 0 está ativa, a sessão 1 é alocada (*session_in = 1*). Através do sinal *ack_out* é indicado que a *conexão física* e a *sessão* foram estabelecidas e o IP origem pode começar a transmitir a célula.

8. Ao receber o sinal de reconhecimento (*ack_in = '1'*), o IP origem começa a transmitir a célula em rajada.
9. O IP destino começa a receber a célula.

3.4 Avaliação de Desempenho

Os experimentos apresentados a seguir avaliam os benefícios proporcionados pela adição de um nível de sessão ao chaveamento por circuito. A NoC utilizada tem as características apresentadas na Tabela 7, com *buffers* de entrada de dezesseis posições. Não são usados nem canais virtuais nem replicados.

Tabela 7 – Características da NoC utilizada.

Tamanho de Flit/phit	8 bits
Controle de fluxo	Baseado em créditos
Topologia da NoC	Malha 4x4
Algoritmo de roteamento	XY determinístico
Modo de chaveamento	Circuito

A Figura 29 ilustra um cenário no qual quatro origens (IP00, IP10, IP20 e IP30) enviam simultaneamente mensagens para o mesmo IP destino (IP03). As linhas indicam o fluxo das mensagens. Nesse cenário é possível observar a competição dos fluxos por recursos em comum. O objetivo desse primeiro experimento é comparar, em termos de desempenho, o método de transmissão proposto, variando o número de sessões de 2 a 4, com a transmissão das mensagens utilizando apenas o chaveamento por circuito. Os IPs origem geram dados (*flits*) a uma taxa de 20% da largura de banda do canal e os armazenam nos *buffers* origem. Cada IP origem envia 50 mensagens de 1280 bytes. Na transmissão utilizando apenas chaveamento por circuito, no momento em que o *buffer* origem armazena o primeiro *flit* da mensagem, o IP tenta estabelecer uma conexão física com o IP destino para transmitir a mensagem inteira (1280 bytes). Para cada mensagem a ser transmitida é estabelecida *uma* conexão física, a qual é desfeita pelo último *flit*. Na transmissão com controle de sessão, as mensagens são divididas em células de 128 bytes antes da transmissão. Para cada célula armazenada o IP tenta estabelecer uma conexão física com o IP destino para transmiti-la.

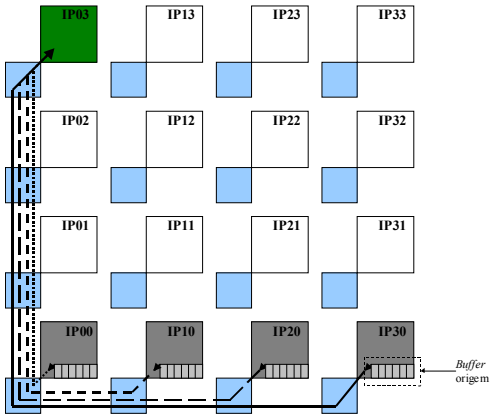


Figura 29 – Quatro IPs origem enviando simultaneamente mensagens para o mesmo IP destino.

A Tabela 8 apresenta a vazão média por mensagem, relativa a largura de banda da porta local da NoC, e o tempo total (em ciclos de *clock*) para os IPs transmitirem todas as mensagens. A vazão de uma mensagem é calculada como a razão entre o tamanho da mensagem (1280 bytes) e o tempo transcorrido entre a chegada (IP destino) do primeiro pacote de estabelecimento de conexão e o último *flit* da mensagem. Na transmissão com chaveamento por circuito, o IP00 consegue estabelecer a conexão física antes dos demais IPs. Por isso sua vazão média é próxima da taxa de geração de dados (20%), pois os *flit* são transmitidos na medida em que são armazenados no *buffer* origem. Enquanto o IP00 está transmitindo a uma taxa inferior à largura de banda do canal da NoC, os demais IPs não conseguem estabelecer a conexão física e os *flits* da mensagem vão sendo armazenados no *buffer* origem. Quando o IP00 termina de transmitir a mensagem, os outro IPs já estão com uma mensagem inteira (ou quase inteira) armazenada no *buffer* origem. Consequentemente, depois de estabelecerem a conexão física, a mensagem inteira é transmitida em rajada, tornando a vazão média dos demais IP próxima de 100%. Esse processo se repete para todas as mensagens, de maneira que o IP00 nunca está com uma mensagem inteira armazenada quando consegue estabelecer a conexão física. Na transmissão com chaveamento por circuito e controle de sessão, observa-se que esse efeito vai sendo reduzindo na medida em que o número de sessões aumenta. Com 4 sessões a vazão média de todos IPs se aproxima da taxa de geração de dados. Graças ao controle de sessão, os recursos da rede são compartilhados de maneira mais eficiente por fluxos concorrentes. Em relação ao tempo total de transmissão, observa-se um ganho de 25% utilizando 4 sessões, quando comparado com a transmissão utilizando somente o chaveamento por circuito.

Tabela 8 – Resultados de vazão média por mensagem e tempo total de transmissão.

	Origem	**Destino**	**Chaveamento por circuito**	**Chaveamento por circuito e Controle de sessão**		
				2 Sessões	**3 Sessões**	**4 Sessões**
Vazão média	IP00	IP03	25%	40%	22%	21%
	IP10	IP03	99%	48%	35%	21%
	IP20	IP03	98%	36%	29%	21%
	IP30	IP03	98%	60%	36%	28%
Tempo total de transmissão (ciclos de *clock*)			**454517**	**384185**	**356500**	**341048**

O tamanho de célula adotado na transmissão das mensagens é um parâmetro fundamental no desempenho total da rede. O próximo experimento avalia o desempenho em função do tamanho da célula e da taxa de injeção. A Figura 30 ilustra o cenário de tráfego utilizado para essa avaliação. Todos os fluxos têm pelo menos um fluxo competindo pelos mesmos recursos. Os seis IPs origem (IP00, IP10, IP20, IP21, IP30 e IP31) enviam mensagens de 1280 bytes. As linhas indicam o caminho das mensagens. Os IPs IP00 e IP10 iniciam a transmissão primeiro, introduzindo situações de bloqueio para os demais fluxos. A taxa de geração de dados dos IPs origem varia de 16,6% a 40% da largura de banda do canal na NoC.

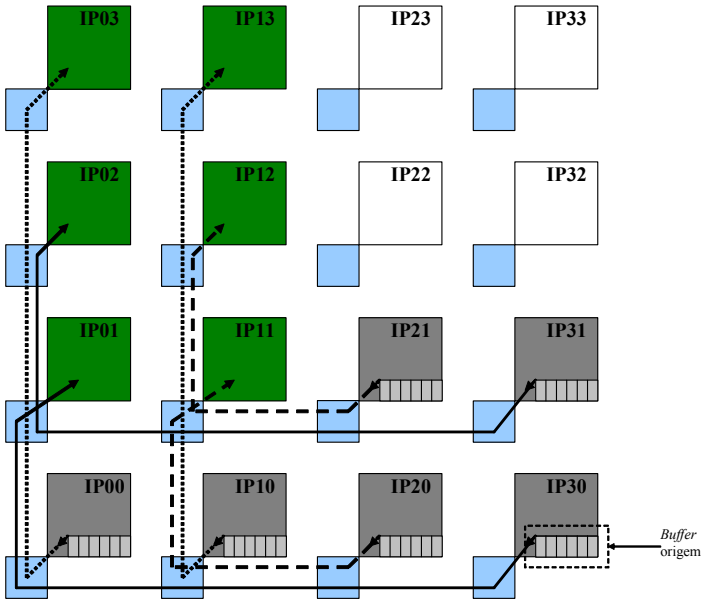


Figura 30 – Cenário para avaliação de desempenho em função do tamanho da célula e da taxa de injeção.

A Figura 31 mostra o número de ciclos de *clock* necessário para transmitir todas as mensagens em função do tamanho da célula (TC) e da taxa de injeção (TI). Para baixas taxas de injeção (16,6% e 20%) o tempo gasto na transmissão das mensagens cresce linearmente com o tamanho da célula. O tempo ocioso entre células em baixas TI favorece o compartilhamento do canal por diferentes fluxos e reduz o impacto do tempo para o estabelecimento da conexão. Já em taxas mais altas (33,2% e 40%), utilizando células pequenas (TC=32 a TC=96), o tempo ocioso entre células diminui por que elas ficam prontas para serem transmitidas mais cedo. Tão logo uma célula é transmitida já existe outra armazenada no *buffer* origem. Como conseqüência tem-se um aumento no congestionamento, pois existem muito pacotes tentando estabelecer conexão. O tempo gasto no estabelecimento de conexão em altas TI tem um alto impacto na latência quando células pequenas são usadas. Por exemplo, o tempo gasto para o estabelecimento de uma conexão passando por cinco roteadores é de aproximadamente 25 ciclos de *clock*, se não houver contenção. Uma célula de 32 *flits* é transmitida em 32 ciclos de *clock*. Consequentemente, cada célula pequena tem sua latência duplicada devido ao tempo gasto no estabelecimento de conexão. Conforme o tamanho da célula cresce (TC=128 a TC=256), o tempo ocioso entre células aumenta, permitindo o compartilhamento do canal em altas TI. No entanto, para células grandes, a largura de banda do canal é dominada por um único fluxo, aumentando novamente o tempo total para a transmissão das mensagens. Em TI intermediárias (25%), utilizando células pequenas (TC=32 e TC=64), o desempenho segue o mesmo padrão das altas TI, enquanto que utilizando células maiores, o desempenho segue o padrão das baixas TI. Esse experimento mostrou que um tamanho de célula intermediário, como 128 *flits* por exemplo, pode ser a melhor solução para diferentes taxas de injeção.

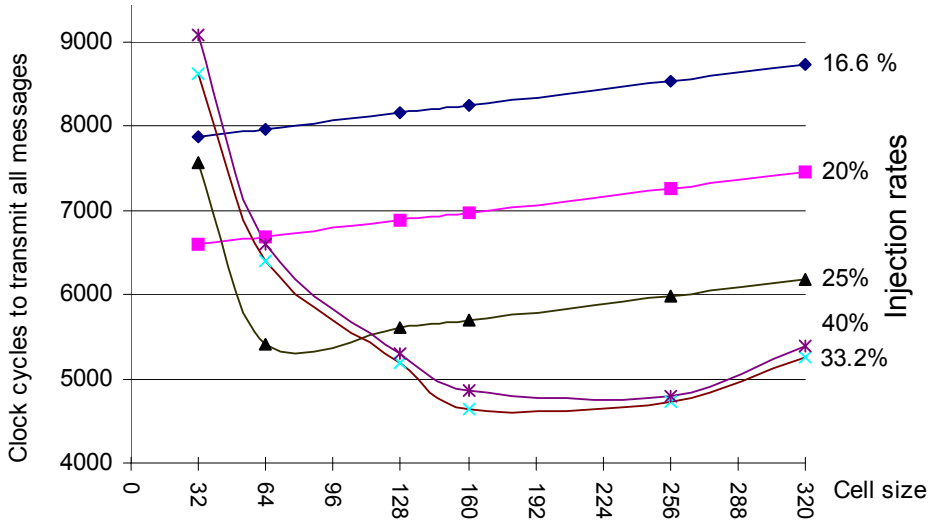


Figura 31 – Tempo total de transmissão em função do tamanho da célula e da taxa de injeção.

Existindo somente uma sessão por IP, o IP destino não necessita de um *buffer* de sessão. Entretanto, o custo para adicionar sessões simultâneas é um *buffer* de sessão por sessão, cada uma com capacidade para armazenar uma mensagem inteira de tamanho máximo.

Em SoCs baseados em NoCs, somente alguns poucos IPs recebem fluxos simultâneos. Por exemplo, em um MPSoC, memórias compartilhadas podem receber mensagens simultâneas de escrita, ou um IP de comunicação pode também receber pacotes Ethernet simultâneos para transmitir ao mundo exterior. Nessas situações, sessões simultâneas são uma solução para reduzir *hot spots* e a latência total.

3.5 Conclusões do Capítulo

Este capítulo propôs um método de transmissão de mensagens visando ajustar as taxas da aplicação às taxas dos canais da NoC. O método proposto baseia-se em armazenamento e transmissão em rajada, maximizando a utilização dos recursos da NoC e permitindo um eficiente compartilhamento destes por diferentes fluxos. Os resultados mostraram ganho de desempenho, em relação à transmissão utilizando apenas chaveamento por circuito, e o impacto do tamanho de célula utilizado no desempenho geral.

O nível de sessão compartilha os canais físicos de maneira similar a canais virtuais. A principal diferença está no nível de abstração. Canais virtuais compartilham os canais físicos no nível de pacotes, enquanto o controle de sessão compartilha os canais físicos no nível de fluxo. Essa técnica também pode ser usada para reduzir *hot spots*, visto que ela permite que os IPs mantenham várias sessões simultâneas.

4 ADAPTATIVIDADE DO ROTEAMENTO EM FUNÇÃO DO TRÁFEGO

Os algoritmos de roteamento podem ser em geral classificados como determinísticos ou adaptativos [NI93]. O roteamento determinístico fornece sempre o mesmo caminho entre um determinado par origem-destino, pois, por definição, o caminho de mensagens é definido unicamente pelas posições relativas de origem e destino da mensagem. Por outro lado, no roteamento adaptativo, o caminho é definido em função do estado instantâneo da rede, incluindo pontos de congestionamento, falhas nos enlaces e/ou roteadores, etc. A principal vantagem do roteamento determinístico é a sua simplicidade, além de prover baixa latência quando a rede não está congestionada. Em contrapartida, o roteamento adaptativo evita canais congestionados usando caminhos alternativos, adequando-se melhor a redes com tráfego intenso.

Este capítulo propõe uma estratégia que pode ser aplicada a algoritmos de roteamento adaptativos, tendo por princípio que as decisões de roteamento levem em consideração as condições do tráfego local da rede (nível de congestionamento dos roteadores vizinhos). Como estudo de caso é apresentado o algoritmo de roteamento Hamiltoniano, o qual é utilizado para dar suporte a uma eficiente família de algoritmos de *multicast* para multicomputadores com topologia malha bidimensional [LIN94]. Em vista de manter a compatibilidade no roteamento dos diferentes tipos de tráfegos, o algoritmo Hamiltoniano é utilizado tanto para o roteamento de mensagens *unicast* quanto para *multicast*. O estudo de caso apresentado envolve apenas tráfego *unicast*. Esse algoritmo é classificado como determinístico e é livre de *deadlock* quando aplicado a tráfegos *unicast*. O objetivo deste capítulo é relaxar o algoritmo de roteamento Hamiltoniano determinístico de maneira a torná-lo parcialmente adaptativo em função do tráfego. Mais precisamente, cada roteador monitora o nível de congestionamento dos roteadores vizinhos e toma decisões baseadas nessas informações. Este grau de adaptatividade visa evitar canais congestionados através da exploração de caminhos alternativos, conseqüentemente aumentando a vazão da rede.

4.1 Trabalhos Relacionados

Em [HU04] é apresentado o roteamento *DyAD* (*Dynamic Adaptive Deterministic*), que combina as vantagens dos roteamentos determinístico e adaptativo mínimos. As decisões de roteamento levam em consideração a carga dos roteadores vizinhos. Quando a rede não está congestionada, o roteamento *DyAD* trabalha no modo determinístico, caso contrário o modo adaptativo é ativado. O algoritmo de roteamento adaptativo utilizado é o *odd-even* [CHI00], para topologias malha 2-D. Ele proíbe as mudanças de direção *East*→*North* e *East*→*South* em colunas pares, e *North*→*West* e *South*→*West* em colunas ímpares. O algoritmo de roteamento determinístico usado é uma variação do *odd-even* chamada de *oe-fixed*. Esse algoritmo foi criado retirando-se a adaptatividade do *odd-even*, ou seja, se para um determinado destino o *odd-even* poderia rotear o pacote para as portas *p1* ou *p2*, no *oe-fixed* ele só roteia para *p1*. A Figura 32 ilustra

a arquitetura do roteador *DyAD-OE*.

Baseado em *flags* de congestionamento (*Congestion Flag*) recebidas dos roteadores vizinhos, o módulo *Mode Controller* indica para os módulos *Port Controller* qual roteamento deve ser usado. Se alguma das *flags* estiver ativa, todos *Port Controllers* passam a operar no modo adaptativo, caso contrário o modo determinístico é utilizado. Dentre as possíveis portas para as quais o pacote pode ser repassado, quando o modo adaptativo é usado, é selecionada aquela que corresponde ao *buffer* vizinho com mais posições livres. Uma vez selecionada a porta de saída pelo *Port Controller*, este faz uma requisição ao *Crossbar Arbiter*, o qual utiliza uma política *first-come-first-served* para selecionar qual porta de entrada será servida. Observe também que os módulos *Port Controller* monitoram a ocupação dos *buffers* e enviam *flags* de congestionamento para os roteadores vizinhos.

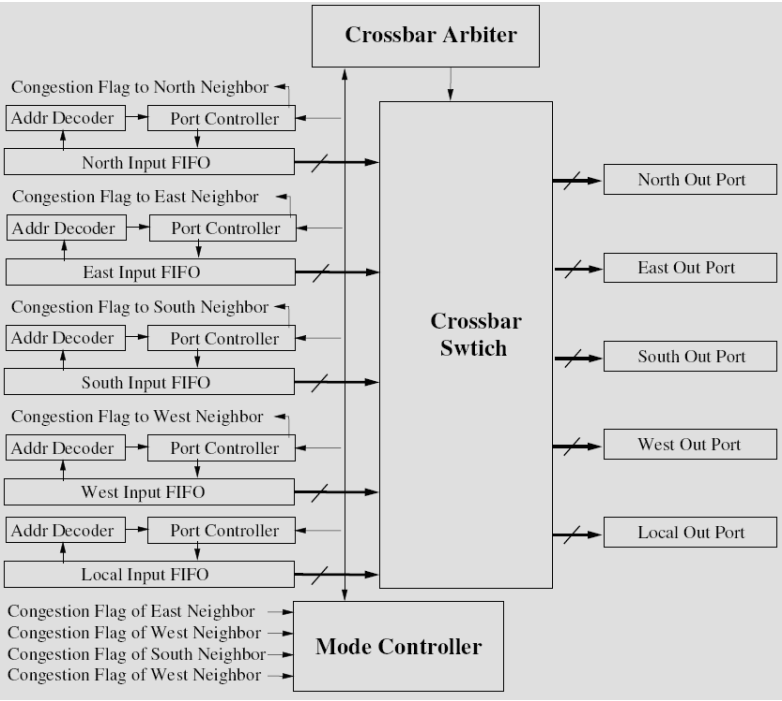


Figura 32 – Arquitetura do roteador *DyAD-OE* [HU04].

Em [LI06] é proposto o *DyXY* (*Dynamic XY*) baseado no algoritmo de roteamento XY. Ele proporciona adaptatividade baseado nas condições de congestionamento dos roteadores vizinhos. O algoritmo garante ser livre de *deadlock* e *livelock* proporcionando sempre um dos caminhos mínimos possíveis entre um par origem-destino qualquer. Enquanto o pacote não estiver alinhado em um dos eixos (X ou Y) com o roteador destino, o próximo roteador vizinho (em direção ao destino) é selecionado baseado no *stress value*. O *stress value* é o parâmetro que representa a condição de congestionamento do roteador. O seu valor corresponde ao número de posições ocupadas em todos os *buffers* de entrada do roteador. Cada roteador armazena os *stress values* de todos os vizinhos, os quais são atualizados baseado em um mecanismo dirigido a eventos. A Figura 33a ilustra a interconexão entre roteadores vizinhos e a Figura 33b ilustra a arquitetura do roteador.

