UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Marcelo Daniel Berejuck

REDE INTRA-CHIP COM PREVISIBILIDADE DE LATÊNCIA PARA USO EM SISTEMAS DE TEMPO REAL

Florianópolis

2015

Marcelo Daniel Berejuck

REDE INTRA-CHIP COM PREVISIBILIDADE DE LATÊNCIA PARA USO EM SISTEMAS DE TEMPO REAL

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Doutor em Engenharia de Automação e Sistemas.
Orientador: Prof. Antônio Augusto Fröhlich, Dr.

Florianópolis

2015

Catalogação na fonte elaborada pela biblioteca da
Universidade Federal de Santa Catarina

Berejuck, Marcelo Daniel
Rede Intra-chip com previsibilidade de latência para uso em Sistemas de Tempo Real. / Marcelo Daniel Berejuck ; orientador, Antônio Augusto Fröhlich - Florianópolis, SC, 2015.
195 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Inclui referências

1. Engenharia de Automação e Sistemas. 2. Network-on-Chip. 3. Sistemas de Tempo Real. I. Fröhlich, Antônio Augusto. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. III. Título.

Marcelo Daniel Berejuck

REDE INTRA-CHIP COM PREVISIBILIDADE DE LATÊNCIA PARA USO EM SISTEMAS DE TEMPO REAL

Esta Tese foi julgada aprovada para a obtenção do Título de "Doutor em Engenharia de Automação e Sistemas", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis, 31 de Março 2015.

Prof. Rômulo Silva de Oliveira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Antônio Augusto Fröhlich, Dr.
Presidente

Prof. Fabiano Hessel, Dr., PUC-RS

Prof. Luigi Carro, Dr., UFRGS

Prof. Cesar Albenes Zeferino, Dr., UNIVALI

Prof. Carlos Barros Montez, Dr., UFSC

Prof. Eduardo Augusto Bezerra, Dr., UFSC

Dedico este trabalho para minhas amadas filhas, Mariana e Gabriela.

AGRADECIMENTOS

"Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas Graças a Deus, não sou o que era antes."(*Marthin Luther King*). Agradeço ao Professor Dr. Antônio Augusto Fröhlich por confiar em mim e ter aceito ser meu Orientador. Em especial, agradeço à minha família, pelo apoio durante estes anos de estudos em que tiveram que abdicar de minha presença para que eu pudesse concluir o Doutorado. Como disse Simone de Beauvoir, *"Todas as vitórias ocultam uma abdicação"*.

A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.

Arthur Schopenhauer

RESUMO

Sistemas intra-chip ou SoC (acrônimo de *Systems-on-Chip*) com múltiplas unidades de processamento heterogêneas têm sido usados pela indústria de silício como solução para disponibilizar o desempenho demandado pelas modernas aplicações multimídia. No entanto, a integração de um crescente número de unidades de processamento especializadas em um mesmo SoC impõem um desafio para os mecanismos de interconexão de tais sistemas, que agora são obrigados a lidar com um grande número de fluxos de comunicação muito distintos, com requisitos de latência e largura de banda também muito distintos. Como solução, a indústria do silício vem utilizando redes intra-chip ou NoCs (acrônimo de *Networks-on-Chip*) com previsibilidade de latência para interligar tais unidades de processamento neste tipo de SoC. No entanto, muitas aplicações neste domínio obteriam mais benefícios de uma NoC que pudesse otimizar a utilização dos recursos para fluxos multimídia que toleram variações razoáveis na Qualidade de Serviço (em inglês *Quality of Service* - QoS). Será demonstrado ao longo deste documento que muitos destes sistemas são concebidos em torno de alguns fluxos de comunicações de tempo real muito restritos, que precisam ser tratados dentro de limites de tempo rigorosos (muitas vezes envolvendo comandos para o controle do sistema ou tarefas de sinalização de estado do sistema) e um grande número de fluxos multimídia menos restritos, que toleram variações muito maiores na latência e na largura de banda. A estratégia de projeto de NoCs predominante na literatura para produzir interconexões para SoCs de tempo real baseia-se no mapeamento dos requisitos de comunicação de tarefas em tempo real (por vezes implementadas em hardware como componentes de propriedade intelectual dedicados) para os recursos de rede disponíveis em fases iniciais do projeto. Este mapeamento, no entanto, muitas vezes é realizado considerando um cenário de pior caso e, portanto, resulta em reserva de recursos que poderiam ser dinamicamente realocados para outros fluxos. Embora adequado para aplicações críticas de tempo real, esta estratégia resulta na má utilização de silício para aplicações multimídia com taxa de bits variável. Neste contexto, esta Tese apresenta uma rede que oferece previsibilidade na latência de pior caso, denominada de RTSNoC, e que foi projetada para o cenário no qual o sistema possui poucos fluxos de comunicação com restrições de tempo real rígidas, relacionados ao controle do sistema, e muitos fluxos de comunicação

multimídia com restrições de tempo real menos rígidas. Na verdade, uma latência de pior caso para tais fluxos multimídia pode ser determinado em tempo de projeto, de modo que os projetistas poderiam de fato modelar os fluxos de multimídia como sendo de tempo real suave (ou *soft real-time*), cuja degradação é proporcional à quantidade de fluxos flui ao longo da rede. No entanto, uma vez que a estratégia de roteamento adotada na RTSNoC não usa qualquer tipo de reserva de recursos em tempo de execução, neste documento tais fluxos serão designados como sendo "fluxos de melhor esforço" (em inglês *Best Effort*-BE). A arquitetura da rede proposta baseia-se na intercalação de *flits* provenientes de diferentes fluxos em um mesmo canal de comunicação entre roteadores da rede, de modo que cada *flit* contém informações de roteamento. Os resultados experimentais demonstram que a latência média de fluxos com variação na taxa de bits injetados na rede proposta é, em média, mais baixa do que em redes que executam a reserva de recursos e estão operando com 80% de tráfego oferecido. Além disso, é demonstrado analiticamente que fluxos de comunicação de tempo real projetados considerando o valor da latência de pior caso da rede sempre atenderão as restrições associadas a tarefas de tempo real rígidas, de modo que não há perda no limite de tempo para a execução de tais tarefas devido à contenção de recursos na rede.
Palavras-chave: *Network-on-Chip*. Sistemas de Tempo Real. *Field Programmable Gate Array*.

ABSTRACT

Systems-on-Chip (SoC) with multiple heterogeneous processing units have been used by the silicon industry as means to deliver the performance required by modern multimedia applications. However, the integration of an increasing number of specialized processing units poses a challenge on the interconnection mechanisms in such systems, which are now required to handle a large number of very distinctive communication flows, with very distinct latency and bandwidth requirements. As a solution, the silicon industry has been using predictable Networks-on-Chip (NoC) to interconnect components in this kind of SoC. Nevertheless, many applications in this domain would profit better from a NoC that could optimize the utilization of resources for multimedia flows that tolerate reasonable variations in the Quality-of-Service(QoS). In this document will be shown that several systems have been conceived around a few very strict real-time communication flows (often involving control or signalling tasks) and a large number of less strict multimedia flows that tolerate much larger variations in latency and bandwidth. In this context, current real-time NoC designs fall short at making good use of hardware resources as they rely on worst-case resource reservation. The prevailing design strategy to produce interconnects for such SoCs relies on mapping the communication requirements of real-time tasks (sometimes implemented in hardware as dedicated IPs) to available network resources at early design stages. This mapping, however, is often performed considering a worst-case scenario and therefore results in the reservation of resources that could otherwise by dynamically reallocated to other flows. Although adequate for critical real-time applications, this strategy results in poor silicon utilization for variable-bit-rate multimedia applications. This document presents a *Worst-Case Latency* (WCL) of a network called RTSNoC that was designed with the aforementioned scenario in mind: few hard real-time control flows and many best-effort multimedia flows. Indeed, a worst-case latency for such best-effort flows can be determined at design-time, so designers could indeed model the multimedia flows as soft real-time (or QoS) flows whose degradation is proportional to the amount of streams flowing across the chip. However, since the routing strategy does not use any kind of resource reservation at run-time, this document will refers to those flows as best-effort. The proposed NoC architecture is based on the interleaving of flits from

different flows in the same communication channel between routers, so each flits carries along routing information. Experimental result showed that the worst-case latency in RTSNoC network was, in average, lower than NoC that adopt resources reservation, when those networks are working over 80% of offered load. Furthermore, it was analytically demonstrated that the communication flows related to real-time designed considering the worst-case latency of the network always will achieve the restrictions related to hard real-time tasks. It means that there is no deadline lost for the execution of those tasks due to the contention of network resources.
Keywords: Network-on-Chip. Real-time System. Field Programmable Gate Array.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AMBA	*Advanced Microcontoller Bus Architecture*
AXI	*Advanced eXtensible Interface*
BE	*Best Effort*
CMOS	*Complementary Metal-Oxide-Semiconductor*
CRC	*Cyclic Redundancy Check*
DMA	*Direct Memory Access*
EDF	*Earliest Deadline First*
FIFO	*First-In, First-Out*
Flit	*Flow control unit*
FPGA	*Field Programmable Gate Array*
GT	*Guaranteed Throughput*
GS	*Guaranteed Services*
I/O	*Input/Output*
LUT	*Look-Up Table*
MPSoC	*Multiprocessor System-on-a-Chip*
NoC	*Network-on-Chip*
OCP	*Open Core Protocol*
PABX	*Private Automatic Branch eXchange*
Phit	*Physical unit*
QoS	*Quality-of-Service*
RM	*Rate Monotonic*
SAF	*Store-and-Forward*
SoC	*Systems-on-a-Chip*
TDM	*Time Division Multiplexing*
VCT	*Virtual Cut-Through*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*
RAM	*Random Access Memory*
ROM	*Read Only Memory*

SUMÁRIO

PARTE I

INTRODUÇÃO

1 INTRODUÇÃO

Este capítulo tem por objetivo estabelecer o contexto em que esta Tese foi escrita, iniciando com uma visão geral sobre sistemas embarcados compostos por múltiplas unidades de processamento e implementados em uma única pastilha de silício (*System-on-Chip* - SoC), fazendo uso de redes intra-chip (ou *Network-on-Chip* - NoC). Posteriormente, a caracterização do problema de pesquisa, a motivação e os objetivos definidos para a investigação científica que resultaram nesta Tese são apresentados. As principais contribuições deste trabalho são então resumidas e o Capítulo é finalizado por uma breve introdução aos assuntos tratados nos próximos capítulos que compõem o documento.

1.1 PRÓLOGO

O aumento na densidade dos circuitos integrados tem permitido aos projetistas implementar múltiplos processadores de diferentes tipos em um mesmo chip. São sistemas completos em uma única pastilha de silício, normalmente conhecidos como *Systems-on-a-Chip*. Um SoC é composto por um conjunto de componentes de hardware que são interconectados por meio de uma estrutura de canais denominada arquitetura de comunicação ou rede de interconexão. Os componentes de hardware são referenciados na literatura pelo termo "núcleo"(do inglês *core*(GUPTA; ZORIAN, 1997)), sejam eles processadores ou blocos de hardware dedicados que realizam funções específicas.

Existem diferentes formas de interconexão passíveis de serem utilizadas em SoCs, entre elas as redes intra-chip (em inglês *Network-on-Chip* - NoC). Uma NoC consiste do arranjo de canais ponto-a-ponto, chaveados por roteadores e compartilhados pelos núcleos do sistema, que suportam a comunicação entre os núcleos por meio da comutação por circuitos ou por pacotes. Para Benini (BENINI LUCA; DE MICHELI, 2006), as NoCs estruturam as conexões globais e, portanto suas propriedades elétricas são otimizadas e bem controladas, permitindo a distribuição de sinais de clock de forma mais adequada para sistemas mais complexos. Além disso, para aquele autor, as NoCs permitem o paralelismo nas comunicações. Para Dally (DALLY; TOWLES, 2001) estas redes facilitam a modularidade devido ao uso de interfaces padronizadas que melhoram o reuso de núcleos consolidados.

Os sistemas embarcados atuais baseados no conceito de SoC de-

mandam um alto nível de integração entre núcleos de processamento que trocam mensagens entre si. Geralmente tais sistemas apresentam requisitos de tempo real relacionados à execução de tarefas computacionais, sendo conhecidos como SoCs de tempo real. Uma técnica empregada para verificar se os tempos de execução de tarefas de um sistema embarcado serão cumpridos é baseada na análise estática da escalonabilidade, que levam em consideração o tempo de pior caso de execução (em inglês *Worst-Case Execution Time* - WCET). O WCET precisa considerar o tempo de computação de cada tarefa em execução no processador, mas também deve levar em conta a latência de comunicação quando algumas das tarefas sob análise requerem dados vindos de fora do processador.

Apesar de vantagens do uso de NoC, destacadas por autores como Dally (DALLY WILLIAM. J.; TOWLES, 2001) e Benini (BENINI LUCA; DE MICHELI, 2006), nem todas as NoCs são adequadas para serem utilizadas na interconexão de SoCs de tempo real. Apenas as redes que oferecem mecanismos de previsibilidade de latência na comunicação são passíveis de uso em tais sistemas, pois a latência na comunicação através da rede precisa ser conhecida e considerada nos testes de escalonabilidade realizados em sistemas de tempo real. As NoCs apresentadas até o momento e que oferecem previsibilidade da latência baseiam-se em técnicas que envolvem reserva de recursos, como multiplexação no tempo e chaveamento de circuitos. Porém, para autores como Kees Goossens *et al.*(GOOSSENS; DIELISSEN; RADULESCU, 2005) a reserva de recursos implica no aumento no uso de recursos de silício. Para lidar com esta relação custo-benefício entre previsibilidade de latência e uso de recursos de silício, em geral, a reserva de recursos é feita apenas para os fluxos de comunicação que precisam ter sua latência conhecida, sendo os demais fluxos tratados com a política de melhor esforço.

Adotar a reserva de recursos apenas para fluxos de comunicação prioritários em uma NoC é uma estratégia cabível para sistemas embarcados de tempo real. Muitos destes sistemas embarcados podem ser modelados como tendo uma menor quantidade de tarefas que exigem tempo de execução rígidos (ou *hard*) e uma quantidade maior de tarefas com tempo de execução menos rígidos (ou *soft*)[1] (PALOPOLI; ABENI; BUTTAZZO; CONTICELLI; DI NATALE, 2000). Consequentemente, em uma NoC dedicada para tais sistemas são esperados poucos fluxos de

[1]Uma tarefa é dita como sendo *soft* se a perda de seu *deadline* causa apenas degradação do desempenho do sistema, porém sem impossibilitar o seu funcionamento normal. Para este tipo de tarefa, a utilidade do resultado pode ou não ser útil após o seu *deadline*. (LIU, 2000).

comunicação com limites rígidos de latência na comunicação entre dois pontos quaisquer da rede, do que fluxos de comunicação menos rígidos em relação à latência. Assim, os fluxos em uma NoC que são relacionados às tarefas com tempo de execução menos rígidos podem ser tratados como sendo fluxos "de melhor esforço" (BE). O aumento da latência na comunicação através de uma NoC para fluxos BE não implica em prejuízo no funcionamento normal do sistema, porém pode degradar a qualidade dos serviços aos quais estas tarefas estão relacionadas. Esta situação passa a ser mais crítica quando a NoC opera em alto tráfego e os fluxos BE apresentam taxa de bits variável, pois neste caso a latência média dos fluxos cresce de modo exponencial (VAN DEN BRAND et al., 2007).

1.2 CENÁRIO ATUAL DE NOCS PARA SOC DE TEMPO REAL

Uma NoC para uso em SoCs de tempo real precisa oferecer garantias de largura de banda e latência para a comunicação entre dois núcleos quaisquer conectados a ela. Existem essencialmente duas maneiras de atingir tais objetivos: através do controle de taxa de transmissão ou com o chaveamento de circuitos. A rede SoCBUS (SATHE; WIKLUND; LIU, 2004) e a rede 4S (WOLKOTTE et al., 2006) são exemplos de NoC que utilizam chaveamento de circuitos. Nestas redes, uma vez que as conexões são estabelecidas, elas não podem ser compartilhadas, de modo que as garantias de banda e vazão são facilmente atingidas. No entanto, esta técnica apresenta baixa utilização de recursos da rede (MLA, 2012), pois fluxos de comunicação tratados como BE não podem compartilhar os recursos reservados.

A técnica TDM (acrônimo de *Time-Division Multiplexing*) é outra opção de chaveamento de circuitos empregada por redes como a AEthereal (GOOSSENS; DIELISSEN; RADULESCU, 2005), dAelite (GOOSSENS; DIELISSEN; RADULESCU, 2012) e Nostrum (JANTSCH et al., 2004). Nesta técnica, um período de tempo é "fatiado" em tempos menores, chamados de *time slots*. Assim, diversos pacotes podem ser transmitidos em um mesmo canal físico através da alocação de um *time slot* para cada pacote que compete pelos mesmos recursos na rede. Esta técnica resolve o problema do compartilhamento de recursos da rede, porém ainda deixa sem solução o aumento significativo na latência para os fluxos tratados com a política de melhor esforço sob alto tráfego na NoC, apontado por (VAN DEN BRAND et al., 2007).

Como alternativa ao chaveamento de circuitos, a técnica de ro-

teadores não bloqueantes com controle de taxa de transmissão pode ser vista na rede MANGO (BJERREGAARD; SPARSO, 2006). Nela, os pacotes são localmente arbitrados nos roteadores, sendo armazenados em *buffers* e o controle da taxa realizado para que as garantias de vazão e latência dos fluxos que circulam na rede sejam atingidas. Esta técnica apresenta a mesma preocupação que a técnica TDM utilizada em outras redes: consumo de recursos de silício. A rede prevê dois roteadores trabalhando em paralelo para cada nodo da rede, sendo um para fluxos de comunicação com prioridade, chamado de `GS` (acrônimo de *Guaranteed Service*) e outro para fluxos tratados com a política de melhor esforço (em inglês *Best Effort* - BE). O controle de taxa é realizado apenas nos roteadores `GS`, sendo que os fluxos de comunicação tratados como BE são roteados com uma técnica de roteamento baseada na origem para realizar o roteamento dos pacotes de dados sem estabelecimento de conexão, de acordo com o caminho de roteamento definido no cabeçalho do pacote. Portanto, esta técnica também deixa sem solução o aumento significativo na latência para os fluxos tratados com a política de melhor esforço sob alto tráfego na NoC.

As soluções apresentadas até agora resolvem o problema da previsibilidade de latência para os fluxos na rede, seja utilizando roteadores não-bloqueantes com controle de taxa de transmissão ou chaveamento de circuitos. Porém, o custo de silício para garantir tal previsibilidade a todos os fluxos da rede em ambas as técnicas é alto (HANSSON et al., 2009), o que limita o conhecimento das latências apenas para fluxos prioritários. Devido ao alto custo de recursos, fluxos BE possuem menor prioridade e são tratados em segundo plano, geralmente adotando-se uma política de melhor esforço. É importante lembrar que muitos sistemas embarcados de tempo real possuem mais aplicações que podem ser tratadas com políticas de melhor esforço sendo executadas do que aplicações de tempo real *hard* e a percepção de qualidade pode ser relevante para alguns daqueles sistemas.

Por fim, todas as técnicas citadas precisam mapear os requisitos de comunicação das tarefas de tempo real para os recursos de rede disponíveis em fases iniciais do projeto. Este mapeamento, no entanto, muitas vezes é realizado considerando um cenário de pior caso, resultando em reserva de recursos que poderiam ser dinamicamente realocados para outros fluxos (GOOSSENS; DIELISSEN; RADULESCU, 2005)(ATIENZA et al., 2008). Embora adequado para aplicações críticas de tempo real, esta estratégia resulta na má utilização de silício para aplicações multimídia com taxa de bits variável (VAN DEN BRAND et al., 2007).

1.3 PROBLEMA DE PESQUISA E SUA MOTIVAÇÃO

A maioria das soluções de NoC para SoCs de tempo real são baseadas em técnicas que envolvem reserva de recursos, como TDM e chaveamento de circuitos. Algumas destas redes são voltadas para fluxos de tempo real *hard*, como a rede Æthereal e a rede Nostrum; enquanto outras, como a SoCBUS, são mais adequadas para fluxos de melhor esforço.

Em todas as redes apresentadas até o momento, todo tráfego precisa ser considerado previamente, durante o projeto da rede, para que se possa garantir as restrições temporais dos fluxos de tempo real *hard*. Consequentemente, para tais redes a quantidade de fluxos que devem receber mecanismos de garantia de latência de comunicação na rede deve ser conhecida previamente; o aumento na quantidade de fluxos com mecanismos de garantia de latência implica no aumento do uso de recursos de silício para o SoC de tempo real. Uma vez que os fluxos prioritários foram atendidos, os demais fluxos são roteados na rede com o uso de mecanismos de melhor esforço e sem garantias de atendimento de limites de latência, que em geral consomem menos recursos de silício para serem implementados. Neste cenário, o problema de pesquisa é projetar uma NoC que garanta uma latência de pior caso (em inglês *Worst-Case Latency* - WCL) para fluxos de tempo real *hard* e que melhore a latência média para fluxos de melhor esforço na rede, relacionados à aplicações multimídia, sem comprometer as garantias de WCL dos fluxos de tempo real *hard*.

1.4 PROBLEMA DE PESQUISA E CONTRIBUIÇÕES

O trabalho apresentado neste documento trata o problema de pesquisa, apresentado na seção anterior, focando em SoCs para uso em sistemas embarcados nos quais o número de fluxos associados a aplicações de tempo real *hard* é menor do que o número de fluxos relacionados a aplicações de multimídia com taxa de bits variável. Além disso, os sistemas embarcados alvo são aqueles que demandam por algum nível de qualidade de serviço na qual a qualidade esteja relacionada aos fluxos pertencentes a aplicações multimídia. Tais sistemas embarcados estão no contexto do problema de pesquisa, no qual é necessário que se garanta o WCL para fluxos de tempo real *hard* e seja melhorado a latência média para os demais fluxos que influenciam na qualidade de alguns serviços não prioritários ao sistema.

Neste contexto, a hipótese levantada nesta Tese pode ser definida da seguinte forma: é possível estabelecer uma estrutura de comunicação para SoCs de tempo real, baseada no conceito de *Network-on-Chip*, sem reserva de recursos, capaz de oferecer previsibilidade na latência de pior caso na comunicação entre dois pontos quaisquer da rede. Além disso, o valor médio da latência para os fluxos tratados como melhor esforço deve ser menor quando comparado com uma rede com reserva de recursos que também ofereça tratamento de melhor esforço para fluxos sem restrições rígidas de tempo real. Assim, o objetivo do trabalho descrito neste documento foi pesquisar e propor uma estrutura de comunicação, baseada no conceito de *Network-on-Chip*, capaz de confirmar a hipótese apontada nesta Tese.

Uma análise elaborada para determinar com exatidão o momento em que é feito um acesso à rede exige que sejam identificados todos os parâmetros que possam afetar tal acesso. Os nodos de uma rede com múltiplos processadores estão competindo por recursos de rede, e o resultado dessa concorrência depende do histórico de execução de todos os nodos envolvidos. Isto torna muito difícil a análise e, possivelmente, insolúvel em alguns casos. Outra questão que deve ser observada é a interferência local de instruções executadas no mesmo nodo de processamento. Por ser uma questão local, ela pode ser detectada e tratada pelo processador em questão, portanto espera-se que seja incorporada na análise de execução do processador. Por estes motivos, latências na comunicação entre os núcleos e a rede, ou latências internas aos núcleos, não são tratadas neste trabalho.

Para atender o objetivo proposto, foi necessário organizar o trabalho de pesquisa em etapas, aqui relacionadas como objetivos específicos:

1. Estabelecer um modelo de comunicação em sistemas de tempo real que utilizam NoC;
2. Realizar um levantamento bibliográfico para caracterizar NoCs descritas na literatura que ofereçam garantias de latência, analisando seus pontos fortes e limitações;
3. Projetar uma NoC que atenda ao objetivo geral proposto;
4. Implementar a NoC proposta e validar seu funcionamento;
5. Comparar o desempenho da rede proposta com a rede Æthereal, baseada em reserva de recursos; e
6. Avaliar o uso da NoC em um sistema de tempo real.

Para a definição do modelo de comunicação foram utilizados como referências a definição de latência na comunicação em uma rede genérica proposta por William J. Dally e Brian Towles (DALLY WILLIAM. J.; TOWLES, 2001), e o trabalho de Kees Goossens *et al.* (MLA, 2012), que decompõem a latência observada em uma NoC em diferentes tipos de transação de dados na rede, como escrita e leitura de dados ou blocos de dados. A revisão bibliográfica foi realizada sobre os principais artigos relacionados à NoCs que utilizam roteadores não-bloqueantes e chaveamento de circuitos para garantir previsibilidade de latência.

Conforme será descrito ao longo deste documento, o modelo de NoC proposto nesta Tese foi um mecanismo que permite a intercalação de *flits*[2] pertencentes a diversos fluxos que competem por um mesmo canal de saída em um roteador da rede. Para isso, foram adicionadas informações de roteamento em cada *flit* de um pacote qualquer que trafegue na rede. A avaliação do modelo proposto foi realizada de três formas distintas. A primeira delas foi a realização de avaliações teóricas sobre o modelo proposto, comparando os resultados obtidos com a rede Æthereal. A rede Æthereal foi usada como referência porque é baseada na técnica TDM, reconhecida como um mecanismo de comunicação que oferece previsibilidade (GOOSSENS; DIELISSEN; RADULESCU, 2005), (JANTSCH et al., 2004). Além disso, ela também foi escolhida por ser a NoC mais citada em artigos científicos segundo a Biblioteca Digital do IEEE[3], dentre as NoCs estudadas. Uma síntese destes estudos é apresentado no Capítulo 2.

A segunda forma de avaliação do modelo proposto foi a síntese da NoC em um FPGA. Neste caso, também foram feitas comparações das latências medidas na rede implementada no FPGA com valores teóricos da rede Æthereal. Por fim, foi feita a avaliação do modelo da NoC proposta em um SoC de sistema embarcado tipo PABX (acrônimo de *Private Automatic Branch eXchange*) produzido pela empresa Intelbras S.A.. Segundo o fabricante, um sistema tipo PABX possui poucos fluxos relacionados a tarefas de tempo real *hard* e muitos fluxos relacionados a tarefas multimídia com taxa de bits variável, que podem ser tratados como BE.

Assim sendo, as principais contribuições desta Tese são:

- Um modelo do fluxo de tráfego em SoCs de tempo real;
- A implementação de uma NoC com garantia de previsibilidade

[2] Acrônimo de *FLow control unIT*.

[3] Acrônimo de *Institute of Electrical and Electronic Engineers*. Disponível em: http://ieeexplore.ieee.org/

de latência sem a necessidade de reserva de recursos da rede, e que apresenta uma latência média para fluxos com taxa de bits variável menor do que redes que trabalham com reserva de recursos;

- Manter o valor médio da latência na rede baixo para fluxos BE, considerando que a rede opera com um tráfego oferecido superior a 80%;

- Todos os fluxos possíveis entre dois núcleos quaisquer da rede possuem valores de latência de pior caso fixos, e estes valores são independentes de possíveis variações na taxa de bits dos outros fluxos que competem pelos mesmos recursos da rede.

1.5 ORGANIZAÇÃO DO DOCUMENTO

Esta Tese foi dividida em capítulos, sendo que cada um deles foi sub-dividido em seções e subseções. O primeiro capítulo, no qual esta seção está inserida, tratou da contextualização do problema tratado nesta Tese.

O segundo capítulo apresenta o resultado do levantamento bibliográfico. Foram selecionadas sete *Network-on-Chip* que são focadas em aspectos relevantes para sistemas de tempo real, como a previsibilidade de latência na comunicação, e que apresentam as técnicas descritas anteriormente. Apenas as sete *Network-on-Chip* relacionadas neste documento trataram de técnicas diretamente relacionadas à fluxos de tempo real. Por exemplo, as redes SoCIN (ZEFERINO; SUSIN, 2003) e Hermes (MORAES et al., 2004) apresentam soluções para garantir qualidade de serviços, porém não fazem referência direta aos problemas relacionados à tempo real rígido e sim à qualidade de serviços (em inglês *Quality-of-Service* - QoS), o que sob o ponto de vista da comunidade científica que atua com Tempo Real, não são sinônimos. A rede SoCIN implementou três mecanismos de QoS: (*i*) chaveamento de circuitos, (*ii*) canais virtuais e (*iii*) envelhecimento de pacotes (BEREJUCK; ZEFERINO, 2009). Por sua vez, a rede Hermes também teve implementações de QoS: (*i*) chaveamento de circuitos, (*ii*) escalonamento baseado em prioridades fixas, (*iii*) escalonamento baseado em prioridades dinâmicas e (*iv*) escalonamento baseado em taxas (MELLO, 2006). Por focarem apenas em QoS e não em aplicações explicitamente de tempo real, tais redes não foram incluídas neste documento.

Assim, para as redes analisadas, o capítulo descreve a topologia

de cada rede, os aspectos funcionais dos seus roteadores e as técnicas adotadas por cada rede para garantir a latência dos fluxos. Além disso, propriedades importantes de cada rede são analisadas, como o tempo de espera, que está relacionado ao tempo que algumas redes necessitam para requisitar ou liberar recursos da rede e a latência que um fluxo pode sofrer numa comunicação entre dois roteadores quaisquer. O capítulo é finalizado com um resumo sobre as redes analisadas.

O terceiro capítulo descreve a análise da latência adicional que uma NoC pode oferecer ao tempo de execução de programas em núcleos de processamento conectados aos nodos da rede. São definidas naquele capítulo algumas terminologias e modelos necessários para que se possa descrever formalmente os fluxos de tráfego em um SoC baseado no uso de NoC, além de definir um modelo formal para estimar a latência de pior caso em uma NoC.

O quarto capítulo apresenta a arquitetura de uma rede-em-chip proposta nesta Tese. O capítulo começa tratando do conceito de intercalação de *flits* adotado na rede proposta, seguido da definição de escopo do trabalho que foi desenvolvido para esta Tese. Em seguida é descrito o desenvolvimento da NoC proposta e são apresentados detalhes e discussões a respeito das escolhas de projeto para a rede, como topologia, enlaces, arbitragem, chaveamento, rotamento, formato do pacote, etc.

O quinto capítulo apresenta os resultados experimentais baseados em experimentos realizados com redes implementadas com a NoC proposta, avaliando o impacto no uso de recursos de silício devido às técnicas adotadas para as redes implementadas. Finalizando o capítulo, são descritos os experimentos realizados para a avaliação de desempenho da NoC proposta, sob o ponto de vista da latência oferecida pela rede na comunicação entre nodos quaisquer da rede.

No sexto capítulo são apresentados os resultados obtidos com um estudo de caso no qual a rede foi avaliada experimentalmente em um SoC de tempo real, originalmente desenvolvido para uma aplicação industrial de central telefônica, chamada de SoC PABX. Este capítulo explica, em linhas gerais, o funcionamento do SoC PABX e apresenta os resultados obtidos nos experimentos realizados.

O sétimo capítulo apresenta as considerações finais sobre o trabalho realizado, destacando as contribuições da Tese e, finalizando este documento, é apresentado o Capítulo 8 que relaciona alguns temas para serem pesquisados futuramente.

PARTE II

TRABALHOS RELACIONADOS

2 TRABALHOS RELACIONADOS

A maioria dos projetos de NoC publicados na literatura foram planejados para oferecer suporte à fluxos de tráfego de melhor esfoço naquelas redes, e o foco é tipicamente a minimização da latência média dos fluxos e a maximização da utilização da largura de banda das redes. Uma NoC para ser aplicada em uma plataforma de tempo real precisa garantir, além da largura de banda, que a latência de pior caso seja conhecida para os fluxos na rede que estão relacionados à aplicações de tempo real, que estão sendo executadas nos nodos da rede. O que pôde ser observado na literatura foram NoCs que adotam técnicas que oferecem alguma forma de conexão fim-a-fim, de modo a oferecer tais garantias temporais. De um modo geral, tais técnicas estão relacionadas a roteadores não bloqueantes com controle de fluxo e chaveamento de circuitos, algumas vezes adotando mecanismos de Multiplexação por Divisão de Tempo (*Time Division Multiplexing* - TDM).

Este capítulo apresenta o resultado dos estudos feitos sobre sete *Network-on-Chip* que são focadas em aspectos relevantes para sistemas de tempo real, como a previsibilidade de latência na comunicação, e que apresentam as técnicas descritas acima. Este capítulo foi divido em seções, sendo que as primeiras sete seções apresentam aquelas NoCs, descrevendo a topologia de cada rede, os aspectos funcionais dos seus roteadores e as técnicas adotadas por cada uma delas para garantir a latência dos fluxos. Além disso, as seções destacam três propriedades de cada rede: (*i*) o tempo de espera, que está relacionado ao tempo que algumas redes necessitam para requisitar ou liberar recursos da rede; (*ii*) a latência que um fluxo pode sofrer numa comunicação entre dois roteadores quaisquer; e (*iii*) a vazão apresentada por cada rede. Cada seção possui uma subseção chamada de resumo, onde as principais características avaliadas na rede são destacadas, e outra subseção denominada de Evolução da "rede", onde são apresentados resumos dos principais artigos publicados até os dias de hoje, dando uma visão geral de que caminhos foram seguidos pelos autores de cada uma das redes descritas neste capítulo. Por fim, a Seção 2.8 finaliza este capítulo apresentando um resumo geral sobre as redes analisadas.

2.1 SOCBUS

A Rede SoCBUS é uma NoC baseada na técnica de Chaveamento de Circuitos e foi desenvolvida por Sathe e Wiklund (SATHE; WIKLUND; LIU, 2004) (WIKLUND; LIU, 2003). A Rede utiliza um procedimento baseado no envio de pacotes para estabelecer conexões ponto-a-ponto, ou seja, reservar circuitos através da rede. Quando a conexão é estabelecida, todos os recursos ao longo do caminho estabelecido ficam reservados apenas para esta conexão. O envio de dados pela conexão acontece em uma transação de quatro fases. Inicialmente, a conexão é estabelecida por um pacote enviado através da rede requisitando temporariamente os recursos necessários como roteadores e enlaces. Em seguida um pacote de aceite (em inglês *acknowledge*) ou recusa é enviado de volta. Se a requisição foi aceita, a conexão é mantida; caso contrário ela é desfeita. Neste ponto os dados já podem ser transferidos utilizando os recursos reservados. Por fim, após os dados terem chegado ao seu destino um pacote é enviado para trás, confirmando o recebimento dos dados e cancelando a requisição dos circuitos ao mesmo tempo, se necessário.

