UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

EDGARD DE FARIA CORRÊA

Redes-em-Chip para Sistemas Embarcados Visando a Otimização de Medidas de Qualidade de Serviço para Aplicações de Tempo Real.

Tese apresentada como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação

Prof. Dr. Luigi Carro
Orientador

Prof. Dr. Flávio Rech Wagner
Co-orientador

Porto Alegre, maio de 2007.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Corrêa, Edgard de Faria

Redes-em-Chip para Sistemas Embarcados Visando a Otimização de Medidas de Qualidade de Serviço para Aplicações de Tempo Real. / Edgard de Faria Corrêa – Porto Alegre: Programa de Pós-Graduação em Computação da UFRGS, 2007.

131 f.:il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2007. Orientador: Luigi Carro; Co-orientador: Flávio Rech Wagner.

1.Sistemas Embarcados. 2. Redes-em-Chip 3. Microeletrônica. I. Carro, Luigi. II. Wagner, Flávio Rech. III. Título.

AGRADECIMENTOS

Quando se embarca em novos projetos, em novos mares, é preciso determinação e motivação para se conseguir atingir os objetivos. Entretanto, somente isso não é suficiente. Para conseguir concluir este trabalho tive o valioso apoio de algumas pessoas e instituições.

Minha família me ensinou a nadar, a perder o medo de navegar. Meu pai, Francisco Ivanir de Araújo Corrêa, minha mãe, Zelia Faria Corrêa, meu irmão, Guilherme Augusto de Faria Corrêa e minhas irmãs, Tereza Odete de Faria Corrêa e Ana Lúcia de Faria Corrêa, me incentivaram a lançar-me por mares, mesmo que distantes de onde eles ficariam. Isso possibilitou aventurar-me mundo afora, com a certeza que ao voltar teria um ancoradouro seguro.

Para me lançar por esses mares, necessitava de apoio e financiamento na "construção/manutenção" do barco, o que foi proporcionado pela UFRN e pela CAPES.

A UFRGS, através do Instituto de Informática, do GME e do LSE ofereceu "bússola, GPS e rota de navegação" com recursos de nível internacional, um ótimo ambiente de trabalho e profissionais competentes e atenciosos: Ida Rossi, Silvania Vidal de Azevedo, Luis Otávio Luz Soares, Beatriz Regina Bastos Haro, Ângela Regina Rosa da Silva, Sulamar Figueira Marcelino, Eliane Ricardo Iranço, Elisiane da Silveira Ribeiro e Lesley Barbosa Lopes.

E quando o "barco" esteve do outro lado do Atlântico, durante o estágio sanduíche em Paris, o apoio da equipe do ASIM do LIP6/UPMC foi fundamental.

Ao longo da navegação, muitas correções de rota foram necessárias. As sugestões de professores durante bancas e apresentações foi essencial. Agradeço, em especial, a: César Albenes Zeferino, Ivan Saraiva Silva, Carlos Eduardo Pereira, Altamiro Amadeu Susin e Érika Fernandes Cota.

Nas tempestades ou calmarias, onde parecia que o "barco" não chegaria a lugar nenhum, muitos me incentivaram a seguir em frente. Poucas páginas não seriam suficientes para elencar todos eles. Deixo o registro de alguns, que nos momentos mais críticos, me ajudaram a continuar a jornada: Rodrigo da Silva Cardozo, Alexandre Irigon Gervini, Márcio Eduardo Kreutz, César Augusto Missio Marcon e Eduardo Wenzel Brião.

Nos momentos finais, quando os problemas sempre se agigantam, o apoio para finalizar este trabalho, evitando que, depois de tantos meses, eu terminasse "sem chegar a uma praia", veio especialmente de: Leonardo Alves de Paula e Silva, Eduardo Wisnieski Basso, Julio Carlos Balzano de Mattos e Rafael Luiz Marques Ary.

Apesar de estar em “terras estrangeiras”, amigos com os quais contei neste período, conseguiram suavizar a saudade de casa e pude vencer as “ondas que ameaçavam o barco” ao longo desse tempo. Não conseguirei citar todos, mas preciso agradecer a: Gabriela Conceição e família, Viviana Cocco Mariani e Leandro dos Santos Coelho, Gizele Aparecida de Oliveira de Paula e Silva, Adja Ferreira de Andrade, Michael da Costa Móra, Maria Vitoria Kessler Sá Brito e Janaina Sá Brito, Marcio Bystronski, Cristina Meinhardt, Arnaldo Pereira de Azevedo Filho, Lucas Brusamarello, Digeorgia Natalie da Silva, Lisane Brisolara de Brisolara, André Borin Soares e Janaina, Rodrigo Bittencourt Motta, Karla Paloma Costa Bacalhau, Amélia Carla Sobrinho Bifano, Christina Bolcskei, Lucimara Leite, Ligia Amparo, Cláudia da Silva, Rawad Zgheib, Igor Houwat, Daniele Patricia Pontes Paiva e Adriano Cavalcanti Filho.

E, no tempo que estive longe, em outros portos, amigos cuidaram, durante a minha ausência, para que pudesse voltar com tranqüilidade. Meu agradecimento também aos que deixaram o meu retorno mais tranqüilo: Marcilia de Lima Verde da Silva, Mirian Dantas dos Santos, Solange Álvares dos Santos, Pollyanna Sousa, Paulo Roberto Miranda Meirelles, André Luis Leite e Munique Therense Costa de Morais.

Finalmente, agradeço aos responsáveis por eu ter conseguido concluir esse projeto. Eles me orientaram em todos os momentos. Luigi Carro e Flávio Rech Wagner me ensinaram que, ética, participação ativa, respeito às capacidades e limites dos orientados, são elementos importantes para se conseguir sucesso numa empreitada desse porte. Essa convivência me encorajou a continuar embarcando em mais viagens através dos mares.

Edgard de Faria Corrêa

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ACP	*Application Communication Pattern*
APCG	*APplication Characterization Graph*
API	*Application Program Interface*
ASIM	*Architecture des Systèmes Intégrés & Micro-électronique*
ASIP	*Application Specific Instruction-Set Processor*
aSoC	*adaptative SoC*
BIST	*Built-In Self-Test*
BOP	*Begin of Packet*
bps	bits por segundo
CBDA	*Centrally-Buffered, Dynamically-Allocated*
CDCM	*Communication Dependence and Computation Model*
CPU	*Central Processing Unit*
CTG	*Communication Task Graph*
CWG	*Communication Weighted Graph*
CWM	*Communication Weighted Model*
DAMQ	*Dynamically-Allocated, Multi-Queue*
DMA	*Direct Memory Access*
DSP	*Digital Signal Processing*
EOP	*End of Packet*
FCFS	*First Come, First Served*
FIFO	*First In, First Out*
FLIT	*FLow control unIT*
FPGA	*Field-Programmable Gate Array*
GALS	*Globally Asynchronous and Locally Synchronous*
GME	Grupo de Microeletrônica
IP	*Intellectual Property*

ITRS	*International Technology Roadmap for Semiconductors*
JPEG	*Joint Photographic Experts Group*
LIP6	*Laboratoire d'Informatique de Paris 6*
LRS	*Least Recently Served*
LS	*List Scheduling*
LSE	Laboratório de Sistemas Embarcados
MILP	*Mixed-Integer Linear Programming*
MJPEG	*Motion* JPEG
MP-SoC	Multi Processor SoC
MWMR	*Multi Writer Multi Reader*
NoC	N*etwork-on-Chip*
PE	*Processing Elements*
PHIT	*PHysical unIT*
POSIX	*Portable Operating System Interface for UNIX*
PowerPC	*Performance Optimization With Enhanced RISC*
PPGC	Programa de Pós-Graduação em Computação
QoS	*Quality of Service*
RISC	Reduced Instruction Set Computer
RTL	*Register Transfer Level*
RTOS	*Real Time Operating Systems*
SAFC	*Statically Allocated, Fully Connected*
SAMQ	*Statically Allocated Multi-Queue*
SoC	*System-on-Chip*
SoCIN	*System-on-Chip Interconnection Network*
SoCLib	*SoC Library*
SPIN	*Scalable Programmable Integrated Network*
TDMA	*Time Division Multiple Access*
UFRGS	Universidade Federal do Rio Grande do Sul
UPMC	*Université Pierre et Marie Curie*
VLIW	*Very Long Instruction Word*
WCET	*Worst Case Execution Time*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O avanço da tecnologia, com a possibilidade de inclusão de um número cada vez maior de transistores em uma única pastilha de silício, tem permitido integração de diversos blocos, formando sistemas completos em um único chip. Esses sistemas em chip possuem uma maior capacidade, mas também uma maior complexidade de projeto. Um dos aspectos a ser resolvido no projeto é que infra-estrutura de comunicação será utilizada na interconexão dos diversos blocos do sistema. Nos últimos anos, as propostas têm apontado para a utilização de redes em chip (NoC – do inglês, *Network on Chip*) para solucionar este problema de comunicação. Essas redes possuem capacidade de reuso de componentes, escalabilidade, paralelismo, embora apresentem maiores custos e latência que outras soluções. Entretanto, a latência pode ser atenuada, em alguns casos, através de ajustes na configuração da rede, tais como: topologia, arbitragem, mecanismos de controle de fluxo, política de roteamento, tamanho dos *buffers*. Por outro lado, os sistemas embarcados apresentam, geralmente, requisitos cada vez mais rígidos em relação à qualidade de serviço (QoS – do inglês, *Quality of Service*) e a restrições temporais. Dessa forma, esses requisitos temporais e de QoS aumentam ainda mais a complexidade do projeto de sistemas embarcados. Em virtude desse aumento da complexidade, o ideal é que a exploração do espaço de projeto seja feita no nível de abstração mais alto possível. Com isso, espera-se manter o tempo de projeto dentro dos níveis adequados, além de permitir uma exploração de espaço de projeto mais ampla e rápida. Nessa exploração, a configuração da rede têm impacto direto sobre os requisitos temporais e de QoS. Esta tese situa-se no contexto de investigar a influência da estrutura de comunicação no atendimento aos requisitos de QoS das aplicações de tempo real. Frente aos requisitos dessas aplicações, especificamente em relação ao atendimento dos *deadlines* das tarefas e a latência das comunicações, este trabalho apresenta mecanismos de ajustes no planejamento e configuração da NoC em sistemas embarcados, objetivando a garantia desses requisitos. As estratégias utilizadas nos ajustes das características da NoC objetivam permitir o uso mínimo de recursos para atender os requisitos das aplicações de tempo real, dentro das exigências de QoS. Os resultados apresentados comprovam que o ajuste correto nos parâmetros da estrutura de comunicação tem impacto direto no desempenho do sistema, especificamente em relação ao atendimento dos *deadlines* das mensagens e na redução da latência das comunicações.

Palavras-Chave: Sistemas embarcados, redes em chip, qualidade de serviço, tempo real, microeletrônica.

Networks on Chip in Embedded Systems for Optimization of Quality of Service Measurement for Real Time Applications

ABSTRACT

With the technology advancing, a huge number of transistors can be included in a single chip. As a consequence, it is possible to integrate many blocks to build a complete system on a chip (SoC). These SoCs have more capacity, but their designs are more complex. One of the problems to solve is the design of the communication infrastructure to interconnect the systems blocks. In the last years, the utilization of networks as a solution for the communication problem has been proposed. These Networks-on-Chip (NoCs) have some interesting characteristics, such as reuse of components, scalability, and parallelism. On the other side, NoCs have higher costs and latency if compared to others solutions. The latency can be reduced, in some cases, by the adaptation of the network configuration, for instance adjusting topology, arbitration, flow control mechanisms, routing policy, size of buffers, etc. However, in general, embedded systems have increasingly rigid requirements regarding quality of service (QoS) and timing constraints. These timing and QoS requirements increase the complexity of embedded systems design. Due to this increased complexity, it is better that the design space exploration is performed at the highest possible abstraction level. With this, it is expected that the design time can be kept within adequate values, besides allowing a faster and broader design space exploration. In this exploration, the network configuration has direct impact upon timing and QoS requirements. The context of this thesis is the investigation of the influence of the communication structure on meeting QoS requirements in real time applications, in particular with respect to the fulfillment of task deadlines and latencies. This work shows mechanisms for adaptation of the NoC configuration for embedded systems, in order to meet the application requirements. The strategies used in the adjustment of the NoC characteristics allow the minimum use of resources to meet the real time application constraints, among the QoS requirements. The presented results demonstrate that the correct adjustment in the communication structure parameters has direct impact on the system performance, specifically with respect to the fulfillment of message deadlines and to the reduction of the communication latencies.

Keywords: Embedded systems, networks on chip, quality of service, real time, microelectronics.

1 INTRODUÇÃO

Atualmente, com o avanço da tecnologia, uma única pastilha de silício pode ter um grande número de transistores. Uma conseqüência disso é a possibilidade de integrar diversos blocos, formando um sistema completo em um único chip – os sistemas embarcados em chip, ou sistemas em chip (SoC – do inglês, *System on Chip*) (WOLF 2001).

Esses sistemas embarcados em um único chip utilizam cada vez mais componentes desenvolvidos em projetos anteriores ou por terceiros. O aumento nessa capacidade de integração traz novas alternativas, mas aumenta também a complexidade. Com esse aumento no número de componentes, uma questão fundamental no projeto desse tipo de sistema é a infra-estrutura de comunicação que será utilizada.

O problema, do ponto de vista da comunicação, reside no fato de que esses sistemas serão tão complexos que inviabilizarão o uso de interconexões dedicadas, pois demandando um esforço maior, dificultaria o reuso dessa abordagem. Por outro lado, os requisitos de desempenho em comunicação, como largura de banda escalável e baixa latência, dificilmente serão atendidos pelas arquiteturas baseadas em barramento, devido a limitações como a exigência de maior potência para atender um número maior de elementos conectados, e a capacitância parasita gerada por esses elementos. Além disso, existem outros problemas, como largura de banda não escalável e arbitragem centralizada, que dificultarão em muito o uso de barramentos em SoCs complexos (BENINI 2002).

Nesse contexto, uma solução proposta nos últimos anos é o uso de redes de interconexão chaveada, semelhante àquelas encontradas em computadores paralelos. Essas redes, conhecidas como redes em chip (NoCs – do inglês, *Networks-on-Chip*) (BENINI 2002), têm como vantagens a largura de banda escalável, o uso de conexões ponto-a-ponto curtas e o paralelismo na comunicação, entre outras, embora tenham como desvantagens maiores custos e latência na comunicação. Esses problemas serão certamente atenuados pela grande disponibilidade de transistores e por soluções arquiteturais que permitirão reduzir a latência da rede e seus efeitos no desempenho da aplicação. A latência na comunicação pode ser atenuada também, quando possível, através de ajustes em algumas das características da NoC, como por exemplo: política de roteamento, velocidade do roteador e tamanho dos *buffers* (ZEFERINO 2003a).

Em virtude também do aumento da complexidade, o ideal é que, a exploração do espaço de projeto de sistemas embarcados seja feita no nível de abstração mais alto

possível. Dessa forma, seria possível explorar um espaço de projeto mais amplo, sem aumentar muito o tempo de lançamento do produto no mercado.

Em função dos requisitos dos sistemas embarcados, especialmente em relação às limitações do tempo de projeto, espera-se que os SoCs sejam dominados pelo software, ou seja, pelas restrições das aplicações que serão executadas. Estas aplicações, em geral, apresentam requisitos cada vez mais rígidos em relação à qualidade de serviço e a restrições temporais.

Quando os SoCs forem utilizados para aplicações de tempo real, o desempenho e a latência da NoC precisam ser avaliados de forma mais criteriosa, pois em sistemas de tempo real a previsibilidade é um dos quesitos mandatórios. O desempenho influi diretamente sobre outro quesito fundamental em sistemas de tempo real: a qualidade de serviço (QoS – do inglês, *Quality of Service*). Para determinadas aplicações de tempo real é necessário garantir um nível mínimo de QoS de forma que os seus requisitos temporais sejam atendidos. Assim, a previsibilidade do tempo de execução, bem como do tempo de comunicação através da rede, precisam ser conhecidos e determinísticos.

Em um sistema embarcado baseado em redes, o QoS pode receber interferência de técnicas de redes de interconexão tradicionais, como os mecanismos de controle, que podem prover diferentes prioridades para as diversas aplicações ou fluxos de dados, ou ainda, a garantia de determinado nível de desempenho mínimo de acordo com os requisitos das aplicações.

1.1 Caracterização do Escopo do Trabalho

As redes em chip representam uma alternativa para interconexão dos módulos em SoCs, principalmente, por permitirem ao sistema escalabilidade, reuso de componentes e paralelismo. Ao mesmo tempo, permitem atender restrições de sistemas embarcados, como consumo de energia e distribuição de relógio (BENINI 2002).

Entretanto, aplicações de tempo real trazem uma complexidade adicional, pois acrescentam novas restrições ao projeto de sistemas embarcados. Uma forma de tentar atender aos requisitos dessas aplicações, dentro dessas restrições, é através da configuração de características da NoC como tipo de topologia, arbitragem, mecanismos de controle de fluxo, política de roteamento, entre outras. O ajuste adequado dessas características pode fornecer a qualidade de serviço necessária a essas aplicações embarcadas de tempo real.

O escopo desse trabalho aborda as implicações dos requisitos de aplicações de tempo real em sistemas embarcados baseados em redes em chip. Procura-se investigar a influência da estrutura de comunicação no atendimento aos requisitos de QoS dessas aplicações. Essa investigação inclui a avaliação que ajustes realizados na NoC têm sobre os requisitos temporais das aplicações, especialmente em relação ao atendimento dos *deadlines* das mensagens e ao tempo médio consumido nessas comunicações.

1.2 Objetivos e Contribuições

Frente aos requisitos das aplicações de tempo real, especificamente em relação ao atendimento dos *deadlines* das tarefas e à latência das comunicações, este trabalho busca apresentar mecanismos de ajustes no planejamento e configuração da NoC em sistemas embarcados, objetivando a garantia desses requisitos.

Algumas características desses mecanismos de ajustes foram consideradas fixas, ou seja, não sofreram variações durante os experimentos. São elas: a topologia em grelha 2D; o roteamento determinístico XY e o chaveamento baseado em pacotes e do tipo *wormhole*. Essa decisão baseou-se no fato de que essas características, originárias da rede considerada nos experimentos, são utilizadas na grande maioria das NoCs encontradas na literatura. Os experimentos realizados consideram como arquitetura alvo de comunicação a rede SoCIN (ZEFERINO 2003a), previamente desenvolvida no PPGC da UFRGS.

Entre os ajustes, procurou-se verificar o posicionamento dos núcleos IP na NoC; a variação na arbitragem e no tipo de controle de fluxo utilizados nos roteadores; o impacto do tamanho e ordenamento das filas nos *buffers*. Outras estratégias que se pretendeu avaliar são o incremento, por envelhecimento, da prioridade dos pacotes menos prioritários e, o descarte de pacotes antigos.

Em relação aos núcleos IP da NoC, procurou-se verificar a influência do posicionamento destes em relação ao atendimento dos requisitos temporais. A estratégia de posicionamento dos núcleos IP baseou-se na largura de banda requerida para a comunicação. Avaliou-se também o posicionamento baseado na largura de banda e na prioridade da comunicação.

Na arbitragem comparou-se a versão original da SoCIN, que possui árbitros rotativos do tipo Round Robin, com uma versão com árbitros baseados em prioridades, onde as prioridades associadas a cada tipo de mensagem são consideradas no momento da arbitragem.

No controle de fluxo, além do mecanismo original do tipo *handshake*, avaliou-se a utilização de canais virtuais. Foram considerados quatro canais virtuais para cada canal físico de entrada.

Quanto à memorização, dois aspectos foram investigados. O primeiro, em relação à profundidade dos *buffers*. O outro aspecto, em relação ao ordenamento na fila desses *buffers*, comparando a estratégia original FIFO com outra, onde a fila é ordenada por prioridade.

Todas essas estratégias, aqui citadas, já foram abordadas em trabalhos de outros autores. Elas foram consideradas neste trabalho como forma de melhor avaliar o espaço de projeto, considerando a associação dessas estratégias com duas outras utilizadas em redes de interconexão tradicionais: o incremento, por envelhecimento, da prioridade dos pacotes e o descarte de pacotes antigos. Essas duas novas estratégias já foram utilizadas em redes de interconexão, mas até onde foi investigado, não existe ainda trabalhos que as utilizem em NoCs.

O objetivo na utilização de todas essas estratégias é permitir o mínimo uso de recursos da rede para atender os requisitos das aplicações de tempo real, dentro das exigências de QoS, principalmente no atendimento dos *deadlines* das mensagens e na latência dessas comunicações.

O objetivo deste trabalho é propor uma metodologia que, através da aplicação de estratégias de redes de interconexão ainda não usadas em NoCs, atenda os requisitos das aplicações de tempo real.

1.3 Organização do Texto

Esta tese está organizada da seguinte forma. No Capítulo 2 são apresentadas as alternativas arquiteturais de comunicação em sistemas embarcados. E, como este trabalho utiliza redes em chip como solução arquitetural de comunicação, alguns conceitos e características sobre essas redes são também detalhados.

No Capítulo 3 são discutidas as restrições adicionais a projetos de sistemas embarcados que são impostas por aplicações de tempo real. E são apresentadas brevemente algumas características desse tipo de aplicação.

O Capítulo 4 traça um panorama sobre o espaço de projeto em sistemas embarcados baseados em NoCs, apresentando o estado da arte na pesquisa dessas redes, bem como, a metodologia proposta nesta tese para adequar as NoCs às aplicações de tempo real.

No Capítulo 5, as características das aplicações de tempo real são discutidas em mais detalhes no que se refere a definição dos diferentes tipos e suas principais propriedades, principalmente, em relação ao tráfego que as caracteriza.

No Capítulo 6 são apresentados os parâmetros avaliados na configuração da NoC, discutindo o impacto de cada um deles no sistema e, quando possível, comparando os custos de cada alternativa.

O Capítulo 7 detalha os experimentos realizados e faz uma avaliação dos resultados obtidos. Neste capítulo também é descrito o simulador de NoC desenvolvido para a realização desse trabalho.

Finalmente, no Capítulo 8 são feitas algumas considerações finais e um resumo de todas as considerações apresentadas a cada capítulo. Além disso, são avaliados os resultados obtidos nesta tese e discutidos algumas possibilidades de trabalhos futuros.

2 ARQUITETURAS DE COMUNICAÇÃO EM SISTEMAS EMBARCADOS

Este capítulo apresenta uma visão geral sobre as arquiteturas de comunicações adequadas para sistemas embarcados em um único chip. É feita uma breve discussão sobre alguns dos tipos de arquiteturas utilizados em sistemas embarcados. Depois são apresentadas as principais características de arquiteturas baseadas em redes de interconexão.

Os sistemas embarcados (WOLF 2001) apresentam características em comum com sistemas computacionais de propósitos gerais, mas não possuem a uniformidade desses. Cada aplicação pode apresentar requisitos diferentes de desempenho, consumo de potência e área ocupada, o que vai acarretar em uma combinação distinta de módulos de hardware e software para atender a estes requisitos.

Em muitas aplicações é adequada a integração do sistema em uma única pastilha de silício (chip). Esse tipo de sistema é denominado Sistema em Chip (SoC – do inglês, *System on Chip*). A arquitetura de hardware de um SoC embarcado pode conter diversos componentes: um ou vários processadores, memórias, interfaces para periféricos e blocos dedicados. Esses componentes são interligados por uma estrutura de comunicação que pode variar de um barramento a uma rede complexa, denominada Rede em Chip (NoC – do inglês, *Network on Chip*) (BENINI 2002). Os processadores podem ser de diversos tipos, desde RISC, VLIW ou DSP até ASIPs (processadores com conjunto de instruções para aplicações específicas – do inglês, *Application Specific Instruction-Set Processor*), conforme a aplicação. Em muitos casos, o software da aplicação pode ser composto por múltiplos processos, os quais podem ser distribuídos entres diversos processadores, comunicando-se através de diferentes mecanismos. Isso torna necessária a existência de um sistema operacional de tempo real (RTOS – do inglês, *Real Time Operating Systems*) que fornece, pelo menos, serviços de comunicação e escalonamento de processos (FARINES 2000).

Em geral, os projetos de sistemas embarcados são dominados pelo software. Assim, as restrições das aplicações e de um possível RTOS poderão ser determinantes no alcance do espaço de projeto. O ideal é que a exploração desse espaço de projeto seja feita no nível de abstração mais alto possível. Esta exploração consiste em avaliar as alternativas de arquitetura de hardware e de software, verificando o impacto sobre desempenho, consumo de área e de energia do sistema, entre outros. E para que esta prévia avaliação corresponda ao sistema real é necessária a utilização de bons estimadores.

2.1 Sistemas em um Único Chip

Para atender as restrições de tempo de projeto com a complexidade atual dos sistemas, onde SoCs com milhões de transistores devem ser projetados em poucos meses, tem sido comum a adoção do paradigma de projeto baseado em plataformas (KEUTZER 2000), que são arquiteturas de hardware e software especificas para um domínio de aplicação, mas altamente parametrizáveis. São possíveis ajustes em parâmetros como: número de componentes de cada tipo, estrutura de comunicação, tamanho da memória, tipos de dispositivos de E/S, dentre outros. Assim é viabilizado o reuso de componentes (KEATING 1998) ou núcleos IP previamente desenvolvidos e testados (BERGAMASCHI 2001), o que reduz o tempo de projeto.

O projeto de um SoC embarcado consiste, então, em se encontrar um derivativo da plataforma que atenda aos requisitos da aplicação, partindo-se de uma especificação de alto nível, explorando-se as possíveis soluções arquiteturais, estimando-se o impacto de diferentes particionamentos de funções entre hardware e software. Feita a configuração da arquitetura, é necessária a síntese da estrutura de comunicação que integrará os componentes de hardware (LYONNARD 2001). Este tipo de projeto depende cada vez mais do software. Por exemplo, embora a plataforma de hardware de um GPS possa ser similar à de um controle de freios ABS, o software certamente não será.

Os núcleos dos SoCs podem ser implementados de diferentes maneiras. Eles podem ser descritos em uma linguagem de hardware que pode ser mapeada para diversos processos de fabricação (núcleos *soft core*); em um *netlist* pronto para as etapas de posicionamento e roteamento (núcleos *firm core*); ou implementados em nível de layout, com informações referentes às principais características do circuito (núcleos *hard core*). Um núcleo, portanto, é resultado de tecnologia, de software e de experiência do projetista e, por isso, está sujeito aos direitos autorais. O núcleo é a propriedade intelectual (IP) que o projetista licencia ao usuário. Os núcleos IPs em um SoC podem ser novos ou herdados de projetos já existentes, bem como ser obtidos de uma ou mais bibliotecas. Se os núcleos são provenientes de terceiros, a integração e o teste podem ser difíceis, com a necessidade de alterações no projeto do núcleo para adequar a uma interface comum.

Em um SoC é necessário ainda ter uma estrutura que permita a comunicação entre os núcleos. Uma arquitetura de comunicação é um conjunto de elementos que provê meios para o tráfego de informações. Essa arquitetura deve permitir a troca de mensagens entre os núcleos a ela conectados. Conectividade é a medida do número de componentes ligados a cada canal da estrutura de comunicação. Segundo este critério as arquiteturas de comunicação podem ser classificadas em ponto-a-ponto, multiponto ou mista (HENNESSY 2003). A arquitetura de comunicação é do tipo ponto-a-ponto quando fornece apenas canais dedicados para a comunicação entre dois núcleos. Quando a arquitetura fornece canais para a comunicação de mais de dois núcleos ela é multiponto. Quando implementa os dois tipos anteriores, ela é do tipo mista. Redes em Chip (NoCs) são exemplos de estruturas de comunicação ponto-a-ponto, enquanto os barramentos são exemplos de estruturas de comunicação multiponto. A Figura 2.1 mostra exemplos de uma arquitetura ponto-a-ponto (a) e de uma arquitetura multiponto (b).

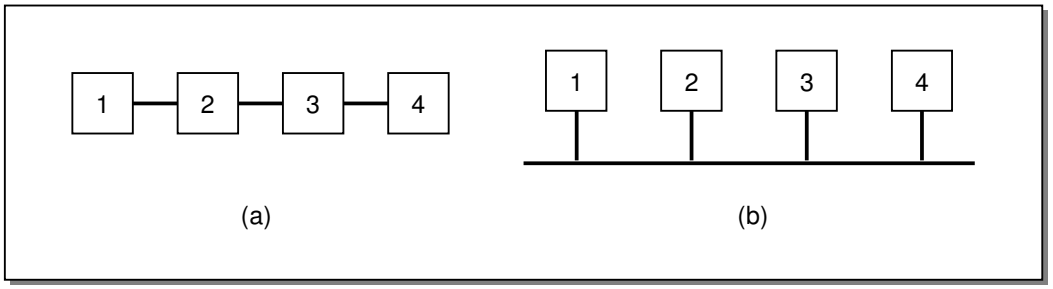

Figura 2.1: Arquiteturas de comunicação: (a) ponto-a-ponto; (b) multiponto.

2.2 Arquitetura de Comunicação entre os Núcleos nos SoCs

A comunicação baseada em conexões multiponto permite que mais de um núcleo compartilhe um mesmo canal físico da estrutura de comunicação para a troca de informações. O compartilhamento dessa estrutura é temporal. Embora as conexões multipontos permitam apenas um escritor em um dado intervalo de tempo, pode existir mais de um leitor durante este intervalo. O compartilhamento temporal é um fator limitante do paralelismo da aplicação, sendo que este limite é dado pela comunicação de forma serial na estrutura de comunicação. Esta limitação pode reduzir o tempo global de execução para muitas aplicações. Porém, para outras, sobretudo aquelas voltadas à computação e não à comunicação, o baixo paralelismo da infra-estrutura de comunicação não compromete o tempo de execução da aplicação.

2.2.1 Conexões Multipontos

Uma estrutura típica da arquitetura de comunicação multiponto é o barramento. O barramento é o meio físico onde os núcleos são conectados através de chaves. Chaves são elementos que habilitam a conexão entre sistemas, tais como *tri-states*. Hennessy e Patterson (HENNESSY 2003) descrevem que muitos autores classificam os barramentos em:

- ***barramentos para conexão entre processador e memória***, que são normalmente barramentos curtos e de alta velocidade;
- ***barramentos de entrada e saída***, que geralmente são longos, podem ter muitos dispositivos conectados, têm uma grande variedade de largura de banda de dados e, em geral, seguem algum padrão.

Um barramento em SoCs normalmente consiste em um conjunto de fios que conectam diferentes módulos do sistema (em geral, núcleos da aplicação), e sobre o qual dados são transmitidos e recebidos. Estes núcleos podem ser classificados como mestres e/ou escravos do barramento. Um núcleo mestre é uma unidade que controla a transferência em um barramento, ou seja, pode solicitar a transmissão ou a recepção de dados através do barramento. Um núcleo escravo é uma unidade que apenas responde às solicitações dos mestres. Um exemplo dessa situação é um microprocessador, atuando como mestre do barramento, e uma memória atuando como escrava. As informações são lidas ou escritas da/na memória a partir dos sinais gerados pelo microprocessador.

Barramentos com diversos núcleos conectados têm métodos de arbitragem para controlar o acesso dos núcleos ao barramento. Estes métodos são normalmente classificados em centralizados ou distribuídos (HENNESSY 2003). No método

centralizado, um dispositivo denominado árbitro é responsável pela atribuição de acesso ao barramento. Neste caso, existem sinais de requisição e de permissão entre o árbitro e os núcleos. O acesso do núcleo ao barramento depende do barramento estar livre e da prioridade deste núcleo em relação aos demais. No método de arbitragem distribuído, não há um árbitro, a prioridade de acesso ao barramento é determinada pelos próprios núcleos. Uma das formas utilizadas para descentralizar a arbitragem é delegar o monitoramento das linhas de requisição aos próprios núcleos do barramento. Desta maneira, cada núcleo sabe sua prioridade na ordem das requisições e se pode ou não utilizar o barramento.

As principais vantagens de implantar estruturas de comunicação com barramentos são o baixo custo e a extensibilidade. O custo é baixo, pois apenas um conjunto de fios é compartilhado por vários dispositivos, enquanto a extensibilidade é dada pela facilidade em acrescentar novos dispositivos ao barramento.

Entretanto, algumas das desvantagens na utilização de barramentos são: a redução do paralelismo da comunicação; a baixa escalabilidade/extensibilidade; o alto consumo de energia e, finalmente, a limitação na velocidade de operação com o aumento do número de núcleos conectados.

O ***paralelismo*** inexiste em barramentos simples, pois não são permitidas comunicações simultâneas, uma vez que todos os núcleos compartilham um único canal de comunicação. Alternativas como a utilização de múltiplos barramentos (Figura 2.2), onde segmentos de barramentos são interconectados, podem aumentar um pouco o número de transações simultâneas, mas ainda assim ficam restritas a uma transação por segmento.

Segundo Benini (BENINI 2002) e Guerrier (GUERRIER 2000), a ***escalabilidade*** dos barramentos é limitada a poucas dezenas de núcleos. O comprimento do barramento e o aumento do número de dispositivos a ele conectado também vão provocar uma redução na ***velocidade de operação***.

Devido a fatores físicos, como o comprimento do barramento e o número de dispositivos a ele conectado, o ***consumo de energia*** pode ser elevado. Esses fatores implicam em um aumento da carga capacitiva do barramento e, por conseqüência, em um aumento do consumo de energia para efetuar a transição de um sinal.

Outra estrutura de comunicação multiponto é o barramento segmentado (RAGHUNATHAN 2003). Esta estrutura é implantada com a divisão de um barramento simples em um barramento com múltiplos segmentos conectados por portas de roteamento. Porta de roteamento é o recurso de comunicação utilizado para conectar dois ou mais segmentos de barramento. A complexidade da porta de roteamento depende do número de segmentos conectados e dos protocolos de comunicação. Para uma topologia de barramentos com muitos segmentos, essa porta pode ser implementada por um circuito roteador, que determina o caminho da mensagem por um endereço definido pelo protocolo de comunicação. Para uma topologia mais simples, a união dos segmentos pode ser feita com *tri-states*.

Assim, cada segmento conectará um número menor de núcleos e o comprimento de cada segmento pode ser menor que o comprimento do barramento original, conferindo aos segmentos melhores características elétricas, tal como a redução da impedância. Em conseqüência, o consumo de energia pode ser reduzido e a freqüência de operação pode

ser aumentada. Além do mais o paralelismo é aumentado, uma vez que as comunicações dentro de cada segmento independem das comunicações internas nos demais.

A Figura 2.2 mostra um exemplo de barramento segmentado. Quando a porta de roteamento que interliga os segmentos é habilitada, a funcionalidade do barramento segmentado se assemelha à do barramento simples (HSIEH 2002).

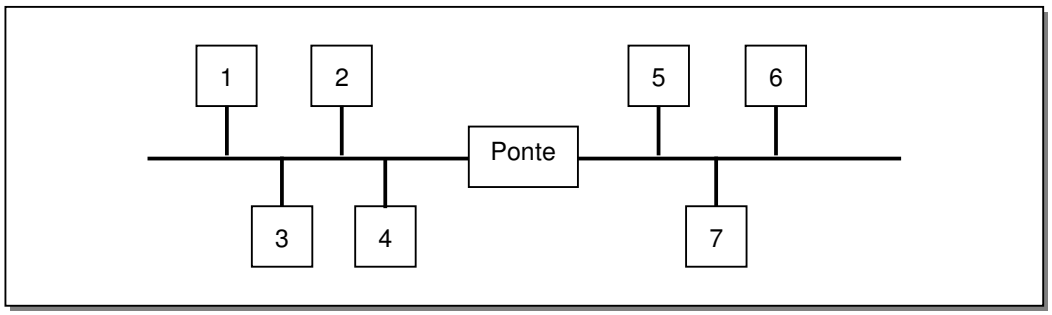

Figura 2.2: Barramento segmentado.

Essa solução de barramento segmentado pode ser ainda melhorada, porém existem limites. Por exemplo, podem ser implantados barramentos mais largos de modo a aumentar o número de bits transferidos a cada ciclo, mas um barramento muito largo não é eficiente para a transferência de palavras de dado menores. Como solução, pode-se concatenar mais de uma transação, mas isso leva a um aumento da latência. Uma alternativa para se obter uma maior largura de banda seria o aumento da freqüência do relógio, mas isso não pode ser feito arbitrariamente devido a restrições elétricas decorrentes das capacitâncias parasitas associadas às unidades conectadas ao barramento e também ao longo comprimento de suas linhas (GUERRIER 2000).

Dentre as arquiteturas de barramento segmentado, bastante conhecidas e utilizadas, estão AMBA da ARM (ARM 2003) e CoreConnect da IBM (IBM 2004). Em geral, estas arquiteturas de barramentos estão vinculadas à arquitetura de um processador, tal como o AMBA vinculado ao processador ARM e o CoreConnect vinculado ao processador PowerPC.

A segmentação de barramentos oferece diversas vantagens arquiteturais (HSIEH 2002), tais como: aumento da concorrência e redução no consumo de energia durante as transições do barramento. A transferência de dados pode proceder em paralelo em diferentes segmentos devido ao isolamento fornecido pelas portas de roteamento (RAGHUNATHAN 2003). Todavia, a segmentação de barramentos oferece alguns problemas de ordem física e lógica que afetam sua escalabilidade. Entre os problemas de ordem física estão as diferenças entre as freqüências de operação e largura de banda de cada segmento. Entre os problemas de ordem lógica está a adaptação de diferentes protocolos de comunicação que possam existir.

Um caso especial de barramento segmentado é o barramento hierárquico (RAGHUNATHAN 2003). Este é construído de forma a ter uma estrutura hierárquica de segmentos, onde cada nível da hierarquia reflete um nível lógico de comunicação. O que determina a construção de conexões hierárquicas é a necessidade de comunicação da aplicação que irá executar sobre esta infra-estrutura. Em essência, cada segmento da hierarquia se comunica com um conjunto de núcleos e com os segmentos imediatamente acima ou abaixo da hierarquia.

2.2.2 Conexões Ponto-a-ponto

Na comunicação baseada em conexões ponto-a-ponto os núcleos são interligados diretamente. Esta estrutura de comunicação oferece alto grau de paralelismo, pois a comunicação entre dois núcleos ocorre de forma independente dos demais. Isso ocorre porque a comunicação entre núcleos é realizada através de canais de comunicação exclusivos.

A estrutura de comunicação dedicada ponto-a-ponto é eficaz para a intercomunicação entre um pequeno número de núcleos. Porém, o número de conexões dedicadas aumenta proporcionalmente com o quadrado do número de núcleos. Este fator e a sua irregularidade, não conferem escalabilidade para esta infra-estrutura (ZEFERINO 2003a).

Apesar das desvantagens, as arquiteturas de comunicação do tipo multiponto, como os barramentos, atendem a grande parte dos SoCs atuais. Porém, a integração de até quatro bilhões de transistores a uma freqüência de 10 GHz em um chip está prevista para até 2012 (ITRS 2005). Essa alta integração irá trazer novos desafios para os projetistas de sistemas. Devido aos efeitos físicos da redução dos transistores, serão necessárias maiores habilidades e experiência do projetista para lidar com fenômenos como ruídos e a dificuldade de manter um relógio sincronizando em todas as partes do sistema. Isso ocorrerá por causa do aumento dos comprimentos dos fios de interconexão e da diminuição dos transistores, o que torna mais significativo o atraso nas interconexões. Além destes fenômenos físicos, o projetista terá que lidar com sistemas heterogêneos cada vez mais complexos, com partes projetadas por diferentes pessoas, com diferentes linguagens e ferramentas (JANTSCH 2003). Além dessas questões, a escalabilidade é outro fator limitante, à medida que cresce a complexidade dos projetos computacionais, sendo que se projeta a existência de SoCs com dezenas a centenas de núcleos (KUMAR 2003).

Em virtude dessas perspectivas, muitos projetistas propõem mudar o paradigma de projeto inteiramente síncrono para um paradigma globalmente assíncrono e localmente síncrono (GALS – do inglês, *Globally Asynchronous and Locally Synchronous*) (SMITH 2004). Nos projetos GALS existem vários sinais de relógio físicos e ou lógicos. Um projeto com o paradigma GALS não implica na necessidade de se ter um sinal de relógio distinto para cada domínio síncrono, mas sim, que cada domínio trate do(s) seu(s) relógio(s) de forma independente dos demais domínios. Ou seja, os relógios podem ser fisicamente os mesmos, mas tratados logicamente como relógios distintos. Esses relógios distintos permitem que a aplicação seja subdividida em domínios síncronos e a interface entre esses domínios síncronos seja realizada através de um recurso assíncrono de comunicação (HEMANI 1999).

No intuito de resolver os problemas inerentes aos futuros SoCs, diversos autores (LIANG 2000) (GUERRIER 2000) (DALLY 2001) (BENINI 2002) (KUMAR 2002) (KARIM 2001) (SGROI 2001) (SAASTAMOINEN 2002) (ZEFERINO 2003b) (MORAES 2003) propõem o uso de uma rede de interconexões chaveadas dentro do chip. As principais vantagens desse tipo de rede com relação às outras arquiteturas de comunicação usadas são: (*i*) aumento do reuso, tanto dos núcleos como da plataforma de comunicação; (*ii*) desenvolvimento de um sistema GALS, eliminando o problema de sincronização do relógio em todo o sistema e permitindo que os núcleos trabalhem com diferentes relógios; e (*iii*) desempenho constante do relógio com o aumento de núcleos

do sistema. A Tabela 2.1 mostra uma comparação entre os prós e contras da utilização de barramentos e NoCs.