A arquitetura do roteador é composta pelos módulos: (*i*) *Input buffers*; (*ii*) *Input arbiter*; (*iii*) *History buffer*; (*iv*) *Crossbar Switch*; (*v*) *Controller* e (*vi*) *Stress value counters*. A cada ciclo de *clock*, o módulo *History buffer* armazena as requisições das portas de entrada, as quais são selecionadas pelo módulo *Input arbiter*. O roteamento é realizado pelo módulo *Controller* baseado nos *stress values* vizinhos armazenados no módulo *Stress value counters*. Ele também é responsável por distribuir o *stress value* atual do roteador para os roteadores vizinhos.

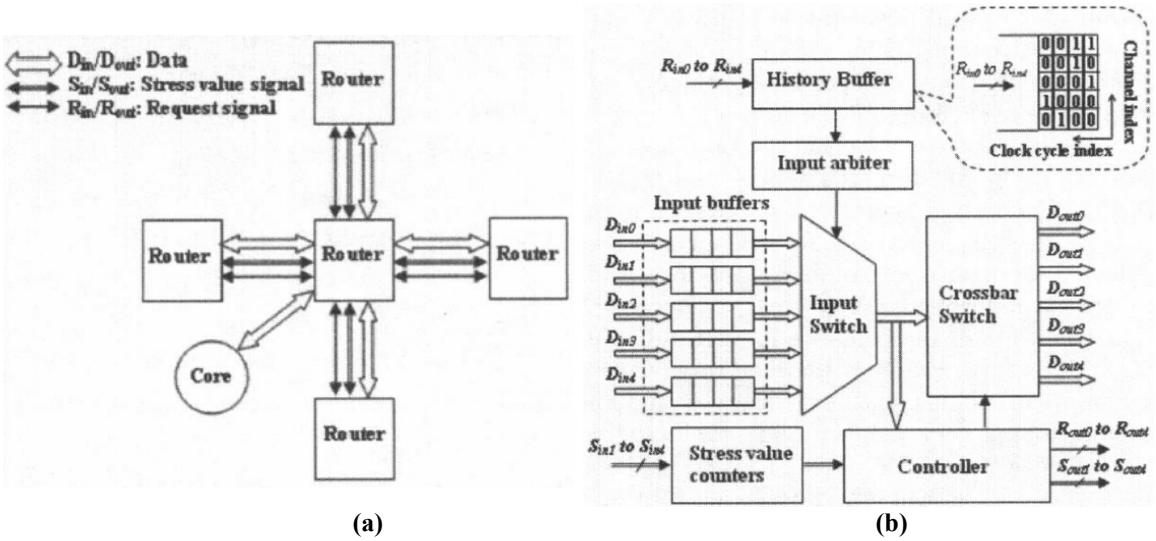


Figura 33 – (a) interconexão entre roteadores vizinhos; (b) arquitetura do roteador [LI06].

Em [SOB06] é apresentado um roteamento dinâmico, praticamente igual ao *DyAD*, diferindo apenas nos tipos de roteamento usados. Para a seleção da porta de saída são utilizados dois algoritmos de roteamento adaptativo, um mínimo e um não-mínimo. Quando um congestionamento se forma nas vizinhanças do roteador, o roteamento não-mínimo é ativado, caso contrário o roteamento mínimo é usado.

Uma extensão da proposta de [SOB06] é apresentada em [DAN06]. A principal diferença em relação ao *DyAD* está na arbitragem da porta de entrada, que, ao invés de usar a política *first-come-first-served*, essa leva em consideração o congestionamento dos roteadores vizinhos. A Figura 34 ilustra a arquitetura do roteador. Observe que a arquitetura é semelhante a do roteador *DyAD* (Figura 32). A principal diferença está no módulo *Crossbar Arbiter*, o qual recebe sinais que indicam estado de congestionamento (*CS – Congestion Status*) dos roteadores vizinhos e usa essa informação como prioridade para selecionar uma porta de entrada. A porta de entrada correspondente ao vizinho mais congestionado (maior prioridade) é a selecionada. O estado de congestionamento de cada roteador é calculado pelo módulo *CARS* (*Contention Aware Routing Selection*), que nada mais é do que um indicativo de quantos *buffers* congestionados o roteador tem. Por exemplo, se dois *buffers* de entrada estiverem com seus *flags* de congestionamento (*Congestion Flag*) ativos, o estado de congestionamento será dois. Apesar do autor não comentar, essa arbitragem parece suscetível a ocorrência de *starvation*, visto que um mecanismo de prioridades é empregado.

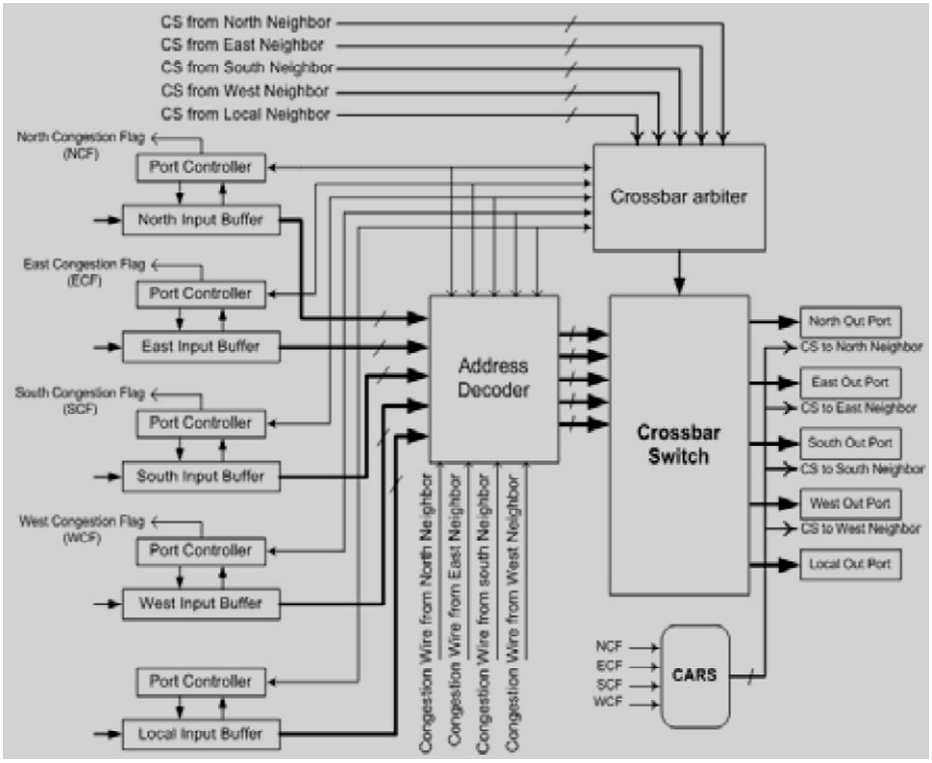


Figura 34 – Arquitetura do roteador [DAN06].

4.2 Descrição do Algoritmo Hamiltoniano e Arquitetura

Um caminho Hamiltoniano é um caminho acíclico no qual é possível atingir todos os nodos de um grafo passando apenas uma vez em cada nodo [HAR72]. Uma malha bidimensional contém vários caminhos Hamiltonianos. Pode-se definir um caminho Hamiltoniano sobre uma rede malha rotulando os roteadores de *0* a *N-1*, sendo *N* o número de roteadores. A Figura 35 ilustra um exemplo de caminho Hamiltoniano definido sobre uma rede malha 4x4.

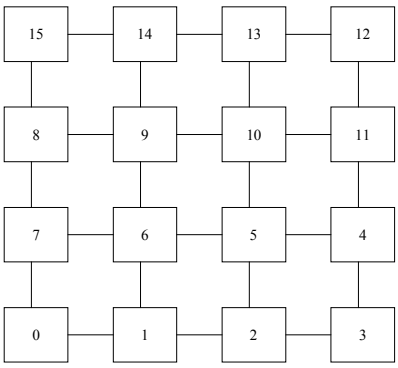


Figura 35 – Caminho Hamiltoniano definido sobre uma rede malha 4x4.

A partir da atribuição dos rótulos aos roteadores, a rede pode ser dividida em duas sub-redes disjuntas e acíclicas. Uma sub-rede contém os canais que vão de um rótulo menor para um maior (sub-rede dos canais maiores) e a outra contém os canais que vão de um rótulo maior para um menor (sub-rede dos canais menores) As Figura 36 ilustra a divisão da rede em duas sub-redes.

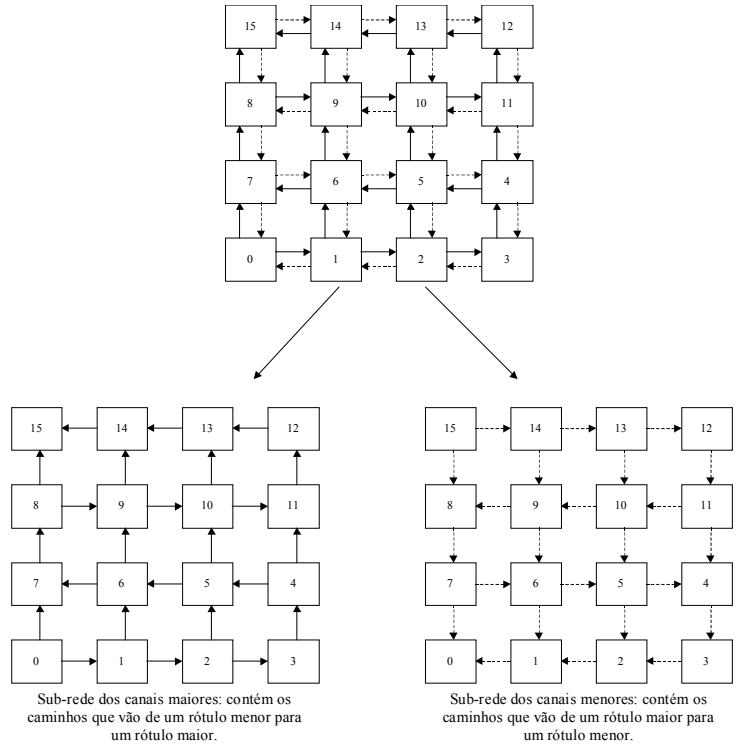


Figura 36 – Divisão da rede em duas sub-redes disjuntas e acíclicas.

A partir do caminho Hamiltoniano, um algoritmo de roteamento determinístico pode ser definido de maneira que, um pacote em um roteador com rótulo menor que o destino, é encaminhado para *o roteador vizinho com o maior rótulo*, o qual seja *menor ou igual* ao destino (segue pela sub-rede dos canais maiores). A Figura 37 ilustra dois exemplos do algoritmo roteamento Hamiltoniano determinístico. Sejam 7 o roteador origem do pacote e 13 o destino, o caminho seguido é: $7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 13$ (Figura 37a). De maneira semelhante, quando um pacote está em um roteador com rótulo maior que o destino, ele é encaminhado para *o roteador vizinho com menor rótulo*, o qual seja *maior ou igual* ao destino (segue pela sub-rede dos canais menores). Sejam 13 o roteador origem do pacote e 0 o destino, o caminho seguido é: $13 \rightarrow 10 \rightarrow 5 \rightarrow 2 \rightarrow 1 \rightarrow 0$ (Figura 37b).

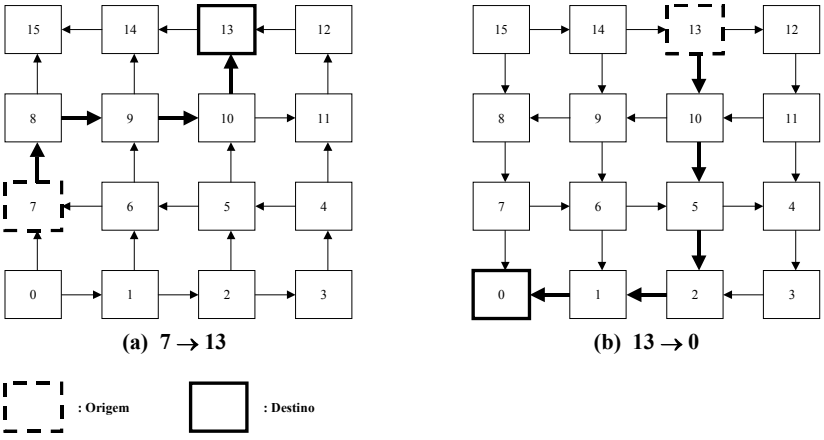


Figura 37 – Exemplos do roteamento Hamiltoniano determinístico.

Esse algoritmo de roteamento é livre de *deadlock* graças à propriedade acíclica dos caminhos Hamiltonianos e a disjunção das sub-redes. Também é livre de *livelock*, pois o rótulo do próximo roteador sempre está entre o atual e o destino. Para o caminho Hamiltoniano definido na Figura 35, o algoritmo definido é mínimo, pois cada vez que o pacote é repassado para um roteador vizinho ele se aproxima do destino. No entanto, para outros caminhos Hamiltonianos essa propriedade pode não ser válida. A Figura 38 ilustra um caminho Hamiltoniano alternativo (rotulação de roteadores alternativa), no qual o algoritmo de roteamento definido não é mínimo. Sejam 2 o roteador origem e 10 o destino, caminho seguido é $2 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10$ ao invés de $2 \rightarrow 3 \rightarrow 10$.

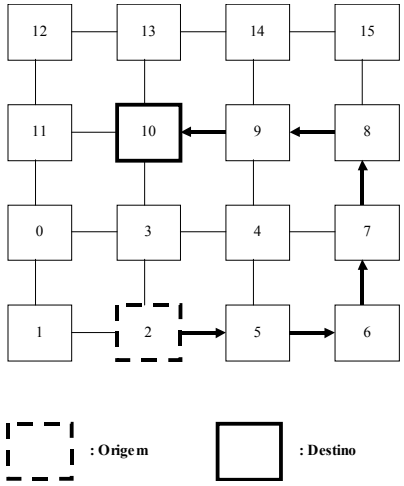


Figura 38 – Caminho Hamiltoniano alternativo.

Através do relaxamento do algoritmo determinístico pode-se criar uma versão parcialmente adaptativa, definindo-se que o pacote deve ser encaminhado para *um dos roteadores vizinhos maiores ou menores*, dependendo da posição da origem em relação ao destino. A partir desse relaxamento, o algoritmo de roteamento perde a propriedade mínima e muda sua classificação de determinístico para adaptativo. Para torná-lo adaptativo em função do tráfego, o critério para a escolha do próximo roteador vizinho, dentre os possíveis, baseia-se no congestionamento dos *buffers* de entrada dos roteadores vizinhos. A Figura 39 ilustra dois exemplos do algoritmo roteamento Hamiltoniano parcialmente adaptativo em função do tráfego. Na Figura 39a, o roteador 10 está com o *buffer South* congestionado. Sejam 5 o roteador origem do pacote e 14 o destino, o caminho seguido é $5 \rightarrow 6 \rightarrow 9 \rightarrow 14$ ao invés de $5 \rightarrow 10 \rightarrow 13 \rightarrow 14$ (determinístico). Na Figura 39b, os roteadores 5 e 6 estão com o *buffer North* congestionado. Sejam 10 o roteador origem do pacote e 0 o destino, o caminho seguido é $10 \rightarrow 9 \rightarrow 8 \rightarrow 7 \rightarrow 0$ ao invés de $10 \rightarrow 5 \rightarrow 2 \rightarrow 1 \rightarrow 0$ (determinístico).

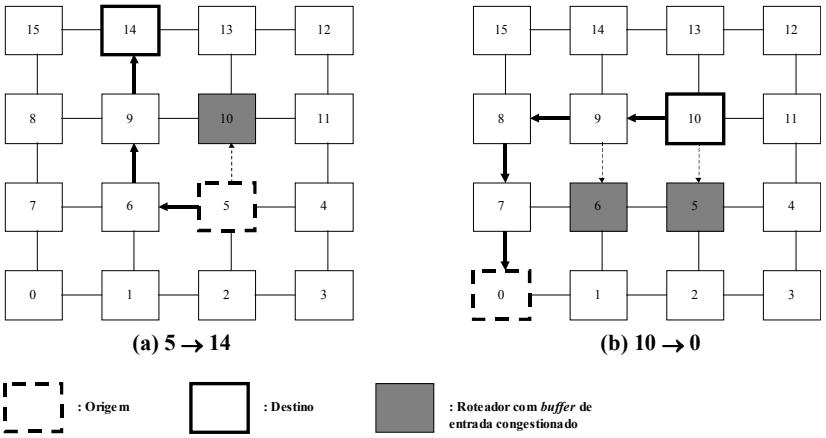


Figura 39 - Exemplos do roteamento Hamiltoniano parcialmente adaptativo em função do tráfego.

A sinalização de congestionamento é enviada para os roteadores vizinhos através do sinal *sideband cong*. Para cada *buffer* do roteador existe um sinal *cong* que indica seu estado, o qual é enviado para o vizinho correspondente. A Figura 40a ilustra um resumo das interfaces entre roteadores vizinhos, onde o sinal *cong* é enviado. Na Figura 40a, todos os sinais são mostrados unidirecionais. Na prática, eles existem em ambas direções como mostra a Figura 40b. Observe por exemplo o caso do roteador 6. O sinal *cong* correspondente ao *buffer South* é enviado para o roteador 5, enquanto que o sinal *cong* correspondente ao *buffer East* é enviado para o roteador 7. A partir da análise desses sinais durante o roteamento, o próximo roteador vizinho é selecionado. Se ambos roteadores vizinhos retornados pelo algoritmo de roteamento estiverem com o sinal *cong* ativo, é selecionado o maior ou o menor, dependendo da posição da origem em relação ao destino. O sinal *cong* é ativado a partir do momento em que um limiar de ocupação do *buffer* é atingido. Utilizando-se mais bits na implementação do sinal *cong*, é possível indicar diferentes níveis de congestionamento ou até mesmo o número de posições ocupadas do *buffer*.

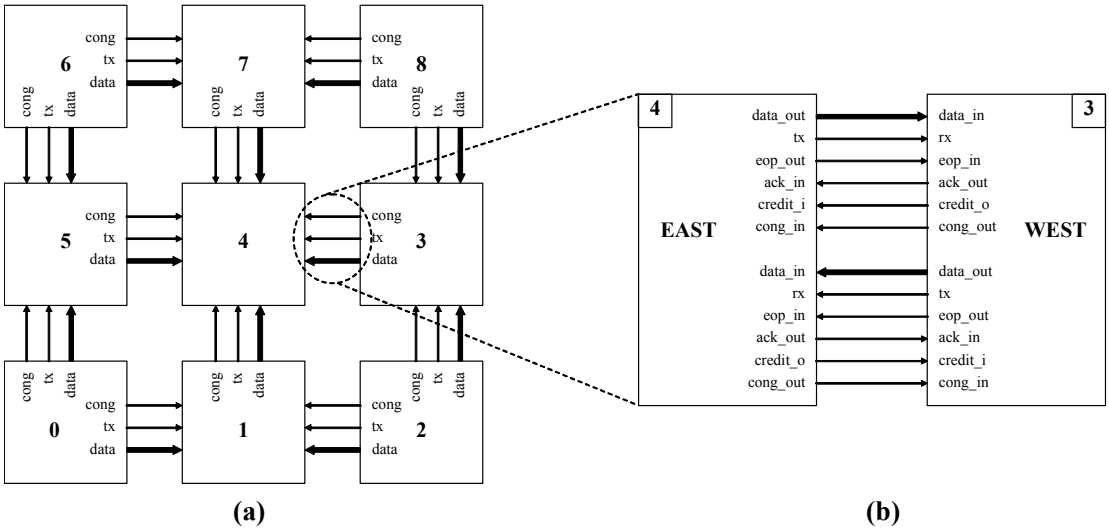


Figura 40 - Envio do sinal *cong* para os roteadores vizinhos.