A rede SoCBUS é implementada como uma malha regular 2D utilizando roteadores com cinco portas, na qual quatro portas conectam roteadores vizinhos e uma porta é utilizada para um núcleo de processamento, conforme ilustra a Figura 1. Adaptadores de rede são implementados como adaptadores (ou *wrappers*) entre os núcleos de processamento e a rede. Eles são responsáveis pelo domínio de *clock* e o armazenamento de pacotes para a transmissão.

Os roteadores são conectados através de enlaces bidirecionais que oferecem linhas de dados e de controle de informação, conforme mostrado na Figura 2. Na versão apresentada pelos autores daquela rede, um total de dezenove sinais de conexão foram usados em cada direção. Dezesseis sinais carregam dados e requisições de roteamento de pacotes, um fio é usado para controle de encaminhamento e dois sinais são usados como controle reverso. O pacote de requisição de roteamento é composto por dezesseis bits, sendo oito bits utilizados como endereço de destino e os outros oito bits são considerados bits auxiliares. Um sistema de relógio Mesócrono[1] é utilizado nos enlaces para compensar problemas de *clock skew*, que é a diferença entre os tempos de chegada do sinal de relógio nos seus circuitos receptores. Os sinais de controle de encaminhamento e controle reverso gerenciam a

[1] Em um sistema Mesócrono, todos os relógios possuem a mesma frequência, mas não necessariamente a mesma fase (SODERQUIST, 2002)

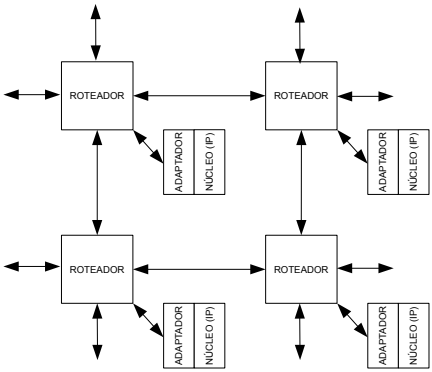


Figura 1 – Exemplo de uma rede SoCBUS com nós de roteamento, adaptadores e núcleos de processamento. Fonte: adaptado de (WIKLUND; LIU, 2003).

temporização nas transmissões. Os controles reversos indicam os sinais positivo e negativo de *acknowledgements* para cada negociação.

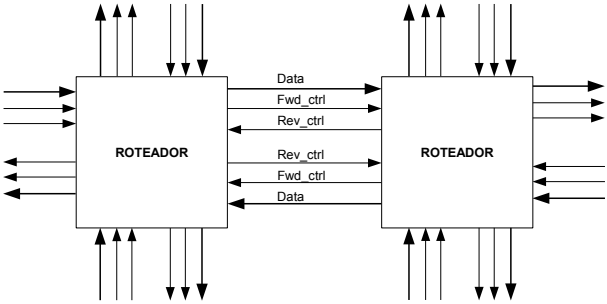


Figura 2 – Exemplo de uma interconexão entre roteadores na SoCBUS. Fonte: adaptado de (SATHE; WIKLUND; LIU, 2004).

A Figura 3 ilustra uma estrutura em blocos representando o fluxo de dados dentro de um roteador da rede SoCBUS. Cada canal do roteador é composto de uma máquina de estados finita de entrada, um módulo decodificador de endereços e uma máquina de estados finita de saída. O codificador de prioridade e o árbitro são únicos para cada roteador.

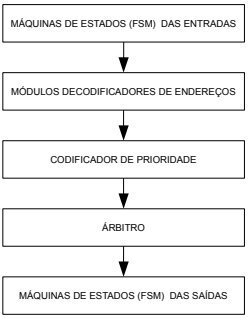


Figura 3 – Fluxo de dados em um roteador da rede SoCBUS. Fonte: adaptado de (SATHE; WIKLUND; LIU, 2004).

Durante o procedimento de requisição dos circuitos um algoritmo de busca por caminhos de menor tamanho é utilizado. A cada salto (ou em inglês *hop*) entre roteadores, o pacote de estabelecimento de circuito é roteado para um dos roteadores disponíveis, baseado no endereço de destino. Esta busca por roteadores é feita de modo *round-robin*. No caos de não haver roteadores disponíveis, é gerado um procedimento de falha, no qual as conexões previamente estabelecidas são desfeitas e uma nova tentativa de estabelecimento de conexão é tentada mais tarde. Após ter tido sucesso no estabelecimento da conexão, os dados são encaminhados do endereço de origem para o endereço de destino através dos roteadores e enlaces reservados, sem postergação e utilizando completamente a largura de banda da conexão.

2.1.1 Propriedades para análise de WCL

Quando uma conexão é estabelecida, ela passa a ser proprietária dos recursos alocados, como roteadores e enlaces, e por este motivo tem garantias de que os requisitos de tempo real serão atendidos. No entanto, recursos não são compartilhados entre conexões, resultando em uma baixa utilização de recursos na rede. é possível que uma requisição de conexão possa não ser atendida devido a falta de recursos. Tentativas de conexão sem sucesso levam a repetição do envio de pacote de solicitação de conexão, o que acaba por desperdiçar largura de banda. Este comportamento dinâmico pode causa flutuações no tempo de execução das tarefas de modo não preditivo, comprometendo as propriedades de

tempo real. A seguir são apresentadas algumas análises feitas sobre a rede SoCBUS. Por uma questão de simplificação, foi considerando que os pacotes da rede são compostos de apenas um *flit*.

Tempo de espera - Para que um pacote de dados atravesse a rede, é necessário que um pacote de estabelecimento de conexão seja enviado antes. Para que o pacote de estabelecimento de conexão atravesse a rede SoCBUS são necessários no mínimo quatro ciclos de relógio por roteador. Por outro lado, para que o pacote de aceite seja enviado de volta, é necessário apenas um ciclo de relógio por roteador. Portanto, para um caminho de h *hops*, o processo de estabelecimento de conexão $T_{wait;req} = 4.h + h$ ciclos de relógio. Para cada transação é necessário considerar um *overhead* de h ciclos para efetivar a desconexão quando a transação foi terminada, porém isto pode ser feito durante a mensagem de retorno gerada pelo núcleo de destino, de acordo com os autores da rede SoCBUS. O parâmetro $T_{wait;reply}$ é zero, uma vez que o circuito possui os recursos e não há espera para obter acesso à rede para envio da mensagem de resposta.

Latência - Para que um pacote atravesse um roteador da rede SoCBUS, uma vez que a conexão já foi estabelecida, é necessário um ciclo de relógio. Consequentemente, a latência total para que um pacote atravesse o caminho ponto-a-ponto é de $T_{latency} = h$; ou seja, é igual ao número de *hops* do caminho.

Vazão - De acordo com os seus autores, a vazão na rede SoCBUS é de um pacote por ciclo de relógio, durante o tempo em que a rede estiver com a conexão ativa. Para a transmissão de n pacotes, o tempo necessário para a transferência do bloco de dados é de n ciclos de relógio.

2.1.2 Resumo

Como a rede SoCBUS possui uma vazão alta ela parece ser uma solução efetiva para a transferência de grandes quantidades de dados, como no caso de aplicações de *data streaming*. Naquela Rede as conexões, quando estabelecidas, não compartilham recursos e as garantias de tempo real são facilmente atingidas. No entanto, esta abordagem pode resultar em uma baixa utilização de recursos da rede.

Para transferência de pacotes simples, o custo (*overhead*) na negociação de recursos é alto quando comparado com a latência na transmissão do pacote. No entanto, o sucesso em uma conexão não é garantido, fazendo da SoCBUS uma solução inadequada para plataformas de tempo real rígidas (*hard real-time*).

2.1.3 Evolução da rede SoCBUS

O projeto da rede SoCBUS iniciou-se no ano de 1999, e o primeiro artigo com uma descrição completa sobre seu funcionamento foi publicado no ano de 2003 (WIKLUND; LIU, 2003). No ano de 2004, os autores (WIKLUND; SATHE; LIU, 2004) publicaram um artigo intitulado "*Network on chip simulations for benchmarking*", no qual destacam a importância do estabelecimento de testes que validem de modo eficiente o projeto de uma NoC.

Aquele artigo descreve o ambiente de projeto e simulação desenvolvido no projeto da rede SoCBUS. Este ambiente foi usado como base para desenvolver os procedimentos necessários de *benchmarking* para avaliar o desempenho das redes por eles desenvolvidas. Os autores apresentaram dois modelos de *benchmarking* que foram usados na SoCBUS. Tais exemplos mostram como o ambiente de simulação pode ser utilizado para descobrir os gargalos da rede (em inglês *bottleneck*). Como foi mencionado anteriormente, a rede SoCBUS é vista por muitos autores como uma rede não adequada para sistemas de tempo real rígidos. Porém, os autores (WIKLUND; SATHE; LIU, 2004) demonstram como a rede SoCBUS pode ser adequada para sua utilização em tais sistemas. Basicamente eles demonstraram, com o uso de *benchmarking*, a importância e impacto da programação em nível de sistema no desempenho de uma interconexão intra-chip. Ou seja, a importância do conhecimento prévio do comportamento total do SoC em tempo de projeto para que a técnica de chaveamento de circuitos possa ser adequadamente utilizada em sistemas de tempo real rígidos.

No ano de 2005, os autores (WIKLUND; EHLIAR; LIU, 2005) apresentaram um artigo intitulado "*Design of an Internet Core Router Using the SoCBUS Network on Chip*". O objetivo do artigo era apresentar os resultados do desenvolvimento de uma solução de SoC para um roteador de 16 portas TCP/IP para redes Ethernet de 10 Gbps. O projeto foi baseado em uma rede SoCBUS para comunicação interna entre entre as unidades funcionais. Aqueles autores realizaram simulações com base em três classes de tráfego mostrado um desempenho de pico em cerca de 14-16 Gbps por porta para os tipos de fluxo de tráfego comuns encontrados numa rede Ethernet comercial, e cerca de 2,6 Gbps por porta para um tráfego de tamanho mínimo de pacotes, sem descartar pacotes. As simulações mostraram ainda os fatores limitantes no projeto, tornando possível aumentar o desempenho daquele roteador através de reprojeto do SoC.

O artigo intitulado "*Low-Latency, Low-Area Overhead and High*

Throughput NoC Architecture for FPGA Based Computing System" e apresentado pelos autores (SHELKE; PATIL, 2014), mostra uma nova rede baseada no conhecimento prévio que aqueles autores tinham na rede SoCBUS. Naquele artigo os autores descrevem uma arquitetura de FPGA de código aberto baseada na rede SoCBUS com alterações, de modo que a rede apresenta baixa sobrecarga de área, alto desempenho e baixa latência na transmissão de pacotes em comparação com outros artigos da SoCBUS publicados anteriormente. A arquitetura foi otimizada para para um FPGA do fabricante Xilinx® e a NoC é capaz de operar em uma frequência de 305,573 MHz num FPGA da família Virtex-5 xc5vlx110t-3-ff1136 daquele fabricante. Os autores desenvolveram interfaces de rede baseadas no padrão genérico Wishbone de modo que IP (acrônimo de *Intellectual Property*) compatíveis podem ser ligados à NoC.

A Figura 4 mostra a linha do tempo dos artigos aqui apresentados.

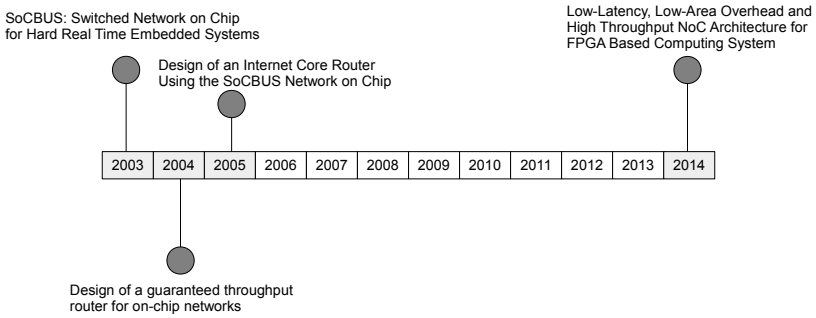


Figura 4 – Linha do tempo das publicações relacionadas com a rede SoCBUS.

2.2 4S *PROJECT*

A rede 4S (*Smart chipS for Smart Surroundings*) foi desenvolvida na Universidade de Twente (WOLKOTTE et al., 2006)(WOLKOTTE; RAUWERDA; SMIT, 2005)(KAVALDJIEV et al., 2005) e é uma rede puramente baseada em chaveamento de circuitos que foi desenvolvida para oferecer garantia de serviços. Ela implementa o chaveamento de fios

(*wires*) nos roteadores, estaticamente conectando linhas de entrada e saída, de modo similar às caixas de conexão (*switch boxes*) usadas em FPGAs. circuitos ponto-a-ponto são formados de modo estático, reservando porções das linhas de enlace para tais circuitos. O roteamento dos circuitos é pré-definido para a rede durante a inicialização da rede.

O conceito chave desta NoC é a organização dos enlaces em caminhos distintos. Os enlaces entre roteadores são divididos em porções com tamanhos fixos, chamados de *lanes*, conforme ilustra a Figura 5. O número e o tamanho de cada *lane* em um enlace é uma parâmetro de projeto e é fixado ao longo das *lanes* do sistema. Cada circuito ponto-a-ponto recebe um número de lanes, ficando assim com uma porção da largura de banda disponível. A Figura 5 mostra uma representação na qual o enlace foi dividido em quatro *lanes*.

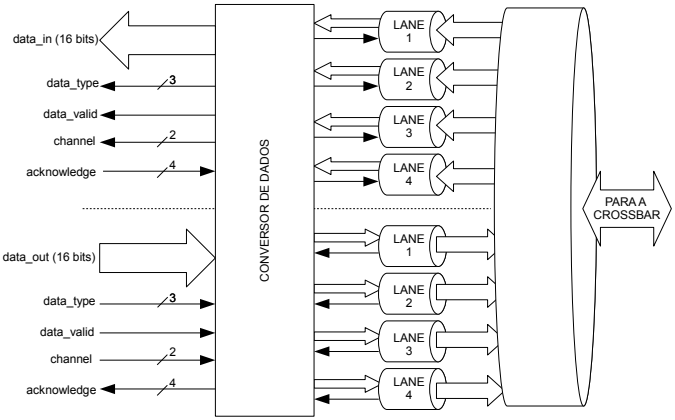


Figura 5 – Diagrama em blocos do enlace e conversor de dados da rede 4S. Fonte: adaptado de (WOLKOTTE et al., 2006).

Um roteador no sistema consiste basicamente de uma *crossbar* que faz o chaveamento das *lanes* de entrada para as respectivas *lanes* de saída. Ele é constituído por três módulos principais: o conversor de dados, a *crossbar* e o módulo de configuração da *crossbar*. tais módulos estão representados na Figura 6. Uma tabela de configuração é armazenada no roteador para prover informações sobre as conexões entre *lanes*. Tal tabela é representada na Figura 6 pelo bloco "Configuração da *Crossbar*". Uma unidade de controle de rede é conectada a cada roteador, de modo a configurar a *crossbar* com as conexões pré-definidas. Um componente conversor de dados conecta o roteador com o núcleo

de processamento, fazendo o papel de Adaptador de Rede.

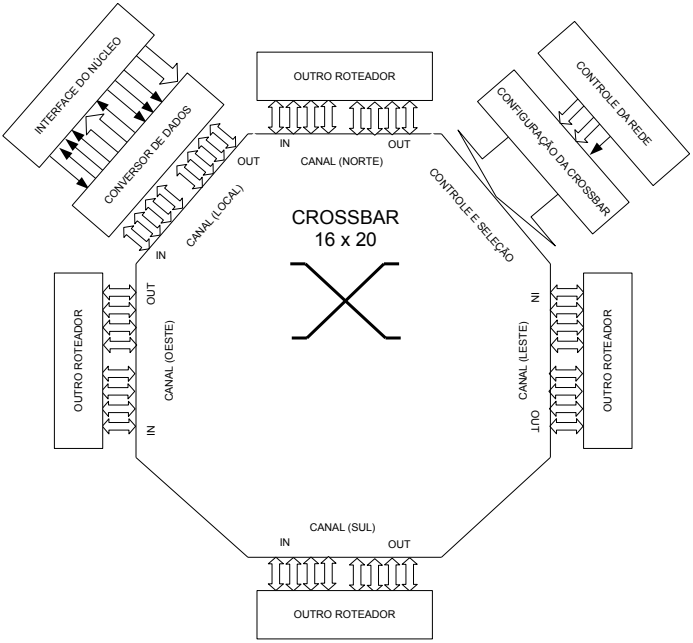


Figura 6 – Diagrama em blocos do roteador da rede 4S. Fonte: adaptado de (WOLKOTTE et al., 2006).

Ele é responsável por converter os pacotes de dados da interface do nodo de processamento em blocos de dados que se encaixam na largura das *lanes*, e realizando a operação inversa na outra extremidade do circuito. Além disso, é responsável pela adição de um cabeçalho com a largura de um *lane* para cada pacote de dados. As saídas do roteador são registradas, então roteadores tem um pipeline de um estágio e largura de um *lane*.

Para um pacote ser inserido na rede, ele é inicialmente particionado pelo conversor de dados em *phits* com a largura de um *lane*. Em um ciclo de relógio, um *phit* pode ir da saída de um roteador através de uma *lane* até outro roteador. *Lanes* são fios sem nenhum tipo de *pipeline*. Portanto, para um pacote atravessar um *hop* (roteador e enlace) são necessários tantos ciclos de relógio quantos forem os *phits* a serem enviados.

O controle fim-a-fim é implementado utilizando um esquema baseado em créditos. Para evitar *overflow* nos *buffers* entre pontos fim-a-fim de um circuito, um sinal de aceite (*acknowledge*) é enviado para trás para cada *lane* em separado através do uso de linhas dedicadas. Além disso, um mecanismo de contador é utilizado, o qual reflete a capacidade de armazenamento do *buffer* a cada extremidade do circuito. O sinal de *acknowledge* é usado como forma de crédito para dar informação sobre o espaço disponível deixado no nodo de destino .

2.2.1 Propriedades para análise de WCL

Redes baseadas na técnica de Chaveamento de Circuitos reservam circuitos para oferecer garantias de serviços. Neste caso, recursos como enlaces não são compartilhados. Cada conexão utiliza *lanes* específicas em um roteador pré-definido. Assim, apenas algumas conexões podem ser realizadas e a utilização dos circuitos é potencialmente baixa. Para dar suporte a conexões adicionais, *lanes* com mais fios seriam necessárias, o que aumentaria o custo de hardware.

Tempo de Espera - Depois que a configuração foi estabelecida na inicialização da rede, todos os recursos alocados estão dedicados aos circuitos que foram pré-definidos. Isto resulta em um tempo de espera nulo para acessar a rede. Portanto os parâmetros $T_{wait;req}$ e $T_{wait;reply}$ são zero.

Latência - O atraso que ocorre para um pacote atravessar um roteador tem dependência na profundidade do *buffer* do roteador (B). O tempo para um pacote atravessar um enlace depende do tamanho do pacote, da largura das *lanes* e do número de *lanes* que foram alocados para um circuito em específico. Para um pacote com tamanho de b bits, largura da *lane* L e uma quantidade de l *lanes* pertencentes a um circuito, $b/B.l$ ciclos de relógio são necessários para atravessar um enlace. Devido a serialização e encaminhamento imediato dos *phits* nos roteadores, os *phits* pode prosseguir sem ter de esperar pela chegada do pacote inteiro nos roteadores. A latência total fim-a-fim de um pacote através de um caminho com h *hops* é dada por $T_{latency} = h.B + b/B.l$ ciclos de relógio.

Vazão - O número de pacotes que podem ser injetados na rede por ciclos de relógio depende da quantidade de *phits* que um pacote pode ser particionado, b/L *phits*. Se l *lanes* são atribuídas ao circuito, então a a vazão será de $(b.l)/L$ bits por ciclo de relógio. Portanto, um bloco com n pacotes iria necessitar de $(n.L)/(b.l)$ ciclos.

2.2.2 Resumo

Para a NoC 4S, a latência e a vazão são altamente dependentes da largura das *lanes*, as quais são uma característica de projeto. Garantias de latência são trivialmente garantidas e a rede é adequada para a transferência de grandes quantidades de dados, como aplicações de *streaming*. No entanto, os recursos da rede não são compartilhados entre os circuitos, o que causa um sub-utilização de tais recursos.

2.2.3 Evolução da rede 4S Project

A primeira publicação sobre a rede 4S Project foi no ano de 2005. Naquele ano os autores (KAVALDJIEV et al., 2005) publicaram o artigo intitulado "*Energy Model of Networks-on-Chip and a Bus*" onde analisavam os gastos de energia entre uma NoC e um barramento de dados padrão (paralelo). Naquele mesmo ano os autores (WOLKOTTE; RAUWERDA; SMIT, 2005) publicaram o artigo "*An Energy-Efficient Reconfigurable Circuit-Switched Network-on-Chip.*" apresentando de modo formal o roteador da rede 4S Project.

No ano de 2006 os autores (WOLKOTTE et al., 2006) apresentaram o artigo intitulado "*Design of a guaranteed throughput router for on-chip networks*" no qual eles apresentaram o projeto de um protótipo de roteador com cinco entrada e cinco saídas. O diferencial alegado pelo autores estava na vazão garantida escalável no roteador GT (acrônimo de *Guaranteed Throughput*). O roteador é construído com um conjunto de blocos de hardware parametrizados e reusáveis, que constituem o roteador. O roteador proposto suporta chaveamento tipo *wormhole*[2] e consome uma área de silício de 0,1 mm^2 na tecnologia CMOS de 0.18 micron.

No artigo "*Fast, Accurate and Detailed NoC Simulations*" apresentado pelos autores (WOLKOTE et al., 2007) no ano de 2005, a rede 4S Project foi utilizada para descrever um método para simular um grande homogênea paralela e várias heterogêneas *Network-on-Chips* em um único dispositivo FPGA. Segundo seus autores, o método é adequado para sistemas paralelos onde longos ciclo de simulação e precisão de bits são necessários. Como estudo de caso, aqueles autores usaram uma versão da rede 4S Project que foi modelado e simulado em lin-

[2]No chaveamento *wormhole* um pacote é dividido em *flits*(acrônimo de *FLow control unIT*) que avançam pela rede em um modo *pipeline*, no qual os dados são enviados de forma serial e realizados em ciclos subsequentes de relógio.

guagem de programação SystemC. A mesma simulação realizada em SystemC foi implementada em um FPGA, o que permitiu aos autores do trabalho observar o comportamento da NoC sob uma grande variedade de padrões de tráfego. Comparado com a simulação realizada em SystemC, aqueles autores alegam ter conseguido um aumento de velocidade de 80-300, sem comprometer a precisão ciclo e nível de bits.

Os autores (SMIT et al., 2008) focaram em algoritmos e arquiteturas multi-core reconfiguráveis para *streaming* para aplicações de processamento digital de sinais (DSP) no artigo "*Multi-core Architectures and Streaming Applications*. Os autores utilizaram a rede 4S Project para definir a reconfiguração dinâmica de núcleos (ou *cores*) heterogêneos na arquitetura de um SoC. Aquela arquitetura um núcleo pode ser uma unidade programável em nível de *word*, uma unidade programável de propósito geral ou uma unidade de DSP ou microprocessador. Os núcleos do SoC foram interconectados por uma rede 4S Project reconfigurável. Segundo seus autores, a programabilidade individual dos núcleos permite que o sistema atinja múltiplos domínios de aplicações.

A Figura 7 mostra a linha do tempo dos artigos aqui apresentados.

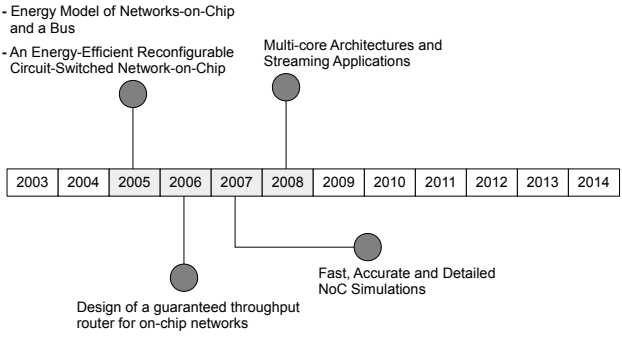


Figura 7 – Linha do tempo das publicações relacionadas com a rede 4S Project.

2.3 NOSTRUM

A Rede Nostrum foi apresentada por Jantsch e Millberg (JANTSCH et al., 2004) (MILLBERG et al., 2004) e é uma rede baseada em chaveamento de circuitos que utiliza canais virtuais como mecanismo para multiplexação. Ela suporta tanto fluxos com garantia de serviços (em inglês *Guaranteed Services* - GS) ou melhor esforço (em inglês *Best Effort* - BE). Para fluxos BE, os pacotes são roteados através da rede e a decisão de quando executar o roteamento é feita dinamicamente em cada roteador. Para fluxos GS, são utilizados canais virtuais (em inglês *Virtual Channels* - *VC*) com caminhos pré-definidos e que são armazenados em tabelas de roteamento (*loo-up tables*). Ela também usa o conceito de *Temporally Disjointed Networks* - *TDN*) e *Looped Containers* (LC) para facilitar o fluxos BE e GS.

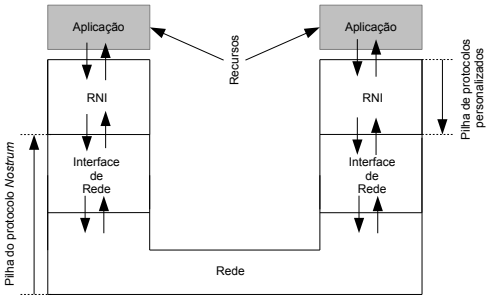


Figura 8 – Diagrama em blocos das interfaces da rede Nostrum. Fonte: adaptado de (JANTSCH et al., 2004).

O uso de TDNs garante a independência entre diferentes tipos de tráfego (GS e BE) e também entre diferentes canais virtuais (VC). Para facilitar o entendimentos do mecanismo, os TDNs podem ser vistos como diferentes colorações entre nodos vizinhos. Dois nodos vizinhos não podem ter a mesma cor e pacotes são transmitidos de nodos de um tipo específico de cor para nodos com diferentes cores. Durante uma fatia de tempo pré-fixado (em inglês *time slot*) os pacotes movem-se simultaneamente de um roteador para outro, mudando de uma cor para outra, tal que pacotes que residem em roteadores de diferentes cores em determinados períodos de tempo. Desta forma, a multiplexação por divisão no tempo (em inglês *Time Division Multiplexing* - *TDM*) é alcançada. O número máximo de de diferentes TDNs que podem existir

numa rede depende da topologia da rede e do número de estágios de *pipeline* do roteador. Estes fatores são importantes na determinação do número de TDNs porque roteadores vizinhos e também estágios de *pipeline* de um determinado roteador devem ter cores diferentes.

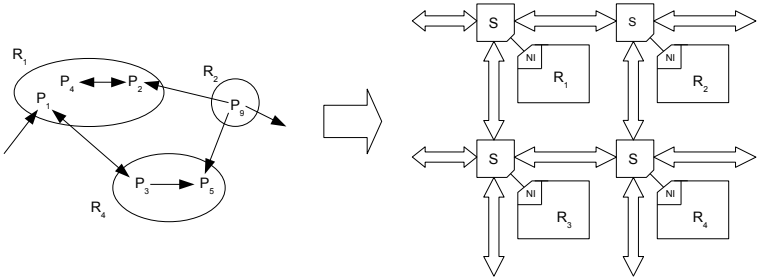


Figura 9 – mapeamento de recursos na rede Nostrum. Fonte: adaptado de (JANTSCH et al., 2004).

O mecanismo de LC é utilizado para garantir largura de banda para os canais virtuais. Um canal virtual na rede pode ser implementado como um *caminho fechado* entre o ponto de origem de um pacote e seu destino, e o seu retorno do destino para a origem. Os *Looped Containers* utilizam estes caminhos, carregado pacotes válidos ou mesmo vazios. O número de LCs que são reservados para um canal virtual reflete a garantia de largura de banda que o canal virtual tem.

A combinação entre TDNs e LCs implementa efetivamente um mecanismo de multiplexação por divisão no tempo. O número máximo de TDNs que existem em uma rede determina o número máximo de canais virtuais que podem compartilhar um enlace e também o número de LCs disponíveis para reserva na comunicação. Consequentemente, os LCs são equivalentes a *time slots* em uma rede TDM e os TDNs equivalem aos períodos de tempo dos *time slots*.

O número máximo de TDNs independentes é uma característica do projeto da rede, e corresponde a um período de *time slot* em uma rede TDM. Os canais virtuais são estaticamente em tempo de projeto e são configurados na inicialização da rede. No entanto, o número de LCs trafegando um canal virtual pode mudar em tempo de execução, ajustando a largura de banda oferecida aos requisitos do projeto. O número máximo de LCs está limitado ao número de TDNs na rede.

Outra característica da rede Nostrum é que o caminho fechado pode suportar múltiplos circuitos intercalando os LCs (atribuídos a

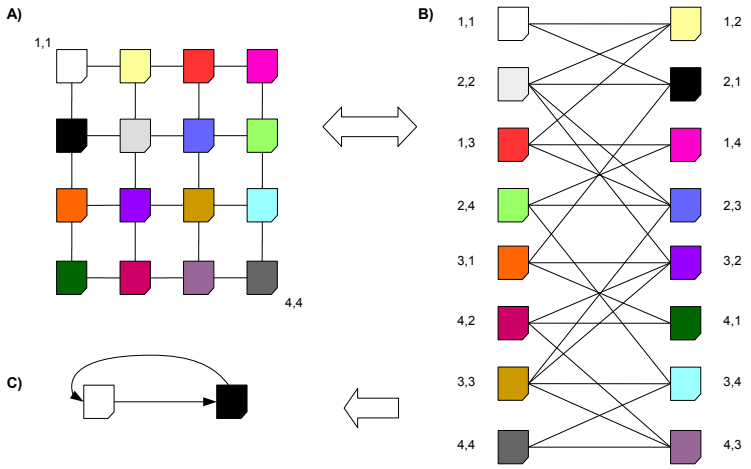


Figura 10 – Redes separadas (*disjoint*) devido à topologia da rede Nostrum. Fonte: adaptado de (MILLBERG et al., 2004).

este circuito), dentro do caminho fechado. Os canais virtuais permanecem independentes e com garantias temporais, de modo similar ao que acontece aos *time slots* em um escalonamento TDM.

Nenhum controle de fluxo é implementado na rede Nostrum, pois os pontos de origem e destino dos pacotes são considerados aptos para controlar qualquer tráfego, entrando dentro do mecanismo de multiplexação de tempo pré-definido. Para isso, *buffers* devem ser levados em consideração nos pontos de origem e destino dos pacotes de modo a manter os dados sem possibilidade de perda de dados (*overflowing*), o que requer alguma análise prévia para o cálculo de espaço necessário em cada buffer além de ser uma solução mais dispendiosa devido ao uso de *buffers* super-dimensionados.

O adaptador de rede da Nostrum é dividido em dois componentes principais. o primeiro deles é a interface de rede (em inglês *Network Interface* - NI) que oferece um conjunto de serviços aos núcleos de processamento, como ordenação de pacotes e desfragmentação de dados para serem transmitidos. O segundo é a Interface para Recursos de Rede (*Resource Network Interface* - RNI), que faz a adaptação do protocolo de transação usado pelos núcleos para o conjunto de serviços oferecidos pela NI e vice-versa.

2.3.1 Propriedades para análise de WCL

Na rede Nostrum os recursos são compartilhados pelos canais virtuais, e para todo canal virtual existente na rede, a largura de banda é garantida. O número de canais virtuais que existem na rede é definido estaticamente em tempo de projeto, dependendo da topologia e de outras características arquiteturais. A largura de banda pode ser ajustada em tempo de execução emitindo ou retirando LCs para o transporte de pacotes em um canal virtual. O tempo de ida e volta de um LC em um canal virtual é um múltiplo do número máximo de TDNs, T, passando por este canal virtual. Na rede Nostrum o tamanho de um pacote é assumido como sendo de um *flit*.

Tempo de Espera - Os canais virtuais são configurados na inicialização da rede e contém um número de LCs dedicados. O tempo de espera para um pacote considerando todo o caminho fechado é o tempo que ele espera no nodo de origem até que um LC de pacotes do circuito correspondente chegue. No pior caso, para um canal virtual de T TDNs e um LC dedicado a este canal virtual, um pacote irá esperar por $T_{wait;req} = T$ ciclos. Se t LCs são dedicados à este canal virtual, o pacote irá esperar no pior caso por $T_{wait;req} = T - t$ ciclos. O valor de t varia entre $[1; T]$ e portanto, o tempo de espera varia entre $T - 1$ e 0. O mesmo vale para o tempo de espera para a mensagem de resposta, $T_{wait;reply}$. O tempo de espera também depende da alocação de LCs no canal virtual. De acordo com os autores da rede Nostrum, em um caso típico os LC alocados para um canal virtual não irão aparecer de forma sucessiva, mas serão espalhados por todo o período alocado. Isto indica que $T_{wait;req} = T - t$ é um valor pessimista mas com um limite garantido de tempo de espera.

Latência - Após um pacote ter sido carregado em um LC, ele trafega através do caminho deste canal virtual sem tempo de espera posterior. A latência sofrida pelo pacote em cada roteador é dada por B, considerando B estágios de *pipeline* em cada roteador. A Latência ponto-a-ponto para um pacote trafegando através de h *hopes* é dada por $T_{latency} = B.h$ ciclos

Vazão - O número de pacotes que podem ser injetados na rede por ciclo de relógio (*clock*) depende do número de LCs que um canal virtual dispõem. Para cada pacote a ser injetado, ele deve esperar por um LC e portanto, ele usa um LC para trafegar na rede. Para t LCs em um canal virtual, com T TDNs, implica que em um período de T *time slots* t pacotes podem ser injetados na rede. Portanto, este canal virtual tem uma vazão de t/T pacotes por ciclo. Como t varia entre

$[1;T]$, a vazão varia entre $[1/T;1]$ pacotes por ciclo de relógio. Para o envio de um bloco de n pacotes, $(n.T)/t$ ciclos são necessários.

2.3.2 Resumo

A rede Nostrum baseada em TDM e requer sincronização de operação. A latência na transferência de um único pacote é proporcional ao número de *hops* no caminho, além de algum tempo de espera para o início da transmissão. Este tempo de espera é a fração de um período do TDM.