Tabela 2.1: Comparação entre barramentos e NoCs

Barramentos		**NoCs**
Cada novo componente insere capacitância parasita e o desempenho elétrico é degradado com o crescimento do sistema.	- +	São usados somente ligações ponto-a-ponto para qualquer tamanho de rede. Logo, o desempenho não é degradado com o crescimento do número de componentes.
O sincronismo do barramento é complexo em um processo submicrônico.	- +	Fios podem ser postos numa estrutura de *pipeline*, pois as ligações são ponto-a-ponto.
A arbitragem do barramento pode ser um gargalo, pois seu atraso cresce com o número de mestres no sistema.	- +	Decisões de roteamento podem ser distribuídas, se o protocolo da rede é descentralizado.
O árbitro do barramento é uma instância específica.	- +	O mesmo roteador pode ser reusado por toda a rede independente do tamanho desta.
A testabilidade de barramentos é complexa e é lenta.	- +	Uma estrutura local e dedicada do tipo BIST é rápida e oferece boa cobertura de teste.
A largura de banda é limitada e compartilhada por todos os núcleos conectados.	- +	A largura de banda agregada cresce com o aumento do tamanho da rede.
A latência é a velocidade do fio, desde que o árbitro consiga o controle do barramento.	+ -	A contenção interna da rede pode aumentar a latência
Qualquer tipo de barramento é quase diretamente compatível com a maioria dos IPs, incluindo o software executando nas CPUs.	+ -	IPs orientados a barramentos necessitam de adaptadores (*wrappers*). Software necessita de sincronização explícita em sistemas multiprocessados.
Os conceitos são simples e bem difundidos.	+ -	Os projetistas precisam se adaptar a novos conceitos.

Fonte: GUERRIER, 2000. p. 255.

A Seção 2.3 detalha alguns dos conceitos básicos das características das redes em chip e discute um pouco sobre projeto para NoCs.

2.3 Redes em chip

As *micro-networks* (TEWKSBURRY 1992) ou *Networks on Chip* (BENINI 2002) propostas como alternativas de comunicação para os SoCs baseiam-se nas redes de interconexão chaveada usadas em computadores paralelos. Essas redes têm como

vantagens a largura de banda escalável, o uso de conexões ponto-a-ponto curtas e o paralelismo na comunicação, entre outras. Dessa forma, alguns dos atributos que caracterizam as redes de interconexão são válidos também para as NoCs.

Tewksburry (TEWKSBURRY 1992) foi o primeiro a sugerir estruturas de comunicação para SoCs baseadas em rede de interconexão, denominadas como *micro-networks*. Mas, somente na década seguinte a pesquisa sobre redes em chip iria se difundir, quando diversos trabalhos (GUERRIER 2000) (LIANG 2000) (DALLY 2001) (KARIM 2001) (SGROI 2001) (BENINI 2002) (KUMAR 2002) (SAASTAMOINEN 2002) (ZEFERINO 2003b) (MORAES 2003) apresentaram propostas para comunicação de SoCs baseadas em NoCs. Então, na seqüência, são apresentados alguns conceitos e definições básicas, juntamente com os principais mecanismos e características das redes de interconexão, tais como: roteamento, chaveamento, controle de fluxo, memorização e arbitragem. Problemas inerentes (*starvation, livelock* e *deadlock*) a redes de interconexão, e que precisam ser tratados, também são discutidos. O estado da arte no espaço de projeto de NoCs é discutido em maior profundidade no Capítulo 4.

2.3.1 Redes de Interconexão Chaveada

As redes de interconexão são utilizadas para interconectar os nodos de máquinas paralelas. Um nodo pode ser simplesmente um processador, um módulo de memória ou até um computador completo com processador e memória local e uma interface de rede. Em um computador multiprocessado (comumente chamado de multiprocessador), a rede de interconexão oferece a infra-estrutura necessária para que os diversos processadores acessem os módulos de memória compartilhada. A comunicação entre os processos executados nos processadores ocorre pelo acesso a variáveis compartilhadas em memória. Esses acessos são protegidos por mecanismos de exclusão mútua. Já em um multicomputador (ou multiprocessador de memória distribuída), a rede interconecta os computadores da máquina e, por não existir compartilhamento de memória, a comunicação ocorre pela troca de mensagens.

Uma rede de interconexão é constituída basicamente por roteadores (em inglês, *routers*) e enlaces ou canais (em inglês, *links*). Os enlaces ligam os roteadores entre si e aos nodos da máquina paralela, enquanto que os roteadores estabelecem o caminho necessário à transferência de dados pela rede. Esses dados são transferidos sob a forma de mensagens, as quais podem ser divididas em unidades menores chamadas pacotes. A forma como os roteadores estão conectados entre si, e como os núcleos estão conectados aos roteadores, define a topologia da rede. Uma rede de interconexão é caracterizada pela sua topologia e por um conjunto de mecanismos que definem a forma como ocorrerá a transferência de mensagens pela rede. As diferentes alternativas de topologia e de mecanismos de comunicação têm impacto direto no desempenho da rede, o qual pode ser avaliado através de algumas métricas, como a largura de banda, a vazão e a latência. Esses mecanismos e características são detalhados na seqüência do texto.

2.3.2 Conceitos

Alguns conceitos utilizados em redes de interconexão chaveada podem ser também aplicados a NoCs. Dentre esses, podem-se destacar alguns elementos: roteador; enlaces ou canais; mensagens; pacotes. Alguns parâmetros utilizados em medidas de avaliação do desempenho de redes de interconexão também são aplicáveis a NoCs, tais como:

largura de banda; vazão e latência. Esses elementos e medidas de avaliação são brevemente apresentados a seguir.

2.3.2.1 Roteador

A estrutura de um roteador consiste de um núcleo de chaveamento (em inglês, *crossbar*), uma lógica de controle para roteamento e arbitragem, e portas de entrada e de saída para comunicação com outros roteadores e/ou com os nodos. Essas portas de comunicação incluem canais de entrada e de saída, os quais podem possuir uma pequena memória para armazenamento temporário de dados (*buffer* de memorização, fila ou, somente, *buffer*). As portas possuem ainda controladores de enlace para a implantação do protocolo físico de comunicação. Eles regulam o tráfego das informações que entram e saem do roteador. A Figura 2.3 ilustra um exemplo de roteador.

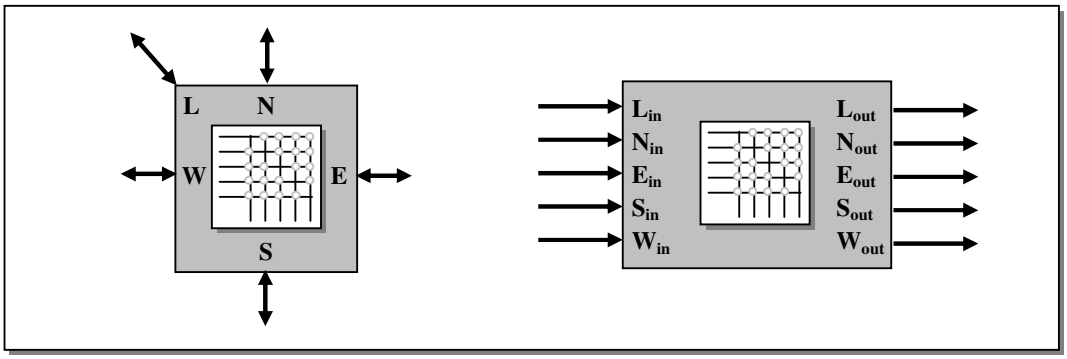

Figura 2.3: Exemplo de um roteador.

2.3.2.2 Enlaces ou Canais

Os enlaces são canais ponto-a-ponto unidirecionais e assíncronos que conectam um roteador a outro ou a um nodo da rede. Essas comunicações costumam ser bidirecionais, constituídas por canais unidirecionais opostos. Assim, é possível a transmissão simultânea de dados em ambas as direções do canal (DUATO 2003).

2.3.2.3 Mensagens e Pacotes

As informações trocadas entre os nodos de origem e de destino numa comunicação são organizadas através de mensagens. Essas mensagens, em geral, possuem três partes: um cabeçalho (em inglês, *header*), os dados propriamente ditos ou carga útil (em inglês, *payload*) e um delimitador de fim (em inglês, *trailer*). Dessa forma o cabeçalho e o delimitador de fim formam um invólucro ao redor dos dados da mensagem. O cabeçalho inclui informações de roteamento e de controle utilizadas pelos roteadores para propagar a mensagem em direção ao nodo de destino da comunicação. O delimitador de fim, por sua vez, inclui informações usadas para a detecção de erros e para a sinalização do final da mensagem. Em geral, as mensagens são divididas em pacotes para transmissão. Esses pacotes podem conter frações de uma mensagem ou uma mensagem inteira. Um pacote é a menor unidade de informação que contém detalhes sobre o roteamento e seqüenciamento dos dados e mantém uma estrutura semelhante à de uma mensagem, com um cabeçalho, uma carga útil e um delimitador de fim. Um pacote é constituído por uma seqüência de palavras cuja largura é igual à

largura física do canal, a qual é denominada PHIT (acrônimo do inglês, *PHysical unIT*). Em outras palavras, um PHIT é definido pelo número de bits de dados transmitidos simultaneamente, sendo igual a 1 bit nos enlaces seriais e a **n** bits nos enlaces paralelos de **n** bits. Além dos bits de dados, o PHIT pode incluir sinais de enquadramento e de controle da integridade dos dados. Um pacote pode ainda ser dividido em FLITs (acrônimo do inglês, *FLow control unIT*), cujo tamanho é múltiplo do PHIT, como ilustra a Figura 2.4.

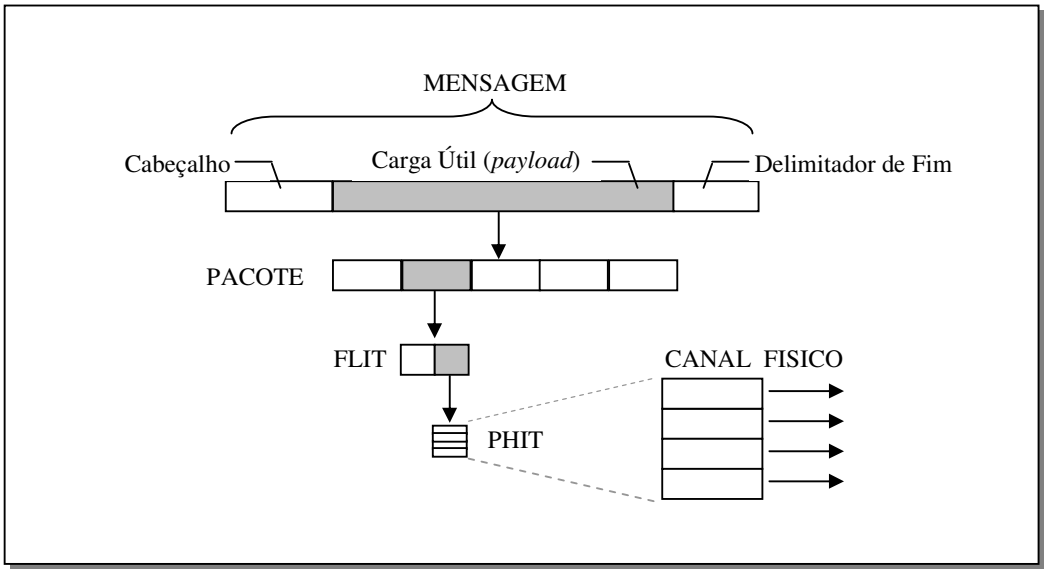

Figura 2.4: Estrutura das mensagens.

Para garantir a transferência de mensagem entre os nodos, torna-se necessário impedir que situações como ***deadlock***, ***livelock*** e ***starvation*** venham a ocorrer (DUATO 2003):

O *deadlock* é uma dependência cíclica entre as solicitações de acesso a recursos de comunicação e armazenamento que bloqueiam indefinidamente certos caminhos da estrutura de comunicação.

A situação de *livelock* ocorre quando a informação transmitida jamais atinge o seu destino, devido a esta percorrer continuamente caminhos cíclicos que não incluem o núcleo destino da informação. O *livelock* é um fenômeno que está normalmente associado aos algoritmos de roteamento na rede.

O *starvation* é definido como sendo a situação onde ocorre uma postergação indefinida de acesso a recursos de comunicação. Um exemplo de *starvation* é a solicitação de um canal de saída por um pacote armazenado em um *buffer* e este canal permanece bloqueado porque o canal de saída é sempre alocado para outro solicitante de mais alta prioridade. Este problema está diretamente relacionado a algoritmos de arbitragem dos roteadores.

As diferentes alternativas de topologia e de mecanismos de comunicação têm impacto direto no desempenho da rede, o qual pode ser avaliado através de algumas métricas, como a ***largura de banda***, a ***vazão***, a ***latência*** e o *jitter*.

2.3.2.4 Largura de Banda

A largura de banda (em inglês, *bandwidth*) refere-se à taxa máxima com a qual a rede de interconexão pode propagar as informações uma vez que uma mensagem entra na rede. Tradicionalmente, no cálculo da largura de banda, são contados os bits do cabeçalho, da carga útil e do delimitador de fim da mensagem, sendo que a unidade de medida utilizada é "bit por segundo" ou bps (HENNESSY 2003).

2.3.2.5 Vazão

A vazão (em inglês, *throughput*) é definida em (DALLY 2004) como o número de mensagens que a rede entrega na unidade de tempo e, em (HENNESSY 2003), como a largura de banda da rede para uma dada aplicação. Ambas as definições não são muito claras. Na primeira, o valor varia se o tamanho das mensagens varia. Na segunda, a vazão pode ser confundida com a largura de banda máxima da rede. Em (DUATO 2003), é apresentado um conceito mais nítido. Vazão é definida como o tráfego máximo aceito pela rede ou, em outras palavras, a quantidade máxima de informação entregue na unidade de tempo. Para que a vazão seja independente do tamanho das mensagens e da rede, seu valor pode ser normalizado, dividindo-o pelo tamanho das mensagens e pelo tamanho da rede. Como resultado, a vazão normalizada é medida em bits por nodo por ciclo de relógio (ou por microssegundo).

2.3.2.6 Latência

A latência é o tempo envolvido desde o início da transmissão de uma mensagem até o momento em que ela é completamente recebida. Para alguns domínios de aplicação, a latência individual de cada mensagem pode ser de pouca importância, principalmente quando são utilizadas cargas sintéticas (DUATO 2003). Nesse caso, o valor médio das latências (ou latência média) seria mais significativo para a avaliação da rede. Porém, se algumas mensagens experimentam latências muito maiores que o valor médio, isso pode ter um impacto importante no desempenho de algumas aplicações. Logo, além da latência média, o desvio padrão pode ser importante para ajudar a identificar essas situações, além do valor máximo para aquelas categorias de aplicação de tempo real.

A latência é medida em unidades de tempo. Porém, como muitas comparações de alternativas de projeto são realizadas utilizando-se simuladores de rede, a latência pode ser medida em ciclos de relógio do simulador.

A latência L de uma rede com carga é dada pela soma dos tempos de:

- sobrecarga (em inglês, *overhead*): refere-se aos tempos gastos pelos nodos fonte e destino para, respectivamente, injetar e retirar a mensagem da rede.
- ocupação do canal: medida do tempo gasto para transferir a mensagem pelos enlaces utilizados na rota entre os nodos fonte e destinatário.
- atraso de roteamento e chaveamento: tempo gasto para a determinação da rota a ser utilizada para a propagação da mensagem através do roteador.
- atraso de contenção: períodos de tempo nos quais a mensagem não consegue avançar devido ao congestionamento da rede.

Assim, a latência total depende das estratégias utilizadas no roteamento e no chaveamento. Na Sub-seção 2.3.3 discutiremos essas e outras características importantes para as NoCs.

2.3.2.7 Jitter

A variação no atraso de envio de um pacote (em inglês, *jitter*) é calculada como a diferença entre a latência mínima e máxima dos pacotes pertencentes a um mesmo fluxo lógico de dados (FELICIJAN 2004), como por exemplo, uma mensagem ou uma rajada de dados de uma aplicação multimídia.

2.3.3 Características

Uma rede de interconexão pode ser caracterizada pela sua topologia e pelas estratégias utilizadas por ela para: o roteamento, o controle de fluxo, o chaveamento, a arbitragem e a memorização.

2.3.3.1 Topologia

A topologia de rede consiste na estrutura formada pela ligação dos roteadores. Essa organização pode ser expressa na forma de grafo, onde os roteadores são vértices e os canais de comunicação são arcos (NI 1993). Esse grafo é do tipo $G(N,C)$ no qual N representa o conjunto de roteadores da rede e C representa o conjunto de canais de comunicação.

A topologia representa a descrição completa do arranjo de interconexões entre elementos de roteamento desta. As topologias podem ser regulares, quando é possível definir um padrão deste arranjo com base na estrutura dos elementos de roteamento destes. Caso este padrão não possa ser identificado, a topologia é denominada irregular. A Figura 2.5 ilustra o conceito, mostrando alguns tipos de topologias regulares.

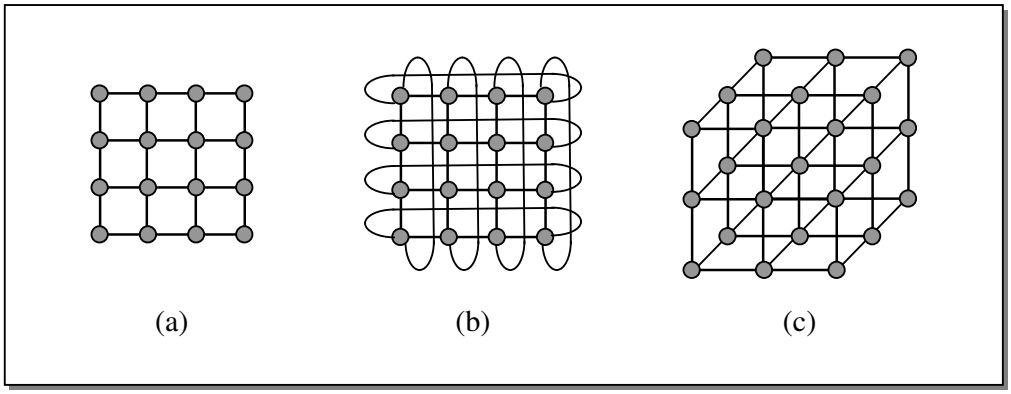

Figura 2.5: Exemplos de topologias regulares de redes diretas: (a) grelha 2D (*mesh*); (b) torus 2D e (c) hipercubo 3D.

As topologias que são construídas usando estruturas 2D podem ser agrupadas em dois grandes grupos: (*i*) redes diretas e (*ii*) redes indiretas (DALLY 2004) (DUATO 2003).

As ***redes diretas*** são caracterizadas pela associação de cada roteador a pelo menos um núcleo, formando um elemento único dentro da rede chamado de nodo (em inglês, *tile*). Cada nodo é ligado ponto-a-ponto a outros nodos. Se uma mensagem é enviada, ela passará por vários nodos antes de chegar ao seu destino, utilizando para isso o roteador de cada nodo.

Uma rede direta pode ter vários níveis de conectividade. Por exemplo, na Figura 2.5 são mostradas as redes diretas mais usadas: a rede grelha e a torus. Estas redes possuem conexão com, no máximo, quatro nodos vizinhos. Se elas possuíssem um nível ideal de conectividade, cada nodo teria uma conexão para qualquer nodo da rede, o que aumentaria muito o custo da rede e diminuiria a sua escalabilidade.

Nas topologias de redes indiretas os roteadores não são necessariamente ligados a um núcleo, sendo que apenas alguns roteadores possuirão conexão com os núcleos do sistema. Entre as redes indiretas destacam-se a matriz de chaveamento (*crossbar*), ilustrada na Figura 2.6(a) e as redes multi-estágio, como as do tipo árvore gorda (ANDRIAHANTENAINA 2003) que é mostrada na Figura 2.6(b). A rede de matriz de chaveamento consiste em uma boa solução em desempenho, mas, para redes muito grandes, sua complexidade e custo crescem ao quadrado em relação ao número de núcleos do sistema, tornando-a muito custosa nestes casos (OST 2004).

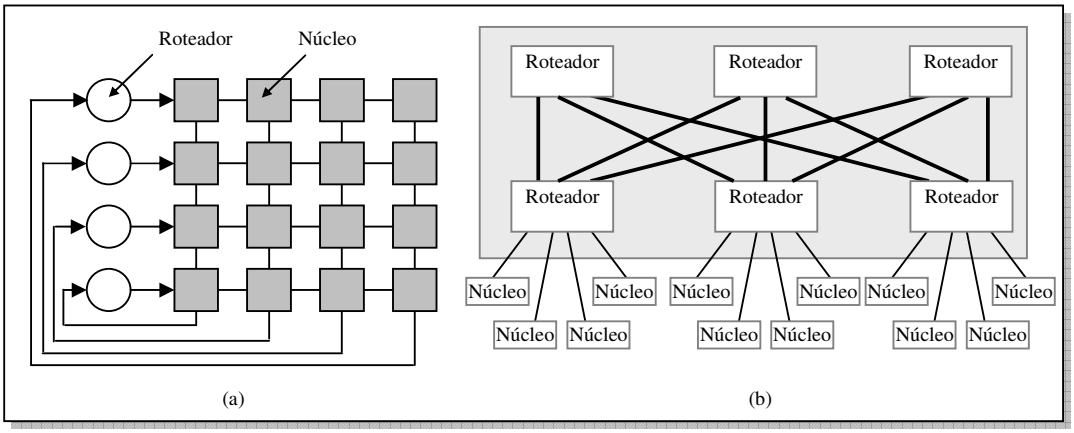

Figura 2.6: Exemplos de redes indiretas: (a) matriz de chaveamento (*crossbar*) e (b) árvore gorda.

Uma mensagem trocada entre dois nodos não-vizinhos deve passar por um ou mais nodos intermediários. Se uma mensagem recebida por um nodo é destinada a outro nodo dentro da rede, o roteador do primeiro deve repassá-la para algum dos seus nodos vizinhos para que a mensagem avance em direção ao seu destino. Apenas os roteadores são envolvidos nessa comunicação, sem interferência dos processadores dos nodos intermediários. Um algoritmo de roteamento é utilizado pelo roteador a fim de decidir para qual nodo vizinho a mensagem deve ser repassada.

2.3.3.2 Roteamento

O roteamento é o método utilizado por um pacote (ou mensagem) para escolher um caminho para seguir através dos canais da rede. O algoritmo de roteamento define o caminho pelo qual um pacote será transmitido pela rede, da sua fonte até seu destino, sendo alta a sua influência no desempenho da rede. Além de influenciar o desempenho da rede, o algoritmo pode garantir que a rede seja imune a problemas como *deadlock* e *livelock* e capacitá-la para transmitir pacotes, apesar de falhas em seus componentes. Mas o algoritmo de roteamento também depende da topologia usada na rede.

Os algoritmos de roteamento podem ser classificados de acordo com seu objetivo. Existem algumas propostas de taxonomia para algoritmos de roteamento (ASHRAF 1998) (DALLY 2004) (DUATO 2003) (NI 1993) usando critérios tais como:

- o *momento de realização do roteamento*: ***dinâmico*** quando realizado em tempo de execução e ***estático*** quando realizado em tempo de projeto;
- o *número de destinos das mensagens*: ***unicast*** quando os pacotes têm apenas um destino e ***multicast*** quando os pacotes têm mais de um destino;
- a *forma de implantação*: ***baseado em tabelas*** quando o caminho é definido de acordo com uma tabela armazenada em memória ou ***baseado em máquinas de estado*** quando o caminho é definido a partir de um algoritmo;
- o *local onde a decisão de roteamento é tomada*: ***centralizado***, quando um único elemento define o caminho de todos os pacotes; ***origem***, quando o caminho do pacote é definido na origem; e ***distribuído***, quando o caminho do pacote é definido por cada roteador que o pacote atravessar na rede durante sua transmissão até o destino. O roteamento distribuído permite ao pacote ter um cabeçalho menor do que quando o roteamento é do tipo origem, pois não é necessário conter no cabeçalho toda a rota do pacote;
- o *processo de seleção do caminho*: ***determinístico***, quando o caminho é sempre o mesmo para todas as comunicações com mesma origem e mesmo destino, ou ***adaptativo***, quando o caminho entre a origem e o destino é determinado por fatores da rede, tais como condições de tráfego. Se houver, por exemplo, uma falha de componente ou caminho bloqueado o roteamento determinístico ficará bloqueado até o recurso ser liberado, enquanto o roteamento adaptativo prevê vários caminhos possíveis entre a origem e o destino. A seleção de um caminho depende de informações como tráfego da rede e estado dos canais, possibilitando assim evitar áreas congestionadas ou que apresentem falhas.

Um exemplo de roteamento determinístico é o XY para topologias grelha. Neste roteamento, o pacote percorre um caminho que passa por todas as conexões e roteadores no eixo X (horizontal) até chegar ao endereço em X do roteador destino, então percorre todos os roteadores no eixo Y (vertical) até chegar ao roteador destino.

2.3.3.3 Chaveamento

Os dados são transferidos na rede entre os nodos da rede, através dos canais físicos que interligam esses nodos. A escolha da forma na qual esses dados são transferidos da entrada de um roteador para um de seus canais de saída é o que determina o comportamento das chaves internas de cada roteador, ou seja, é o que é definido por chaveamento. As duas técnicas principais de chaveamento são: (*i*) chaveamento por circuito e (*ii*) chaveamento por pacote.

O ***chaveamento por circuito*** (em inglês, *circuit switching*) estabelece um caminho completo (circuito) do núcleo de origem até o núcleo de destino para, em seguida, enviar a mensagem pela rede. Neste tipo de chaveamento os canais físicos necessários para a transmissão da mensagem ficam todos reservados, não podendo ser usados até o fim da transmissão. O uso de *buffers* nesta metodologia é mínimo, pois não ocorrerá contenção da mensagem na rede, visto que o caminho já estará todo pré-estabelecido. Os *buffers* serão necessários apenas para conter o cabeçalho responsável por reservar os recursos da rede. A desvantagem deste método é que os canais reservados não podem

ser utilizados por outra mensagem, gerando menor utilização da rede e, possivelmente, aumento no tempo médio de transmissão das mensagens.

No ***chaveamento por pacote*** (em inglês, *packet switching*), a mensagem é dividida em pacotes, onde cada pacote possui um cabeçalho com informações necessárias à sua transmissão pela rede. Os pacotes informam a cada roteador qual caminho seguirão na rede, não existindo um caminho pré-definido e sem estabelecer um circuito fixo do núcleo de origem até o núcleo de destino. Os pacotes reservam seus caminhos dinamicamente na medida em que avançam em direção ao destinatário. Como não há um caminho pré-definido, é possível uma maior utilização da rede, pois não há reserva de recursos.

O tratamento por pacote implica na utilização de um desses modos de chaveamento (DUATO 2003): (*i*) *store-and-forward*; (*ii*) *virtual cut-through* e (*iii*) *wormhole*.

No chaveamento por pacote no modo *store-and-forward* (armazena e repassa) um pacote tem que ser completamente armazenado em um roteador antes de ser enviado ao próximo roteador. Ou seja, o roteador somente selecionará o canal de saída para transmitir o pacote apos tê-lo armazenado totalmente em seu *buffer*. Este modo exige grande armazenamento nos roteadores, sendo que os *buffers* devem ser dimensionados para o tamanho máximo do pacote. Além disso, há uma sobrecarga na comunicação devido ao tempo que os roteadores precisam para encaminhar cada pacote. A latência de comunicação irá aumentar proporcionalmente com o aumento do pacote, pois este só é repassado após ser completamente armazenado no roteador.

No modo *virtual cut-through*, um roteador pode enviar um pacote a partir do momento no qual o próximo roteador garanta que pode receber todo o pacote. O dimensionamento dos *buffers* deve ser feito para conter um pacote inteiro. No pior caso, quando a rede estiver completamente carregada, ela portar-se-á como uma rede de chaveamento por pacote *store-and-forward*, armazenando o pacote inteiro no *buffer* de entrada até a liberação do canal de saída.

O *wormhole* é uma variação do *virtual cut-through* que procura reduzir o tamanho dos *buffers* de armazenamento. Os pacotes são quebrados e transmitidos entre os roteadores em unidades menores, denominadas de *flits*[1]. Os *buffers* são projetados para armazenar poucos *flits*, de modo que os *flits* restantes do pacote ficarão armazenados nos demais roteadores da rede. Uma desvantagem associada a este modo é que apenas o *flit* de cabeçalho contém informações sobre o endereçamento destino. Logo, os demais *flits* que compõem o pacote devem seguir o mesmo caminho reservado para o cabeçalho. Se um cabeçalho não puder avançar na rede em função de um congestionamento, todos os *flits* restantes são bloqueados ao longo do caminho, até que este seja liberado. Neste tipo de chaveamento um canal só é liberado após a passagem de todos os *flits* que compõem o pacote.

A grande vantagem do chaveamento *wormhole* é que os requisitos de *buffer* são menores que os das outras duas abordagens, possibilitando a construção de roteadores

---

[1] *FLIT: FLow control unIT* (unidade de controle de fluxo) é uma unidade lógica do modo de chaveamento *wormhole* que é múltipla do PHIT. PHIT é uma unidade física que indica o número de fios físicos para transmissão de dados entre dois roteadores, ou entre um núcleo e um roteador.

pequenos e rápidos. Em geral, o chaveamento *wormhole* é o mais vantajoso com relação à utilização da rede e ao custo dos roteadores, sendo o mais usado atualmente.

Em (DUATO 2003) são apresentados modelos de latência para as diferentes técnicas de chaveamento. Nesses modelos são desconsiderados a sobrecarga de comunicação dos nodos de origem e de destino e o atraso devido à contenção na rede. Em outras palavras, os modelos assumem que a rede trabalha sem carga e leva-se em conta apenas a ocupação do canal e os atrasos de roteamento e chaveamento.

A Tabela 2.2 apresenta os parâmetros utilizados nos modelos de latência. É assumido que a mensagem (ou o pacote) possui um cabeçalho com tamanho igual a W bits (largura física do canal) e uma carga útil de tamanho igual a L bits (ou L/W palavras de 1 PHIT). Os canais dos enlaces da rede operam a uma freqüência de B Hz, com um tempo de propagação igual à t_w. Em cada roteador, são gastos t_r segundos para rotear os cabeçalhos mais t_s segundos para propagação de um PHIT (palavra de W bits). Os nodos de origem e de destino da mensagem são separados por D enlaces, sendo que o número de roteadores percorridos pela mensagem é igual a D+1.

Tabela 2.2: Parâmetros dos modelos de latência.

Parâmetro	**Unidade**	**Definição**
L	bits	tamanho da carga útil da mensagem
W	bits	tamanho do PHIT
B	Hz	freqüência de transmissão
t_w	segundos	tempo de propagação nos canais dos enlaces (= 1/*B*)
t_s	segundos	tempo de propagação nos canais internos do roteador
t_r	segundos	tempo para o roteador fazer o roteamento
D		número de enlaces entre os nodos fonte e destinatário

A Tabela 2.3 mostra os modelos de latência (DUATO 2003), incluindo também o requisito de memorização mínima de cada técnica de chaveamento.

Tabela 2.3: Modelos de latência.

Técnica de chaveamento	**Latência sem contenção** (desconsiderando a sobrecarga)	**Memorização mínima**
Circuito	$D \cdot [t_r + 2.(t_s + t_w)] + t_w.(L/W)$	1 FLIT
Store-and-forward	$D \cdot \{t_r + (t_s + t_w) \cdot [(L + W)/W] \}$	1 pacote
Virtual cut-through	$D \cdot (t_r + t_s + t_w) + \max\{t_s, t_w\} \cdot [L/W]$	1 pacote
Wormhole	$D \cdot (t_r + t_s + t_w) + \max\{t_s, t_w\} \cdot [L/W]$	1 FLIT

2.3.3.4 Controle de Fluxo

O controle de fluxo trata da alocação de recursos da rede, como *buffers* e canais. Esses recursos são necessários para que uma mensagem possa percorrer a rede. O controle de fluxo realiza a regulação de tráfego nos canais. Esta política de regulação é implantada nas redes de interconexão em nível de enlace, utilizando *buffers* para o armazenamento dos dados em transferência. Se o *buffer* de entrada do receptor estiver cheio, o transmissor irá manter os dados no seu *buffer* até que o recurso ocupado no receptor esteja livre. Existem três alternativas usualmente utilizadas na implementação do controle de fluxo (ZEFERINO 2003a): (*i*) *handshake*; (*ii*) controle baseado em canais virtuais e (*iii*) controle baseado em créditos.

O controle de fluxo do tipo ***handshake*** consiste em duas linhas, uma de validação e outra de reconhecimento (*acknowledge*). Pela linha de validação o nodo emissor avisa que possui mensagem para enviar. O nodo receptor irá confirmar se há espaço em seu *buffer* através da linha de reconhecimento, estabelecendo a transferência da mensagem.

O controle de fluxo baseado em ***canais virtuais*** é utilizado em redes de interconexão com chaveamento *wormhole* e foi concebido com o objetivo de resolver os problemas de *deadlock* nestas redes. Esse método consiste em dividir o *buffer* de entrada em filas independentes de profundidades menores que irão formar os canais-virtuais, procurando desta forma evitar o bloqueio de cabeça de linha (HoL – do inglês, *Head-of-Line blocking*). Este bloqueio ocorre quando, durante a transmissão de vários *flits* seguidos, o *buffer* de entrada do receptor enche, provocando o bloqueio do canal físico por onde o pacote é transmitido. O uso de canais virtuais permite uma maior utilização do canal físico, que poderá utilizar outro canal virtual em caso de bloqueio. A Figura 2.7 ilustra como o uso de canais virtuais pode auxiliar na prevenção de bloqueios de recursos devido a dependências ou a recursos compartilhados da rede. Na Figura 2.7(a) a rede não possui canais virtuais e o fluxo de dados da comunicação B fica bloqueado pela comunicação A, que está bloqueada em outro ponto da rede. Já na Figura 2.7(b), onde existem canais virtuais, o fluxo de B é enviado por um canal virtual diferente no mesmo canal físico, mas cujo buffer possui uma fila distinta daquele utilizado por A. Assim, mesmo que a comunicação A continue sendo bloqueada em um ponto posterior da rede, com o uso de canais virtuais, B poderá continuar seu tráfego pela rede.

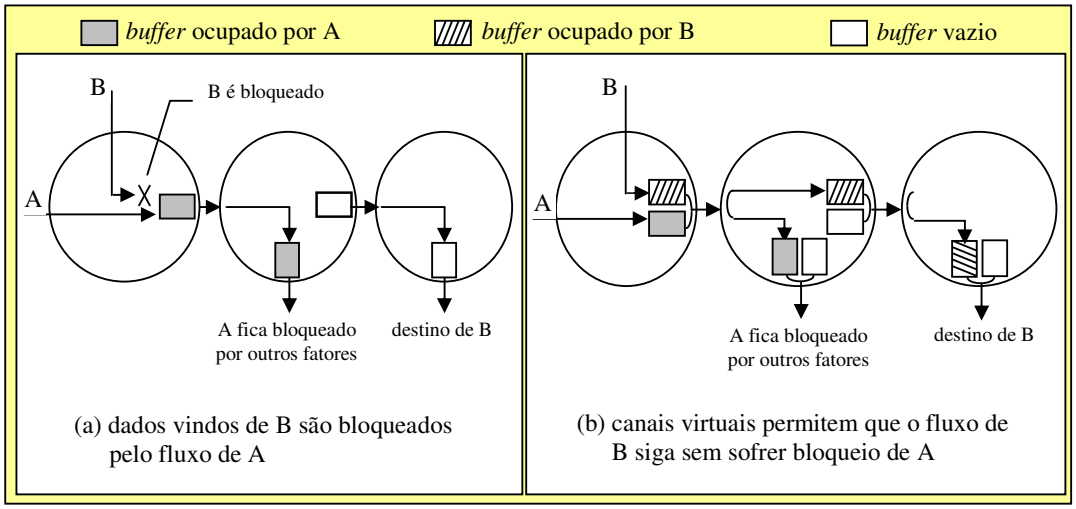

Figura 2.7: Exemplo de utilização de canais virtuais: (a) rede sem canais virtuais e (b) rede usando canais virtuais (BJERREGAARD, 2006).

No controle de fluxo ***baseado em créditos***, uma transmissão só ocorre se houver espaço no *buffer* do receptor. O protocolo opera baseado no número de créditos, que corresponde ao espaço no *buffer* que o receptor possui. Assim, o transmissor recebe informação relativa ao número de créditos do receptor e, se este possuir créditos, a mensagem é enviada e o número de créditos é decrementado. Se a mensagem é passada adiante, esvaziando uma posição do *buffer*, o número de créditos é incrementado novamente.

2.3.3.5 Arbitragem

Enquanto o roteamento determina qual canal de saída dever ser usado por um pacote que está em uma das entradas, a arbitragem define qual canal de entrada (ou *buffer* de entrada) poderá utilizar um determinado canal de saída (ou *buffer* de saída). Em outras palavras, o roteamento é um mecanismo de seleção de saída e a arbitragem é um mecanismo de seleção de entrada. A arbitragem é fundamental para a resolução de conflitos decorrentes da existência de múltiplos pacotes competindo por um mesmo canal de saída. O mecanismo de arbitragem deve ser capaz de resolver esses conflitos, selecionando um dos pacotes com base em algum critério e sem levar qualquer pacote a sofrer *starvation*, ou seja, ficar indefinidamente esperando por uma oportunidade para avançar em direção ao nodo de destino. O árbitro de um roteador pode ser implementado de forma: (*i*) centralizada ou (*ii*) distribuída.

Na abordagem ***centralizada***, os mecanismos de roteamento e de arbitragem são implementados em um único módulo. Esse módulo recebe os cabeçalhos dos pacotes, executa o roteamento e determina o canal de saída a ser utilizado por cada pacote. A partir disso, ele faz a arbitragem, selecionando os pacotes a serem conectados em cada saída, configura o *crossbar* e habilita os *buffers* selecionados para os mesmos avançarem seus pacotes. Essa abordagem é utilizada em árbitros que procuram maximizar a utilização do *crossbar*, realizando uma arbitragem global, a qual considera todos os pacotes que estejam prontos para serem transmitidos nos *buffers* dos canais de entrada, bem como o estado presente dos canais de saída. Entretanto, essa centralização impõe maiores restrições quanto à capacidade de roteamento de pacotes no tempo.

Na abordagem ***distribuída***, o roteamento e a arbitragem dos canais são realizados de forma independente. Cada canal possui seus módulos de roteamento e arbitragem associados às suas portas de entrada e saída, respectivamente. Esta abordagem não permite a visão global dos canais de saída, mas tem como vantagem possibilitar árbitros mais rápidos com uma arquitetura mais simples.

Existem vários mecanismos de arbitragem, baseados em diferentes critérios tais como (DALLY 2004): prioridades estáticas, prioridades dinâmicas, prioridades cíclicas (*Round-Robin*), escalonamento por deadline, FCFS (*First Come First Served*), LRS (*Least Recently Served*), entre outros.

2.3.3.6 Memorização

Os roteadores com chaveamento por pacote devem ser capazes de armazenar aqueles pacotes destinados a saídas que já estejam sendo utilizadas por outros pacotes e, então, realizar o controle de fluxo de modo a evitar a perda de dados recebidos em cada canal de entrada. Isso exige a implementação de algum esquema de memorização para a manutenção dos pacotes bloqueados dentro do roteador. Ou seja, a memorização é o

esquema de filas usado para guardar mensagens destinadas a canais de saídas que já foram requisitados por um outro pacote e que, por isso, estão bloqueadas na rede.

Um roteador ideal deveria dispor de uma capacidade de armazenamento infinita e garantir que um pacote armazenado não seja bloqueado por outros pacotes quando a sua saída for liberada. Entretanto, nos casos reais, a memória do roteador é limitada e, dependendo da estratégia utilizada, o bloqueio de um pacote por outro é inevitável. A organização dos *buffers* de memória (independentes ou compartilhados) e suas posições (na entrada, na saída ou centralizados) afetam o desempenho do roteador.

Em (DALLY 2004) são apresentadas algumas das estratégias de memorização que podem ser utilizadas em um roteador. Elas podem ser agrupadas em dois tipos: (*i*) centralizada e (*ii*) distribuída

A memorização ***centralizada*** é feita com uma estrutura de armazenamento que recebe e guarda os pacotes bloqueados de todas as entradas. Nessa abordagem, um *buffer* centralizado é utilizado para armazenar os pacotes bloqueados em todos os canais de entrada e o espaço de endereçamento é dinamicamente distribuído entre os pacotes bloqueados. Esse *buffer* é denominado CBDA (*Centrally-Buffered, Dynamically-Allocated*) e, no pior caso, deve oferecer uma largura de banda igual à soma das larguras de banda de todos os canais. Ou seja, em um roteador NxN, o *buffer* deve possuir 2N portas de modo a permitir N acessos simultâneos de leitura e N acessos simultâneos de escrita. A vantagem da memória centralizada está no fato dela, dinamicamente, alocar os endereços de memória entre os canais de entrada, permitindo que canais que estejam recebendo mais pacotes bloqueados em um determinado instante possam ocupar mais memória. É necessário apenas limitar o espaço de memória que cada canal pode vir a alocar, pois se uma entrada ocupar totalmente o *buffer* irá afetar as outras comunicações. Esse problema pode ser evitado pela limitação do espaço alocável a cada canal.

Na memorização ***distribuída*** o espaço de memória é distribuído sob a forma de partições entre os canais de entrada do roteador. Essas partições são implantadas através de *buffers* independentes, cada um com portas de entrada próprias. Cada canal de entrada recebe uma estrutura independente de memorização, sem a possibilidade de compartilhamento dos *buffers* entre os canais. Dentre as várias alternativas possíveis de implantação de *buffers*, Tamir e Frazier (TAMIR 1992) destacam: (*i*) FIFO; (*ii*) SAFC; (*iii*) SAMQ e (*iv*) DAMQ.

O tipo FIFO (do inglês – *First In, First Out*) é a alternativa de menor custo. Os dados neste *buffer* são lidos na mesma ordem em que são escritos. O problema desse tipo de *buffer* é que ele pode subutilizar a rede se ocorrer um bloqueio de cabeça de rede (HoL), quando um pacote bloqueado não permite a transmissão de outro pacote cuja saída está livre.

Com o objetivo de contornar esse problema do *buffer* do tipo FIFO pode-se utilizar os outros tipos, onde a estratégia consiste, basicamente, em dividir o *buffer* em partições.