4.3 Avaliação de Desempenho

Os resultados apresentados nessa seção visam mostrar o ganho de desempenho obtido a partir da adição da adaptatividade ao algoritmo de roteamento Hamiltoniano. Quatro versões do algoritmo foram comparadas: (*i*) determinística (Ham. Deter.); (*ii*) puramente adaptativa (Ham. Adap.), na qual o critério para a escolha do próximo roteador vizinho é a primeira porta de saída livre; (*iii*) adaptativa em função do tráfego versão 1 (Ham. Cong. v1), com o sinal *cong* (1 bit) indicando se o *buffer* está congestionado ou não e (*iv*) adaptativa em função do tráfego versão 2 (Ham. Cong. v2), com o sinal *cong* (4 bits) indicando o número de posições ocupadas no *buffer*. O algoritmo XY determinístico é utilizado como referência. A NoC utilizada tem as características apresentadas na Tabela 9. O tamanho de *buffer* utilizado é 16 posições e o sinal *cong* é ativado a partir de 11 posições ocupadas. Não são usados canais virtuais nem replicados.

Tabela 9 – Características da NoC utilizada.

Freqüência de operação	50MHz
Tamanho de Flit/phit	8 bits
Controle de fluxo	Baseado em créditos
Topologia da NoC	Malha 4x4
Algoritmo de roteamento	Hamiltoniano
Modo de chaveamento	Pacotes/*Wormhole*

O padrão de tráfego utilizado foi o complemento, visando obter situações de congestionamento para a ativação da adaptatividade. Cada IP envia 200 pacotes de 32 *flits*. O experimento foi repetido para taxas de injeção variando de 50 Mbps a 400 Mbps (taxa máxima dos canais da rede operando a 50MHz e com largura de *phit* igual a 8 bits). A Figura 41 apresenta os resultados obtidos em termos de tempo total para transmissão de todos os pacotes.

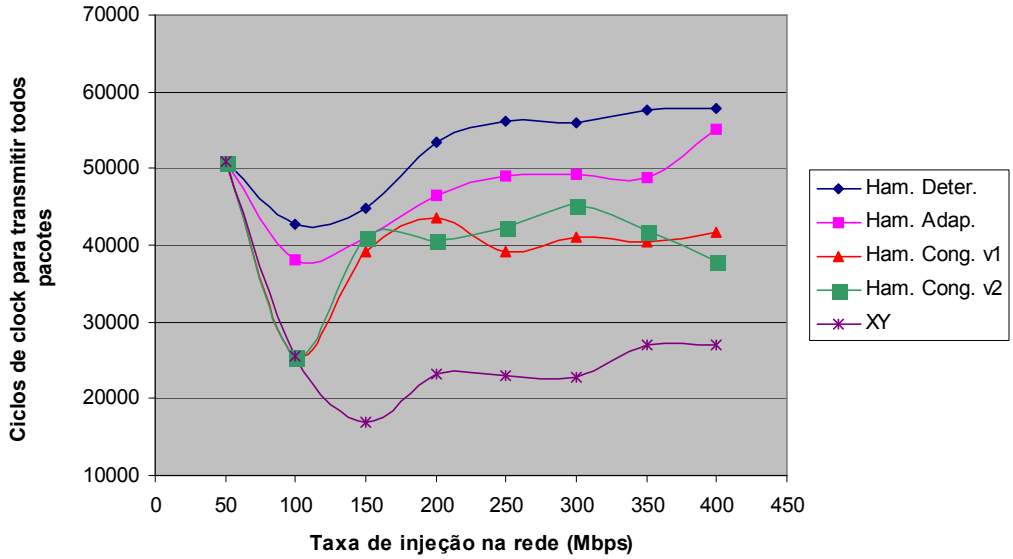


Figura 41 – Comparação das quatro versões do roteamento Hamiltoniano.

Como mostra o gráfico, em taxas de injeção baixas (50 Mbps) as quatro versões apresentam o mesmo desempenho, pois não há congestionamento. A partir do crescimento da taxa de injeção, as versões adaptativas passam a apresentar um desempenho superior à determinística. As versões adaptativas em função do tráfego mantêm-se com melhor desempenho porque tomam decisões baseadas em indicativos de congestionamento, procurando distribuir o tráfego na rede e evitando *hot spots*. A partir da taxa de injeção igual a 150Mbps (37,5% da taxa máxima dos canais da rede), o algoritmo XY passa a apresentar um desempenho significativamente superior às versões adaptativas em função do tráfego.

O algoritmo de roteamento Hamiltoniano possui uma limitação em relação ao XY, a qual é responsável pelo seu desempenho inferior. Como explicado anteriormente (Seção 4.2, Figura 36), o algoritmo Hamiltoniano divide a rede em duas sub-redes disjuntas, de maneira que a sub-rede dos canais maiores abrange metade dos canais físicos e a sub-rede dos canais menores abrange a outra metade. Dessa maneira, os fluxos que circulam em uma determinada sub-rede têm uma limitação em relação ao número máximo de canais físicos que podem ser alocados. Essa limitação não existe no algoritmo XY, pois não há divisão da rede. A Figura 42 ilustra 6 fluxos do tráfego complemento utilizado no experimento realizado. A Figura 42a ilustra a alocação dos canais físicos pelos fluxos usando o algoritmo XY e a Figura 42b ilustra a alocação correspondente, usando o algoritmo Hamiltoniano. Os números entre os roteadores representam a contagem dos canais físicos alocados pelos fluxos. Neste exemplo o algoritmo Hamiltoniano utiliza apenas a sub-rede dos canais maiores, pois os rótulos das origens são menores que os destinos. As linhas indicam o caminho dos fluxos da origem até o destino. Observe que no caso do XY (Figura 42a) os 6 fluxos alocam um total de 24 canais físicos, enquanto que no caso do Hamiltoniano os mesmos 6 fluxos alocam 19 canais físicos. Como conseqüência, o algoritmo Hamiltoniano aumenta o congestionamento, degradando seu desempenho em relação ao XY. Observe por exemplo na Figura 42b, entre os roteadores 7 e 8, três fluxos competindo por um mesmo canal físico. No caso do XY têm-se no máximo dois fluxos competindo por um mesmo canal físico.

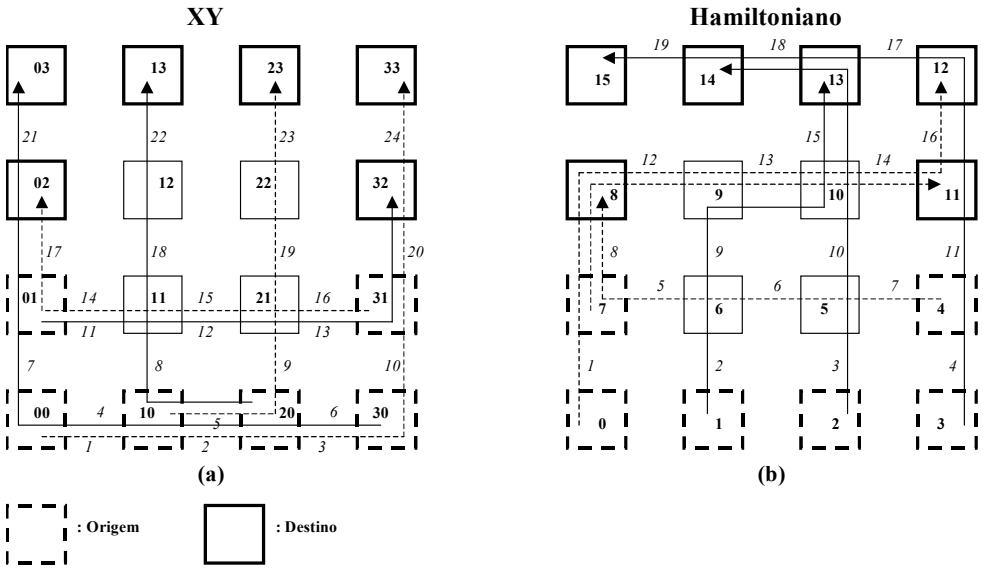


Figura 42 – Alocação dos canais físicos para os algoritmos de roteamento XY (a) e Hamiltoniano (b).

A adaptatividade adicionada ao algoritmo de roteamento Hamiltoniano faz um melhor aproveitamento da disponibilidade de canais físicos nas sub-redes porque produz caminhos que vão além do escopo da versão determinística. No entanto, o número máximo de canais físicos que podem ser alocados em cada sub-rede continua o mesmo, 24 no caso da rede utilizada. Se no cenário da Figura 42b fosse adicionado um novo fluxo na sub-rede dos canais maiores de origem 12 e destino 15, o número de canais alocados permaneceria 19. Adicionando o fluxo correspondente na Figura 42a (33 $\rightarrow$ 03), o número de canais alocados passaria de 24 para 27, excedendo o número máximo permitido em uma sub-rede, no caso do algoritmo Hamiltoniano. A Figura 43 apresenta o ganho de vazão (relativo à largura de banda da porta local) proporcionado pela adição da adaptatividade em função do tráfego (versão 1) ao algoritmo de roteamento Hamiltoniano determinístico. A partir do gráfico nota-se que a rede com roteamento determinístico satura em 150 Mbps e torna-se instável, com uma oscilação na vazão. Já com roteamento adaptativo a rede satura em 250 Mbps.Visto que a adaptatividade explora caminhos alternativos, a contenção de pacotes nos IPs origem reduz e tem-se um número maior de pacotes sendo transmitidos simultaneamente. Como conseqüência a vazão da rede aumenta.

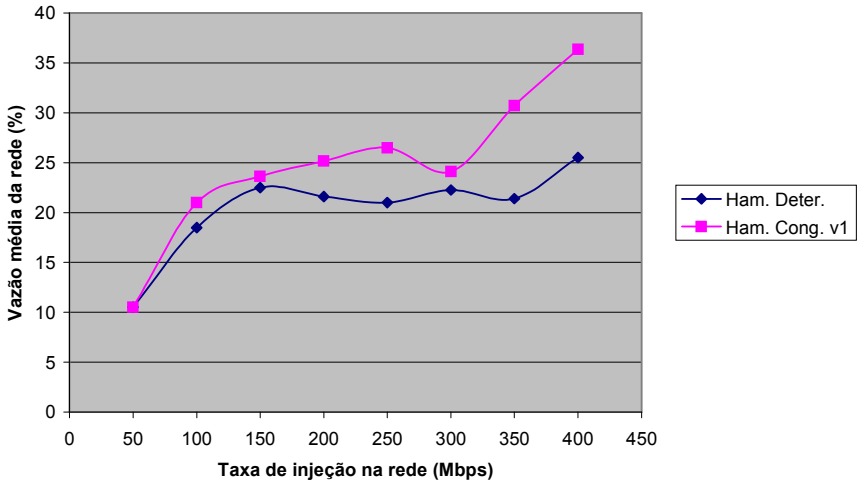


Figura 43 – Vazão média da rede para o padrão de tráfego complemento.

4.4 Conclusões do Capítulo

O estudo de caso apresentado mostrou que o grau de adaptatividade adicionado ao algoritmo de roteamento Hamiltoniano melhorou seu desempenho em situações de tráfego intenso. Através da adaptatividade, caminhos alternativos são explorados dentro da rede resultando em um aumento da sua vazão. A partir de indicativos de congestionamento, a adaptatividade foi otimizada visando a distribuição do tráfego em áreas menos congestionadas. Visto que a adaptatividade explora caminhos além dos mínimos, a latência por pacote pode aumentar em algumas situações, no entanto o desempenho geral da rede aumenta devido ao fato de que tem-se mais pacotes sendo transmitidos simultaneamente. O algoritmo de roteamento Hamiltoniano parcialmente adaptativo

em função do tráfego, proposto como estudo de caso, é uma contribuição deste trabalho tanto para a área de NoCs quanto para a área de multicomputadores, onde foi originalmente proposto em uma versão determinística para dar suporte a uma família de algoritmos de *multicast* para topologias malha bidimensional.

Em relação ao desempenho, quando comparado ao algoritmo XY, mostrou-se que a divisão da rede em duas sub-redes disjuntas implica uma limitação quanto à alocação de canais físicos nas sub-redes. Essa limitação impede que o algoritmo de roteamento Hamiltoniano, mesmo nas versões adaptativas, supere o desempenho do algoritmo XY em tráfegos intensos. No entanto, redes que implementam algoritmos de *multicast* tendo o suporte do algoritmo de roteamento Hamiltoniano podem tirar vantagem da adaptatividade em função do tráfego para melhorar o desempenho geral do sistema, pois o mesmo algoritmo de roteamento é aplicado tanto para tráfego *unicast* quanto *multicast*.

5 *MULTICAST*

Os ganhos de desempenho proporcionados pelas arquiteturas paralelas (MPSoCs) não estão relacionados somente ao poder computacional dos vários elementos de processamento. A arquitetura de interconexão, responsável pela intercomunicação dos elementos de processamento, tem um papel relevante no desempenho geral do sistema. NoCs podem ser vistas como a principal arquitetura de interconexão responsável pelo futuro das tecnologias multiprocessadas, as quais estão rapidamente prevalecendo em SoCs modernos [BJE06]. Visto que a infra-estrutura de comunicação dos MPSoCs está mudando de barramentos para NoCs, alguns serviços proporcionados pelos barramentos também devem estar disponíveis em NoCs, em especial a comunicação *multicast*. Mensagens *multicast* são responsáveis pela execução eficiente de diversas aplicações paralelas como algoritmos de pesquisa, busca em grafos e operações com matrizes (inversão e multiplicação por exemplo). Elas também são empregadas na implementação de diferentes protocolos como controle e configuração de rede, sincronização e coerência de *cache*. Apesar das arquiteturas baseadas em barramentos (simples, segmentados e hierárquicos) não serem escaláveis e proporcionarem baixo suporte a comunicações paralelas, elas nativamente suportam comunicação *multicast* e *broadcast*. Em NoCs, a comunicação *multicast* depende de algoritmos especiais comumente suscetíveis a *deadlock*. Além do mais, visto que mensagens *multicast* alocam vários recursos da rede simultaneamente, o desempenho geral do sistema pode ser seriamente degradado.

Este capítulo apresenta a implementação de um algoritmo de *multicast* para NoCs com topologia malha bidimensional que implementam *wormhole*. O algoritmo implementado é livre de *deadlock* e permite a transmissão simultânea de várias mensagens *multicast*.

5.1 Trabalhos Relacionados

Algoritmos de *multicast* para topologias malha que implementam *wormhole* têm sido extensivamente estudados e propostos para máquinas paralelas. Na área de NoCs são poucos os trabalhos que propõem algoritmos de *multicast*. Essa seção apresenta algumas propostas de *multicast/broadcast* para multicomputadores e NoCs, ambos com topologia malha.

5.1.1 *Multicast* em Multicomputadores

Os algoritmos *dual-path* e *multipath* propostos por Lin et al. em [LIN94] são baseados no algoritmo de roteamento Hamiltoniano (Capítulo 4, Seção 4.2). Os destinos de uma mensagem *multicast* são divididos em subconjuntos (no máximo quatro) e uma cópia da mensagem é enviada para cada um deles. Cada cópia é entregue aos seus destinos na ordem determinada pelo caminho Hamiltoniano definido para a rede. As diferentes cópias da mensagem usam conjuntos disjuntos de canais físicos e são roteadas independentemente.

No algoritmo *dual-path* os destinos são divididos em até dois conjuntos; maiores que a origem e menores que a origem. A Figura 44 ilustra o funcionamento do algoritmo, considerando o

nodo 14 a origem da mensagem *multicast* e os nodos destinos 0, 5, 13, 28, 29, 30 e 31. Neste exemplo, duas cópias da mensagem são criadas; uma é enviada para os destinos 0, 5 e 13, e a outra para os destinos 28, 29, 30 e 31.

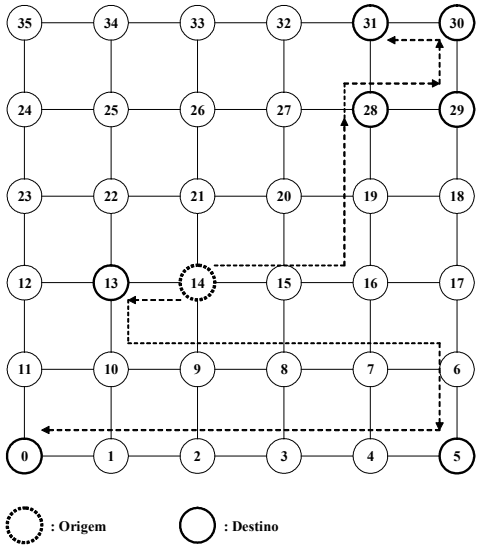


Figura 44 – Exemplo de funcionamento do algoritmo *dual-path*.

O algoritmo *dual-path* usa no máximo duas cópias da mensagem em uma comunicação *multicast*. Isso pode aumentar o caminho (mais *hops*) para algumas mensagens *multicast*. O algoritmo *multipath* procura reduzir os caminhos usando até quatro cópias da mensagem *multicast*. Nesse algoritmo, os destinos da mensagem são divididos em até quatro subconjuntos, de maneira que os destinos contidos em um mesmo subconjunto estão localizados no mesmo quadrante. A Figura 45 ilustra o funcionamento do algoritmo *multipath* considerando a mesma origem e destinos da Figura 44. Neste exemplo, três cópias da mensagem são criadas; uma é enviada para o destino 0, uma para o destino 5 e uma para os destinos 28, 29, 30 e 31.

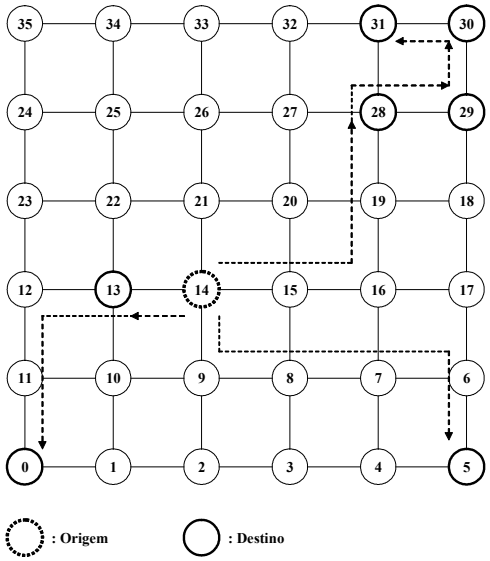


Figura 45 - Exemplo de funcionamento do algoritmo *multipath*.