Conforme foi descrito na subseção anterior, a vazão na rede é inversamente proporcional ao período do TDM. Outra característica desta rede é que a largura de banda garantida é uma fração da largura de banda total. A temporização total na rede é afetada pelos fatores de topologia de toda a rede, e todo o tráfego influencia no comportamento temporal da rede. Isto leva a um comportamento de temporização pré-definida e previsível, bem adequada para sistemas de tempo real.

2.3.3 Evolução da rede Nostrum

O artigo intitulado "*Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip.*" foi apresentado pelos autores (JANTSCH et al., 2004), no qual eles discutiram o mecanismo de comunicação da rede Nostrum, o qual oferece serviço de largura de banda garantida, segundo seus autores, além de serviços de melhor esforço (BE). A largura de banda garantida é obtida através de canais virtuais (em inglês *Virtual Channels* - VC). O VCs são implementados usando uma combinação de dois conceitos que aqueles autores chamaram de "contêiner de loop" e "redes temporariamente disjuntas" (em inglês *Temporally Disjoint Networks* - TDN). Os contêiner de loop são usados para garantir o acesso à rede de forma independente da carga de rede atual, sem perder pacotes; e os TDNs são utilizados, a fim de alcançar vários VCs, além do tráfego de melhor esforço, na rede. Os TDNs são uma consequência da política de roteamento defletível usado, e dá origem a um TDM explícito dentro da rede (TDM é acrônimo de *Time-Division Multiplexing*). Para demonstrar o conceito da rede, aqueles autores implementaram o mecanismo proposto em linguagem VHDL e realizaram simulações com o código sintetizado. Segundo eles, O custo em termos de hardware adicional

necessário, assim como a largura de banda adicional é baixo e inferior a 2 % em ambos os casos. De acordo com as simulações feitas por eles, o tráfego BE praticamente não é afetado pelas VCs.

No artigo "*The Nostrum backbone - a communication protocol stack for Networks on Chip*." os autores (MILLBERG et al., 2004) propuseram uma pilha de protocolos de comunicação para ser usada na rede Nostrum. A fim de auxiliar o projetista no processo de seleção de quais partes dos protocolos utilizar e suas respectivas instalações que devem ser incluídas em cada projeto, foi proposta uma abordagem em camadas para a comunicação. Os autores sugeriram uma nomenclatura para descrever as interfaces das camadas individuais e as definições das camadas da pilha de protocolos de serviços. O conceito inclui suporte para fluxos de melhor esforço (BE), bem como suporte para o tráfego de fluxos com largura de banda garantida, usando circuitos virtuais. O artigo é finalizado com uma implementação de um caso de uso industrial.

Um dos principais autores da rede Nostrum, (JANTSCH, 2006), publicou em 2006 o artigo "*Models of Computation for Networks on Chip*". Segundo o aquele autor, as NoCs oferecem a oportunidade de apresentar um novo nível de abstração que define um conjunto de serviços da plataforma com características de desempenho e energia. Ao fazer a implementação destes serviços que podem ser totalmente irrelevantes para um determinado projeto de sistema, a separação efetiva no projeto do sistema do projeto dos componentes pode ser alcançada. O artigo discute os princípios para a formulação de serviços em *Network-on-Chip* para estabelecer um modelo computacional abstrato que expõe todas as propriedades relevantes da funcionalidade da plataforma, desempenho e consumo de energia enquanto oculta todos os detalhes irrelevantes da implementação. Como em muitos outros casos de abstrações bem sucedidas, estes princípios são baseados na separação da funcionalidade dos aspectos de tempo e de energia para permitir que o projetista continue raciocinando sobre estas propriedades no nível do sistema. Como exemplo, o autor formulou um documento de referência para ser usado na rede Nostrum. O documento baseia-se na largura de banda garantida (GB) e tráfego de melhor esforço (BE). Naquele documento os tráfegos GB e BE são caracterizados em termos de fórmulas pré-determinadas e permite a composição eficiente do tráfego na rede.

Os autores (PENOLAZZI; JANTSCH, 2006) publicaram em 2006 o artigo "*A High Level Power Model for the Nostrum NoC*" no qual eles propuseram um modelo de energia para a rede Nostrum. Para tal, um modelo empírico de energia dos canais de comunicação e roteadores

foi formulado e validado com o uso da ferramenta de síntese Synopsys Power Compiler®. O modelo, chamado de Nos-HPM (acrônimo de *Nostrum de Alto Nível Poder Model*) permite uma análise rápida do consumo de energia com uma precisão de 5 %. As simulações de sistemas com o modelo Nos-HPM podem ser executadas até 500 vezes mais rápido do que a ferramenta de síntese Synopsys Power Compiler® para uma rede de 4 × 4. Os autores encontraram um consumo máximo de energia de 0,7 `W` para uma malha de 4 × 4 e 3,5 `W` para uma rede em malha regular de 8 × 8, ambos implementados na tecnologia 0.18 μ`m` UPC CMOS. No pior dos casos a energia média por ciclo para um pacote de 128 bits foi de 508 `pJ`, enquanto que é de 20 `pJ` para um byte de carga útil. O consumo de energia de todos os canais de comunicação é equivalente ou ligeiramente superior do que o poder consumo de todos os roteadores, segundo aqueles autores. Os resultados obtidos foram comparados com os resultados de alguns trabalhos relacionados ao tema.

No artigo publicado por (VITKOVSKI et al., 2008), intitulado "*Low-power and error protection coding for network-on-chip traffic*", os autores exploraram o consumo de energia em uma rede Nostrum, analisando a eficácia de um codificador de baixo consumo de energia utilizado em técnicas de proteção contra erros de transmissão. Para o roteador sob estudo, foi utilizado um rotador da rede Nostrum. De acordo com aqueles autores, as simulações realizadas mostraram que apenas uma pequena fração da energia é dissipada nos blocos lógicos do roteador, enquanto que a maior parte da dissipação é devido aos fios de interconexão dos canais de comunicação. Os autores investigaram uma série de mecanismos de baixo consumo de energia e de proteção de dados, analisando o seu impacto no consumo de energia de toda a rede. O esquema de codificação em barramento-invertido e um conjunto limitado de códigos de Hamming foram implementados tanto nos canais de comunicação quanto na camada de rede. No entanto, destacam um ponto e um contraponto para os esquemas de codificação de dados. O ponto é que todos os esquemas de codificação de dados de baixa potência intuitivamente levam a crer que eles têm pouco potencial para diminuir o consumo de energia devido às suas significativas sobrecargas (*overhead*). Por outro lado, os mecanismos de proteção de erro tem um potencial significativo para diminuir o consumo de energia, porque eles permitem que a rede opere com uma tensão mais baixa. Segundo aqueles autores, as experiências mostram uma diminuição de 20 % no consumo de energia para uma dada taxa de erro e para um dado desempenho.

No artigo "*Priority Based Forced Requeue to Reduce Worst-Case Latencies for Bursty Traffic*", os autores (MILLBERG; JANTSCH, 2009) apresentaram um método de re-enfileiramento forçado baseado em prioridade como forma de diminuir as latências de pior caso em NoCs oferecendo melhor serviços para fluxos BE. O re-enfileiramento forçado é conhecer de antemão os pacotes de menor prioridade que ainda não foram injetados na rede e re-ordená-los com outras prioridades. A primeira vantagem dessa abordagem, aplicável a qualquer NoC oferecendo serviços de melhor esforço, é que os pacotes que ainda não entraram na rede agora podem competir com os pacotes que já foram injetados na rede e, portanto, há um controle mais rígido sobre os limites de tempo de admissão dos pacotes. O segundo benefício, relacionado diretamente com características construtivas da rede Nostrum, é que o re-embaralhamento de pacotes reduz drasticamente a latência dentro da rede para tráfego em rajadas, devido a um menor risco na ocorrência de colisões na saída da rede. O artigo apresenta um detalhamento do estudo no método de re-enfileiramento forçado com diferentes tráfegos. Segundo os seus autores, os resultados experimentais mostram uma redução de 50 % na latência de pior caso, sob a perspectiva do sistema, graças a uma re-ordenação na distribuição de latência, mantendo o mesmo valor de latência média.

O artigo "*Theorem Proving Techniques for the Formal Verification of NoC Communications with Non-minimal Adaptive Routing* apresentado por (HELMY; PIERRE; JANTSCH, 2010) foca na verificação formal na comunicações em redes intra-chip. Eles descrevem uma versão melhorada da metodologia conhecida como GeNoC e fizeram sua avaliação na rede Nostrum que engloba várias características não triviais, como o algoritmo de roteamento não-mínimo denominado de "deflectivo". Os autores demonstraram como as características das camadas do protocolo da rede Nostrum podem ser capturadas pela versão da GeNoC por eles alterada, que permite uma formalização passo-a-passo das operações de comunicação levando em conta vários detalhes de protocolo em consideração.

O artigo "*Run-time Partitioning of Hybrid Distributed Shared Memory on Multi-core Network-on-Chips*" apresentado por (XIAOWEN et al., 2010) trata da interconexão de múltiplos processadores conectados em uma NoC. Segundo aqueles autores, nestes sistemas as memórias são preferencialmente distribuídas e dar suporte ao DSM (acrônimo do inglês *Distributed Shared Memory*) é essencial para uma boa reutilização de código legado e facilita programação. No entanto, a organização DSM implica na sobrecarga inerente de traduzir endereços de

memória virtuais em endereços de memória física, resultando em um desempenho negativo. Aqueles autores observaram que, em aplicações de programação paralela, dados diferentes têm propriedades diferentes (privados ou compartilhados). Para os dados de acesso privado, é desnecessário realizar a tradução de endereços virtuais para físicos. O trabalho realizado por aqueles pesquisadores estava focado em diminuir a sobrecarga na tradução de endereço virtual para físico, melhorando assim o desempenho do sistema. Eles propuseram um DSM de organização híbrida que oferece suporte ao particionamento dos dados em tempo de execução de acordo com suas propriedades. O objetivo do DSM híbrido é oferecer suporte ao acesso rápido em memórias físicas para acessos à dados privados e a manutenção de um espaço de memória global para dados compartilhados. Foi construída uma plataforma de hardware baseada em uma rede Nostrum. Segundo seus autores, os resultados experimentais de aplicações reais mostraram que a organização híbrida DSM com o particionamento de tempo de execução apresenta vantagem de desempenho sobre o DSM convencional. O percentual de melhoria de desempenho depende do tamanho do caso de uso, da maneira de particionamento dos dados e da relação computação/comunicação de aplicações paralelas, entre outras. Nos experimentos daqueles autores a melhoria máxima foi de 34,42 %, e a melhoria mínima foi de 3,68 %.

Os autores (NAEEM; JANTSCH; ZHONGHAI, 2013) apresentaram no artigo "*Scalability Analysis of Memory Consistency Models in NoC-based Distributed Shared Memory SoCs*" no qual analisaram seis modelos de consistência de memória em uma NoC de um sistema multi-core baseado em memória distribuída. Os modelos são listados em seus nomes e abreviaturas originais na língua inglesa. São eles:

- protected release consistency (PRC);
- release consistency (RC);
- weak consistency (WC);
- partial store ordering (PSO);
- total store ordering (TSO); e
- sequential consistency (SC).

As implementações são baseadas em um contador de transação e em uma abordagem baseada em pilha-de-endereço. A análise de escalabilidade foi baseada em diferentes cargas de trabalho mapeados em vários tamanhos de redes que tinham tamanhos diferentes. Para os

experimentos, foi utilizado um sistema multi-core configurável baseado em uma plataforma com a rede Nostrum com uma topologia de malha de 2-D e um algoritmo de roteamento denominado de "algoritmo de deflexão". No artigo os autores descrevem o desempenho que os modelos apresentam nas diversas condições de carga e tamanhos de redes, informando qual o consumo de recursos de silício de cada implementação.

A Figura 11 mostra a linha do tempo dos artigos aqui apresentados.

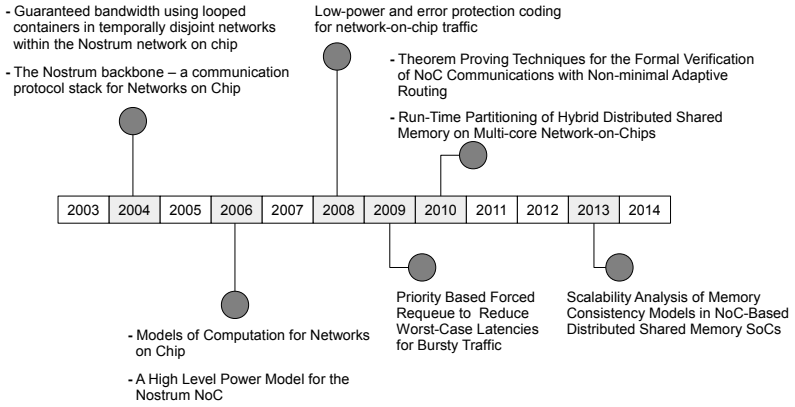


Figura 11 – Linha do tempo das publicações relacionadas com a rede Nostrum.

2.4 MANGO

A rede MANGO (acrônimo de *Message passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces*) foi apresentada por Bjerregaard e Sparzo (BJERREGAARD; SPARSO, 2005), (BJERREGAARD; SPARSO, 2006), (BJERREGAARD; SPARSO, 2005). De modo similar às outras redes apresentadas, ela consiste de adaptadores de rede (*Network Adapters* - NA), roteadores e canais de comunicação. Os roteadores podem ser conectados em uma estrutura tipo grelha, tanto homogênea como heterogênea. A Figura 12 mostra um exemplo de SoC implementado com a rede MANGO heterogênea. Naquela Figura, as regiões em cor cinza representam áreas do SoC sem domínio de *clock*, ou seja, regiões assíncronas. As áreas demarcadas

com linha tracejada representam representam regiões de *clock* independentes e normalmente associadas ao sincronismo local dos núcleos que compõem o SoC, representados por quadrados brancos indicados por *IPcore*. Os adaptadores de rede são representadas na Figura 12 por paralelogramos de cantos arredondados e indicados por *NA*. Cada NA é conectado a um roteador, e é responsável pela sincronização entre o núcleo, que possui o seu sincronismo de relógio próprio, e a rede que é assíncrona. Além disso, cada adaptador oferece serviços de comunicação de alto nível, utilizando o protocolo OCP (acrônimo de *Open Core Protocol*) (BJERREGAARD; SPARSO, 2005), e indicados na Figura 12 como elipses em cor preta.

O protocolo OCP pode ser entendido como um padrão de reuso de hardware. Ele consiste em um conjunto de sinais e protocolos de comunicação que têm como objetivo padronizar a comunicação entre blocos núcleos em um SoC. O padrão OCP define uma interface ponto a ponto entre duas entidades de comunicação denominadas mestre e escravo. A interface mestre é a responsável pelo controle da comunicação e somente ela pode iniciar ou parar uma troca de informações. A interface escravo, por sua vez, responde aos comandos da interface mestre, realizando as operações correspondentes à sua solicitação e enviando ou recebendo dados.

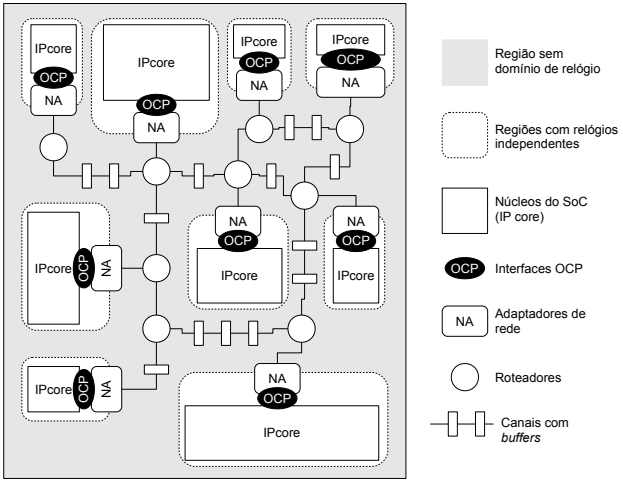


Figura 12 – Exemplo de SoC baseado na rede MANGO. Fonte: adaptado de (BJERREGAARD; SPARSO, 2005).

Os dois últimos elementos representados na Figura 12 são os roteadores, representados por círculos brancos, e os enlaces de comunicação dotados de *buffers* representados por retângulos brancos. Cada canal de comunicação na rede MANGO pode implementar uma quantidade diferente de *buffers*, logicamente separados em canais virtuais. Tais canais virtuais competem pelo acesso ao canal físico de comunicação. Deste modo, um canal de comunicação naquela NoC pode ser compartilhado por múltiplos circuitos fim-a-fim, e o compartilhamento é controlado por uma arbitragem local nos canais de saída de cada roteador da rede. Isto efetivamente divide cada canal em um número de canais virtuais, sendo um para cada circuito que está utilizando o canal.

A Figura 13 mostra uma figura conceitual do roteador MANGO, representando suas principais implementações. O roteador implementa um número uni-direcional de canais; dois deles são usados para a conexão de um núcleo local, através de um NA. O restante dos canais são, via conexões ponto-a-ponto, usados para conectar o roteador aos roteadores vizinhos. Cada um destes canais possui um número independente de *buffers* para canais virtuais (VC). Canais de rede e canais de conexão local implementam o mesmo tipo de interface. É função do adaptador de rede (NA) traduzir a comunicação recebida da rede no formato de pacotes de rede, ou a ser enviada por ela. Internamente, o roteador consiste de um roteador de melhor esforço (BE), um roteador com garantias de serviço (GS), *buffers* de saída e árbitros de canais.

Os roteadores BE e GS são implementados em separado. Isto foi feito por seus autores para oferecer modularidade ao roteador. Esta abordagem é similar ao que foi feita no roteador da rede Æthereal (GOOSSENS; DIELISSEN; RADULESCU, 2005). O roteador BE adota a técnica de roteamento baseado na origem para realizar o roteamento dos pacotes de dados sem estabelecimento de conexão, de acordo com o caminho de roteamento definido no cabeçalho do pacote. Um subconjunto de canais virtuais são alocados para o roteamento BE.

O roteamento GS utiliza os canais virtuais remanescentes para realizar o roteamento de dados, sem cabeçalho; ou seja, realiza conexões estaticamente programadas. Para oferecer garantias de rígidas de tempo real *hard real-time*[3], as conexões GS precisam ser logicamente independentes de outros fluxos de tráfego da rede. Para isso, na rede MANGO é feita a implementação de um circuito ponto-a-ponto lógico entre dois canais na rede, através da reserva de uma sequência de ca-

[3]Uma tarefa é dita como sendo *hard* se a perda de um *deadline* impossibilitar o funcionamento normal do sistema, sendo que o resultado da tarefa não é útil após o seu *deadline* (LIU, 2000).

nais virtuais independentes. O roteador GS oferece chaveamento sem bloqueio entre canais de entrada e *buffers* de saída, e portanto o roteamento GS pode ser realizado apenas com base na arbitragem de acesso ao canal.

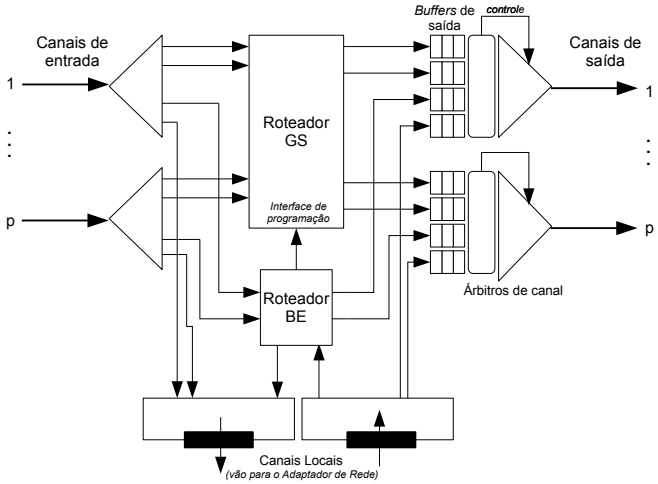


Figura 13 – Conceito do roteador MANGO. Fonte: adaptado de (BJERREGAARD; SPARSO, 2005).

O chaveamento da entrada para a saída do roteador é executado por uma *crossbar* e ajustado estaticamente na inicialização da rede. Portanto, um circuito é uma sequência de canais de saída, canais virtuais e memórias FIFO (*output buffers* na Figura 13) conectadas por matrizes *crossbar*. É utilizado um mecanismo de controle de fluxo baseado em créditos entre as FIFOs dos canais virtuais para prevenir postergações indefinidas entre os canais compartilhados.

Os módulos de controle de admissão e arbitragem nos canais de saída dos roteadores determinam a largura de banda e latência da conexão. Um esquema de compartilhamento justo fornece garantias iguais para todos os canais virtuais que compartilham o canal. Um esquema alternativo de escalonamento baseado em prioridades pode ser usado, o qual garante o acesso ao enlace primeiro para os canais virtuais com maior prioridade.

2.4.1 Propriedades para análise de WCL

A MANGO consegue facilitar o projeto modular e a arquitetura GALS (acrônimo de *Global Asynchronous, Local Synchronous*) resolvendo as questões de relógio e sincronização ao longo do *chip*, embora assegurando garantias de tempo real. Ela não requer controle fim-a-fim visto que uma arbitragem mais refinada é feita nos roteadores aproveitando o peso dos adaptadores de rede. No entanto, os roteadores apresentam um maior custo de área de silício, visto que eles devem prever armazenamento para cada canal virtual. As propriedades temporais dependem do número de canais virtuais que compartilham um enlace (C) e do esquema de arbitragem para o enlace.

Tempo de espera - não há tempo de espera inicial no ponto de origem do pacote, pois os circuitos virtuais são configurados estaticamente na inicialização. Existe algum tempo de espera pela arbitragem em cada enlace, o qual será considerado na latência de transmissão de cada roteador do caminho. Portanto, os parâmetros $T_{wait;req}$ e $T_{wait;reply}$ são iguais a zero.

Latência - os pacotes na MANGO consistem de um número de *flits*. A latência de um *flit* para atravessar um *hop* (roteador e enlace) depende da profundidade do *pipeline do roteador* (B) e do esquema de arbitragem adotado para o enlace. Assumindo um esquema de arbitragem de compartilhamento justo, um *flit* requer C ciclos de relógio de espera para ter acesso ao enlace, no pior caso. Este é o número de canais virtuais, ou seja *buffers* de saída, que compartilham o enlace. Consequentemente, a latência de um *hop* é de $C + B$ ciclos. *Flits* podem ser encaminhados para o próximo *hop* sem ter de esperar pela chegada de todo o pacote. Portanto, a latência de um pacote de f *flits* atravessando um caminho de h roteadores é $T_{latency} = (h + f).(C + B)$ ciclos.

Vazão - todo *flit* tem de passar pela arbitragem de acesso ao enlace. Um pacote de f *flits* irá injetado com sucesso na rede em $(f.C)$ ciclos. A taxa de injeção de pacotes na rede é de $1/(f.C)$ pacotes por ciclo. Para a transferência de um bloco com n pacotes são necessários $n.f.C$ ciclos de relógio.

2.4.2 Resumo

A latência de transmissão de um único pacote na rede MANGO depende do número de canais virtuais que compartilham o enlace.

Além disso, não há tempo de espera no início da transmissão, pois o tráfego é controlado localmente. A vazão é inversamente proporcional ao número de canais virtuais que estão usando um determinado enlace. Os parâmetros de tempo na rede MANGO são afetados por fatores locais de tráfego e as decisões são tomadas localmente, diferente do que ocorre em redes TDM tradicionais que se baseiam em um escalonamento global. Esta característica oferece flexibilidade e ajuste às necessidades de tráfego. O custo desta abordagem é alto em termos de área de silício, pois cada canal virtual é separado em cada roteador.

2.4.3 Evolução da rede MANGO

Os autores (BJERREGAARD; SPARSO, 2005) publicaram o primeiro artigo relacionado aos estudos que levaram à rede MANGO. O artigo era intitulado "*Scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip*" e tratava sobre a importância de serviços garantidos em NoCs, na medida em que proporcionam previsibilidade nas dinâmicas complexas de estruturas de comunicação compartilhadas. O artigo discutia a implementação de GS em uma rede-on-chip assíncrona. Os autores apresentaram uma nova disciplina de escalonamento chamada garantia latência assíncrona de escalonamento (em inglês *Asynchronous Latency Guarantee* - ALG), que fornece garantias de latência e largura de banda no acesso a um meio compartilhado, i.e. uma ligação física compartilhada entre um certo número de canais virtuais. Segundo os seus autores, a principal contribuição do algoritmo ALG é o acoplamento entre a latência e a largura de banda, ambas com garantias de atendimento. Os autores implementaram uma simulação do algoritmo em tecnologia CMOS de 120 `nm`. A velocidade de operação alcançada pelo experimento foi de 702 `MDI/s`.

Os autores (BJERREGAARD; SPARSO, 2005) publicaram o artigo "*A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip*" apresentando o roteador da rede MANGO. Na visão deles, redes on-chip para futuros projetos de SoCs precisam de implementações simples e de alto desempenho. A fim de promover a integridade de nível de sistema, serviços garantidos (acrônimo do inglês *Guaranteed Services* - GS) precisam ser fornecidos. Aqueles autores propuseram a arquitetura de um roteador para NoC para suportar isto, e demonstraram com um projeto de célula padrão na tecnologia CMOS. A aplicação é baseada em técnicas de

circuito sem relógio, e, portanto, inerentemente suporta um fluxo de projeto GALS. A técnica GALS combina o uso de sistemas síncronos e assíncronos. Os componentes são projetados como se fossem operar de forma síncrona, formando um subsistema que recebe um único sinal de *clock*. A composição de diversos subsistemas formam o SoC tendo cada um deles o seu próprio sinal de *clock*. Desta forma, diz-se que cada subsistema trabalha com o seu próprio domínio de relógio (localmente síncrono) e que não estão necessariamente sincronizados entre sí (globalmente assíncrono) (ROYAL, CHEUNG, 2003). O roteador apresentado pelos autores explora canais virtuais para fornecer GS orientados a conexão, bem como conexão para serviços de roteamento para fluxos de melhor esforço (BE). A arquitetura é altamente flexível, em que o suporte para diferentes tipos de roteamento BE e arbitragem GS podem ser facilmente conectado ao roteador.

O artigo "*An OCP Compliant Network Adapter for GALS-based SoC Design Using the MANGO Network-on-Chip* foi apresentado por (BJERREGAARD; MAHADEVAN; OLSEN; SPARSOE, 2005), e focou na reusabilidade de componentes de propriedade intelectual (em inglês *Intelectual Property* - IP). Neste artigo os autores descrevem uma arquitetura de um adaptador OCP (acrônimo do inglês *Open Computer Project*) e um adaptador de rede (em inglês *Network Adapter* - NA) para a rede MANGO. O NA oferece o desacoplamento entre a comunicação e a computação, proporcionando transações OCP mapeadas em memória com base em serviços de transmissão de mensagens primitivas da rede. Além disso, facilita os sistemas tipo GALS, adaptando-se à rede sem relógio. Os autores avaliaram o desempenho destes adaptadores e a área de silício ocupada utilizando como referência a tecnologia CMOS de 130 `nm`.

Os autores (BJERREGAARD; SPARSO, 2006) apresentaram uma versão completa da rede MANGO, já utilizando o adaptador OCP, no artigo "*Implementation of guaranteed services in the MANGO clockless network-on-chip*". No artigo, os autores apresentam as principais vantagens da rede MANGO como a implementação sem relógio centralizado, adaptadores para pontos de acesso à rede padronizados e serviços de comunicação com garantias de serviço. Segundo aqueles autores, entre as vantagens da utilização de técnicas de circuito sem relógio estão a baixa latência para encaminhamento de pacotes em *pipeline* e o consumo zero de energia em dinâmica de *idle*. A multiplexação por divisão de tempo, geralmente utilizada para proporcionar garantias de largura de banda em NoCs sincronizados, no entanto, não é possível em um ambiente sem relógio. Como solução, a rede MANGO fornece uma

alternativa de alto desempenho, com garantias rígidas de serviço orientado por conexão, utilizando técnicas de circuito sem relógio. Os circuitos da rede MANGO são apresentados em detalhes neste artigo, além da implementação de uma rede MANGO em malha regular 5 × 5, na tecnologia CMOS de 130 nm.

"*Packetizing OCP Transactions in the MANGO Network-on-Chip*" é o artigo onde os autores (BJERREGAARD; SPARSO, 2006) tratam do empacotamento de dados para transmissão na rede MANGO através dos adaptadores OCP. Neste artigo eles apresentam o "modelo de programação centrada no núcleo" para estabelecer e utilizar ligações GS na rede MANGO. Eles demonstram como as transações OCP são empacotadas e enviadas através da rede compartilhada, e ilustram como isso afeta o desempenho ponto-a-ponto. Os autores demonstram a previsibilidade da latência de comunicação em canais de comunicação compartilhados utilizando uma ferramenta dedicada para tráfego na rede MANGO.

No artigo "A Simple Clockless Network-on-Chip for a Commercial Audio DSP Chip" os autores (STENSGAARD; BJERREGAARD, SPARSOE; PEDERSEN, 2006) apresentam o projeto de uma comutação de pacotes na rede MANGO como um substituto para a existente infraestrutura de comunicação baseada em *crossbar* em um Processador Digital de Sinais comercial (também conhecido pelo acrônimo DSP) para tratamento de áudio. Ambas as soluções são apresentadas utilizando tecnologia CMOS de 180 nm e comparados em termos de área ocupada, consumo de energia e complexidade de roteamento dos dados para processamento. Apesar da solução com NoC apresentar um maior consumo de energia e maior ocupação de área de silício, esta solução ainda responde por menos do que 1% da área total do chip e do consumo de energia. Baseado nestes resultados, aqueles autores acham que a solução com NoC é mais adequada devido aos benefícios por ela proporcionados, como: (*i*) a NoC é modular, escalável, e em contraste com a *crossbar* existente, o que permite que todos os blocos possam comunicar entre si; (*ii*) o comprimento total dos fios utilizados no SoC é diminuído até 22%, o que facilita o processo de layout e torna o design menos propenso a congestionamentos no roteamento; (*iii*) os blocos comunicantes são dissociados por meio da NoC, oferecendo um assincronismo-global, sincronismo-local (GALS) onde o sistema de relógio independente dos blocos individuais é ativado. Segundo aqueles autores, o estudo apresentado neste artigo mostra que NoCs são viáveis mesmo para pequenos sistemas.

Os autores (BJERREGAARD; STENSGAARD; SPARSOE, 2007) apre-

sentaram o artigo "*A Scalable, Timing-Safe, Network-on-Chip Architecture with an Integrated Clock Distribution Method*", no qual eles apresentam uma estratégia de comunicação integrada e uma estratégia baseada em relógio Mesocrono. Em um sistema Mesocrono, todos os relógios possuem a mesma frequência, mas não necessariamente a mesma fase (SODERQUIST, 2002). Com isto, são evitados erros relacionados à temporização, mantendo uma perspectiva do sistema globalmente sincronizada. A arquitetura é escalável pois a integridade da temporização é baseada puramente em observações locais. Os autores demonstraram resultados experimentais para comprovar suas hipóteses, utilizando a tecnologia CMOS de 90 nm para implementar a rede experimental (MANGO).

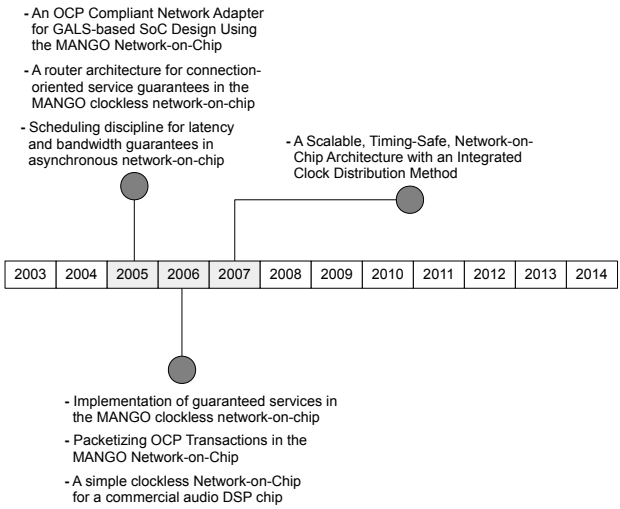


Figura 14 – Linha do tempo das publicações relacionadas com a rede MANGO.

2.5 ÆTHEREAL

A rede Æthereal foi proposta por Kees Goossens *et al.* (GOOSSENS; DIELISSEN; RADULESCU, 2005) e é baseada em chaveamento de

circuitos TDM, oferecendo mecanismos para o tratamento de fluxos BE e GS. A alocação temporal é um conceito fundamental para a rede Æthereal. Ela possui *time slots* de duração fixa no qual um bloco de dados pode ser encaminhado através de um roteador. A duração de cada *time slot* é a mesma para todos os roteadores da rede e todos os roteadores trabalham de modo sincronizado. Isto significa que cada roteador em um mesmo período de tempo pode encaminhar um bloco de dados para cada canal de saída. A alocação geral dos *slots* por canal de saída, por roteador, é definida em tempo de projeto e a rede configurada em tempo de execução. Um *time slot* é repetido periodicamente e este período é o mesmo para toda a rede.

Cada roteador da rede Æthereal/Aelite de N entradas e N saídas possui uma tabela de roteamento temporal, denominada de *Tabela de Slots*. Toda tabela de *slots*, T, possui S *time slots* (linhas) e N saídas do roteador (colunas). A uma noção lógica de sincronismo na Æthereal/Aelite: todo roteador da rede ocupa o mesmo *time slot* de duração fixa. Em um *slot*, s, um nó da rede (uma interface de rede) pode ler e escrever até um bloco de dados por canal de entrada e saída, respectivamente. No próximo *slot*, $(s + 1)$, a nó da rede escreve e lê blocos de dados em seus canais de saída apropriados. Deste modo, os blocos de dados propagam pela rede em um modo *store-and-forward* e não pode sofrer *deadlock*. A latência que um bloco de dados sofre por roteador é igual a duração do *slot*, e a reserva de *slot* garante a largura de banda em múltiplos blocos de dados por S *slots*.