No *buffer* SAFC (do inglês – *Statically Allocated, Fully Connected*), conforme mostra a Figura 2.8(a), estas partições são fixas, gerando uma menor utilização dele com relação ao *buffer* FIFO. Além disso, o *crossbar* deixa de ser NxN, onde N é o número de portas de saída e entrada, para se tornar um *crossbar* N^2xN, pois cada partição do *buffer* pode ser endereçada para qualquer entrada. Esta estratégia apresenta

uma série de inconvenientes. Primeiramente, há um custo adicional referente ao controle do *crossbar* e ao controle dos N *buffers* por porta de entrada. Em segundo lugar, a taxa de utilização dos *buffers* é limitada a 1/N do espaço de armazenamento total, ou seja, ela é pior que a dos *buffers* FIFO. Por fim, o controle de fluxo é mais complexo, pois deve ser realizado para cada *buffer* de entrada e exige um pré-roteamento dos pacotes recebidos para que os mesmos sejam direcionados à partição correspondente à saída a ser requisitada.

Esse problema de gerenciamento do *crossbar* pode ser simplificado se as saídas dos *buffers* de entrada atribuídos a um mesmo canal de saída forem multiplexadas. Essa estratégia, ilustrada na Figura 2.8(b), é denominada de SAMQ (do inglês – *Statically Allocated Multi-Queue*) e reduz o custo do *crossbar*. Entretanto, ela não resolve o problema da menor utilização dos *buffers* e do controle de fluxo mais complexo.

Com o objetivo de minimizar o problema na questão de utilização dos *buffers*, Tamir (TAMIR 1992) propôs uma estrutura de *buffer* denominada DAMQ (do inglês – *Dynamically-Allocated, Multi-Queue*), a qual é representada na Figura 2.8(c). Nessa estratégia, o espaço de memorização associado a um canal de entrada é particionado dinamicamente entre os canais de saída de acordo com a demanda dos pacotes recebidos. Assim, evita-se o problema do bloqueio HoL dos *buffers* FIFO, aumenta-se a utilização do espaço de memória disponível e o controle de fluxo é mais simples que nas estratégias SAFC e SAMQ, não requerendo pré-roteamento. Entretanto, o gerenciamento do *buffer* DAMQ, baseado em listas encadeadas, torna a sua implantação física mais complexa.

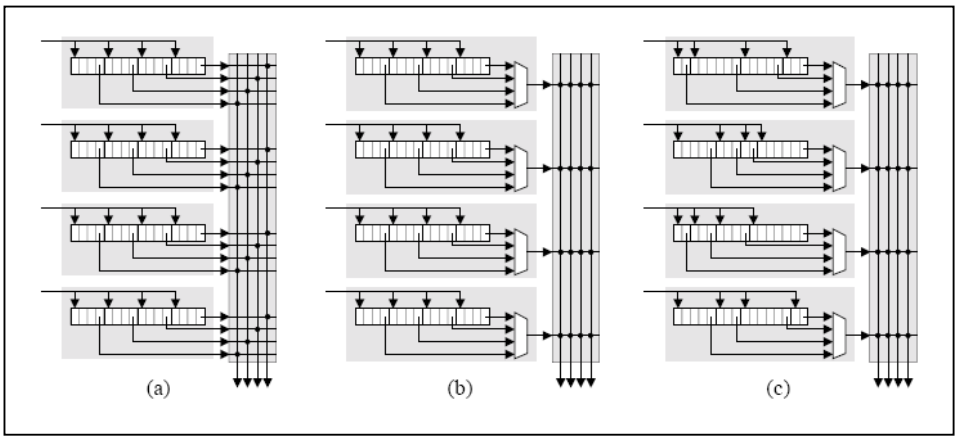

Figura 2.8: Roteador com quatro *buffers*: (a) SAFC; (b) SAMQ; (c) DAMQ.

Além dessas abordagens distribuídas na entrada do roteador, é possível também dividir o espaço de memorização entre as saídas, sendo que cada partição pode ser implantada como *buffer* FIFO. O problema, nesse caso, é que cada *buffer* deve ser capaz de suportar a demanda simultânea das N entradas. O *buffer* pode ser implantado com N portas de escrita ou com uma porta de escrita operando a uma velocidade N vezes maior que a das entradas. Além disso, esta estratégia requer um controle de fluxo interno entre portas de entrada e de saída do roteador.

Em resumo, as alternativas de comunicação de um SoC (conexão total; barramento e NoC) apresentam as características elencadas na Tabela 2.4.

Tabela 2.4: Características das arquiteturas de comunicação

Arquitetura	Paralelismo	Consumo de Energia	Escalabilidade	Reuso
Conexão total	Presente	Baixo (em relação ao barramento)	Quase Inexistente	Restrito
Barramento	Ausente	Maior consumo	Limitada	Possível
NoC	Presente	Baixo (em relação ao barramento)	Quase Ilimitada (limite da área disponível)	Possível

2.4 Considerações

Este capítulo abordou as arquiteturas de comunicação em sistemas embarcados, com ênfase nas redes em chip, que estão se mostrando como a alternativa mais adequada para os SoCs atuais e futuros. Discutiram-se as vantagens e desvantagens da utilização de NoCs. Apresentou-se uma visão geral sobre os conceitos básicos dessas redes e suas características, tais como: topologia, roteamento, chaveamento, controle de fluxo, arbitragem e memorização. Dentre as topologias mais utilizadas atualmente, aparecem as redes diretas (grelha, torus e hipercubo) e as redes indiretas com múltiplos estágios (*crossbar* e árvore gorda). Para o roteamento, foram elencados alguns tipos de algoritmos, sendo que os mais utilizados, em geral, são aqueles que permitem um menor custo. O mesmo ocorre no chaveamento, onde a técnica de chaveamento por pacote *wormhole* é a mais difundida, uma vez que permite a utilização de roteadores menores, mais rápidos e mais baratos. No controle de fluxo, a técnica baseada em créditos é a mais utilizada, combinada ou não com a técnica de canais virtuais. A escolha do mecanismo de arbitragem adequado representa uma boa ferramenta para evitar problemas que precisam ser tratados, como *starvation*. A garantia de ausência de outros problemas (*livelock* e *deadlock*) depende de uma estratégia de roteamento adequada. Na memorização foram mostradas algumas estratégias e a eficiência delas para tratar o problema de bloqueio de cabeça de rede (HoL) dos pacotes.

Neste capítulo abordaram-se questões mais básicas de arquiteturas de comunicação em sistemas embarcados baseados em NoC, identificando conceitos e características básicos, sem aprofundar muito na questão do espaço de projeto. Isso será feito no Capítulo 4, onde, além das características das NoCs, as especificidades das aplicações de tempo real são consideradas no espaço de projeto para sistemas embarcados.

No próximo capítulo serão apresentadas as características principais das aplicações de tempo real em sistemas embarcados, discutindo as suas restrições e o impacto das mesmas nesse tipo de sistema.

3 RESTRIÇÕES DE APLICAÇÕES TEMPO REAL EM SISTEMAS EMBARCADOS

Segundo Wolf (2001), sistemas embarcados apresentam, além dos requisitos funcionais, alguns requisitos não-funcionais, tais como: minimização do consumo de energia; redução do peso do equipamento e da área ocupada em silício; diminuição do custo; consumo máximo de energia. Eles possuem, por outro lado, restrições que delimitam o seu escopo como, por exemplo, o número máximo de pinos de entrada e saída. O projeto de um sistema embarcado é guiado por estes requisitos e restrições, que determinam a natureza dos elementos computacionais que o implantarão.

Além disso, as pressões do mercado para esse tipo de sistema se apresentam, principalmente, na necessidade de um curto tempo para o lançamento do produto no mercado (*time to market*), bem como em um tempo de vida reduzido desse produto. Isso leva os projetistas a deslocar a maior parte da funcionalidade dos sistemas embarcados para software. Por outro lado, as funcionalidades que necessitam de alto desempenho são normalmente implantadas em componentes de hardware dedicado.

Associado a isso, os avanços tecnológicos já permitem a conexão e integração de centenas de milhões de transistores. Conseqüentemente, diversos sistemas computacionais podem ser implementados de forma integrada em um único chip, em SoCs. Dessa forma, diversos componentes, como por exemplo, processadores, memórias e controladores, podem ser colocados na mesma pastilha de silício.

Assim, é importante que as metodologias de projetos adotadas permitam a modularização do sistema e se baseiem no reuso de módulos pré-existentes. Essa modularização do sistema permite reduzir a sua complexidade e reusar esses módulos funcionais menores, denominados de núcleos IP (do inglês, *Intellectual Property*), e dessa forma reduzir o tempo de projeto e, por conseqüência, o *time to market*. Esses núcleos IP podem ser reutilizados de outros projetos desenvolvidos anteriormente ou adquiridos de terceiros. E os núcleos IP podem ser desde processadores, memórias, conversores, controladores, até sub-sistemas completos. Com as futuras tecnologias será possível a integração de um maior número de núcleos IP, da ordem de dezenas a centenas em uma mesma pastilha de silício (BENINI 2002). Assim será possível o desenvolvimento de novas aplicações mais complexas. Porém, acrescenta novas dificuldades com relação à especificação, à distribuição dos núcleos IP na NoC e à avaliação das opções de projeto, assim como questões associadas à estrutura de comunicação e à integração dos módulos IP.

Com o objetivo de facilitar essa integração, padrões de interfaceamento para esses núcleos IP têm sido propostos (VSI 2000) (OCP 2001). Esses padrões podem e devem ser utilizados pelo fornecedor da arquitetura de comunicação, pois dessa forma a interligação dessa arquitetura com os diferentes núcleos IP é facilitada. Primeiramente é evitado que os fornecedores dos IP precisem desenvolver uma versão para cada arquitetura de comunicação. E também, do ponto de vista do integrador, essa abordagem evita que o mesmo tenha que conhecer várias interfaces adaptadoras (*wrappers*) para implantar os adaptadores para cada arquitetura.

Como discutido no Capítulo 2, no projeto de SoCs cada vez mais o paradigma das redes em chip vem sendo adotado como arquitetura de comunicação. Tenta-se, com as NoCs, resolver os problemas de projeto e as limitações dos atuais SoCs que usam arquiteturas de comunicação baseadas em barramentos (KUMAR 2002).

Em função dos requisitos dos sistemas embarcados, especialmente em relação às limitações do tempo de projeto, espera-se que os SoCs atuais e futuros sejam dominados pelas aplicações. Estas aplicações, por sua vez, apresentam requisitos cada vez mais rígidos em relação à qualidade de serviço e a restrições temporais.

O objetivo deste capítulo é discutir as restrições que aplicações de tempo real impõem e o impacto sobre os sistemas embarcados, que já possuem suas especificidades e restrições próprias.

3.1 Aplicações de Tempo Real

Um sistema de tempo real é um sistema computacional que deve reagir a estímulos oriundos do seu ambiente em prazos específicos (FARINES 2000). O atendimento desses prazos resulta em requisitos de natureza temporal sobre o comportamento desses sistemas. Em conseqüência, em cada reação, o sistema de tempo real deve entregar um resultado correto dentro de um prazo especifico, sob pena de ocorrer uma falha temporal. O correto comportamento de um sistema de tempo real, portanto, não depende somente só da integridade dos resultados obtidos (correção lógica), mas também dos valores de tempo em que são produzidos (correção temporal) (STANKOVIC 1988). Uma reação que ocorra além do prazo especificado pode ser sem utilidade ou até representar uma ameaça à integridade do sistema ou do ambiente com o qual ele interage.

Uma concepção errônea que existe sobre sistemas de tempo real é que o problema de tempo real das aplicações se resolverá com o aumento da velocidade computacional. A velocidade computacional ajuda a melhorar o desempenho, reduzindo o tempo médio das respostas. Entretanto, o objetivo de um sistema de tempo real é garantir a previsibilidade, ou seja, o atendimento dos requisitos temporais de cada uma das tarefas do sistema (STANKOVIC 1988).

3.1.1 Características

A maioria das aplicações de tempo real se comporta, então, como sistemas reativos com restrições temporais. Esses sistemas reativos interagem com o ambiente a ser controlado através de interfaces de instrumentação e de sensores e atuadores. Mas existem também casos onde as restrições temporais não são impostas por essas interfaces de instrumentação e sim pelas exigências dos serviços a serem oferecidos ao

usuário. Nesses casos utiliza-se a noção de serviço de tempo real (FARINES 2000). Um exemplo deste último caso são as aplicações multimídia.

Em geral, muitos dos sistemas embarcados encontram-se na categoria de sistemas de tempo real do tipo reativo. Entretanto, espera-se um aumento na quantidade de SoCs dominados por aplicações do tipo multimídia. Dessa forma, os sistemas embarcados deverão se adequar aos requisitos tanto dos sistemas do tipo reativo quanto dos que oferecem serviços de tempo real.

3.1.2 Tipos

De acordo com os critérios de segurança, segundo a intensidade e a repercussão das falhas, muitos autores (KOPETZ 1993)(RAMAMRITHAM 1989)(STANKOVIC 1990) classificam os sistemas de tempo real em:

- *sistema tempo real hard*: as conseqüências de uma falha temporal excedem em muito os benefícios normais obtidos pelo funcionamento correto do sistema, ou seja, é fatal para o sistema.
- *sistema tempo real soft*: uma falha temporal é da mesma ordem de grandeza que os benefícios do sistema em operação normal, ou seja, não há uma "destruição" do sistema.

Para melhor abordar os impactos das aplicações de tempo real em sistemas embarcados baseados em NoCs, podemos dividir os tipos de sistema de tempo real em relação às conseqüências do não atendimento dos prazos (*deadlines*), classificando-os em: (i) *hard* e (ii) *soft*; como também, de acordo com o fluxo dos dados, em: (i) orientados a controle (*control flow*) e (ii) orientados a fluxo de dados (*dataflow*). A seguir apresentamos algumas das características de cada um desses tipos de aplicações de tempo real.

3.1.2.1 Aplicações de Tempo Real Hard

Como nos sistemas de tempo real hard as conseqüências do não atendimento de um *deadline* podem ser fatais para o sistema, é necessária a garantia de que nenhum *deadline* será perdido. Alguns exemplos desse tipo são: os sistemas de controle de vôo, os sistemas de sinalização de ferrovias, os sistemas de controle de usinas nucleares, dentre outros.

Com o objetivo de garantir o atendimento de todos os *deadlines*, sem a perda de nenhum, este tipo de sistema utiliza o tempo mais longo de execução de cada tarefa, ou seja, o pior tempo (WCET – do inglês, *Worst Case Execution Time*), como padrão para a alocação dos recursos do sistema. Dessa forma se busca uma garantia determinística que, mesmo no pior caso, o sistema terá condições de atender os requisitos das aplicações.

3.1.2.2 Aplicações de Tempo Real Soft

Nos sistemas de tempo real soft, onde as conseqüências do não atendimento de um *deadline* são da mesma ordem de grandeza dos benefícios normais do sistema, é necessária a garantia de que um número mínimo de *deadlines* não será perdido. Alguns exemplos desse tipo são: os sistemas de telefonia, os sistemas multimídia, dentre outros.

Com o objetivo de garantir o atendimento do número mínimo dos *deadlines*, este tipo de sistema utiliza, muitas vezes, a estratégia do melhor esforço, ou seja, se não for

possível garantir todos os *deadlines* o sistema perde alguns, dentro do limite definido pela aplicação. Em geral, existe uma degradação, mas o resultado ainda será aceitável. Esse limite aceitável que é definido pela aplicação, através de medidas de qualidade de serviço (QoS – do inglês, *Quality of Service*), e pode ser associado ao desempenho do sistema. No atendimento da QoS pode-se considerar a utilização dos recursos baseados em um tempo médio, por exemplo. O importante será garantir, pelo menos, o nível mínimo de desempenho do sistema.

3.1.2.3 Aplicações Orientadas a Fluxo de Dados (dataflow)

Segundo Lee (LEE 2003), aplicações orientadas a fluxo de dados (em inglês, *dataflow*) podem ser consideradas como integrantes de um modelo de computação baseado em atores, onde esses atores são componentes encarregados de suas próprias ações e se comunicam por troca de mensagens. Essas mensagens são enviadas seguindo um fluxo, onde os atores entram em execução a partir do momento que os dados de entrada estão disponíveis. A comunicação orientada a fluxo de dados é um caso especial do modelo de processos de Kahn (KAHN 1974), no qual processos concorrentes se comunicam através de canais FIFO unidirecionais, onde a escrita nos canais é não bloqueante e a leitura é bloqueante.

Processamento de sinais e processamento de imagens são exemplos de aplicações orientadas a fluxo de dados. Matlab e Simulink, da MathWorks, são exemplos de ferramentas que utilizam esse paradigma de comunicação orientado a fluxo de dados.

3.1.2.4 Aplicações Orientadas a Controle (control flow)

Enquanto as aplicações orientadas a fluxo de dados seguem um fluxo unidirecional FIFO, nas aplicações orientadas a controle (em inglês, *control flow*) as comunicações podem ser bidirecionais e não seguem necessariamente um fluxo. Os diversos nodos podem se comunicar com diversos outros seguindo algum algoritmo de controle ou máquina de estados finitos (FSM – do inglês, *Finite State Machine*). Tipicamente, as aplicações dominadas por fluxo de controle apresentam uma baixa taxa de transferência de dados.

3.1.3 Qualidade de Serviço

Em todos os tipos de aplicação existem requisitos a serem atendidos, seja em relação ao atendimento dos prazos ou em relação ao fluxo dos dados. A definição de mecanismos para atender a esses requisitos pode ser denominada como qualidade de serviço.

Nas redes chaveadas por pacote, o termo QoS se refere aos mecanismos de controle que podem prover diferentes prioridades para diversas aplicações ou fluxos de dados, ou garantir um certo nível de desempenho de acordo com os requisitos dessas aplicações.

A garantia de QoS é particularmente importante para os casos onde a capacidade da rede é limitada, especialmente para aplicações que apresentam restrições temporais. Ou seja, em sistemas embarcados, onde os recursos são extremamente limitados, o suporte a QoS pode garantir o atendimento aos requisitos das aplicações com os restritos recursos do sistema. Algumas aplicações multimídia, por exemplo, necessitam de definições precisas de taxa de transmissão de dados e atraso (latência) máximo da rede.

Aplicações de tempo real *hard* precisam de garantia de serviço QoS, ou seja, é necessária uma reserva dos recursos (largura de banda, canal, *buffer*, por exemplo)

baseada no cenário de pior caso. Como essa reserva é baseada no pior caso, em geral, esses recursos não são utilizados todo o tempo.

Aplicações de tempo real *soft* podem ser atendidas com estratégias alternativas de otimização do uso de recursos, baseadas no caso médio. Essas estratégias de melhor esforço (*best effort*) podem ocasionar perdas de pacotes ou dos prazos (*deadlines*). Assim, é necessário que o número de perdas ou o aumento no atraso ocorram dentro dos limites aceitos pro essas aplicações.

3.2 Restrições em Sistemas Embarcados

Os sistemas embarcados possuem restrições como, por exemplo, as já citadas no início do capítulo: minimização do consumo de energia; redução do peso do equipamento e da área ocupada; diminuição do custo. Quando esses sistemas embarcados executam aplicações de tempo real, outras restrições inerentes a esse tipo de aplicação também precisam ser consideradas. Sistemas de tempo real têm a previsibilidade como princípio básico, e para isso, é necessário o conhecimento da latência máxima entre origem e destino das mensagens, bem como o respeito ao prazo (*deadline*) de cada mensagem. Além desses parâmetros, a perda de pacotes não pode ultrapassar o limite estabelecido pelo tipo de aplicação. Nas aplicações do tipo *hard*, por exemplo, não é permitida a perda de nenhum pacote, a menos que, após a perda de um pacote, ainda haja tempo para retransmiti-lo.

Em relação às aplicações também podem existir requisitos específicos. Por exemplo, algumas aplicações multimídia, além de exigências em relação à latência e à vazão, podem existir requisitos mínimos de *jitter* na comunicação dos dados.

A alocação de recursos de forma determinística, com o objetivo de atender a aplicações de tempo real *hard*, pode ocasionar um aumento do consumo total de energia do sistema. Isso se deve, principalmente, pelo fato dessa alocação basear-se no pior caso e não no caso médio. Dessa forma, outras aplicações poderão ter seus tempos de transmissão aumentados devido ao bloqueio desses recursos pelas aplicações do tipo *hard* e assim, o tempo total do sistema aumenta e, por conseguinte, aumenta o consumo de energia. E essa demanda por mais recursos, em geral, implica também em incremento de área ocupada.

Por outro lado, considerando-se o caso médio, com o uso da estratégia de melhor esforço, pode-se conseguir uma redução do consumo de energia e de área ocupada. Entretanto, por considerar o caso médio, podem ocorrer perdas de pacotes ou do prazo das mensagens, podendo ocasionar o não atendimento dos requisitos temporais.

Essa necessidade de equilibrar os requisitos dos sistemas embarcados com os requisitos das aplicações de tempo real faz com que existam delimitações claras no espaço de projeto.

3.3 Considerações

Este capítulo tratou das restrições de aplicações de tempo real em sistemas embarcados. Foram apresentadas algumas das características dos sistemas embarcados e das aplicações de tempo real, incluindo as restrições de cada um. Os requisitos também foram discutidos, em especial em relação à qualidade de serviço. QoS é essencial às aplicações multimídia que deverão predominar nos SoCs.

Iniciou-se também uma discussão sobre as delimitações do espaço de projeto para aplicações de tempo real em sistemas embarcados, em específico em relação ao impacto das restrições de tempo real nas restrições dos sistemas embarcados. Esse espaço de projeto será discutido em maior profundidade no Capítulo 4.

4 ESPAÇO DE PROJETO EM SISTEMAS EMBARCADOS BASEADOS EM NOCS PARA APLICAÇÕES TEMPO REAL

Em substituição aos barramentos utilizados nos atuais SoCs, as NoCs vêm sendo apontadas como uma solução, pois têm como vantagens a largura de banda escalável, o uso de conexões ponto-a-ponto curtas e o paralelismo na comunicação, entre outras. Embora tenham, como desvantagem, maiores custos e latência na comunicação, esses problemas serão certamente atenuados pela grande disponibilidade de transistores e por soluções arquiteturais que permitirão reduzir a latência da rede e seus efeitos no desempenho da aplicação (BENINI 2002).

A latência na comunicação pode ser atenuada também, quando possível, através de ajustes em algumas das características da NoC como, por exemplo, política de roteamento, velocidade do roteador, tamanho dos *buffers*, etc (ZEFERINO 2003a).

Entretanto, quando os SoCs forem utilizados para aplicações de tempo real o desempenho e a latência da NoC precisam ser avaliados de forma mais criteriosa, pois, em sistemas de tempo real a previsibilidade é um dos quesitos mandatórios. O desempenho influi diretamente sobre outro quesito fundamental em sistemas de tempo real: QoS. Determinadas aplicações de tempo real necessitam garantir um nível de QoS de forma que os seus requisitos sejam atendidos. Assim, a previsibilidade do tempo de execução, bem como do tempo de comunicação através da rede, precisam ser conhecidos e determinísticos.

Várias são as atividades de projeto para a construção de sistemas embarcados, incluindo: especificação, particionamento, escalonamento, alocação, síntese, mapeamento e validação.

No projeto de sistemas embarcados, alguns aspectos precisam ser considerados para que seja viável o uso de NoCs. Dentre esses aspectos podem-se destacar:

- o projeto de NoC para um determinado sistema embarcado deve respeitar restrições de projeto que envolvem: área, potência, temperatura, energia, custo, atendimento de restrições temporais;
- projetos envolvendo SoCs com aplicações que exijam alto desempenho no processamento de dados, são melhores implementados usando NoCs por causa de sua característica de escalabilidade e principalmente a de paralelismo;

- NoCs, para atender determinadas restrições de projeto, devem ser específicas para uma dada aplicação, customizadas para atender essas restrições.

As características especificas de uma NoC são definidas nas várias etapas do projeto. Dentre as diversas questões que podem ser feitas, pode-se elencar:

- qual é a topologia mais adequada para ser usada no projeto?
- qual é a largura de banda que será utilizada?
- qual será o tamanho de *buffer*?
- que tipo de arbitragem utilizar?
- qual é o melhor algoritmo de roteamento?
- que tipo de chaveamento será utilizado?
- qual é a melhor distribuição dos núcleos IP na NoC?
- qual é o melhor escalonamento a ser feito para execução das tarefas na NoC?
- existem outras técnicas para melhor especificar determinadas características da NoC com o objetivo de atender restrições cada vez mais rígidas?

Essas restrições mais rígidas podem ser, por exemplo, o atendimento de restrições temporais. Dessa forma, todas essas questões precisam ser respondidas, levando-se em conta as restrições impostas no projeto de sistemas embarcados. Então, todas essas questões precisam ser respondidas, levando-se em conta as restrições impostas para projetar o sistema embarcado implementado, usando como arquitetura de comunicação uma NoC. Para responder tais questões, é necessária uma metodologia de projeto de NoCs que garanta o desenvolvimento do mesmo em um tempo razoavelmente curto (*time-to-market*).

Nas seções seguintes serão discutidos tópicos sobre projeto de NoCs, tais como: o estado atual das pesquisas em NoCs, o espaço de projeto existente, além de identificar alguns tipos de ferramentas utilizadas nas diversas etapas de desenvolvimento desses projetos, e apresentar também exemplos de fluxos de projeto propostos na literatura.

4.1 Estado da Arte em NoCs

Segundo Bjerregaard e Mahadevan (BJERREGAARD 2006), a pesquisa atual sobre NoCs envolve desde questões de implementação do nível físico até aspectos no nível do sistema e das aplicações, incluindo metodologias e ferramentas. Os autores distinguem quatro grandes áreas no espectro de pesquisas em NoCs, conforme ilustra a Figura 4.1. Cada uma dessas áreas corresponde a uma camada do sistema, que são correspondentes ao modelo OSI, como mostra a Figura 4.2. Mas, ao contrário do modelo OSI para redes de computadores, o projeto de NoCs se beneficia, muitas vezes, do conhecimento prévio dos núcleos que serão conectados a rede e, algumas vezes, do conhecimento das características do tráfego que será aplicado, baseado nas características das aplicações.

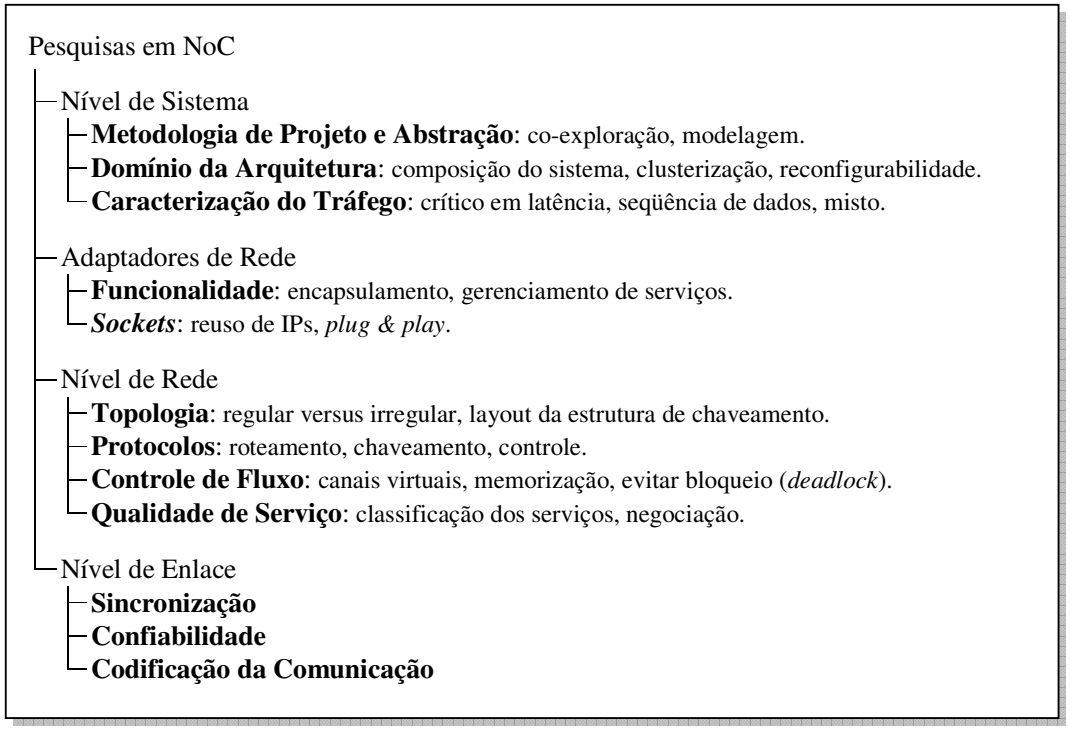

Figura 4.1: Áreas atuais de pesquisas em NoCs (BJERREGAARD, 2006)

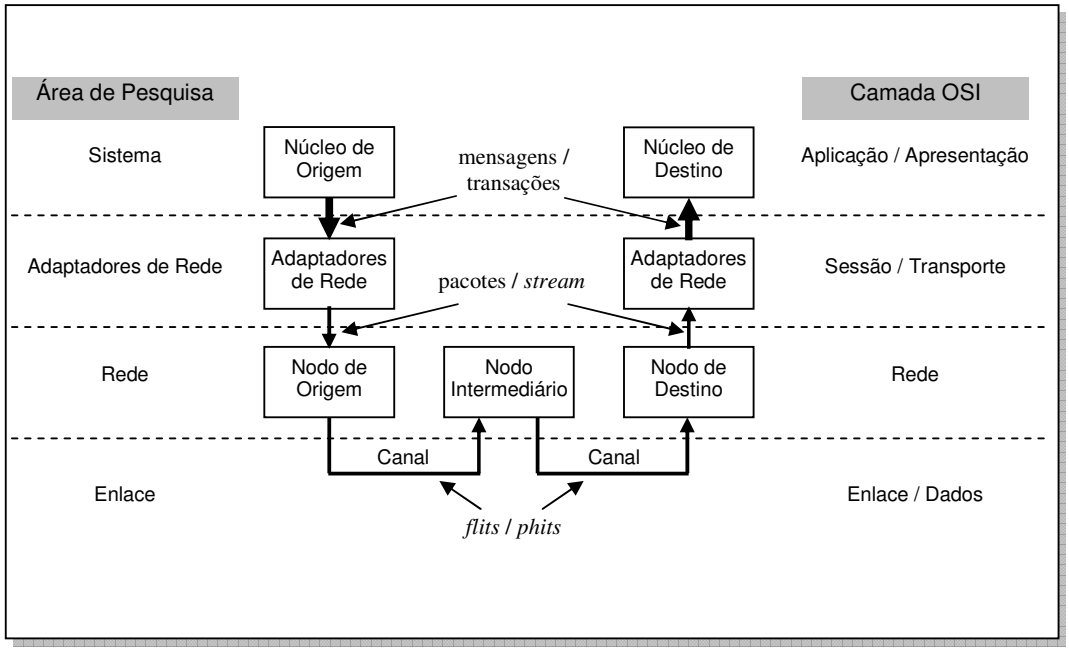

Figura 4.2: Fluxo de dados através dos componentes da NoC (BJERREGAARD, 2006)

4.1.1 Nível de Sistema

O nível de sistema é a camada de maior abstração e envolve as aplicações e a arquitetura dos núcleos e da rede. Neste nível, muitos detalhes de implantação da rede devem permanecer "ocultos". São tratadas aqui as questões referentes à: (*i*) modelagem, (*ii*) metodologia de projeto e co-exploração e (*iii*) caracterização do tráfego.

A ***modelagem*** da NoC em modelos abstratos de software é o primeiro passo para compreender os requisitos da arquitetura da NoC com o respectivo tráfego. Conceitualmente o propósito da modelagem é explorar o vasto espaço de projeto e avaliar o compromisso (*trade-off*) entre potência, área, tempo de projeto, etc., ao mesmo tempo em que atende os requisitos das aplicações, bem como as restrições tecnológicas. Essa modelagem pode ser analítica ou baseada em simulação. Outro aspecto é o ambiente utilizado nessa etapa de modelagem, que pode ir desde uma estrutura puramente abstrata até estruturas com precisão ciclo-a-ciclo.

Em Madsen et al. (MADSEN 2003), os autores utilizam um modelo com alocadores, escalonadores e sincronizadores. O alocador traduz o requisito do caminho a ser percorrido pela mensagem em termos de recursos necessários, como: largura de banda, *buffers*, etc. O escalonador executa a transferência da mensagem de acordo com os serviços disponíveis na rede. Já o sincronizador modela a dependência entre as mensagens, permitindo a concorrência. Esses três componentes podem ser descritos para uma grande variedade de arquiteturas de NoCs.

Alguns autores (BOLOTIN 2004) (XU 2004) utilizam simuladores, como o OPNET (OPNET 2007), desenvolvidos originalmente para redes de computadores. O OPNET provê uma ferramenta para modelagem hierárquica da rede, incluindo processos (máquinas de estado), descrição da topologia da rede e simulação de diferentes cenários de tráfego. Entretanto, como assinalado em Xu et al. (XU 2004), essa ferramenta necessita ser adaptada para ambientes síncronos, requerendo um projeto explícito de sincronização de relógio.

Em Banerjee et al. (BANERJEE 2004) os autores optam por modelos baseados em VHDL e precisão ciclo-a-ciclo para avaliar potência e desempenho da arquitetura da NoC. Eles avaliam latência, vazão, potência dinâmica e de fuga (*leakage power*) da NoC. O projeto é parametrizado no nível RTL, onde podem ser definidos o tamanho dos pacotes, número e profundidade dos canais virtuais, técnica de chaveamento. O projeto RTL é, então, sintetizado no nível de *netlist* SPICE. Os autores afirmam que este trabalho difere do simulador Orion (WANG 2002) que mede o desempenho de potência para interconexões de rede, pelo fato do Orion não considerar em suas estimativas o consumo de potência devido à corrente de fuga.

Devido ao aumento no tamanho das aplicações, outros autores (GENKO 2005) propõem a emulação de NoCs como alternativa ao modelos de NoC baseados em simulação. Modelos, em níveis de abstração mais baixos do que a emulação, também têm sido usados com linguagens como SystemC (SYSTEMC 2007) e SystemVerilog (FITZPATRICK 2004).

Dois aspectos importantes em uma ***metodologia de projeto e co-exploração*** são: (*i*) a parametrização do sistema e (*ii*) a granularidade da NoC. A parametrização no nível de sistema permite uma exploração do espaço de projeto de uma forma mais eficiente e padronizada, através da execução de variados experimentos com a simples modificação dos parâmetros. A granularidade representa o nível de descrição da NoC ou de seus

componentes. Essa descrição da NoC é feita em um largo espectro que vai desde uma granularidade mais "grossa", onde a rede é considerada como o menor elemento, sem subdivisões; até uma granularidade mais "fina", onde a rede é compostas por blocos mais detalhados e é possível modificar esses componentes de mais baixo nível.

O ***tráfego*** de uma NoC pode incluir os mais diversos tipos de comunicação, tais como: acesso à memória, envio de seqüência de dados[2] (*streaming*) de áudio ou vídeo, interrupções. As diversas combinações das características da rede (topologia, protocolos, tamanho dos pacotes e mecanismos de controle do fluxo de dados, etc.) permitem uma comunicação eficiente para um ou mais dos tipos de tráfego predominantes. Por exemplo, na rede CLICHÉ (KUMAR 2002), os conceitos de chaveamento por pacotes foram aplicados em uma rede de topologia grelha 2D, enquanto a rede SPIN (GUERRIER 2000) aplica esses conceitos em uma topologia do tipo árvore gorda. Essas decisões de projeto foram baseadas no tráfego esperado nos respectivos sistemas. Dessa forma, a caracterização do tráfego que será aplicado na rede é um primeiro e importante passo em um projeto de NoCs.

Uma NoC deve acomodar os diferentes tipos de tráfego. Bjerregaard e Mahadevan (BJERREGAARD 2006) identificam que independente da composição do sistema, da clusterização, da topologia, do protocolo, o tráfego da rede estará em uma dessas três grandes categorias: (*i*) latência crítica; (*ii*) seqüência de dados e (*iii*) miscelânea. O tráfego de latência crítica é aquele cuja demanda por latência é rígida. Em geral, são mensagens de tamanho reduzido. Alguns exemplos são as interrupções e os acessos à memória. Na seqüência de dados, o tamanho das mensagens é, muitas vezes, grande e demanda QoS em relação a largura de banda. Como exemplos, têm-se os dados de uma aplicação MJPEG e o acesso direto à memória (DMA). O tipo de tráfego é considerado uma miscelânea quando não existem requisitos específicos de compromissos da rede.

A categorização do tráfego pode ser ainda mais refinada além desses três tipos. Bolotin et al. (BOLOTIN 2004) usa uma categorização de tráfego com quatro denominações: (*i*) sinalização; (*ii*) tempo real; (*iii*) leitura/escrita e (*iv*) transferência de blocos de dados. Em relação à categorização inicial pode-se dizer que: o tráfego de sinalização é de latência crítica; o de tempo real é seqüência de dados; e os de leitura/escrita e de transferência de blocos são do tipo de miscelânea, sendo o tamanho das mensagens o único fator a distinguir um tipo do outro.

4.1.2 Adaptadores de Rede

Os adaptadores de rede possuem duas interfaces: uma com o núcleo a ser conectado à rede e outra com a própria rede. O propósito desses adaptadores é fazer a interface entre o núcleo e a rede, oferecendo serviços de comunicação de forma transparente com o mínimo de esforço por parte do núcleo. Esses serviços de comunicação são em um nível de abstração mais alto do que o hardware da rede e utilizam as primitivas de comunicação oferecidas por este hardware.

A utilização de adaptadores permite uma clara separação entre computação e comunicação. Isso é possível pelo fato dos adaptadores fornecerem encapsulamento do tráfego e gerenciamento dos serviços da rede. O encapsulamento envolve o tratamento do controle de fluxo da rede fim-a-fim, que implica em atividades do tipo: endereçamento global dos pacotes; roteamento de tarefas, reordenamento dos pacotes

[2] Para o conceito de *data streaming* será usado neste texto a nomenclatura seqüência de dados.

armazenados, gerenciamento dos *buffers* para evitar congestionamento, criação de pacotes, negociação de QoS dos serviços entre os núcleos e a rede, dentre outras.

Os adaptadores permitem também o reuso de núcleos IPs, especialmente se esses núcleos seguirem algum dos padrões de interfaceamento que têm sido propostos com o objetivo de facilitar a integração (VSI 2000) (OCP 2001)[3]. Dessa forma, se a rede e o núcleo IP seguem um desses padrões o trabalho de adaptação torna-se mais fácil. Para os fornecedores de núcleos IPs não será necessário desenvolver versões para diversos tipos de redes ou de outras arquiteturas de comunicação. Pelo lado do integrador, ele não precisará desenvolver adaptadores para cada arquitetura. Entretanto, o custo de utilização desses padrões não é desprezível, como é demonstrado em Ost et al. (OST 2005), onde a introdução do OCP na rede HERMES torna as comunicações mais lentas, comparando-se com a versão original da NoC. Uma possível solução para isso seria o particionamento desse adaptador entre software e hardware. A parte de software ficaria possivelmente no núcleo enquanto o hardware seria implantado no adaptador.

4.1.3 Nível de Rede

A função da rede é realizar a entrega de mensagens da origem até o destino. Isto é feito com o suporte de hardware para primitivas básicas de comunicação. Uma rede em chip é definida principalmente pela sua topologia, pelos seus protocolos, pelos seus mecanismos de controle de fluxo e pelos serviços oferecidos, como QoS. Essas características foram detalhadas na Seção 2.3. Nesta sub-seção e nas seguintes serão discutidos os trabalhos constantes na literatura e suas características. A Tabela 4.1 lista algumas das principais[4] NoCs encontradas na literatura com suas respectivas características de topologia, roteamento, chaveamento e suporte à QoS.

4.1.3.1 Topologia

Nas redes Æthereal (RIJPKEMA 2001), CHAIN (BAINBRIDGE 2002) e Xpipes (DALL'OSSO 2003) a definição da topologia faz parte do fluxo de projeto. Dessa forma o projeto dessas redes está em um nível maior de abstração que as demais, cujos projetos são baseados em plataformas. Dentre estas últimas, existe uma predominância da topologia do tipo Grelha 2D. As razões dessa escolha, segundo Moraes (MORAES 2004), são: (*i*) facilidade de implementação com as atuais tecnologias planares de concepção de circuitos; (*ii*) escalabilidade da rede e (*iii*) simplicidade para utilizar estratégias de roteamento do tipo XY. Bjerregaard (BJERREGAARD 2006) ainda acrescenta que as redes de Grelha 2D permitem uma melhor utilização dos canais. Uma alternativa proposta é a Torus (DALLY 2001), cujo diâmetro é menor se comparado à Grelha. Porém, tanto a Grelha, quanto a Torus apresentam uma latência inerente. Na tentativa de reduzir o diâmetro da rede e, conseqüentemente a latência, outros autores propuseram alternativas. A árvore gorda foi utilizada nas redes SPIN (GUERRIER 2000) e T-SoC (PANDE 2003), enquanto soluções baseadas em anel são usadas na Octagon (KARIM 2002) e na Proteo (SAASTAMOINEN 2002). Estas últimas alternativas são úteis, especialmente, para explorar tráfegos locais. Algumas das redes

---

[3] Informações sobre NoCs e o suporte a interfaces de padronização estão disponíveis, por WWW, em http://www.ocpip.org/university/biblio_main/noc/ (acesso em 13 de março de 2007).

[4] Esta lista não é exaustiva e pretende apenas mostrar um panorama do estado da arte das pesquisas em NoC, especialmente em relação ao suporte de QoS. Em (Moraes 2004) e (Bjerregaard 2006) também são encontrados panoramas semelhantes.

que optam pela topologia em Grelha oferecem a possibilidade de adaptação da distribuição dos nodos, transformando-a de regular em uma topologia irregular. São exemplos disso as redes: aSoC (LIANG 2000), QNoC (BOLOTIN 2004) e MANGO (BJERREGAARD 2004).