Diferente dos algoritmos anteriores, o algoritmo *column-path* [BOP98] foi proposto para ser compatível com o algoritmo de roteamento *e-cube*. O *column-path* divide os destinos da mensagem em até *2k* subconjuntos (*k* corresponde ao número de colunas da malha), de maneira que sejam envidas no máximo duas mensagens em cada coluna. Se uma coluna da malha tem um ou mais destinos em linhas acima do nodo origem, então uma cópia da mensagem é enviada para servir todos esses destinos. Similarmente, se uma coluna tem um ou mais destinos em linhas abaixo do nodo origem, então uma cópia da mensagem é enviada para servir todos esses destinos. Se todos os destinos em uma coluna estiverem acima ou abaixo do nodo origem, então somente uma cópia da mensagem é enviada para a coluna, caso contrário, duas cópias da mensagem são enviadas. A Figura 46 ilustra o funcionamento do algoritmo considerando o nodo 22 a origem da mensagem *multicast* e os nodos destinos 00, 50, 12, 44, 54, 45 e 55. Neste exemplo, cinco cópias da mensagem são usadas para atingir todos destinos. Apesar dos destinos 50, 54 e 55 estarem na mesma coluna, duas cópias da mensagem são enviadas nessa coluna, pois dois destinos estão em linhas acima da origem (54 e 55) e um abaixo (50). Mesmo o destino 12 estando no caminho da cópia enviada para o destino 00, duas cópias da mensagem são usadas, pois os destinos encontram-se em colunas diferentes.

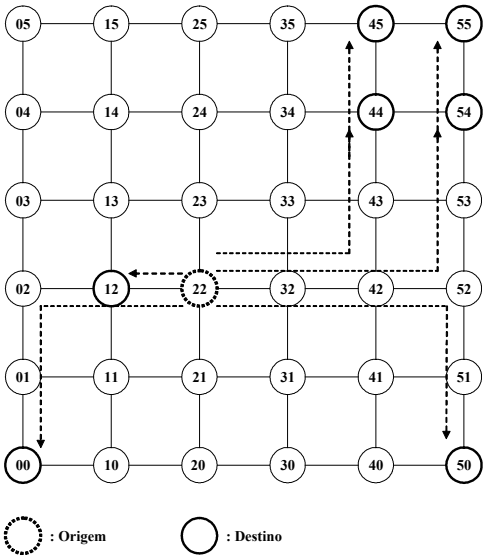
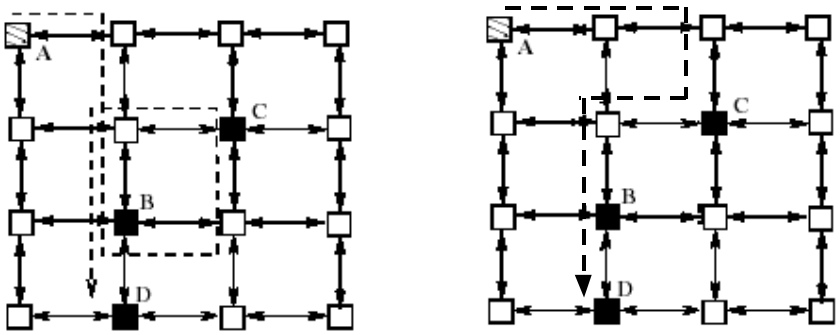


Figura 46 – Exemplo do funcionamento do algoritmo *column-path*. Rótulo dos roteadores compatível com o algortimo de roteamento *e-cube*.

5.1.2 *Multicast* em NoCs

Em [LU06] é proposto um algoritmo de *multicast* orientado a conexão. O algoritmo de roteamento utilizado é o XY. Somente uma cópia da mensagem é enviada nesse algoritmo. Antes de enviar a mensagem, a origem do *multicast* deve estabelecer uma conexão que inclua todos os destinos. A conexão é estabelecida por um pacote contendo no cabeçalho a identificação dos destinos, ordenados de tal maneira que o pacote passe por todos destino sem gerar ciclos A geração de ciclos no estabelecimento da conexão pode ocasionar *deadlock*. Quando o último destino receber o pacote, ele envia uma resposta para o roteador origem, o qual começa a transmitir a mensagem. A

Figura 47a ilustra um estabelecimento de conexão com um ciclo, pois o ordenamento dos destinos foi A→B→C→D. O ordenamento correto é apresentado na Figura 47b, ou seja, A→C→B→D. O XY empregado nessa implementação é ponto-a-ponto, por essa razão observa-se mudanças de direção de Y para X no caminho entre A e D. O inconveniente dessa solução está justamente na ordenação dos nodos destino, a qual os Autores não explicam como é feita.



(a) Com ciclo (b) Sem ciclo

Figura 47 – Exemplos de estabelecimento de conexão.

Bolotin et al. apresentam em [BOL07] um algoritmo de *broadcast* baseado em *flooding*[1], chamado de *Store-&-Forward Broadcast*. Ele é utilizado para a invalidação de linhas de *cache* em sistemas distribuídos baseados em NoCs. O mecanismo proposto envia em *broadcast* apenas mensagens de controle, as quais são em geral curtas e por isso são repassadas utilizando *store-and-forward*. Para o roteamento dos pacotes é utilizado o XY, de maneira que os roteadores que recebem a mensagem por um canal no eixo X (*East* ou *West*) replicam a mensagem nos canais do eixo Y (*North* e *South*) e a encaminham no eixo X. Os roteadores que recebem a mensagem por um canal no eixo Y (*North* ou *South*), somente a encaminham no mesmo eixo. Essa proposta não é muito atrativa por limitar o tamanho das mensagens, além de não dar suporte direto a tráfego *multicast*.

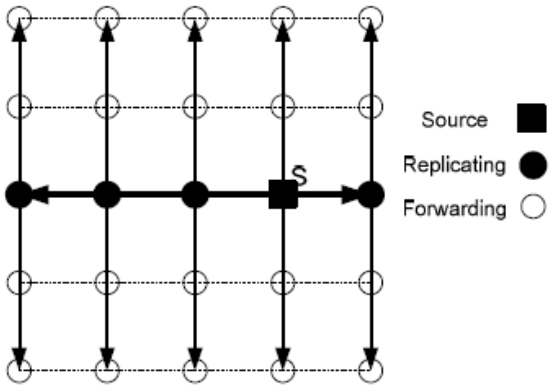


Figura 48 – Exemplo do funcionamento do *Store-&-Forward Broadcast* [BOL07].

Em [SET06b] é proposto um pseudo-algoritmo de *multicast*, baseado na idéia de *multicast* em *crossbar*. Visto que a arquitetura de roteador utilizada possui várias portas locais [SET06a], a

[1] Replicação das mensagens aos vizinhos próximos, na forma de uma propagação de onda.

mesma mensagem pode ser entregue simultaneamente aos vários IPs conectados. Os IPs destino da mensagem são identificados em um campo especial no cabeçalho do pacote. A Figura 49 ilustra o cabeçalho do pacote utilizado. Os campos X e Y identificam o roteador destino enquanto que o campo *nLID* identifica os IPs destino da mensagem. No campo *nLID*, cada bit identifica um IP conectado ao roteador. Essa proposta mostra-se muito limitada, sendo por isso referenciada como pseudo-algortimo. Claramente ela não pode ser empregada nas tradicionais arquiteturas de NoCs onde tem-se apenas um IP conectado a cada roteador.

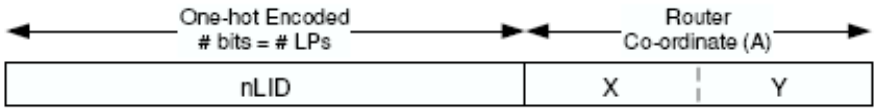


Figura 49 – Cabeçalho do pacote.

A abordagem proposta em [BOL07] é livre de *deadlock* devido ao uso de *store-and-forward* ao invés de *wormhole* no encaminhamento dos pacotes. O pseudo-*multicast* também é livre de *deadlock*, pois não implica nenhuma dependência entre os canais da rede. No entanto, todas as demais propostas de *multicast/broadcast* são suscetíveis a *deadlock* quando implementadas em malhas que implementam *wormhole*. O *deadlock* surge quando tem-se a transmissão simultânea de mensagens *multicast* com destinos em comum.

5.2 *Deadlock* em *Multicast Wormhole*

Diversos algoritmos de roteamento *unicast* livres de *deadlock* têm sido propostos para topologias malha que implementam *wormhole*, os quais podem ser estendidos para tráfego *multicast*. Estes algoritmos garantem ausência de dependência cíclica entre os canais de comunicação (canais entre roteadores vizinhos) através de restrições de roteamento. Visto que um tráfego *unicast* aloca um único canal de consumo[2] (no destino), não há dependência cíclica entre mensagens *unicast* simultâneas. No entanto, algoritmos *unicast* livres de *deadlock*, quando aplicados a tráfegos *multicast*, garantem ausência de *deadlock* apenas para a transmissão de uma mensagem *multicast* por vez. Devido ao fato do tráfego *multicast* possuir mais de um destino, vários canais de consumo e comunicação são alocados para uma única mensagem, por isso mensagens *multicast* simultâneas são suscetíveis a *deadlock* nos canais de consumo. Qualquer algoritmo de *multicast* que permite a alocação simultânea de vários canais de consumo e comunicação é suscetível a este tipo de *deadlock* [BOP98].

A Figura 50 ilustra uma situação de *deadlock* em canais de consumo, devido a transmissão simultânea de duas mensagens *multicast* em uma malha que implementa *wormhole*. O canal de consumo corresponde à interface roteador → IP, representado na Figura 50 pela seta que vai do *input buffer* até o IP *buffer* (*sink*). Ambas mensagens tem como destino os roteadores 1 e 2. Observe no roteador 2 que o pacote 1 (sufixo 1) não pode avançar porque o pacote 2 (sufixo 2) está alocando

[2] Canal de ejeção da porta local, o qual o IP usa para consumir os dados oriundos da rede. Corresponde à interface roteador → IP.

o canal de consumo (*sink2*), o qual é necessário para o pacote 1 prosseguir. O mesmo acontece no roteador 1, onde o pacote 2 não pode avançar porque o pacote 1 está alocando o canal de consumo (*sink1*). Claramente essa situação não ocorreria se o repasse de pacotes utilizado fosse *store-and-forward*, pois o repasse da mensagem só ocorreria no caso de existir espaço em *buffer* (*input buffer* e IP *buffer*) para seu armazenamento completo. No entanto, o tamanho máximo do pacote seria, nesse caso, limitado ao tamanho dos *buffers*.

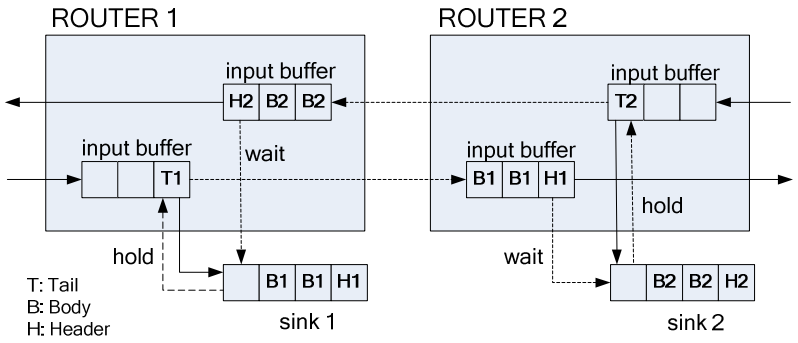


Figura 50 – *Deadlock* em canais de consumo [LU06].

Para minimizar a latência da mensagem, quando esta chega a um roteador destino, os *flits* são encaminhados simultaneamente para o canal de consumo e para o canal de comunicação com o roteador vizinho. Uma abordagem alternativa seria a *absort-and-retransmit*, na qual a mensagem é primeiramente *absorvida* pelo IP e em seguida *retransmitida* para o roteador vizinho. Apesar de essa abordagem lembrar o *store-and-forward* e aumentar a latência da mensagem, ela também é suscetível ao *deadlock* nos canais de consumo, como é mostrado detalhadamente em [BOP98]. Neste trabalho, Boppana et al. também apresentam uma solução para eliminar esse tipo de *deadlock*, baseada na replicação ou multiplexação (canais virtuais) dos canais de consumo.

5.3 Descrição da Implementação do Algoritmo *Dual-Path*

O algoritmo *dual-path multicast* foi adaptado para NoCs em duas versões: (*i*) utilizando chaveamento por circuito, proposta nesse trabalho; (*ii*) utilizando chaveamento por pacotes/*wormhole*, como foi proposto originalmente em [LIN94]. A proposta de utilizar chaveamento por circuito ao invés de chaveamento por pacotes visa enviar somente uma cópia da mensagem *multicast*, independente do número de subconjuntos de destinos. O algoritmo de roteamento utilizado, tanto para tráfego *unicast* quanto para *multicast*, é o Hamiltoniano determinístico (Capítulo 4, Seção 4.2). Dessa maneira é possível eliminar o *deadlock* entre os dois tipos de tráfegos, pois há compatibilidade no roteamento das mensagens [BOP98].

5.3.1 Algoritmo *Dual-Path* Orientado a Conexão

O primeiro passo no algoritmo *dual-path* orientado a conexão é estabelecer uma conexão entre o IP origem e os IPs destinos da mensagem. O IP origem deve dividir os destinos em dois subconjuntos, um contendo os destinos com rótulos maiores que a origem e outro contendo os

menores. Cada subconjunto é inserido dentro de um *pacote de estabelecimento de conexão*. O pacote para o estabelecimento de conexão com os roteadores maiores que a origem deve conter os rótulos ordenados em ordem crescente, enquanto que o outro pacote deve conter os rótulos ordenados em ordem decrescente. A Figura 51 ilustra o formato dos pacotes de estabelecimento de conexão considerando o roteador origem 6 e os destinos 1, 3, 5, 8, 10, 12 e 14.

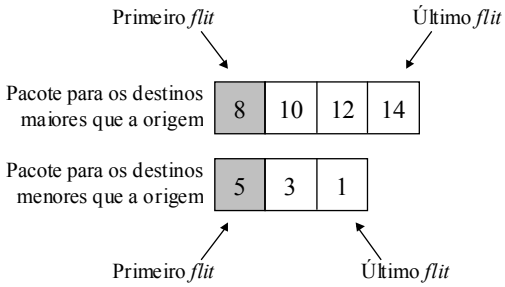


Figura 51 – Pacotes de estabelecimento de conexão.

Os pacotes de estabelecimento de conexão têm tamanho variado, dependendo do número de destinos de cada subconjunto. A Figura 52 ilustra o formato dos *flits* que compõem os pacotes, considerando-se um tamanho de *flit* igual a 8 bits. Através do campo *Controle* de cada *flit*, o último destino do subconjunto é identificado. O algoritmo *dual-path* também é compatível com roteadores com várias portas locais, bastando que seja indicada, juntamente com o roteador destino, a porta local.

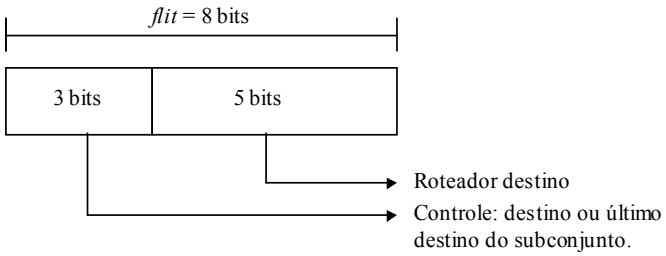


Figura 52 – Formato dos *flits* do pacote de estabelecimento de conexão.

Primeiramente é enviado o pacote para o estabelecimento de conexão com os destinos maiores. O IP origem espera por um sinal de reconhecimento (*ack*) indicando que a conexão está estabelecida. Em seguida é enviado o outro pacote e novamente o roteador origem espera pelo sinal de reconhecimento. O sinal de reconhecimento é gerado pelo último IP destino a receber o pacote. Tendo recebido os dois sinais de reconhecimento, o segundo passo do algoritmo inicia.

Durante o estabelecimento de conexão, o algoritmo de roteamento Hamiltoniano é executado baseado no primeiro *flit* do pacote. Quando o pacote chega ao roteador contido no primeiro *flit* do pacote, o canal de consumo é alocado. Em seguida esse *flit* é removido e o roteamento é executado novamente baseado no próximo *flit*. Esse processo se repete até o último *flit* do pacote atingir o último roteador destino da mensagem. A Figura 53a e a Figura 53b mostram o estabelecimento de conexão com os roteadores maiores e menores que a origem, respectivamente.

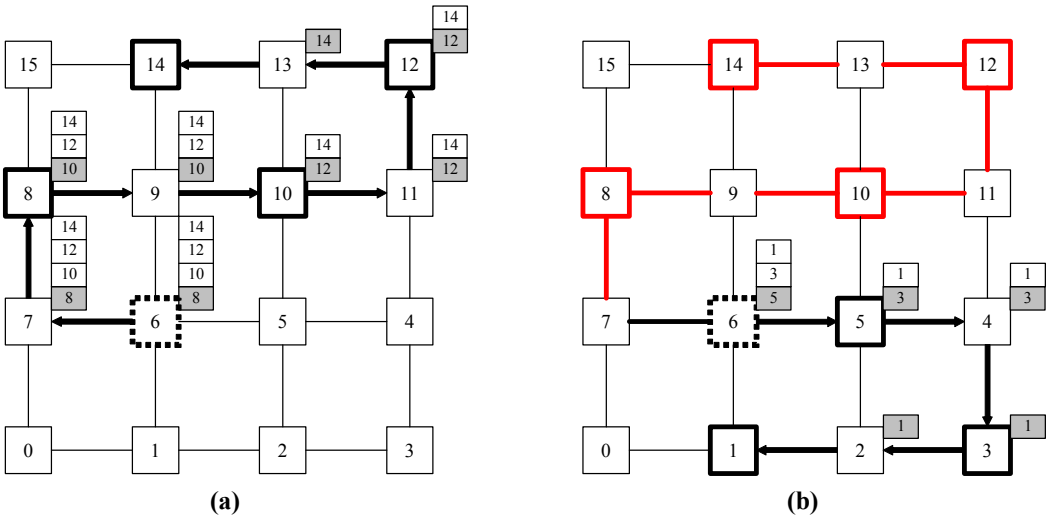


Figura 53 – (a) estabelecimento de conexão com os roteadores maiores que a origem; (b) estabelecimento de conexão com os roteadores menores que a origem.

O segundo passo do algoritmo é o envio de uma única cópia da mensagem *multicast*, a qual é transmitida simultaneamente para todos os destinos através da conexão estabelecida no passo anterior. Na medida em que o último *flit* da mensagem vai sendo transmitido, a conexão vai sendo desfeita. Em casos onde existem apenas destinos maiores ou menores que a origem, somente um pacote de estabelecimento de conexão é enviado e o segundo passo do algoritmo inicia logo após a recepção do sinal de reconhecimento.

5.3.2 Algoritmo *Dual-Path* com Chaveamento por Pacotes

Nesta versão do algoritmo duas cópias independentes da mensagem são criadas e a cada uma delas é anexado um cabeçalho. Os cabeçalhos anexados correspondem aos pacotes de estabelecimento de conexão apresentados na Figura 51. A Figura 54 ilustra um exemplo das duas cópias da mensagem *multicast* com os cabeçalhos anexados, considerando o roteador origem 6 e os destinos 1, 3, 5, 8, 10, 12 e 14. A partir do campo *Controle* presente nos *flits* de cabeçalho é possível identificar o último destino do subconjunto. Os *flits* subseqüentes correspondem à mensagem *multicast*.

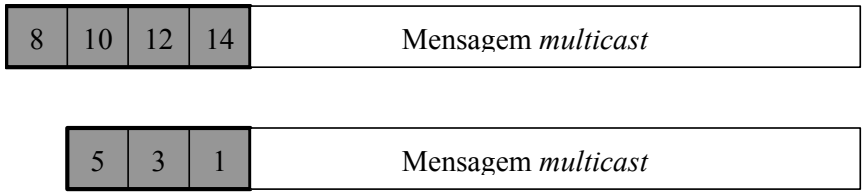


Figura 54 – Cabeçalhos anexados às cópias das mensagens.