A tabela de *slots* associa as entradas com as saídas do roteador para todo *slot*: $T(s, o) = i$, o que significa que se houver blocos de dados presentes em uma entrada i, tais blocos serão comutados para a saída o a cada $s + kS$ *slots*, sendo que $k \in N$. Uma entrada está vazia quando não há reserva para uma saída qualquer em um determinado *slot*. Devido ao seu aspecto construtivo, não há contenção na rede porque há pelo menos uma entrada para cada saída, para cada *slot*

A Figura 15 o conceito de roteamento livre de contenção proposto pelos autores da rede Æthereal/Aelite, com seus roteadores e tabelas de *slots*. A rede contém três roteadores, R_1, R_2 e R_3, no *time slot* $s = 2$, indicado na terceira entrada de cada tabela T daquela Figura (os *slots* foram numerados iniciando em zero). O tamanho das tabelas de *slots* é $S = 4$, e a Figura descreve somente as colunas relevantes. As três setas na cor cinza denominadas de a, b e c representam as conexões; os três círculos pretos rotulados com as letras a, b e c representam blocos de dados nas correspondentes conexões. O roteador R_1 faz o chaveamento do bloco b da entrada i_1 para a saída o_2, como indicado na Tabela

$T_1(2, o_2) = i_1$. De modo similar, o roteador R_2 faz o chaveamento do bloco a para a saída o_2, e R_3 chaveia o bloco c para a saída o_1.

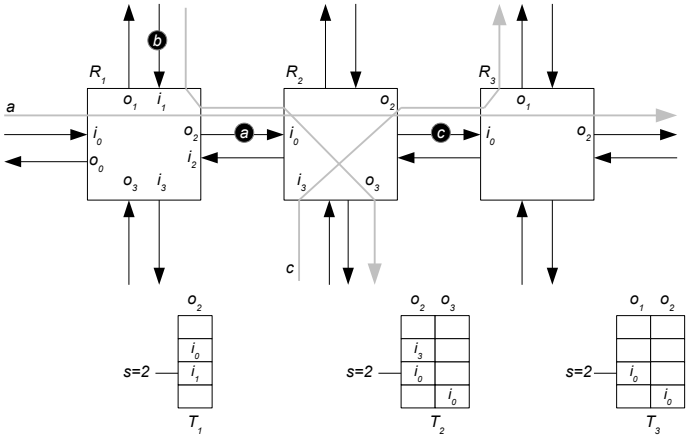


Figura 15 – Roteamento sem contenção: rede com três roteadores (R1, R2, and R3) no *time slot* $s = 2$, com as correspondentes tabelas de *slots* (T1, T2, and T3). Fonte: adaptado de (GOOSSENS; DIELISSEN; RADULESCU, 2005).

Circuitos virtuais são definidos como caminhos entre um nodo até outro através de um número de saltos na rede (*hops*). As garantias temporais são fornecidas por circuitos virtuais, atribuindo-lhes um número de intervalos de tempo dentro de um período. Um bloco de dados em um nodo de origem espera por um *time slot* dedicado e quando ele chega o pacote é encaminhado de um roteador para outro, em um *time slot* sequencial, sem bloqueio ou tempo de espera adicional.

Assim como a rede Nostrum, a rede Æthereal suporta tanto fluxos com garantia de serviços (GS) ou melhor esforço (BE). Fluxos GS são tratados pelo mecanismo de roteamento baseado na alocação de *slots*, enquanto que os fluxos BE são roteados por um mecanismo convencional de roteamento *wormhole*, e a arbitragem é feita por um algoritmo *round-robin*. Os fluxos BE utilizam o conceito de roteamento na origem, no qual o cabeçalho do pacote (*header*) contém o caminho de roteamento, desde a origem até o seu destino. Cada roteador remove os bits necessários ($log_2 N$, sendo que N é o tamanho da rede $N \times N$) da informação do caminho para determinar para qual canal de saída o pacote deve ser direcionado. Devido a não existência de múltiplas classes

de *buffers*, os pacotes BEs podem sofrer *deadlock*. Uma estratégia de roteamento apropriada deve ser feita em tempo de projeto para evitar tal problema.

Para aumentar a reutilização de recursos, a rede Æthereal permite que fluxos BEs utilizem *time slots* não reservados para fluxos prioritários, assim como os *time slots* reservados, porém não utilizados momentaneamente pelos fluxos GS. Os roteadores BEs tem a menor prioridade, o que significa que um canal só pode ser utilizado por um fluxo BE quando não houver fluxo GS fazendo uso do mesmo.

A Figura 16 apresenta um diagrama com os principais blocos funionais do roteador da Æthereal. Naquela Figura, os fluxos BEs compartilham com os fluxos GS os canais de comunicação entre os roteadores. Os fluxos GS não são tratados pelo Analisador de Cabeçalho (*Header Parsing Unit - HPU* naquela Figura), porque eles possuem reserva garantida de *slots* em tempo de projeto. A Unidade de Reconfiguração na Figura 16 é logicamente separada do roteador, e tratada como se fosse um outro canal de saída do roteador. Um roteador $(N+1) \times (N+1)$ associado a uma RCU efetivamente implementam um roteador $(N \times N)$. A função da unidade RCU é programar as tabelas de *slots*, de modo a garantir que as filas BEs não sofrerão *overflow*. Tal programação é baseada no conceito de fluxo baseado em créditos. *Buffers* disponíveis nos adaptadores de rede armazenam pacotes de dados em filas e créditos são enviados do roteador de destino para o roteador de origem quando existem espaço suficiente para armazenar mais pacotes, evitando assim um *buffer overflow*

2.5.1 Propriedades para análise de WCL

A Æthereal é baseada no conceito de períodos de *time slots* e no escalonamento destes *time slots*. Isto implica que um período de um escalonamento é seguidamente repetido. Para um circuito virtual, considerando um período de P *time slots*, p como o número de *slots* que podem ser alocados para o circuito. As propriedades do tempo de execução dependem de parâmetros de projeto, como profundidade do *pipeline* em cada roteador (B), a duração de um *time slot* em ciclos de relógio (s) e o número de *flits* que constituem o pacote (f).

Tempo de espera - para que um pacote de dados seja transmitido através de um circuito virtual, ele precisa aguardar pela chegada do *slot* de tempo correspondente. Se o circuito virtual possui p *time slots*, o tempo de espera no pior caso seria $T_{wait;req} = (P - p).s$ ciclos

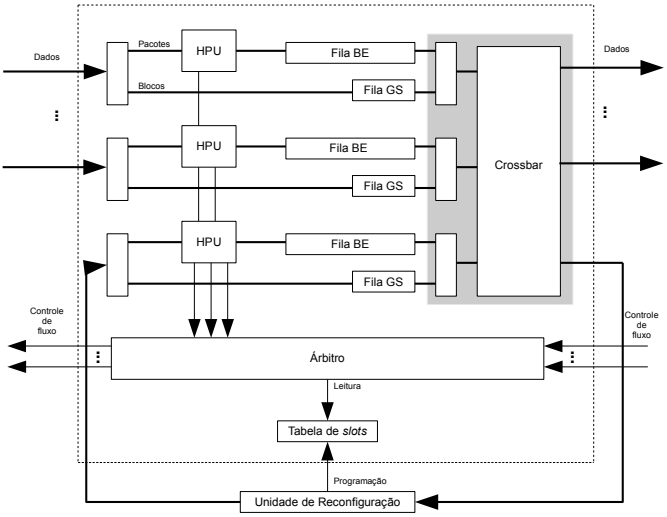


Figura 16 – Arquitetura interna do roteador da Æthereal. Fonte: adaptado de (GOOSSENS; DIELISSEN; RADULESCU, 2005).

de relógio, sendo que p varia entre os limites $[1; P]$. O tempo de espera é afetado pela posição de diferentes *time slots* atribuídos a um circuito virtual dentro de um período. O tempo $T_{wait;req} = (P - p).s$ é um limite de pior caso, com p *time slots*.

Latência - a latência de um pacote em cada roteador é função da profundidade do *pipeline* (B), o qual é igual a duração de um *time slot* em ciclos de relógio (s), e do número de *flits* do pacote (f). Não há espera nos roteadores, pois cada *flit* é encaminhado imediatamente para o próximo roteador. Portanto, a latência de um pacote, considerando h roteadores no caminho entre a origem do pacote até o seu destino, é dada por $T_{latency} = h.B + f$ ciclos de relógio.

Vazão - em termos de vazão, o escalonamento é organizado em períodos de P *time slots* e a vazão de um circuito depende dos *time slots* que ele possui. Com p *time slots* alocados para um circuito virtual, p pacotes podem ser injetados na rede em um período. Assim, um circuito pode alcançar a vazão total de $p/(P.s)$ pacotes por ciclo de relógio. Para a transferência de um bloco de n pacotes, $(p.n)/(P.s)$ são necessários.

2.5.2 Resumo

A rede Ætheral oferece uma latência para a transmissão de pacotes proporcional ao número de roteadores no caminho entre a origem e o destino de tais pacotes. Algum tempo de espera tem de ser considerado no início da transmissão, o qual depende do período do TDM. Como rede TDM, ela é fortemente baseada na noção de tempo. A vazão é inversamente proporcional ao período do TDM e possui dependência da reserva de *time slots*. De acordo com os seus autores, uma visão geral da rede deve ser considerada em todos os seus parâmetros. Por exemplo, o período TDM e escalonamento de *time slot* são decisões que afetam todo o tráfego da rede. Portanto o seu uso em sistemas de tempo real exige uma análise estática em tempo de projeto para poder oferecer garantias temporais para o sistema.

2.5.3 Evolução da rede Æthereal

O artigo intitulado "*Trade-offs in the design of a router with both guaranteed and best-effort services for Networks-on-Chip*", apresentado por (RIJPKEMA et al., 2003), é a primeira publicação sobre a estrutura de um roteador da rede Æthereal. O artigo discute as questões de fluxos com garantia de vazão (em inglês *Guaranteed Throughput - GT*) e fluxos tratados com a política de melhor esforço (em inglês *Best Effort - BE*). No artigo os autores discutem também a relação custo-benefício entre complexidade do hardware e eficiência dos roteadores para justificar as escolhas feitas para o roteador da rede. A primeira versão da rede foi implementada em tecnologia de 130 `nm`. Um roteador da rede ocupou uma área de 26 $\texttt{mm}^2$, com uma largura de banda de 80 `Gbps`.

Os autores (RADULESCU et al., 2005) apresentaram no artigo "*An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration*" uma interface de rede (NI) para a rede Æthereal. A NI desacopla "computação" de "comunicação", através da oferta de uma abstração de memória compartilhada, que é independente da implementação da rede. Aqueles autores adotaram um protocolo *transaction based* para alcançar compatibilidade com os protocolos de barramento existentes, como AXI, OCP, e DTL. A NI tem uma arquitetura modular, que permite instanciação flexível. Ela fornece tanto vazão garantida quanto melhor esforço através de conexões específicas. Uma instância desta interface de rede com quatro portas ocupou uma área de 0,25 $\texttt{mm}^2$ adotando uma tecnologia de 130 `nm`, com

capacidade de operação em `500 MHz`.

A publicação do artigo "*Æthereal network on chip: concepts, architectures, and implementations*" pelos autores (GOOSSENS; DIELISSEN; RADULESCU, 2005) é uma síntese dos trabalhos desenvolvidos anteriormente, no qual o nome da rede é oficializado na comunidade científica (Æthereal). Basicamente, o artigo apresenta um detalhamento do funcionamento dos roteadores e da rede, baseada em TDM. As informações sobre o funcionamento da rede que foi apresentado no início desta seção foram obtidos neste artigo.

No artigo "*Trade-Offs in the Configuration of a Network on Chip for Multiple Use-Cases*" os autores (HANSSON; GOOSSENS, 2007) exploram computação reconfigurável com o uso de NoC. Para mitigar a complexidade na programação dos muitos registros que controlam a NoC de sistemas baseados em computação reconfigurável, aqueles autores apresentam no artigo uma abstração sob a forma de uma biblioteca de configuração. A biblioteca foi chamada de *Æthereal Run-Time (ART) library*. Uma premissa desta biblioteca é deixar o sistema modificado em um estado consistente, a partir do qual operação normal pode continuar. Neste artigo aqueles autores tratam das instalações para controlar a mudança em NOC reconfigurável. Mostram as adições arquiteturais e as muitas relações de custo-benefício na concepção de uma biblioteca de tempo de execução para uma NoC quando usada em computação reconfigurável. No artigo foram feitas análises qualitativas e quantitativas para avaliar o desempenho, os requisitos de memória, a previsibilidade e reutilização das diferentes implementações.

O artigo "*A Unified Approach to Mapping and Routing on a Network-on-Chip for Both Best-Effort and Guaranteed Service Traffic*", apresentado pelos autores (HANSSON; GOOSSENS; RADULESCU, 2007), trata do mapeamento de núcleos em uma NoC, utilizando a rede Æthereal como plataforma de validação. Uma das etapas fundamentais no design baseado em Network-on-Chip é o mapeamento espacial de núcleos e o encaminhamento da comunicação entre esses núcleos. Soluções para os problemas de mapeamento e roteamento primeiro mapeiam núcleos conhecidos e em seguida a comunicação, normalmente usando funções objetivos separadas e, eventualmente conflitantes. Neste artigo, aqueles autores apresentaram um algoritmo de objetivo-único chamado de UMARS+ (acrônimo do inglês *Unified MApping, Routing, and Slot allocation*). Segundo os seus autores, a relação complexidade-tempo da UMARS+ é baixa e os resultados experimentais indicaram um tempo de execução de 20% maior do que a seleção de convencional de um caminho de comunicação. Os autores

também aplicaram o algoritmo para um decodificador MPEG *System-on-Chip*, reduzindo a área em 33%, a dissipação de energia por 35%, e a latência de pior caso reduzida por um fator de quatro sobre uma abordagem tradicional de projeto.

Os autores (GOSSENS; BENNEBROEK; HUR; WAHLAH, 2008) propuseram no artigo "*Hardwired Networks on Chip in FPGAs to Unify Functional and Configuration Interconnects*" que as redes em chip (NoC) sejam interconectadas dentro de um FPGA (acrônimo do inglês *Field Programmable Gate Array*) por um número maior de conexões fixas. Apesar de algumas áreas do FPGA terem uma função fixa, esta perda de flexibilidade é compensada por outros benefícios. Neste artigo os autores oferecem uma visão detalhada da arquitetura de NoC proposta, sua configuração e programação. Segundo eles, o esquema proposto melhora o tempo de geração do arquivo de configuração do FPGA, conhecido como *bitstream*. Além disso, a configuração e programação da NoC proposta é muito mais rápida do que quando se utiliza uma NoC convencional. Aqueles autores utilizaram a rede Æthereal para realizar comparações com a NoC proposta.

Os autores (GOOSSENS; MHAMDI; SENIN, 2009) propuseram no artigo "*Internet-Router Buffered Crossbars Based on Networks on Chip*" usar uma rede multi-hop em um chip (NoC) em substituição ao uso de *crossbar* na interconexão intra-chip. Como benefícios do uso de uma NoC multi-hop, os autores destacaram a velocidade de conexão devido a possibilidade de arbitragem distribuída, a carga equilibrada devido a possibilidade de distribuição de fluxos por caminhos alternativos, a escalabilidade do sistema, entre outros. Os autores montaram protótipos de NoCs 32×32 utilizando tecnologia de 65 `nm`. Segundo aqueles autores, os protótipos podem operar na frequência de 413 `MHz`.

O artigo "*A quantitative evaluation of a Network on Chip design flow for multi-core consumer multimedia applications*", publicado pelos autores (HANSSON; GOOSENS, 2011), é focado em *Network-on-Chip* voltadas para aplicações de tempo real. A preocupação dos autores é que muitas aplicações têm requisitos de tempo real rígido ou requisitos mais flexíveis, porém todos eles exigem alguns limites sobre a latência e taxa de transferência dos dados na rede. Para acomodar o número crescente de requisitos de aplicação, a NoC deve oferecer escalabilidade no nível físico, arquitetônico e funcional. Como a principal contribuição deste trabalho, os autores demonstraram um fluxo completo de projeto de interconexão operacional para múltiplas aplicações em tempo real, e quantitativamente avaliaram a escalabilidade funcional em dois projetos industriais de larga escala. Os autores ilustraram os passos do fluxo,

desde a especificação de requisitos para todos os meios de simulação dos arquivos de *netlist* sintetizados nas tecnologias conhecidas como *low-power standard-cell technology* de 90 `nm` e 65 `nm`. A rede Æthereal foi usada como referência naqueles experimentos. Os autores demonstraram que o fluxo de projeto de interconexão oferece escalabilidade em nível físico, arquitetura, assim como em nível funcional.

No artigo "*The aethereal network on chip after ten years: goals, evolution, lessons, and future*" os autores (STEFAN; GOOSSENS, 2011) fazem uma retrospectiva sobre a rede Æthereal. Eles avaliam as metas iniciais proposta para a rede e discutem o que foi atingido até o ano de 2011, ano da publicação do artigo. Os autores fazem perguntas como: será que as metas propostas no início de 2001 serão cumpridas? O que resta dos conceitos iniciais? Neste trabalho, os autores respondem estas e outras questões, avaliam diferentes implementações, com base em uma nova métrica de projeto: análise de custos. O artigo é finalizado com questões ainda em aberto e direções futuras.

A Figura 17 mostra a linha do tempo dos artigos aqui apresentados. Note que foram incluídos na figura os artigos relativos as rede Aelite e dAElite, ambas são evoluções da rede Æthereal para resolver problemas específicos que estavam em aberto nesta rede. As duas seções que seguem fazem um detalhamento destas redes, Aelite e dAElite. Aquelas seções não terão uma subseção para tratar da evolução de cada uma delas, uma vez que tais redes não tiveram publicações posteriores àquelas citadas no início de cada seção.

2.6 AELITE

A rede Aelite foi proposta por Andreas Hanson and Kees Goossens (STEFAN; GOOSSENS, 2011) e é uma versão da rede Æthereal que oferece apenas mecanismos para o tratamento de fluxos GS, apresentando desta forma roteadores com menor uso de recursos de silício. Diferente de outras redes TDM, a Aelite permite tanto o uso de canais mesócronos como assíncronos, oferecendo desta forma uma escalabilidade em nível físico. Segundo os seus autores, a escalabilidade funcional é conseguida através de aplicações completamente isoladas, aliada ao fato de ter uma arquitetura de roteador que não limita o número de níveis de serviço ou de conexões.

O roteador da rede Aelite, apresentado na Figura 18, consiste de três estágios de *pipeline*, correspondendo a um tamanho de *flit* de três palavras. O primeiro estágio sincroniza os dados de entrada. Em se-

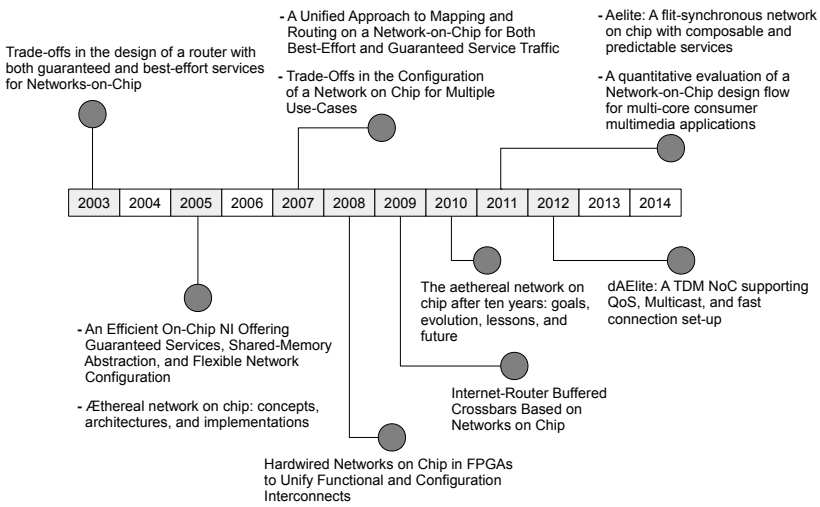


Figura 17 – Linha do tempo das publicações relacionadas com a rede Æthereal.

guida, um bloco Analisador de Cabeçalho, similar ao usado no roteador da rede Æthereal (*Header Parsing Unit - HPU* na Figura 18), determina o canal de saída baseado no caminho informado no cabeçalho do pacote. A Saída selecionada permanece a mesma até o recebimento do bit de Fim de Pacote (*End-of-Packet - EoP*) Diferente da arquitetura da Æthereal, os bits de *valid* e EOP são sinais de controle explícito e não precisam ser decodificados, o que remove o bloco HPU do caminho crítico dentro do roteador. Os números de canais de saída são codificados de modo *one-hot*[4] antes de serem encaminhados à *crossbar*, o que determinada as conexões entre canais de entrada e de saída. Com isso, três ciclos de *clock* após um *flit* ser sido apresentado a um roteador, ele aparece no canal de saída, conforme indicado por uma seta em linhas pontilhada no alto da Figura 18.

O roteador da Aelite é parametrizável apenas em seu número de bits do canal de dados, número de canais de entrada e número de canais

[4]Em circuitos digitais, *one-hot* refere-se à um grupo de bits de entre os quais as combinações legais dos valores são apenas aqueles com um único bit em nível lógico 1 e todos os outros em nível lógico 0.

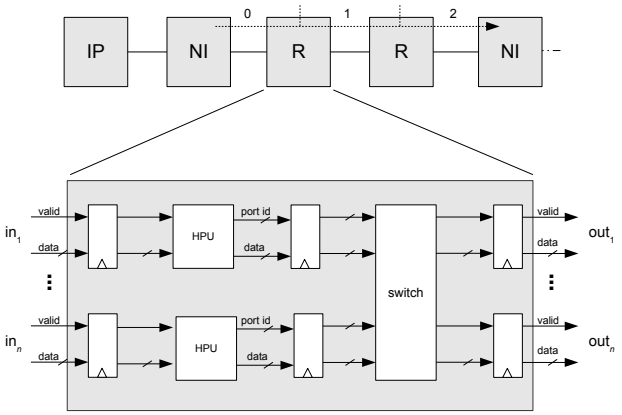


Figura 18 – Estrutura interna do roteador da Aelite. Fonte: adaptado de (HANSSON et al., 2009).

de saída (estes dois últimos podem ser diferentes entre sí). Não há tabela de roteamento no roteador e há apenas um *buffer* com capacidade de armazenamento de uma palavra (*word*) nos canais de entrada. Além disso, não há arbitragem no roteador, porque a contenção de recursos é eliminada através de um escalonamento dos *flits* feito em tempo de projeto (*off-line*).

A arquitetura do canal da rede Aelite consiste de uma FIFO bisíncrona e de uma máquina de estados (*Finite State Machine - FSM*, como mostrado na Figura 19. A FIFO ajusta as diferenças de fase entre os sinais de *clock* de escrita e de leitura, enquanto que a máquina de estados controla o fluxo de dados no canal. Os autores da Aelite consideraram que a diferença entre o *clock* de escrita e o de leitura é no máximo metade de um ciclo de *clock*, além de considerarem que as memórias FIFO possuem um atraso no encaminhamento dos dados menor que o número de palavras em um *flit* (de um a dois ciclos) e taxa nominal de uma palavra por ciclo de *clock*.

A máquina de estados apresentada na Figura 19 garante que são necessários sempre três ciclos de *clock* (no domínio do *clock* de leitura) para um *flit* atravessar o canal. Conforme ilustrado naquela Figura por uma seta pontilhada, esta manipulação de *flits* introduz uma latência adicional. No entanto os três ciclos são suficientes para absorver a latência da FIFO e a diferença de fase entre os sinais de *clock* de escrita e de leitura. Além disso, como a diferença de fase é garan-

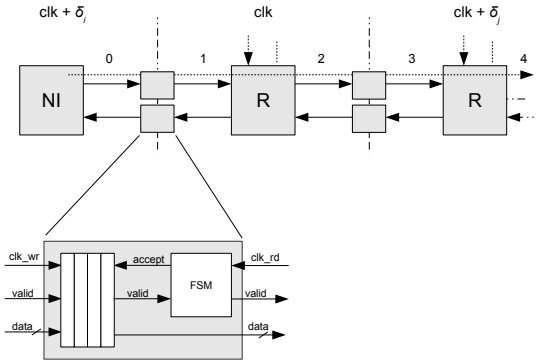


Figura 19 – Estrutura interna do link da Aelite. Fonte: adaptado de (HANSSON et al., 2009).

tida dentro de certos limites, a FIFO é definida com uma capacidade de armazenamento suficiente para nunca ficar completamente cheia (4 palavras).

Para permitir que as interfaces de rede (NI) e os roteadores da rede pudessem trabalhar com canais assíncronos, os autores da Aelite *encapsularam* os elementos da rede com um *wrapper*, como no exemplo apresentado na Figura 20. O *wrapper* trabalha de forma síncrona, porém deixando a complexidade no tratamento do domínio de *clock* para os canais de comunicação. Ele consiste de uma interface de canal (*Port Interface* - PI), um controlador para tal interface de canal (*Port Interface Controller* - PIC) e o elemento da rede em si, que no caso daquela Figura é um roteador.

Cada canal do roteador é gerenciado por um bloco PI separado. Na Figura 20 os blocos PI relativos aos canais de entrada recebem a denominação de IPI, enquanto que os canais de saída são representados por OPI. Cada IPI e OPI é constituído de uma FIFO síncrona e associada a um contador. Na IPI o contador registra quantas palavras estão presentes na FIFO. Por outro lado, o contador disponível nas interfaces OPI refletem quanto espaço ainda não foi reservado na FIFO de saída. Portanto, o contador em uma OPI é decrementado tão logo um dado da FIFO tenha sido encaminhado para o roteador, ao invés de registrar a chegada de novo dado na FIFO. As interfaces IPI e OPI sinalizam o controlador PIC quando ao menos um *flit* ou um espaço para um *flit* está disponível, respectivamente. É o controlador PIC que garante o

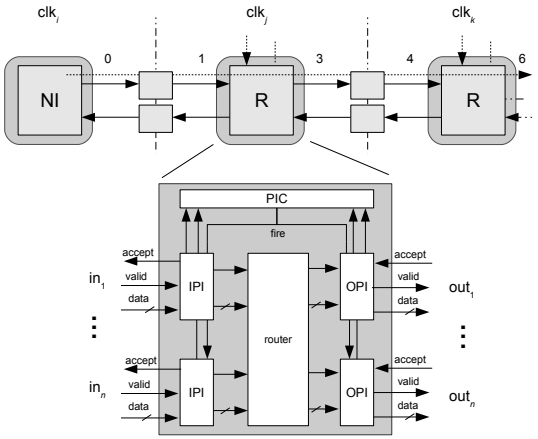


Figura 20 – Estrutura interna do *wrapper* da rede Aelite. Fonte: adaptado de ().

sincronismo na transferência de *flits* através do roteador.

2.6.1 Propriedades para análise de WCL

De acordo com os autores da rede Aelite, ela possui as mesmas propriedade para análise de WCL que a rede Æthereal (MLA, 2012). Por este motivo, o Tempo de espera, a vazão e a latência não serão tratados nesta subseção, pois tais valores são similares aos apresentados na Subseção 2.5.1.

2.6.2 Resumo

A rede Aelite oferece escalabilidade em nível físico, arquitetural e funcional. Diferente de sua antecessora, a rede Æthereal, ela trata apenas de fluxos GS. Como vantagens da rede Aelite sobre a rede Æthereal, destaca-se sua a capacidade de interconectar núcleos através de canais mesócronos ou assíncronos, além de apresentar um menor custo de silício. Porém, uma deficiência na abordagem adotada no tratamento de canais assíncronos é a necessidade da rede operar obrigatoriamente na mesma frequência que a interface de rede (NI) mais lenta colocada

na rede. Este fato precisa ser levado em consideração se o sistema for de tempo real.

2.7 DAELITE

A rede dAelite foi proposta por Angelo Ambrose *et al.* (GOOSSENS; DIELISSEN; RADULESCU, 2012) como uma NoC que oferece largura de banda e latência garantidas garantidas por conexão, além de comunicação *multicast* e tempo de conexão mais baixo. A largura de banda, latência e comunicação *multicast* são alcançados devido ao uso de mecanismo de conexão TDM. De fato, a dAelite é uma rede baseada na tecnologia da rede Æthereal e o tempo de conexão é o seu principal diferencial em relação às redes TDM que a antecederam (Æthereal e Aelite).

Tanto a rede Ælite como Aelite necessitam que os núcleos conectados naquelas redes requisitem as conexões TDM necessárias para realizar suas transferências de dados, demandando um tempo adicional antes que a comunicação esteja estabelecida apenas para a transferência de dados. Tais requisições estão relacionadas aos seguintes tempos: *(i)* o tempo gasto para ter acesso à rede; *(ii)* o tempo de envio de uma requisição de transferência de dados através da rede; e *(iii)* o tempo para enviar a resposta através da rede, de volta para o nodo requisitante. Alem disso, uma vez que um determinado núcleo não necessita mais de uma conexão para transferir seus dados, ele devem liberar o canal alocado, gerando nova troca de dados para a desconexão.

A Figura 21 mostra um exemplo de plataforma que foi concebida pelos autores da rede dAelite para lidar com os tempos de alocação e desconexão de canais de comunicação. Naquela Figura, os núcleos são conectados à interface de rede através de barramentos locais e representados por retângulos com a letra S, os quais fazem a multiplexação/demultiplexação dos dados a serem enviados ou recebidos de outros núcleos também conectados na rede.

O mecanismo de configuração da rede foi implementado como uma rede dedicada para *broadcast* com uma topologia em árvore, com canais de comunicação operando em paralelo a um sub-conjunto dos canais normais de rede de dados. Um núcleo, denominado de *host*, tem o controle exclusivo sobre a infraestrutura de configuração através de um módulo de configuração. O conjunto de canais que formam a árvore de configuração é escolhido de forma a minimizar a distância do *host* para qualquer outro ponto da rede. A estrutura de configuração é

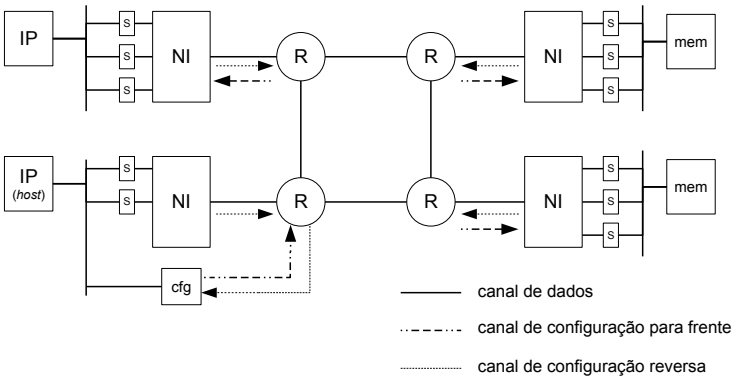


Figura 21 – Exemplo de uma plataforma com a rede dAelite. Fonte: adaptado de (GOOSSENS; DIELISSEN; RADULESCU, 2012).

usada para estabelecer as conexões de dados atualizando as tabelas de *slots* dentro dos roteadores e interfaces de rede (NI), para configurar e ler o estado das interfaces de redes e para configurar os barramentos adjacentes a rede.

Os canais de configuração consistem de uma conexão de encaminhamento, representada por setas com três traços e dois pontos na Figura 21, e outra conexão reversa, representada por setas pontilhadas na mesma Figura. A conexão de encaminhamento é do tipo *broadcast*, ou seja, cada nodo encaminha os dados recebidos em seus canais de entrada para todos os seus canais de saída. A resposta converge para o caminho da conexão reversa na mesma estrutura da árvore. Não há arbitragem no caminho de resposta e, como resultado, uma política de apenas uma requisição ativa por vez é aplicada para a rede.

A estrutura do roteador da rede dAelite é apresentada na Figura 22. O mecanismo de roteamento é distribuído e cada roteador contém uma tabela de *slot* para armazenar o escalonamento TDM. Pacotes que chegam são roteados de acordo com esta tabela de roteamento. Devido à inexistência de contenção que o mecanismo TDM da rede apresenta, não há necessidade de controle de fluxo em nível de canal. Um submódulo de configuração interpreta as mensagens recebidas através dos canais de configuração e atualiza as tabelas. Os roteadores também são nodos na árvore de *broadcast* de configuração, e eles encaminham os seus dados de configuração para um número parametrizável de nodos

vizinhos.

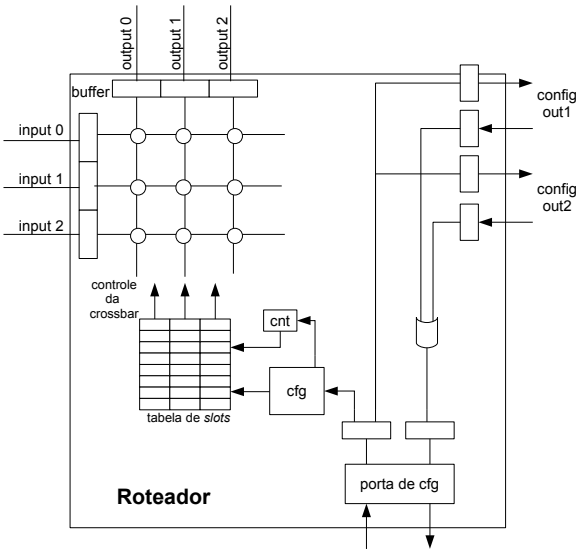


Figura 22 – Estrutura interna do roteador da rede dAelite. Fonte: adaptado de (GOOSSENS; DIELISSEN; RADULESCU, 2012).

A latência da rede dAelite é fixada em dois ciclos por roteador, sendo um ciclo para o canal transversal e um para a *crossbar* transversal. Os dados são então armazenados duas vezes dentro do roteador. Por motivos de simetria, os dados também são armazenados duas vezes em cada *hop* de configuração da rede.

2.7.1 Propriedades para análise de WCL

Assim como a rede Aelite, a rede dAelite pode ser vista como uma evolução da rede Æthereal, pois todas elas são baseadas nos mesmos conceitos de alocação temporal e roteamento livre de contenções. A diferença está no fato da dAelite possuir canais específicos para estabelecer conexões e que trabalham em paralelo com os canais de transferência de dados na rede. Desta forma, o estabelecimento de conexão ou desconexão podem ser planejados de modo a minimizar o tempo total de conexão. Assim, como a configuração pode ser estabelecida em paralelo à operação da rede, isto resulta em um tempo de espera nulo

para acessar a rede. Portanto os parâmetros $T_{wait;req}$ e $T_{wait;reply}$ são zero.

Os demais parâmetros, latência e vazão, são os mesmos da rede Æthereal. Por este motivo, eles não serão tratados nesta subseção, pois são similares aos apresentados na Subseção 2.5.1.

2.7.2 Resumo

O principal benefício apresentado pela rede dAelite em relação às suas predecessoras Æthereal e Aelite, é a redução no tempo de estabelecimento de conexão e desconexão de canais do TDM. Tal redução está baseada no uso de canais de comunicação específicos, de modo que a rede opera de modo paralelo, entre estabelecimento de conexão e transferência de dados. De acordo com os seus autores, Angelo Ambrose *et al.* (GOOSSENS; DIELISSEN; RADULESCU, 2012), apesar de utilizar mais canais para minimizar os tempos de conexão e desconexão, a rede dAelite apresenta um baixo consumo de recursos de silício, quando comparada com as soluções da Æthereal e Aelite.