4.1.3.2 Roteamento

O algoritmo de roteamento define o caminho que o pacote deve seguir da origem até o destino. Dependendo de onde é tomada esta decisão, ele pode ser definido como origem ou distribuído. No roteamento definido na origem, todo o caminho é decidido no nodo de origem, enquanto no roteamento distribuído essa decisão é tomada a cada passo. Dependendo de como é definido o caminho para enviar os pacotes, o roteamento pode ser definido como determinístico ou adaptativo. No determinístico, o caminho é definido com base na localização dos nodos de origem e de destino na rede. No caso do adaptativo, o caminho também é função do tráfego da rede (DUATO 2003) (NI 1993).

Um roteamento determinístico bastante utilizado em NoCs é o XY. Neste tipo de roteamento, os pacotes percorrem primeiro as linhas (coordenada X) para somente depois percorrerem as colunas (coordenada Y) até atingir o destino.

O roteamento da rede, em árvore gorda, SPIN (GUERRIER 2000) é dinâmico (realizado em tempo de execução) e pode ser adaptativo e distribuído ou determinístico e origem, dependendo do sentido de deslocamento da mensagem. Os pacotes que, ao saírem de seu núcleo de origem, sobem a rede, utilizam o mecanismo de roteamento adaptativo (onde a mensagem pode ser transmitida por qualquer canal livre) e distribuído (são os roteadores que decidem as portas de saída a serem utilizadas). Para os pacotes que descem, o roteamento é determinístico e origem. A porta de saída a ser utilizada é identificada no endereço do destinatário e já definida pelo emissor do pacote.

Na rede, em grelha 2D, SoCBUS (WIKLUND 2003) o roteamento é XY adaptativo (variação não-determinística do XY determinístico). As decisões são baseadas somente na origem e no destino. Entretanto, se existir mais de uma direção possível para um pacote em um determinado ponto do caminho, a direção de envio será escolhida seguindo um escalonamento circular (tipo Round-Robin). Caso a primeira escolha esteja ocupada, uma segunda seleção é feita, e estando esta também ocupada um sinal é enviado de volta para emissor.

A rede Nostrum (MILLBERG 2004), também em grelha 2D, utiliza um roteamento baseado no congestionamento, o "batata-quente" (do inglês – *hot potato*). A idéia principal desse tipo de roteamento é que, conhecendo o tráfego dos nodos vizinhos, em caso de congestionamento, o roteador envie o pacote para o vizinho menos sobrecarregado, evitando as áreas congestionadas. Esse envio é feito de forma imediata, pois não são usados *buffers* nos roteadores.

Outra rede em grelha 2D, a HERMES (MORAES 2004), usam alternativas parcialmente adaptativas (MELLO 2004) ao roteamento XY: (*i*) primeiro pelo oeste; (*ii*) por último o norte e (*iii*) primeiro no sentido negativo. Na comparação destas três alternativas com o XY determinístico, os protocolos adaptativos apresentam uma maior rapidez no envio, mas globalmente, o determinístico mostra-se superior. Isso se deve ao fato dos protocolos adaptativos possuírem uma tendência de concentrar o tráfego no centro da rede, provocando assim um congestionamento nesta área.

Tabela 4.1: Características de algumas das principais NoCs da literatura

Rede	Topologia	Roteamento	Chaveamento	Suporte a QoS	Referências
SPIN	Árvore gorda	Adaptativo e Determinístico	*Wormhole*	–	(Guerrier 2000)
aSoC	Grelha 2D[5]	Estático e Determinístico	Circuito[6]	Chaveamento por circuito	(Liang 2000)
Toroidal	Torus 2D	Determinístico	*Wormhole*	Canais virtuais	(Dally 2001)
Æthereal	Customizável[7]	Determinístico	Circuito	Chaveamento por circuito	(Rijpkema 2001) (Goossens 2005)
Nostrum	Grelha 2D	Adaptativo	*Wormhole* ou TDMA[8]	Circuito virtual [9]	(Kumar 2002) (Millberg 2004)
Octagon	Anel cordal	Adaptativo	Circuito ou pacotes	Chaveamento por circuito	(Karim 2002)
Proteo	Anel bidirecional	Determinístico	*Wormhole*	S.I.[10]	(Saastamoinen 2002) (Sigüenza-Tortosa 2004)
CHAIN	Customizável[7]	Determinístico	*Wormhole*	Canais virtuais	(Bainbridge 2002) (Felicijan 2004)
Xpipes	Customizável[7]	Determinístico	*Wormhole*	–	(Dall'Osso 2003)
SoCIN	Grelha 2D	Determinístico	*Wormhole*	–	(Zeferino 2003b)
SoCBUS	Grelha 2D	Adaptativo	Circuito	Chaveamento por circuito	(Wiklund 2003)
T-SoC	Árvore gorda	Adaptativo	*Wormhole*	Canais virtuais	(Pande 2003)
HERMES	Grelha 2D	Parcialmente adaptativo	*Wormhole*	–	(Moraes 2004)
QNoC	Grelha 2D[5]	Determinístico	*Wormhole*	Canais virtuais	(Bolotin 2004)
MANGO	Grelha 2D[5]	Determinístico	*Wormhole* ou ALG [11]	Canais virtuais	(Bjerregaard 2004) (Bjerregaard 2005)

[5] Pode ser regular ou irregular.
[6] Sendo o roteamento estático, o chaveamento também é definido em tempo de projeto, sem ser preciso enviar cabeçalho para estabelecer o circuito.
[7] Em tempo de projeto pode-se definir a topologia em: grelha, torus, hipercubo, árvore gorda...
[8] *Time Division Multiple Access* – reserva uma fatia de tempo para cada canal virtual, estabelecendo um circuito virtual da origem até o destino.
[9] Estabelece um circuito virtual através de mecanismos de multiplexação no tempo (TDMA).
[10] (Sem Informações) - informação não disponível na literatura.
[11] *Asynchronous Latency Guarantee* – esquema que provê a garantia de latência de comunicação desde a origem até o destino.

4.1.3.3 Chaveamento

A grande maioria das NoCs existentes utilizam o chaveamento baseado em pacotes, pois o uso de pacotes permite uma utilização mais otimizada dos recursos da rede, uma vez que não há a reserva do caminho como no chaveamento por circuitos. Por outro lado, algumas NoCs, como Æthereal (GOOSSENS 2005), Octagon (KARIM 2002) e SoCBUS (WIKLUND 2003) utilizam o chaveamento por circuitos pelo fato deste evitar a influência da contenção da rede na comunicação, isto é a interferência de outros tráfegos, visto que o caminho já estará todo pré-estabelecido.

Uma variação do chaveamento por circuito é utilizada pela rede aSoC (LIANG 2000). A principal diferença ocorre no fato da rede aSoC não precisar enviar um cabeçalho reservando recursos, já que os circuitos são estabelecidos pelo algoritmo de escalonamento antes da execução da aplicação, ainda em tempo de projeto. Estes caminhos são estabelecidos procurando maximizar o paralelismo nas comunicações.

As redes MANGO (BJERREGAARD 2005) e Nostrum (MILLBERG 2004) utilizam técnicas de chaveamento baseadas no controle de acesso. A MANGO utiliza mecanismos de prioridade e canais virtuais para estabelecer circuitos e garantir a latência de comunicação da origem até o destino. Esse esquema é denominado de ALG (em inglês, *Asynchronous Latency Guarantee*). A rede Nostrum utiliza técnicas de multiplexação no tempo para estabelecer circuitos virtuais entre a origem e o destino das mensagens. Na Nostrum também existe outra possibilidade de projeto que é o chaveamento por pacotes, utilizando a técnica de *wormhole*.

A técnica de *wormhole* é mais utilizada atualmente pelas NoCs que fazem chaveamento por pacotes. A vantagem, em relação a outras abordagens, é que os requisitos de *buffer* são menores, possibilitando a construção de roteadores pequenos e rápidos.

4.1.3.4 Suporte a Serviços de QoS

QoS é o serviço de quantificação que é provido pela rede para a aplicação. No contexto de NoC, essa aplicação está vincula a um ou mais núcleos. Segundo Bjerregaard e Mahadevan (BJERREGAARD 2006) isto envolve dois aspectos: (*i*) definição dos serviços através de uma quantificação e (*ii*) negociação dos serviços. Esses serviços podem ser: baixa latência, alta vazão, consumo de potência reduzido, limite de *jitter*, entre outros. A negociação implica no equilíbrio entre a demanda do serviço pelo núcleo e a disponibilidade deste serviço na rede. Essencialmente, prover QoS implica na reserva de uma certa parte dos recursos da rede para uma determinada conexão. Esses recursos consistem, por exemplo, de espaço de memorização nos *buffers* e largura de banda (*bandwidth*).

Em sistemas síncronos, essa reserva de banda é feita, geralmente, por divisão multiplexada do tempo de acesso (TDMA – do inglês *Time Division Multiple Access*), onde em cada fatia de tempo uma única conexão será permitida. E o número de fatias de tempo destinadas a uma comunicação será proporcional à largura de banda necessária. As redes aSoC (LIANG 2000), ÆTHEREAL (RIJPKEMA 2003) e Nostrum (MILLBERG 2004) utilizam variantes do TDMA para prover garantias de largura de banda para cada conexão.

Entretanto, para redes assíncronas o TDM não é aplicável, pois o mesmo requer sincronização global entre os elementos da rede. Para essas redes, uma alternativa é

utilizar um algoritmo de escalonamento que trate as requisições na entrada do roteador. Esse escalonamento é feito de acordo com o nível de QoS exigido por cada tipo de comunicação. Algumas redes utilizam canais virtuais no suporte a este escalonamento. Os canais virtuais permitem isolar os tráfegos entre os diversos níveis de comunicação, favorecendo, dessa forma, a simplificação do processo de escalonamento. Cada canal virtual ganha acesso ao canal físico durante certo período. Esse acesso é realizado através de estratégias que podem considerar uma fatia de tempo para cada canal virtual ou a prioridade das mensagens envolvidas. Exemplos de redes que adotam essas técnicas são: CHAIN (FELICIJAN 2004), T-SoC (PANDE 2003), QNoC (BOLOTIN 2004) e MANGO (BJERREGAARD 2004).

O algoritmo de escalonamento tem sua eficiência dependente também de uma política de memorização dos pacotes nos roteadores. Isso é necessário quando a quantidade de tráfego das diversas comunicações que chegam ao roteador excedem a capacidade da largura de banda dos respectivos canais de saída. Nessa situação alguns canais de entrada serão atendidos antes dos outros. E aqueles canais que foram preteridos precisam armazenar temporariamente os pacotes pendentes em *buffers* de entrada até que eles possam ser enviados para o próximo nodo.

Bjerregaard e Mahadevan (BJERREGAARD 2006) enfatizam que *buffers* são parte integrante de qualquer roteador de rede e, em geral, constituem a maior parte da área ocupada desses roteadores. Dessa forma, eles consideram vital equilibrar a quantidade de *buffers* com os requisitos de desempenho, considerando aspectos como o tamanho e a localização desses *buffers* nos roteadores. Kumar et al. (KUMAR 2002) mostram também que o simples aumento dos *buffers* não resolve o problema de congestionamento. Por outro lado, *buffers* são úteis na absorção de tráfegos em rajadas.

Segundo Felicijan e Furber (FELICIJAN 2004), o gerenciamento dessa memorização temporária nos *buffers* deve tratar também efetivamente de questões de QoS, tais como: (*i*) prover espaço de memorização suficiente, com garantia de QoS, para acomodar qualquer excesso de tráfego nos canais de entrada dos roteadores, e (*ii*) assegurar que não ocorrerão situações de bloqueio de cabeça de linha (HoL) dos pacotes.

Rijpkema et al. (RIJPKEMA 2003) caracterizam a natureza da QoS em relação a NoCs. Eles classificam em duas classes básicas de serviços: (*i*) melhor esforço (BE – do inglês, *best effort*) e (*ii*) serviços garantidos (GS – do inglês, *guaranteed services*). Basicamente, GS implica em previsibilidade, enquanto BE considera a utilização média dos recursos. Eles identificam também diferentes níveis de compromissos que tem efeitos sobre a previsibilidade e o comportamento da comunicação: (*i*) corretude dos resultados; (*ii*) completude da transação e (*iii*) limites do desempenho. E advogam, ainda, que em NoCs é necessária a combinação das classes BE e GS. Neste mesmo trabalho eles aplicam essa combinação na rede Æthereal, cujo fluxo de dados é baseado em alocação de fatias (*slots*) de tempo nas tabelas de controle dos roteadores. O tráfego GS é orientado à conexão e tem reserva das fatias de tempos em cada roteador no caminho entre a origem e o destino. O tráfego BE não faz reserva dessas fatias. No tráfego que combina as duas classes (BE e GS), o tráfego BE usa qualquer fatia reservada pelo GS que não tenha sido utilizada.

Bjerregaard e Mahadevan (BJERREGAARD 2006) apresentam algumas características dos serviços de melhor esforço e dos serviços garantidos. Apesar dos serviços do tipo BE aparentarem não ter nenhuma garantia, muitas NoCs consideram

que BE inclui o tráfego para o qual existem garantias de corretude e completude, enquanto GS incluiria garantias adicionais de desempenho. A comunicação no GS deve ser logicamente independente dos outros tráfegos do sistema, sendo necessário um roteamento orientado à conexão. Para isto, essas conexões são instanciadas através de circuitos virtuais, que utilizam recursos lógicos independentes para evitar contenção.

Æthereal (RIJPKEMA 2003), Nostrum (MILLBERG 2004) e aSoC (LIANG 2000) são exemplos de redes que utilizam técnicas de TDM para implementar roteamento orientado à conexão e, assim, oferecer suporte à garantia de largura de banda para essas comunicações. Já na rede MANGO (BJERREGAARD 2004) o GS é oferecido através de uma seqüência de canais virtuais que estabelecem conexões virtuais fim-a-fim. A utilização de canais virtuais para estabelecer um roteamento orientado à conexão também é feita pelas redes CHAIN (FELICIJAN 2004), QNoC (BOLOTIN 2004) (ROSTISLAV 2005) e ANoC (BEIGNÉ 2005). A diferença é que estas últimas redes não fornecem serviço do tipo GS, mas fornecem diversos níveis do tipo BE. Elas oferecem serviços diferenciados para os diversos tipos de comunicação através da priorização de filas nos canais virtuais.

4.1.4 Nível de Enlace

O nível de enlace é o nível mais baixo de uma NoC. As pesquisas exploram as ligações entre os nodos. Essas ligações consistem de um ou mais canais, que podem ser físicos ou virtuais. Algumas das áreas de pesquisas no nível de enlace são: (*i*) sincronização; (*ii*) implementação; (*iii*) confiabilidade e (*iv*) codificação dos dados.

Na sincronização em NoCs o conceito de GALS vem emergindo como a solução mais vantajosa. Bjerregaard e Mahadevan (BJERREGAARD 2006) apontam como maior vantagem o fato que, com exceção da corrente de fuga, não há consumo de potência quando os canais não estão transmitindo dados (*idle*) quando se utiliza circuitos assíncronos dentro do conceito de GALS. Dentre as NoCs que são baseadas em técnicas de circuitos assíncronos estão: CHAIN (FELICIJAN 2003); MANGO (BJERREGAARD 2005); ANoC (BEIGNÉ 2005); QNoC (ROSTISLAV 2005); SoCIN (ZEFERINO 2004b).

Na área de implementação, tem sido investigadas questões como: segmentação de fios, *pipelining*. Na área de confiabilidade são estudadas questões de detecção e correção de erros, interferência (em inglês, *crosstalk*) entre os fios devido a crescente miniaturização dos componentes nos chips. Já a codificação de dados tem sido investigada por alguns autores (JANTSCH 2005)(PALMA 2006) como forma de reduzir o consumo de potência através de uma menor variação no valor dos bits dos dados transmitidos.

4.2 Espaço de Projeto em Sistemas Embarcados baseados em NoCs

As NoCs são parte integrante de SoCs. Dessa forma, alguns conceitos da arquitetura e da metodologia de projetos de SoCs podem ser aplicados no projeto de NoCs. A construção de sistemas embarcados inclui diversas atividades, como: especificação, particionamento, escalonamento, alocação, síntese, mapeamento e validação.

4.2.1 Fluxo de Projetos de NoCs

Diversos autores propõem fluxos de projetos para NoCs. Em comum eles têm, geralmente, uma dependência forte na etapa de modelagem, isto é, quando esta é bem realizada os resultados obtidos serão melhores.

Pop e Kumar (POP 2004) apresentam uma metodologia de projeto baseado em plataforma, que se baseia no reuso de núcleos IP e na especialização da estrutura de comunicação. O reuso de núcleos IP e o alto grau de escalabilidade que a NoC provê permitem um refinamento da rede para certas aplicações. A implementação de NoCs pelo reuso maciço de núcleos IP de aplicações similares, que tenham sido utilizados anteriormente em outros projetos, resultam em um menor *time-to-market*. Estas características são muito diferentes dos projetos baseados em barramento no qual a escalabilidade é limitada. A Figura 4.3 apresenta algumas etapas exigidas do fluxo de projeto para uma dada aplicação a ser implementada em uma NoC. O ponto inicial do fluxo de projeto é uma descrição genérica da rede que especifica características gerais da arquitetura tais como: topologia, restrições de projeto no que diz respeito ao tamanho dos núcleos IP, princípios de comunicação, etc. A próxima etapa de projeto tem como função especializar a NoC genérica para uma aplicação ou domínio de aplicações. Esta especialização implica em: decidir o tamanho da rede; selecionar os núcleos IP adequados; definir as características dos roteadores (incluindo os algoritmos de roteamento); selecionar o protocolo de comunicação que será estabelecido para executar tal aplicação. As etapas seguintes, depois da especialização da NoC, são: (*i*) mapeamento; (*ii*) particionamento e (*iii*) escalonamento. As funções destas etapas são: conectar os núcleos de *hardware* em posições adequadas na rede e ordenar adequadamente as tarefas, explorando o paralelismo, para a execução da aplicação, dentro dos requisitos de projeto. Após a execução das etapas de mapeamento e escalonamento, são realizadas iterações para refinamento e novas otimizações para assegurar que o modelo da NoC mapeado e escalonado respeite as restrições de projeto.

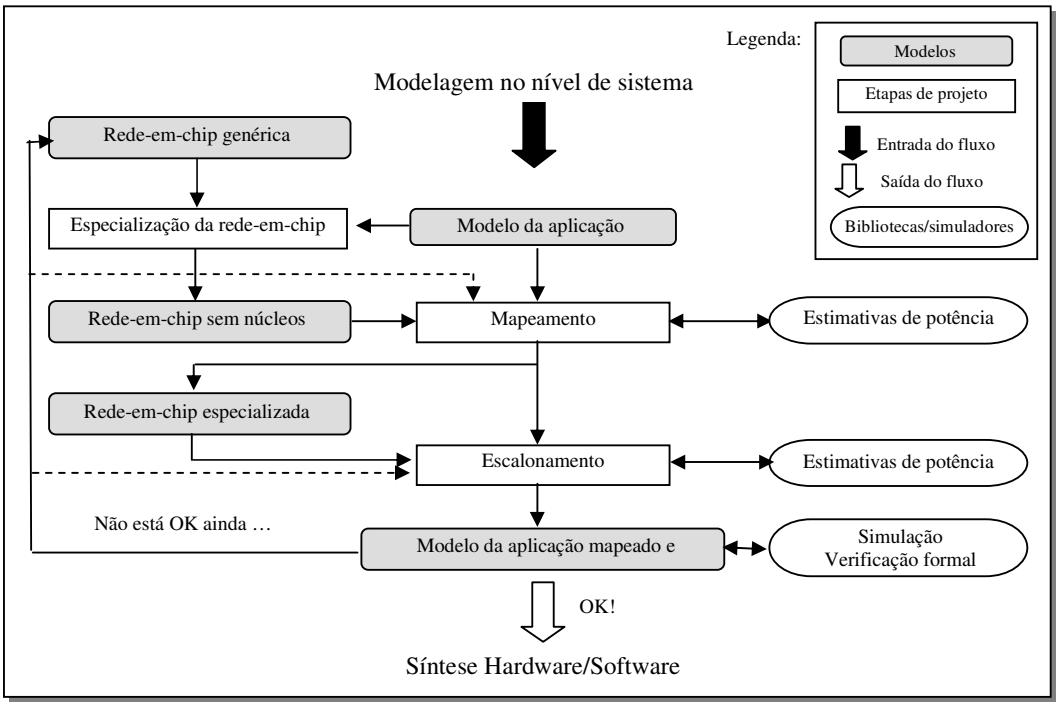

Figura 4.3: Fluxo genérico de projeto de NoCs (POP 2004).

O fluxo de projeto da rede Æthereal (GOOSSENS 2005), ilustrado na Figura 4.4, utiliza componentes parametrizàveis e reutilizáveis. Em tempo de projeto, esses componentes são instanciados e conectados em uma determinada topologia. Além disso, os IPs são conectados às suas respectivas interfaces de rede, ou seja, é feito o mapeamento. O resultado é uma descrição em hardware da estrutura da NoC. Os roteadores e interfaces de rede podem ser parametrizados, assim como algumas características da própria NoC, como: tamanho das tabelas de alocação em cada roteador, freqüência de operação. Os roteadores são parametrizáveis em relação ao número de portas de entrada e saída e ao tamanho das filas de tráfego BE. Nas interfaces de rede podem ser configurados parâmetros, tais como: número de portas, números de conexões por porta, tamanho dos *buffers* por conexão, além do tipo de porta. Todas as instâncias podem ser reconfiguradas em tempo de execução. As conexões também podem ser reconfiguradas, por exemplo, com diferentes garantias de vazão e latência através de ajustes no hardware da NoC. Desde que este ajuste não exceda os limites do hardware da rede, características, como: número de conexões por portas, tamanho da tabela de alocação de fatias (*slots*) podem ser alteradas para atender as novas necessidades da conexão.

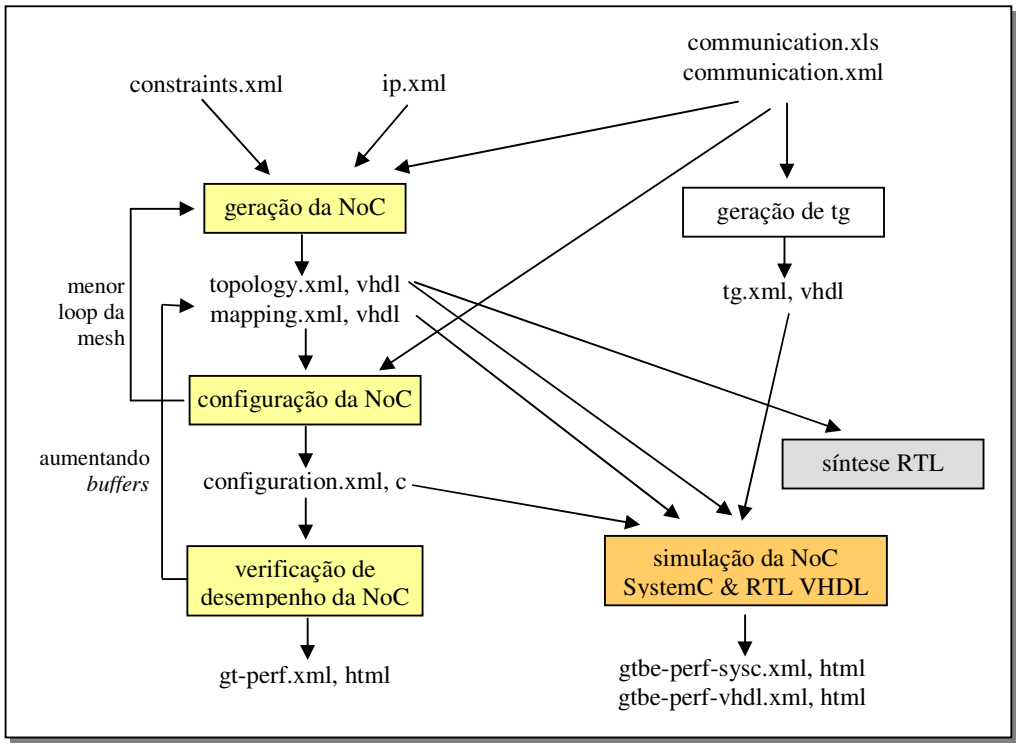

Figura 4.4: Fluxo de projeto da NoC Æthereal (GOOSSENS 2005).

Os autores enfatizam também que a configuração e a geração da rede são interdependentes, e parte de um complexo problema de otimização de recursos, que deve encontrar a melhor combinação de topologia, mapeamento e atribuições de vazão, minimizando o número de roteadores, interfaces de rede, tamanho de *buffers* e latências. Dessa forma, se esse processo for feito corretamente, por construção, nenhuma simulação de desempenho será requerida para as conexões. A simulação será útil para checar se o comportamento da comunicação dos núcleos IPs foi corretamente

caracterizado. Para o tráfego GS, isso poderá ser checado independentemente para cada conexão. O usuário pode executar o fluxo de forma completamente automática a partir dos arquivos de entrada (restrições, IP, comunicações), ou pode criar/modificar determinados arquivos intermediários (topologia, mapeamento, configuração). Mas, em ambos os casos, o fluxo executa automaticamente. Ou seja, mesmo que haja uma intervenção manual na configuração da NoC, por exemplo, a verificação automática de desempenho ainda será realizada.

Em Bolotin et al. (BOLOTIN 2004) é apresentado o fluxo de projeto da rede QNoC, conforme mostra a Figura 4.5. Inicialmente, os módulos funcionais do sistema são definidos e conectados a uma rede ideal com largura de banda ilimitada e atraso programável. Então, a comunicação entre módulos é caracterizada através de uma análise da interconexão dos módulos e de suas especificações de tráfego. Para verificar as estimativas de tráfego é feita uma simulação do sistema, em alto nível. Na seqüência, é feito o posicionamento dos módulos buscando uma minimização da densidade de tráfego. Depois de determinados os requisitos de comunicação e realizado o posicionamento, a rede QNoC pode ser construída de acordo com: (*i*) o número e posicionamento dos módulos e, (*ii*) os níveis de serviço de QoS que serão suportados. Feito isso, é realizada uma otimização com a remoção dos canais e nodos que não sejam necessários e um ajuste da capacidade de largura de banda dos demais canais para a carga estimada. Finalizando o fluxo, a rede é simulada e analisada de forma mais precisa por um simulador de rede.

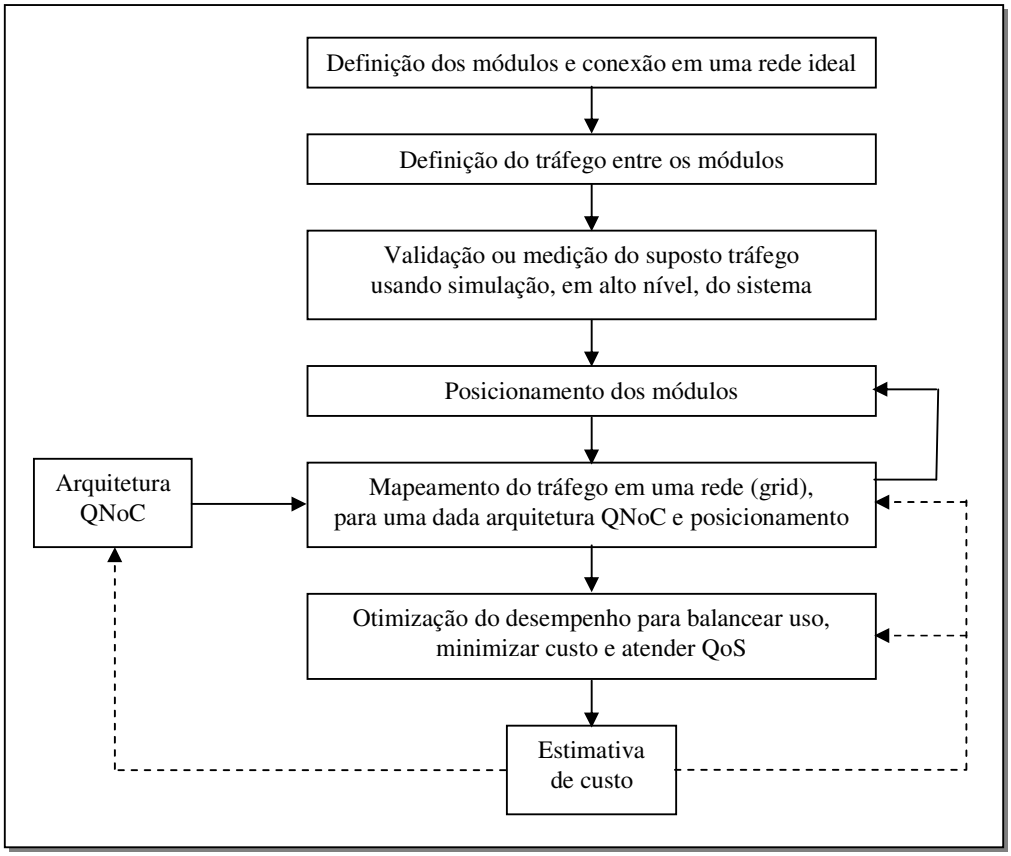

Figura 4.5: Fluxo de projeto da rede QNoC (BOLOTIN 2004).

O fluxo de projeto da QNoC é semelhante ao fluxo da Æthereal. Mas, segundo os autores deste último (GOOSSENS 2005), a diferença consiste na simulação para verificar o desempenho do sistema, que na QNoC é expressa como uma propriedade estatística. E também porque no fluxo de projeto da Æthereal a geração da rede com mapeamento dos IPs e o balanceamento da largura de banda são executados em paralelo, enquanto no fluxo da QNoC esses eventos são seqüenciais.

A metodologia de projeto da Xpipes (DALL'OSSO 2003) é dividida em duas partes. Na primeira é realizada uma descrição parametrizável, em SystemC, de todos os componentes da rede. Essa descrição serve como um modelo de projeto de propósito geral. Então, em tempo de instanciação, os componentes individuais são conectados e configurados de acordo com as especificações dos parâmetros.

Xpipes Lite (STERGIOU 2005) é uma biblioteca SystemC parametrizável e sintetizável de NoC, com módulos de interface de rede, de canais e de chaveamento (*switches*), otimizados para baixa latência e alta freqüência de operação. A comunicação é através de chaveamento de pacotes, com roteamento determinístico e controle de fluxo do tipo *wormhole*.

A Figura 4.6 ilustra o fluxo de projeto da Xpipes. O compilador instancia a NoC desejada através de arquivos de configuração que são gerados automaticamente em um passo anterior. Os arquivos de configuração contêm parâmetros: (*i*) das interfaces de rede; (*ii*) dos canais e (*iii*) de chaveamento. Os parâmetros das interfaces de rede incluem: tipo (mestre, escravo ou ambos), endereço na NoC e ponto de conexão. Nos canais são configurados: ponto de conexão, número de estágios de *pipeline*. Já no chaveamento, a configuração envolve: número de entradas e de saídas, capacidade dos *buffers* de saída. Além desses parâmetros, outros mais gerais também podem ser configurados, como: tamanho do *flit*, espaço de endereçamento dos núcleos.

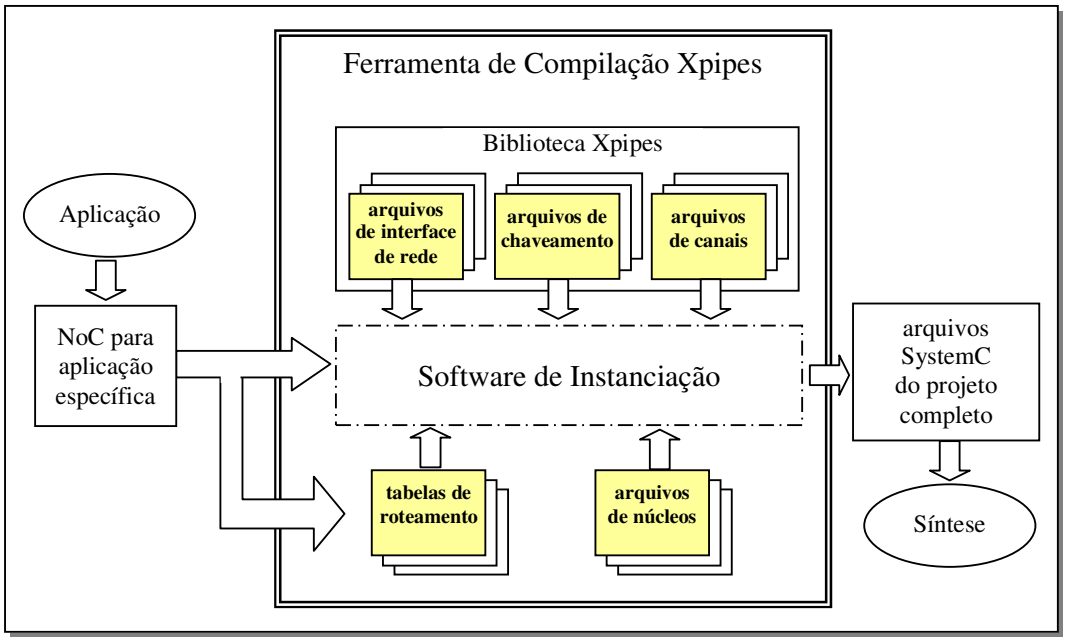

Figura 4.6: Fluxo de projeto da rede Xpipes (STEGIOU 2005).

O princípio do fluxo de projeto da Proteo (SIGÜENZA-TORTOSA 2002) é semelhante ao da Xpipes, consistindo de uma biblioteca de componentes pré-definidos parametrizáveis que permitem a implantação de diferentes topologias, protocolos e configurações. No entanto, os autores da Xpipes advogam que, nesta metodologia, a abordagem é levada a limites maiores, através da instanciação de uma NoC específica para a aplicação proveniente de uma biblioteca de macros sintetizáveis.

As etapas de ***mapeamento*** e ***escalonamento*** mostradas na Figura 4.3 são umas das mais difíceis tarefas de serem realizadas no contexto do projeto. Um mapeamento e um escalonamento adequados são cruciais para extrair o máximo do desempenho de uma aplicação em uma dada plataforma computacional. Essas etapas são realizadas baseadas em modelos que descrevem a arquitetura, as tarefas e o comportamento delas em relação à comunicação e à computação, por exemplo.

Além dessas duas etapas, a de ***particionamento*** também é fundamental quando são associadas diferentes tarefas aos elementos de processamento nos núcleos IP.

Modelo da Arquitetura da NoC

O modelo da arquitetura da NoC é geralmente especificado por um grafo dirigido $N(P,C)$, onde P é o conjunto de vértices que representam o conjunto de elementos processantes (PE – do inglês, *Processing Elements*) e C é o conjunto de arestas que representa os canais de comunicação entre os PE (POP 2004). Um vértice P pode ser desde diversos tipos de processadores até núcleos IP heterogêneos e uma aresta C pode ser desde canais ponto a ponto até barramentos.

Modelo de Tarefas

No fluxo de projeto da Figura 4.3, a entrada é uma modelagem no nível de sistema da aplicação, ou seja, uma modelagem da aplicação em um alto nível de abstração. Este modelo geralmente é representado por um grafo dirigido $G(V,E)$, onde V é o conjunto de vértices que representa o conjunto de tarefas da aplicação e E é o conjunto de arestas que representa as comunicações entre as tarefas.

Existem alguns trabalhos que utilizam outros tipos de modelos de tarefas (HU 2003a) (HU 2004b) (KREUTZ 2005b). Existem alguns parâmetros que podem ser associados a priori no modelo. Estes parâmetros podem ser: tempo de execução da tarefa para cada PE, *deadline*, período, volume de comunicação de cada aresta, consumo de energia por ciclo dentro de uma certa tensão, consumo de energia por PE, consumo de energia por bit ignorando distâncias de comunicação, memória para armazenamento de tarefas e dados, etc.

Modelo de Comunicação

Murali e DeMicheli (MURALI 2004a) propõem uma descrição da aplicação através de um grafo de núcleos[12] (em inglês, *core graph*), que é utilizado também por Murali, Benini e DeMicheli em (MURALI 2005). Hu e Marculescu (HU 2003a) propõem, por sua vez, um grafo de caracterização da aplicação (APCG – do inglês, *APplication Characterization Graph*). Para representar essas descrições pode-se utilizar o Modelo de Comunicação com Pesos (CWM – do inglês *Communication Weighted Model*), usado em (MARCON 2005a).

[12] A descrição de uma aplicação é feita através de grafos de tarefas. Em casos particulares, onde cada núcleo tem somente uma tarefa associada, o grafo da aplicação pode representar o grafo de núcleos.

O CWM realiza a atividade de mapeamento, modelando uma aplicação apenas pela sua comunicação e dependência, especificamente, pelo somatório de todas as comunicações que ocorrem de um núcleo para outro. Essa comunicação é dada pela soma de todos os bits de todos os pacotes transmitidos de um núcleo para outro. Com este modelo, os autores avaliam a qualidade de mapeamentos frente ao requisito de minimização do consumo de energia.

O modelo CWM pode ser representado por um grafo de comunicação com pesos (CWG – do inglês *Communication Weighted Graph*), que descreve a aplicação através de seus núcleos e da comunicação entre estes núcleos.

Definição 4.1 – Um grafo de comunicação com pesos é um grafo dirigido representado pelo par CWG = <N,W>, onde N = $\{n_1,n_2, \ldots , n_c\}$ é o conjunto de vértices e W = $\{W_{12}, W_{13}, \ldots , W_{1C}, W_{21}, W_{23}, \ldots , W_{2C}, \ldots , W_{C1}, W_{C2}, \ldots , W_{CC-1}\}$ é o conjunto de arestas. O conjunto de vértices representa os núcleos da aplicação e o conjunto de arestas representa as comunicações entre cada par de núcleos. Seja w_{ijq} o número de bits da q-ésima mensagem m_q que trafega no meio físico do núcleo n_i para o núcleo n_j, e seja k_{ij} o total de mensagens enviadas de n_i para n_j, então $\omega_{ij} = \sum_{q=1}^{k_{ij}} w_{ijq}$ é o total de bits enviados de n_i para n_j. Cada aresta $w_{ij} \in W$ é uma tripla $w_{ij} =< n_i, n_j, \omega_{ij} >$, tal que $n_i \neq n_j$ e $\omega_{ij} > 0$. No limite máximo do número de comunicações, que ocorre quando todos os núcleos estão conectados entre si, tem-se $|W| = 2 * C_C^2$. Isso significa que o número de arestas de um CWG é igual a duas vezes a combinação de todos os núcleos dois a dois.

A Figura 4.7 apresenta um exemplo de uma aplicação sintética e sua representação através do modelo CWM e do respectivo grafo do tipo CWG.

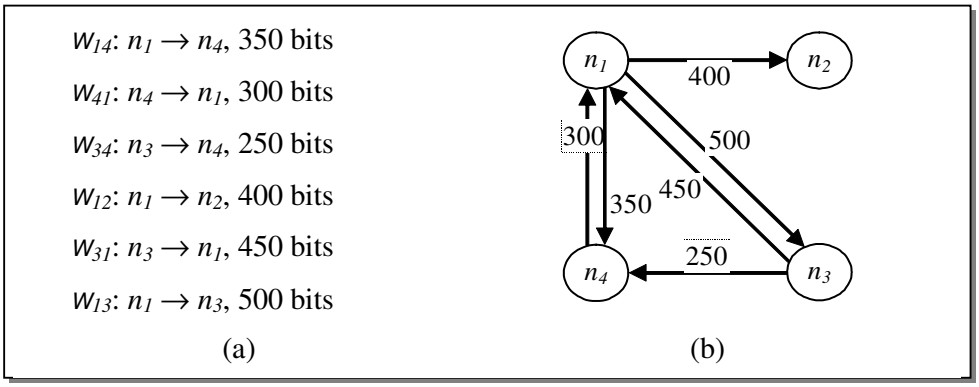

Figura 4.7: Exemplo de uma aplicação sintética: (a) CWM com a descrição da comunicação e (b) representação através de um CWG (MARCON 2005a).

O conjunto de mensagens envolvidas na aplicação sintética é mostrado na Figura 4.7(a), onde são descritos os núcleos envolvidos em cada comunicação e as respectivas quantidades de bits transmitidas no meio físico. Os vértices do grafo da Figura 4.7(b) representam os núcleos, e os valores que estão nas arestas representam a quantidade total de bits transmitidos de um vértice para outro (peso de comunicação) e a dependência entre eles.

Mapeamento

Mapeamento é uma atividade bastante ampla, mesmo dentro do escopo de projeto de sistemas computacionais. No contexto de mapeamento em NoCs, o mapeamento é a atividade que associa núcleos a roteadores, sendo que cada núcleo é conectado à infra-estrutura de comunicação por um canal de comunicação.

Os canais da infra-estrutura de comunicação são os pontos de acesso dos roteadores, onde os núcleos podem ser conectados para utilizar a infra-estrutura de comunicação (NoC). Assim, o mapeamento se refere à associação de núcleos a roteadores da NoC. O comportamento dos núcleos da aplicação, bem como a interação entre eles, que reflete a computação e a comunicação da aplicação, é descrito num formato interno próprio. Por sua vez, o mapeamento gera como saída um novo formato interno (modelo ou grafo), onde cada núcleo passa a ser associado a um roteador.

O mapeamento requer o conhecimento de requisitos e restrições de projeto e da arquitetura alvo. Os requisitos de projeto servem para guiar o mapeamento através de funções objetivo, que determinam o custo de cada mapa. No contexto de mapeamento de NoCs, os requisitos abordados são a minimização do consumo de energia da estrutura de comunicação, a redução do tempo de execução da aplicação, a redução de temperatura no chip, redução de latência, etc. A arquitetura alvo, por sua vez, determina qual o impacto destes requisitos. Restrições, como área, podem determinar se um núcleo pode ou não ser mapeado em um roteador.