O IP origem injeta ambas mensagens na rede sem estabelecimento de conexão, uma após a outra. Cada uma das cópias atinge os destinos na ordem definida no cabeçalho. A cada destino atingido pela mensagem, o canal de consumo é alocado. Em seguida o primeiro *flit* do cabeçalho é removido e o roteamento Hamiltoniano é executado novamente para alocar um canal de comunicação com o próximo roteador vizinho. Uma vez alocados ambos canais (consumo e

comunicação), a mensagem *multicast* é transmitida simultaneamente para o IP local e para o roteador vizinho.

5.3.3 Prevenção de *Deadlock*

O algoritmo de roteamento Hamiltoniano é livre de *deadlock* em se tratando de tráfego *unicast*, pois os canais de comunicação da rede (canais entre roteadores vizinhos) são divididos entre duas sub-redes disjuntas e acíclicas (Capítulo 4, Seção 4.2, Figura 36). Portanto, mensagens que circulam em uma mesma sub-rede não geram ciclos entre si (rede acíclica) e não se misturam com as mensagens da outra sub-rede (redes disjuntas). No entanto, os canais de consumo são compartilhados entre as duas sub-redes, de certa maneira unindo-as. Em se tratando de tráfego *unicast* isso não é problema. Mensagens *unicast* de diferentes sub-redes que competem por um mesmo canal de consumo não geram dependência cíclica entre si, pois a primeira a alocar o canal de consumo vai liberá-lo logo após encerrar a transmissão. Visto que mensagens *multicast* alocam vários canais de consumo, uma dependência cíclica pode ocorrer quando mensagens *multicast* de diferentes sub-redes competem por canais de consumo em comum (Figura 50), ocasionando *deadlock*. Para resolver esse tipo de *deadlock* é necessário que as sub-redes também sejam disjuntas em relação aos canais de consumo, de maneira que cada sub-rede tenha o seu canal de consumo. O número mínimo de canais de consumo por roteador necessário para prevenir o *deadlock* em canais de consumo é igual ao número de sub-redes disjuntas e acíclicas da rede [BOP98]. Visto que o algoritmo Hamiltoniando divide a rede em duas sub-redes disjuntas e acíclicas, dois canais de consumo por roteador são suficientes para eliminar o *deadlock*.

Para eliminar o *deadlock* nos canais de consumo (Seção 5.2) em ambas versões do algoritmo, optou-se pela replicação dos canais de consumo, visto que ela oferece melhor desempenho e menor consumo de área quando comparada à multiplexação (Capítulo 2, Seção 2.4). Nessa abordagem a porta local do roteador é replicada somente no sentido Roteador → IP, obtendo-se assim dois canais de consumo por roteador (Figura 55). Quando a mensagem *multicast* (ou o pacote de estabelecimento de conexão) é enviada para os roteadores maiores que a origem, o canal de consumo 1 é alocado, caso contrário é alocado o canal de consumo 2. Para mensagens *unicast* essa ordenação dos canais de consumo é desnecessária e o primeiro canal de consumo livre é alocado.

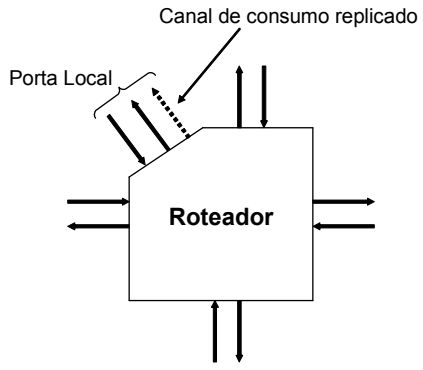


Figura 55 - Replicação do canal de consumo da porta local.

5.4 Validação do Algoritmo *Dual-Path*

O algoritmo *dual-path* foi descrito em VHDL e validado por simulação funcional e prototipação em FPGA. A Figura 56 ilustra um cenário no qual o IP6 envia uma mensagem *multicast* para os IPs 4, 5, 7 e 8. A seguir será apresentada a validação, por simulação funcional, correspondente às duas versões do algoritmo.

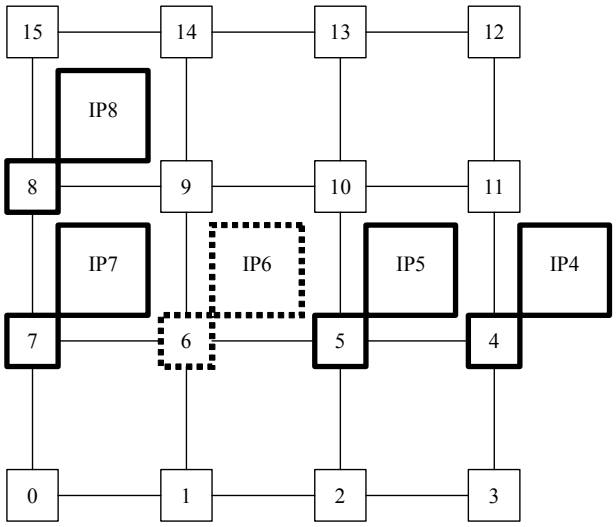


Figura 56 – Mensagem *multicast* enviada do IP6 para os IPs 4, 5, 7 e 8.

A Figura 57 e a Figura 58 ilustram o estabelecimento de conexão com os IPs destino, correspondente à versão orienta a conexão. A Figura 59 ilustra a transmissão da mensagem *multicast*. São mostradas as formas de onda correspondentes à interface de saída do IP origem e as interfaces de entrada dos IPs destino. A descrição dos passos da simulação segue às figuras.

1. O IP origem envia o pacote de estabelecimento de conexão com os destinos maiores (IP7 e IP8). O *flit* F0 sinaliza para a NoC que será executado um *multicast* bidirecional (dois subconjuntos de destinos). Em casos de *multicast* unidirecional (um conjunto de destinos), nenhuma sinalização é necessária. Os demais *flits* correspondem aos destinos maiores. O valor "A" no *flit* A7 indica que o destino 7 não é o último do subconjunto. O valor "C" no *flit* C8 indica que o destino 8 é o último do subconjunto.
2. O pacote de estabelecimento de conexão atinge o roteador/IP7 e um canal de consumo é alocado. Em seguida o *flit* A7 é removido e o roteamento é executado novamente baseado no *flit* C8. Em roteadores nos quais o roteamento é executado duas vezes, a mensagem leva mais tempo para ser repassada, pois é necessária a alocação de uma porta local (canal de consumo) e uma porta de saída (canal de comunicação).
3. O pacote de estabelecimento de conexão atinge o roteador/IP8 e um canal de consumo é alocado. Visto que se trata do último destino do subconjunto, o IP8 gera um sinal de reconhecimento (*ack_out = '1'*) confirmando o estabelecimento da conexão.
4. O sinal de reconhecimento é recebido pelo IP origem (*ack_in = '1'*).

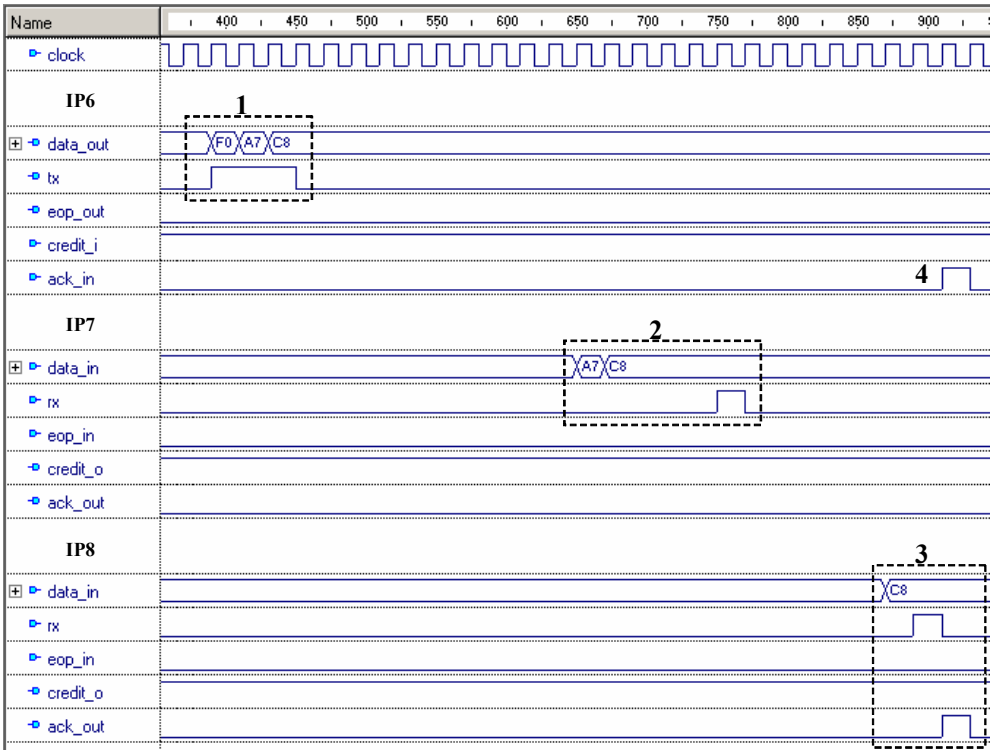


Figura 57 – Estabelecimento de conexão com os destinos maiores (IP7 e IP8).

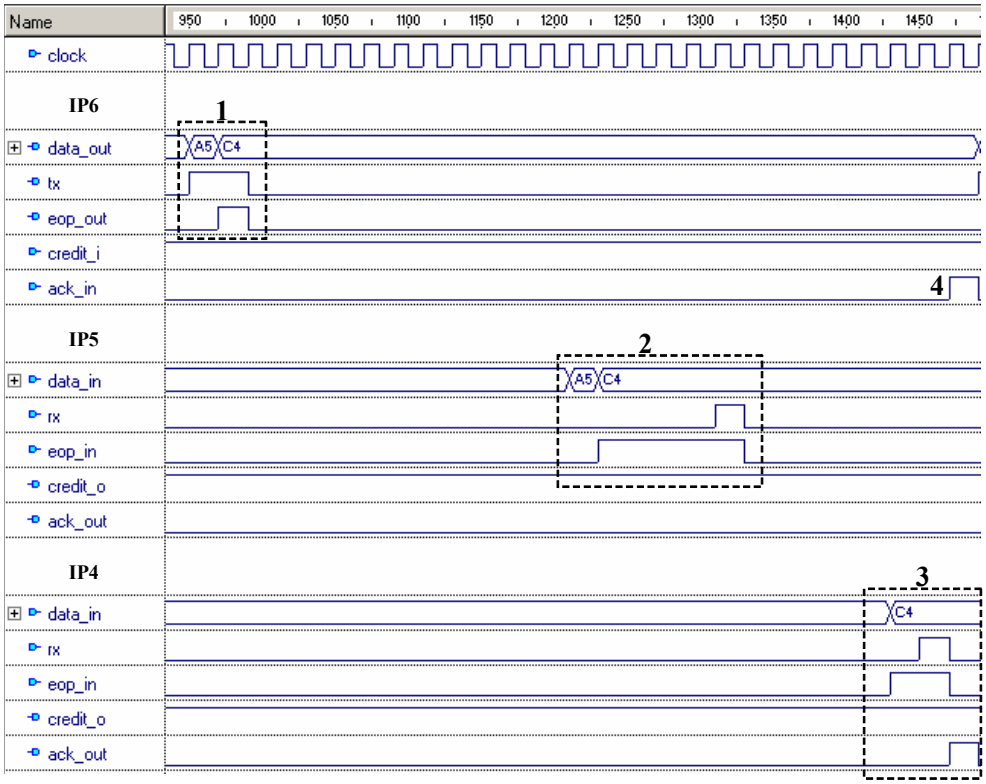


Figura 58 – Estabelecimento de conexão com os destinos menores (IP4 e IP5).

1. O IP origem envia o pacote de estabelecimento de conexão com os destinos menores. Para o estabelecimento de conexão com os destinos menores não é necessário o *flit* de sinalização F0.
2. O pacote de estabelecimento de conexão atinge o roteador/IP5 e um canal de consumo é alocado. Em seguida o *flit* A5 é removido e o roteamento é executado novamente baseado no *flit* C4.
3. O pacote de estabelecimento de conexão atinge o roteador/IP4 e um canal de consumo é alocado. Visto que se trata do último destino do subconjunto, o IP4 gera um sinal de reconhecimento (*ack_out = '1'*) confirmando o estabelecimento da conexão.
4. O sinal de reconhecimento é recebido pelo IP origem (*ack_in = '1'*). Tendo recebido os dois sinais de reconhecimento, a mensagem pode ser injetada na rede.

A Figura 59 ilustra o IP6 transmitindo uma única cóipa da mensagem *multicast*, a qual é recebida simultaneamente pelos destinos IP4, IP5, IP7 e IP8. A descrição dos passos da simulação segue à figura. Observe que os destinos equidistantes da origem (IP6) começam a receber a mensagem simultaneamente.

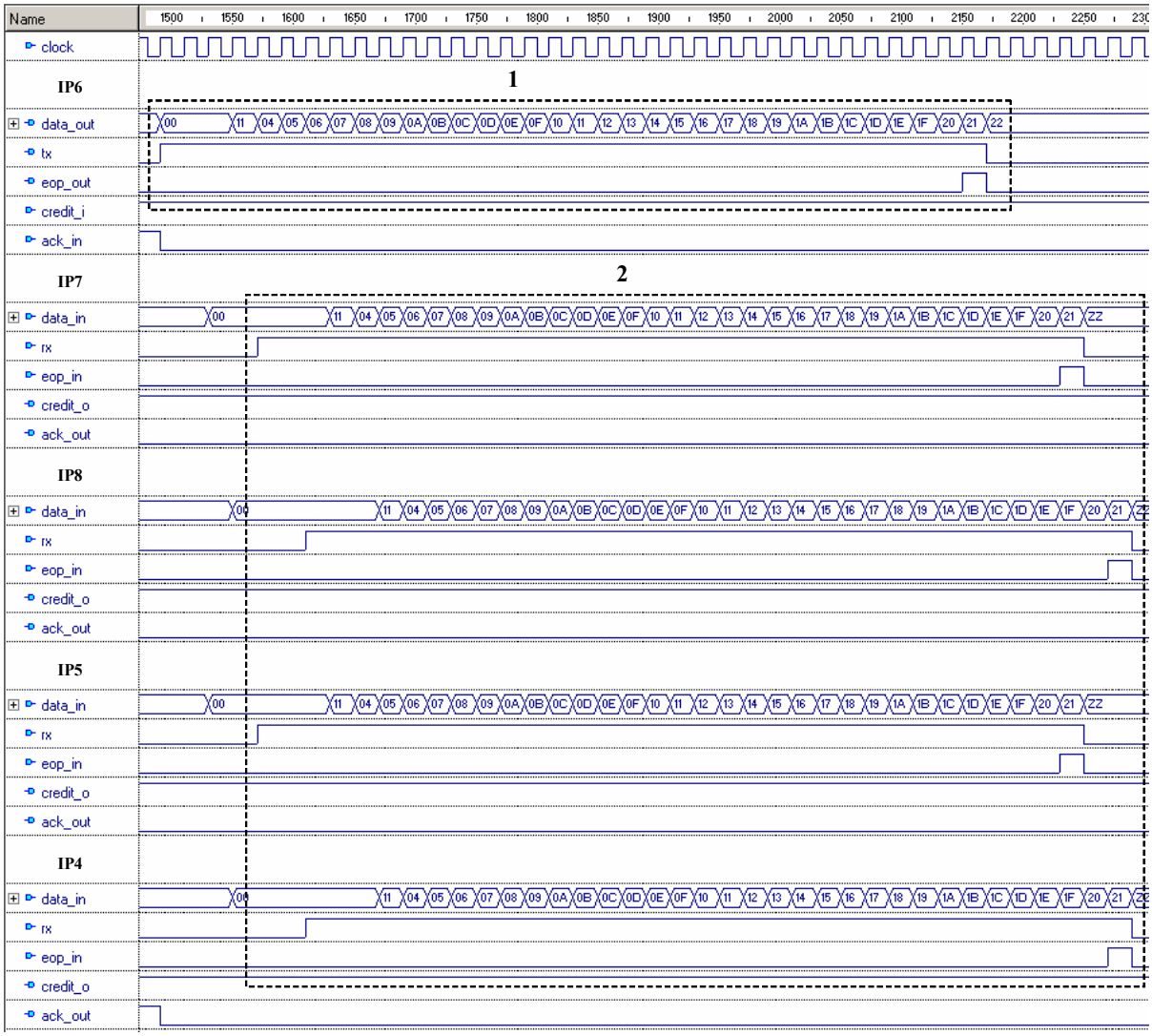


Figura 59 – Transmissão da mensagem *multicast*.

1. O IP origem transmite a mensagem *multicast*.
2. Os IPs destino recebem a mensagem simultaneamente. Observe que a recepção da mensagem pelos IPs 8 e 4 está defasada em relação a recepção dos IPs 7 e 5. Isso ocorre devido a distância (número de *hops*) dos IPs destino em relação a origem.

A Figura 60 e a Figura 61 ilustram a transmissão da mensagem *multicast* para os destinos maiores e menores que a origem, respectivamente, utilizando chaveamento por pacotes. São mostradas as formas de onda correspondentes à interface de saída do IP origem e as interfaces de entrada dos IPs destino. A descrição dos passos da simulação segue às figuras.

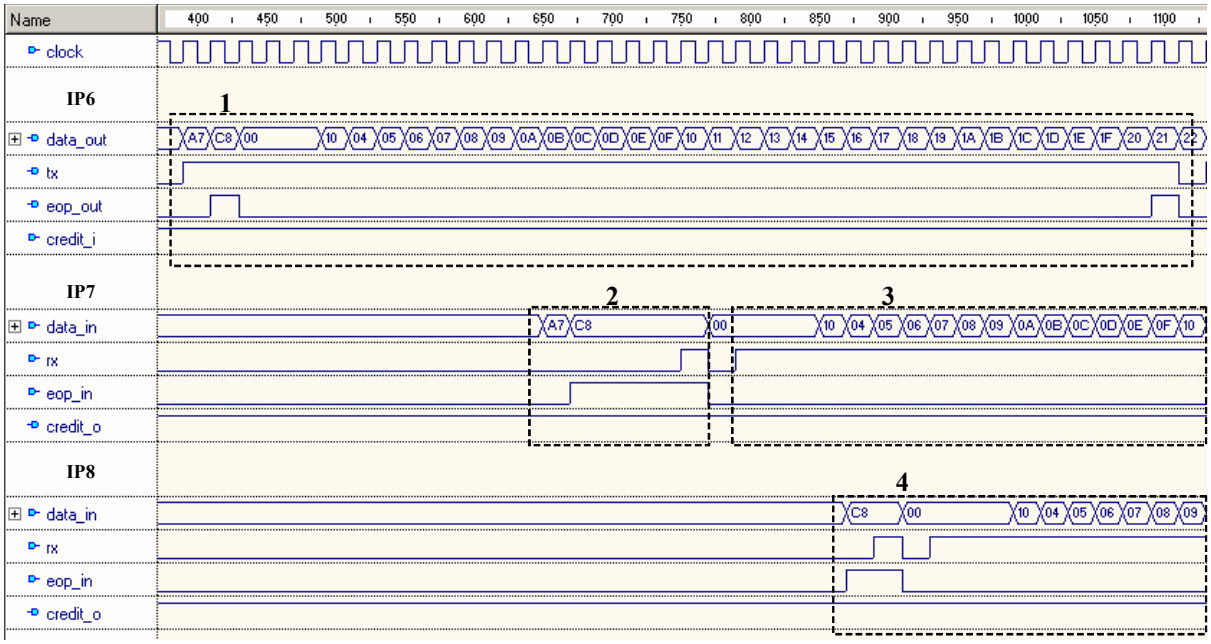


Figura 60 – Transmissão da mensagem multicast para os destinos maiores (IP7 e IP8).