2.8 RESUMO DOS TRABALHOS RELACIONADOS

A partir das informações apresentadas na Seção 3.3, é possível concluir que a contribuição de uma NoC no tempo total de execução de uma aplicação que esteja sendo executada em um dos nós da rede pode ser baseada em três métricas:

- a latência de um pacote para atravessar a rede de um ponto a outro;
- a vazão da rede, como a taxa na qual pacotes podem ser inseridos na rede em um ciclo de relógio (ou *clock*); e
- o tempo de espera ($T_{wait;req}/T_{wait;reply}$) até que o acesso à rede possa ser efetivado.

Um conjunto de NoCs que compõem o estado da arte em redes com previsibilidade de latência foi analisado neste Capítulo, levando em conta as três métricas acima citadas. Para oferecer tal previsibilidade, os autores das rede analisadas empregaram técnicas que oferecem alguma forma de conexão fim-a-fim. De um modo geral, tais técnicas estão relacionadas a roteadores não bloqueantes com controle de fluxo

e chaveamento de circuitos, algumas vezes adotando mecanismos de Multiplexação por Divisão de Tempo (*Time Division Multiplexing* - TDM).

As redes SoCBUS e 4S são exemplos de NoCs que adotam a abordagem de chaveamento de circuitos. As conexões, quando estabelecidas, não compartilham os recursos alocados para estabelecer tais conexões, de modo que as garantias de tempo real podem facilmente ser atendidas para os fluxos que trafegam naqueles circuitos. Porém, o não compartilhamento de recursos alocados podem resultar em baixa utilização dos recursos da rede.

A técnica de TDM é uma outra opção de chaveamento de circuitos, empregadas nas redes Æthereal e Nostrum. Esta técnica implementa um conceito de chaveamento de circuitos virtuais,o que pode resultar em um melhor aproveitamento dos recursos da rede. Na implementação de TDM é necessário uma noção de tempo comum para toda a rede, e este é um desafio que consiste em preservar a sincronização de todo o projeto do SoC. Para lidar com este problema, canais mesócronos foram utilizados por aquelas redes.

Uma alternativa ao chaveamento de circuitos foi apresentada pelos autores da rede MANGO. Eles apresentaram o conceito de roteadores não bloqueantes com controle de taxa, no qual os pacotes são arbitrados e priorizados localmente nos roteadores para atingir as garantias temporais necessárias para os fluxos.

As informações sobre as redes que foram apresentadas neste capítulo foram organizadas na Tabela 1. A lista de parâmetros a seguir (Legenda) apresenta a definição de cada parâmetro ou símbolo usado naquela Tabela.

Legenda:

* CC - Chaveamento de Circuitos.
‡ RNB - Roteador sem bloqueio.

h - número de saltos (*hops*) entre roteadores em um caminho na rede.
B - tamanho do *buffer* nos roteadores.
n - número de pacotes em uma transação.
f - número de *flits* em um pacote.
b - número de bits em um pacote.
l - pistas (*lanes*) alocadas para um circuito virtual.
L - tamanho das *lanes* em bits.
t - *containers* alocados para um canal virtual.
T - tamanho máximo de um caminho, ida e volta (*loop*).
p - *time slots* alocados para um circuito virtual.
P - período dos *time slots* em um escalonamento.
s - duração de um *time slot* em ciclos de *clock*.
C - número de canais virtuais em um enlace.

Tabela 1 – Resumo das NoCs estudadas e que apresentam mecanismos para garantir WCL.

	Network-on-Chip					
	SoCBUS	**Nostrum**	**Æthereal/Aelite**	**MANGO**	**4S Project**	**dAElite**
Ano	2004	2004	2005/2010	2005	2006	2012
Técnica	CC*	TDM	TDM	RNB‡	CC*	TDM
WCL	Durante o projeto.	Durante o projeto.	Durante o projeto.	Durante o projeto.	Durante o projeto.	Durante o projeto.
Vantagens	- Alta vazão para blocos de dados;	- Vazão garantida devido ao uso de TDM; - Latência previsível.	- Um pacote por ciclo de clock; - Fluxos BE podem usar canais GS sem uso momentâneo (apenas na AEthereal). - Latência previsível.	- Tempo de espera para inicio de transmissão é nulo; - Flexibilidade no ajuste de parâmetros temporais devido ao fluxo local.	- Alta vazão para blocos de dados; - Tempo de espera para inicio de transmissão é nulo;	- Um pacote por ciclo de clock; - Fluxos BE podem usar canais GS sem uso momentâneo; - Latência previsível. - Menor tempo para alocação de canal.
Limitações	- Não compartilha recursos alocados; - Alocação de recursos não é garantida. - Garantia de latência exige análise conhecimento prévio dos fluxos da rede.	- Tempo adicional para alocação de canal; - Vazão é inversamente proporcional ao período do TDM. - Garantia de latência exige análise conhecimento prévio dos fluxos da rede.	- Vazão é inversamente proporcional ao período do TDM. - Tempo adicional para alocação de canal; - Garantia de latência exige análise conhecimento prévio dos fluxos da rede.	- Vazão é inversamente proporcional ao número de VC. - Garantia de latência exige análise conhecimento prévio dos fluxos da rede.	- Garantia de latência exige análise conhecimento prévio dos fluxos da rede. - Não compartilha recursos alocados;	- Garantia de latência exige análise conhecimento prévio dos fluxos da rede. - Vazão é inversamente proporcional ao período do TDM.
Wait time ($T_{wait;req}$) ($T_{wait;reply}$)	 $4.h + h$ 0	 $T - t$ $T - t$	 $(P - p).s$ $(P - p).s$	 0 0	 0 0	 0 0
Latência ($T_{latency}$)	 h	 $h.B$	 $h.B + f$	 $(h + f).(C + B)$	 $h.B + b/(L.l)$	 $h.B + f$
Vazão	$1packet/cycle$	$(n.T)/t$	$(p.n)/(P.s)$	$n.f.C$	$(n.L)/(b.l)$	$(p.n)/(P.s)$

PARTE III

RTSNOC

3 MODELAGEM DE REDE PARA FLUXOS DE MELHOR ESFORÇO EM SISTEMAS DE TEMPO REAL

Sistemas de tempo real que utilizam redes para à conexão de múltiplos processadores precisam levar em conta que a latência de comunicação entre dois pontos quaisquer da rede não é necessariamente constante. Por exemplo, uma tarefa em execução em um processador conectado em um nodo da rede poderá experimentar latências diferentes ao acessar um determinado recurso externo ao processador e conectado em outro nodo, devido à influência dos diversos fluxos de tráfego da rede. Assim, a latência envolvida na comunicação em uma rede depende de como ela foi implementada e pode depender também de interferências causadas por tráfegos independentes que circulam na rede. As Seções que seguem apresentam definições, terminologias e modelos necessários para a análise de latência em um sistema de tempo real que utiliza uma NoC como mecanismo de comunicação entre núcleos de processamento, os quais são utilizados nas análises da rede proposta nesta Tese.

3.1 MODELO DE FLUXO DE TRÁFEGO

Define-se aqui que uma NoC compreende um conjunto Γ de n fluxos de tráfego $\Gamma = \{\sigma_1, \sigma_2, ...\sigma_n\}$ e cada fluxo de tráfego σ_i é formalmente representado pela seguinte 7-tupla de atributos:

$$\sigma_i = (v_{si}, v_{di}, P_i, T_i, D_i, L_i, f_i) \tag{3.1}$$

Esta tupla descreve um fluxo de tráfego σ_i de um nodo de origem v_{si} para um nodo de destino v_{di} com prioridade P_i, período constante T_i entre transmissões de pacotes sucessivos, *deadline* para entrega do pacote D_i, latência de um *flit* para ser transmitido entre os nodos de origem e destino na rede L_i e tamanho do pacote dado pelo número de *flits* f_i que o compõem.

Observe que o modelo de fluxo σ_i não contempla o *jitter* de lançamento dos pacotes na rede (J_i^R). Neste modelo está sendo assumido que tal *jitter* é nulo, eliminando assim a influência de atrasos no envio dos pacotes periódicos provocados pelos núcleos de processamento, e que estão fora do escopo deste trabalho. Cada fluxo de tráfego σ_i possui um valor numérico que define seu nível de prioridade

P_i. Todos os pacotes que pertencem ao fluxo σ_i herdam a mesma prioridade P_i. Assume-se aqui que o fluxo de tráfego é priorizado por uma política de escalonamento implementada nos roteadores da rede modelada, e que tal política leva em conta três premissas: *(i)* pacotes pressupõem a requisição de recursos da rede para serem transmitidos; *(ii)* a transmissão de um pacote que já possui recursos alocados não pode sofrer preempção; e *(iii)* os recursos da rede devem ser liberados ao finalizar a transmissão de um pacote. A questão sobre qual o escalonamento adotado na rede está fora do escopo desta análise.

Cada fluxo de tráfego possui como restrição um *deadline* D_i, o que significa que todos os pacotes pertencentes ao fluxo σ_i devem ser entregues do roteador de origem até o roteador de destino dentro do limite máximo estabelecido por D_i, mesmo em situação de congestionamento (pior caso). Além disso, o modelo apresentado tem como restrição que cada fluxo de tráfego σ_i deva ter seu *deadline* menor ou igual ao seu período, ou seja $D_i \leq T_i \ \ \forall \ \sigma_i$.

Para Buttzzo (BUTTAZZO, 2005) esta condição evita a ocorrência de "auto-bloqueio"dos recursos pelo fluxo, porém não evita que este fluxo cause a perda de *deadline* de outros fluxos que necessitam dos mesmos recursos. A partir do modelo de fluxo de tráfego representado pela 7-tupla, é possível definir uma condição para a qual os fluxos pertencentes à um subconjunto de fluxos, representado por $\gamma = \{\sigma_1, \sigma_2, ...\sigma_k\} \in \Gamma$, e que competem pelos mesmos recursos da rede têm seus *deadlines* atendidos. Para isso, é necessário assumir as seguintes premissas:

- a largura de banda BW[1] é dividida entre os k fluxos que competem pelos mesmos recursos da rede, de modo igualitário ($BW_i = \frac{BW}{k}$). Para isso, está sendo considerado que todos os pacotes, de todos os fluxos pertencentes ao conjunto γ, tem o mesmo tamanho (ou seja: $f_1 = f_2 = ... = f_k$);

- não há uso de algoritmos de roteamento adaptativo nos roteadores da rede, de modo que o caminho de roteamento entre dois nodos quaisquer da rede é sempre o mesmo; e

- os k fluxos pertencentes ao subconjunto γ são fluxos de melhor esforço (BE), e por este motivo todos têm a mesma prioridade P_i.

[1] A largura de banda pode ser definida como a taxa de dados, em bits por segundo, que a rede aceita por canal de comunicação. (DALLY WILLIAM. J.; TOWLES, 2001)

Sabendo que cada pacote pertencente a um fluxo σ_i possui f_i *flits*, e que cada *flit* pertencente à σ_i compete com *flits* pertencentes à outros `k` fluxos pelos mesmos recursos da rede para ser enviado de um nodo v_{si} ao nodo v_{di}, então o tempo máximo para que um pacote seja enviado entre estes nodos deve ser:

$$(\frac{k \times f}{BW_i}) \tag{3.2}$$

Então, assumindo-se que o tempo máximo para um pacote ser transmitido deva ser menor ou igual ao seu *deadline*, tem-se a inequação:

$$(\frac{k \times f}{BW_i}) \leq D_i \tag{3.3}$$

a qual define o limite máximo de tempo para que um pacote pertencente ao fluxo σ_i seja entregue ao seu nodo de destino.

3.2 ANÁLISE DE LATÊNCIA

Esta seção apresenta os conceitos básicos de latência com o objetivo de tornar claro como o método de intercalação de *flits* pode ser adequado para melhorar a latência média de fluxos com taxa de bits variável que competem por um mesmo canal de comunicação da rede.

Uma NoC dedicada para plataformas em tempo real deve assegurar garantia de latência para comunicação individual entre dois núcleos quaisquer na rede. Os autores William J. Dally e Brian Towles (DALLY WILLIAM. J.; TOWLES, 2001) definem a latência da rede como sendo o tempo necessário para que um pacote atravesse a rede, a partir do momento em que o cabeçalho do pacote chega ao canal de entrada do roteador de origem até o momento em que o terminador do pacote sai pelo canal de saída do roteador de destino do pacote. Segundo aqueles autores, a latência pode ser dividida em dois componentes:

$$L = T_h + \frac{F}{b} \tag{3.4}$$

sendo que `L` é a latência do pacote, $\mathtt{T_h}$ é o tempo necessário para o cabeçalho do pacote atravessar a rede e $\frac{\mathtt{F}}{\mathtt{b}}$ é o tempo para que o restante do pacote de comprimento `F` para atravessar um canal com largura de banda `b`. Na ausência de contenção, a latência de cabeçalho pode ser visto como a soma de dois fatores determinados pela topologia:

o atraso dentro do roteador, e o número de roteadores em um caminho entre a origem e destino. Com base nesses dois fatores, a equação 3.4 pode ser reescrita da seguinte forma:

$$L = (H_{path}.t_r) + \frac{F}{b} \tag{3.5}$$

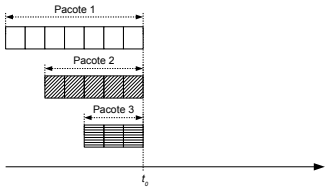
sendo que $\mathtt{H_{path}}$ é o número de saltos (*hopes*) no caminho entre a origem e o destinatário do pacote e $\mathtt{t_r}$ é o atraso causado por cada roteador naquele caminho. Por uma questão de simplificação na análise, nesta Tese não estão sendo incluídos nas equações 3.4 e 3.5 os outros fatores considerados por William J. Dally e Brian Towles como o atraso inerente aos "fios" que formam os canais físicos, a distância física entre a origem e o destino de um pacote dentro do SoC, ou a velocidade de propagação dos dados na rede.

3.2.1 Latência utilizando intercalação de flits

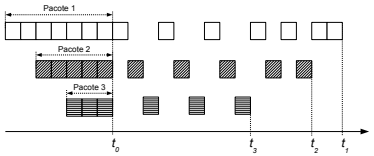
Suponha que há três solicitações para enviar pacotes através de um mesmo canal de comunicação no instante $\mathtt{t_0}$, como mostrado na Figura 23(a), e a sequência de escalonamento para estas solicitações no instante $\mathtt{t_0}$ é {`pacote 1`, `pacote 2`, `pacote 3` }. Define-se aqui a "intercalação de *flits*" como sendo o método em que os pacotes de todos os pedidos são divididos em *flits*, e estes *flits* são enviados através do canal de comunicação, sendo encaminhado um *flit* de cada pacote a cada ciclo de transmissão. A intercalação de *flits* para este exemplo é mostrado na Figura 23(b) e um exemplo de chaveamento *wormhole* é mostrado na Figura 23(c).

Note que o tempo para o pacote `3` "atravessar" o canal usando intercalação de *flits* é ($\mathtt{t_3}$ - $\mathtt{t_0}$), o qual é menor do que o tempo gasto no chaveamento *wormhole*. Isso significa que pacotes de tamanhos menores têm latência média menor quando o método de intercalação de *flits* é utilizado, enquanto os pacotes de tamanho maiores têm um acréscimo na sua latência média.

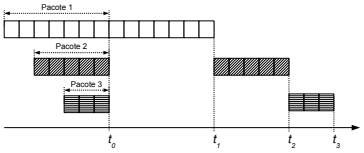
Esta é uma relação custo-benefício quando a intercalação de *flits* é adotada. Lembre-se que os sistemas abordados neste trabalho têm fluxos com taxa de bits variável, e os pacotes `1`, `2`, e `3` neste exemplo deverão ter tamanhos de pacotes diferentes ao longo do tempo. Um exemplo de aplicação multimídia com taxa variável de bits é apresentado no Capítulo 6, no qual fluxos multimídia com taxa de bits variável competem pelos mesmos recursos de rede.



(a) Três pacotes de diferentes fluxos estão requisitando o mesmo canal de comunicação.



(b) Os *flits* dos três pacotes são intercalados no canal de comunicação.



(c) Exemplo de um chaveamento *wormhole* para os mesmos três pacotes.

Figura 23 – Exemplo de intercalação de *flits* em um mesmo canal de comunicação.

3.2.2 Contenção de recursos de rede e a Latência

Lembre que a Equação 3.5 baseia-se no pressuposto de ausência de contenção de recursos na rede. A presente Subsecção apresenta uma avaliação daquela Equação em duas redes hipotéticas: uma rede com suporte para fluxos BE e outra baseado na intercalação de *flits*. Ambas as redes foram avaliadas considerando-se vários valores de tráfego oferecido.

Para redes de melhor esforço, os canais de comunicação são compartilhadas por vários fluxos. Foram feitas simulações com a equação 3.5 para as redes hipotéticas adotando como pressuposto que a rede BE foi implementada com um árbitro round-robin e chaveamento *wormhole*. A latência para um fluxo σ_i foi calculada de acordo com a seguinte equação:

$$L = (H_{path}.t_r) + \frac{F}{|b - b_{occupied}|} \tag{3.6}$$

sendo que $|b - b_{occupied}|$ é a largura de banda disponível para o fluxo em análise σ_i, levando em conta que $b_{occupied}$ da largura de banda

inteira b foi utilizada por outros fluxos (tráfego oferecido). Lembre-se que para o chaveamento *wormhole*, se um pacote pertencente a um fluxo σ_i solicita um canal de comunicação que está sendo usado por outro pacote pertencente a um fluxo σ_j, ele deve esperar o pacote pertencente a σ_j terminar a transmissão e liberar o canal de comunicação antes de começar a sua transmissão.

A expressão de latência para uma rede hipotética baseada na intercalação de *flits* deve levar em conta que não há nenhuma reserva de recursos na rede, como acontece no caso da rede com chaveamento *wormhole*, e os *flits* de um pacote pode ser intercalado com *flits* de outros pacotes que estejam requisitando os mesmos recursos de rede. Assim, a latência do pacote analisado deve considerar que `N` pacotes podem competir em cada roteador no caminho, desde sua origem até o seu destino ($\mathtt{H_{path}}$). Isso significa que, sob o ponto de vista de latência, o tamanho do pacote cresce `N` vezes. Assim, a expressão de latência é dada como segue:

$$L = (H_{path}.t_r) + N.\frac{F}{b} \tag{3.7}$$

Uma simulação de um SoC usando essas redes hipotéticas foi feita com base nas equações 3.6 e 3.7. Isso foi feito considerando-se as seguintes condições:

- O número de roteadores no caminho das redes sob análise é `4` para ambas;
- O atraso causado por cada roteador em ambas as redes é o mesmo, e foi definido como sendo `3`;
- Existem três pacotes pertencentes a fluxos diferentes requisitando os mesmos recursos da rede, em ambas as redes. Um destes fluxos, chamado aqui de σ_i, é o fluxo sob análise cujos pacotes possuem tamanhos fixos de `100` *flits*, e os outros dois fluxos possuem pacotes com tamanhos variáveis, com valores entre `0` até `64` *flits*.

A Figura 24 apresenta os resultados gerados pela simulação com a equação 3.6 (cor cinza) e Equação 3.7 (cor preta). Como esperado, a latência para o σ_i é a mesma, quando não há outros fluxos solicitando os mesmos recursos da rede (tráfego oferecido = `0`). Com outros dois fluxos competindo pelos mesmos recursos, a latência para σ_i na NoC baseada na intercalação de *flits* cresce quase três vezes; no entanto, mantém-se constante até o uso máximo da largura de banda. Por outro lado, a latência de σ_i na rede de melhor esforço cresce consideravelmente

quando a carga oferecida à rede é cerca de 70%, como resultado de congestionamento da rede para o fluxo em análise. O resultado para a rede baseada na intercalação de *flits* era esperado porque a latência depende do número de *flits* do fluxo em análise, e da quantidade de fluxos que requerem os mesmos recursos, como mostrado na equação 3.7 e representado de modo genérico na Figura 23(b).

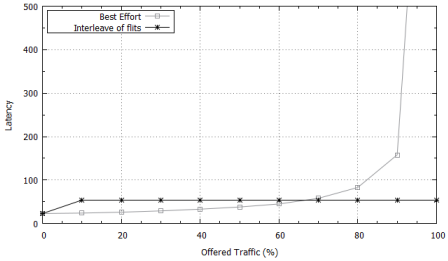


Figura 24 – Simulação de latencia em função do tráfego oferecido para uma rede hipotética BE e outra baseada na intercalação de *flits*.

Resultado semelhante foi apresentado por (VAN DEN BRAND et al., 2007), como mostra a Figura 25. Aqueles autores realizaram experimentos relacionando a latência da rede de uma conexão BE em função do tráfego oferecido medido para uma única ligação na NoC Æthereal. O gráfico mostra que a latência é pequena e quase constante até um determinado ponto, após o qual a latência cresce abruptamente. Este ponto fica em torno de 50% da carga oferecida, antes de saturar. Nesse exemplo, a latência satura em um valor abaixo de 500 `ns` (em torno de 1200 MB/s) e, segundo aqueles autores, isto acontece porque filas (*buffers*) entre as unidades de processamento e interfaces de rede não foram levados em conta na análise.

Com base nas informações expostas nesta seção, é possível afirmar que o método de intercalação de *flits* pode reduzir a latência média sob alto tráfego em SoCs dedicados para multimídia com taxa de bits variável, o que atende ao problema de pesquisa apontado na Seção 1.3.

3.3 ANÁLISE DE LATÊNCIA EM UMA NOC EM SISTEMAS DE TEMPO REAL

Esta seção apresenta uma classificação da latência direta que uma NoC impõem ao tempo de computação de uma tarefa que está

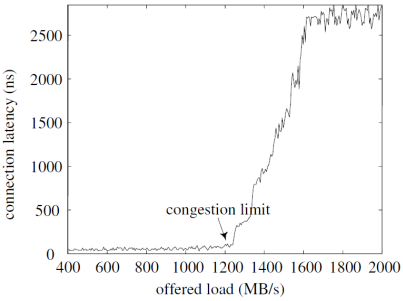


Figura 25 – Latência de rede para uma conexão BE na NoC Æthereal em função do tráfego oferecido. Referência: (VAN DEN BRAND et al., 2007)

sendo executada em um nodo qualquer da rede. Os tipos de acessos a uma NoC podem ser descritos em termos da quantidade de dados que são transferidos em cada acesso. Os autores Kees Goossens et al. (MLA, 2012) sugerem que estes acessos pode ser transferências de uma *palavra* de dados ou de transferência de blocos de dados. Levando-se em consideração os tipos de transações possíveis com estes tipos de acessos, i.e. leitura ou escrita, quatro tipos de transferências de dados podem ser identificados:

- leitura de uma palavra de dados;
- escrita de uma palavra de dados;
- leitura de um bloco de dados; e
- escrita de um bloco de dados.

Pode-se considerar que tais acessos poderiam ser acessos a núcleos genéricos que demandam por transferência de dados. O tempo de execução de uma transferência de dados envolvendo um acesso a rede inclui o tempo necessário para que um determinado dado atravesse a rede (T_{noc}) e o tempo necessário para que o núcleo de destino acesse o dado recém chegado ao roteador de destino (T_{core}):

$$T_{transaction} = T_{noc} + T_{core} \tag{3.8}$$

O tempo definido como T_{core} depende das características dos núcleos conectado na rede. Uma transferência de dados na NoC pode incluir até três tipos de atrasos. Um atraso está relacionado ao tempo

gasto esperando para ter acesso a rede ($T_{wait;req}$). Outro atraso está relacionado com o envio de uma requisição de transferência de dados através da rede (T_{req}). Por fim, uma resposta deveria ser enviada de volta, dependendo do caso, que também demanda algum tempo para ter acesso a rede ($T_{wait;reply}$) e algum tempo para enviar a resposta através da rede (T_{reply}), de volta para o nodo requisitante.

Em geral, a contribuição de uma NoC para a latência em uma transferência de dados pode ser expressa por:

$$T_{noc} = L = T_{wait;req} + T_{req} + T_{wait;reply} + T_{reply} \tag{3.9}$$

A equação 3.9 é uma expressão geral e pode ser adaptada de acordo com uma dos quatro tipos de transferências citados acima. Por exemplo, em uma escrita ou leitura de uma palavra de dados os termos T_{req} e T_{reply} envolve a latência de um pacote ponto-a-ponto e pode ser substituído por $T_{latency}$. O processo de escrita ou leitura de blocos de dados depende do número de pacotes em cada bloco (n) e a taxa de transmissão na qual os pacotes podem ser injetados na rede (vazão ou em inglês *throughput*), e é dada pela relação $n/throughput$.

A Tabela 2 apresenta as expressões derivadas da Equação 3.9 para os tipos de transferências de dados apresentados. Note que naquela Tabela existem duas expressões para escrita de palavras de dados e escrita de bloco de dados. Para ambos os tipos de transferências, o parâmetro T_{reply} é um pacote que representa um sinal de aceite (em inglês *acknowledge*) por parte do nodo destino, enquanto que em modos de escrita sem *acknowledge* tal parâmetro pode ser ignorado.

Tipo de transação	Expressão para a latência L
Leitura de palavra de dados	$T_{wait;req} + T_{latency} + T_{wait;reply} + T_{latency}$
Escrita de palavra de dados (com *acknowledge*) (sem *acknowledge*)	 $T_{wait;req} + T_{latency} + T_{wait;reply} + T_{latency}$ $T_{wait;req} + T_{latency}$
Leitura de blocode dados	$T_{wait;req} + T_{latency} + T_{wait;reply} + n/throughput + T_{latency}$
Escrita de blocode dados (com *acknowledge*) (sem *acknowledge*)	 $T_{wait;req} + n/throughput + T_{latency} + T_{wait;reply} + T_{latency}$ $T_{wait;req} + n/throughput + T_{latency}$

Tabela 2 – Contribuição para a latência em uma NoC de acordo com o tipo de transação. Referencia: (MLA, 2012)
.

3.4 CONSIDERAÇÕES

O objetivo deste Capítulo foi apresentar um modelamento formal para a análise da latência adicional que uma NoC pode oferecer ao tempo de execução de programas em núcleos de processamento conectados aos nodos de uma NoC. Inicialmente foi definido formalmente um fluxo na rede, através de uma 7-tupla de atributos, seguido da definição de latência direta. Em seguida, foram analisadas as possíveis interferências que um fluxo pode sofrer em uma NoC, as quais foram classificadas como interferências direta e indireta. A partir desta definições, foi apresentado um modelo formal de pior caso de latência em uma NoC (WCL). A latência direta em uma rede intra-chip depende do tipo de transferência de dados que está sendo realizado, e por este motivo, foi apresentada uma classificação de tipos de transferências possíveis em NoCs proposto na literatura, logo após a definição formal de WCL. O próximo Capítulo apresenta uma revisão sobre redes intra-chip que oferecem previsibilidade de latência na comunicação, atributo importante para a sua utilização em SoCs de tempo real.

4 PROJETO LÓGICO DA REDE

Este capítulo apresenta a arquitetura de uma rede-em-chip denominada RTSNoC (acrônimo de *Real-Time Services Netowrk-on-Chip*), a qual possui topologia direta 2-D. Ela é uma *Network-on-Chip* com previsão de latências e projetada para sistemas embarcados de tempo real, garantindo uma latência de pior caso para fluxos de tempo real *hard* e melhorando a latência média para fluxos de tempo real *soft* na rede. Para alcançar tal objetivo, foi proposto um chaveamento por intercalação de *flits* em um mesmo canal de comunicação, de modo que todo *flit* contém informações de roteamento, associado à um algoritmo de escalonamento e uma alocação de memória feita apenas nos pontos finais de conexão da rede. O Capítulo começa com a definição de escopo do trabalho que foi desenvolvido para esta Tese. Em seguida é descrito o desenvolvimento da NoC proposta, no qual são apresentados detalhes e discussões a respeito das escolhas de projeto para a rede, como topologia, enlaces, arbitragem, chaveamento, rotamento, formato do pacote, etc.

4.1 CONCEITO DA REDE RTSNOC

O que foi percebido após os estudos realizados sobre trabalhos relacionados ao tema desta Tese é que, de um modo geral, as soluções de NoC apresentadas são baseadas em duas grandes categorias de NoCs em tempo real: reserva de recursos e arbitragem em tempo de execução. Ambas categorias oferecem garantias de previsão da latência de pior caso para os fluxos que trafegam naquelas redes. No entanto, o conhecimento prévio do WCL oferecido por aquelas redes tem uma relação custo benefício a ser considerada.

Nas redes analisadas, todo tráfego precisa ser considerado previamente para que se possa garantir as restrições temporais de fluxos de tempo real. Portanto, uma análise estática precisa ser feita antes da alocação dos núcleos e do dimensionamento dos recursos da rede, como por exemplo a definição de número de canais TDM a serem empregados numa rede baseada em tal tecnologia. Neste ponto, a relação custo-benefício entre as latências dos fluxos na rede e o custo de silício necessário para implementar o mecanismo que irá garantir tais latência deve orientar a definição de quantos fluxos irão receber mecanismos com prioridade de roteamento em todo o sistema e, consequentemente,

oferecer apenas mecanismos de melhor esforço para os fluxos restantes. Entretanto, sistemas embarcados normalmente têm mais aplicações de tempo real *soft* do que *hard*, e a percepção de qualidade, ocasionada pela perda de *deadlines* relativos às aplicações de tempo real *soft*, pode ser relevante para alguns daqueles sistemas.

A rede RTSNoC apresentada neste documento está focada neste contexto: sistemas embarcados nos quais existem mais aplicações de tempo real *soft* do que aplicações de tempo real *hard*; além disso, os sistemas embarcados alvo são aqueles que demandam por algum grau de qualidade de serviços relacionada aos fluxos de tempo real *soft*. Para atender esta demanda, a rede RTSNoC utiliza um chaveamento com intercalação de *flits* de diferentes pacotes em um mesmo canal de comunicação. Ou seja, ao invés de armazenar um pacote inteiro para depois encaminhá-lo na rede, como é usual nas redes de interconexão, como *store-and-forward*[1], o armazenamento no roteador da RTSNoC é de apenas um *flit*. Para isso, foram adicionadas as informações de roteamento em todos os *flits* que pertencem a um pacote. Com isso, cada *flit* recebido por um roteador pode, ou não, pertencer à um mesmo fluxo. Para tratar estes *flits* de modo que nenhum fluxo fique bloqueado ou tenha uma latência muito grande, foi concebido um algoritmo de arbitragem que oferece prioridade para *flits* procedentes de roteadores mais distantes, similar ao algoritmo *round-robin* com peso.

O roteamento por intercalação de *flits* requer um árbitro que ofereça um certo grau de flexibilidade na concessão de recursos durante o processo de arbitragem, de modo que algumas requisições podem receber um número maior de *grants*, porém limitado. O árbitro *round-robin* com peso apresenta este comportamento. Cada requisição i recebe um peso w_i que indica a máxima fração f_i dos *grants* que a requisição i vai receber de acordo com a razão $f_i = w_i/W$, em que W é dado pela seguinte expressão:

$$W = \sum_{j=0}^{n-1} w_j \tag{4.1}$$

Portanto, uma requisição com maior peso vai receber a maior fração dos *grants*, enquanto que a requisição com menor peso irá receber

[1] O chaveamento por pacotes consiste em dividir as mensagens em pacotes de comprimento fixo, cada um incluindo um cabeçalho com as informações necessárias para o seu roteamento pela rede. Os pacotes são enviados um a um e cada pacote reserva apenas os recursos necessários para avançar de nodo em nodo. Em cada canal de entrada do roteador, deve ser incluído um *buffer* com capacidade mínima para armazenar um pacote inteiro.

a menor fração. Por exemplo, suponha que um árbitro *round-robin* com peso tenha de quatro entradas de requisição com os seguintes pesos: 1, 3, 5 e 7, respectivamente. Neste caso, as requisições de entrada irão receber 1/16, 3/16, 5/16 e 7/16 dos *grants*, respectivamente.

Conforme mostrado na Figura 26, um árbitro *round-robin* com peso pode ser implementado com um árbitro *round-robin* precedido de um circuito que desabilita uma requisição de uma entrada que já tenha usado a sua cota de *grants*. Aquela Figura ilustra uma célula lógica para uma requisição de entrada e uma saída de *grant*, a qual consiste de um registrador contendo o peso da célula lógica (*Weight*), um contador ($Cont = 0$) uma lógia AND (&) e um árbitro *round-robin* convencional (*Arbiter*).

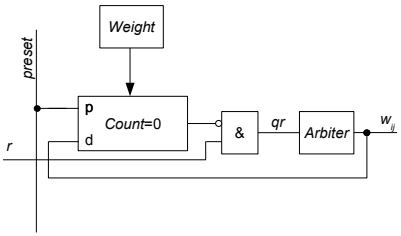


Figura 26 – Representação de uma célula lógica do árbitro *round-robin* com peso. Fonte: adaptado de (DALLY WILLIAM. J.; TOWLES, 2001).

Na Figura 26, quando a linha de *present* é ativada, o contador é carregado com o peso armazenado no registrador de peso (*Weight*). Enquanto o contador não for igual a zero, a lógica AND é habilitada e a requisição chega ao árbitro. Cada vez que um *grant* é concedido para aquela requisição, o contador é decrementado de uma unidade. Quando o contador chegar a zero, a lógica AND desabilita a chegada de novas requisições ao árbitro, até que um novo sinal de *preset* seja ativado.

O uso desta técnica de incluir dados de roteamento para todo *flit* de um pacote proporciona um aumento no custo de silício da rede. Porém, tal custo é baseado em elementos lógicos e conexões metálicas, e segundo os pesquisadores Mark (MARK; FAN, 2004) e Kuon (KUON; ROSE, 2007), eles representam um custo menor do que o uso de blocos de memória que seriam necessários para armazenar pacotes, ou parte de pacotes, nos roteadores. Baseado nesta mesma linha de raciocínio, outra técnica empregada foi utilizar blocos de memória apenas nas in-

terfaces de rede que conectam os núcleos à rede, de modo que nenhum bloco de memória foi empregado nos roteadores.

4.2 DEFINIÇÕES DE ESCOPO DO TRABALHO

As subseções que seguem apresentam as definições de escopo deste trabalho, nas quais são estabelecidos o segmento alvo para a NoC proposta, a tecnologia adotada para a implementação da rede, métricas para consumo de recursos de silício e as métricas adotadas para avaliação de latência da rede proposta. Os resultados obtidos neste trabalho estão baseados nos critérios e ferramentas estabelecidos nestas subseções.