No mapeamento, a restrição de área pode implicar que certos mapas sejam inválidos, pois certos núcleos somente poderiam ser mapeados nos roteadores cuja dimensão soada a destes núcleos não ultrapassasse a área permitida. Ao final de um mapeamento, deve ser testado se o mapa obtido atende aos requisitos de projeto. Em caso positivo, o fluxo de projeto segue e o formato interno de saída do mapeamento se torna entrada de novas atividades. No caso do mapa obtido não atender aos requisitos de projeto, a atividade de mapeamento pode ser refeita para obter novo mapa de forma equivalente a da Figura 4.3. Este processo pode ser refeito enquanto não forem atendidos os requisitos de projeto, ou enquanto não for atingido um limite de iterações. No caso da condição de parada ser pelo número de iterações, indica o fluxo de projeto não conseguiu oferecer um mapa que atenda aos requisitos. A cada novo laço, que envolve mapeamento, são explorados novos compromissos entre computação e comunicação.

Particionamento

O particionamento é a atividade de associar tarefas a elementos de processamento dos núcleos. E é assim denominado por poder gerar diferentes blocos, ou seja, partições. E uma vez associadas as tarefas aos núcleos, a etapa seguinte é a de escalonamento.

Nos casos onde os núcleos puderem possuir mais de uma tarefa a etapa de particionamento deve ser feita antes do escalonamento. Ela pode ser feita antes ou depois do mapeamento dos núcleos na NoC, mas será obrigatoriamente antes do escalonamento das tarefas.

Escalonamento

O escalonamento é a ordenação temporal das tarefas e da comunicação entre os recursos aos quais tais tarefas estão associadas. No escalonamento, devem existir mecanismos de exclusão mútua entre as execuções das tarefas em um mesmo recurso. As tarefas devem ser escalonadas de forma a atender os requisitos de projeto. Por exemplo, se o requisito de projeto é obter um melhor desempenho, o processo de escalonamento deve obter o máximo de paralelismo das tarefas ditas independentes para aquele dado tempo. Em um determinado intervalo de tempo, as tarefas independentes devem ser executadas em recursos diferentes a fim de obter o máximo paralelismo, e conseqüentemente, obter um melhor desempenho. O paralelismo máximo em NoCs é obtido quando, para cada núcleo, somente uma tarefa é associada. Neste caso o escalonamento local (dentro de cada núcleo) não é necessário.

4.2.2 Ferramentas para o Projeto de NoCs

Nos projetos baseado em NoCs, é necessário que o projetista utilize diversas ferramentas. Neste contexto, segundo Pop e Kumar (POP 2004), as mais importantes ferramentas são para:

- seleção dos núcleos IP;
- avaliação e verificação dos núcleos IP;
- configuração da NoC;
- simulação arquitetural da NoC;
- estimativa do consumo de potência da NoC;
- mapeamento e escalonamento da aplicação.

Muitas outras ferramentas existentes de síntese de hardware e software também precisam ser integradas, para implementação de uma ferramenta de projeto de um sistema baseado em NoCs. Segundo Kumar e Pop (POP 2004), o ponto chave do projeto em NoC é o desenvolvimento de uma ferramenta eficiente de mapeamento de aplicações em múltiplos recursos computacionais.

Na exploração do espaço de projeto de NoCs os parâmetros que afetam os requisitos dessas redes são vários, como por exemplo: o tipo de arquitetura de interconexão usada, a topologia da mesma, a largura de banda e tráfego de mensagens na arquitetura de interconexão, a distribuição de tarefas entre os processadores, o tamanho dos *buffers* dos roteadores, o posicionamento dos recursos na rede, dentre outros. Em função disso é necessária a utilização de uma metodologia para a exploração desse espaço de projeto.

Nas várias metodologias, que buscam explorar o espaço de projeto em NoCs, é vital a utilização do mapeamento automático de núcleos IP na rede. O objetivo é encontrar uma solução ótima para uma dada função. Existem diferentes tipos de abordagens que usam diversas funções objetivo para otimizar um dado sistema que utiliza NoC. Na seqüência são citados alguns trabalhos que apresentam alguns métodos de escalonamento, mapeamento e garantias de qualidade de serviço e *deadlines*. A Tabela 4.2, baseada em tabela similar apresentada por Brião (BRIÃO 2005), resume os métodos apresentados por estes trabalhos, apresentando suas características e peculiaridades.

Tabela 4.2: Propostas de Metodologias para Exploração do Espaço de Projeto de NoCs

Autor	**Metodologia**	**Modelo de Entrada**	**Topologia**	**Heurística**	**Restrições**	**Objetivo**	**Suporte a Tempo Real**
Ascia 2004	mapeamento de núcleos IP	CWM	grelha regular	algoritmo genético	desempenho e consumo de energia	encontrar compromisso entre desempenho e consumo de energia	não
Bolotin 2004	personalização de NoC	[13]	irregular e personalizável	personalizável	largura de banda ou desempenho	garantir QoS, minimizando energia, tráfego e área	*hard deadlines*
Hu 2003b	mapeamento de núcleos IP	CTG	grelha regular	*branch-and-bound*	largura de banda e desempenho	compromisso entre o consumo de energia e o desempenho	não
Hu 2004a	otimização e alocação de *buffers* na NoC	CWM	grelha regular	*greedy*	desempenho	minimizar área	não
Hu 2004b	particionamento e escalonamento de tarefas	CTG	grelha regular	*greedy*	desempenho	reduzir consumo de energia	*hard deadlines*
Hung 2004	mapeamento de núcleos IP	CWM	grelha regular	algoritmo genético	comunicação	minimizar comunicação entre núcleos IP e balanceamento de temperatura.	não

[13] grafo de núcleos IP com tráfego modelado entre eles.

Autor	Metodologia	Modelo de Entrada	Topologia	Heurística	Restrições	Objetivo	Suporte a Tempo Real
Kreutz 2005b	mapeamento de tarefas/ escolha de topologia	ACP	grelha regular, árvore gorda, *torus*	*tabu search*	desempenho	minimizar latência e consumo de energia	não
Lei 2003	mapeamento de núcleos IP	CWM	grelha regular	algoritmo genético	desempenho	minimizar o tempo global de execução do sistema	*soft deadlines*
Marcon 2005b	mapeamento de tarefas	CDCM	grelha regular	*simulated annealing*	desempenho e energia	minimização de energia e tempo de execução	não
Murali 2004a	mapeamento de núcleos IP	CWM	grelha regular	NMAP [14]	largura de banda	minimizar o atraso médio de comunicação dividindo o tráfego de mensagens entre os núcleos	não
Nollet 2005	mapeamento (migração de tarefas)	CWM	grelha regular	*backtracking*	desempenho	minimizar o atraso de comunicação dinamicamente	não
Ogras 2005	mapeamento, roteamento e síntese de NoCs personalizadas	CWM	topologia personalizada para aplicação	*branch-and-bound*	desempenho	minimização de energia global	não
Rhee 2004	mapeamento de núcleos IP	CWM	grelha regular	MILP [15]	largura de banda	minimização de energia e de congestionamentos	não

[14] NMAP – algoritmo, apresentado pelos autores, que mapeia núcleos em NoC, considerando a largura de banda e objetivando a redução do atraso médio de comunicação.
[15] MILP – Mixed-Integer Linear Programming.

Autor	Metodologia	Modelo de Entrada	Topologia	Heurística	Restrições	Objetivo	Suporte a Tempo Real
Shin 2004	mapeamento, alocação de canal, seleção de tensão	CWM	grelha regular	algoritmo genético	desempenho	minimização de energia e redução de área	*hard deadlines*
Smit 2004	mapeamento de tarefas	CWM	grelha regular	*MinWeight*	desempenho	minimizar comunicação entre núcleos	não
Srinivasan 2005	mapeamento, roteamento e síntese de NoCs	CWM	topologia personalizada para dada aplicação	algoritmo genético	desempenho e consumo de potência	minimizar área e consumo de potência	não
Varatkar 2003	mapeamento, escalonamento e seleção de tensão	CWM	grelha regular	LS/ILP [16]	comunicação	minimizar freqüência para reduzir o consumo de potência e, diminuição do volume de comunicação do sistema	*soft deadlines*
Este trabalho	mapeamento de núcleos IP	CWM	grelha regular	*simulated annealing*	largura de banda e desempenho	garantir QoS, minimizando energia, tráfego e área	*soft deadlines*

[16] LS/ILP – List Scheduling / Integer Linear Programming

Bolotin et al. (2004) propõem uma personalização da NoC para garantir QoS. Em Hu (2004a) a metodologia proposta é a de alocação e otimização de *buffers* na NoC. Outras propostas (KREUTZ 2005a) (NOLLET 2005) (SRINIVASAN 2005) (VARATKAR 2003) utilizam o mapeamento de núcleos IP ou de tarefas como parte da metodologia. As demais têm o mapeamento como foco na metodologia.

Na maioria dos trabalhos, o CWM é utilizado como modelo de entrada. A topologia mais utilizada é a grelha regular. As heurísticas utilizadas são variadas, enquanto as restrições consideradas incluem em 75% dos casos o desempenho como fator de restrição de projeto. Outros fatores também considerados são a largura de banda e o consumo de potência.

O suporte a tempo real é abordado somente em (BOLOTIN 2004) (HU 2003a) (LEI 2003) (SHIN 2004) (VARATKAR 2003). Entretanto, outros trabalhos (KREUTZ 2005b) (MARCON 2005b) (MURALI 2004a) (NOLLET 2005) (SMIT 2004) têm como objetivos a redução da latência da rede ou do tempo médio da comunicação. Nos próximos parágrafos estes trabalhos são descritos de forma resumida, ressaltando o suporte a tempo real, se existir, e as estratégias para a redução do tempo de comunicação.

Tempo Médio de Comunicação

Kreutz et al. (2005b) apresentam um método de mapeamento de tarefas em NoCs usando heurística *Tabu Search* (GLOVER 1993). Além disso, os autores introduzem um modelo chamado ACP (em inglês, *Application Communication Pattern*) que permite capturar não somente a capacidade de comunicação, mas também a ordem da comunicação. Este modelo é usado para avaliar o consumo de energia e a latência na comparação de diferentes topologias de NoCs. Porém, este método não trata requisitos de tempo real, como os *deadlines* das mensagens. Ele busca somente uma redução no tempo médio de comunicação.

Marcon et al. (2005b) apresentam um método de mapeamento de tarefas em NoCs usando um modelo de tarefas chamado CDCM (*Communication Dependence and Computation Model*) para capturar o volume e a temporização da comunicação da aplicação. Esse modelo é representado pelas tarefas e pelas dependências entre elas, modeladas pelo CDCM. O método de mapeamento de tarefas é baseado na heurística de *Simulated Annealing* (KIRKPATRICK 1983) e nas restrições de energia e tempos de execução da aplicação. Porém, o modelo CDCM não é adequado para modelagem de aplicações que realizam razoável número de trocas de mensagens na rede. Além disto, este trabalho não tem suporte a sistemas de tempo real, pois não garante nenhum tipo de *deadline*.

Murali e DeMicheli (2004a) apresentam um algoritmo para automatizar o posicionamento de núcleos que satisfaça as restrições de largura de banda de uma NoC de topologia grelha. O algoritmo minimiza o atraso médio de comunicação global do sistema, pela divisão de tráfego de mensagens ao longo da rede. Como extensão deste trabalho, Murali e DeMicheli (2004b) desenvolveram uma ferramenta chamada SUNMAP. Esta ferramenta seleciona automaticamente a melhor topologia para uma dada aplicação e gera um mapeamento dos núcleos de hardware na topologia, de forma a atender o conjunto de restrições da aplicação, usando o mesmo modelo apresentado

em Murali e DeMicheli (2004a). Este trabalho não é aplicável em sistemas de tempo real, pois a metodologia não garante o atendimento de *deadlines*.

Nollet et al (2005) apresentam um método de gerenciamento de recursos em tempo de execução de NoCs. A contribuição principal do artigo é a descrição de uma heurística para gerenciamento de recursos na NoC, a qual é aplicada em PE reconfiguráveis de grão fino (ex: FPGA) e que utiliza mecanismos de migração de tarefas e é baseada em *backtracking*, que é uma solução clássica, a qual realiza um ou mais arranjos de tarefas antecipados para resolver o mapeamento. Esta heurística, segundo o artigo, não suporta sistemas de tempo real.

Smit et al. (2004) apresentam um algoritmo que realiza o mapeamento de vários PE em tempo de execução (mapeamento dinâmico). O algoritmo é baseado na heurística *MinWeight* e busca minimizar a quantidade total do consumo de energia e prover QoS. O objetivo é minimizar o consumo de energia. O desempenho é apenas uma restrição adicional que deve ser respeitada e não apresenta suporte a sistemas de tempo real.

Suporte a Tempo Real

Bolotin et al. (2004) propõem uma NoC com qualidade de serviço (QoS), a QNoC. O tráfego de comunicação entre módulos é classificado em quatro classes de serviços. Uma vez que as restrições da comunicação de uma aplicação-alvo tenham sido identificadas, a NoC é então personalizada para atender estas restrições. Inicialmente, os módulos são posicionados de forma a reduzir a densidade de tráfego. Depois, os roteadores e as conexões não utilizados são removidos, e a largura de banda alocada permanece com os roteadores e conexões ativas de acordo com sua carga relativa. Deste modo, a utilização da conexão é balanceada. Essa proposta apresenta um fluxo de projeto para configuração da QNoC, porém não é apresentado nenhum algoritmo de posicionamento dos núcleos IP, somente aponta que este algoritmo deverá ter o objetivo de otimizar a NoC minimizando consumo de energia, tráfego e área do sistema, garantindo QoS. Essa otimização é realizada com uma heurística personalizável, ou seja, a critério do projetista. Este deverá desenvolver um módulo para realização do mapeamento usando heurísticas de posicionamento. A partir deste posicionamento gerado pelo projetista é que alguns parâmetros da rede são configurados para atender os requisitos das aplicações e garantir o QoS. Esta metodologia é aplicável a sistemas de tempo real e garante *hard deadlines*.

Hu e Marculescu (2003a) apresentam um algoritmo de mapeamento de núcleos IP para NoCs do tipo grelha regular, cujo objetivo é minimizar a energia total de comunicação, garantindo a restrição de desempenho através da reserva de largura de banda, mas não garantem o atendimento à aplicações de tempo real.

Em Hu e Marculescu (2003b) é apresentada uma extensão de Hu e Marculescu (2003a). Nesse artigo, é apresentado um algoritmo que automaticamente faz o mapeamento de núcleos IP em uma arquitetura genérica e regular de NoC, criando uma função de roteamento determinística livre de *deadlocks* de modo que a energia total de comunicação seja minimizada. Ao mesmo tempo, o desempenho do sistema mapeado deve ser garantido para satisfazer as restrições especificadas através da largura de banda reservada. Ou seja, a garantia do atendimento às restrições temporais se dá pela reserva de recursos, especificamente, a largura de banda.

Já em Hu e Marculescu (2004a), esses mesmos autores apresentam duas contribuições: (*i*) um algoritmo que otimiza a alocação de recursos de armazenamento em *buffers*, em diferentes canais de roteadores, de acordo com a característica de comunicação da aplicação alvo. (*ii*) um modelo analítico usado para analisar uma dada configuração de *buffer* e detectar gargalos de desempenho nos canais dos roteadores.

No algoritmo, dado, em uma comunicação entre dois núcleos IP, um total de espaço disponível de *buffer*, a taxa de dados de chegada no *buffer* e outros parâmetros relevantes (algoritmo de roteamento, atraso de arbitragem, etc.), o algoritmo automaticamente decide a profundidade do *buffer* para cada canal de entrada em cada roteador. Esta configuração na profundidade dos *buffers* é importante no cenário do consumo de potência.

A análise do modelo analítico é feita pela resolução de um conjunto de equações não-lineares extraídas de modelos de filas. A principal vantagem dos modelos analíticos em relação à simulação de tráfegos de dados é que tais modelos fazem rápida análise do impacto dos padrões de comunicação da aplicação específica no desempenho global do sistema. A minimização de custos em área e a maximização do desempenho da NoC são os principais objetivos deste trabalho. Em resumo, este trabalho (HU 2004a) procura otimizar os recursos de armazenamento em *buffers* com o objetivo de reduzir a latência e melhorar o desempenho da rede, mas não trata do roteamento que utiliza chaveamento do tipo *wormhole*. Este tipo de chaveamento é usado pela maioria das NoC atuais. E sem considerá-lo não é possível prever como se dará o funcionamento com a carga (dados) trafegando pela rede quando esse tipo de roteamento é utilizado. Dessa forma este trabalho não é aplicável em sistemas de tempo real.

Estes mesmos autores introduziram, em outro trabalho mais recente (HU 2004b), um modelo que captura a comunicação (troca de mensagens) e a computação das tarefas, os quais são representados por um grafo de comunicação de tarefas (CTG – do inglês, *Communication Task Graph*). O algoritmo realiza o escalonamento das tarefas em cada PE e se necessário pode modificar o particionamento das tarefas e realizar migração dessas tarefas. O objetivo é minimizar o consumo de energia atendendo as restrições temporais.

Lei e Kumar (LEI 2003) apresentam um algoritmo genético para mapeamento de um grafo de tarefas na arquitetura de NoC. O objetivo principal deste trabalho é realizar o mapeamento de núcleos IP, provendo minimização do tempo de execução global da aplicação em uma NoC regular. Neste trabalho também foi desenvolvido um modelo de atraso de comunicação para uma arquitetura específica para estimar o tempo de execução. É um modelo simples, porém razoável para a realização da estimativa de execução da aplicação. O modelo não assume os parâmetros internos da NoC, como tamanho dos *buffers*, algoritmos de roteamento, etc. Esta metodologia não garante os *hard deadlines* porque ela utiliza uma métrica de atraso médio, e não o pior caso na execução de tarefas (tempo de execução no caminho crítico do grafo de tarefas).

Shin e Kim (SHIN 2004) propõem uma metodologia estática que realiza alocação do caminho de roteamento mínimo entre os núcleos IP e aloca a velocidade adequada para cada canal da NoC objetivando a redução da energia de comunicação. Esta mesma metodologia faz também a adequação da tensão em cada canal, enquanto garante as restrições de tempo real. A metodologia propõe também algoritmos de mapeamento de tarefas, para minimizar a comunicação entre os núcleos IP, enquanto é realizada a

alocação de tensão estática e gerenciamento de energia estático, que objetivam reduzir o consumo de energia da comunicação dos canais. Além disso, o algoritmo de mapeamento também procura reduzir a área do sistema como um todo. *Hard deadlines* são garantidos, porque mesmo se o escalonamento de comunicação não é executado, o pior atraso de comunicação dos canais é usado. Nos passos para a especialização e redução de energia da NoC, tais como mapeamento de tarefas e alocação do roteamento mínimo, é usado o algoritmo genético para busca heurística no espaço de soluções. Os algoritmos de escalonamento de tarefas e alocação de velocidade nos canais são baseados em *List Scheduling* (LS). LS usa a mobilidade de cada tarefa para determinar sua prioridade estática, a qual é baseada na estimativa pessimista do atraso de comunicação. No entanto, este trabalho não explora o espaço de projeto da NoC, como a configuração de seus parâmetros internos. Para atender os requisitos de tempo real, baseia-se somente na reserva de recursos.

Varatkar e Marculescu (VARATKAR 2003) apresentam uma metodologia para minimizar o consumo de energia global em uma NoC, pelas técnicas de: (*i*) mapeamento, escalonamento e (*ii*) seleção de tensão, maximizando a folga de escalonamento entre as tarefas. O primeiro passo executado, voltado para redução de comunicação entre os processadores, executa o mapeamento e o escalonamento das tarefas. No segundo, é ajustada a tensão para maximizar a folga de escalonamento (com o objetivo de diminuir a freqüência, e conseqüentemente, baixar a potência). *Hard deadlines* não são garantidos, porque o atraso de comunicação é ignorado quando é realizada a verificação dos *hard deadlines* das tarefas. Por isso, esta metodologia é apenas usada em sistemas de tempo real para tarefas *soft*. Além disso, este trabalho não trata do mapeamento de núcleos nem da exploração de projeto da arquitetura de comunicação.

A partir destas propostas de exploração de espaço de projeto que utilizam o posicionamento de núcleos como uma das etapas, pode-se observar que:

1. Kreutz et al. (KREUTZ 2005b) buscam reduzir o consumo de energia e da latência de comunicação; Marcon et al. (MARCON 2005b) procuram também a redução do consumo de energia e do tempo de execução através de um método de mapeamento de tarefas em NoCs; Murali e DeMicheli (MURALI 2004a) tentam atender as restrições de largura de banda, minimizando o atraso médio; Nollet et al. (NOLLET 2005) buscam minimizar o atraso de comunicação dinamicamente; Smit et al. (SMIT 2004) objetivam também minimizar o consumo de energia através do mapeamento de tarefas e da redução da comunicação entre os núcleos. Mas, em todos estes trabalhos, não existe suporte a tempo real, pois não são considerados os *deadlines* das comunicações.
2. Bolotin et al. (BOLOTIN 2004) propõem um fluxo de projeto para configuração da NoC atendendo às restrições temporais, porém não é apresentado nenhum algoritmo de posicionamento dos núcleos IP, somente é apontado que este algoritmo deverá ter o objetivo de otimizar a NoC minimizando consumo de energia, tráfego e área do sistema, garantindo QoS.
3. Hu e Marculescu, em seus trabalhos (HU 2003a) (HU 2003b) (HU 2004a) (HU 2004b), apresentam formas de minimizar a energia total consumida na

comunicação. Somente em (HU 2003b) e (HU 2004b) existe o suporte a tempo real, mas ainda assim, somente através da reserva de recursos baseada no WCET.

4. Lei e Kumar (LEI 2003) buscam minimizar o tempo global de execução da aplicação através de um modelo de atraso de comunicação. Ele garante os *soft deadlines*, pois usa uma métrica de atraso médio e não o WCET. Além disso, este modelo não assume os parâmetros internos da NoC, como tamanho dos *buffers*, algoritmos de roteamento, etc.
5. Os trabalhos de Shin e Kim (SHIN 2004) e de Varatkar e Marculescu (VARATKAR 2003) propõem a utilização da técnica de seleção de tensão para obter a redução do consumo de energia. Porém, não considera o mapeamento de núcleos nem a exploração de projeto da arquitetura de comunicação.

Diferentemente, este trabalho propõe uma metodologia, cujo objetivo é, através das estratégias de mapeamento e de definição das características da NoC, prover QoS para aplicações de tempo real, reduzindo o número de *deadlines* perdidos e, simultaneamente, reduzir o tempo médio de transmissão das mensagens. Com a redução dos *deadlines* pretende-se evitar ou minimizar a necessidade do aumento dos recursos (largura de banda, etc.) para garantir os requisitos de tempo real. E, na redução do tempo médio, pretende-se reduzir o consumo de energia. Na Sessão 4.3 e nos capítulos seguintes são apresentados os detalhes desta metodologia.

4.3 Metodologia Proposta na Resolução dos Problemas

Neste trabalho, propõe-se uma metodologia que tem por objetivos prover QoS para aplicações de tempo real, reduzindo o número de *deadlines* perdidos, e, ao mesmo tempo, reduzir o tempo médio de transmissão das mensagens e, assim, conseqüentemente, minimizar o consumo de energia.

Para atingir esses objetivos, essa metodologia executa as seguintes etapas: (*i*) realiza o mapeamento dos núcleos de hardware, posicionando-os na NoC; (*ii*) verifica o impacto da utilização de diferentes mecanismos da NoC, como: tipos de controle de fluxo e de arbitragem nos roteadores da NoC. A NoC considerada nessa metodologia foi a SoCIN (ZEFERINO 2003a). O fluxo de exploração de espaço de projeto seguido por esta metodologia é apresentado na Figura 4.8.

4.3.1 Especificação da Aplicação

Em virtude do objetivo de prover QoS para reduzir a perda de *deadlines* e o tempo médio de transmissão, é necessário obter, das aplicações, as informações referentes aos requisitos de tempo real. Isso implica em conhecer o volume de comunicação, de forma a definir a largura de banda necessária para os canais entre os núcleos, e as limitações temporais das mensagens trocadas entre os núcleos para definir os *deadlines* dessas mensagens. A identificação do tipo de aplicação e das características do tráfego são partes fundamentais dessa etapa inicial do fluxo de projeto de uma NoC.

No Capítulo 5 são discutidas algumas formas de especificar as aplicações de tempo real em sistemas embarcados, extraindo seus requisitos e gerando os grafos de

comunicação necessários à etapa seguinte de posicionamento. São também abordados alguns tipos de comunicação e seus impactos nos sistemas embarcados.

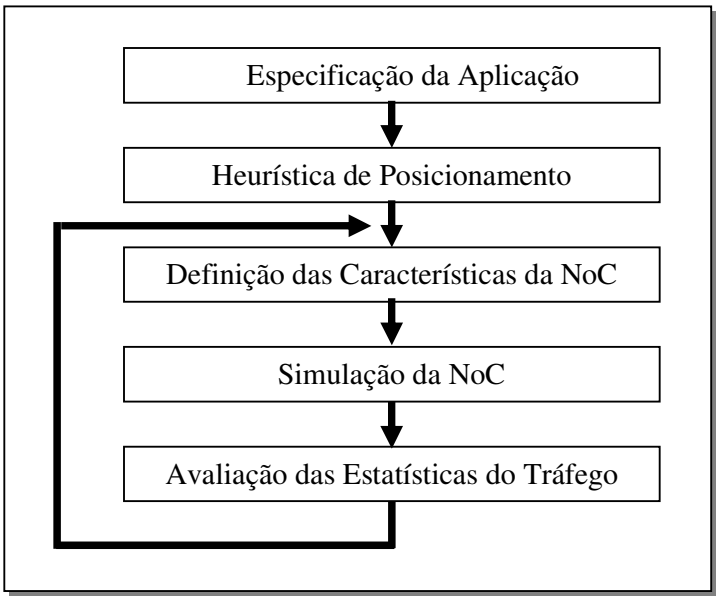

Figura 4.8: Fluxo de projeto da metodologia.

4.3.2 Posicionamento

Para realizar o mapeamento, através do algoritmo de posicionamento, é necessário obter um grafo de comunicação dos núcleos IP que servirá de entrada para a ferramenta utilizada no posicionamento. Esse grafo, gerado na etapa de especificação da aplicação, deve conter as informações referentes à largura de banda necessária para os canais de comunicação entre os núcleos IP, bem como os *deadlines* associados às mensagens que circularão nestes canais. O escalonamento não é necessário, pois é considerado um processo por núcleo.

Este grafo é um tipo semelhante ao CWM apresentado na Sub-seção 4.2.2 e ilustrado na Figura 4.7. O CWM modela uma aplicação pela sua comunicação e dependência, especificamente pela comunicação que ocorre de um núcleo para outro. Essa comunicação é dada pela soma de todos os bits de todos os pacotes transmitidos de um núcleo para outro. Assim, nos vértices do grafo estarão os núcleos IP e os valores indicados nas arestas representarão a quantidade total dos bits transmitidos, de um vértice a outro (peso de comunicação), associados aos *deadlines* de cada mensagem (peso do requisito temporal). Maiores detalhes sobre o posicionamento dos núcleos são apresentados na Seção 6.2.

4.3.3 Configuração da NoC

Na configuração da NoC são definidos parâmetros que serão simulados na etapa seguinte. Esses parâmetros envolvem características da rede, como: arbitragem, controle de fluxo e memorização. Outros parâmetros se mantiveram fixos, como: topologia em

grelha 2D, o roteamento determinístico XY e o chaveamento baseado em pacotes, do tipo *wormhole*. Os critérios para a escolha desses parâmetros que seriam invariáveis se basearam em dois fatores. Primeiro, por serem características originais da NoC utilizada neste trabalho, a SoCIN. Segundo, por já existir um grande número de pesquisas na área que apontam uma tendência de utilização dessas alternativas.

Para a ***arbitragem*** e para o ***controle de fluxo*** foram feitas variações em relação aos mecanismos utilizados pela rede SoCIN. Assim, as alternativas comparadas foram: (*i*) características originais da SoCIN, que são o controle de fluxo *handshake* e a arbitragem *Round Robin*; (*ii*) controle de fluxo *handshake* e arbitragem baseada em prioridades e, (*iii*) controle de fluxo com canais virtuais e arbitragem baseada em prioridades.

Em relação à ***memorização***, foram avaliados dois parâmetros: (*i*) profundidade das filas dos *buffers* das portas de entrada e, (*ii*) preempção dessas filas, baseada na prioridade das mensagens.

Além dessas características, avaliou-se ainda o impacto no desempenho da NoC de outras estratégias, como: (*i*) aumento, por ***envelhecimento***, da prioridade das mensagens e, (*ii*) ***descarte*** de pacotes antigos.

Maiores detalhes sobre essas estratégias e sobre a configuração dos parâmetros da NoC são apresentados no Capítulo 6.

4.3.4 Simulação e Avaliação das Estatísticas do Tráfego

Após a configuração dos parâmetros da NoC, é feita a simulação. Essa simulação é realizada para cada conjunto de parâmetros, gerando informações sobre o percentual de *deadlines* perdidos e os tempos (mínimos, médios e máximos) das comunicações. Esses resultados são então avaliados em relação aos requisitos das aplicações.

No Capítulo 7 são apresentados os detalhes sobre a simulação, incluindo detalhes sobre a ferramenta, quantidade de simulações realizadas por experimento. São apresentados também detalhes sobre os experimentos realizados e sobre a avaliação das estatísticas de tráfego.

4.4 Considerações

Este capítulo tratou do espaço de projeto, em sistemas embarcados, para aplicações de tempo real, utilizando NoCs como plataforma de comunicação. Inicialmente foi discutido o estado da arte em pesquisas de NoC. Depois, foram apresentadas propostas de fluxos de projeto de NoCs encontrados na literatura, seguidas da discussão sobre algumas das ferramentas utilizadas nesse tipo de projeto. Finalmente, foi introduzida a metodologia proposta por este trabalho para o projeto de NoCs com suporte a aplicações de tempo real.

Nos capítulos seguintes serão abordadas questões referentes aos experimentos, que visam avaliar o impacto das características da NoC nos requisitos das aplicações. Serão discutidas as etapas anteriores, necessárias para a realização dos experimentos, incluindo desde a extração dos requisitos das aplicações até a configuração dos parâmetros da NoC.

5 CARACTERÍSTICAS DAS APLICAÇÕES

Os sistemas embarcados estão cada vez mais dependentes das aplicações, fazendo com que seus projetos sejam orientados ao domínio dessas aplicações. Com isso, torna-se fundamental no desenvolvimento desse tipo de projeto o conhecimento do comportamento das aplicações. Assim, a identificação do tipo de aplicação, bem como a caracterização do tráfego são algumas das etapas iniciais e importantes, que se bem modeladas podem garantir um sistema mais adequado aos requisitos das aplicações, ao mesmo tempo em que atende às restrições de projeto.

5.1 Tipos de Aplicação e Caracterização do Tráfego

As aplicações embarcadas apresentam, em geral, restrições temporais. Além dessas restrições, outro aspecto importante na caracterização dessas aplicações é o tráfego, ou seja, a forma como os dados são transmitidos e o comportamento desse fluxo de dados entre os diversos componentes do sistema onde a aplicação irá executar.

5.1.1 Tipos de Aplicações

Quanto às restrições temporais, uma classificação existente é em relação ao atendimento dos prazos (*deadlines*), como está descrito na Seção 3.1. Essa classificação é feita em: ***hard***, que trabalha com garantia determinística, e ***soft***, que usa garantia de melhor esforço.

Outra classificação pode ser feita em relação ao fluxo dos dados: ***dataflow***, quando são orientadas a fluxo de dados e ***control flow***, quando são orientadas a controle.

5.1.2 Características do Tráfego

Além do tipo de aplicação, as características do tráfego são importantes na modelagem da NoC. Uma NoC pode acomodar diferentes tipos de tráfego. Bjerregaard e Mahadevan (BJERREGAARD 2006) classificam o tráfego em NoCs dentro de três grandes grupos: (*i*) latência crítica; (*ii*) seqüência de dados; e (*iii*) miscelânea. Eles argumentam que independente da composição do sistema, da topologia ou dos protocolos utilizados, o tráfego em uma NoC terá as características de uma dessas três categorias.

Outros autores podem usar uma classificação mais detalhada, como Bolotin et al. (BOLOTIN 2004). Entretanto, é possível dizer que essas denominações poderiam ser classificadas nas três categorias propostas por Bjerregaard (BJERREGAARD 2006).

Latência Crítica

A latência é o tempo envolvido desde o início da transmissão de uma mensagem até o momento em que ela é completamente recebida. A latência é medida em unidades de tempo ou em ciclos de relógio, no caso de um ambiente de simulação.

Em uma rede, a latência é a soma dos tempos de: (*i*) sobrecarga; (*ii*) ocupação do canal; (*iii*) atraso de roteamento e chaveamento; e (*iv*) atraso de contenção. A sobrecarga refere-se aos tempos gastos pelos nodos de origem e de destino para, respectivamente, injetar e retirar a mensagem da rede. A ocupação do canal é o tempo gasto para transferir a mensagem através dos enlaces utilizados na rota entre os nodos de origem e de destino. Os tempos gastos na determinação da rota e no envio da mensagem através do roteador representam o atraso de roteamento e chaveamento. Finalmente, o atraso de contenção é o tempo no qual a mensagem não consegue avançar devido ao congestionamento da rede.

Quando o valor dessa latência não pode sofrer variações muito severas, esse tráfego é dito de latência crítica. Em geral, nesse tipo de tráfego, as mensagens são de tamanho reduzido. Alguns exemplos de aplicações consideradas de latência crítica são as interrupções e os acessos à memória.

Seqüência de Dados

O tráfego de seqüência de dados, em geral, provém de uma aplicação orientada a fluxo de dados (*dataflow*). Esse tipo de tráfego costuma apresentar tamanhos grandes de mensagens e uma demanda de QoS em relação à largura de banda. Algumas aplicações multimídia, por exemplo, necessitam de definições precisas de taxa de transmissão de dados e de atraso máximo da rede.

Outro aspecto temporal que geralmente é crucial para esse tipo de tráfego é o *jitter*. Ou seja, é necessário que a variação no atraso de envio dos pacotes de cada mensagem esteja dentro de um limite pré-determinado. Exemplos do tráfego em seqüência de dados são: o acesso direto à memória (DMA – do inglês *Direct Memory Access*); e aplicações do tipo MJPEG (*Motion JPEG, onde: JPEG - Joint Photographic Experts Group*).

Miscelânea

O tráfego será do tipo miscelânea quando não existirem requisitos específicos de compromisso por parte da rede. Um exemplo desse tipo são as transferências de blocos de dados.

5.2 Extração dos Requisitos das Aplicações

Para realizar a etapa de mapeamento, através do algoritmo de posicionamento, é necessário obter um grafo de comunicação dos núcleos IP que servirá de entrada para a ferramenta utilizada no posicionamento. Esse grafo deve conter as informações referentes à largura de banda necessária para os canais de comunicação entre os núcleos IP, bem como os *deadlines* associados às mensagens que circularão nestes canais.

Este grafo é um tipo semelhante ao CWM descrito na Sub-seção 4.2.1 e ilustrado na Figura 4.7. O CWM modela uma aplicação pela soma de todos os bits de todos os pacotes transmitidos de um núcleo para outro. Assim, nos vértices do grafo estarão os núcleos IP e os valores indicados nas arestas representarão a quantidade total de bits transmitidos de um vértice a outro (peso de comunicação), associados aos *deadlines* de cada mensagem (peso do requisito temporal). Sendo *N*, o número de núcleos da NoC, o tamanho máximo do grafo é *N**(*N*-1), que ocorre quando todos os núcleos (ou vértices) enviam e recebem mensagens de todos os demais.

Em sistemas embarcados, baseados em NoCs, existe uma dificuldade em se encontrar dados de bons exemplos de aplicações reais. A maioria dos experimentos nesta área é feita com aplicações sintéticas. Assim, uma alternativa para se obter grafos de aplicações sintéticas adequados para a metodologia proposta é através da ferramenta para geração automática de grafos de tarefas (TGFF – do inglês, *Task Graphs for Free*) (DICK 1998). Neste caso, a etapa de especificação da aplicação ilustrada na Figura 4.7 seria uma geração do grafo através do TGFF. Os dados gerados pelo TGFF se referem ao volume de tráfego, ou seja, à largura de banda necessária. Os deadlines são atribuídos com base no tipo de aplicação considerada.

Neste trabalho foram usadas ambas as alternativas. Uma aplicação real de MJPEG tem um comportamento corresponde a uma aplicação orientada a fluxo de dados e cujo tráfego é em seqüência de dados. E uma aplicação sintética gerada através do TGFF, com comportamento e tráfego heterogêneo. Maiores detalhes sobre essas aplicações, utilizadas nos experimentos, são apresentadas na seqüência desse texto.

5.2.1 MJPEG

No intuito de verificar a metodologia também com aplicações não sintéticas, utilizamos uma aplicação multi-tarefas de descompressão de um fluxo de imagens MJPEG (*Motion* JPEG) desenvolvida no Departamento de *Architecture des Systèmes Intégrés & Microélectronique* (ASIM) do Laboratório de Informática (LIP6) da Universidade Pierre et Marie Curie (*Université Paris 6*) que foi objeto do estágio de doutorado realizado entre novembro de 2004 e agosto de 2005.

O objetivo deste estágio foi portar a aplicação MJPEG, já existente, de forma que esta aplicação utilizasse uma biblioteca de funções de comunicação MWMR (*Multi Writer Multi Reader*). Essa biblioteca de comunicação, construída sobre uma API POSIX, permite a execução de uma aplicação em software, seja em uma estação de trabalho UNIX, seja sobre uma arquitetura multiprocessada integrada em um chip, controlada pelo sistema operacional MUTEK. MUTEK é um sistema operacional de tempo real embarcado, multiprocessado, multi-threads, compatível com POSIX, desenvolvido pelo LIP6. A primeira parte do trabalho consistiu da substituição dos canais de comunicação originais pelos canais que utilizavam MWMR em todas as tarefas da aplicação, ainda no domínio Linux/POSIX. A segunda parte consistiu em portar essa aplicação sobre uma arquitetura multiprocessada em SoC (MP-SoC) construída com a plataforma SoCLib (SOCLIB 2007), substituindo duas das tarefas em software por componentes (tarefas) em *hardware*.

Um fluxo MJPEG pode ser visto como um conjunto de imagens JPEG (*Joint Photographic Experts Group*). Quando da codificação das imagens, estas foram divididas em blocos de 64 *pixels*, como mostra a Figura 5.1(a). A seguir, a imagem é

tratada em relação aos blocos e para cada bloco é calculada a transformada discreta do cosseno. E, em todos os blocos, cada uma das 64 freqüências é dividida por um coeficiente de quantificação. As freqüências resultantes de maior valor são eliminadas, sendo mantidas aquelas de valores menores. Após isso, é feito um reordenamento zigue-zague do bloco, conforme ilustrado na Figura 5.1(b). Este reordenamento permite simplificar a representação por agrupar as freqüências de valores equivalentes, facilitando a eliminação daquelas desnecessárias. Nessa reordenação é formada a tabela de Huffman, que possui os coeficientes referentes a cada bloco.

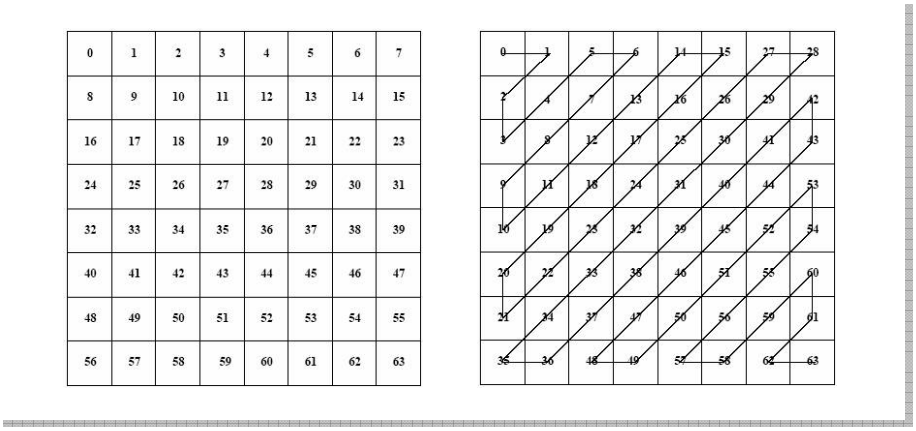

Figura 5.1: Estrutura de um bloco JPEG: a) ordem natural; b) ordem zigue-zague.

A decodificação é composta de diversas etapas (conforme mostra a Figura 5.2): TG, DEMUX, VLD, IQZZ, IDCT e RAMDAC. Os valores numéricos representam a quantidade de bytes que é transferida entre cada uma das etapas.

O início da decodificação é através da leitura das imagens codificadas. Elas são representadas por um gerador de tráfego (TG – do inglês, *Traffic Generator*). Após a leitura dos arquivos das imagens, o TG envia as imagens JPEG através de uma interface FIFO para o demultiplexador. O demultiplexador (DEMUX) analisa o fluxo recebido e realiza a demultiplexação das informações: o tamanho da imagem; as tabelas de Huffman e as tabelas de quantificação. Em seguida, os dados são enviados para a etapa seguinte. Nesta etapa, VLD (*Variable Length Decoder*), é efetuada a decodificação de Huffman e os dados são transferidos à próxima etapa. A tarefa IQZZ realiza a quantificação inversa (IQ – do inglês, *Invert Quantification*) e o reordenamento zigue-zague (ZZ). Ela recebe uma tabela com os fatores multiplicadores e fornece como saída a imagem na ordem natural (Figura 5.1(a)). O IDCT (*Inverse Discrete Cosine Transform*) realiza a transformada inversa do cosseno com os dados recebidos do IQZZ. Na seqüência, os dados são enviados para o LIBU (*Line Builder*) que efetua um re-seqüenciamento dos *pixels*. Ele recebe os *pixels* organizados em blocos e os reorganiza em linhas. Em seguida, os *pixels* ordenados em linhas são enviados para o RAMDAC, que realiza a visualização das imagens a cada 40 ms.