1. O IP origem transmite a primeira cópia da mensagem *multicast* para os destinos maiores (IP7 e IP8). Os dois primeiros *flits* (A7 e C8) correspondem ao cabeçalho da mensagem.
2. A mensagem atinge o roteador/IP7 e um canal de consumo é alocado. Em seguida o *flit* A7 é removido e o roteamento é executado novamente baseado no *flit* C8.
3. Após alocar um canal de consumo no roteador 7 e um canal de comunicação com um roteador vizinho, a mensagem é recebida pelo IP7 ao mesmo tempo em que é encaminha para o roteador vizinho.
4. A mensagem atinge o último destino do subconjunto. Um canal de consumo é alocado e a mensagem começa a ser entregue ao IP8. A partir desse momento, os destinos maiores estão recebendo simultaneamente a mensagem.

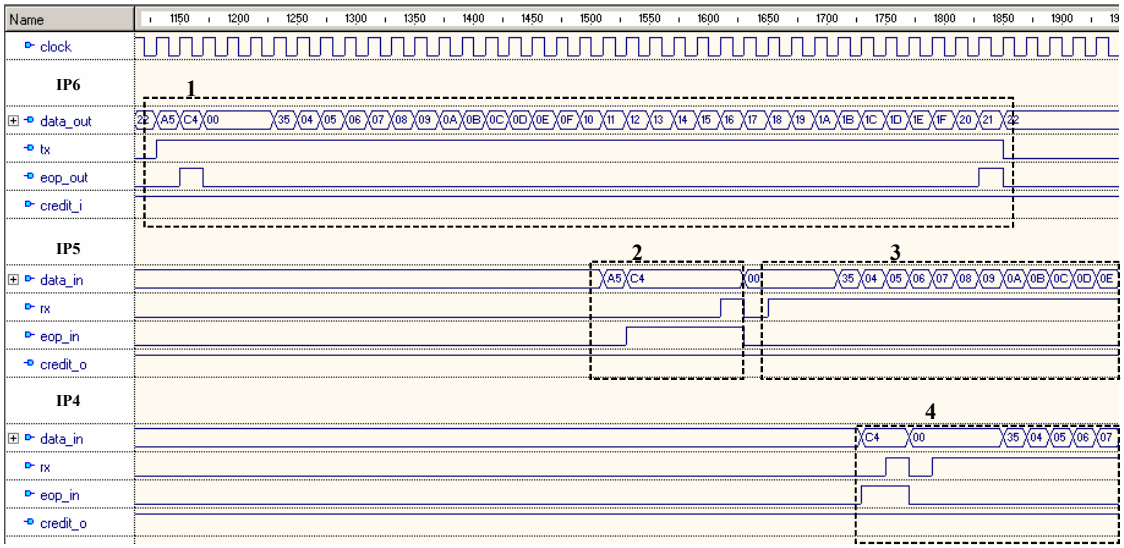


Figura 61 - Transmissão da mensagem multicast para os destinos menores (IP4 e IP5).

1. O IP origem transmite a segunda cópia da mensagem *multicast* para os destinos menores (IP4 e IP5). Os dois primeiros *flits* (A5 e C4) correspondem ao cabeçalho da mensagem.
2. A mensagem atinge o roteador/IP5 e um canal de consumo é alocado. Em seguida o *flit* A5 é removido e o roteamento é executado novamente baseado no *flit* C4.
3. Após alocar um canal de consumo no roteador 5 e um canal de comunicação com um roteador vizinho, a mensagem é recebida pelo IP5 ao mesmo tempo em que é encaminhada para o roteador vizinho.
4. A mensagem atinge o último destino do subconjunto. Um canal de consumo é alocado e a mensagem começa a ser entregue ao IP4. A partir desse momento, os destinos menores estão recebendo simultaneamente a mensagem.

5.5 Avaliação de Desempenho e Consumo de Área e Energia

Nessa seção as duas versões do algoritmo *dual-path* (CP – chaveamento por pacotes e CC – chaveamento por circuito) são comparadas com o algoritmo individual. No algoritmo individual, uma mensagem *unicast* é gerada para cada destino do *multicast* [BOP98], resultando em *n* mensagens individuais para *n* destinos. O algoritmo individual utilizado nos seguintes experimentos é baseado no roteamento XY. O tráfego *unicast* envolvido em todos experimentos é orientado a conexão. O tempo para a preparação dos subconjuntos de destinos no algoritmo *dual-path* não é levado em consideração, pois trata-se de tempo do computação do IP origem. A NoC utilizada nos experimentos tem as características apresentadas naTabela 10, com *buffers* de entrada de dezesseis posições. Não são usados canais virtuais nem replicados.

Tabela 10 – Características da NoC utilizada.

Tamanho de Flit/phit	8 bits
Controle de fluxo	Baseado em créditos
Topologia da NoC	Malha 4x4
Algoritmo de roteamento	Hamiltoniano determinístico
Modo de chaveamento	Pacotes/*Wormhole* e Circuito

5.5.1 Tráfego *Multicast*

O primeiro experimento envolve somente tráfego *multicast*. Esse tipo de cenário não é comum em aplicações reais, no entanto o propósito é verificar o desempenho dos algoritmos em situações de tráfego intenso. Neste cenário todos IPs (um por roteador) enviam ao mesmo tempo 200 mensagens *multicast* de 100 *flits* na taxa da rede. O número de destinos das mensagens varia de 2 a 14. A Figura 62 apresenta os resultados obtidos.

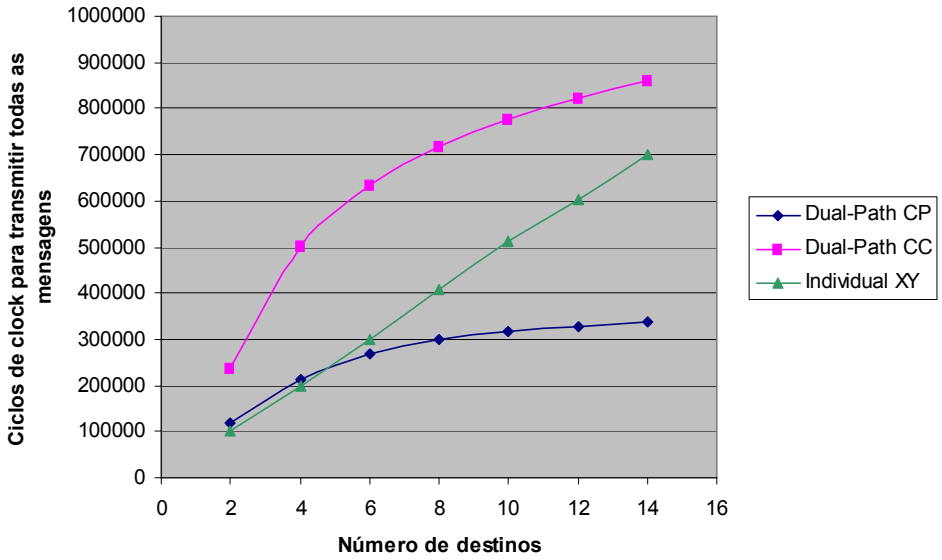


Figura 62 – Desempenho dos algoritmos *multicast* em função do número de destinos.

Como esperado, o número de ciclos para transmitir todas mensagens usando o algoritmo individual cresce linearmente com o número de destinos, pois um novo destino implica uma nova cópia da mensagem. Visto que o algoritmo *dual-path*, em suas duas versões, é menos sensível ao número de destinos das mensagens, o impacto da latência tende a ser cada vez menor. O baixo desempenho apresentado pela versão orientada a conexão do algoritmo (*dual-path* CC) justifica-se pelo alto tempo para o estabelecimento da conexão em situações de tráfego intenso. Mesmo tendo parte dos recursos alocados (por exemplo, circuito estabelecido com os destinos maiores) é necessária a alocação dos demais antes de iniciar a transmissão da mensagem. Como conseqüência tem-se congestionamento na rede. O desempenho superior da versão com chaveamento por pacotes (*dual-path* CP), mesmo tendo de enviar até duas cópias da mensagem, deve-se justamente ao fato

da desnecessidade de estabelecimento de conexão. Uma vez alocados os canais de consumo e comunicação em um roteador, a mensagem pode ser encaminhada.

5.5.2 Tráfegos *Unicast* e *Multicast* Simultâneos

Neste segundo experimento, tráfegos *unicast* e *multicast* são injetados simultaneamente na rede. Neste cenário todos IPs (um por roteador) enviam 200 mensagens de 100 *flits* a uma taxa de 14% da largura de banda do canal (abaixo do ponto de saturação da rede). Do total de mensagens enviadas por cada IP, 10% são *multicast* e os 90% restantes são *unicast*. O número de destinos das mensagens *multicast* varia de 2 a 14. A Figura 63 apresenta os resultados obtidos.

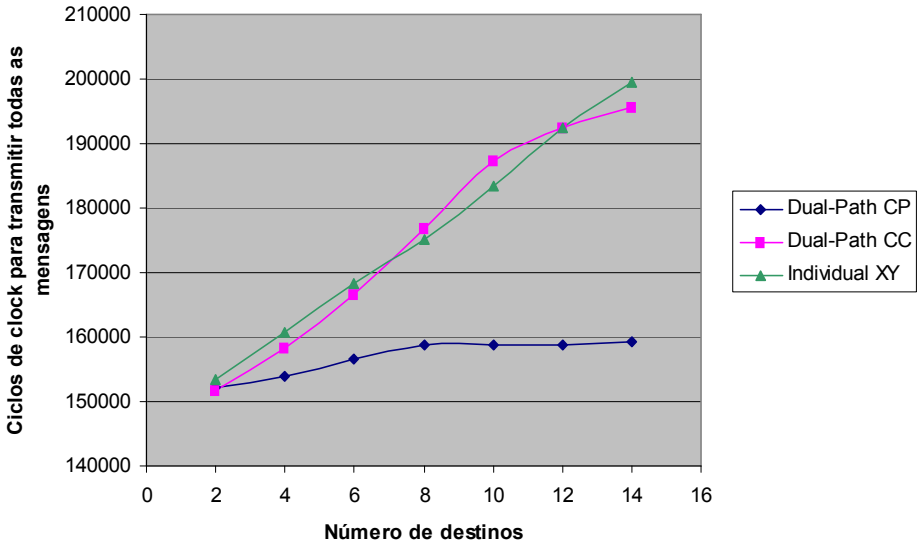


Figura 63 – 10% de tráfego *multicast* e 90% de tráfego *unicast* a uma taxa de injeção igual a 14%.

Comparando-se a Figura 62 (alta taxa de injeção) à Figura 63 (baixa taxa de injeção), o impacto do tempo de estabelecimento de conexão no tempo total para a entrega das mensagens reduz e o *dual-path* CC apresenta um desempenho semelhante ao algoritmo individual. O *dual-path* CP mostra-se superior em altas e baixas taxas de injeção.

A Figura 64 ilustra o desempenho dos algoritmos na medida em que o tráfego vai se intensificando devido ao aumento da quantidade total de mensagens *multicast*. Neste cenário todos IPs (um por roteador) enviam 200 mensagens de 100 *flits* a uma taxa de 14% da largura de banda do canal. Do total de mensagens enviadas por cada IP, a quantidade de mensagens *multicast* varia de 10% a 90%. O número de destinos das mensagens *multicast* foi fixado em 4. Para baixas taxas de injeção, poucos destinos (4) e uma baixa percentagem de mensagens *multicast*, os algoritmos apresentam um desempenho similar (diferença inferior a 10%). O desempenho do *dual-path* CP é superior nos demais casos. O desempenho do algoritmo individual é uma função do número de destinos. O *dual-path* CC é recomendado somente para baixas taxas de injeção devido ao tempo para o estabelecimento de conexão.

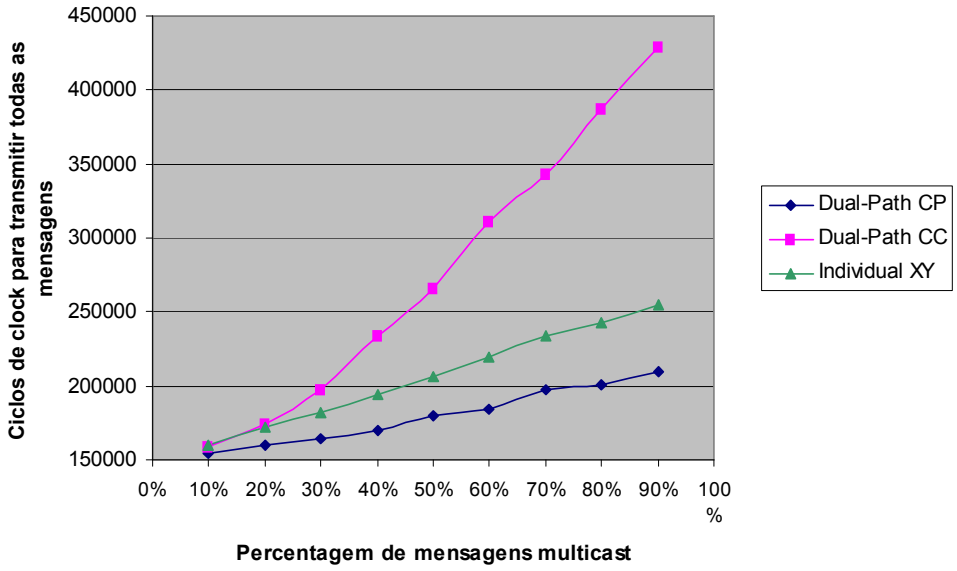


Figura 64 - Desempenho dos algoritmos *multicast* em função da percentagem de mensagens *multicast*.

5.5.3 Impacto do Tráfego *Multicast* sobre o Tráfego *Unicast*

A Figura 65 compara o impacto do tráfego *multicast* sobre um tráfego puramente *unicast*, para os diferentes algoritmos. Cada IP (um por roteador) envia 200 mensagens de 50 *flits* a uma taxa de injeção que varia de 9% a 50%. No tráfego puramente *unicast* (curva *Unicast* na Figura 65), 100% das mensagens enviadas por cada IP são *unicast* com destinos uniformemente aleatórios. As demais curvas ilustram o impacto do tráfego *multicast* no desempenho, considerando 10% do total de mensagens enviadas por cada IP sendo *multicast* (8 destinos) e os 90% restantes *unicast*.

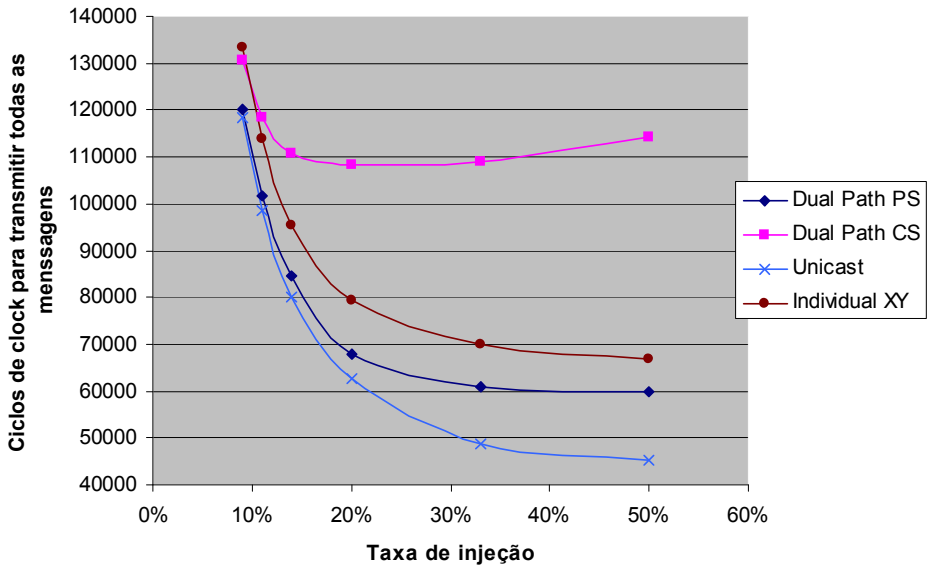


Figura 65 – Impacto do tráfego *multicast* sobre o tráfego puramente *unicast*.

Para baixas taxas de injeção, como 14%, o tempo total para a transmissão das mensagens utilizando o *dual-path* CP aumenta somente 5,7%, enquanto que utilizando o *dual-path* CC e o individual o tempo aumenta 38% e 19% respectivamente. Para taxas de injeção típicas, entre 20% e 25%, o *dual-path* CP tem um impacto médio de 10% no desempenho. Em altas taxas de injeção, como 50%, o impacto do *dual-path* CP no desempenho é de 32%, enquanto que o do individual é de 47%.

A partir da Figura 65 também é possível quantificar os ganhos de desempenho proporcionados pela inclusão de serviços de *multicast* em NoCs. Isto pode ser observando pelo o intervalo entre as curvas dos algoritmos *dual-path* CP e individual. Para taxas de injeção entre 20% e 50% o ganho médio de desempeno é 12,3% quando se inclui um serviço de *multicast* na rede (para mensagens com 8 destinos e 10% de tráfego *multicast*).

5.5.4 Consumo de Área

A Tabela 11 apresenta a área de FPGA consumida pela implementação do algoritmo *dual-path* sobre um roteador de cinco portas. As características do roteador são listadas na Tabela 10 (exceto a topologia). Para um único roteador de cinco portas, um consumo extra de área de 20,2% pode ser observado quando o algoritmo *dual-path* é utilizado. Parte da lógica adicional necessária para a implementação do algoritmo corresponde à análise do cabeçalho que contém os destinos da mensagem *multicast*. A maior parte do custo em área corresponde à replicação do canal de consumo, que além de prevenir o *deadlock*, também aumenta a vazão da rede.

Tabela 11 - Resultados de área para a implementação do algoritmo *dual-path*, usando o dispositivo Virtex 2VP30.

Recurso	Roteador de 5 portas	Roteador de 5 portas com *dual-path*	Disponibilidade
Slices	302	363	13696
LUTs	603	726	27392
Flip Flops	189	211	29060

5.5.5 Consumo de Energia

Este experimento visa comparar os algoritmos individual e *dual-path* CP em termos de consumo de energia e a potência dissipada. No cenário utilizado todos IPs enviam um total de 200 mensagens de 50 *flits* a uma taxa de 20% da largura de banda do canal. Do total de mensagens enviadas por cada IP, a quantidade de mensagens *multicast* varia de 10% a 90%. O número de destinos das mensagens *multicast* foi fixado em 8.

O algoritmo individual envia a mesma cópia da mensagem *multicast* para cada um dos seus destinos utilizando sempre caminhos mínimos. Já o algoritmo *dual-path* CP envia no máximo duas cópias, entretanto o caminho estabelecido para englobar os destinos da mensagem *multicast* é mais longo. Essa relação entre o número de cópias enviadas e o comprimento dos caminhos utilizados gera uma compensação entre os algoritmos, fazendo com que ambos tenham uma dissipação de

potência semelhante, como pode ser observado na Figura 66.