4.2.1 Segmento de SoCs alvo para a rede RTSNoC

A rede RTSNoC apresentada neste documento está focada em sistemas de tempo real, baseados no conceito de SoC, nos quais existem mais aplicações de tempo real *soft* em execução do que aplicações de tempo real *hard*; além disso, os sistemas alvo demandam por algum grau de qualidade de serviços relacionada aos fluxos de tempo real *soft*. Um exemplo de aplicação industrial que está neste contexto é o sistema conhecido como PABX (acrônimo de *Private Automatic Branch eXchange*). São sistemas de tempo real com baixa quantidade de fluxos relacionados a aplicações de tempo real *hard* e muitos fluxos relacionados a aplicações de tempo real *soft*. Estas aplicações de tempo real *soft* estão relacionadas a comunicação de voz que podem ter taxas de bits variáveis quando envolvem comunicação sobre protocolo de Internet (em inglês *Internet Protocol* - IP) e comutação e imagens para video iterativos relacionados ao serviço de videoconferência. Normalmente um sistema PABX aceita algum nível de degradação no desempenho, sem que isso danifique o sistema ou cause o seu travamento.

A degradação de desempenho que pode ocorrer em um sistema PABX acaba por influenciar na qualidade de áudio ou no tempo de sinalização para o usuário, como atraso no envio de tons de sinalização. Apesar destes sistemas aceitarem algum nível de degradação de desempenho, a percepção de qualidade pode ser importante para alguns usuários deste tipo de sistema.

4.2.2 Tecnologias adotadas nos experimentos

SoCs abrangem um vasto espectro de objetivos, além de diferentes tecnologias de implementação. Neste trabalho foi considerada a classificação sobre projeto de circuitos integrados baseados em NoC proposta por Cidon (CIDON, 2010). Naquela classificação, uma NoC pode ser implementada em quatro diferentes tecnologias:

- ASIC (*Application-Specific Integrated Circuit*): tecnologia utilizada na implementação de SoCs dedicados à tarefas específicas;
- ASSP (*Application-Specific Standard Product*): algumas vezes denominado de ASIP (*Application-Specific Instruction set Processors*), são SoCs que possuem múltiplos processadores com arquiteturas específicas dedicados para uma família de aplicações;
- CMP (*Chip multiprocessors*): São SoCs que implementam múltiplos processadores de propósito geral, para realizar tarefas a serem definidas em tempo de execução pelas aplicações, como processamento paralelo; e
- FPGA (*Field Programmable Gate Array*): dispositivos lógicos programáveis com hardware flexível que permite uma vasta gama de implementações de SoCs.

Nos estudos realizados sobre trabalhos relacionados ao tema desta Tese percebeu-se que os autores geralmente publicam os resultados de suas investigações científicas utilizando as tecnologias FPGA e *standard-cell based*[2] ASIC.

Por este motivo, estas duas tecnologias foram utilizadas neste trabalho para avaliar o uso de recursos de silício da NoC proposta. Além disso, foi feita a integração da rede proposta neste trabalho aos núcleos funcionais de um SoC de sistema embarcado tipo PABX, cujos núcleos funcionais já foram implementados e validados pelo seu fabricante na tecnologia FPGA.

A avaliação da rede RTSNoC em ASIC foi feita com o uso da ferramenta de síntese *Synopsys® Ultra Design Compiler*. Neste caso, apenas a área de silício foi analisada, sendo que simulações e implementações não foram consideradas para esta tecnologia. No caso da

[2]São células lógicas pré-projetadas que oferecem funções lógicas, como AND, XOR, multiplexadores ou Flip Flop, para uso em ferramentas de síntese para ASICs. (SMITH, 1997)

tecnologia FPGA foi possível avaliar o consumo de recursos de silício, realizar simulações e implementações físicas; esta última, devido a sua facilidade de implementação e baixo custo, quando comparada com o processo industrial necessário para a implementação na tecnologia ASIC.

4.2.3 Métrica de consumo de recursos de silício

Pode ser observado na literatura que trata de redes intra-chip que o consumo de área silício é frequentemente utilizado como métrica de avaliação de NoCs. Quando as NoCs são implementadas na tecnologia FPGA, os autores costumam relacionar a quantidade de elementos lógicos gastos na implementação da rede.

Os dispositivos FPGAs são circuitos programáveis compostos por um conjunto de células lógicas ou blocos lógicos alocados em forma de uma matriz. Em algumas arquiteturas, os blocos lógicos possuem recursos sequenciais tais como Flip-flops e/ou registradores. Cada fabricante nomeia seu bloco lógico, podendo haver mais de um nome para um mesmo fabricante. A estrutura básica de um FPGA pode variar de fabricante para fabricante, de família para família ou até em uma mesma família de um mesmo fabricante podem existir variações; ainda assim, alguns elementos fundamentais são mantidos.

Neste trabalho, as implementações físicas foram feitas na ferramenta de síntese ISE® pertencente ao fabricante Xilinx®, tendo como dispositivo FPGA alvo o modelo XC6VLX75T-3-FF484, da família Virtex 6®. Nessa família, os blocos de CLB são compostos por dois SLICES, denominados de SLICEM e SLICEL. Cada SLICE é composto por um mesmo conjunto de circuitos eletrônicos básicos, como LUTs (*Lookup Table*) e Flip-flops, sendo que o que diferencia estes SLICEs é a existência de um bloco de memória RAM apenas no SLICEM. A Figura 27 apresenta a ilustração de um dos SLICEs disponíveis no FPGA (SLICEM). Em todas as implementações realizadas foram coletados os valores relativos ao consumo de SLICEs.

A ferramenta ISE gera um relatório de síntese, apresentando o consumo dos elementos lógicos do projeto sintetizado. Este consumo é dado em *Slice LUTs* e *Slice Registers*, além de outros componentes como pinos de entrada e saída, etc. Neste trabalho são adotadas como métricas de consumo de lógica os valores de *Slice LUTs* e *Slice Registers* reportados pela ferramenta de síntese ISE, além da soma daqueles valores *Slice LUTs* e *Slice Registers*.

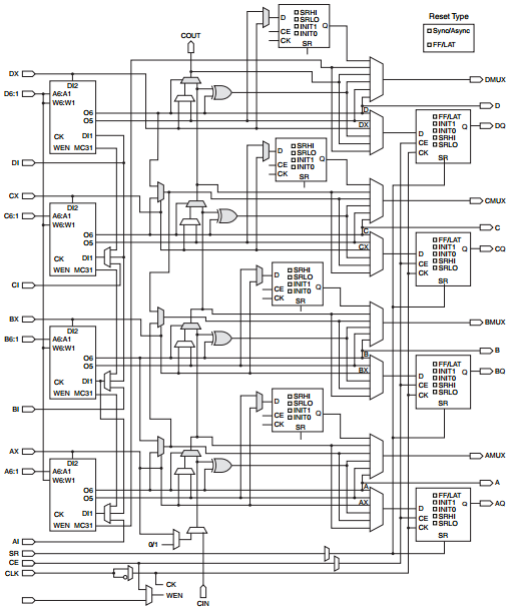


Figura 27 – Diagrama interno de um SLICEM, da família de FPGAs Virtex 6®.Fonte: (UG364, 2010).

O consumo de recursos de silício para a tecnologia ASIC é expresso neste trabalho em termos de área de silício utilizada para a implementação de um roteador da rede RTSNoC. Foi utilizada a ferramenta de síntese *Synopsys*® *Ultra Design Compiler* utilizando a biblioteca SAED 90nm *Low Power*. Além da área consumida para cada componente do roteador RTSNoC, também foi calculado, de modo aproximado, o número de portas lógicas usados para implementar o roteador RTSNoC na tecnologia de 90nm. O número de portas lógicas foi obtido através da relação entre a área do componente e a área da menor porta lógica da tecnologia alvo. Para a tecnologia SAED 90nm, tal porta lógica é uma porta NAND com 5.5296μm^2. Esta referência é útil para se ter uma estimativa da área de silício consumida em outras tecnologias, como por exemplo 65nm ou 120nm.

4.2.4 Métrica de desempenho

O conceito de interconexão intra-chip é conhecido como *Network-on-Chip* (NoC), terminologia proposta por Hemani *et al.* (HEMANI et al., 2002). De maneira simplificada pode-se dizer que o projeto de uma NoC consiste em adaptar modelos, técnicas e ferramentas de projetos de Redes de Computadores para uso em projetos de SoC. Assim, uma *Network-on-Chip* é uma rede de interconexão usada para interligar diversos núcleos de processamento dentro de um mesmo chip.

A definição de quais as métricas adequadas para avaliar o desempenho de redes de interconexão varia entre os autores. Por exemplo, Peterson e Davie (PETERSON LARRY L.; DAVIE, 2007) sugerem a largura de banda (*bandwitdh*), a vazão (*throughput*) e a latência (*latency*) como adequadas para tal; enquanto outros como Dally e Towles (DALLY WILLIAM. J.; TOWLES, 2001) sugerem o custo de implementação da rede, além da vazão e da latência. Neste trabalho foram consideradas como métricas de desempenho para redes de interconexão a vazão e a latência, formalmente definidas a seguir.

A vazão é a taxa em que pacotes são entregues por uma rede utilizando como referência um tráfego padrão. Normalmente é medido contando-se o número de pacotes que chegam ao seu destino sobre um determinado intervalo de tempo para cada fluxo (para um par origem - destino específicos). A vazão é algumas vezes chamada de tráfego aceito e comparada com o tráfego injetado na rede como forma de avaliação de desempenho desta (DALLY WILLIAM. J.; TOWLES, 2001). A Figura 28 mostra um gráfico de uma rede hipotética usada para exemplificar este tipo de análise. Naquela Figura, a vazão é comparada com o tráfego injetado na rede. No exemplo, até o nível de saturação, a vazão é igual à demanda e a curva é uma linha reta ascendente. Se for aumentado o tráfego injetado, a rede pode atingir seu ponto de saturação, que é o maior nível de demanda no qual a vazão é igual à demanda. Se a demanda for aumentada a partir deste ponto, a rede não estará apta a entregar os pacotes com a mesma taxa em que são gerados, o que pode ser observado com a linha reta a partir do tráfego injetado em 50% de sua capacidade de geração.

A latência é o tempo necessário para que um pacote atravesse a rede de uma origem até um destino. Assim como é feito na análise por vazão, a latência é comparada com o tráfego injetado na rede como forma de avaliação de desempenho desta (DALLY WILLIAM. J.; TOWLES, 2001). De forma similar ao exemplo da análise de vazão, a Figura 29 mostra um exemplo de gráfico para uma rede hipotética, no qual a

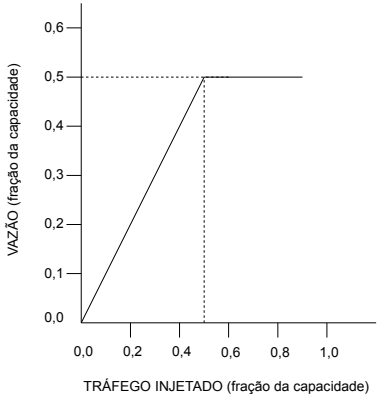


Figura 28 – Gráfico *Vazão* × *Tráfego injetado* para uma rede qualquer tomada como exemplo. Fonte: adaptado de (DALLY WILLIAM. J.; TOWLES, 2001).

latência é comparada com o tráfego injetado na rede. Com o aumento de tráfego, aumentam as disputas por recursos da rede, fazendo com que os pacotes tenham de esperar mais tempo para serem entregues. No exemplo, até o ponto de 80% do tráfego injetado este aumento é pouco significativo. Qualquer aumento adicional ao tráfego injetado passará a exigir um tempo substancialmente maior para que os pacotes sejam entregues aos seus destinos.

O objetivo deste trabalho foi o desenvolvimento de uma rede intra-chip para ser utilizada em sistemas embarcados aplicados à sistemas de tempo real. Nestes sistemas, o tempo máximo para a execução de uma tarefa é um parâmetro fundamental. Por isso, neste trabalho foi adotada a métrica de latência, para mensurar o tempo gasto em uma comunicação entre dois pontos quaisquer da rede.

A latência será medida de duas formas. A primeira delas é utilizando ciclos de *clock* da rede e utilizada nas avaliações feitas através do uso de *testbenches* na ferramenta de simulação ISim®, do fabricante de FPGA Xilinx®. Neste caso, a latência será medida considerando o número de ciclos de *clock* necessários para que um *flit* atravesse a rede, desde o canal de entrada no roteador de origem do pacote até o canal de saída do roteador de destino. Para simplificar a análise, mantendo o foco apenas na latência oferecida pela NoC, foi assumido neste trabalho que o tempo de processamento dos nós da rede é nulo

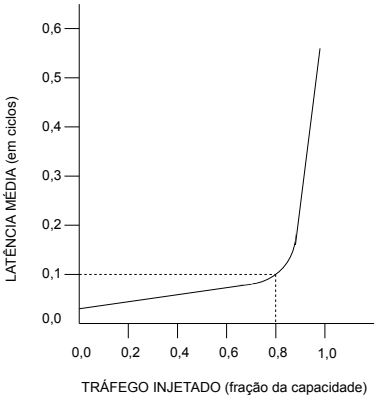


Figura 29 – Gráfico *Latência* × *Tráfego injetado* para uma rede qualquer tomada como exemplo. Fonte: adaptado de (DALLY WILLIAM. J.; TOWLES, 2001).

e, consequentemente, não há contabilização de ciclos de *clock* para tal processamento.

A segunda forma de medida de latência será utilizando o tempo, na unidade de micro segundos (μs). Esta forma de medida foi feita após a integração da rede RTSNoC com os núcleos funcionais de um sistema PABX, no qual o tempo de comunicação utilizado como referência era nesta ordem de grandeza. Todas as medições de latência serão apresentadas nos Capítulos 5 e 6.

O estudo de caso do PABX é um caso real onde a latência de pior caso é uma métrica de comunicação que, se não atendida, pode vir a impactar nas métricas de desempenho de computação, como a perda de *deadline* de tarefas que estão sendo executadas nos processadores/núcleos conectados na rede. No caso da rede proposta nesta Tese, foi avaliado se o WCL calculado foi ultrapassado em algum momento nos experimentos. Tais resultados são apresentados na Seção 5.3.

4.3 DESENVOLVIMENTO DA REDE

Esta seção descreve a implementação da rede RTSNoC, desenvolvida para comprovar a hipótese apresentada nesta Tese. Para melhor organização, a seção foi dividida em subseções sendo que cada uma delas descreve os componentes da rede, desde a topologia adotada até os elementos que compõem a rede, como roteadores, e detalhando partes daqueles elementos que são importantes para o entendimento da solução como um todo.

4.3.1 Topologia

A rede RTSNoC adota a topologia ortogonal 2-D. Seus roteadores podem ser configurados em tempo de síntese para terem de cinco a oito canais de interconexão, indicados pelas letras iniciais dos pontos cardeais de uma bússola, conforme mostra a Figura 30. A escolha da quantidade de canais por roteador e o número de bits de dados em canal é um parâmetro de projeto que pode ser determinado pelo projetista antes da síntese da rede.

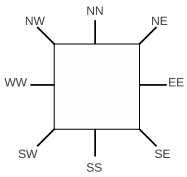


Figura 30 – Roteador RTSNoC e seus canais de comunicação.

Geralmente roteadores para uso em NoC não possuem um número grande de canais de comunicação, pois a complexidade de algumas partes do roteador podem crescer exponencialmente com um número maior de núcleos conectado à ele. Apesar disso, o roteador suporta até oito conexões para aproveitar o senso de localidade no sistema, permitindo aos projetistas alocarem os núcleos que comunicam entre si mais intensamente. Esta abordagem reduz o número de saltos entre roteadores na rede (ou em inglês *hops*), o que implica na redução da latência de comunicação na rede. Neste caso, é muito melhor alocar aqueles núcleos que trocam muitos pacotes na comunicação em *clusters*.

Dependendo da topologia da rede e da posição dos roteadores

na rede, até quatro canais de comunicação podem ser usados para interconectar um roteador aos outros roteadores da NoC; enquanto que os canais remanescentes podem ser utilizados para conectar núcleos à rede. A Figura 31 ilustra um sistema genérico que utiliza a RTSNoC. Quatro roteadores foram usados para estabelecer uma rede em malha regular 2×2 que pode interconectar até vinte e quatro núcleos. Naquela Figura, cada roteador possui seis núcleos conectados a ele e podem ser vistos como *clusters*, indicado por uma linha pontilhada ao redor do roteador número três.

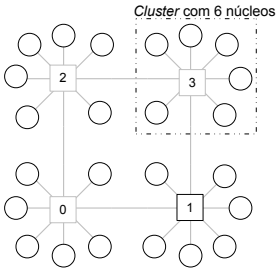
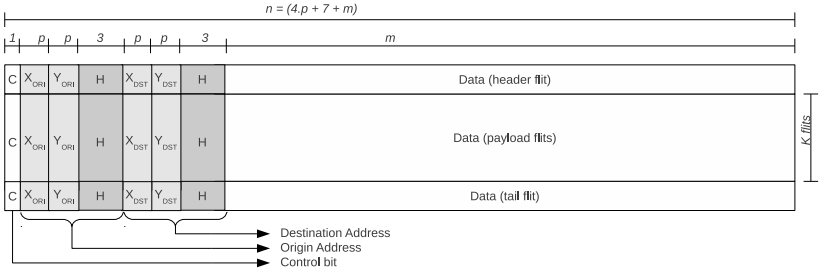


Figura 31 – Exemplo de uma rede em malha regular com 4 roteadores e 24 núcleos.

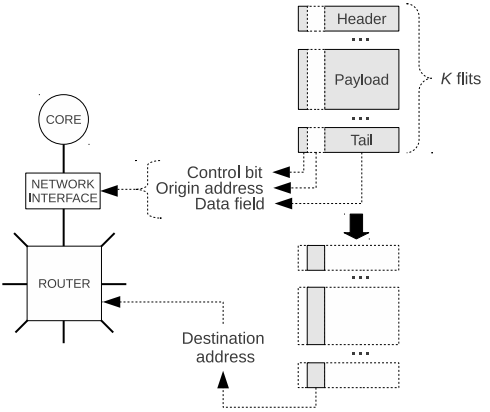
4.3.2 Formato do pacote

Para determinar a latência de pior caso para qualquer fluxo presente na rede, a RTSNoC utiliza uma certa quantidade de dados de controle que precisam ser transmitidos junto com os dados dos pacotes pertencentes àquele fluxo. Assim, cada *flit* do pacote precisa conter os endereços de origem e destino do *flit*, além do *payload*. As redes baseadas na técnica de alocação de recursos da rede possuem *flits* de cabeçalho que contém tais informações de origem e destino do pacote. No entanto, o *flit* de cabeçalho daquelas redes efetua a alocação dos recursos necessários para a transmissão dos outros *flits* do pacote e só liberam os recursos alocados após o envio do *flit* terminador. No caso da rede RTSNoC, os *flits* são roteados individualmente. Esta abordagem foi utilizada porque elimina a necessidade de alocação de recursos na rede, pois todos os *flits* carregam consigo as informações de roteamento. Além disso, outras NoCs precisam enviar um *flit* terminador

informando todos os roteadores envolvidos na transmissão do pacote que as conexões reservadas podem ser liberadas para outros fluxos, o que não é necessário na RTSNoC. A Figura 32(a) ilustra o formato do *flit* na RTSNoC e a Figura 32(b) mostra os elementos da rede que manipulam cada um dos campos daquele *flit*.



(a) Formato do *flit* na RTSNoC. O tamanho dos campos X e Y dependem do número de nodos da rede.



(b) As setas na figura indicam que alguns campos do *flit* são tratados pelos roteadores da rede; enquanto outros são tratados pela Interface de Rede.

Figura 32 – Formato do flit na RTSNoC e componentes que tratam os campos com informações de roteamento.

Os campos Controle de Bit (C) e *Data* são usados por uma interface chamada de Interface de Rede *Network interface*, enquanto que os campos são usados pelos roteadores da rede, como o endereço de destino

do *flit*. Tais elementos serão detalhados nas próximas Subseções.

Um pacote na RTSNoC é composto por qualquer número de *flits*, $f + 2$, em que f é o número de *flits* da carga útil do pacote (ou *payload*) e o valor constante 2 significa que o pacote possui dois *flits* extras usados como *flits* de controle do pacote. A RTSNoC aceita três diferentes tipos de *flits*: cabeçalho (ou *header*), carga útil (ou *payload*) e terminador (ou *tail*). Os *flits* de cabeçalho e terminador são chamados de *flits* de controle. Para identificá-los em um pacote, todo *flit* possui um campo chamado de Bit de Controle (indicado por C na Figura 32(a)) usado para informar se o *flit* é de controle ($C = 1$ quando o *flit* for de cabeçalho ou terminador) ou *flit* de carga útil ($C = 0$).

O cabeçalho pode conter informações para serem usadas pela aplicação executada no núcleo de destino onde o *flit* será entregue. Esta informação, colocada no campo *Data* do *flit* de cabeçalho na Figura 32(a), pode ser interpretada como uma informação relativa a um protocolo de alto nível, visto apenas pelas aplicações nos núcleos da rede. O *flit* terminador é o último de um pacote, e pode conter a última informação válida da carga útil ou qualquer outra informação utilizada pelas aplicações nos núcleos da rede, similar ao *flit* de cabeçalho.

Os *flits* possuem seis campos adicionais chamados de X_{ORI}, Y_{ORI}, H_{ORI}, X_{DST}, Y_{DST} e H_{DST}. Os campos X_{ORI} e Y_{ORI} são usados para informar as coordenadas X e Y da rede onde o *flit* foi gerado (i.e. o endereço de origem do pacote que está sendo transmitido *flit* a *flit*) e o campo H_{ORI} informa o endereço do canal de entrada no roteador de origem onde o *flit* foi gerado. Os campos X_{DST} e Y_{DST} são usados para informar as coordenadas X e Y do roteador de destino na rede onde o *flit* deve ser entregue; enquanto que o campo H_{DST} refere-se ao endereço do canal de saída no roteador de destino onde o *flit* será efetivamente entregue.

4.3.3 Canais físicos

Cada ponto de interconexão do roteador provê dois canais unidirecionais e alguns sinais de controle usados para controlar a transferência de dados entre dois roteadores adjacentes, ou entre um roteador e um núcleo. A Figura 33 ilustra um canal de comunicação e seus sinais. O tamanho de cada canal pode ser configurado em tempo de projeto e baseado nas necessidades do sistema embarcado. Os sinais chamados de DIN e $DOUT$ referem-se aos barramentos de entrada e saída de dados, respectivamente, e que carregam os dados através da

rede.

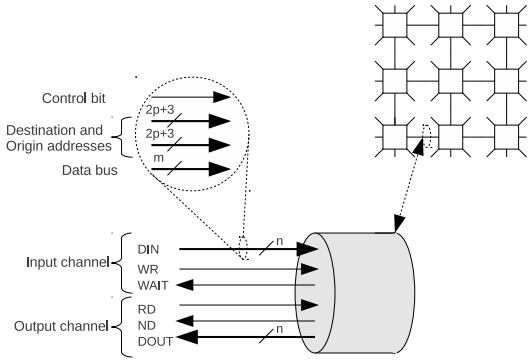


Figura 33 – Sinais do canal de comunicação.

Como ilustrado na Figura 33, os sinais DIN e $DOUT$ possuem n bits cada. Deste total de n bits, m são usados como sendo o campo $Data$, apresentado na Figura 32(a), e $2p + 3$ bits são usados para compôr os endereços de origem e destino do *flit*, sendo que $2p$ daqueles bits representam as coordenadas X e Y e os 3 bits restantes são usados para indicar a porta local H no roteador de destino onde o *flit* deverá ser entregue.

Os sinais RD e WR são sinais de controle usados para ler dados vindos pelo canal de saída e escrever dados no canal de entrada, respectivamente. Os sinais $WAIT$ e ND são usados para controlar o fluxo dos *flits* que trafegam pelo canal; o sinal $WAIT$ informa ao núcleo ou roteador conectado ao canal que é necessário aguardar para poder escrever um novo *flit* no canal de entrada, e o sinal ND informa ao núcleo ou processador conectado ao canal que um novo *flit* está disponível para ser lido no canal de saída.

4.3.4 Interface de Rede

A "Interface de Rede" é um bloco de hardware capaz de adaptar a comunicação entre o núcleo e a rede. Ela é internamente dividida em dois blocos lógicos, chamados de "Adaptador de núcleo" e "Adaptador de roteador", indicados respectivamente como *Core Adapter* e *Router Adapter* na Figura 34.

O adaptador de rede é um bloco lógico que interage com a rede,

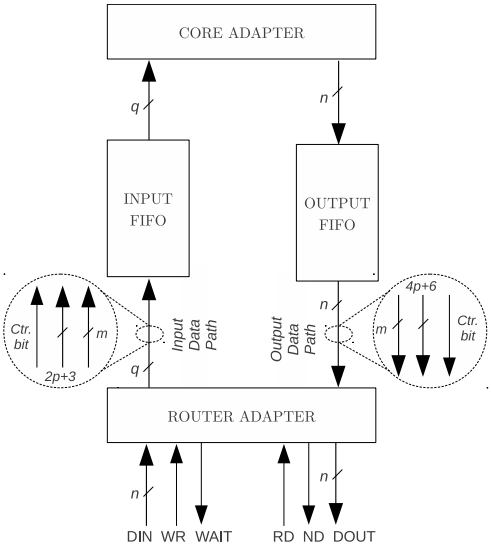


Figura 34 – Estrutura interna da interface e rede.

lidando com os sinais do canal de comunicação físico, apresentado na Subseção 4.3.3. Ele prepara os dados vindos da rede para serem entregues ao núcleo, removendo do *flit* os campos de endereço de destino, pois o endereço de destino é uma informação utilizada apenas pelos roteadores da rede.

O adaptador de núcleo também é um bloco lógico que prepara os dados vindos do núcleo para serem escritos na rede, concatenando os campos de bit de controle (C), endereço de origem e endereço de destino ao dado para gerar o *flit*.

Existem duas memórias FIFO na interface de rede. Uma delas é usada para armazenar os dados recebidos da rede, e é chamada de FIFO de entrada, *Input FIFO* naquela Figura, enquanto que a outra é usada para armazenar os dados recebidos do núcleo, sendo chamada de FIFO de saída*Output FIFO*. A vazão da rede RTSNoC depende do tamanho daquelas memórias FIFO, e este tamanho pode ser definido pelo projetista da rede.

O roteador necessita pelo menos um *flit* armazenado e pronto para ser transferido da FIFO de saída para a rede ou da rede para a FIFO de entrada, para que possa ter uma vazão eficiente. Por aspectos construtivos, a latência interna do roteador para transferir um *flit* de

um canal de entrada para um canal de saída, considerando que não há competição pelo canal de saída, é de dois ciclos de *clock*. A latência entre dois pontos quaisquer da rede não é constante porque a RTSNoC não reserva recursos na rede para a transmissão de pacotes. Isto significa que as memórias FIFO precisam ser projetadas de acordo com dois fatores: (*i*) o tempo que um núcleo gasta escrevendo/lendo um *flit* na interface de rede; e (*ii*) a menor latência esperada na comunicação através da rede. A expressão a seguir descreve a relação entre estes dois fatores:

$$B_{size} = \left\lceil \frac{T_{wr/rd}}{T_{net}} \right\rceil \tag{4.2}$$

sendo que B_{size} é o tamanho do *buffer*, $T_{wr/rd}$ é o tempo que o núcleo gasta para escrever/ler um *flit* na interface de rede, e T_{net} é a menor latência esperada na comunicação entre núcleos através da rede. Quando o valor de B_{size} é unitário, dois registradores são sintetizados, ao invés de blocos de memória FIFO.

4.3.5 Roteador

Um roteador da rede RTSNoC pode ser dividido em sete blocos lógicos: interface de entrada, interface de saída, controlador de fluxo, controlador de roteamento, árbitro, alocador e matriz *crossbar*. A Figura 35 ilustra estes blocos internos que compõem o roteador RTSNoC. Por uma questão de organização, esta seção que descreve o roteador foi dividida em sete subseções, em que cada subseção apresenta um dos sete blocos lógico que compõem o roteador.

4.3.5.1 Interface de Entrada

A interface de entrada, descrita como *Input Interface* na Figura 35, é um bloco lógico que recebe *flits* vindos da interface de rede, ou de um canal de saída pertencente a outro roteador. Ela possui um registrador capaz de armazenar um *flit*. Após receber um *flit*, a interface de entrada informa ao controlador de fluxo que há um novo *flit* aguardando para ser roteado.

O diagrama em bloco da Figura 36 mostra os sinais de entrada e saída da interface de entrada. O sinal i_DIN é o barramento de n bits que vem do canal de comunicação de entrada do roteador, sendo que o

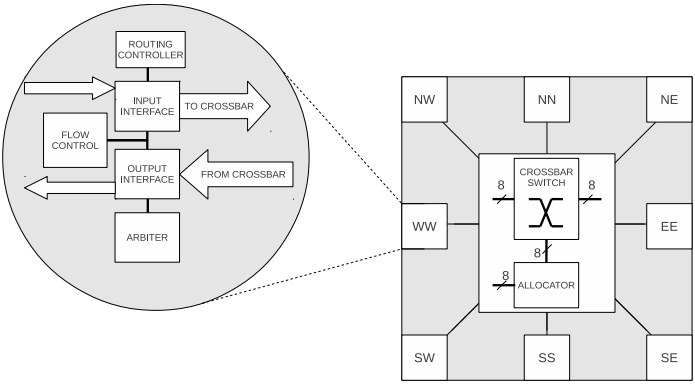


Figura 35 – Diagrama em blocos mostrando a estrutura interna do roteador RTSNoC.

flit que chega nesta entrada é armazenado na saída o_DOUT, também de n bits quando o sinal i_WR é ativado em nível lógico um. Após o registro do *flit* de entrada, o endereço de destino do *flit* é disponibilizado no sinal de saída o_DST para ser usado pelo bloco controlador de fluxo.

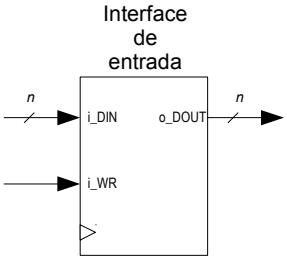


Figura 36 – Sinais de entrada e saída do bloco *Input Interface*.

4.3.5.2 Controlador de Fluxo

O diagrama em bloco da Figura 37 mostra os sinais de entrada e saída do controlador de fluxo. No mesmo momento em que um *flit* é escrito na interface de entrada, o endereço de destino é retirado do

flit e armazenado no bloco controlador de fluxo. Neste momento, o controlador coloca o sinal o_WAIT em nível lógico um, bloqueando a entrada de outro *flit* na interface de entrada.

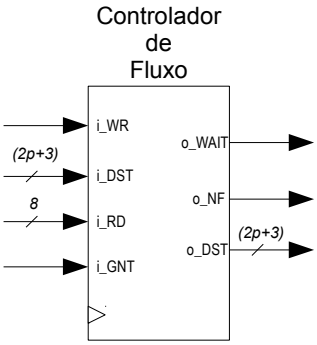


Figura 37 – Sinais de entrada e saída do bloco *Flow Controller*.

O sinal o_NF é ativado em nível lógico um informando ao controlador de roteamento que um novo canal de saída está sendo requisitado, sendo que o endereço de destino é colocado no sinal o_DST, conectado diretamente no controlador de roteamento. O sinal i_GNT é uma confirmação de que o *flit* foi encaminhado para a saída solicitada, gerado pelo bloco de arbitragem. Ao recebê-lo, o sinal o_NF é colocado em nível lógico zero. O sinal o_WAIT só é liberado quando a saída do canal de destino for lida; isto é indicado pelo sinal i_RD, que é um sinal de leitura vindo diretamente do canal de comunicação para onde o *flit* foi encaminhado.

4.3.5.3 Controlador de Roteamento

O bloco controlador de roteamento executa o roteamento estático XY, mostrado no Algoritmo 1. Ele foi adotado na RTSNoC devido a sua simplicidade de implementação e por evitar *deadlock* na rede (NAVABI et al., 2005). Neste roteamento, o algoritmo verifica se o valor das coordenadas X_{DST} e Y_{DST} é igual aos valores pré-fixados para o roteador em tempo de projeto. Se for diferente, o algoritmo gera uma requisição para o canal mais adequado ao roteamento.

A Figura 38 mostra o diagrama em bloco do controlador de roteamento. O controlador recebe a indicação da existência de um novo *flit*

Algorithm 1 XY routing

Require: X_{DST}, Y_{DST}, H_{DST}
Ensure: Request
  if (($X_{DST} = X_{router}$) **and** ($X_{DST} = X_{router}$)) **then**
    Request local channel according H_{DST}
  else
    if ($X_{DST} \neq X_{router}$) **then**
      if ($X_{DST} > X_{router}$) **then**
        Request East channel
      else
        Request West channel
      end if
    else if ($Y_{DST} \neq Y_{router}$) **then**
      if ($Y_{DST} > Y_{router}$) **then**
        Request North channel
      else
        Request South channel
      end if
    end if
  end if]

para roteamento, vinda do controlador de fluxo, através do sinal i_NF. Ele verifica o endereço de destino, disponibilizado no sinal i_DST e executa o algoritmo de roteamento XY para definir qual o canal de saída para o encaminhamento do *flit*.

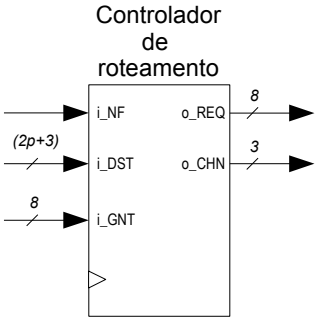


Figura 38 – Sinais de entrada e saída do bloco *Routing Controller*.

Para que seja possível o roteamento XY, foi estabelecido um sistema de coordenadas na RTSNoC no qual cada roteador da rede é identificado por um par de coordenada (x, y), como mostra a Figura 39. Tais coordenadas são usadas nos campos de destino X_{DST} e Y_{DST} no *flit*. O algoritmo utiliza estas coordenadas de destino para que o *flit* seja encaminhado através do roteador para o canal de saída mais apropriado.

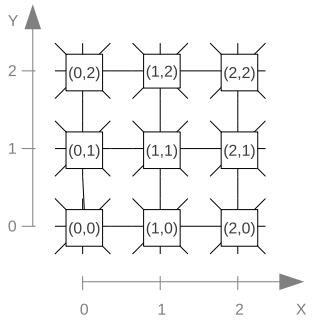


Figura 39 – Sistema de coordenadas em uma rede RTSNoC.