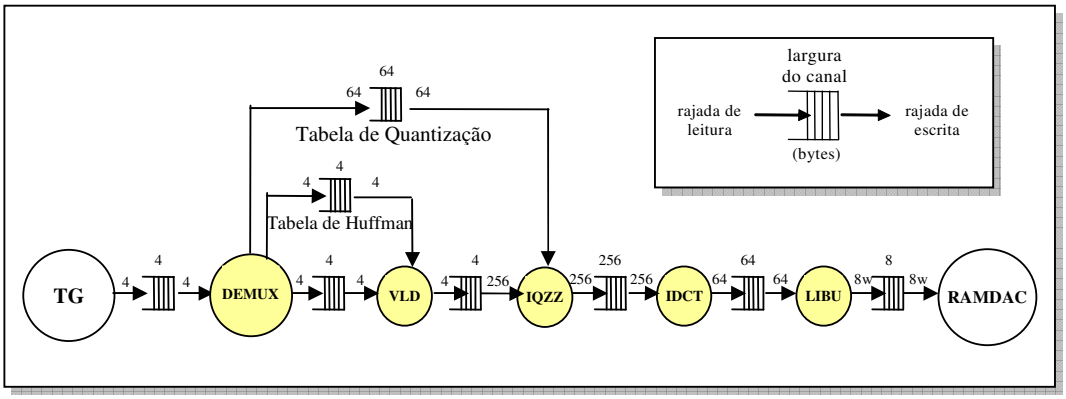

Figura 5.2: Grafo da aplicação MJPEG.

Apesar desta aplicação ter sido executada numa plataforma MP-SoC baseada em barramentos, ela pode ser facilmente portada para uma NoC, pois cada tarefa é independente das demais, o que permite a execução aproveitando o máximo de paralelismo. No contexto deste trabalho o que é necessário é extrair o comportamento da aplicação MJPEG, especificamente no que diz respeito à comunicação entre as tarefas e aos requisitos temporais.

Esta aplicação MJPEG é orientada a fluxo de dados e possui um tráfego predominantemente em seqüência de dados. A exceção fica por conta da tarefa RAMDAC que a cada 40ms faz a leitura dos dados no buffer de entrada independente se todos os dados já se encontram disponíveis, ou não. E após isso, envia um sinal para as demais tarefas descartarem os dados da imagem atual e começarem a tratar os dados da próxima. O RAMDAC conseguirá exibir a imagem correta se todos os dados já estiverem disponíveis. Caso contrário, o que se visualizará será uma imagem distorcida ou totalmente diferente da original.

5.2.2 Geração de Tráfego Heterogêneo

Visto que em uma NoC podem existir os mais diversos tipos de comunicação, desde acesso à memória, envio de seqüência de dados, até interrupções, era necessário realizar experimentos que contemplassem todas essas situações. Com este objetivo, de se ter um tráfego mais heterogêneo, utilizou-se uma ferramenta de geração automática de grafos de tarefas, o TGFF (DICK 1998), para gerar grafos de uma aplicação sintética.

Os tipos de comunicação utilizados, apresentados na Tabela 5.1, seguiram a mesma categorização utilizada por Bolotin et al. (BOLOTIN 2004), onde o tráfego possui quatro níveis de serviço: (*i*) sinalização; (*ii*) tempo real; (*iii*) leitura/escrita e (*iv*) transferência de blocos de dados.

O tráfego de ***sinalização*** é o que possui maior prioridade. Esse tipo de tráfego envolve mensagens de tamanho reduzido como, por exemplo, interrupções e os acessos à memória. As sinalizações são do tipo críticas em latência.

O tráfego de ***tempo real*** apresenta mensagens um pouco maiores e necessita de uma demanda maior de QoS em relação à largura de banda. Esse tráfego é do tipo seqüência de dados, como DMA e os dados de uma aplicação MJPEG.

Já os demais tráfegos, de ***leitura/escrita*** e de ***transferência de blocos de dados***, não apresentam requisitos específicos em relação à rede, exceto uma limitação temporal (*deadline*). A diferenciação entre os dois se dá, basicamente, pela quantidade de dados envolvida.

Na Tabela 5.1, estão descritas as características de cada classe de serviço: o *deadline*, o tamanho médio dos pacotes, a carga total e a prioridade associada. O valor dos *deadlines* está em nanosegundos, considerando que um ciclo de simulação tem a duração de 1ns. O valor do tamanho médio dos pacotes está em *flits*. A carga total de cada nível está expressa em valores relativos. As cargas dos níveis de leitura/escrita e de transferência de bloco de dados representam um volume oito vezes superior às cargas dos níveis de sinalização e de tempo real. A escala de prioridade cresce da classe de transferência de blocos de dados, que possui a menor prioridade, até a classe de sinalização, cuja prioridade é a maior.

Tabela 5.1: Classes de serviço da aplicação sintética

Nível do Serviço	**Deadline (ns)**	**Tamanho Médio dos Pacotes (*flits*)**	**Carga Total**	**Prioridade**
sinalização	20	2	1x	4
tempo real	500	4	1x	3
leitura / escrita	150	40	8x	2
transferência de blocos de dados	1000	1000	8x	1

5.3 Considerações

Neste capítulo abordaram-se as características das aplicações em NoC. Discutiu-se sobre a identificação dos tipos de comunicação e sobre a caracterização do tráfego dessas aplicações. Essas duas atividades são partes fundamentais da etapa inicial do fluxo de projeto de uma NoC – a especificação da aplicação. A partir dessas informações, o projeto tem mais chances de se aproximar do melhor resultado, que atende os requisitos das aplicações e ao mesmo tempo as restrições da tecnologia.

Nos capítulos seguintes serão mostradas as estratégias utilizadas na configuração da NoC e discutidos os critérios utilizados para a seleção dos parâmetros considerados dentre os que caracterizam uma NoC (posicionamento, arbitragem, memorização, etc.). Na seqüência são apresentados os experimentos realizados e discutidos os seus resultados.

6 CONFIGURAÇÃO DA NOC

No projeto de sistemas embarcados, o atendimento aos requisitos das aplicações tem de ser atingido dentro dos limites da tecnologia. Quando esses sistemas utilizam NoCs como plataforma de comunicação é necessário também que o projeto da NoC respeite todas as restrições de projeto, sejam elas de: área, potência, temperatura, energia, atendimento de restrições temporais, tempo de lançamento do produto no mercado (*time to market*), ou outras. Em alguns casos, para atender determinadas restrições de projeto, a NoC necessita ser específica para uma dada aplicação.

A configuração da NoC consiste de duas etapas: a definição do posicionamento dos núcleos na rede e a configuração das demais características da rede, tais como: topologia, arbitragem, memorização, mecanismos de controle de fluxo, etc.

6.1 Parâmetros Considerados

Um problema inicial na definição das características da NoC é selecionar quais serão avaliadas, isto é, serão variáveis, e quais serão fixas, ou seja, não serão parametrizáveis. Por exemplo, para definir o posicionamento dos núcleos IPs é necessário saber como será a topologia. Mas, a questão aqui é primeiramente saber se serão avaliadas mais de uma topologia ou se as avaliações de posicionamento considerarão somente uma determinada topologia. E sendo assim, esta característica (topologia) não seria parametrizável, seria fixa.

Dentre as características, algumas sofreram variação durante os experimentos, outras foram consideradas como não parametrizáveis, ou seja, se mantiveram iguais em todos os experimentos. Estas últimas foram: a ***topologia***, em grelha 2D; o ***roteamento***, determinístico XY; o ***chaveamento***, baseado em pacotes e do tipo *wormhole*. Tais decisões se basearam especificamente em dois fatores. Primeiramente, por serem essas as características originais da NoC considerada neste trabalho, a SoCIN. O outro fator é por já existirem extensas pesquisas nesses tópicos, ou ainda, pelo fato de serem uma característica comum à maioria das NoCs na literatura, mostrando uma tendência a utilização dessas alternativas. Outros fatores, como: arbitragem, memorização, mecanismos de controle de fluxo, além do posicionamento dos núcleos foram avaliados e são discutidos mais detalhadamente na seqüência desse texto.

6.2 Posicionamento dos Núcleos

Uma vez definido o grafo da aplicação a partir de suas especificações ou através do TGFF (no caso das aplicações sintéticas) é possível seguir para a etapa seguinte da metodologia, que é o mapeamento dos núcleos IP, realizado através do algoritmo de posicionamento.

No mapeamento, são comparadas três alternativas de posicionamento dos núcleos IP: (*i*) randômico, onde o algoritmo de posicionamento não é acionado; (*ii*) baseado somente na largura de banda, onde o algoritmo considera as exigências da largura de banda das mensagens dos canais de comunicação entre os núcleos e; (*iii*) baseado na largura de banda e nas restrições temporais, quando o algoritmo considera, além das exigências das larguras de banda das mensagens, as prioridades associadas a essas mensagens. Essas prioridades são relacionadas aos requisitos temporais (*deadlines* das mensagens).

No caso (*ii*) o valor do peso associado a cada aresta do grafo considera a largura de banda das mensagens. No caso (*iii*), além da largura de banda é considerado também o deadline das mensagens. Neste caso o valor do peso associado a cada aresta é igual ao produto do valor da largura de banda com o valor da prioridade associada às mensagens do canal em questão.

O objetivo do mapeamento é encontrar uma solução ótima para uma dada função. Neste caso, posicionar próximos os núcleos que apresentam altas taxas de comunicação. Para isto, existem diferentes tipos de abordagens que usam diversas funções objetivo para otimizar um dado sistema que utiliza NoC. O algoritmo de posicionamento utilizado é do tipo *Simulated Annealing* e é descrito em detalhes na Sub-seção 6.2.1.

6.2.1 Algoritmo de Posicionamento

Dada uma aplicação distribuída, é necessário determinar o melhor posicionamento de núcleos IP em uma topologia de rede, com o objetivo de atender às restrições de tempo real da aplicação, ao mesmo tempo em que reduz o consumo global de energia da rede. Em outras palavras, a posição dos núcleos de acordo com a topologia da rede pode ser definida pelo desempenho de comunicação da aplicação.

Para modelar e resolver o problema de mapeamento foi adotado o Modelo de Comunicação com Pesos (CWM – do inglês *Communication Weighted Model*), apresentado em (MARCON 2005a). Foi definida uma caracterização da comunicação da aplicação (ACC – do inglês, *Application Communication Characterization*) que representa como ocorre a comunicação entre os núcleos. Isso pode ser descrito pelas relações entre as tarefas da aplicação, pelo paralelismo entre as mensagens trocadas entre os núcleos e pela largura de banda e prioridade requerida para cada mensagem, conforme mostra a Figura 6.1(a). Esse relacionamento pode ser modelado como um grafo orientado, onde os vértices representam as tarefas da aplicação (executando nos núcleos IP) e os arcos expressam o relacionamento entre essas tarefas. Além disso, pode-se dizer que os vértices representam os emissores e os destinatários das mensagens e os arcos representam o caminho para as mensagens. O conjunto de arcos expressa, assim, o padrão de comunicação da aplicação. Os arcos possuem pesos e duas alternativas de peso foram exploradas. Na primeira, o peso, representado pelo arco, corresponde somente à largura de banda requerida pelo conjunto de mensagens trocadas

entre os núcleos representados pelos vértices. Na segunda exploração, a prioridade da mensagem é também associada ao arco. Dessa forma o peso é dado pelo produto entre a prioridade e a largura de banda.

Neste trabalho é considerado o caso de paralelismo máximo da rede, ou seja, cada núcleo IP possui apenas uma tarefa, não sendo necessário existir o escalonamento local em cada um desses núcleos.

Definição 6.1 – O ACC $G = (V,A)$ é um grafo direto, onde cada vértice $v_i \in V$ (i = 1,2,…,m) representa uma tarefa da aplicação que envia e/ou recebe mensagens e cada arco $a_j \in A$ (j = 1,2,…,n) é o caminho direto entre o vértice emissor e o vértice receptor para uma dada mensagem. Para cada arco direto $a_j = (u,v)$, um peso define a largura de banda $b(a_j)$ requerida pela aplicação para a mensagem representada por a_j, ou o produto largura de banda * prioridade. Uma posição $p(v_i)$ é definida para cada vértice v_i.

A arquitetura de comunicação também pode ser modelada como um grafo, cujos vértices representam os roteadores (ou seja, os núcleos de comunicação) na NoC e o conjunto orientado de arcos expressa os canais de comunicação dados pela topologia de rede. Esta estrutura de dados é definida como ACG (*Architecture Communication Graph*), ilustrada na Figura 6.1(b).

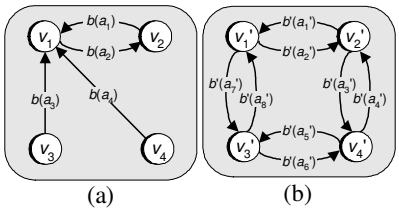

Figura 6.1: Exemplos de: (a) ACC; (b) ACG.

Definição 6.2 – O ACG é um grafo direto, $G' = (V',A')$, onde cada vértice $v_q' \in V'$ (q = 1,2,…,m) representa um roteador na arquitetura de comunicação e cada arco $a_r' \in A'$ (r = 1,2,…,l) representa um canal entre dois roteadores diretamente conectados em uma determinada topologia de rede. Além disso, para redes diretas, cada roteador possui uma porta local, onde o núcleo IP é conectado. Para cada arco direto $a_r' = (u',v')$, um peso expressa a largura de banda disponível $b'(a_r')$ do respectivo canal de comunicação. Este parâmetro é obtido das características físicas da rede como largura do canal e freqüência. A forma como os arcos são conectados representa a topologia da rede.

Para obter um posicionamento, é necessário mapear cada tarefa da aplicação (vértice do ACC) para uma porta local associada a um roteador (vértice do ACG).

Definição 6.3 – Dado, um ACC e um ACG, para cada vértice $v_i \in G$ em ACC existe um vértice correspondente $v_q' \in G'$ em ACG, e vice-versa, ou seja, existe uma função de mapeamento bijetora $F: V \rightarrow V'$ s.t. $\forall v_i \in V, \exists v_q' \in V' (v_q' = F(v_i) \Lambda v_i = F^{-1}(v_q') \Lambda p(v_i) = p(v_q'))$.

Finalmente, para cada mensagem da aplicação, é necessário encontrar no ACG um caminho entre o vértice emissor e o vértice receptor, de modo a determinar se a largura de banda oferecida pelo caminho atende àquela requerida pela aplicação.

Definição 6.4 – Um caminho $C = (v_1', a_1', v_2', a_2', ..., v_{m-1}', a_{m-1}', v_m')$ em ACG é uma seqüência alternativa de vértices do emissor até o receptor da mensagem. Um caminho é formado de acordo com a estratégia de roteamento implementada nos roteadores da rede representados no ACG.

Usando essa representação de grafo, o problema pode ser formulado como de cobertura de linhas de uma matriz $M = (a_{ij})$, de m linhas, n colunas, por um sub-conjunto j; $j = 1,\ldots,n$; de colunas $M_j = (a_{ij})$; $i = 1,\ldots,m$; a um custo mínimo. Se definirmos, $x_j = 1$, quando a coluna j (com custo $c_j > 0$) está na solução e, $x_j = 0$, caso contrário, então o problema pode ser formulado como:

$$Minimize \quad Z = \sum_{j=1}^{n} c_j x_j \tag{6.1}$$

$$em \quad relação \quad a: \quad \sum a_{ij} x_j = 1 \quad i = 1,...,m \tag{6.2}$$

$$x_j \in \{0,1\} \quad j = 1,...,n \tag{6.3}$$

A equação (6.2) assegura que cada linha é coberta por uma coluna e, por sua vez, a declaração (6.3) é a restrição de integridade. É possível, então, formular esse problema de posicionamento como um problema de mapeamento: *M* é o conjunto de vértices no ACG usado para cada mensagem da aplicação, $G' = (V', A')$; $M \subseteq V'$. Isso ocorre porque todas as *n* mensagens de uma aplicação cobrem um conjunto *M* de roteadores da NoC quando elas são enviadas na rede através de seus caminhos *C*.

Definição 6.5 – Um conjunto $M_j = (a_{ij} = 1 \mid i \in M)$ é um caminho C_j para uma das *n* mensagens na aplicação: $M_j = (C_j \mid v_q' \in C_j. \; a_{ij} = 1 \; \Lambda \; i = p(v_q'))$.

Na formulação desse problema, c_j expressa a largura de banda (ou o produto entre esta e a prioridade) que a mensagem M_j pode requerer. A semântica adotada para c_j na função objetivo diz que se $c_j = 0$, a sua largura de banda associada é igual ou maior que a largura de banda requerida pela aplicação; por outro lado, se $c_j = 1$, a largura de banda requerida não será alcançada pelo posicionamento atual.

Definição 6.6 – Admitindo que $b'(a_r')$ seja a largura de banda para um determinado arco a_r' em um caminho C_j, e que *B* seja a largura de banda requerida para a mensagem j:

$$c_j = 0, \quad se \quad MIN b'(a_w') \leq B \quad (w = 1 \text{ até } m-1) \tag{6.4}$$

$$c_j = 1, \quad caso \quad contrário \tag{6.5}$$

Como conseqüência, a minimização da função objetivo, como modelada aqui, significa que seus valores serão os mais próximos possíveis do zero. Assim, quando Z é zero, todas as larguras de banda são alcançadas.

É comprovado que esse tipo de problema de mapeamento é do tipo NP-completo (GAREY 1979) e existe nos últimos anos na literatura a proposição de diversos

algoritmos de otimização ou baseados em heurística. A escolha do algoritmo, para realizar o mapeamento na metodologia proposta, foi pelo *Simulated Annealing* (KIRKPATRICK 1983). Essa escolha se deve ao fato de já existir disponível no Grupo de Microeletrônica (GME) da UFRGS uma ferramenta desenvolvida (DRAGON LEMON 2007) baseada neste algoritmo.

Simulated Annealing é uma generalização do método de Monte Carlo para o exame de equações de estado e estados térmicos de um sistema de *n* integrantes. O conceito é baseado na maneira pela qual os líquidos ou os metais recristalizam no processo de aquecimento (*annealing*). No processo de aquecimento, é feito, inicialmente, um derretimento desordenado a uma alta temperatura *T*. Depois é esfriado lentamente de modo que o sistema em algum ponto se aproxime do equilíbrio termodinâmico. Como o procedimento de esfriamento prossegue, o sistema começa a ficar mais ordenado e se aproxima do estado de "congelamento" com T=0.

Simulated Annealing tem sido usado em diversos problemas de otimização combinacional e tem sido particularmente bem sucedido nos problemas de projeto de circuitos (KIRKPATRICK 1983).

6.2.2 Ferramenta de Posicionamento

A ferramenta Dragon Lemon (DRAGON LEMON 2007) é um ambiente desenvolvido em C++ no GME da UFRGS que permite o posicionamento de núcleos em uma NoC a partir do grafo de comunicação entre esses núcleos. O arquivo de entrada segue o formato mostrado na Figura 6.2, cujas informações serão descritas em mais detalhes na Sub-seção 7.1.2. Essa ferramenta utiliza algoritmos do tipo *Simulated Annealing* para realizar o posicionamento e oferece uma interface gráfica, como ilustra a Figura 6.3. Através dessa interface gráfica é possível configurar, entre outros parâmetros: o tipo de topologia da rede; a temperatura inicial; o número de iterações e de repetições que serão executadas; quantos resultados serão considerados e salvos.

```
net_size 4 4
nodes A B C D E F G H I J K L M N O P

* topology
A – B     160   1000  1
A – D     20    20    4
B – E     20    500   3
C – P     20    20    4
D – M     160   1000  1
D – O     20    20    4
E – L     20    500   3
F – O     160   150   2
G – I     20    20    4
H – J     160   150   2
...       …     …     …
P – L     160   150   2
```

Figura 6.2: Exemplo de arquivo de entrada da ferramenta de posicionamento.

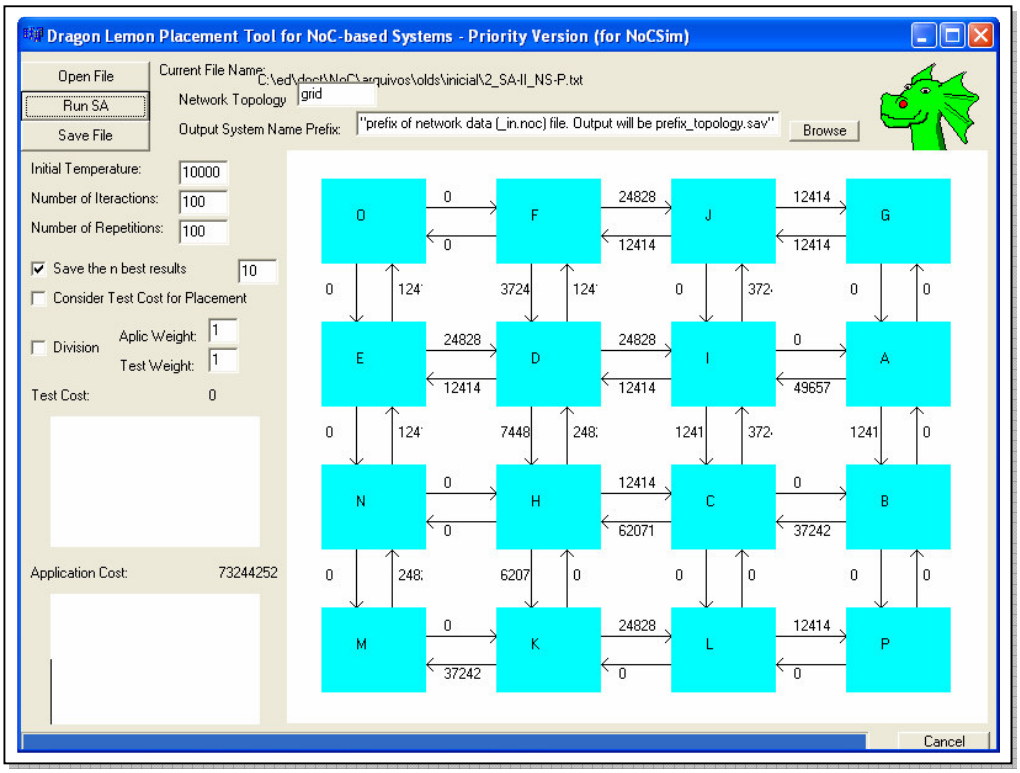

Figura 6.3: Interface gráfica da ferramenta de posicionamento.

```
net_size 4 4
nodes A B C D E F G H I J K L M N O P

* topology
A – B     160   1000  1
A – D     20    20    4
B – E     20    500   3
C – P     20    20    4
D – M     160   1000  1
D – O     20    20    4
E – L     20    500   3
F – O     160   150   2
G – I     20    20    4
H – J     160   150   2
...       …     …     …
P – L     160   150   2

# placement
D   K   G   C
O   M   J   N
E   L   P   A
F   I   B   H
```

Figura 6.4: Exemplo de arquivo de saída da ferramenta de posicionamento.

Pequenas modificações foram realizadas na versão original do Dragon Lemon para que o formato dos arquivos de saída tivesse o mesmo formato utilizado na entrada do simulador, cujo exemplo é ilustrado na Figura 6.4 e maiores detalhes sobre as informações desse arquivo serão vistos na Sub-seção 7.1.2. O simulador é o responsável pela avaliação das demais características da NoC e será detalhado no Capítulo 7.

As demais características da NoC, como arbitragem, mecanismos de controle de fluxo, memorização, além de outras estratégias propostas, foram avaliadas através de simulação. Nas próximas seções serão detalhadas as alternativas utilizadas, nessa avaliação, para cada uma dessas características.

6.3 Arbitragem

A rede SoCIN utiliza um esquema de arbitragem distribuída, onde cada porta de saída de um roteador possui um árbitro. Este árbitro aplica o critério de prioridades corrente para selecionar uma das requisições, escolhendo o *buffer* de uma das portas de entrada do roteador para ser conectado ao canal de saída, quando este canal de saída estiver livre.

Originalmente, a SoCIN possui árbitros que utilizam critérios baseados em prioridades dinâmicas rotativas, o Round Robin. Neste tipo, o canal de entrada selecionado na arbitragem corrente terá o menor nível de prioridade na arbitragem seguinte. Esse tipo de árbitro evita o problema de postergação indefinida (*starvation*), mas não evita o bloqueio de cabeça de linha (HoL) que pode ocorrer devido ao chaveamento por pacotes do tipo *wormhole*.

Como alternativa ao Round Robin, foi avaliada a utilização de árbitros baseados na prioridade das mensagens. Essa prioridade é associada a cada tipo de mensagem em tempo de projeto. A atribuição do nível dessa prioridade pode levar em conta critérios como menor deadline, menor período, etc. Essa informação está contida no cabeçalho dos pacotes de todas as mensagens.

6.3.1 Comparativo de Custos

A arbitragem rotativa (Round Robin) necessita de um conjunto de registradores para armazenar a informação referente à prioridade rotativa. Para cada porta é associada uma prioridade, que a cada arbitragem é alterada.

Na arbitragem baseada na prioridade das mensagens é necessária uma lógica adicional para verificar dentre as portas que desejam enviar, qual a que possui a mensagem de maior prioridade. E no caso de existir mais de uma porta com mensagens de mesma prioridade, o árbitro segue a prioridade rotativa (Round Robin) para selecionar entre essas portas.

A arbitragem baseada na prioridade das mensagens necessita de uma lógica adicional se comparada com a arbitragem Round Robin. Isso implica em maior área e consumo de potência. Entretanto com a primeira é possível fornecer QoS baseada em diferenciação de classes de serviço.

6.4 Controle de Fluxo

O controle de fluxo realiza a regulação de tráfego nos canais e é implantado em nível de enlace. Na rede SoCIN, o controle de fluxo é do tipo *handshake*, no qual o emissor informa a intenção de enviar um dado ao receptor através de uma linha de validação e o receptor confirma a disponibilidade de espaço em buffer para receber esse dado através de uma linha de reconhecimento (*acknowledgement*). O *handshake* é derivado do protocolo FIFO, associado aos *buffers* de entrada. Dessa forma, os dados são atendidos na ordem de chegada, e em caso de bloqueio, por ter atingido o limite do receptor, pode ocorrer o bloqueio de cabeça de linha (HoL).

Uma alternativa para evitar o HoL é realizar o controle de fluxo através de canais virtuais. Esse método consiste em dividir o *buffer* de entrada em filas independentes de profundidades menores que irão formar os canais virtuais. Assim, o uso de canais virtuais permite uma maior utilização do canal físico, que poderá utilizar outro canal virtual em caso de bloqueio.

Então, além do mecanismo original de controle de fluxo do tipo *handshake*, foi avaliada também a alternativa com canais virtuais. E, na Seção 6.5, serão discutidas também as alternativas em relação à memorização.

6.4.1 Comparativo de Custos

A utilização de canais virtuais para o controle de fluxo pode evitar bloqueios e congestão de tráfego devido a mensagens de menor prioridade. Isso ocorre devido ao isolamento de tráfego entre os canais virtuais. Assim, mesmo que uma mensagem menos prioritária fique bloqueada, outra de maior prioridade poderá ser enviada através de outro canal virtual.

Outra vantagem dos canais virtuais é que eles podem prover a diferenciação de serviços para aplicações que necessitam de QoS, como apresentam alguns autores (KUMAR 2002)(BOLOTIN 2004)(MORAES 2004). Entretanto, o uso de canais virtuais implica em um consumo maior de área e potência devido ao custo do controle e das filas nos *buffers* (BJERREGAARD 2006).

Uma análise mais detalha, entre o custo de utilização de canais virtuais e o custo de utilização de um único *buffer* com preempção, é apresentada na Sub-seção 6.5.1.

6.5 Memorização

A memorização, na rede SoCIN, é feita através de *buffers* na entrada dos roteadores. A utilização de chaveamento por pacotes, do tipo *wormhole*, permite uma maior economia no tamanho desses *buffers* uma vez que não é necessário armazenar todo o pacote, mas apenas *flits* (pelo menos um). Mas essa profundidade mínima também tem relação direta com o tipo de aplicação, pois, dependendo do tamanho dos pacotes que essa aplicação gera, um tamanho maior dos *buffers* pode permitir uma menor contenção na rede. Alguns autores (BOLOTIN 2004) (BJERREGAARD 2006) mostram que esse aumento na profundidade dos *buffers* não é uma solução para evitar a contenção, pois o benefício no desempenho é inferior ao acréscimo na área e no consumo de potência. Entretanto, um aumento na profundidade dos *buffers* pode ser útil para absorver tráfego em rajadas (BJERREGAARD 2006). Assim, a profundidade dos *buffers* também foi considerada como parâmetro a ser avaliado.

A memorização nos *buffers* utiliza o protocolo FIFO, onde os dados são armazenados e ordenados pela ordem de chegada. Ou seja, serão transmitidos para o próximo roteador os dados que chegaram primeiro no *buffer* do roteador atual. Nesse caso, e quando não são utilizados canais virtuais no controle de fluxo, o protocolo FIFO pode ocasionar bloqueio de mensagens mais prioritárias por outras de menor prioridade. Então, uma alternativa avaliada para o caso de buffer único, sem canais virtuais, foi o ordenamento desse *buffer*, através da preempção, de acordo com a prioridade da mensagem, que é indicada no cabeçalho dos pacotes.

6.5.1 Comparativo de Custos

A variação na profundidade do buffer foi avaliada através de simulação, onde foram medidos a latência média das comunicações e o percentual de *deadlines* perdidos para cada tamanho de *buffer* avaliado.

Um exemplo de implementação de canais virtuais em um roteador semelhante ao da rede SoCIN é mostrado na Figura 6.5, onde cada uma das quatro portas direcionais (que são conectadas a outros roteadores) tem canais virtuais. A porta local, que conecta o roteador ao núcleo tem um *buffer* único, sem canais virtuais. A relação entre a área (em número de multiplexadores) e o número de canais virtuais é linear, como mostra a equação (6.6).

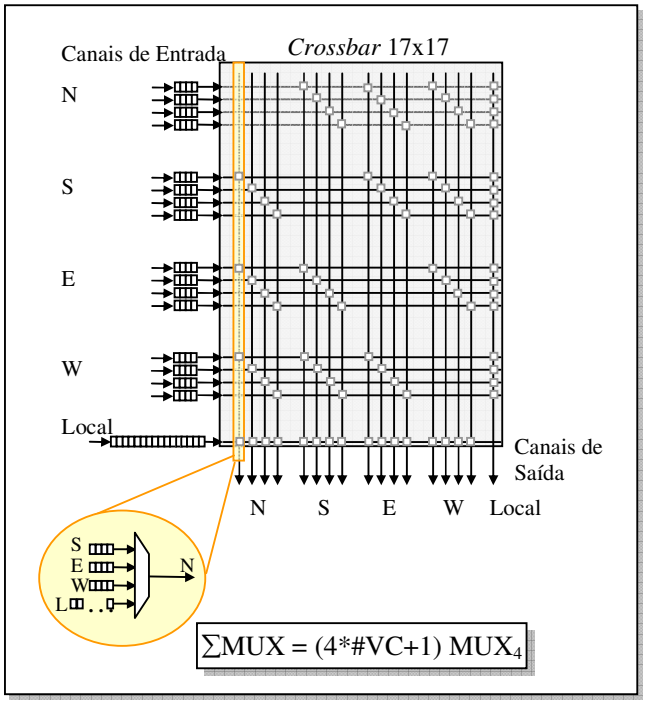

Figura 6.5: Exemplo de roteador com canais virtuais.

$$Area_{VC} = (4*\#\,\mathrm{VC}+1)*\mathrm{MUX}_4 \qquad (6.6)$$

Para o caso onde os *buffers* de entrada do roteador não possuem canais virtuais, é necessário usar um mecanismo de preempção como o ilustrado na Figura 6.6. Então, o

custo de área, mostrado na equação (6.10), é a soma da área do *crossbar* e da área do mecanismo de preempção. O *crossbar* é menos complexo e tem uma área menor que o caso onde se usa canais virtuais, de acordo com o que está expresso na equação (6.7). Entretanto, a área do mecanismo de preempção é dependente da profundidade do *buffer*, como mostrado na equação (6.8).

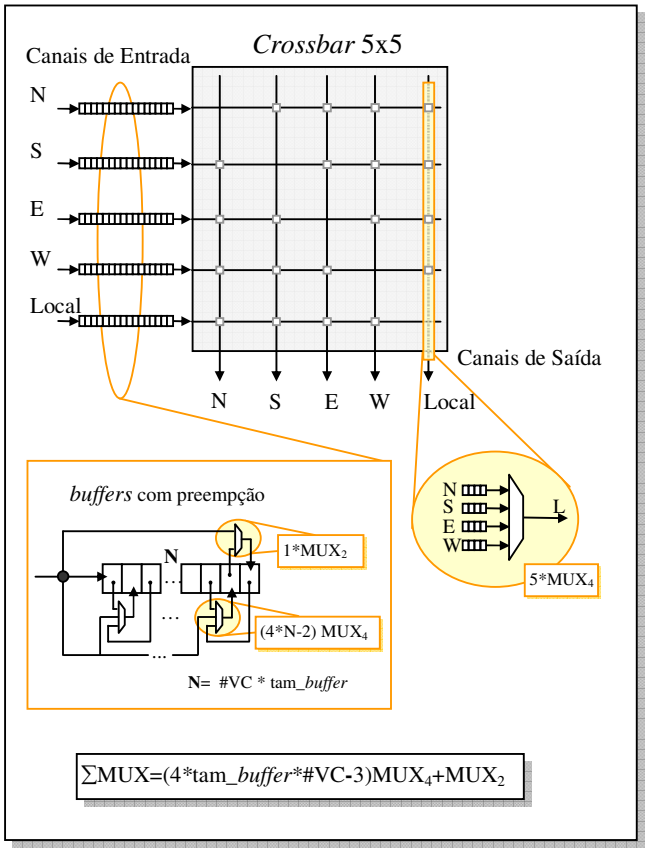

Figura 6.6: Exemplo de roteador com preempção nos *buffers* de entrada.

$$Area_{crossbar} = 5 * \mathrm{MUX}_4 \tag{6.7}$$

$$Area_{mecanismo\_preempçao} = (4 * N - 2) * \mathrm{MUX}_4 + MUX_2 \tag{6.8}$$

$$\text{onde, } N = tam\_buffer * .\#\mathrm{VC} \tag{6.9}$$

$$Area_{sem\_VC} = (4 * tam\_buffer * .\#\mathrm{VC} - 3) * \mathrm{MUX}_4 + MUX_2 \tag{6.10}$$

Então, o custo de área para a estratégia sem canais virtuais é menor que uma estratégia que utilize canais virtuais se N na equação (6.9) é pequeno. Em outras palavras, a área da NoC com o mecanismo de preempção nos *buffers* de entrada tem menor custo que com canais virtuais se a profundidade do *buffer* (*tam_buffer*) é pequena.

6.6 Estratégias Adicionais Propostas

Todas essas estratégias apresentadas nas seções anteriores deste capítulo já foram exploradas em trabalhos de outros autores, conforme indicado na Tabela 4.1. Elas foram utilizadas também neste trabalho como forma de estabelecer os limites no espaço de projeto, principalmente quando associadas com estratégias adicionais, que são descritas nesta seção: (*i*) incremento, por envelhecimento, da prioridade dos pacotes e, (*ii*) descarte de pacotes antigos. Essas duas novas estratégias já foram utilizadas em redes de interconexão, mas até onde se sabe, não existem ainda trabalhos que as utilizem em NoCs.

6.6.1 Incremento de Prioridade por Envelhecimento

Quando são utilizadas estratégias de controle de fluxo baseadas em prioridade, o árbitro seleciona a porta que possui o pacote cuja prioridade é a mais alta. Caso exista mais de uma porta, cujos pacotes tenham a mesma prioridade, a arbitragem Round Robin será aplicada. Aqueles pacotes, cuja prioridade seja menor serão preteridos. Com isso, existe a possibilidade de que esses pacotes com prioridades menores sofram postergação indefinida.

Uma solução para este problema, em redes de interconexão, foi apresentada por Li e Mutka (LI 1996). Neste trabalho, os autores sugerem um incremento de prioridade para aqueles pacotes que estejam bloqueados por outros de maior prioridade. Em NoCs, não temos conhecimento de outro trabalho que utilize esta estratégia para resolver este tipo de problema.

O incremento de prioridade por envelhecimento ocorrerá quando um pacote de baixa prioridade for bloqueado por outros pacotes mais prioritários. Esse incremento será efetivado depois de um determinado número de ciclos. Essa quantidade de ciclos que o sistema aguarda até incrementar a prioridade pode ser configurada pelo projetista. A idéia é que haja um número mínimo de ciclos de espera até o incremento da prioridade. O número mínimo se explica pelo fato de que todos os pacotes sofrem um atraso mínimo ao passar pelo roteador. Na SoCIN, por exemplo, são necessários pelo menos três ciclos para um pacote seguir para o roteador seguinte. Na primeira etapa, a porta de entrada faz a requisição à porta de saída e é feita em um ciclo. Na segunda etapa, a porta de saída concede o direito de enviar os dados para a porta de saída. Mas, isso somente acontecerá no ciclo seguinte se houver espaço no *buffer* do próximo roteador e se a porta de entrada tiver sido a escolhida para enviar o pacote. Após a seleção da porta de entrada, na terceira etapa, os dados são enviados. Assim, caso a segunda etapa aconteça sem bloqueios, ou seja, em um ciclo, o tempo mínimo que um pacote gasta por roteador é de três ciclos. Nos experimentos realizados, que serão detalhados no Capítulo 7, utilizamos como critério o incremento após três ou cinco ciclos, a partir da segunda tentativa de envio de um pacote. E esse incremento de prioridade do pacote permanece para os roteadores seguintes.

6.6.2 Descarte de Pacotes Antigos

Devido às limitações dos sistemas embarcados, a economia de recursos neste tipo de sistema torna-se ainda mais importante do que nos sistemas convencionais. Dessa forma, outra estratégia analisada foi o descarte de pacotes antigos, cujos prazos (*deadlines*) não conseguirão ser atendidos. Essa estratégia também já foi utilizada em

redes de interconexão (LI 1994), mas não temos conhecimento de sua utilização em NoCs.

O descarte de pacotes antigos permite a liberação de recursos, como canais e *buffers*, para outros pacotes. Isso permitiria que estes outros pacotes ainda pudessem chegar aos seus destinos dentro de seus respectivos prazos.

Entretanto, quando se efetua o descarte de pacotes, o custo para realizar novo envio desses dados descartados pode ser maior que o benefício obtido de liberação dos recursos. Uma forma de evitar esse reenvio de pacotes descartados é limitar o descarte somente para os tipos de mensagens onde possam ocorrer perdas de alguns pacotes, isto é, para aqueles tipos de mensagens que não representam aplicações críticas.

Além do tipo de mensagem, outra questão que precisa ser definida, para decidir sobre o descarte de um pacote, é o momento em que isso será feito. Uma opção é descartar somente quando o prazo estiver perdido, ou seja, quando essa informação no cabeçalho do pacote indicar que o tempo para atingir o destino foi ultrapassado. Porém, se considerarmos a latência mínima que o pacote ainda terá até o destino (incluindo o atraso nos roteadores intermediários) poderemos realizar esse descarte mais cedo, sem que isso implique em descarte de dados válidos, isto é, de dados que ainda tivessem chance de chegar ao destino dentro prazo.

No caso da rede SoCIN, as informações do roteamento XY (onde é possível extrair quantos roteadores intermediários existem até o destino) aliadas às informações do prazo da mensagem (contida no cabeçalho dos pacotes) permitem que o descarte seja feito ainda antes de atingido o momento do *deadline*. Isso é possível, considerando o tempo que o pacote ainda dispensará para percorrer os roteadores intermediários.

Na equação (6.11) é mostrada a fórmula considerada na decisão do descarte. O prazo expressa o número de ciclos que o pacote ainda tem para atingir o destino. E *hops* é o número de roteadores no caminho até o destino. O seu valor é multiplicado por *n*, que é o atraso mínimo, em número de ciclos, para cada roteador. No caso da SoCIN, o valor mínimo para *n* é de três ciclos, que representa a latência mínima por roteador.

$$prazo - n * hops < 0 \tag{6.11}$$

A estratégia do descarte de pacotes antigos busca liberar recursos para os pacotes mais prioritários ou para aqueles que possuam a mesma prioridade, mas que ainda tem condições de atender os seus prazos. E os pacotes descartados devem ser de aplicações que possam perder pacotes como, por exemplo, as de tempo real *soft*.

6.7 Considerações

Neste capítulo foram apresentadas as estratégias de configuração da NoC. Algumas características foram consideradas fixas e não parametrizáveis, ou seja, assumidas como restrição de projeto. Outras foram avaliadas, permitindo a variação através da configuração de parâmetros.

Dentre as características consideradas como fixas estão: (*i*) a topologia; (*ii*) o roteamento; e (*iii*) o chaveamento. A topologia utilizada foi a grelha 2D. No

roteamento, o algoritmo usado foi o XY determinístico. O chaveamento é feito por pacotes e do tipo *wormhole*.

As características avaliadas foram: (*i*) arbitragem; (*ii*) mecanismos de controle de fluxo; e (*iii*) memorização. Na arbitragem foram comparados o tipo de árbitro original da SoCIN, o Round Robin, com um árbitro baseado em prioridades. E essas prioridades são associadas de acordo com o tipo de cada mensagem. No controle de fluxo, foi comparada a utilização de canais virtuais para realizar esse controle com a utilização de *buffers* únicos por canal físico. Em relação à memorização, a profundidade dos *buffers* de entrada foi avaliada, bem como, o mecanismo de ordenamento. No ordenamento, o mecanismo original foi comparado com um mecanismo preemptivo, baseado em filas FIFO, de acordo com a prioridade do pacote.

Além dessas estratégias, outras técnicas de redes de interconexão foram também avaliadas: (*i*) incremento de prioridade, por envelhecimento e (*ii*) descarte de pacotes antigos. O incremento de prioridade é utilizado para evitar a postergação indefinida de pacotes com baixa prioridade quando são utilizados mecanismos de controle de fluxo baseados em prioridade. Já o descarte de pacotes serve para liberar recursos, através do descarte de pacotes de mensagens menos prioritárias, cujos *deadlines* não conseguirão ser atendidos.

No próximo capítulo serão apresentados os experimentos realizados para avaliar essas estratégias. São mostrados também detalhes do simulador de NoC desenvolvido para realizar este trabalho.