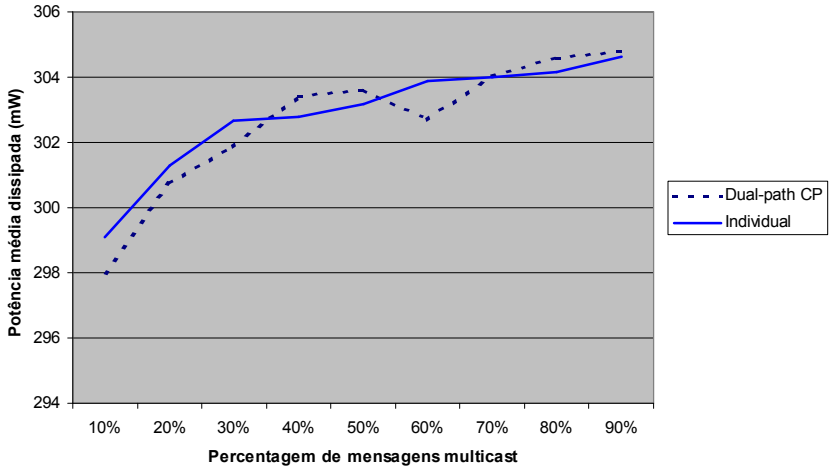


Figura 66 – Potência média dissipada pela NoC em função da percentagem de mensagens *multicast*.

O consumo de energia dos algoritmos está relacionado ao tempo total para a transmissão das mensagens. Enquanto as mensagens estão sendo transmitidas os circuitos da NoC estão em atividade de chaveamento e ela está dissipando potência, conseqüentemente a energia está sendo consumida. Visto que a potência dissipada por ambos algoritmos é semelhante e o *dual-path* CP leva menos tempo para transmitir as mensagens, este proporciona um consumo de energia menor, como mostra a Figura 67.

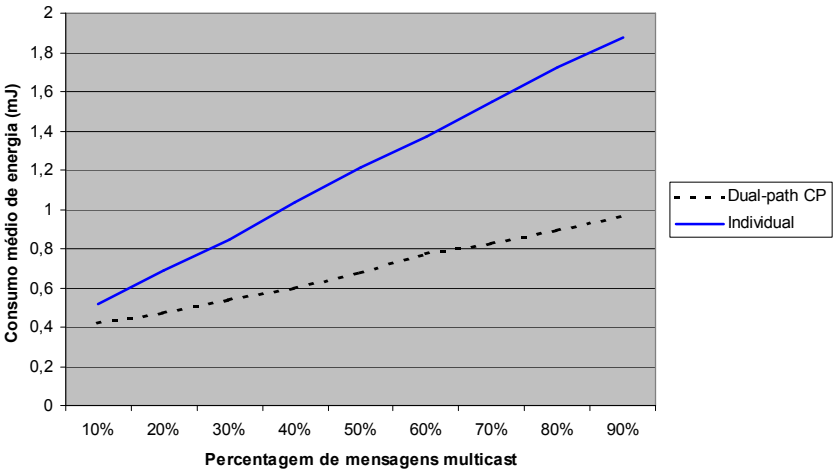


Figura 67 – Consumo médio de energia da NoC em função da percentagem de mensagens *multicast*.

O gráfico da Figura 67 demonstra a efetividade da utilização do serviço de *multicast* em NoCs. Dado que este tipo de tráfego é comum em MPSoCs, e,g. coerência de *cache*, o uso desta técnica permite aumentar a autonomia da bateria em sistemas embarcados.

5.6 Conclusões do Capítulo

Esse capítulo apresentou uma adaptação livre de *deadlock* do algoritmo *dual-path* [LIN94] para NoCs em duas versões, variando o modo de chaveamento. O *deadlock* é eliminado graças à replicação dos canais de consumo. Essa característica possibilita a transmissão simultânea de mensagens *multicast*. A versão orientada a conexão do algoritmo, proposta neste trabalho, é uma contribuição tanto para a área de NoCs quanto para a área de multicomputadores, enquanto que a adaptação do algoritmo é uma contribuição específica para a área de NoCs.

A versão do algoritmo usando chaveamento por pacotes, mesmo enviando até duas cópias da mensagem, é superior à versão orientada a conexão devido a desnecessidade de estabelecimento de conexão. A versão orientada a conexão apresenta um desempenho aceitável somente em baixas taxas de injeção onde o congestionamento é baixo ou até mesmo inexistente.

Três outros resultados foram apresentados. O primeiro vem da comparação entre os algoritmos *dual-path* CP e o individual, onde é possível observar a perda de desempenho proporcionada pela ausência de um serviço de *multicast* na rede. O segundo é relativo impacto do tráfego *multicast* sobre um tráfego puramente *unicast*. Utilizando o *dual-path* CP a degradação de desempenho provocada pelo tráfego *multicast*, para típicas taxas de aplicações, foi inferior a 10%. O terceiro mostrou que a energia consumida pela NoC para transmitir mensagens *multicast* usando o algoritmo *dual-path* CP é menor do que usando o algoritmo individual. Considerando que ambos algoritmos dissipam a mesma potência, o consumo de energia é proporcional ao tempo total para a transmissão das mensagens.

6 SPILLING

Uma das grandes vantagens proporcionada pelas NoCs em relação às estruturas de interconexão baseadas em barramentos é a redução da contenção, visto que uma NoC oferece um suporte maior ao paralelismo devido a disponibilidade de diversos caminhos para a comunicação entre IPs. No entanto, em situações de tráfego intenso a contenção pode ser um problema mesmo em NoCs, devido à falta de um caminho disponível entre a origem e o destino de um pacote. A contenção pode implicar conseqüências como perda de desempenho do sistema ou descarte de pacotes. O desempenho do sistema pode ser afetado devido à necessidade dos IPs concluírem a transmissão de pacotes antes de voltarem a processar novos dados. Em casos onde os IPs recebem dados de fontes externas a taxas fixas, pode haver a necessidade de descarte de pacotes. A contenção pode ser reduzida através da adoção de algoritmos de roteamento adaptativos, pois estes exploram caminhos que vão além do escopo dos caminhos explorados por algoritmos determinísticos, ao custo do possível de aumento da latência por pacote, no caso de algoritmos não mínimos.

Este capítulo propõem uma nova solução para reduzir a contenção em sistemas baseados em NoCs. Através de um módulo chamado *spilling* conectado a um roteador da NoC na forma de um IP, os demais IPs do sistema podem desviar seus pacotes para esse módulo, o qual se encarrega de retransmiti-los para os seus devidos destinos. Essa solução pode ser potencializada através da adoção de um algoritmo de roteamento adaptativo.

6.1 Descrição do Módulo *Spilling*

O módulo *spilling* atua como um intermediário entre origem e destino, e pode ser utilizado na ausência de um caminho disponível entre estes. O módulo possui uma memória de grande capacidade para o armazenamento temporário dos pacotes. Essa memória é segmentada em páginas, as quais são acessadas na forma de FIFOs. Cada página armazena pacotes de um mesmo IP origem. Na medida em que os pacotes são armazenados, o módulo tenta retransmiti-los para os seus destinos. Os pacotes são armazenados juntamente com a identificação do seu destino.

A Figura 68 ilustra a arquitetura do módulo *spilling*. Uma FIFO (*LUT ram3*) é utilizada para o ajuste de taxas entre a NoC e a memória SDRAM. Visto que a SDRAM não permite leitura e escrita simultânea, somente uma FIFO multiplexada é suficiente para o ajuste de taxas tanto na escrita quanto na leitura. As *LUT rams 1* e *2* armazenam os ponteiros *first* e *last* das FIFOS implementadas nas páginas da SDRAM. As operações de leitura e de escrita na SDRAM são executadas pelo módulo SDRAM Ctrl. Quando o módulo *spilling* recebe um pacote, o modo de escrita é ativado *(rd_wr = '0'*). Neste modo o multiplexador *MUX1* seleciona como entrada da FIFO (*LUT ram3*) os dados oriundos da rede e o *MUX2* seleciona como endereço da SDRAM o ponteiro *last* da FIFO correspondente à origem do pacote. Sempre que houver um pacote armazenado em alguma das FIFOs da SDRAM, o módulo *spilling* tenta retransmiti-lo para o seu

destino. O modo de leitura é ativado (*rd_wr = '1'*) para a retransmissão de pacotes. Neste modo o multiplexador *MUX1* seleciona como entrada da FIFO (*LUT ram3*) os dados oriundos do SDRAM Ctrl (*data_out*) e o *MUX2* seleciona como endereço da SDRAM o ponteiro *first* correspondente a uma das FIFOs da SDRAM que tenha algum pacote para ser retransmitido. Visto que a capacidade de armazenamento da SDRAM é grande, a recepção de pacotes tem prioridade sobre a retransmissão. Essa priorização contribui para a redução da contenção.

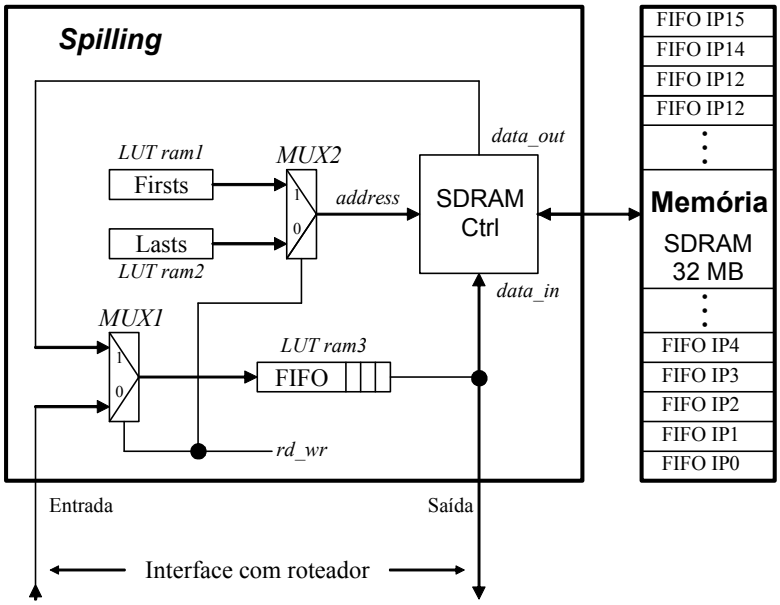


Figura 68 – Arquitetura do módulo *spilling*.

Ao desviar pacotes para o módulo *spilling*, os pacotes oriundos de uma mesma origem com destino fixo, podem chegar fora de ordem no destino. Considere por exemplo uma origem qualquer que tenha cinco pacotes endereçados para o mesmo destino. Supondo que por falta de disponibilidade de um caminho até o destino, os três primeiros pacotes foram desviados para o módulo *spilling*. Os dois pacotes restantes a origem conseguiu enviar diretamente para o destino. Como resultado, os dois últimos pacotes podem ter chegado ao destino antes do módulo *spilling* retransmitir os três primeiros. Para evitar esse desordenamento de pacotes, a origem deve continuar a desviar pacotes para o módulo *spilling* até que este envie uma notificação indicando que todos os pacotes já foram retransmitidos. Sempre que o módulo *spilling* retransmite todos pacotes armazenados em alguma das FIFOs da SDRAM, uma notificação é enviada para a origem correspondente à FIFO recém esvaziada.

6.2 Validação

O módulo *spilling* foi descrito em VHDL e validado por simulação funcional e prototipação em FPGA. Para a simulação funcional foi utilizada uma descrição VHDL da memória SDRAM. A Figura 69 ilustra um cenário no qual o IP4 tem pacotes endereçados para o IP10. No entanto os pacotes são enviados para o módulo *spilling* retransmiti-los. O cenário apresentado não apresenta contenção, o que justificaria o desvio para o módulo *spilling*. O objetivo é apenas validar

o módulo *spilling*, ilustrando a recepção e retransmissão de um pacote. A seguir serão apresentadas as formas de onda correspondentes à recepção (Figura 70) e à retransmissão (Figura 71) de um pacote pelo módulo *spilling*. As formas de onda apresentam as interfaces de entrada e saída do módulo *spilling*. A descrição dos passos da simulação segue às figuras.

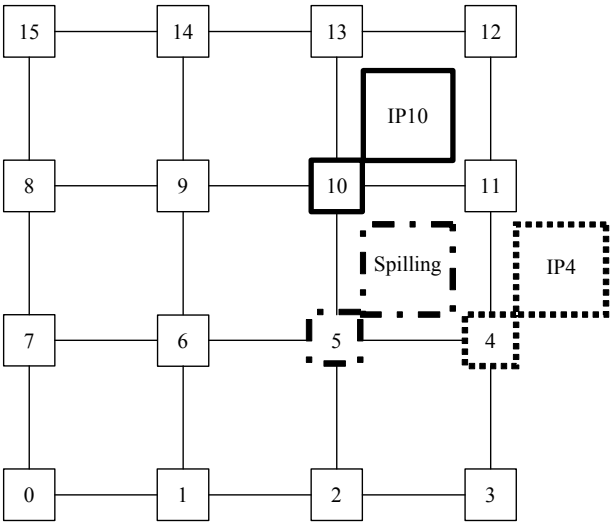


Figura 69 – Módulo *spilling* como intermediário entre os IPs 4 e 10.

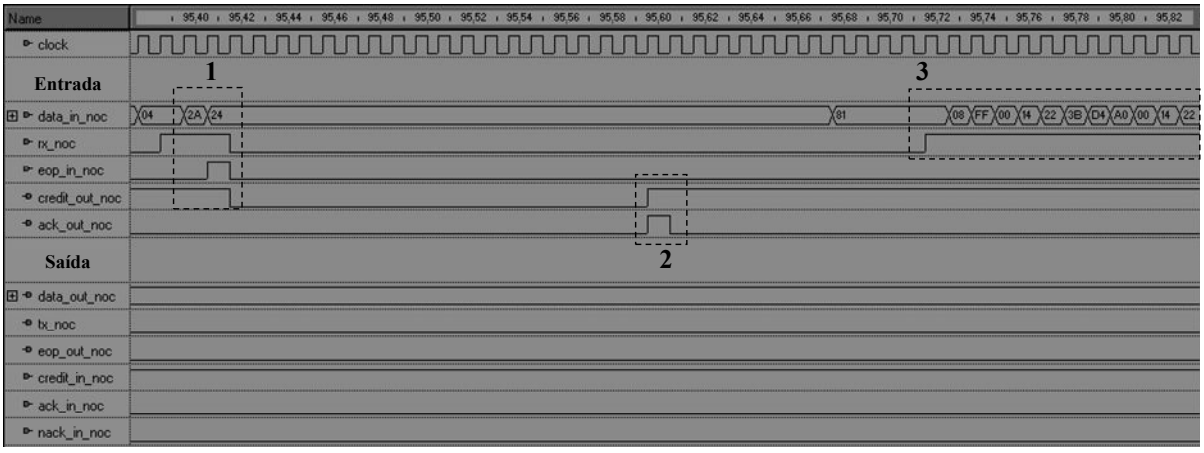


Figura 70 – Recepção de pacote pelo módulo *spilling*.

1. O módulo *spilling* recebe um estabelecimento de conexão indicando o destino do pacote ("*0xA*" = 10) e a origem ("*0x4*"). O valor "2" que precede o destino e a origem faz parte do protocolo de estabelecimento de conexão. Em seguida essa informação é armazenada na SDRAM (tempo entre os passos 1 e 2).
2. Concluído o armazenamento do cabeçalho do pacote na SDRAM, o módulo *spilling* sinaliza através dos sinais *ack_out_noc = '1'* e *credit_out_noc = '1'* que está pronto para receber o pacote.
3. O módulo *spilling* começa a receber o pacote, o qual vai sendo armazenado na SDRAM na medida em que é recebido.

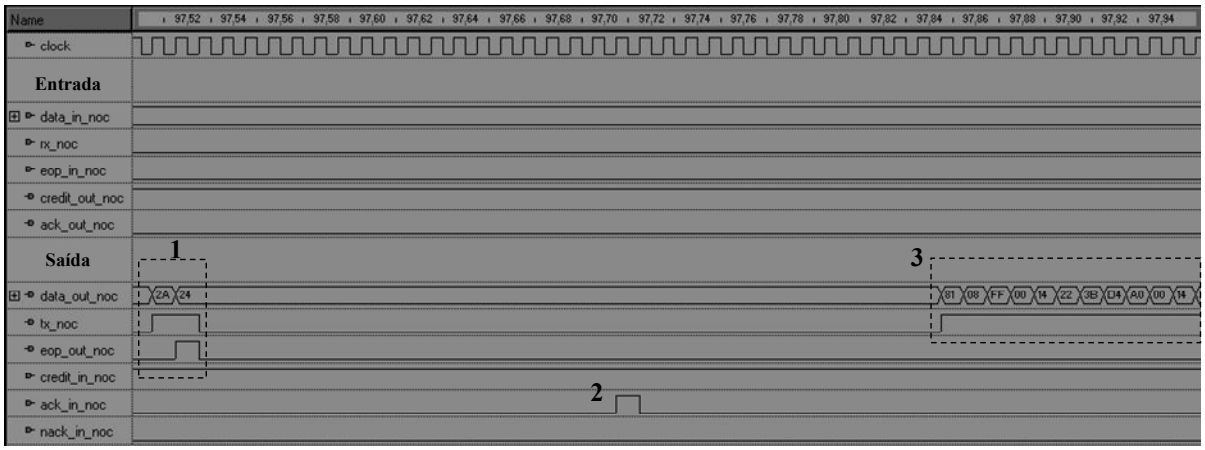


Figura 71 – Retransmissão de pacote pelo módulo *spilling*.

1. O módulo *spilling* tenta estabelecer uma conexão com o destino do pacote.
2. O módulo *spilling* recebe através do sinal *ack_in_noc = '1'* a indicação de que a conexão está estabelecida. Em seguida busca o pacote na SDRAM (tempo entre os passos 2 e 3).
3. O pacote é retransmitido para o destino na medida em que é lido da SDRAM.

6.3 Avaliação de Desempenho e Consumo de Área

Esta seção apresenta o ganho de desempenho, em termos de descarte de pacotes, proporcionado pela adição do módulo *spilling* ao sistema e o consumo de área do módulo implementado. A NoC utilizada tem as características apresentadas na Tabela 12, com *buffers* de entrada de dezesseis posições. Este experimento utiliza uma instância da arquitetura MOTIM (Capítulo 1, Seção 1.1.1, Figura 2), a qual implementa canais replicados (grau de replicação igual a 2) e controle de sessão (3 sessões por roteador).

Tabela 12 – Características da NoC utilizada.

Tamanho de Flit/phit	8 bits
Controle de fluxo	Baseado em créditos
Topologia da NoC	Malha 4x4
Algoritmo de roteamento	Hamiltoniano Determinístico
Modo de chaveamento	Pacotes/*Wormhole* e Circuito

A Figura 72 ilustra o cenário utilizado para a avaliação do desempenho em função da taxa fixa de recepção. Os IPs 0, 1, 4, 12, 13 e 14 recebem dados a uma taxa fixa de uma fonte externa, indicada pelas setas. Cada IP recebe 10 pacotes Ethernet de 500 bytes, os quais são divididos em células de 128 bytes e armazenados temporariamente em um *buffer* antes do envio. Os pacotes recebidos pelos IPs têm como destino o IP10. Esse cenário visa criar um *hot spot* no IP10, aumentando a contenção da rede. Os IPs desviam pacotes para o módulo *spilling* quando o *buffer* para armazenamento temporário atinge 80% de ocupação e não é possível estabelecer uma conexão com o destino.