Uma vez definido o canal de saída, é sinalizado em nível lógico um o sinal *o_REQ* relativo àquele canal. Este sinal é conectado na entrada do árbitro do canal de saída. Paralelo a isso, o canal de saída resultante do roteamento é indicado no sinal *o_CHN*, de três bits. Este sinal é disponibilizado no bloco alocador para posterior chaveamento do *crossbar*. Na Figura 38, o sinal de *o_REQ* só é retirado após a chegada do *grant* vindo do árbitro pelo no sinal *i_GNT*.

4.3.5.4 Árbitro

Cada canal de saída do roteador possui o seu próprio bloco de arbitragem (árbitro) para receber as requisições geradas pelos controladores de roteamento presentes nos canais de entrada do roteador RTSNoC. A Figura 40 ilustras os sinais de entrada e saída deste bloco.

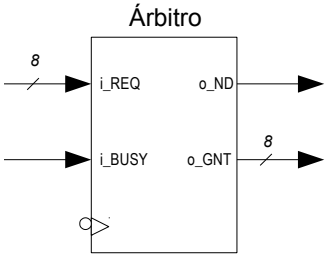


Figura 40 – Sinais de entrada e saída do bloco *Arbiter*.

Cada árbitro implementa um algoritmo de escalonamento, ilustrado através de um fluxograma na Figura 41. Todo canal do roteador recebe um nível de prioridade, diferente dos demais canais, na inicialização do sistema (*reset* do sistema). As maiores prioridades são dadas aos canais de entrada `NN`, `SS`, `EE` e `WW`, pois na rede RTSNoC eles são usados para interconectar com outros roteadores, que foi concebida com a topologia em malha direta 2-D.

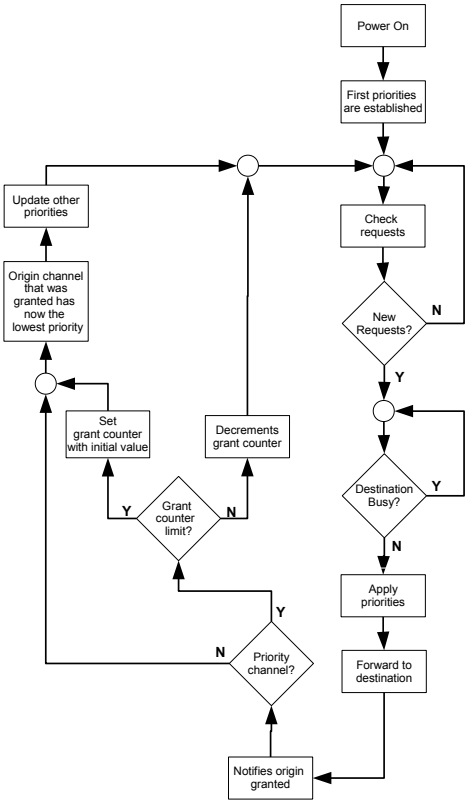


Figura 41 – Fluxograma mostrando o algoritmo de escalonamento implementado no bloco de arbitragem.

No momento em que as requisições chegam, elas já estão com suas prioridades previamente estabelecidas, e o sinal de *grant* é enviado aos blocos de controle de fluxo, controle de roteamento e alocador liberando a comutação para o canal de saída da vez, colocando em nível

lógico um o sinal o_GNT referente àquele canal de entrada. Paralelo a isso, o sinal o_ND é colocado em nível lógico um para indicar ao bloco Interface de Saída a existência de um novo *flit* para ser armazenado junto ao canal de saída. O processo de arbitragem só tem os níveis de prioridade dos canais de entrada alterados após a indicação de que o bloco de saída não está mais ocupado, ou seja, que o dado foi lido por outro roteador ou núcleo, conectado naquele canal de saída do roteador. Esta indicação é feita através do sinal i_BUSY, colocado em nível lógico zero para indicar que o bloco de saída foi lido.

A quantidade de *flits* que um canal com prioridade pode enviar sequencialmente depende do número de requisições simultâneas que podem ocorrer ao mesmo tempo em algum daqueles canais prioritários. Por exemplo, considere uma rede RTSNoC 2×2, conforme mostra a Figura 42. Suponha que o núcleo, representado por um círculo na cor cinza conectado ao roteador um (1) possa receber fluxos vindos de todos os outros vinte e três núcleos conectados na rede possam enviar pacotes para aquele núcleo. Sabendo que o roteamento é XY, então o número de *flits* que cada canal de entrada do roteador um pode encaminhar para aquele núcleo será o indicado na Tabela 3. Note que a quantidade de *flits* possíveis para os canais de entrada NN e WW só serão adotados se houverem requisições simultâneas vindas de todos os núcleos conectados nos roteadores zero, dois e três.

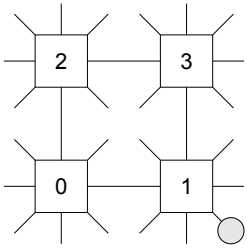


Figura 42 – Exemplo de malha 2×2.

O Algoritmo 2 mostra o pseudo-código do árbitro apresentado no fluxograma da Figura 41 e que foi desenvolvido para esta Tese. De acordo com o algoritmo, todo *flit* tem sua requisição de roteamento atendida se ele tiver a maior prioridade, ou se não houver mais requisições no árbitro requisitando o mesmo canal de saída. Uma vez que uma requisição foi atendida, o canal que requisitou o envio do seu *flit* recebe a menor prioridade para roteamento no próximo ciclo de *clock*,

Tabela 3 – Número de *flit* para cada *grant* recebido pelos canais de entrada.

Canal de entrada	*flits/grant*
NN	12
NE	1
EE	1
SE	1
SS	1
SW	1
WW	6
NW	1

sendo atendido no ciclo seguinte se não houverem outras requisições pendentes. A exceção é o canal ser prioritário por ser canal de interconexão entre roteadores (`NN`, `EE`, `SS` e `WW`). Assim que uma requisição é atendida, O árbitro envia um comando para o bloco lógico chamado de "alocador", informando qual chaveamento entre canal de entrada e saída deve ser realizado.

Algorithm 2 Arbiter

```
Require: Request, Destination
Ensure: Grant, Channel
  if (Reset = active) then
     Channel priorities ← initial values
     Channel counters ← initial values
  else
     if (Request ≠ null) then
        Check priorities
        wait until Destination ≠ busy
        Forward flit
        Notify origin granted
        if (Request = North or East or South or West) then
           if (Channel counter > 0) then
              Decrements channel counter
           else
              Channel counter ← initial value
              Channel granted ← lower priority
              Update priorities for other channels
           end if
        else
           Channel granted ← lower priority
           Update priorities for other channels
        end if
     end if
  end if
```

4.3.5.5 Bloco Alocador

Um "Alocador" n x m é uma unidade que aceita vetores de n m-bit como entrada e gera vetores de n m-bit de saída, conforme mostrado no centro da Figura 35. O Alocador do roteador RTSNoC possui oito vetores de três bits como entrada e oito vetores de três bits como saída. Ele usa as suas interfaces de entrada para receber comandos vindos dos blocos de arbitragem e controlar a matriz *crossbar*, usando suas interfaces de saída para executar as conexões solicitadas dos árbitros.

O bloco Alocador, ilustrado na Figura 43 recebe os *grants* vindos dos árbitros alocados junto aos canais de saída, através dos sinais i_GNT, associando o endereço de três bits disponibilizados nos sinais i_CHN_xx e enviando esta informação para a *crossbar*, através dos sinais o_SWT_xx. O alocador trabalha de modo paralelo, de modo a dar melhor desempenho para o *crossbar*. Isto significa que, em um ciclo de *clock*, ele pode receber até oito comandos vindos dos árbitros de canal e comandar até oito conexões na matriz *crossbar*.

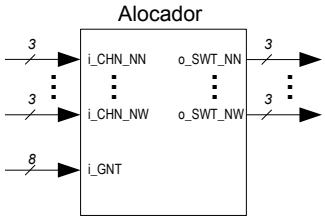


Figura 43 – Sinais de entrada e saída do bloco *Allocator*.

4.3.5.6 Matriz *Crossbar*

Uma matriz *crossbar* n x m é uma estrutura que conecta diretamente n entradas em m saídas sem nenhum estágio intermediário. Assim, a matriz *crossbar* do roteador RTSNoC consiste de m n:1 multiplexadores, um para cada canal de saída do roteador, implementando uma matriz quadrada com oito canais de entrada por oito canais de saída, conforme mostrado na Figura 35. O *crossbar* foi configurado para conectar qualquer canal de entrada com qualquer canal de saída, porém sob a restrição de que cada canal de entrada é conectado em

apenas um canal de saída, assim como um canal de saída pode ser conectado em apenas um canal de entrada. A estrutura de multiplexadores adotada garante que esta restrição é sempre cumprida. Além disso, a matriz *crossbar* não utiliza ciclos de *clock* para executar os chaveamentos entre entradas e saídas

A Figura 44 ilustra os sinais de entrada e saída da *crossbar*. Ela recebe as saídas das interfaces de entrada nos sinais i_NN, i_NE, etc. Estes canais de entrada são conectados aos canais de saída, o_NN, o_NE, etc., de acordo com os comandos presentes nas entradas i_SWT_NN, i_SWT_NE, e assim por diante.

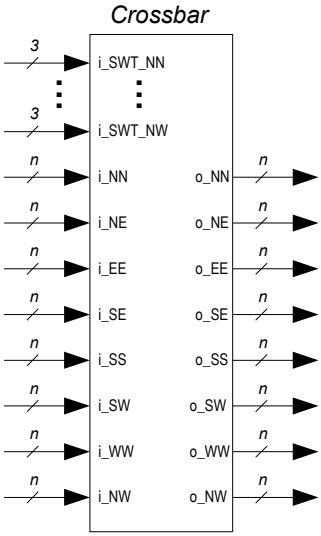


Figura 44 – Sinais de entrada e saída do bloco *Crossbar*.

4.3.5.7 Interface de Saída

O sinal i_DIN recebe o *flit* a ser disponibilizado no canal de saída e é conectado diretamente na *crossbar*, e seus sinais de entrada e saída estão representados na Figura 45. Naquela Figura, o dado é disponibilizado na saída do bloco o_DOUT após o sinal i_DIN receber um nível lógico um vindo do bloco de arbitragem. Ao ser armazenado, o bloco coloca o sinal o_ND em nível lógico um, indicando ao canal de

saída que um novo *flit* está disponível para ser lido. Paralelo a isso, o sinal o_BSY é colocado em nível lógico um para indicar ao árbitro que o bloco de saída está aguardando por uma leitura,. Os sinais i_ND e i_BSY são zerados apenas quando a leitura for realizada pelo roteador ou núcleo conectado ao canal de saída, e indicado pelo sinal i_RD em nível lógico um.

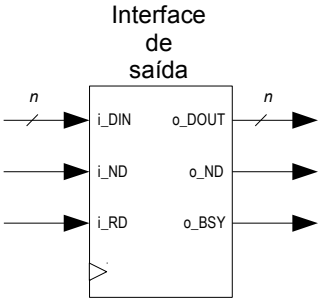


Figura 45 – Sinais de entrada e saída do bloco *Output Interface*.

4.4 LARGURA DE BANDA NA RTSNOC

Os roteadores da rede RTSNoC foram projetados para oferecer uma latência de dois ciclos de *clock* para encaminhar um *flit* de um canal de entrada para um canal de saída qualquer. Para minimizar o número de ciclos de *clock*, foi adotada a técnica de *alternate phase*. A técnica consiste em sincronizar os flip-flops de entrada e saída do caminho combinacional com bordas inversas. Por exemplo, na Figura 46 o flip-flop da entrada é sincronizado pela borda de descida do sinal de *clock*, enquanto o flip-flop de saída é sincronizado pela borda de subida do mesmo sinal de *clock*.

Uma vantagem desta técnica é não necessitar de recursos de silício adicionais, além da redução do número de ciclos de *clock* necessários para encaminhar um sinal (bordas de subida, por exemplo). Como desvantagem, há a necessidade de um mapeamento prévio (no tempo) do projeto como um todo, para evitar que sinais fiquem fora do momento adequado para suas avaliações, uma vez que as ferramentas de síntese para FPGA ou ASIC somente verificam este aspecto (o momento em que sinais devem estar sendo avaliados concomitantemente)

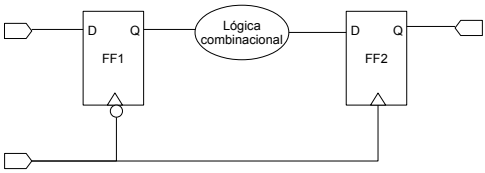


Figura 46 – Representação da técnica de fases alternadas.

se o projeto seguir integralmente a mesma borda de relógio.

Assim, na RTSNoC a vazão para um fluxo qualquer que não sofre competição com outros fluxos pelo mesmo nodo destino é de um *flit* a cada dois ciclos de *clock*. Para o caso genérico em que vários fluxos podem estar enviando pacotes para um mesmo destino, a vazão tende a diminuir para um determinado fluxo. Em outras palavras, a latência de um canal é sempre de um *flit* a cada dois ciclos de *clock*, mas a vazão por fluxo é menor já que a arbitragem adotada na rede faz com que um *flit* de cada fluxo seja encaminhado ao nodo destino. Desta forma, a vazão da rede RTSNoC pode ser descrita para um determinado fluxo como sendo:

$$B_i = \frac{1}{2.k} \tag{4.3}$$

sendo que k é o número de fluxos que estão sendo enviados para um mesmo nodo destino simultaneamente. Pode-se observar pela Equação 4.3 que a vazão para um determinado fluxo é dependente da quantidade de fluxos que estão sendo enviados para o mesmo destino em um determinado instante. Nas redes apresentadas no Capítulo 2, a vazão é constante para todas que oferecem garantias de latência. No caso das redes TDM que também oferecem suporte a fluxos de melhor esforço, como a Æthereal, esta vazão constante só é válida para fluxos com garantia de serviços.

Além disso, conforme citado naquele Capítulo, o uso de DCM ou PLL em conjunto com o uso de caminhos com topologias projetadas para sinais de *clock* também ajudam a minimizar os problemas relacionados com *clock skew* e *jitter*. Nos experimentos realizados nesta Tese, que serão apresentados no Capítulo 5, foram utilizados FPGAs que disponibilizam DCMs e tais caminhos dedicados para sinais de *clock*. Estes recursos foram incorporados às implementações feitas para os experimentos.

A Figura 47 ilustra o modo de sincronização dos componentes internos do roteador da RTSNoC. A sincronização foi feita com base na técnica conhecida como alternância de fase. Para simplificar, a Figura mostra apenas um canal de entrada e um canal de saída.

Na RTSNoC cada *flit* tem o mesmo tamanho que um *phit*. Na Figura 47, cada *phit* possui n bits e $(n-m)$ bits de cada *phit* contém informações sobre o destino de roteamento do *flit*. Estes $(n-m)$ bits são usados pelo controlador de fluxo (*Flow Control* naquela Figura) e pelo controlador de roteamento (*Routing control*). Estes dois componentes, assim como a interface de entrada (*Input Interface*) estão localizados nos canais de entrada do roteador, indicado por *Input channel* naquela Figura. Da mesma forma, o canal de saída (*Output channel*) é representado pela interface de saída (*Output Interface*) e pelo Árbitro (*Arbiter*).

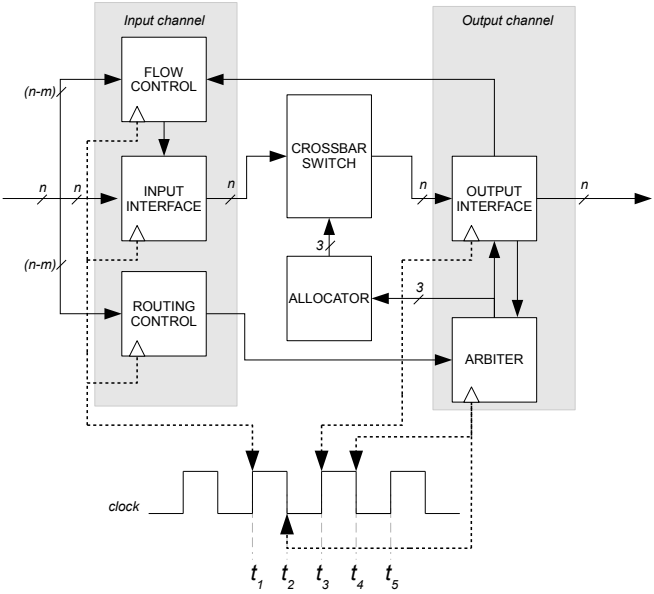


Figura 47 – Sincronização dos componentes internos do roteador RTSNoC com o sinal de *clock*.

Um sinal de *clock* foi representado na base da Figura 47. Note que no instante t_1 todos os três componentes do canal de entrada são ativados pela borda positiva do *clock*. No instante seguinte, t_2, o árbitro

recebe as requisições dos canais de entrada e verifica se o canal de saída (*output interface* na Figura) está disponível para receber um novo *flit*. Se ele estiver livre, o árbitro aplica as prioridades vigentes naquele momento, enviando o sinal de controle para o Alocador da *Crossbar* e informando o canal de saída que um dado está pronto para ser entregue à ele. No instante t_3 o canal de saída armazena o *flit* disponível na saída da *crossbar* e sinaliza o núcleo ou canal de entrada de outro roteador conectado à ele que um novo *flit* está disponível para entrega/encaminhamento. No instante t_4 o árbitro faz a atualização dos níveis de prioridades, conforme apresentado na Subseção 4.3.5.4.

O sinal de leitura RD, descrito na Seção 4.3.3, é enviado do canal de saída para o canal de entrada que teve seu *flit* encaminhado. Este sinal serve como sinalização de que uma nova requisição para transmissão de outro *flit* poderá ser feita por aquele canal de entrada no próximo ciclo de *clock*. Este ciclo pode ser no instante t_5 indicado na Figura 47, se a leitura no canal de saída for feita imediatamente após a sinalização de novo dado disponível (sinal ND apresentado na Seção 4.3.3).

4.4.1 Análise de Vazão para fluxos BE

No final do Capítulo 2 foi apresentada a Tabela 1, a qual resume as principais características das redes intra-chip estudadas para o desenvolvimento desta Tese. A última linha da Tabela mostra a máxima vazão oferecida por cada uma daquelas redes, sendo que aqueles valores são válidos apenas para fluxos prioritários; ou seja, são válidos para fluxos com garantia de latência e não para fluxos de melhor esforço. Porém, algumas das redes estudadas também oferecem mecanismos de roteamento para fluxos BE, como a Nostrum e a Æthereal. No caso da rede Æthereal, alguns canais TDM são alocados especificamente para fluxos BE, os quais são compartilhados por estes fluxos. Para isso, eles são roteados por um mecanismo convencional de roteamento *wormhole*, e a arbitragem é feita por um algoritmo *round-robin*.

Nesta subseção são apresentados os resultados referentes a simulações que foram realizadas no intuito de avaliar a vazão oferecida pelas redes RTSNoC e Æthereal para fluxos BE. As simulações foram feitas tomando como referência um fluxo genérico σ_i em cada uma das redes, para o qual foi definido um tamanho fixo para todos os seus pacotes. O número de fluxos que interferem diretamente em σ_i foi variado, assim como o tamanho dos seus respectivos pacotes.

A Figura 48 apresenta o resultado da simulação de vazão para um fluxo genérico σ_i gerado por um nodo i da rede Æthereal. A simulação foi feita com o uso das expressões de vazão de cada uma das redes. Ou seja, foi uma simulação analítica com tais expressões e realizadas com auxílio de uma planilha eletrônica, tipo Excel®. A vazão da rede Æthereal foi apresentada na Tabela 2 e a vazão da RTSNoC apresentada na Seção 4.4. Tais expressões de vazão foram colocadas em uma planilha, a qual teve os valores de tamanho de pacotes e número de fluxos interferentes variados. Os valores de vazão são obtidos em função do número de fluxos (K) que competem pelos mesmos recursos que o fluxo σ_i e em função do tamanho dos pacotes daqueles fluxos interferentes, em que todos os pacotes dos fluxos interferentes têm o mesmo tamanho, cujo valor é N vezes o tamanho de um pacote de σ_i.

A vazão da rede Æthereal é dada por $(p.n)/(P.s)$, definido no Capítulo 2, em que P é um período de *time slots*, p é o número de *slots* alocados, s é a duração de um *time slot* em ciclos de relógio e n é o número de pacotes em uma transação. Como a rede Æthereal é definida em tempo de projeto, os valores de P, p, s e n foram fixados com os valores 4, 1, 1 e 1, o que significa que a rede tem um período de 4 ciclos de *clock* e 4 *time slots* de 1 ciclo de *clock* cada. Além disso, foi alocado 1 *time slot* para fluxos BE e cada transação transfere um pacote a cada ciclo de *clock*, sendo que cada pacote é formado por três *flits*. Estes valores foram escolhidos para que se pudesse simular um sistema com três fluxos de tempo real crítico, similar ao estudo de caso que será apresentado no Capítulo 6, ficando apenas um *slot* disponível para fluxos BE.

Os valores de vazão foram normalizados em relação ao valor da largura de banda da rede alocada para os fluxos BE de modo que, se apenas o fluxo σ_i estiver trafegando na NoC, então ele terá a máxima vazão na rede. Note que, se o tamanho de todos os pacotes pertencentes aos fluxos interferentes tiverem o mesmo tamanho dos pacotes do fluxo sob análise (σ_i), então a largura de banda é dividida igualmente entre todos os fluxos. Isto pode ser observado na Figura para os valores de $N = 1$, em que a vazão de σ_i é 0.5 para $K = 1$, 0.33 para $K = 2$, e assim por diante.

Quando o tamanho dos pacotes é maior do que o tamanho dos pacotes do fluxo σ_i, o valor da vazão de σ_i diminui na Figura 48. Isto acontece porque o roteamento proposto pelos autores da rede Æthereal para fluxos BE é *wormhole*, e o nodo i precisa precisa esperar a liberação dos recursos (*time slots*) para poder requisitá-los e então encaminhar um pacote na rede.

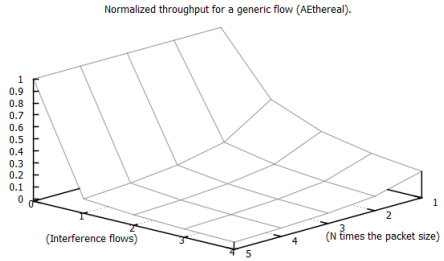


Figura 48 – Simulação de vazão para a rede Æthereal.

Diferente da rede Æthereal que transfere um pacote por ciclo de *clock*, a rede RTSNoC transfere um *flit* a cada dois ciclos de *clock*. Portanto, na simulação o roteador da rede RTSNoC necessita seis ciclos de *clock* para transferir um pacote composto por três *flits*. O resultado da simulação de vazão para a rede RTSNoC é apresentado na Figura 49. Note que a vazão do fluxo σ_i é mantida constante para uma mesma quantidade de fluxos interferentes, independente do tamanho do tamanho dos pacotes que são injetados na rede por tais fluxos. Isto acontece porque o roteador da rede RTSNoC faz o encaminhamento dos pacotes de todos os fluxos *flit* a *flit* sem a contenção de recursos da rede, o que não acontece no roteador da rede Æthereal.

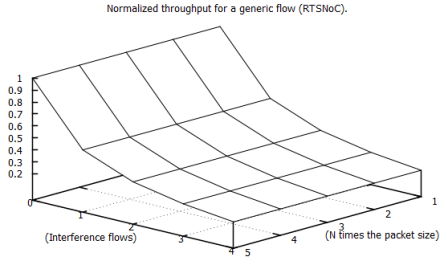


Figura 49 – Simulação de vazão para a rede RTSNoC.

Para que se possa ter uma visualização melhor entre os resultados das simulações feitas para as duas redes, as duas curvas de vazão são apresentados juntas na Figura 50. A rede RTSNoC permite que a vazão de um fluxo BE seja maior com o aumento no número de tráfegos interferentes, quando comparada com uma solução de roteamento *wormhole*

sobre canais TDM. É importante lembrar que o aumento na vazão pode ser melhorado numa rede baseada na tecnologia TDM com a alocação de mais canais, no entanto isso deve ser feito em tempo de projeto para todas as redes até agora apresentadas na literatura.

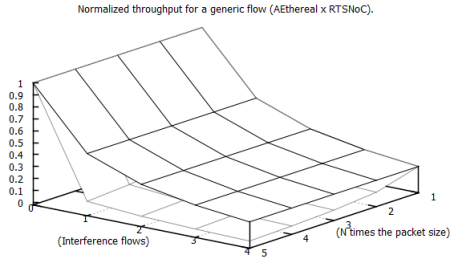


Figura 50 – Resultados das simulações de vazão para as redes RTSNoC e Æthereal.

4.5 LATÊNCIA DE PIOR CASO NA RTSNOC

Esta seção apresenta uma análise sobre a contribuição na latência inferida pela rede RTSNoC. Na análise de latência, foram levados em conta os modelos de latência apresentados no Capítulo 3. Inicialmente, considere que as análises de latência são feitas em uma rede RTSNoC genérica e representada por um conjunto de fluxos $\Gamma = \{\sigma_1, \sigma_2, ...\sigma_n\}$, sendo que cada fluxo é representado individualmente por sua 7-tupla $\sigma_i = (v_{si}, v_{di}, P_i, T_i, D_i, L_i, f_i)$.

A latência do pior caso para um pacote que pertence a um fluxo σ_i na rede RTSNoC é definido da mesma maneira que foi definido na equação 3.4, sendo a soma da latência experimentada por todos *flits* que pertencem ao mesmo pacote, no caminho entre um nodo de origem e nodo de destino com `h` roteadores.

O primeiro *flit* de um pacote pode ter uma latência diferente dos outros *flits* do mesmo pacote. Isso acontece porque, uma vez que o primeiro *flit* chegar ao nodo de destino, então os outros *flits* subsequentes à ele deveriam ser roteados com as prioridades estabelecidas pelo primeiro *flit* no caminho entre o nodo de origem e o de destino. Isto não ocorrerá caso haja alguma alteração em outros fluxos ao longo deste caminho após a passagem do *flit* de cabeçalho do pacote. Para a análise da latência de pior caso na RTSNoC, serão levadas em conta as

seguintes premissas:

- os *flits* da carga útil (ou *payload*) serão encaminhados com as mesmas prioridades estabelecidas pelo *flit* de cabeçalho em todos os roteadores no caminho entre o nodo de origem e o nodo de destino; e

- todos os fluxos que possam competir pelos mesmos recursos que o fluxo sob análise estão enviando seus pacotes para o mesmo destino, antes mesmo do *flit* de cabeçalho do pacote sob análise ter sido injetado na rede.

De acordo com a equação de latência (3.5) apresentada na Subseção 3.2, a primeira latência a ser analisado está relacionada ao *flit* de cabeçalho do pacote. Não há contenção de recursos na rede RTSNoC, devido à técnica de intercalação de *flits* adotada na sua implementação. Se `N` fluxos estão competindo pelo mesmo canal de saída em um roteador, então cada um de seus *flits* é enviado pelo canal solicitado a cada ciclo de arbitragem de acordo com a prioridade dada pelo algoritmo de arbitragem apresentado na Subseção 4.3.5.4. Assim, a latência máxima esperada para o *flit* de cabeçalho, $\mathtt{L_{header}}$, que pertence a um pacote do fluxo σ_i é dada pela seguinte expressão:

$$L_{header(i)} = \sum_{j=1}^{h} 2(N_j) \tag{4.4}$$

Lembrando que a vazão na rede RTSNoC para um determinado fluxo é dada pela expressão $B_i = \dfrac{1}{2.k}$, então a latência da carga útil e do *flit* terminador é dada como segue:

$$L_{payload;tail(i)} = \frac{(f-1)}{\dfrac{1}{2k}} = 2k(f-1) \tag{4.5}$$

sendo que `k` é a quantidade de pacotes que competem pelo mesmo nodo de destino na rede e `(f-1)` é a quantidade de *flits* da carga útil mais o *flit* terminador do pacote analisado. A partir das expressões 4.4 e 4.5 é possível descobrir a latência de pior caso para qualquer pacote na rede RTSNoC, da seguinte forma:

$$WCL_{packet(i)} = \sum_{j=1}^{h} 2(N_j) + 2k(f-1) + 2B \tag{4.6}$$

O valor de `B` foi multiplicado por dois porque nesta análise foi considerada a possibilidade de as duas memórias FIFO estarem cheias com outros *flits*, tanto na interface de rede no nodo de origem como na interface de rede no nodo de destino.

Note que os parâmetros na equação 4.6 são bem conhecidos. É importante lembrar que o algoritmo XY, implementado nos roteadores, é um algoritmo estático e impõe que todos os *flits* que pertencem a um mesmo pacote devem ser encaminhados por um único caminho. Devido ao comportamento deste algoritmo, o valor máximo de `N` será sempre o mesmo, porque o número de núcleos e roteadores no caminho são fixos. O parâmetro `k` também é bem conhecido devido ao tamanho da rede, e, portanto, é possível presumir o número máximo de fluxos originados em outros nodos da rede e que podem competir pelo mesmo nodo de destino. Além disso, o parâmetro `B` é definido durante a definição do projeto da rede e o parâmetro `f` é conhecido pelo nodo de origem. Isso significa que os fluxos de tempo real projetados considerando a WCL absoluta da rede RTSNoC terão sempre atendidas as suas restrições temporais, de modo que não *deadlines*[3] perdidos devido a contenção de rede.

4.5.1 Análise de WCL para fluxos BE

Apesar de uma comparação imediata entre as NoCs apresentadas no Capítulo 2 não poder ser feita de forma simples devido as diferentes configurações suportadas por cada uma daquelas redes, é possível realizar algumas simulações para entender o impacto no WCL causado por alterações no conjunto de fluxos na rede, desde que algumas premissas seja adotadas. Nesta subseção são apresentados os resultados referentes a uma simulação realizada com as expressões de latência de pior caso para as redes RTSNoC e Æthereal. No caso da rede Æthereal, está sendo considerado que a WCL é a expressão de latência que consta na Tabela 1, apresentada no final do Capítulo 2. Ou seja, foi uma simulação analítica com tais expressões e realizadas com auxílio de uma planilha eletrônica, tipo Excel®.

[3]Os sistemas de tempo real são caracterizados por atividades computacionais com restrições temporais. Uma restrição temporal típica é o *prazo de execução*, usualmente chamado de *deadline*, o qual representa o tempo antes do qual a tarefa deve ser concluída para evitar prejuízos ao sistema. Se um *deadline* é especificado em relação ao tempo de chegada, ele é chamado de *deadline* relativo, enquanto que se ele for especificado em relação ao tempo *zero*, ele é chamado de *deadline* absoluto (BUTTAZZO, 2005).

Como premissa básica, foi adotada a condição de que todo canal TDM da rede Æthereal pode ser utilizado por mais de um fluxo, gerando assim concorrência entre dois ou mais fluxos por recursos daquela rede. Esta é uma situação prevista pelos próprios autores da Æthereal que afirmam ser possível o uso de canais dedicados para fluxos GT por fluxos BE, quando aqueles fluxos prioritários não estiverem sendo utilizados.

Tabela 4 – Lista de parâmetros adotados nas simulações de WCL considerando as funções $f(x, y)$ e $g(x, y)$.

Parâmetro	Valor
B	3
n	1
f	3
p	1
P	4
s	3
N	variável
h	6

Os valores adotados nas equações de latência direta foram $B = 3$ e $h = 6$. O número de *flits* e número de fluxos que competem por um recurso (N para a rede RTSNoC) foram usados como variáveis. Assim, foram obtidas duas equações em função de tais variáveis: $f(x, y) = f(flit, \sigma)$ para a rede Æthereal e $g(x, y) = g(flit, \sigma)$ para a rede RTSNoC. O número de *flits* por pacote ficou dentro do intervalo $[1; 50]$ e o número de fluxos que competem por um mesmo recurso ficou dentro do intervalo $[0; 10]$.

A Figura 51 apresenta os resultados das duas superfícies geradas por $f(x, y)$ e $g(x, y)$. Naquela Figura, a superfície na cor cinza claro representa a função $f(x, y)$ da rede Æthereal e a superfície na cor cinza escuro representa a função $g(x, y)$ da rede RTSNoC. A linha de intersecção entre as duas superfícies geradas pelas funções $f(x, y)$ e $g(x, y)$ é o limite entre as duas redes, sob o ponto de vista das relações custo-benefício envolvendo a quantidade de fluxos, o número de *flits* por pacote e a WCL. Note que o valor da WCL para a rede RTSNoC foi menor do que o valor apresentado pela rede Æthereal para pacotes com quantidades menores de *flits*, mesmo com um grande número de fluxos competindo pelos mesmos recursos da rede. Isto acontece devido ao tempo extra que a rede Æthereal necessita para alocar e liberar os canais TDM para os fluxos de melhor esforço. A RTSNoC não necessita daquele tempo extra porque ela não faz reserva de recursos da rede.

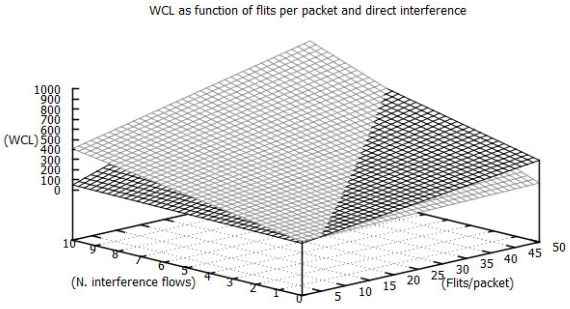


Figura 51 – Simulação de WCL para as redes Æthereal e RTSNoC.

4.6 CONSIDERAÇÕES

Neste Capítulo foi apresentado o projeto lógico da rede RTSNoC. Inicialmente foi apresentada uma seção descrevendo em linhas gerais o conceito da rede proposta. Em seguida, uma seção foi dedicada para apresentar as definições de escopo do trabalho, descrevendo o segmento de SoC alvo para esta rede, as tecnologias adotadas nos experimentos e as métricas de avaliação da rede proposta. Uma terceira seção apresentou de forma detalhada o desenvolvimento da rede, estabelecendo a topologia à qual ela se aplica, o formato da comunicação e os principais elementos internos que constituem os roteadores da rede.