7 EXPERIMENTOS E AVALIAÇÃO DOS RESULTADOS

A escolha dos experimentos se balizou pelas possibilidades do espaço de projeto. Dessa forma, procuramos utilizar aplicações que pudessem cobrir o espectro dos tipos de tráfego em uma NoC. Nos experimentos, utilizamos duas aplicações: (*i*) uma aplicação real, multi-tarefas, de descompressão de um fluxo de imagens MJPEG e (*ii*) uma aplicação sintética com tráfego heterogêneo.

As medidas levantadas na simulação foram: (*i*) latência média das comunicações e (*ii*) percentual de mensagens com prazos perdidos. Essas medidas são consideradas para o total das mensagens e também para as mensagens mais prioritárias, isto é, que pertencem à classe de serviço mais restritiva.

Para realizar esses experimentos desenvolvemos uma ferramenta que simula o funcionamento de uma NoC em grelha 2D e permite a configuração dos parâmetros descritos no Capítulo 6. No restante deste capítulo descrevemos em detalhes este simulador e as demais etapas dos experimentos realizados.

7.1 Ferramenta de Simulação

No intuito de avaliar o desempenho da rede SoCIN, em relação a aplicações de tempo real, desenvolveu-se um simulador em alto nível (NoCSim) para avaliar a latência das comunicações e o atendimento dos prazos das mensagens.

O simulador foi desenvolvido em C++ e representa o funcionamento de uma NoC do tipo grelha 2D, com roteamento XY e cujo chaveamento é feito por pacotes, do tipo *wormhole*. Além dessas características, que são fixas, outras podem ser parametrizáveis, como: (*i*) ***arbitragem***, rotativa ou baseada na prioridade da mensagem; (*ii*) ***controle de fluxo***, com ou sem canais virtuais; (*iii*) ***memorização***, com ou sem preempção no buffer de entrada, além de permitir a configuração da profundidade desses *buffers*; (*iv*) ***incremento da prioridade*** dos pacotes, por envelhecimento, para evitar postergação indefinida e (*v*) ***descarte*** de pacotes antigos.

A Sub-seção 7.1.1 apresenta uma visão geral do simulador, com uma descrição de suas classes. Na sub-seção seguinte (7.1.2) é descrito o seu funcionamento e são apresentados os formatos dos dados de entrada e de saída, bem como a sua interface com o usuário.

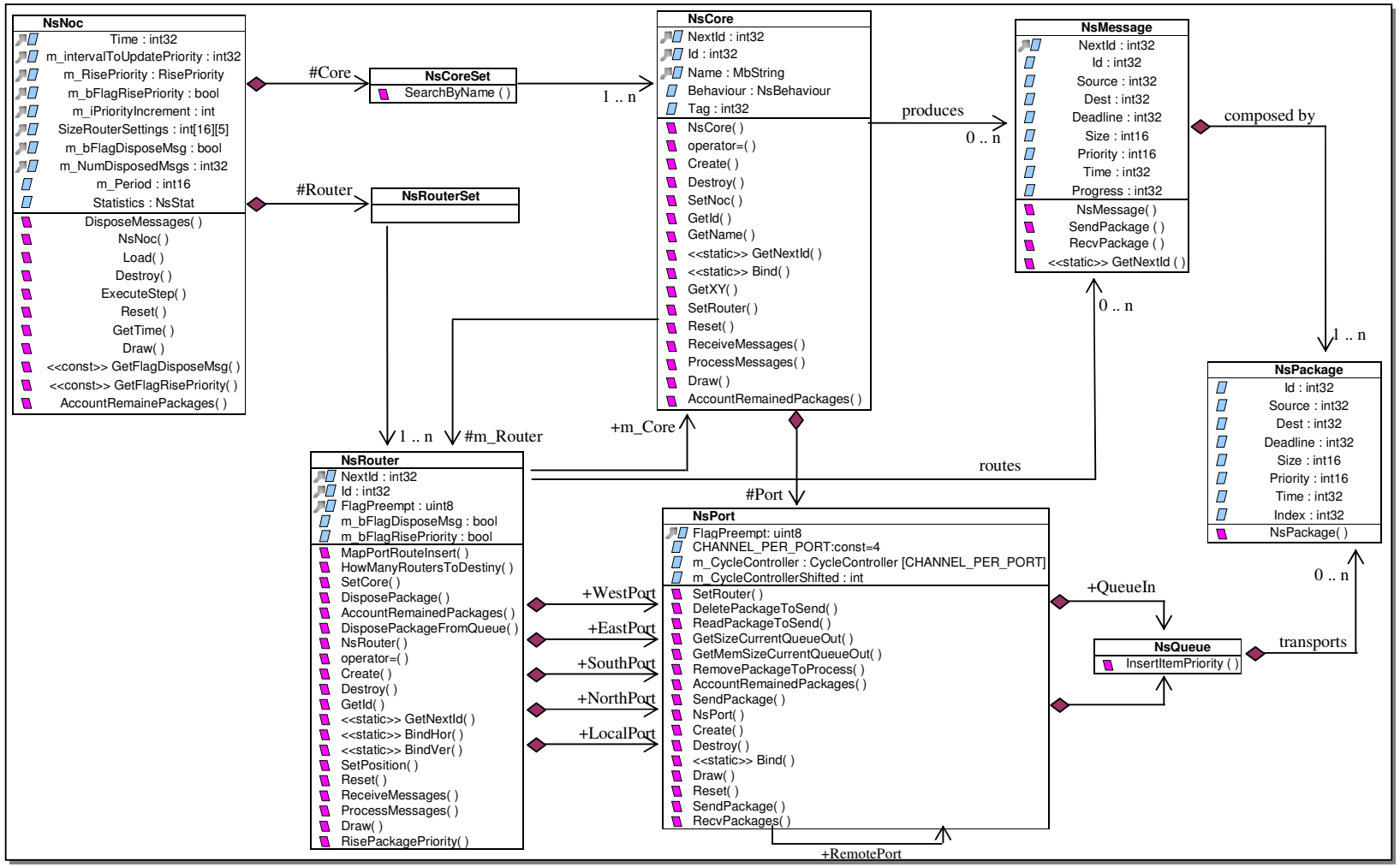

Figura 7.1: Modelagem do simulador de rede NoCSim.

7.1.1 Visão Geral

O objetivo do NoCSim é realizar a simulação do funcionamento da NoC de acordo com as características escolhidas (através dos parâmetros configurados) e com o posicionamento dos núcleos IP resultante da ferramenta Dragon Lemon (descrita na Seção 6.2).

O NoCSim foi desenvolvido em C++, orientado a objetos, e as suas classes seguem a modelagem apresentada na Figura 7.1. São sete as classes principais: NsNoc; NsCore; NsRouter; NsPort; NsMessage; NsQueue e NsPackage.

NsNoc é a classe principal que instancia a NoC, composta por um conjunto de núcleos (NsCore) e roteadores (NsRouter). O número de núcleos e roteadores é definido pelo número de nodos, informado no arquivo de entrada.

A classe NsCore instancia os núcleos, e cada instancia desses núcleos é associada a uma instância do roteador (NsRouter). Cada instância criada pela NsCore possui uma instância da classe NsPort, o que significa que o núcleo é ligado a uma porta. Essa porta é a porta local, que conecta cada núcleo a cada roteador. Essa classe também é responsável por produzir as instâncias da classe NsMessage. Em outras palavras, são os núcleos que geram as mensagens.

NsRouter é a classe que gera os roteadores, que são associados aos núcleos e possuem cinco instâncias da classe NsPort. Essas cinco portas são as quatro direcionais (N,S,E,W) que são conectadas aos outros roteadores e porta local que é conectada ao núcleo. Essa classe é responsável pelo roteamento das instâncias da classe NsMessage, ou seja, suas instâncias (os roteadores) realizam o roteamento dos pacotes.

A classe NsPort instancia as cinco portas dos roteadores e a porta local dos núcleos. Ela possui instâncias da NsQueue, que são a fila de entrada e a fila de saída de cada porta.

Finalmente, a classe NsMessage é responsável pela instanciação das mensagens. Cada instância dessa classe é composta de instâncias da classe NsPackage, que são os pacotes que trafegam na rede.

7.1.2 Funcionalidade

O NoCSim utiliza como entrada o arquivo gerado na saída da ferramenta de posicionamento, cujo formato é o ilustrado na Figura 6.4. Na primeira linha consta a informação do tamanho da rede. Na segunda linha tem a denominação dos núcleos. Em seguida seguem as linhas com a "topologia" das comunicações, isto é, as linhas que descrevem quais núcleos se comunicam e com quem. Em cada uma dessas linhas é descrito o emissor da mensagem, seguido do receptor, da taxa de comunicação, do deadline e da prioridade associada. Após as linhas que descrevem as comunicações vem a indicação do posicionamento dos núcleos na rede.

A execução do simulador pode ser feita através de interface gráfica, como ilustrado na Figura 7.2, ou através de linha de comando. Em ambos os casos, as entradas são: o arquivo com a descrição da rede; os parâmetros de configuração da rede e os parâmetros de configuração da própria simulação.

O arquivo que descreve a rede é o arquivo gerado na saída da ferramenta de posicionamento, contendo o tamanho da NoC, a descrição das comunicações e o posicionamento dos núcleos.

Os parâmetros de configuração da NoC são: tamanho dos *buffers*; tamanhos mínimo e máximo dos pacotes; quantidade de canais virtuais; tipo dos *buffers* (FIFO ou preemptivo); se haverá descarte dos pacotes antigos; e, se haverá incremento de prioridade e após quantos ciclos isso ocorrerá. Uma outra característica, também parametrizável, é o número mínimo de ciclos gasto no roteamento, que no caso da SoCIN equivale a três ciclos.

Quanto aos parâmetros de configuração da simulação, eles são: o tempo total da simulação, em número de ciclos; e, a quantidade de passos que serão executados.

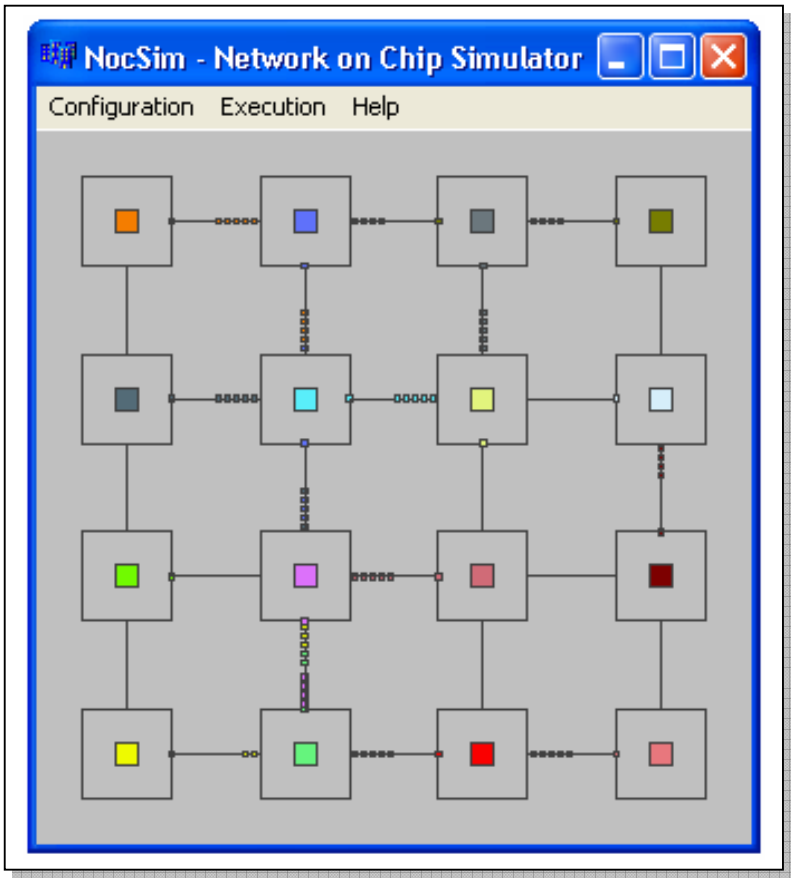

Figura 7.2: Interface gráfica do simulador NoCSim.

De acordo com a taxa de comunicação (indicada no arquivo de entrada) e com os tamanhos mínimo e máximo dos pacotes (indicados através dos parâmetros de configuração do simulador) os núcleos decidem, de forma randômica, se naquele ciclo de simulação irão enviar algum pacote. Essa randomicidade tenta garantir um comportamento não-viciado e assim ser equivalente ao comportamento de um sistema real. Porém, mesmo o envio de pacotes ocorrendo de forma randômica, a comunicação entre os nodos respeita todas as taxas de comunicação da aplicação.

Finalizada a simulação, os dados são salvos em um arquivo do tipo CSV (*Comma-Separated Values*) (SHAFRANOVICH 2005). Esse tipo de arquivo pode ser facilmente utilizado por ferramentas de tratamento de dados, como planilhas eletrônicas, por exemplo. Esses resultados apresentam, para cada comunicação: o total de pacotes

enviados; as latências mínima, máxima e média; o número de pacote cujos prazos foram perdidos e o número daqueles que ainda estavam circulando na rede ao término da simulação. E, ao final do arquivo, são apresentados: o total de pacotes enviados, a latência média total e o percentual de pacotes cujos prazos foram perdidos.

7.2 Avaliação das Estatísticas de Tráfego

Na etapa de posicionamento, configuramos a ferramenta para escolher os dez melhores resultados para cada carga, de cada aplicação utilizada. Esses resultados foram então utilizados na simulação de cada conjunto de parâmetros do simulador.

O arquivo resultante da simulação, no formato CSV, contém os dados da execução do simulador para cada um desses conjuntos de parâmetros. Para evitar qualquer "contaminação" pela randomicidade no envio dos pacotes pelos núcleos, cada conjunto foi simulado dez vezes, utilizando os dez arquivos resultantes da ferramenta de posicionamento. O objetivo disso é utilizar a médias desses dez resultados e não apenas o resultado de uma simulação isolada. Com isso, esperamos reduzir a possibilidade de desvios nos resultados.

A partir do cálculo das médias dessas simulações realizadas é possível comparar as estatísticas do tráfego entre as possíveis configurações da NoC, analisando o espaço de projeto e selecionando as alternativas que melhor atinjam os requisitos das aplicações dentro das restrições do sistema. Em virtude do grande volume de simulações de dados a serem tratados, algumas etapas da avaliação foram automatizadas. Inicialmente, para a execução das diversas simulações, utilizamos arquivos em lote (*scripts*). Nestes arquivos, cada linha representa uma execução em linha de comando do simulador. Depois de realizadas as simulações, os dados são colocados em planilhas, onde são calculadas as médias para conjunto de parâmetros e são gerados gráficos através de macros. Caso seja necessário repetir alguma simulação, repete-se a execução do arquivo de lote e executam-se novamente as macros que calculam as médias e as que geram os gráficos. Dessa forma, procuramos reduzir a necessidade de intervenção manual. Entretanto, como exposto na Seção 7.3, mesmo com essa automatização o grande número de parâmetros considerados pode dificultar a análise para escolha da melhor configuração.

7.3 Experimentos Realizados

Mesmo com a fixação de alguns parâmetros (topologia, roteamento, chaveamento), a quantidade de parâmetros a serem avaliados ainda representa um largo espectro do espaço de projeto. De forma a melhor avaliar todas as possibilidades de configuração, utilizou-se dois tipos de aplicações e realizaram-se variações nas características como resumido na Tabela 7.1. Além desses parâmetros, as estratégias elencadas na Tabela 7.2 também foram avaliadas para a aplicação sintética.

Em todos os experimentos foi utilizada uma rede em grelha 4x4 por ser um tamanho de rede onde já é possível avaliar os parâmetros do espaço de projeto que se pretendia verificar e por ser um tamanho ainda não tão grande que dificultasse, especialmente em relação ao tempo dispensado nas simulações. Neste trabalho utilizou-se dois tipos de aplicações. A primeira, uma aplicação real, multi-tarefas, de descompressão de um fluxo de imagens MJPEG. Essa aplicação possui um comportamento correspondente a uma aplicação orientada a fluxo de dados cujo tráfego é em seqüência de dados,

conforme indicado no grafo ilustrado na Figura 5.2. Considerando que esta aplicação possui somente sete tarefas e o tamanho da rede utilizada nos experimentos possuía dezesseis nodos, as simulações foram realizadas considerando duas aplicações MJPEG independentes na mesma NoC. A outra aplicação é uma aplicação sintética, gerada através do TGFF, com tráfego heterogêneo.

Em ambas as aplicações, considerou-se pelo menos duas cargas diferentes para os experimentos. Para o MJPEG utilizou-se duas cargas, a original e outra com 50% de tráfego adicional, na tentativa de conseguir um tráfego que pudesse saturar a rede. No caso da aplicação sintética utilizamos três cargas, com 10%, 25% e 50% de utilização da rede, respectivamente. Ainda no caso da carga sintética, a distribuição do volume de carga para cada tipo de classe de serviço segue o que já foi descrito na Tabela 5.1.

Tabela 7.1: Parâmetros considerados na configuração da NoC

Característica	**Alternativas avaliadas**
Posicionamento	(*i*) Largura de banda; (*ii*) Largura de banda + prioridade.
Arbitragem	(*i*) Round Robin; (*ii*) Baseada em prioridade.
Memorização	Tipo de fila: (*i*) FIFO; (*ii*) Preemptiva. Profundidade do buffer (em *flits*): (*i*) 2; (*ii*) 4; (*iii*) 8 e (*iv*) 16.
Controle de Fluxo	(*i*) Canais virtuais; (*ii*) *Handshake*.

Essas variações de parâmetros, listadas na Tabela 7.1, são descritas de forma mais detalhada para a aplicação MJPEG e para a aplicação sintética nas Subseções 7.3.1 e 7.3.2, respectivamente. Na Sub-seção 7.3.2 também são descritos os resultados obtidos com as estratégias da Tabela 7.2.

Tabela 7.2: Estratégias adicionais consideradas na configuração da NoC

Estratégia	**Alternativas avaliadas**
Incremento de prioridade por envelhecimento	(*i*) Desativado; (*ii*) 3 ciclos; (*iii*) 5 ciclos.
Descarte de pacotes antigos	(*i*) Desativado; (*ii*) Ativado.

7.3.1 Aplicação MJPEG de Tráfego em Seqüência de Dados

No caso da aplicação MJPEG, a estratégia de posicionamento já garantiu a ausência de perdas de prazos. Devido ao comportamento desse tipo de aplicação, onde não ocorre concorrência pelos canais, uma vez que os dados seguem um fluxo unidirecional, a simples localização adequada dos núcleos, através do posicionamento, já garante os requisitos da aplicação. Mesmo com a simulação de duas aplicações MJPEG na NoC, a ferramenta de posicionamento já garantiu o isolamento dos fluxos através da localização dos núcleos na rede.

No intuito de avaliar o impacto dos outros parâmetros, utilizou-se uma outra carga, com aumento correspondente a 50% da carga original. Entretanto, a simples adequação da profundidade dos *buffers* foi suficiente para absorver essa carga adicional, não sendo necessário a utilizações de estratégias adicionais como o incremento de prioridade por envelhecimento ou o descarte de pacotes antigos. Nem tão pouco foi necessário o uso de canais virtuais.

7.3.2 Aplicação Sintética de Tráfego Heterogêneo

Com o objetivo de explorar melhor as características da NoC, como paralelismo, realizou-se então simulações com aplicações sintéticas, com as quais era possível a concorrência por canais. Dessa forma, conseguiríamos observar o impacto da configuração de todos os parâmetros da NoC.

Considerando os dez melhores posicionamentos para cada uma das três cargas, são trinta arquivos para simular. Considerando ainda, os parâmetros da Tabela 7.1 e as estratégias da Tabela 7.2, além de variações na largura de banda (três tamanhos diferentes, por exemplo), o número de execuções é da ordem dos milhares. E, apesar dos arquivos de lotes e macros utilizados para automatizar a avaliação dos dados desses milhares de simulações, o número de dados para se comparar é grande. Assim, tentamos apresentar a seguir alguns resultados, por agrupamento de parâmetros, buscando comparar as diversas alternativas de configuração da NoC.

Antes desses experimentos, uma primeira comparação realizada, apresentada em Corrêa et al. (CORRÊA 2004a), analisava o impacto da estratégia de posicionamento sobre a latência média e o percentual de mensagens cujos prazos foram perdidos. Esses resultados também utilizaram aplicação sintética, mas ainda não consideravam a distribuição de tráfego da Tabela 5.1. Dessa forma, essas estatísticas apresentadas na Tabela 7.3 apresentam valores em relação ao total de mensagens da rede, sem diferenciar as mensagens críticas, cujo prazo é rígido (*hard*). Os resultados são apresentados para dois experimentos, tendo o segundo (Experimento II) o mesmo requisito de banda, mas *deadlines* mais rígidos que o primeiro (Experimento I).

Esses resultados, apresentados na Tabela 7.3, mostraram que a utilização de árbitro baseado em prioridade, em relação ao Round Robin, permite uma redução do percentual de perdas de prazos das mensagens, mas, ao mesmo tempo, gera um aumento no tempo médio gasto nas comunicações. Na comparação das estratégias de posicionamento, quando também foi considerada a prioridade, além da largura de banda, como fator de posicionamento, houve ainda uma pequena redução no percentual de prazos perdidos. Entretanto, ocorre um aumento na latência média, o que pode implicar em um maior consumo de energia.

No intuito de confirmar esses resultados iniciais, avaliando também o impacto sobre as mensagens críticas, os novos experimentos foram realizados seguindo a distribuição de carga da Tabela 5.1. Além disso, o objetivo desses novos experimentos, baseados em classes de serviço, era também permitir a avaliação das demais estratégias propostas nesta tese. Os resultados destes experimentos são apresentados na seqüência, com a análise do impacto dos diversos parâmetros de configuração.

Tabela 7.3: Experimentos iniciais para avaliação dos parâmetros da NoC

Estratégias			**Experimento I**		**Experimento II**	
Posicionamento	**Arbitragem**	**Tipo de *Buffer***	**Latência Média**	**% Prazos Perdidos**	**Latência Média**	**% Prazos Perdidos**
BW	RR	FIFO	14.79	33	14.79	20
BW	Prioridade	FIFO	25.36	18	25.36	9
BW	Prioridade	Preemptivo	28.26	17	28.26	8
BW + prioridade	Prioridade	FIFO	27.73	15	27.73	6
BW + prioridade	Prioridade	Preemptivo	28.49	15	28.49	6

BW: largura de banda; RR: Round Robin.

As tabelas 7.4, 7.5 e 7.6 utilizam, respectivamente, larguras de banda de 2, 4 e 8 *flits*/ciclo. Elas apresentam os resultados para as estratégias de posicionamento, arbitragem e tipo de buffer, considerando profundidade de buffer de 8 *flits* e controle de fluxo do tipo *handshake*. Os resultados são apresentados, considerando o total das comunicações críticas (*hard*) e o total de todos os tipos de todas as comunicações.

Tabela 7.4: Experimentos com carga sintética, usando classes de serviços (BW=2*flits*/ciclo).

Estratégias			**Tipo de Mensagem**	**10%**		**25%**		**50%**	
Posicion.	**Arbitragem**	***Buffer***		**tempo**	**prazos**	**tempo**	**prazos**	**tempo**	**prazos**
BW	RR	FIFO	Total	10.34	0.79	10.75	1.11	12.68	1.59
BW	Prioridade	FIFO	***Hard***	***10.87***	***8.35***	***11.40***	***9.47***	***12.87***	***17.75***
			Total	11.66	0.46	12.10	0.60	14.24	0.96
BW	Prioridade	Preemptivo	***Hard***	***11.18***	***9.65***	***11.45***	***10.70***	***12.88***	***18.03***
			Total	11.72	0.50	12.21	0.72	14.29	0.99
BW + prioridade	Prioridade	FIFO	***Hard***	***11.27***	***9.81***	***11.49***	***10.80***	***13.33***	***18.23***
			Total	11.75	0.55	12.25	0.81	14.38	1.02
BW + prioridade	Prioridade	Preemptivo	***Hard***	***11.79***	***10.40***	***11.89***	***11.47***	***13.69***	***18.64***
			Total	11.88	0.59	12.27	0.82	14.51	1.05

BW: largura de banda; RR: Round Robin.

Tabela 7.5: Experimentos com carga sintética, usando classes de serviços (BW=4*flits*/ciclo).

Estratégias			Tipo de Mensagem	10%		25%		50%	
Posicion.	**Arbitragem**	**Buffer**		**tempo**	**prazos**	**tempo**	**prazos**	**tempo**	**prazos**
BW	RR	FIFO	Total	9.86	0.53	10.05	1.03	11.66	1.38
BW	Prioridade	FIFO	***Hard***	***10.02***	***5.20***	***10.20***	***6.54***	***10.60***	***7.88***
			Total	10.22	0.32	11.39	0.54	11.43	0.83
BW	Prioridade	Preemptivo	***Hard***	***10.08***	***5.94***	***10.28***	***6.59***	***10.76***	***7.93***
			Total	10.31	0.34	11.45	0.64	11.55	0.86
BW + prioridade	Prioridade	FIFO	***Hard***	***10.23***	***6.01***	***10.35***	***6.83***	***10.87***	***8.73***
			Total	10.35	0.35	11.56	0.79	12.25	0.88
BW + prioridade	Prioridade	Preemptivo	***Hard***	***10.61***	***6.43***	***10.67***	***7.20***	***10.92***	***8.76***
			Total	10.39	0.37	11.96	0.80	12.56	0.90

BW: largura de banda; RR: Round Robin.

Tabela 7.6: Experimentos com carga sintética, usando classes de serviços (BW=8*flits*/ciclo).

Estratégias			Tipo de Mensagem	10%		25%		50%	
Posicion.	**Arbitragem**	**Buffer**		**tempo**	**prazos**	**tempo**	**prazos**	**tempo**	**prazos**
BW	RR	FIFO	Total	9.17	0.42	9.63	0.64	10.40	1.28
BW	Prioridade	FIFO	***Hard***	***9.80***	***4.49***	***10.13***	***6.47***	***10.46***	***7.87***
			Total	8.57	0.22	9.42	0.35	9.47	0.61
BW	Prioridade	Preemptivo	***Hard***	***9.70***	***4.36***	***10.06***	***6.37***	***10.41***	***7.85***
			Total	8.58	0.23	9.59	0.36	9.67	0.69
BW + prioridade	Prioridade	FIFO	***Hard***	***9.50***	***4.27***	***9.93***	***6.32***	***10.35***	***7.78***
			Total	8.66	0.24	9.68	0.38	9.70	0.79
BW + prioridade	Prioridade	Preemptivo	***Hard***	***9.32***	***3.74***	***9.72***	***6.29***	***10.32***	***7.59***
			Total	8.69	0.28	9.77	0.39	9.82	0.82

BW: largura de banda; RR: Round Robin.

Como nos experimentos anteriores, o árbitro Round Robin proporciona menor latência média, mas um maior número de perdas de prazos, em relação ao uso de arbitragem baseada em prioridades. Entretanto, essa comparação se baseia no total de todas as comunicações, de todos os tipos. Isto porque a arbitragem do tipo Round Robin não faz diferenciação em relação a níveis de serviço, ou seja, em relação às prioridades das mensagens.

Quanto aos *buffers*, o tipo preemptivo permite uma redução, em relação ao tipo FIFO. Essa redução ocorreu no número de perdas dos prazos e no tempo médio das mensagens críticas. Mas, isso somente ocorreu quando a largura de banda utilizada foi de 8 *flits*/ciclo (Tabela 7.6). A explicação reside no fato de que, mesmo este tipo de comunicação sendo mais prioritário, a carga até então sobrecarregava a rede, impedindo de atender os requisitos das aplicações.

Essa mesma conclusão pode ser aplicada ao caso das estratégias de posicionamento. Quando a carga utilizada possui um tráfego extremamente pesado, a utilização do posicionamento baseado em largura de banda e também na prioridade não permitirá redução no número de prazos perdidos ou na latência média.

Então, no intuito de atingir os requisitos das aplicações com a menor quantidade de recursos, outras estratégias foram avaliadas. O uso de canais virtuais, por exemplo, permite o isolamento de tráfego para diferentes tipos de comunicação. Entretanto, essa alternativa aumenta o custo de área e de potência, devido ao acréscimo de memória necessário. Uma alternativa, que utilize *buffers* preemptivos, sem canais virtuais, pode economizar em memória, mas necessita de algum outro mecanismo que garanta às mensagens menos prioritárias que elas não irão sofrer uma postergação indefinida. Para isso, avaliamos a utilização de um mecanismo de incremento de prioridade, por envelhecimento, das mensagens menos prioritárias que estejam bloqueadas depois de um determinado número de ciclos. Na seqüência desse texto, apresentamos alguns resultados onde é possível comparar a utilização do mecanismo de incremento de prioridades por envelhecimento com a utilização de canais virtuais.

A Tabela 7.7 apresenta um comparativo entre as estratégias de incremento de prioridades e de canais virtuais. Esses resultados referem-se ao seguinte conjunto de parâmetros: (*i*) posicionamento baseado somente em largura de banda; (*ii*) largura de banda de 4 *flits*/ciclo; (*iii*) profundidade dos *buffers* de 8 *flits* (4 *buffers* de 2 *flits* cada, no caso dos canais virtuais); (*iv*) arbitragem baseada em prioridades; e, (*v*) controle de fluxo preemptivo.

A Figura 7.3 apresenta um gráfico comparativo da latência média das comunicações entre as estratégias de canais virtuais e de incremento de prioridade, enquanto a Figura 7.4 apresenta o gráfico comparativo dos atendimentos dos prazos para estas mesmas estratégias. Os resultados apresentados nesses gráficos referem-se ao seguinte conjunto de parâmetros: (*i*) posicionamento baseado somente em largura de banda; (*ii*) carga de 25% de tráfego; (*iii*) profundidade dos *buffers* de 8 *flits* (4 *buffers* de 2 *flits* cada, no caso dos canais virtuais); (*iv*) arbitragem baseada em prioridades; e, (*v*) controle de fluxo preemptivo.

Tabela 7.7: Resultados dos experimentos para estratégias com canais virtuais e com incremento de prioridade, em NoC com largura de banda de 4*flits*/ciclo.

Estratégias	Tipo de Mensagem	10%		25%		50%	
		tempo	prazos	tempo	prazos	tempo	prazos
Sem incremento e sem canais virtuais	*Hard*	***10.02***	***5.20***	***10.20***	***6.54***	***10.60***	***7.88***
	Total	10.22	0.32	11.39	0.54	11.43	0.83
Incremento (depois de **3** ciclos)	*Hard*	***9.63***	***5.20***	***10.04***	***5.97***	***10.09***	***6.47***
	Total	9.46	0.33	10.89	0.34	11.06	0.72
Incremento (depois de **5** ciclos)	*Hard*	***9.63***	***5.19***	***9.83***	***5.83***	***9.99***	***6.22***
	Total	9.44	0.32	9.86	0.31	10.85	0.67
Canais Virtuais	*Hard*	***17.30***	***13.65***	***23.47***	***16.24***	***30.58***	***19.75***
	Total	16.54	1.93	23.75	3.87	30.49	4.71

A estratégia de incremento de prioridades por envelhecimento apresenta resultados semelhantes quando são considerados 3 ou 5 ciclos, conforme mostram as respectivas linhas na Tabela 7.7. Esse parâmetro de número de ciclos indica quanto tempo o sistema aguarda antes de incrementar a prioridade de uma mensagem bloqueada por outra mais prioritária. A utilização do valor mínimo de 3 ciclos se deve a ser esse o número mínimo de ciclos que uma mensagem gasta em cada roteador da rede SoCIN, como explicado anteriormente neste texto.

Entretanto, quando comparamos a estratégia de incremento com a estratégia de canais virtuais, a primeira apresenta um melhor resultado com menos recursos, como pode ser observado no gráfico da Figura 7.3, onde a latência média reduz mais rapidamente com um menor aumento de largura de banda. O mesmo é observado no gráfico da Figura 7.4, em relação ao atendimento dos prazos.

Pode-se observar, por exemplo, que quando a largura de banda é de 2 *flits*/ciclo, ambas as estratégias provocam aumento das perdas de prazos. Isso porque elas provocam um aumento na latência média devido ao volume de tráfego, como mostrado nas primeiras colunas da esquerda do gráfico da Figura 7.3. Porém, no caso da estratégia de incremento, o aumento da largura de banda para 4 *flits*/ciclo já é suficiente para provocar uma redução na latência e no número de perdas de prazos. Enquanto, para a estratégia de canais virtuais, o mesmo não ocorre. Para esta, a redução ocorre de forma mais lenta, ou somente com mais recursos de banda ou de memorização (*buffers* maiores).

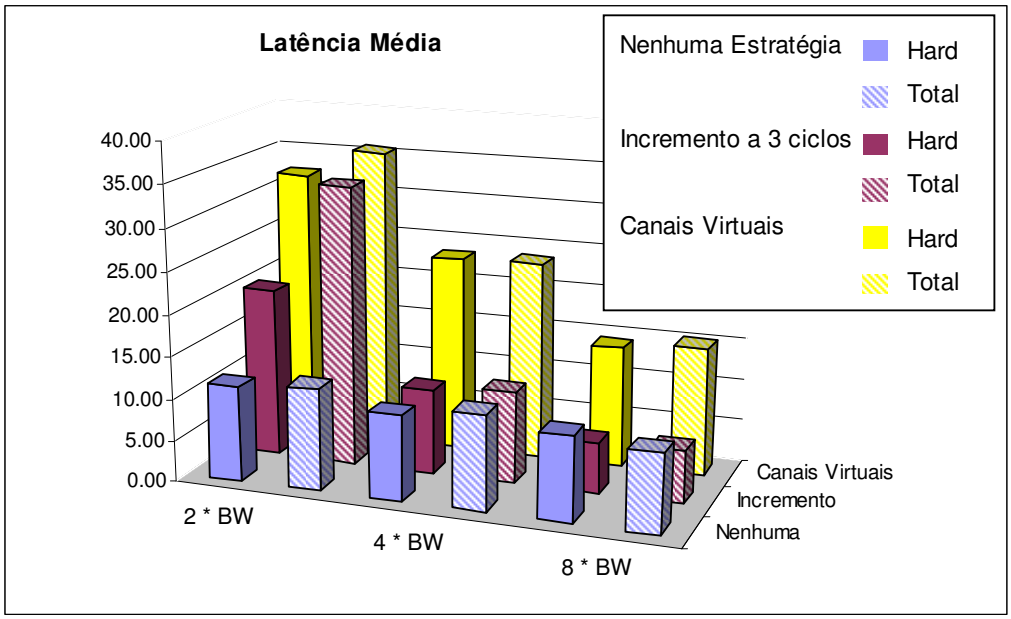

Figura 7.3: Comparativo da latência entre canais virtuais e incremento de prioridade para carga de 25% do tráfego.

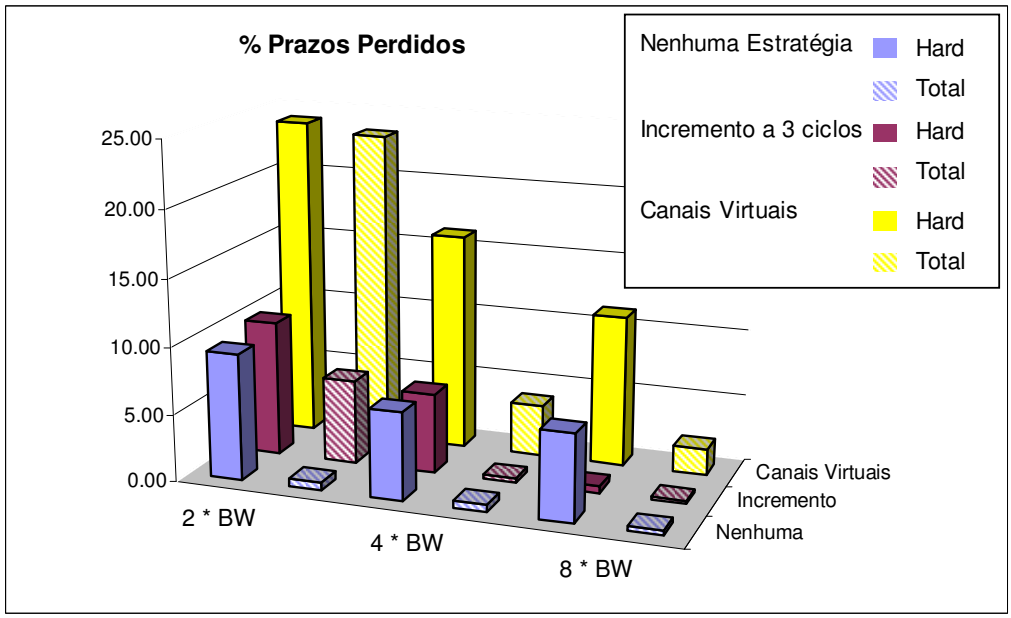

Figura 7.4: Comparativo do atendimento dos prazos nas estratégias com canais virtuais e com incremento de prioridade, para carga de 25% do tráfego.

A Tabela 7.8 apresenta um comparativo entre as estratégias de incremento de prioridades e de descarte. Esses resultados referem-se aos seguintes conjuntos de parâmetros: (*i*) posicionamento baseado somente em largura de banda; (*ii*) largura de banda de 4 *flits*/ciclo; (*iii*) profundidade dos *buffers* de 8 *flits*; (*iv*) arbitragem baseada em prioridades; e, (*v*) controle de fluxo preemptivo.

Tabela 7.8: Resultados dos experimentos para estratégias com descarte e com incremento de prioridade, em NoC com largura de banda de 4*flits*/ciclo.

Estratégias	**Tipo de Mensagem**	**10%**		**25%**		**50%**	
		tempo	**prazos**	**tempo**	**prazos**	**tempo**	**prazos**
Sem incremento e sem descarte	Total	***10.02***	***5.20***	***10.20***	***6.54***	***10.60***	***7.88***
	Hard	10.22	0.32	11.39	0.54	11.43	0.83
Descarte	Total	***10.08***	***5.21***	***10.15***	***6.01***	***10.28***	***7.55***
	Hard	10.25	0.41	10.92	0.43	11.30	0.72
Incremento (depois de **3** ciclos)	Total	***9.93***	***5.20***	***10.04***	***5.97***	***10.09***	***6.47***
	Hard	9.96	0.33	10.89	0.34	11.06	0.72
Descarte + Incremento (depois de **3** ciclos)	Total	***9.02***	***5.17***	***9.81***	***5.81***	***9.85***	***6.30***
	Hard	9.61	0.30	10.45	0.32	10.69	0.66

A Figura 7.5 apresenta um gráfico comparativo da latência média das comunicações entre as estratégias de descarte e de incremento de prioridade, enquanto a Figura 7.6 apresenta o gráfico comparativo dos atendimentos dos prazos para estas mesmas estratégias. Os resultados apresentados nesses gráficos referem-se ao seguinte conjunto de parâmetros: (*i*) posicionamento baseado somente em largura de banda; (*ii*) carga de 25% de tráfego; (*iii*) profundidade dos *buffers* de 8 *flits*; (*iv*) arbitragem baseada em prioridades; e, (*v*) controle de fluxo preemptivo.

Os dados na Tabela 7.8 demonstram que a estratégia de descarte apresenta um resultado inferior ao obtido pela estratégia de incremento de prioridade, como podemos perceber ao observar as respectivas linhas das estratégias. A estratégia de descarte pode apresentar, em alguns casos (por exemplo, com a carga de 10%), um aumento da latência média e do número prazos perdidos, quando comparada com o caso onde nenhuma estratégia é utilizada. Por outro lado, a estratégia de canais virtuais apresenta redução quando comparada com o mesmo caso, bem como, com a estratégia de descarte. Entretanto, se a estratégia de descarte for utilizada conjuntamente com a estratégia de incremento de prioridades implicará em obter os melhores resultados. Resultados melhores comparados com todas as outras opções.

A utilização dessas duas estratégias (incremento e descarte), em conjunto, apresenta um resultado satisfatório mesmo quando individualmente cada uma delas tem um resultado pior do que o caso onde nenhuma delas é utilizada. Isso pode ser verificado nos gráficos da Figura 7.5. No caso onde a largura de banda é 2 *flist*/ciclo, tanto a

estratégia de descarte, quanto a estratégia de incremento de prioridade apresentam uma maior latência média em relação ao caso onde nenhuma dessas estratégias é utilizada. Mas, quando as estratégias são utilizadas em conjunto, ocorre uma redução em relação a todas as demais alternativas. O mesmo ocorre em relação ao percentual de perdas de prazos, como podemos perceber na Figura 7.6.

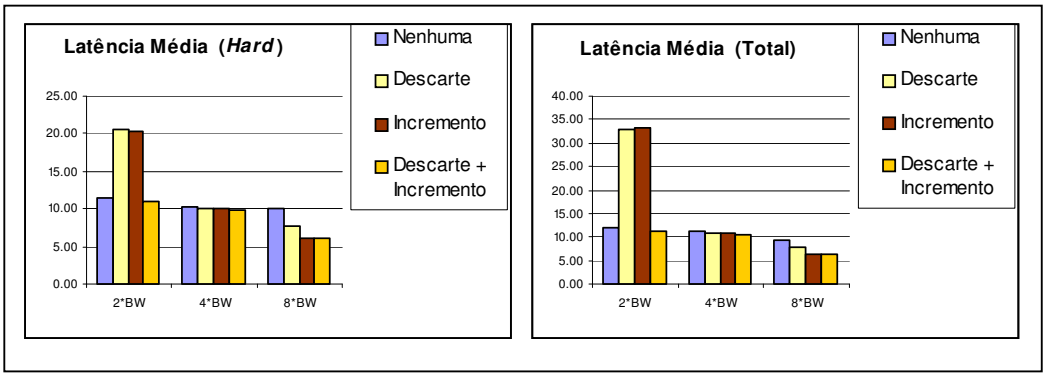

Figura 7.5: Comparativo da latência média entre as estratégias com descarte e com incremento de prioridade, para carga de 25% do tráfego.