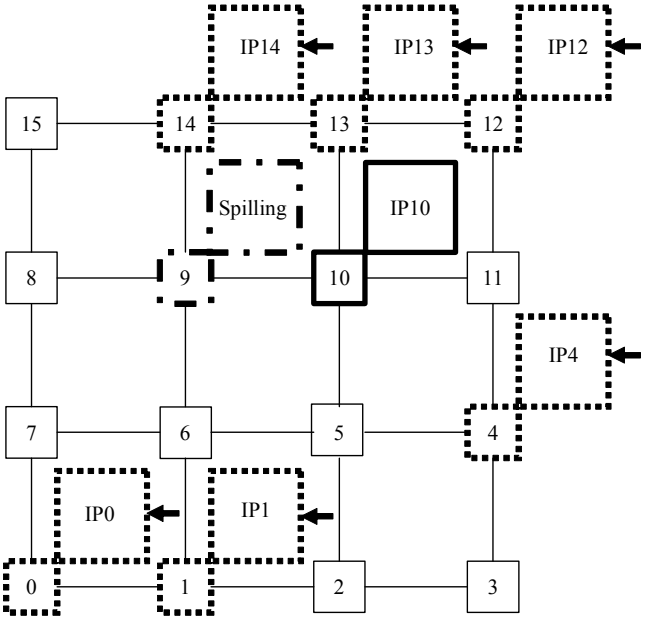


Figura 72 - Cenário para avaliação de desempenho em função da taxa fixa de recepção.

A taxa fixa de recepção pelos IPs varia de 4% a 12,5% da taxa do canal de recepção (seta chegando nos IPs na Figura 72). O descarte de pacotes ocorre quando o IP não tem mais espaço em *buffer* para o armazenamento temporário dos pacotes Ethernet. A Tabela 13 apresenta o número total de pacotes descartados com e sem o uso do módulo *spilling*. Um total de 60 pacotes é destinado ao IP10. A uma taxa fixa de recepção de 12,5% e sem o uso do módulo *spilling*, observa-se o descarte de mais de 50% dos pacotes destinados ao IP10. A partir do uso do módulo *spilling* o descarte de pacotes é reduzido para 15%. Observe que mesmo nas taxas fixas de recepção mais baixas existe o descarte de pacotes.

Tabela 13 – Descarte de pacotes.

Taxa fixa de recepção	Pacotes descartados Sem *spilling*	Pacotes descartados Com *spilling*
4%	10	2
6%	19	4
8%	26	5
10%	27	7
12,5%	31	9

6.3.1 Consumo de Área

A Tabela 14 apresenta a área de FPGA consumida pela implementação do módulo *spilling*. Além dos componentes apresentados na Figura 68, a implementação inclui também máquinas de estados para envio e recepção de pacotes, controle de acesso à FIFO (*LUT ram3*), controle de *refresh* da SDRAM, etc. A implementação avaliada segmenta a SDRAM em 32 páginas, sendo necessário o armazenamento de 32 ponteiros *first* e 32 ponteiros *last*, todos de 18 bits.

Tabela 14 - Resultados de área para a implementação do módulo *spilling*, usando o dispositivo Virtex 2VP30.

Recurso	Módulo *spilling*	Disponibilidade
Slices	809	13696
LUTs	1385	27392
Flip Flops	493	29060

6.4 Conclusões do Capítulo

Esse capítulo apresentou uma nova proposta para a redução da contenção em redes intra-chip a partir de um intermediário entre origem e destino. Os resultados apresentados mostraram uma significativa redução de contenção, tornando a proposta uma atrativa solução para o tratamento de *hot spots*. Contudo, o próprio módulo *spilling* pode acabar se tornando um *hot spot*. Para contornar esse problema e reduzir ainda mais a contenção, novos módulos de *spilling* podem ser adicionados ao sistema. Esses módulos podem ser utilizados por todos os IPs do sistema ou pode-se adotar uma abordagem semelhante a estruturas de memória heterogênea em MPSoCs, onde cada módulo *spilling* é utilizado por um conjunto pré-determinado de IPs. Essa proposta pode ser considerada como uma contribuição original e eficiente para a área de NoCs.

7 CONCLUSÕES E TRABALHOS FUTUROS

Quaisquer avanços relativos à otimização de desempenho são sempre relevantes em qualquer área da computação, em especial na área de sistemas digitais, onde as restrições temporais são cada vez mais estritas. Tendo em vista que as arquiteturas monoprocessados já não suportam mais o poder computacional exigido por diversas aplicações contemporâneas, elas estão dando lugar às arquiteturas multiprocessadas. Graças à computação paralela real proporcionada por essas arquiteturas, é possível atingir níveis de processamento que satisfazem um amplo conjunto de aplicações modernas. Em relação ao desempenho individual dos elementos de processamento, inúmeros avanços têm sidos alcançados ao longo dos anos, implicando módulos de hardware dedicados e/ou reconfiguráveis. Contudo, o gargalo das arquiteturas multiprocessadas reside justamente na comunicação entre os diversos elementos de processamento. Devido principalmente ao baixo suporte a transações paralelas, estruturas de comunicação baseadas em barramentos não são as mais adequadas para esse tipo de arquitetura, no entanto são um legado que aos poucos vem sendo substituído por NoCs.

A presente dissertação apresentou vários mecanismos para otimizar o desempenho de redes intra-chip. Em relação à largura de banda, a *replicação de canais físicos* mostrou-se uma solução eficiente em relação ao consumo de área e principalmente em relação ao desempenho, quando comparada à multiplexação dos canais físicos (canais virtuais). A largura de banda da rede foi duplicada a um custo de área inferior à técnica de canais virtuais. Graças aos avanços nas tecnologias submicrônicas, a replicação de canais físicos se mostra uma abordagem mais atual e adequada às tendências tecnológicas. Como trabalho futuro pretende-se avaliar o impacto da replicação de canais físicos no consumo de potência, visando obter-se o melhor compromisso entre grau de replicação e consumo.

Em vista da discrepância de taxas de transmissão entre aplicação e NoC, foi proposto um método de transmissão baseado em *chaveamento por circuito e nível de sessão*. A partir desse método, o desempenho da rede pode ser otimizado apenas realizando um melhor aproveitamento dos seus recursos. Esse método tem como base a maximização dos recursos através de transmissões em rajada e curta alocação de recursos. Alterações arquiteturais relevantes foram feitas apenas fora da rede, onde *buffers* de sessão foram adicionados em vista de obter-se melhor desempenho em casos de múltiplas origens enviando dados a um mesmo destino.

Outra abordagem que procurou otimizar o desempenho da rede a partir de um melhor aproveitamento dos recursos foi a *aptatividade do roteamento em função do tráfego*. Algoritmos de roteamento adaptativos tendem por natureza explorar caminhos que vão além do escopo dos algoritmos de roteamento determinísticos, desta maneira otimizando o desempenho geral do sistema. Essa abordagem mostrou-se eficiente por adicionar um nível de inteligência aos algoritmos de roteamento adaptativos, através da seleção de uma porta de saída baseada nas condições de congestionamento de roteadores vizinhos. Assim, regiões congestionadas tendem a ser evitadas e o

tráfego pode ser distribuído na rede a fim de evitar *hot spots*. Como trabalho futuro pretende-se empregar a versão parcialmente adaptativa em função do tráfego na implementação do algoritmo de *multicast dual-path*.

Para manter a escalabilidade da rede em relação à transmissão de pacotes, foi implementado o algoritmo de *multicast dual-path*. Esse algoritmo, oriundo da área de multicomputadores, foi adaptado para NoCs em duas versões livre de *deadlock*: (*i*) utilizando chaveamento por pacotes e (*ii*) utilizando chaveamento por circuito. A versão orientada a conexão mostrou-se pouco eficiente devido ao alto tempo gasto no estabelecimento da conexão antes da transmissão da mensagem, entretanto apresentou um desempenho aceitável em situações de baixo tráfego. Já a versão com chaveamento por pacotes mostrou-se bastante eficiente, tanto em situações de tráfego intenso quanto de baixo tráfego. Mostrou-se também que o uso de serviço *multicast* resulta na redução da energia consumida quando comparado ao envio de múltiplas cópias usando *unicast*. Trabalhos futuros incluem: (*i*) avaliar o desempenho dos algoritmos *multicast* em aplicações paralelas, substituindo os IPs geradores de tráfego por elementos de processamento e *(ii*) substituir o algoritmo de roteamento Hamiltoniano determinístico pelo parcialmente adaptativo em função do tráfego (Capítulo 3).

Para minimizar a contenção, foi proposta a adoção de um módulo intermediário entre origem e destino chamado de *spilling*. Através desse módulo os elemento de processamento passaram a ter um destino alternativo para os pacotes a serem enviados, quando não conseguirem um caminho disponível até o destino. O módulo *spilling* armazena temporariamente os pacotes e se encarrega da retransmissão dos mesmos quando houver disponibilidade de caminhos até os destinos. A partir desse destino alternativo, obteve-se uma significativa redução na contenção de pacotes, o que pôde ser comprovado a partir da redução no número de pacotes descartados. Esse mecanismo otimiza o desempenho geral do sistema sem a necessidade de alterações arquiteturais na rede.

NoCs ainda não são as estruturas de interconexão predominantes em sistemas multiprocessados, apesar de claramente serem as mais adequadas. Isto deve-se principalmente a três questões: (*i*) latência, (*ii*) potência e (*iii*) ferramentas de CAD [DAL06]. A partir dos mecanismos propostos, objetiva-se que o desempenho geral das arquiteturas multiprocessadas atinja níveis elevados de desempenho, contribuindo para que NoCs tornem-se a estrutura de conexão predominante nos sistemas digitas modernos. Os mecanismos apresentados podem ainda servir para dar suporte a QoS, proporcionado às aplicações plena capacidade de atingir seus requisitos de desempenho.

Como trabalho futuro, a longo prazo, pretende-se criar uma infra-estrutura envolvendo hardware (interface de rede) e software (*kernel*) que permita às aplicações fazerem plena utilização dos mecanismos proporcionados pela rede para cumprirem seus prazos. Para isso, o desenvolvimento de aplicações, *drivers* e interface de rede se fazem fundamentais. A partir disso, eficientes arquiteturas MPSoCs podem ser criadas e personalizadas para aplicações específicas.

REFERÊNCIAS BIBLIOGRÁFICAS

[BEN02] Benini, L.; Micheli, G. D. "*Networks on chips: A new SoC paradigm*". IEEE Computer v.35(1), 2002, pp. 70–78.

[BEN04] Benini, L.; Bertozzi, D. "*Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chip*". In: Design, Automation and Test in Europe (DATE'04), 2004, pp. 18-31.

[BEN05] Benini, L.; Bertozzi, D. "*Network-on-chip architectures and design methods*". Computers and Digital Techniques, v.152(2), 2005, pp. 261–272.

[BJE05] Bjerregaard, T.; Sparso, J. "*Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip*". In: Design, Automation and Test in Europe (DATE'05), 2005, pp. 1226-1231.

[BJE06] Bjerregaard, T.; Mahadevan, S. *"A survey of research and practices of Network-on-chip"*. ACM Computing Surveys, v.38(1), 2006, pp. 1-51.

[BOL07] Bolotin, E.; Guz, Z.; Cidon, I.; Ginosar, R.; Kolodny, A. *"The Power of Priority: NoC based Distributed Cache Coherency"*. In: First International Symposium on Networks-on-Chip (NOCS'07), 2007, pp. 117-126.

[BOP98] Boppana, R. V.; Chalasani, S.; Raghavendra, C. S. *"Resource Deadlocks and Performance of Wormhole Multicast Routing Algorithms"*. IEEE Transactions on Parallel and Distributed Systems, v.9(8), 1998, pp. 535-549.

[BOU06] Bouhraoua, A; Elrabaa, M. E. "*A High-Throughput Network-on-Chip Architecture for Systems-on-Chip Interconnect*". In: International Symposium on System-on-Chip (SOC'06), 2006, pp. 127-130.

[CHI00] Chiu, G. *"The Odd-Even Turn Model for Adaptive Routing"*. IEEE Transactions on Parallel and Distributed Systems, v.7(11), 2000, pp. 729-738.

[DAL87] Dally, W. J.; Seitz, C. L. *"Deadlock-Free Message Routing in Multiprocessors Interconnection Networks"*. IEEE Transactions on Computers v.36(5), 1987, pp.547-553.

[DAL92] Dally, W. J. "*Virtual-Channel Flow Control*". IEEE Transactions on Parallel and Distributed Systems, v.3(2), 1992, pp. 194-205.

[DAL01] Dally, W. J.; Towles, B. "*Route packets, not wires: On-chip interconnection networks*". In: Design Automation Conference (DAC'01), 2001, pp. 684–689.

[DAL06] Dally, W.; "*DRAFT Final Report: Workshop on On- and Off-Chip Networks for Multi-Core Systems*". Capturado em: www.ece.ucdavis.edu/~ocin06/index.html, julho 2006.

[DAN06] Daneshtalab, M.; Afzali-Kusha, A.; Mohammadi. S. "*Minimizing Hot Spots in NoCs through a Dynamic Routing Algorithm based on Input and Output Selections*". In: International Symposium on System-on-Chip (SOC'06), 2006, pp. 49-52.

[DAY83] Day, J. D.; Zimmermman, H. "*The OSI reference model*". Proceedings of IEEE, v.71(12), 1983, pp. 1334-1340.

[DEH06] Dehyadgari, M.; et al. "*A new protocol stack model for network on chip*". In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI'06), 2006, pp. 440-441.

[GOO05] Goossens, K.; et al. "*Æthereal network-on-chip: concepts, architectures, and implementations*". IEEE Design & Test of Computers, v.22(5), 2005, pp. 414-421.

[GUE00] Guerrier, P.; Greiner, A. "*A generic architecture for on-chip packet-switched interconnections*". In: Design, Automation and Test in Europe (DATE'00), 2000, pp. 250–256.

[HAR72] Harary, F. *"Graph Theory"*. Addison-Wesley, 1972, 274p.

[HIL05] Hilton, C.; Nelson, B. "*A flexible circuit-switched NoC for FPGA-based systems*". Field Programmable Logic and Applications, v.24(26), 2005, pp. 191-196.

[HIL06] Hilton, C.; Nelson, B. "*PNoC: a flexible circuit-switched NoC for FPGA-based systems*". Computers and Digital Techniques, v.153(3), May 2006, pp. 181-188.

[HU04] Hu, J.; Marculescu, R. "*DyAD-Smart Routing for Networks-on-Chip*". In: Design Automation Conference (DAC'04), 2004, pp. 260-263.

[JAN03] Jantsch, A.; Tenhunen, H. "*Networks on Chip*". Kluwer Academic Publishers, 2003, 303p.

[JUN07] Bahn, J. H.; Lee, S. E.; Bagherzadeh, N. *"On Design and Analysis of a Feasible Network-on-Chip (NoC) Architecture"*. In: International Conference on Information Technology (ICIT'07), 2007, pp. 1033-1038.

[KAR02] Karim, F.; Nguyen, A.; Dey, S. "*An Interconnect Architecture for Networking Systems on Chips*". IEEE Micro, v.22(5), 2002, pp. 36-45.

[LAH01] Lahiri, K.; Raghunathan, A.; Dey, S. "*Evaluation of the traffic-performance characteristics of system-on-chip communication architectures*". In: International Conference on VLSI Design (VLSI Design'01), 2001, pp. 29-35.

[LEI06] Leibson, S. "*The Future of Nanometer SOC Design*". In: International Symposium on Systems-on-Chip (SOC'06), 2006, pp. 1-6.

[LER05] Leroy, A.; et al. *"Spatial Division Multiplexing: a Novel Approach for Guaranteed Throughput on NoCs*". In: Hardware/Software Codesign and System Synthesis (CODES+ISSS'05), 2005, pp. 81-86.

[LI06] Li, M.; Zeng, Q.; Jone, W. *"DyXY – A Proximity Congestion-Aware Deadlock-Free Dynamic Routing Method for Networks on Chip"*. In: Design Automation Conference (DAC'06), 2006, pp. 849-852.

[LIN94] Lin, X.; McKinley, P. K.; Ni, L. M. "*Deadlock-free multicast wormhole routing in 2-D mesh multicomputers*". IEEE Transactions on Parallel and Distributed Systems, v.5(8), 1994, pp. 793-804.

[LU06] Lu, Z.; Yin, B.; Jantsch, A.*"Connection-oriented Multicasting in Wormhole-switched Networks on Chip*". In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI'06), 2006, pp. 205-210.

[MEL05] Mello, A.; Tedesco, L.; Calazans, N.; Moraes, F.; "*Virtual Channels in Networks on Chip: Implementantion and Evaluation on Hermes NoC"*. In: Symposium on Integrated Circuits and Systems (SBCCI'05), 2005, pp. 178-183.

[MIL04] Millberg, M.; Nilsson, E.; Thid, R.; Jantsch, A. "*Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks within the Nostrum Network-on-Chip*". In: Design, Automation and Test in Europe (DATE'04), 2004, pp. 890-895.

[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, v.38(1), 2004, pp. 69-93.

[MUL04] Mullins, R.; West, A.; Moore, S. "*Low-Latency Virtual-Channel Routers for On-Chip Networks*". In: International Symposium on Computer Architecture (ISCA'04), 2004, pp. 188-197.

[NI93] Ni, L. M.; McKinley, P. K. "*A Survey of Wormhole Routing Techniques in Direct Networks"*. IEEE Computer Magazine, v.26(2), 1993, pp. 62-76.

[NAR05] Narasimhan, A.; Srinivasan, K.; Sridhar, R. *"A High-Performance Router Design for VDSM NoCs*". In: IEEE International SOC Conference (SOCC'05), 2005, pp. 301-304.

[NGU01] Nguyen, K.; Mohor, I.; Markovic T. "*Ethernet MAC 10/100Mbps*". Capturado em: http://www.opencores.org/projects.cgi/web/ethmac/overview, julho 2001.

[SET06a] Sethuraman, B.; Vemuri, R. *"optiMap: a tool for automated generation of NoC architectures using multi-port routers for FPGAs"*. In: Design, Automation and Test in Europe (DATE'06), 2006, pp. 947-952.

[SET06b] Sethuraman, B.; Vemuri, R. "*Multi2 Router: A Novel Multi Local Port Router Architecture with Broadcast Facility for FPGA-Based Networks-on-Chip*". In: International Conference on Field Programmable Logic and Applications (FPL'06), 2006, pp. 1-4.

[SOB06] Sobhani, A.; Daneshtalab, M.; Neishaburi, M.H.; Mottaghi, M.D.; Afzali-Kusha, A.; Fatemi, O.; Navabi, Z. "*Dynamic Routing Algorithm for Avoiding Hot Spots in On-chip Networks*". In: International Conference on Design and Test of Integrated Systems in Nanoscale Technology (DTIS'06), 2006, pp. 179-183.

[TET06] TETHA. "*Projeto TETHA*". Capturado em: http://www.inf.pucrs.br/~gaph/tetha, dezembro 2006.

[WIK03] Wiklund, D.; Liu, D. "*SoCBUS: Switched Network-on-Chip for Hard Real Time Embedded Systems*". In: International Parallel and Distributed Processing Symposium (IPDPS'03), 2003, pp. 113-116.

[WOL05] Wolkotte, P. T.; et al. *"An Energy-Efficient Reconfigurable Circuit-Switched Network-on-Chip*". In: International Parallel and Distributed Processing Symposium (IPDPS'05), 2005.

[ZEF03] Zeferino, C. A.; Susin, A. A. *"SoCIN: a Parametric and Scalable Network-on-Chip*". In: Symposium on Integrated Circuits and Systems (SBCCI'03), 2003, pp. 169-174.