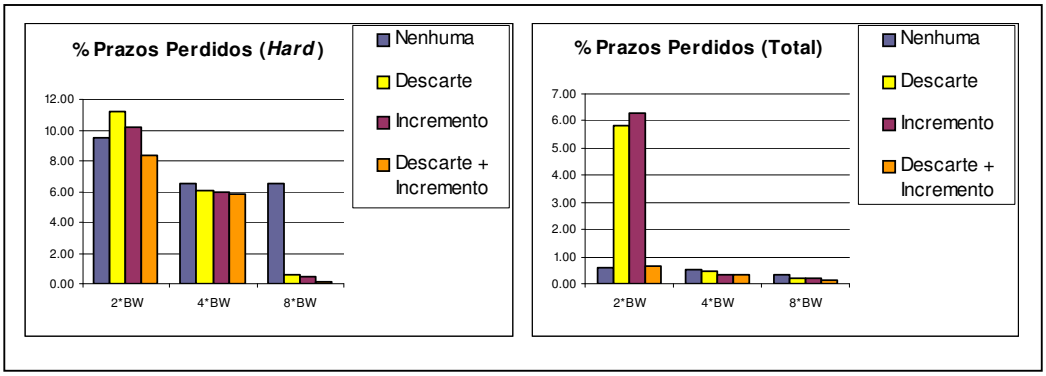

Figura 7.6: Comparativo do atendimento dos prazos nas estratégias com descarte e com incremento de prioridade, para carga de 25% do tráfego.

7.4 Considerações

Neste capítulo foram apresentadas as avaliações das estratégias propostas no Capítulo 6. Inicialmente, mostrou-se os detalhes da ferramenta de simulação de NoC desenvolvida para avaliar essas estratégias. O simulador foi desenvolvido em C++ e simula uma rede em grelha 2D como a rede SoCIN. Depois, detalhou-se os experimentos realizados no intuito de se obter resultados que embasassem as avaliações.

Nesses experimentos, procurou-se aplicações que permitissem a exploração do espaço de projeto, cobrindo o espectro dos tipos de tráfego comuns a uma NoC. Para isso, utilizou-se dois tipos de aplicações: a primeira, real, de descompressão MJPEG, cujo tráfego é em seqüência de dados; e, a outra, sintética, de tráfego heterogêneo.

Para a avaliação das estratégias, as medidas realizadas foram o percentual de mensagens com prazos perdidos e a latência média das comunicações. Essas medidas são consideradas para o total das mensagens e também para as mensagens mais prioritárias, isto é, que pertencem à classe de serviço mais restritiva.

As configurações da NoC avaliadas foram: posicionamento do núcleos; arbitragem, rotatória ou prioritária; mecanismos de controle de fluxo, *handshake* ou canais virtuais; e, memorização, quanto ao tipo de fila dos *buffers* e a profundidade destes. Além dessas estratégias de configuração da NoC, outras técnicas originárias de redes de interconexão tradicionais foram avaliadas: o incremento de prioridade, por envelhecimento, das mensagens menos prioritárias que se encontram bloqueadas depois de um certo tempo; e, o descarte de pacotes antigos.

Dos resultados apresentados, concluiu-se que o desempenho da rede está fundamentalmente ligado ao tipo de aplicação e à carga aplicada. Assim, os projetos de NoCs dependem dos requisitos e do comportamento dessas aplicações. Requisitos, como latência máxima e percentual de perdas de prazos são requisitos que dependem do tipo de aplicação. O comportamento das aplicações inclui características do tráfego, como taxa de transmissão e tamanho dos pacotes.

Em relação à utilização de estratégias, pode-se destacar que as novas estratégias propostas apresentam resultados distintos quando usadas isoladamente e quando são utilizadas em conjunto, como é o caso das estratégias de incremento de prioridades por envelhecimento e de descarte dos pacotes antigos.

No próximo capítulo serão feitas considerações finais, avaliações sobre os resultados obtidos nesta tese e sobre perspectivas futuras.

8 CONSIDERAÇÕES FINAIS

Este trabalho abordou questões sobre o ajuste no planejamento e configuração das redes em chip para atender aos requisitos das aplicações de tempo real em sistemas embarcados. Essas questões tratam de requisitos de qualidade de serviço, especificamente, em relação ao atendimento dos prazos das tarefas e à latência das comunicações.

Neste texto discutimos inicialmente as arquiteturas de comunicação em sistemas embarcados, com ênfase na alternativa de redes em chip, apresentando seus conceitos e características. Discutimos também o impacto das restrições adicionais, em projetos de sistemas embarcados, impostas pelas aplicações de tempo real. Aplicações essas que são caracterizadas pelo tipo de tráfego e pelas suas propriedades. Apresentamos um panorama sobre o espaço de projeto em sistemas embarcados baseados em NoCs, incluindo o estada da arte da pesquisa nesta área. Uma metodologia para adequar as NoCs às aplicações de tempo real foi proposta. Essa metodologia busca, a partir das especificações das aplicações, encontrar um posicionamento dos núcleos baseado nos requisitos da comunicação entre eles e, após isso, configurar os demais parâmetros da NoC, como arbitragem, mecanismos de controle de fluxo e memorização. Além da configuração desses parâmetros, estratégias baseadas em técnicas originárias das redes de interconexão tradicionais foram avaliadas. Essas estratégias são: o descarte de pacotes antigos e o incremento de prioridades por envelhecimento. Detalhamos os parâmetros considerados nessas configurações e as medidas consideradas na avaliação desses parâmetros. A obtenção dessas medidas é feita através do simulador de NoCs desenvolvido para esta tese. E as medidas são: a latência média das comunicações e o percentual de tarefas cujos prazos foram atendidos. Com base nessas medidas, avaliamos as alternativas de configuração da NoC e as comparamos entre si. Diversos experimentos foram simulados para obter as medidas para avaliação.

Neste capítulo apresentamos uma visão geral sobre os resultados deste trabalho, pois as considerações mais específicas já foram feitas ao final de cada capítulo. Além dessa visão geral, procuramos relatar sobre o histórico do trabalho e discutir algumas perspectivas de trabalhos futuros.

Em função dos requisitos dos sistemas embarcados, especialmente em relação às limitações do tempo de projeto, espera-se que os SoCs sejam dominados pelo software, ou seja, pelas restrições das aplicações que serão executadas. E em virtude do aumento da complexidade, o ideal é que a exploração do espaço de projeto de sistemas embarcados seja feito no nível de abstração mais alto possível. Dessa forma é possível

explorar um espaço de projeto mais amplo, sem aumentar muito o tempo de lançamento do produto no mercado.

As aplicações executadas nesses SoCs apresentam, em geral, requisitos cada vez mais rígidos em relação à qualidade de serviço e a restrições temporais. Quando os SoCs forem utilizados para aplicações de tempo real, o desempenho e a latência da NoC precisam ser avaliados de forma mais criteriosa, pois em sistemas de tempo real a previsibilidade é um dos quesitos mandatórios.

As estratégias apresentadas para a configuração dos parâmetros da NoC permitiram abranger um largo espaço de projeto. E os experimentos simulados apresentaram resultados conclusivos sobre o impacto dessas estratégias nos requisitos das aplicações de tempo real.

Os capítulos 4, 5 e 6 ajudam a compreender o espaço de projeto para sistemas embarcados baseados em NoCs quando esse tipo sistema executa aplicações de tempo real. As restrições impostas por essas aplicações ao projeto desses sistemas são determinantes e essenciais na definição do espaço de projeto. E a maneira de configurar essa NoC pode influir no desempenho do sistema. Assim, a escolha das estratégias e dos parâmetros adequados ao tipo de aplicação e ao seu tráfego mostrou-se como o ponto crucial do espaço de projeto e que precisa, muitas vezes, ser reavaliado ou confrontados com outras alternativas.

No Capítulo 4 foram listados alguns questionamentos feitos ao longo de um projeto de NoC, em relação a parâmetros como (*i*) topologia; (*ii*) tipo de núcleo IP; (*iii*) largura de banda; (*iv*) tamanho do buffer; (*v*) tipo de arbitragem; (*vi*) algoritmo de roteamento; (*vii*) chaveamento; (*viii*) distribuição dos núcleos IP na NoC; (*ix*) escalonamento das tarefas e ainda sobre (*x*) a existência de outras técnicas para melhor especificar determinadas características da NoC com restrições mais rígidas.

Alguns desses parâmetros ou alternativas foram considerados fixos, como no caso do tipo de topologia (grelha 2D), do algoritmo de roteamento (XY-determinístico), do chaveamento (*wormhole*). Outros foram abstraídos ou simplificados, como no caso do tipo de núcleo IP, escalonamento das tarefas. O uso de casos simplificados (um processo por núcleo no caso do escalonamento, por exemplo) objetivou avaliar os demais parâmetros de forma independente, sem inserir uma nova variável. A fixação de alguns parâmetros seguiu o consenso de outros trabalhos na literatura, onde estes já haviam sido avaliados.

Os demais parâmetros são avaliados através de experimentos apresentados no Capítulo 7. Este capítulo apresenta os experimentos realizados, para as diversas estratégias e parâmetros, e uma avaliação dos resultados dessas simulações. Esses resultados demonstram que o desempenho da rede está diretamente ligado ao tipo de aplicação e à carga aplicada à rede.

Em relação à utilização das estratégias, destacamos que as novas estratégias propostas apresentam resultados distintos quando usadas isoladamente e quando são utilizadas em conjunto, como é o caso das estratégias de incremento de prioridades por envelhecimento e de descarte dos pacotes antigos.

Fato semelhante ocorre em relação aos parâmetros considerados no posicionamento dos núcleos e no tipo de filas nos *buffers*. Dependendo da combinação dessas características o impacto pode ser favorável ou desfavorável aos objetivos de atender os requisitos das aplicações.

Algumas contribuições deste trabalho que se pode citar são:

- a aplicação de técnicas originárias de redes de interconexão tradicionais no contexto das estratégias de configuração das NoCs para tentar reduzir congestões, evitar postergações indefinidas e, assim, atender aos requisitos de latência e de prazos das comunicações.
- a busca de alternativas de menor custo para controle de fluxo, que ofereçam suporte ao tráfego diferenciado, como nos canais virtuais.
- o desenvolvimento de uma ferramenta de simulação, parametrizável, de alto nível da rede SoCIN, que permite uma avaliação de espaço de projeto mais abrangente.

Conclui-se que, dentre os resultados obtidos, pode-se destacar:

- a confirmação de que o desempenho da rede depende da aplicação e da carga utilizada, e dessa forma, a configuração adequada dessa NoC, buscando atender as características dessa aplicação e dessa carga, influi fortemente em requisitos, como, por exemplo: latência máxima e percentual de perdas de prazos (deadlines).
- em relação às estratégias utilizadas na metodologia deste trabalho, as novas estratégias propostas, baseadas em técnicas originárias de redes de interconexão tradicionais, apresentam resultados distintos quando usadas isoladamente ou quando usadas em conjunto. A associação das estratégias, de incremento por prioridades por envelhecimento e de descarte dos pacotes antigos, permite a obtenção de melhores resultados e de forma mais significativa.

8.1 Histórico da Tese

As atividades do doutorado que geraram esta tese tiveram início em 2002 no Programa de Pós-Graduação em Computação (PPGC) da Universidade Federal do Rio Grande do Sul (UFRGS). Neste ano, durante as disciplinas, os trabalhos envolveram a utilização de NoC como plataforma de comunicação.

Inicialmente a investigação das características de tempo real em sistemas embarcados ainda não previa a utilização de NoCs como plataforma. Durante o ano 2003, as publicações na literatura já apontavam as NoCs como alternativa para comunicação em sistemas embarcados. Dessa forma passamos a utilizar NoC como plataforma para sistemas embarcados voltados para aplicações de tempo real. Neste ano foram publicados alguns trabalhos iniciais que investigavam NoCs (CORRÊA 2003a) e tempo real (CORRÊA 2003b) (GERVINI 2003) em sistemas embarcados.

Em 2004, houve o prosseguimento das pesquisas em relação às NoCs, com a análise do custo de roteadores (CORRÊA 2004b) (OYAMADA 2004) e com o início das pesquisas do impacto do posicionamento dos núcleos no desempenho da rede (CORRÊA 2004a).

No intuito de aprofundar o estudo de NoCs para avaliar o seu impacto em aplicações de tempo real, teve início em novembro de 2004 um estágio no Departamento *Architecture des Systèmes Intégrés & Micro-électronique* (ASIM) do Laboratório de Informática (LIP6) da Universidade Pierre et Marie Curie (*Université Paris 6*), onde havia sido desenvolvida a NoC SPIN. A idéia inicial era trabalhar com a DSPIN

(*Distributed* SPIN). Entretanto, devido a interesses do ASIM, o trabalho desenvolvido lá foi portar uma aplicação multi-tarefas de descompressão de um fluxo de imagens MJPEG (*Motion* JPEG) em um MP-SoC. Esse estágio terminou no segundo semestre de 2005. Como resultado desse estágio, utilizou-se a aplicação estudada (MJPEG) como exemplo para a avaliação da metodologia proposta neste trabalho.

Em 2006 foi apresentada a proposta de tese, que propunha investigar as implicações dos requisitos de aplicações de tempo real em sistemas embarcados baseados em redes em chip. A partir da sugestão da banca, passamos a investigar novas características da NoC que poderiam ser adaptadas no intuito de atender as restrições temporais desse tipo de aplicação.

Buscou-se então, aplicar na NoC, técnicas de redes de interconexão tradicionais no intuito de atender os requisitos temporais das aplicações. As estratégias avaliadas foram a de incremento de prioridades por envelhecimento e a de descarte de pacotes antigos.

8.2 Perspectivas Futuras

A fase de análise dos resultados foi uma etapa árdua em virtude da quantidade de parâmetros envolvidos. Uma possibilidade de melhorar isso é alterar o mecanismo atual, semi-automático, de macros e arquivos de execução em lote, para uma alternativa com maior grau de automatização.

Em relação ao espaço de projeto, uma possibilidade que se pode vislumbrar é em relação ao mecanismo de controle de fluxo preemptivo em substituição aos canais virtuais. Esse mecanismo também permite a seletividade de tráfego como os canais virtuais, mas, estes últimos são mais adequados para redes maiores. No caso destas redes maiores, uma investigação que pode ser feita é em relação à clusterização da rede, em redes menores, onde o mecanismo de preempção dos *buffers* ainda seria competitivo.

Outra questão que precisa ser mais explorada é o suporte ao requisito de *jitter*. Isso é importante quando as aplicações são em seqüência de dados, pois neste tipo de aplicação, em geral, a exigência de *jitter* é mais rígida.

O trabalho aqui apresentado buscou resolver problemas dentro do espaço de projeto de NoCs em sistemas embarcados para aplicações de tempo real. Este espaço de projeto é bastante amplo e com certeza existem muitas outras questões importantes a serem resolvidas. Entretanto, esperamos que este trabalho contribua para resolver algumas delas, bem como, existe a intenção de se continuar trabalhando nesta linha de pesquisa.

Concluindo esta tese, destaca-se a qualidade da pós-graduação em computação da UFRGS principalmente em relação aos seus integrantes, que foram de vital importância para o desenvolvimento deste trabalho. Deve-se ressaltar também o apoio financeiro da Universidade Federal do Rio Grande do Norte (UFRN) e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), sem o qual, a realização deste trabalho não teria sido possível.

REFERÊNCIAS

ANDRIAHANTENAINA, A. et al. SPIN: a scalable, packet switched on-chip micro-network. In: DESIGN AUTOMATION AND TEST ON EUROPE, DATE, 2003, Munich. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p. 70–73.

ARM CORPORATION. **AMBA 2.0 specification**. Disponível em: <http://www.arm.com/armtech/AMBA_Spec>. Acesso em: mar. 2007.

ASCIA, G.; CATANIA, V.; PALESI, M. Multi-objective mapping for mesh-based NoC architectures. In: INTERNATIONAL CONFERENCE ON HARDWARE/SOFTWARE CODESIGN AND SYSTEM SYNTHESIS, CODES, 2004. **Proceedings...** [S.l.: s.n.], 2004. p. 182–187.

ASHRAF, F.; ABD-EL-BARR, M.; AL-TAWIL, K. Introduction to routing in multicomputer networks. **ACM Computer Architecture News**, New York, v.26, n.5, p. 14–21, Dec. 1998.

BAINBRIDGE, W. J.; FURBER, S. B. CHAIN: a delay insensitive chip area interconnect. **IEEE Micro,** Los Alamitos, v. 142, n. 4, p. 16–23, Sept. 2002.

BANERJEE, N.; VELLANKI, P.; CHATHA, K. S. A power and performance model for network-on-chip architectures. In: DESIGN AUTOMATION AND TEST ON EUROPE, DATE, 2004. **Proceedings...** Washington: IEEE Computer Society, 2004. p. 1250–1255.

BEIGNE, E. et al. An asynchronous NoC architecture providing low latency service and its multi-level design framework. In: INTERNATIONAL SYMPOSIUM ON ASYNCHRONOUS CIRCUITS AND SYSTEMS, ASYNC, 11., 2005. **Proceedings...** [S.l.]: IEEE, 2005. p. 54–63.

BENINI, L.; DE MICHELI, G. Networks on chips: a new SoC paradigm. **IEEE Computer**, [S.l.], v. 35, n. 1, p. 70–78, Jan. 2002.

BERGAMASCHI, R. A. et al. Automating the design of SoCs using cores. **IEEE Design & Test of Computers**, Los Alamitos, v. 18, n. 5, p. 32–45, Sept. 2001.

BJERREGAARD, T.; SPARSO, J. Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip. In: NORDIC EVENT IN ASIC DESIGN NORCHIP, 2004. **Proceedings...** [S.l.: s.n.], 2004.

BJERREGAARD, T.; SPARSO, J. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In: DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 2005. **Proceedings...** Washington: IEEE Computer Society, 2005. p. 1226–1231.

BJERREGAARD, T.; MAHADEVAN, S. A survey of research and practices of Network-on-chip. **ACM Computer Surveys**, New York, v. 38, n. 1, p. 1–51, Mar. 2006.

BOLOTIN, E. et al. QNoC: QoS architecture and design process for network on chip. **Journal of Systems Architecture: the EUROMICRO Journal**, New York, v.50, n.2–3, p. 105–128, 2004.

BRIÃO, E. W. **Levantamento e comparação de métodos e técnicas para exploração de espaço de projeto em sistemas baseados em redes-em-chip**. 2005. 51 f. Trabalho Individual (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

CORRÊA, E. F.; CARDOZO, R da S.; COTA, E.; BECK, A. C.; WAGNER, F. R.; CARRO, L.; SUSIN, A. A.; LUBASZEWSKI, M. Testing the wrappers of a network on chip: a case study. In: IEEE LATIN AMERICAN TEST WORKSHOP, LATW, 2003, Natal, Brazil. **Proceedings…** [S.l.]: IEEE Computer Society, 2003a. p. 159–163.

CORRÊA, E. F.; GERVINI, A. I.; WAGNER, F. R.; CARRO, L.; NETTO, J. C.; PEREIRA, C. E. Modelagem e geração de sistema operacional de tempo real para sistemas embarcados. In: WORKSHOP DE TEMPO REAL, WTR, 2003, Natal, Brasil. **Proceedings**… Natal: SBC, 2003b. p. 03–10.

CORRÊA, E. F.; BASSO, E. W.; WILKE, G. R.; WAGNER, F. R.; CARRO, L. The implications of real-time behavior in networks-on-chip architectures. In: IFIP WORKING CONFERENCE ON DISTRIBUTED AND PARALLEL EMBEDDED SYSTEMS, DIPES, 2004. **Proceedings…** Boston: Kluwer Academic Publishers, 2004a. p. 307–316.

CORRÊA, E. F.; ZEFERINO, C. A. ; CARDOZO, R. da S.; SUSIN, A. A.; WAGNER, F. R.; CARRO, L. A heterogeneous router for networks-on-chip. In: WORKSHOP IBERCHIP, 10., 2004, Cartagena. **Memorias**. [S.l.: s.n.], 2004b. p. 1–6.

CORRÊA, E. F.; SILVA; L. A. P.; BASSO, E. W.; WAGNER, F. R.; CARRO, L. Cost analysis for QoS applications in NoCs. Submetido ao IFIP - International Conference on Very Large Scale Integration, VLSI-SoC, 2007a.

CORRÊA, E. F.; SILVA; L. A. P.; WAGNER, F. R.; CARRO, L. Fitting the router characteristics in NoCs to meet QoS requirements. Submetido ao Symposium on Integrated Circuits and Systems Design, SBCCI, 2007b.

DALL'OSSO, M. et al. Xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs. In: INTERNATIONAL CONFERENCE ON COMPUTER DESIGN, ICCD, 2003. **Proceedings...** [S.l.]: IEEE Computer Society, 2003. p. 536–539.

DALLY, W. J. Express cubes: improving the performance of *k*-ary *n*cube interconnection networks. **IEEE Transactions on Computers**, New York, v.40, n.9, p. 1016–1023, Sept. 1991.

DALLY, W.; TOWLES, B. Route packets not wires: on-chip interconnection networks. In: DESIGN AUTOMATION CONFERENCE, DAC, 2001. **Proceedings...** Los Alamitos, IEEE Computer Society, 2001. p. 684–689.

DALLY, W. J.; TOWLES, B. **Principles and practices of interconnection networks**. San Francisco: Morgan Kaufmann, 2004. 550 p.

DICK, R. P.; RHODES, D. L.; WOLF, W. TGFF: task graphs for free. In: INTERNATIONAL. WORKSHOP ON HARDWARE/SOFTWARE CODESIGN, CODES/CASHE, 1998. **Proceedings…** Washington: IEEE Computer Society, 1998. p. 97–101.

DRAGON LEMON. **Dragon Lemon placement tool**. Disponível em: <http://www.inf.ufrgs.br/~renato/dragonlemon>. Acesso em: mar. 2007.

DUATO, J. **Interconnection networks**: an engineering approach. Revised printing. Amsterdam: Morgan Kaufmann, 2003. 600 p.

FARINES, J.-M.; FRAGA, J. da S.; OLIVEIRA, R. S. de. **Sistemas de tempo real**. São Paulo: IME-USP, 2000. 201 p.

FELICIJAN, T.; BAINBRIDGE, W. J.; FURBER, S. B. An asynchronous low latency arbiter for Quality of Service (QoS) applications. In: IEEE INTERNATIONAL CONFERENCE ON MICROELECTRONICS, ICM, 2003, Cairo. **Proceedings…** [S.l.]: IEEE Computer Society, 2003. p. 123–126.

FELICIJAN, T.; FURBER, S. B. An asynchronous on-chip network router with quality-of-service (QoS) support. In: IEEE INTERNATIONAL SOC CONFERENCE, SOCC, 2004, Santa Clara, CA. **Proceedings...** [S.l.]: IEEE Computer Society, 2004. p. 274–277.

FITZPATRICK, T. System verilog for VHDL users. In: DESIGN AUTOMATION AND TEST ON EUROPE, DATE, 2004. **Proceedings...** Washington: IEEE Computer Society, 2004. p. 1334–1341.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability**: a guide to the theory of NP-completeness. San Francisco: W. H. Freeman, 1979. 340 p.

GENKO, N. et al. A novel approach for network on chip emulation. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2005, Kobe. **Proceedings...** Los Alamitos: IEEE Computer Society, 2005. p. 2365–2368.

GERVINI, A. I.; CORRÊA, E. F.; CARRO, L.; WAGNER, F. R. Avaliação de desempenho, área e potência de mecanismos de comunicação em sistemas embarcados. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, SEMISH, 30., 2003, Campinas, Brasil. **Proceedings...** Campinas: SBC, 2003. p. 211–223.

GLOVER, F.; TAILLARD, E; DE WERRA, D. A user's guide to the tabu search. **Annals of Operations Research**, [S. l.], v. 41, p. 3–28, 1993.

GOOSSENS, K.G.W. et al. A design flow for application-specific networks on chip with guaranteed performance to accelerate SoC design and verification. In: DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 2005. **Proceedings...** [S.l.: s.n.], 2005. p. 1182–1187.

GUERRIER, P.; GREINER, A. A generic architecture for on-chip packet-switched interconnections. In: DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 2000. **Proceedings...** [S.l.: s.n.], 2000. p. 250–256.

HEMANI, A. et al. Lowering power consumption in clock by using globally asynchronous locally synchronous design style. In: DESIGN AUTOMATION CONFERENCE, DAC, 1999. **Proceedings...** New York: ACM Press, 1999. p. 873–878.

HENNESSY, J. L.; PATTERSON, D. A. **Arquitetura de computadores**: uma abordagem quantitativa. 3.ed. Rio de Janeiro: Campus, 2003. 827 p.

HSIEH, C. T.; PEDRAM, M. Architectural energy optimization by bus splitting. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v. 21, n. 4, p. 408–414, Apr. 2002.

HU, J.; MARCULESCU, R. Energy-aware mapping for tile-based NoC architectures under performance constraints. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE, ASP-DAC, 2003. **Proceedings…** [S.l.: s.n.], 2003a. p. 233–239.

HU, J.; MARCULESCU, R. Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures. In: DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 2003. **Proceedings...** [S.l.: s.n.], 2003b. p. 688–693.

HU, J.; MARCULESCU, R. Application-specific buffer space allocation for network-on-chip router design. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, ICCAD, 2004. **Proceedings...** [S.l.: s.n.], 2004a. p. 354–361.

HU, J.; MARCULESCU, R. Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints. In: DESIGN AUTOMATION AND TEST ON EUROPE, DATE, 2004. **Proceedings...** Washington: IEEE Computer Society, 2004b. p. 234–239.

HUNG, W. et al. Thermal-aware IP virtualization and placement for network-on-chip architecture. In: INTERNATIONAL CONFERENCE ON COMPUTER DESIGN, ICCD, 2004. **Proceedings...** [S.l.: s.n.], 2004. p. 430–437.

IBM. **The CoreConnect™ bus architecture**. Disponível em: <http://www.ibm.com/chips/techlib/techlib.nsf/productfamilies/CoreConnect_Bus_Architecture>. Acesso em: mar. 2007.

ITRS. **ITRS 2005 - International technology roadmap for semiconductors**. Disponível em: <http://public.itrs.net>. Acesso em: mar. 2007.

JANTSCH, A.; TENHUNEN, H. **Networks on chip**. Boston: Kluwer Academic Publishers, 2003. 303 p.

JANTSCH, A.; LAUTER, R.; VITKOVSKI, A. Power analysis of link level and end-to-end data protection on networks on chip. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2005, Kobe. **Proceedings...** Los Alamitos: IEEE Computer Society, 2005. p. 1770–1773.

KAHN, G. The semantics of a simple language for parallel programming. In: IFIP CONGRESS, 1974. **Proceedings...** [S.l.: s.n.], 1974. p. 471–475.

KARIM, F.; NGUYEN, A.; DEY, S. On-chip communication architecture for OC-768 network processors. In: DESIGN AUTOMATION CONFERENCE, DAC, 2001, Las Vegas. **Proceedings...** New York: ACM Press, 2001.

KARIM, F.; NGUYEN, A.; DEY, S. An interconnect architecture for networking systems on chips. **IEEE Micro**, [S.l.], v. 22, n. 5, p. 36–45, Sept. 2002.

KEATING, M.; BRICAUD, P. **Reuse methodology manual**: for system-on-a-chip designs. Norwell, MA, USA: Kluwer Academic Publishers, 1998. 224 p.

KEUTZER, K. et al. System-level design: orthogonalization of concerns and platform-based design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits**, [S.l.], v.19, n.12, p.1523–1543, Dec. 2000.

KIRKPATRICK, S.; GELATT JUNIOR, C. D.; VECCHI, M. P. Optimization by Simulated Annealing. **Science**, [S.l.], v.220, n.4598. p. 671–680, 1983.

KOPETZ, H.; VERÍSSIMO, P. Real-time and dependability concepts. In: MULLENDER, S. (Ed.). **Distributed Systems**. New York: Addison-Wesley, 1993. p. 411–446.

KREUTZ, M. E. **Método para a otimização de plataformas arquiteturais para sistemas multiprocessados heterogêneos**. 2005a. 171 f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

KREUTZ, M.; MARCON, C.; CALAZANS, N.; SUSIN, A. A. Energy and latency evaluation of NoC topologies. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2005, Kobe. **Proceedings...** Los Alamitos: IEEE Computer Society, 2005b. p. 5866–5869.

KUMAR, S. et al. A network on chip architecture and design methodology. In: SYMPOSIUM ON VERY LARGE INTEGRATION SCALE, VLSI, 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p. 105–112.

KUMAR, S. On packet switched network on chip communication. In: JANTSCH, A.; TENHUNEN, H. (Ed.). **Networks on chip**. Boston: Kluwer Academic Publishers, 2003. p. 85–106.

LEE, E. A. Model-driven development-from object-oriented design to actor-oriented design. In: WORKSHOP ON SOFTWARE ENGINEERING FOR EMBEDDED SYSTEMS, 2003. **Proceedings...** [S.l.: s.n.], 2003.

LEI, T.; KUMAR, S. A two-step genetic algorithm for mapping task graphs to a network on chip architecture. In: EUROMICRO SYMPOSIUM ON DIGITAL SYSTEMS DESIGN, DSD, 2003. **Proceedings...** [S.l.: s.n.], 2003.

LI, J.; MUTKA, M. W. Priority based real-time communication for large scale wormhole networks. In: INTERNATIONAL SYMPOSIUM ON PARALLEL PROCESSING, 1994. **Proceedings...** Washington: IEEE Computer Society, 1994. p. 433–438.

LI, J.; MUTKA, M. W. Real-time virtual channel flow control. **Journal of Parallel and Distributed Computing**, Orlando, v.32, n.1, p. 49–65, Jan. 1996.

LIANG, J.; SWAMINATHAN, S.; TESSIER, R. aSoC: a scalable, single-chip communication architecture. In: INTERNATIONAL CONFERENCE ON PARALLEL ARCHITECTURES AND COMPILATION TECHNIQUES, PACT, 2000, Philadelphia. **Proceedings...** Washington: IEEE Computer Society, 2000. p. 37–46.

LYONNARD, D. et al. Automatic generation of application specific architectures for heterogeneous multiprocessor system-on-chip. In: DESIGN AUTOMATION CONFERENCE, DAC, 2001, New Orleans. **Proceedings…** New York: ACM Press, 2001. p. 518–523.

MADSEN, J. et al. Network-on-chip modeling for system-level multiprocessor simulation. In: INTERNATIONAL REAL-TIME SYSTEMS SYMPOSIUM, RTSS, 2003. **Proceedings...** Washington: IEEE Computer Society, 2003. p. 265–274.

MARCON, C.; BORIN, A.; SUSIN, A.; CARRO, L.; WAGNER, F. Time and energy efficient mapping of embedded applications onto NoCs. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE, ASP-DAC, 2005, Shanghai. **Proceedings…** [S.l.: s.n.], 2005a. p. 33–38

MARCON, C.; CALAZANS, N.; MORAES, F.; SUSIN, A.; REIS, I.; HESSEL, F. Exploring NoC mapping strategies: an energy and timing aware technique. In: DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 2005. **Proceedings...** [S.l.: s.n.], 2005b. v.1, p. 502–507.

MELLO, A. et al. **Evaluation of routing algorithms on mesh based NoCs**. 2004. Technical Report. Faculdade de Informática – PUCRS, Porto Alegre.

MILLBERG, M. et al. The Nostrum backbone - a communication protocol stack for networks on chip. In: INTERNATIONAL CONFERENCE ON VERY LARGE INTEGRATION SCALE DESIGN, VLSI DESIGN, 2004. **Proceedings...** [S.l.: s.n.], 2004. p. 693–696.

MORAES, F. et al. A low area overhead packet-switched networks on chip: architecture and prototyping. In: INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION, VLSI-SoC, 2003, Darmstadt. **Proceedings...** [S.1.: s.n.], 2003. p. 318–323.

MORAES, F. et al. HERMES: an infrastructure for low area overhead packet-switching networks on chip. **Integration: the VLSI Journal**, Amsterdam, v.38, n.1, p. 69–93, Oct. 2004.

MURALI, S.; DE MICHELI, G. Bandwidth-constrained mapping of cores onto NoC architectures. In: DESIGN AUTOMATION AND TEST IN EUROPE, DATE, 2004. **Proceedings…** [S.1.: s.n.], 2004a. p. 896–901.

MURALI, S.; DE MICHELLI, G. SUNMAP: a tool for automatic topology selection and generation for NoCs. In: DESIGN AUTOMATION CONFERENCE, DAC, 2004. **Proceedings...** [S.l.: s.n.], 2004b. p. 914–919.

MURALI, S.; BENINI, L.; DE MICHELLI, G. Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE, ASP-DAC, 2005. **Proceedings…** [S.l.: s.n.], 2005. p. 27–32.

NI, L. M.; McKINLEY, P. K. A survey of wormhole routing techniques in direct networks. **IEEE Computer**, Los Alamitos, v.26, n.2. p. 62–76, Feb. 1993.

NOLLET, V. et al. Centralized run-time resource management in a network-on-chip containing reconfigurable hardware tiles. In: DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 2005. **Proceedings...** [S.l.: s.n.], 2005. p. 234–239.

OCP. **Open core protocol specification – release 1.0**. USA, 2001. 202 p.

OGRAS, U.; MARCULESCU, R. Energy and performance-driven NoC communication architecture synthesis using a decomposition approach. In: DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.352–357

OPNET. **OPNET Technologies**. Disponível em: <http://www.opnet.com>. Acesso em: mar. 2007.

OST, L. C. **Redes intrachip parametrizáveis com interface padrão para síntese em hardware**. 2004. 116 f. Dissertação (Mestrado em Ciência da Computação) – PUCRS, Porto Alegre,

OST, L. et al. MAIA - a framework for networks on chip generation and verification. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE, ASP-DAC, 2005. **Proceedings…** [S.l.: s.n.], 2005. p. 49–52.

OYAMADA, M. S.; GERVINI, A. I.; CORRÊA, E. F.; WAGNER, F. R.; CARRO, L. Análise de desempenho e consumo de potência na comunicação interprocessos em software embarcado**.** In: WORKSHOP IBERCHIP, 10., 2004, Cartagena. **Memorias**. [S.l.: s.n.], 2004. p. 108–109.

PALMA, J. C. S. et al. Evaluating the impact of data encoding techniques on the power consumption in networks-on-chip. In: SYMPOSIUM ON EMERGING VLSI TECHNOLOGIES AND ARCHITECTURES, ISVLSI, 2006. **Proceedings...** Washington: IEEE Computer Society, 2006. p. 426–427.

PANDE, P. P. et al. Design of a switch for network on chip applications. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2003. **Proceedings**… [S.l.]: IEEE Computer Society, 2003. p. 217–220.

POP, R.; KUMAR, S. **A survey of techniques for mapping and scheduling applications to network on chip systems**. 2004. Technical Report. Jönköping University, Jönköping, Sweden.

RAMAMRITHAM, K.; STANKOVIC J. A.; ZHAO, W. Distributed scheduling of tasks with deadlines and resources requirements. **IEEE Transactions on Computers**, New York, v. 38, n. 8, p. 1110–1123, Aug. 1989.

RAGHUNATHAN, V.; SRIVASTAVA, M. B.; GUPTA, R. K. A survey of techniques for energy efficient on-chip communication. In: DESIGN AUTOMATION CONFERENCE, DAC, 2003, New Orleans. **Proceedings…** New York: ACM Press, 2003. p. 900–905.

RHEE, C.-E.; JEONG, H.-Y.; HA, S. Many-to-many core-switch mapping in 2-D mesh NoC architectures. In: INTERNATIONAL CONFERENCE ON COMPUTER DESIGN, ICCD, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.438–443.

RIJPKEMA, E.; GOOSSENS, K. G. W.; WIELAGE, P. A router architecture for networks on silicon. In: WORKSHOP ON EMBEDDED SYSTEMS, PROGRESS, 2., 2001. **Proceedings…** [S.1.: s.n.], 2001. p. 181–188.

RIJPKEMA, E. et al. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In: DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 2003. **Proceedings…** [S.1.: s.n.], 2003. p. 350–355.

ROSTISLAV, D. et al. An asynchronous router for multiple service levels networks on chip. In: INTERNATIONAL SYMPOSIUM ON ASYNCHRONOUS CIRCUITS AND SYSTEMS, ASYNC, 11., 2005. **Proceedings...** [S.l.]: IEEE, 2005. p. 44–53.

SAASTAMOINEN, I. et al. Proteo interconnect IPs for networks-on-chip. In: IP BASED SoC DESIGN, 2002, Grenoble. **Proceedings…** [S.1.: s.n.], 2002.

SGROI, M. et al. Addressing the system-on-chip interconnect woes through communication-based design. In: DESIGN AUTOMATION CONFERENCE, DAC, 2001. **Proceedings...** New York: ACM Press, 2001. p. 667–672.

SHAFRANOVICH, Y. **Common format and MIME type for comma-separated values (CSV) files**: RFC 4180. [S.l.]: Internet Engineering Task Force, Network Working Group, 2005.

SHIN, D.; KIM, J. Power-aware communication optimization for networks-on-chips with voltage scalable links. In: INTERNATIONAL CONFERENCE ON HARDWARE/SOFTWARE CODESIGN AND SYSTEM SYNTHESIS, CODES, 2004. **Proceedings...** [S.l.: s.n.], 2004. p. 170–175.

SIGÜENZA-TORTOSA, D.; NURMI, J. Proteo: a new approach to network-on-chip. In: IASTED INTERNATIONAL CONFERENCE ON COMMUNICATION SYSTEMS AND NETWORKS, CSN, 2002. **Proceedings...** [S.l.: s.n.], 2002.

SIGÜENZA-TORTOSA, D.; AHONEN, T.; NURMI, J. Issues in the development of a practical NoC: the Proteo concept. **Integration: the VLSI Journal**, Amsterdam, v.38, n.1, p. 95–105, Oct. 2004.

SMIT, L. et al. Run-time mapping of applications to a heterogeneous reconfigurable tiled system on chip architecture. In: INTERNATIONAL CONFERENCE ON FIELD-PROGRAMMABLE TECHNOLOGY, FPT, 2004. **Proceedings...** [S.l.: s.n.], 2004. p. 421–424.

SMITH, S. An asynchronous GALS interface with applications. In: IEEE WORKSHOP ON MICROELECTRONICS AND ELECTRON DEVICES, 2004. **Proceedings**… Los Alamitos: IEEE Computer Society, 2004. p. 41–44.

SOCLIB. **SoCLib project home page**. Disponível em: <http://www.soclib.fr/trac/dev>. Acesso em: mar. 2007.

SRINIVASAN, K.; CHATHA, K. ISIS: a genetic algorithm based technique for custom on-chip interconnection network synthesis. In: INTERNATIONAL CONFERENCE ON VERY LARGE INTEGRATION SCALE DESIGN, VLSI DESIGN, 2005. **Proceedings...** [S.l.: s.n.], 2005. p. 623–628.

STANKOVIC, J. Misconceptions about real-time computing: a serious problem for next-generation systems. **IEEE Computer**, New York, p.10–19, Oct. 1988.

STANKOVIC J. A.; RAMAMRITHAM, K. What is predictability for real-time systems? **The Journal of Real-Time Systems**, Norwell, USA, v.2, n.4, p. 247–254, Nov. 1990.

STERGIOU, S. et al. Xpipes lite: a synthesis oriented design library for networks on chips. In: DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 2005. **Proceedings...** [S.l.: s.n.], 2005. p. 1188–1193.

SYSTEMC. **SystemC**: Welcome. Disponível em: <http://www.systemc.org>. Acesso em: mar. 2007.

TAMIR, Y.; FRAZIER, G. L. Dynamically-allocated multi-queue buffers for VLSI communication switches. **IEEE Transactions on Computers**, Washington, v.41, n.6, p. 725–737, June 1992.

TEWKSBURRY, S. K.; UPPULURI M.; HORNAK, L. A. Interconnections/micro-network for integrated circuits. In: GLOBAL TELECOMMUNICATIONS CONFERENCE, GLOBECOM, 1992. **Proceedings...** [S.l.: s.n.], 1992. p. 180–186.

VARATKAR, G.; MARCULESCU, R. Communication-aware task scheduling and voltage selection for total systems energy minimization. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, ICCAD, 2003. **Proceedings...** [S.l.: s.n.], 2003. p. 510–517.

VSI ALLIANCE. **Virtual component interface standard – OCB 2 1.0**. Los Gatos, 2000. 71 p.

WANG, H.-S. et al. Orion: a power-performance simulator for interconnection networks. In: ACM/IEEE INTERNATIONAL SYMPOSIUM ON MICROARCHITECTURE, 2002. **Proceedings…** Washington: IEEE Computer Society, 2002. p. 294–305.

WIKLUND, D.; LIU, D. SoCBUS: switched network on chip for hard real time systems. In: INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM, IPDPS, 2003. **Proceedings...** Washington: IEEE Computer Society, 2003. p. 78–85.

WOLF, W. **Computers as components**: principles of embedded computing system design. San Francisco, CA: Morgan Kaufmann, 2001. 662 p.

XU, J. et al. A case study in networks-on-chip design for embedded video. In: DESIGN AUTOMATION AND TEST ON EUROPE, DATE, 2004. **Proceedings...** Washington: IEEE Computer Society, 2004. p. 770–775.

ZEFERINO, C. A. **Redes-em-chip**: arquiteturas e modelos para avaliação de área e desempenho. 2003a. 242 f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

ZEFERINO, C. A.; SUSIN, A. A. SoCIN: a parametric and scalable network-on-chip. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 2003. **Proceedings…** Washington, DC: IEEE Computer Society, 2003b. p. 169–174.

ZEFERINO, C. A.; KREUTZ, M. E.; SUSIN, A. A. RASoC: a router soft-core for networks-on-chip. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE, DATE, 2004. **Proceedings…** Washington, DC: IEEE Computer Society, 2004a. v.3, p. 30198–30203.

ZEFERINO, C. A.; SANTO, F.G.M.E.; SUSIN, A. A. ParIS: a parameterizable interconnect switch for networks-on-chip. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 2004, **Proceedings…** New York: ACM Press, 2004b. p. 204–209.