As duas últimas seções do capítulo foram dedicadas para tratar da latência de pior caso e da largura de banda. A primeira delas apresentou a expressão matemática que permite a obtenção do WCL para um pacote na rede, seguida de uma análise teórica da latência de pior caso em uma rede com alocação de recursos que permite fluxos BE e a RTSNoC. A segunda seção descreveu como foi obtida a largura de banda de dois ciclos de *clock* na RTSNoC e também apresentou uma análise teórica da vazão que poderia ser esperada para fluxos BE na rede RTSNoC e na rede com alocação de recursos usada como referência. Os valores de WCL e largura de banda foram avaliados em

uma rede RTSNoC com quatro roteadores, sintetizada em FPGA. O próximo capítulo descreve a rede usada nos experimentos e apresenta os valores medidos de latência e vazão, confrontando tais resultados com os valores teóricos esperados para aquela rede.

5 AVALIAÇÃO EXPERIMENTAL

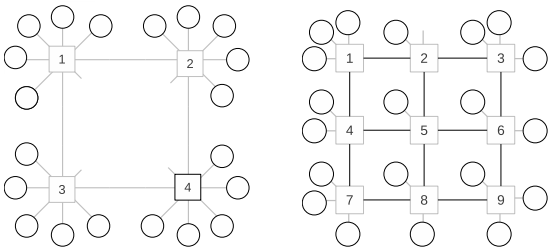
Este capítulo apresenta os resultados experimentais baseados em experimentos realizados com redes RTSNoC, sendo que o Capítulo foi separado em seções. Na primeira seção apresenta resultados relativos à tecnologia FPGA. Ela é dividida em duas subseções, sendo que a primeira avalia o impacto no uso de recursos de silício devido à adoção de até oito canais por roteador, e na segunda subseção é feita a avaliação de uso de recursos de silício devido à inclusão de sinais de roteamento em todos os *flits* da rede RTSNoC. A segunda seção apresenta resultados relativos à tecnologia ASIC, no qual o consumo dos componentes internos do roteador RTSNoC são apresentados e comparados à outras NoCs que tiveram resultados publicados em artigos científicos na tecnologia de 90 nm. A terceira seção apresenta os resultados de experimentos realizados com uma rede RTSNoC composta por quatro roteadores e sintetizada para um dispositivo FPGA. O objetivo foi realizar medidas de latência de pior caso e vazão e confrontar os resultados com os valores teóricos esperados para esta rede. Finalizando o capítulo, a quarta seção descreve os experimentos realizados para a avaliação da latência média oferecida pela rede na comunicação entre nodos da rede. Os valores médios de latência foram comparados graficamente com valores teóricos esperados para a RTSNoC e para a rede Æthereal, sendo que nesta última foi considerada a competição de fluxos BE por canais TDM daquela rede.

5.1 USO DE RECURSOS DE SILÍCIO EM FPGA

5.1.1 Análise com diferentes quantidades de canais

Para entender o impacto causado pela adoção de oito canais de comunicação por roteador, foram realizadas duas sínteses para FPGA de duas redes interconectando vinte núcleos cada uma. Uma das redes foi implementada com roteadores RTSNoC com oito canais de comunicação, ilustrado na Figura 52(a), e a outra rede foi implementada com roteadores RTSNoC com cinco canais de comunicação, conforme mostra a Figura 52(b).

Os roteadores foram sintetizados com canais de trinta e dois bits. Os valores de uso de recursos de silício foram obtidos pela soma da quantidade de *Slice Registers* com os *Slice LUTs* para cada con-



(a) Vinte núcleos interconectados por quatro roteadores, com oito canais por roteador.

(b) Vinte núcleos interconectados por nove roteadores, com cinco canais por roteador.

Figura 52 – Redes usadas nas análises de uso de recursos de silício.

figuração. Os resultados de uso de recursos de silício por roteador e por configuração de rede são mostrados na Tabela 5, em que o uso de recursos de silício para a configuração de rede com oito canais de comunicação foi 28,55% menor do que a rede configurada com roteadores de cinco canais de comunicação.

Tabela 5 – Uso de recursos de silício para diferentes configurações da RTSNoC.

Canais/roteador	Consumo/roteador	Consumo/rede
5 canais	2235	21120 (9 roteadores)
8 canais	3576	15090 (4 roteadores)

5.1.2 Análise sobre o tamanho do *flit*

Conforme mencionado na Seção 4.3 que tratou do desenvolvimento da RTSNoC, esta rede permite a configuração de quatro parâmetros distintos em tempo de projeto: o tamanho do *flit*, o tamanho do pacote, o número de canais em cada roteador e o tamanho dos *buffers* da interface de rede. Esta subseção apresenta uma análise sobre o custo de silício devido a inclusão de dados de roteamento para todos os *flits* de um pacote na RTSNoC.

Para avaliar o uso de recursos de silício devido ao tamanho do *flit* foram variados o número de bits extras necessários para atingir

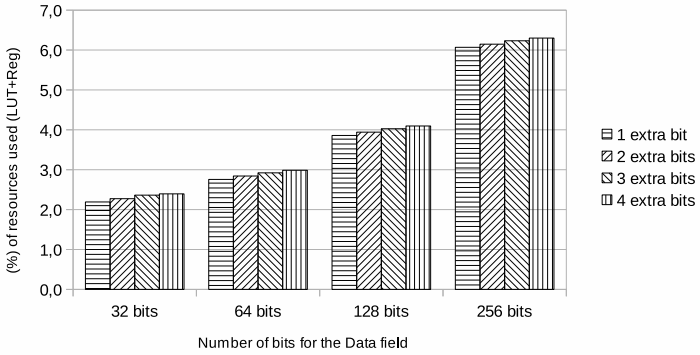
diferentes tamanhos de rede (i.e. o número de bits para cada campo de endereçamento), e o número de bits do campo de dados (*Data field*) no canal de comunicação (16, 32, 64, 128 e 256 bits). A Tabela 6 resume os resultados no uso de recursos de silício, em termos de *Slice Registers* e *Slice LUTs*.

Tabela 6 – Uso de recursos de silício no FPGA variando o tamanho das redes e o tamanho do campo de dados.

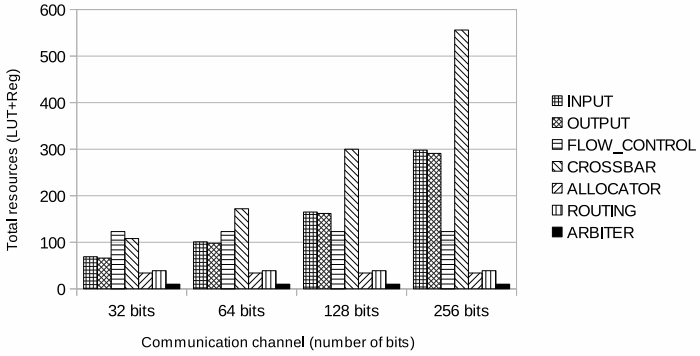
N. de bits extras	*Slice Registers*	*Slice LUTs*	% usado
Canal com 16 bits			
1	850	1821	1.9%
2	894	1889	2.0%
3	937	1669	2.1%
4	979	2021	2.1%
Canal com 32 bits			
1	991	2070	2.2%
2	1031	2145	2.3%
3	1074	2225	2.4%
4	118	2277	2.4%
Canal com 64 bits			
1	1262	2589	2.8%
2	1302	2667	2.8%
3	1344	2737	2.9%
4	1381	2789	3.0%
Canal com 128 bits			
1	1776	3613	3.9%
2	1825	3681	3.9%
3	1864	3761	4.0%
4	1908	3813	4.1%
Canal com 256 bits			
1	2815	5661	6.1%
2	2857	5729	6.1%
3	2895	5809	6.2%
4	2938	5861	6.3%

Os valores da Tabela 6 foram plotados na Figure 53(a). Note que o aumento no número de bits extras para endereçar redes em malha 2×2 até 16×16 não é muito significativo para um mesmo tamanho de campo de dados. O uso de recursos de silício é mais relevante quando o tamanho do campo de dados aumenta. Por exemplo, o aumento no consumo médio de silício devido ao crescimento nos bits usados para endereçar roteadores na rede foi de 0,24 %, enquanto o consumo cresceu em média até 4,18 % quando o campo de dados muda de 16 bits para 256 bits.

A Figura 53(b) mostra o aumento no uso de recursos por componentes do roteador RSTSNoC, considerando diferentes tamanhos de campos de dados dos *flits*. Naquele caso, o consumo foi analisado considerando a soma no consumo de *Slice Registers* com *Slice LUTs*. Como pode ser observado naquela Figura, o aumento no tamanho do campo



(a) Uso de recursos de silício relativo ao número de bits no campo de dados e ao número de bits extras usados no endereço de roteamento.



(b) Recursos usados pelos componentes de hardware do roteador RTSNoC.

Figura 53 – Uso de recursos de silício no FPGA variando o tamanho das redes e o tamanho do campo de dados.

de dados do *flit* afeta o *crossbar*, representado como "multiplexadores", e nas interfaces de entrada e saída do roteador. As interfaces de entrada e saída necessitam mais registradores e *SLICEs* para implementar os seus circuitos e aceitar mais bits nos canais de comunicação. O crescimento no uso de recursos de silício foi maior para a *crossbar* devido a sua estrutura dos multiplexadores internos usados na sua implementação serem baseados apenas em lógica combinacional, tendo

que multiplexar oito canais de entrada para cada um dos oito canais de saída, ou seja, oito multiplexadores 8×1.

Outros componentes, como o controlador de fluxo, alocador, controlador de roteamento e árbitro, são menos afetados pelo aumento no número de bits no campo de dados. eles utilizam apenas as informações de roteamento adicionada aos *flits*. Ou seja, a informação de roteamento depende do número de roteadores que precisam ser endereçados na rede.

5.2 USO DE RECURSOS DE SILÍCIO EM ASIC

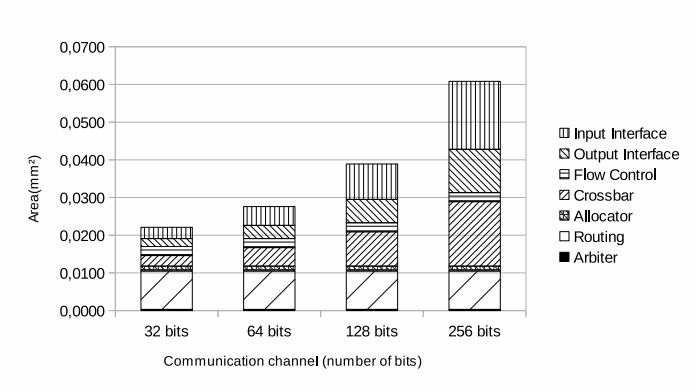
Esta seção apresenta os resultados relativos à análise que foi feita considerando a área de silício consumida pelo roteador RTSNoC quando sintetizado na tecnologia ASIC. Para isso, foi utilizada a ferramenta de síntese *Synopsys*® *Ultra Design Compiler* utilizando a biblioteca SAED 90nm *Low Power* e os resultados foram obtidos em termos de área de silício para um único roteador, considerando differentes tamanhos de canais de comunicação.

O resultado da síntese para a tecnologia de 90nm é apresentada na Tabela 7. O roteador RTSNoC foi implementado considerando oito tamanhos de canais diferentes, de 32 bits até 256 bits. Por exemplo, com estes tamanhos de canais poderiam ser conectados na rede RTSNoC processadores de 32 bits ou GPUs (acrônimo de *Graphic Processing Unit* - Unidade de Processamento Gráfico). A interface com os núcleos de processamento conectados à rede precisa ser adaptada de acordo com a interface de comunicação de cada núcleo. Por esta razão, a área de silício desta interface não foi incluída na Tabela 7.

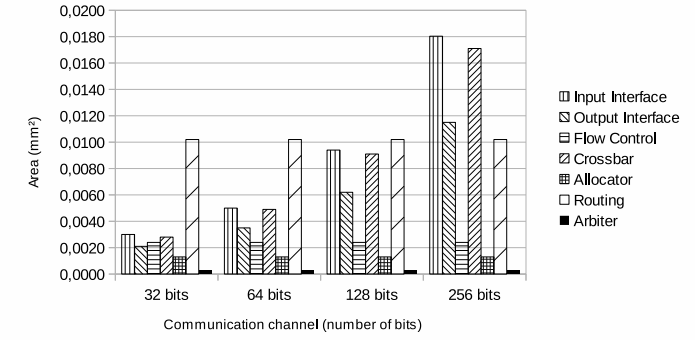
A última coluna da Tabela 7 apresenta, de modo aproximado, o número de portas lógicas usados para implementar o roteador RTSNoC na tecnologia de 90nm. O número de portas lógicas foi obtido através da relação entre a área do componente e a área da menor porta lógica da tecnologia alvo. Para a tecnologia SAED 90nm, tal porta lógica é uma porta NAND com $5.5296 \mu\text{m}^2$. Esta referência é útil para se ter uma estimativa da área de silício consumida em outras tecnologias, como por exemplo 65nm. Além do consumo de área para cada componente do roteador RTSNoC, a Tabela 7 apresenta a contribuição percentual de cada componente na área total de silício utilizada pelo roteador.

Os resultados da Tabela 7 são apresentados graficamente nas Figuras 54(a) e 54(b), sendo que a Figura 54(a) apresenta a contribuição dos componentes internos do roteador em relação à área total de silício

na tecnologia SAED 90*n*m, enquanto que a Figura 54(b) apresenta a área de silício consumida por cada um dos componentes internos do roteador RTSNoC para esta tecnologia. Pode-se observar naquelas Figuras que a interface de entrada (*Input Interface*), a interface de saída (*Output Interface*) e a *Crossbar* têm a maior influência no consumo de área de silício quando aumenta o número de bits do canal de comunicação da rede, similar ao que acontece na tecnologia FPGA.



(a) Contribuição dos componentes internos do roteador RTSNoC em relação à área total de silício.



(b) Área de silício consumida por cada um dos componentes internos do roteador RTSNoC.

Figura 54 – Uso de recursos de silício em ASIC com tecnologia de 90 nm.

Tabela 7 – Área de silício (em mm^2) usada por um único roteador RTSNoC e considerando diferentes tamanhos de canais de dados.

Canal (dados)	Entrada	Saída	Controle de fluxo	MUX	Alocador	Roteamento	Arbitro	Área total	Gates lógicos
32 bits	0.009 (9.44%)	0.0021 (2.25%)	0.0024 (2.47%)	0.0028 (2.96%)	0.0013 (1.32%)	0.0102 (0.35%)	0.0003 (0.35%)	0.0954	0.0172
64 bits	0.0150 (11.95%)	0.0035 (2.77%)	0.0024 (1.88%)	0.0049 (3.93%)	0.0013 (1.01%)	0.0102 (0.27%)	0.0003 (0.27%)	0.1252	0.0226
96 bits	0.0218 (13.97%)	0.0048 (3.09%)	0.0024 (1.51%)	0.0069 (4.43%)	0.0013 (0.81%)	0.0102 (0.22%)	0.0003 (0.22%)	0.1560	0.0282
128 bits	0.0282 (14.48%)	0.0062 (3.18%)	0.0024 (1.21%)	0.0091 (4.69%)	0.0013 (0.65%)	0.0102 (0.18%)	0.0003 (0.18%)	0.1946	0.0352
160 bits	0.0346 (15.71%)	0.0076 (3.43%)	0.0024 (1.07%)	0.0112 (5.09%)	0.0013 (0.57%)	0.0102 (0.16%)	0.0003 (0.16%)	0.2200	0.0398
192 bits	0.0412 (16.56%)	0.0089 (3.57%)	0.0024 (0.95%)	0.0134 (5.38%)	0.0013 (0.51%)	0.0102 (0.14%)	0.0003 (0.14%)	0.2489	0.0450
224 bits	0.0477 (17.13%)	0.0101 (3.63%)	0.0024 (0.85%)	0.0155 (5.57%)	0.0013 (0.45%)	0.0102 (0.12%)	0.0003 (0.12%)	0.2782	0.0503
256 bits	0.0541 (17.13%)	0.0115 (3.64%)	0.0024 (0.75%)	0.0171 (5.41%)	0.0013 (0.40%)	0.0102 (0.11%)	0.0003 (0.11%)	0.3157	0.0571

O consumo de área de silício da RTSNoC também foi comparado com o custo de área para outras NoCs apresentadas na literatura. Uma comparação conclusiva é difícil devido às diferentes técnicas adotadas por cada uma das redes, pois algumas delas implementam mecanismos de alocação de recursos, como TDM, enquanto que outras são baseadas na arbitragem em tempo de execução, e isto tem influência na relação custo-benefício de cada NoC; além disso, os resultados publicados em artigos científicos muitas vezes referem-se à tecnologias diferentes, como 65nm ou 130nm.

Apesar destas dificuldades, é possível fazer uma análise qualitativa de outras redes com a rede RTSNoC, desde que sejam comparadas apenas tecnologias similares (no caso 90nm)e abstraindo as questões relativas à relação custo-benefício de cada rede. A Tabela 8 apresenta os resultados de consumo de área de silício da rede RTSNoC para algumas configurações e compara estes resultados aos valores fornecidos por outras redes publicadas em artigos científicos e que também utilizaram a tecnologia de 90nm. O roteador RTSNoC com canais de comunicação de 32 bits (*payload*) ocupou uma área de silício menor do que o roteador da rede *4S* proposto por (BANERJEE et al., 2009), com 16 bits no canal de dados. Outra versão do roteador da RTSNoC, com 64 bits de *payload*, apresenta um consumo de área de silício 1,16 vezes maior que o roteador da rede *4S* de 16 bits. É importante salientar que o roteador da rede RTSNoC possui um *overhead* devido a inclusão de dados de roteamento para todos os *flits* que trafegam na rede, conforme comentado na Seção 4.3. Como referência, foi incluído o consumo da rede dAElite de 64 bits, proposta por (GOOSSENS; DIELISSEN; RADULESCU, 2012), e apresentada por seus autores como uma rede de baixo custo de silício.

Tabela 8 – Área consumida pelo roteador RTSNoC comparado à outras implementações na tecnologia de 90 nm.

Roteador (NoC)	Características	Área (mm^2)
4S Project	4 SDM lanes 16 bit/lane	0.108
RTSNoC	32 bit (*payload*) e 8 canais	0.095
dAElite	64 bit/canal divididos em 4 *slots* TDM	0.016
RTSNoC	64 bit (*payload*) e 8 canais	0.125

5.3 AVALIAÇÃO DA LATÊNCIA DE PIOR CASO E DA VAZÃO NA REDE RTSNOC

Nas duas últimas seções do Capítulo 4 foram apresentadas a expressão matemática, com a qual é possível calcular o valor da latência de pior caso para a transmissão de pacotes na rede RTSNoC, e a largura de banda esperada nos canais de comunicação da rede RTSNoC. Nesta seção são apresentados os resultados de experimentos realizados com uma rede RTSNoC composta por quatro roteadores e sintetizada para um dispositivo FPGA. O objetivo foi realizar medidas de latência e vazão e confrontar os resultados com os valores teóricos esperados para esta rede.

A medida de latência de um pacote foi feita contabilizando o número de ciclos de *clock* decorridos desde a colocação do primeiro *flit* no canal de entrada de um nodo de origem até a chegada do último *flit* deste pacote no nodo de destino. A Figura 55 ilustra um exemplo genérico desta medida de latência. Um pacote, pertencente ao fluxo σ_i, e composto pelos *flits* $f_0, f_1, ..., f_{n-1}$ é injetado na rede pelo nodo A conectado ao roteador 2, e cujo destino é o nodo B conectado ao roteador 1. O início da medida de latência do pacote acontece quando o *flit* de cabeçalho (f_0) é colocado no canal de entrada do roteador 2, indicado por t na Figura 55; o final da medida acontece quando o *flit* terminador (f_{n-1}) é disponibilizado no canal de saída do roteador 1, e indicado por $t + t'$ naquela Figura.

Nas Subseções que seguem é apresentada a rede usada nos experimentos. Os resultados de latência e vazão obtidos são apresentados e confrontados com os valores teóricos esperados para aquela rede.

5.3.1 Rede utilizada nos experimentos

A rede utilizada nos experimentos foi implementada com quatro roteadores, denominados de 1, 2, 3 e 4, e representados por quadrados brancos na Figura 56. Vinte e quatro núcleos que geram tráfego na rede, representados por círculos e numerados de 0 a 23, foram conectados naqueles roteadores. Destes núcleos, cinco geram pacotes para o mesmo núcleo de destino. Os cinco núcleos que geram tais pacotes são representados por círculos preenchidos na cor cinza e numerados como 3, 7, 13, 18 e 23, enquanto que o núcleo de destino destes pacotes é representado por um círculo preenchido na cor preta e numerado como 12.

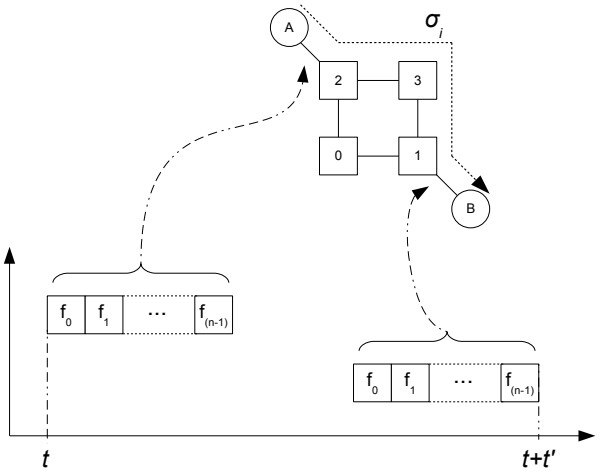


Figura 55 – Representação da medida adotada para a latência de um pacote na rede RTSNoC.

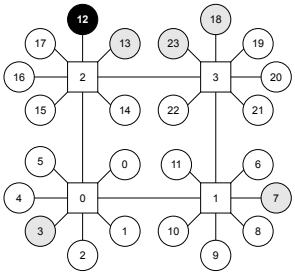


Figura 56 – Rede com 4 roteadores e 24 núcleos usada nos experimentos.

Os pacotes gerados pelos cinco núcleos são compostos por seis *flits*. Para diferenciá-los, foi adotado o padrão mostrado na Tabela 9, em que os *flits* de cabeçalho e terminador de cada pacote possuem o byte mais significativo com o valor 4_h, enquanto que os *flits* referentes a carga útil possuem o valor 0_h. Além disso, os dois bytes menos significativos de cada *flit* de um determinado pacote possuem valores que os diferenciam dos *flits* de outros pacotes. Por exemplo, os *flits* pertencentes ao pacote do fluxo σ_3 terminam com os bytes $3X_h$, em

que X é o número do *flit*, como 1, 2 e assim por diante.

Tabela 9 – Pacotes usados na avaliação da latência e vazão da rede RTSNoC.

	σ_3	σ_7	σ_{13}	σ_{18}	σ_{23}
header	40831_h	40871_h	$408D1_h$	$408E1_h$	$408F1_h$
payload	00832_h	00872_h	$008D2_h$	$008E2_h$	$008F2_h$
	00833_h	00873_h	$008D3_h$	$008E3_h$	$008F3_h$
	00834_h	00874_h	$008D4_h$	$008E4_h$	$008F4_h$
	00835_h	00875_h	$008D5_h$	$008E5_h$	$008F5_h$
tail	40836_h	40876_h	$408D6_h$	$408E6_h$	$408F6_h$

As medidas de latência e vazão foram feitas sobre os *flits* gerados pelo núcleo 7 e os resultados são apresentados nas subseções a seguir.

5.3.2 Medidas de Latência e Vazão

As medidas de latência e vazão de um pacote que trafega na rede experimental descrita na subseção anterior foram feitas adotando as seguintes condições:

- os fluxos σ_{03}, σ_{13}, σ_{18} e σ_{23} geram pacotes indefinidamente, de modo que logo após o envio do *flit* terminador de um pacote, inicia-se o envio do *flit* de cabeçalho pertencente a um novo pacote; e

- o fluxo σ_{07} gera apenas um pacote, sobre o qual são realizadas as medidas de latência e vazão.

A Figura 57(a) apresenta o resultado da simulação feita com a ferramenta ISim®. A frequência de *clock* adotada na ferramenta de simulação foi de $100MH_z$, de modo que cada ciclo de *clock* tem um período de $10ns$. O *flit* de cabeçalho é colocado no canal de entrada do roteador 1 no instante indicado pela letra A ($130ns$) na Figura 57(a), e contornado por um quadrado em linhas pontilhadas. O *flit* de cabeçalho é disponibilizado no canal de saída do roteador 2, onde está localizado o núcleo de destino 12, depois de doze ciclos de *clock*. Este instante é representado na Figura 57(a) pela letra B em que o *flit* é contornado por um quadrado em linhas pontilhadas.

O valor teórico esperado para a latência do primeiro *flit* pode ser calculado utilizando-se a equação 4.6 da seguinte forma:

$$WCL_{header} = \sum_{i=1}^{h} 2N_i = (2 \times 1) + (2 \times 2) + (2 \times 3) = 12 \qquad (5.1)$$

Na RTSNoC o roteamento é XY e com isso todos os *flits* gerados pelo núcleo 7 seguem o caminho entre os roteadores $1 \rightarrow 0 \rightarrow 2$. Assim, o *flit* de cabeçalho inicialmente leva dois ciclos de *clock* para atravessar o roteador 1 porque não há competição com outros fluxos naquele roteador. Para atravessar o roteador 0 podem ser necessários quatro ciclos, pois há a possibilidade de competição pelo mesmo canal de saída com outro *flit* gerado pelo núcleo 03. Por fim, o *flit* de cabeçalho pode necessitar de seis ciclos de *clock* para ser entregue no canal de destino no roteador 2, pois ele compete com o *flit* gerado pelo núcleo 13 e outro *flit* recebido no canal EE daquele roteador. É importante lembrar que estes são valores de pior caso. Dependendo do instante em que o primeiro *flit* é colocado no canal de entrada do rotedor 1 ele pode ter um nível de prioridade maior em determinado roteador do caminho.

A latência teórica para a entrega de um pacote na rede utilizada no experimento pode ser calculada com o uso da Equação 4.6. Neste caso, está sendo assumido como premissa que a leitura dos *flits* entregues no nodo de destino é feita imediatamente após a indicação da chegada de cada *flit*, e sem a possibilidade de encher os *buffers* de memória nas interfaces de rede dos nodos de origem e destino dos pacotes. Assim, o termo $2B$ referente ao tamanho dos *buffers* na equação 4.6 está sendo considerado nulo, e o valor teórico da latência de pior caso para um pacote pertencente ao fluxo σ_{07} (WCL_{07}) é calculado da seguinte forma:

$$WCL_{07} = \sum_{i=1}^{h} (2N_i) + 2k(f-1) = 12 + 2 \times 5 \times (6-1) = 62 \qquad (5.2)$$

Pode-se observar na Figura 57(a) que a latência total para a entrega do pacote é de 62 ciclos de *clock*, sendo doze ciclos para o *flit* de cabeçalho e cinquenta ciclos para os outros cinco *flits*. Neste experimento foi colocado o primeiro *flit* do pacote a ser analisado em um instante de tempo em que foi possível verificar a máxima latência teórica ocorrendo para um pacote. Em outros experimentos feitos com esta rede o valor da latência do *flit* de cabeçalho variou, conforme era esperado, porém sempre abaixo no limite máximo teórico. Além disso,

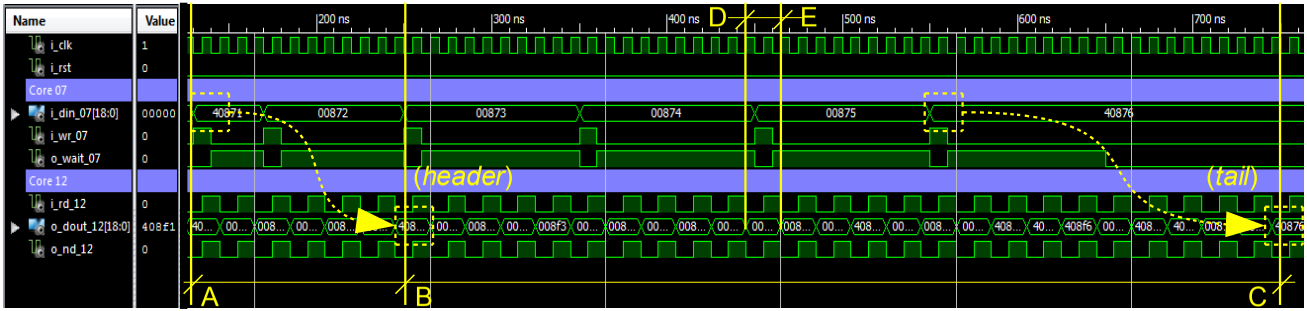
não houve alteração na ordem de entrega dos *flits* do pacote analisado. Isto era esperado, pois o algoritmo de roteamento XY adotado na rede obriga o envio de todos os *flits* de um mesmo pacote a seguirem por um caminho pré-estabelecido. Assim, um determinado *flit* só será entregue após a entrega de seu antecessor. Na mesma Figura, o intervalo indicado pelas letras D e E mostra que um *flit* é entregue no canal de saída a cada dois ciclos de *clock*, confirmando a vazão de um *flit* a cada dois ciclos de *clock*.

A largura de banda é dividida de modo igualitário entre os fluxos que competem pelos mesmos recursos na rede RTSNoC. Para facilitar a visualização dos resultados neste documento, o número de *flits* dos pacotes usados no experimento foi reduzido para quatro, conforme mostra a Tabela 10.

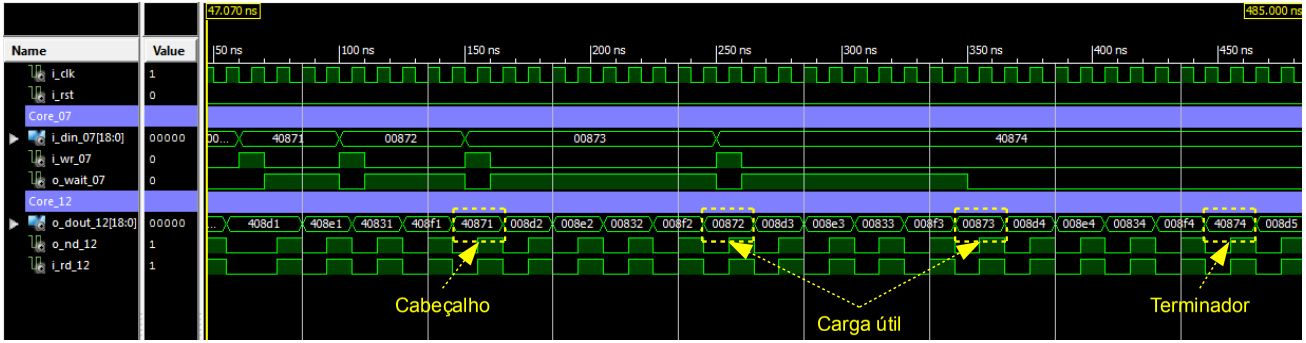
Tabela 10 – Pacotes usados na avaliação da ordem de entrega de *flits* na RTSNoC.

	σ_3	σ_7	σ_{13}	σ_{18}	σ_{23}
header	40831_h	40871_h	$408D1_h$	$408E1_h$	$408F1_h$
payload	00832_h	00872_h	$008D2_h$	$008E2_h$	$008F2_h$
	00833_h	00873_h	$008D3_h$	$008E3_h$	$008F3_h$
tail	40834_h	40874_h	$408D4_h$	$408E4_h$	$408F4_h$

A Figura 57(b) ilustra o resultado do experimento, em que os *flits* pertencentes ao fluxo sob análise (σ_7) foram circundados por quadrados com linhas pontilhadas. Conforme pode ser observado na Figura, não houve troca de ordem na entrega dos *flits* pertencentes ao fluxo σ_7 Note ainda que não houve entrega de dois *flits* consecutivos de um mesmo pacote no período de tempo em que foi feita a entrega do pacotes de teste, e todos os fluxos do experimento tiveram o mesmo período entre *flits* na entrega dos seus pacotes, confirmando que a largura de banda é dividida de modo igualitário entre os fluxos que competem pelos mesmos recursos na rede RTSNoC.



(a) Medida de latência para um pacote do fluxo σ_{07} na rede utilizada no experimento.



(b) Avaliação da entrega de pacotes pela rede utilizada no experimento.

Figura 57 – Avaliações na rede RTSNoC utilizando a ferramenta de simulação ISim®.

5.4 AVALIAÇÃO DA LATÊNCIA MÉDIA

Na Seção 4.5.1 foi apresentada uma simulação realizada com as redes RTSNoC e Æthereal levando em conta valores teóricos de quantidade de fluxos e tamanho dos pacotes para aquelas redes. Esta seção apresenta dois resultados experimentais que foram obtidos para verificar a latência média que os fluxos, relacionados a aplicações de tempo real *soft*, teriam em diferentes configurações da rede RTSNoC. Para avaliar tais latências, duas redes compostas por roteadores RTSNoC foram implementadas e separadas em dois cenários. Estas redes possuem núcleos conectados à elas que oferecem dois serviços: alguns núcleos geram tráfego nas redes, enquanto que outros medem a latência de pacotes gerados pelos núcleos geradores. A latência média medida naquelas redes foi plotada em relação aos valores teóricos de latência das redes Æthereal e RTSNoC. Nas subseções que seguem são apresentados os experimentos e discutidos os resultados obtidos.

5.4.1 Redes usadas nos experimentos

Foram desenvolvidas duas redes compostas por roteadores RTSNoC, ilustradas na Figura 58. As duas redes possuem dois tipos de núcleos conectadas à elas: geradores de tráfego (ou *Traffic Generators* - TG) e medidores de tráfego (ou *Trafic measurements* - TM).

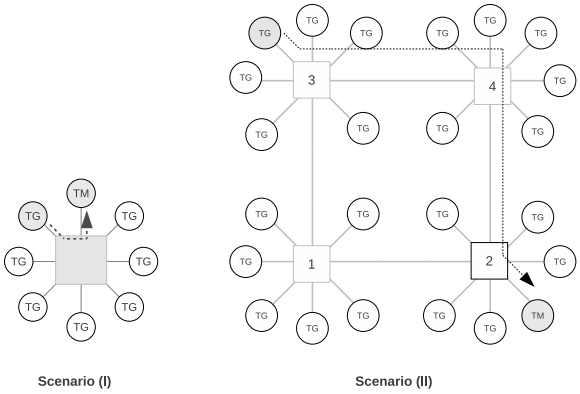


Figura 58 – Redes usadas na avaliação de latência média: (I) RTSNoC com 7 núcleos e (II) RTSNoC com 23 núcleos.

A fim de avaliar apenas a latência média oferecida pelas redes, foi considerado que existe apenas uma tarefa (ou *thread*) gerando pacotes em cada um dos geradores de tráfego TG. Esta consideração é essencial para garantir que a análise está sendo feita somente considerando o atendimento das restrições temporais em um sistema de tempo real baseado na rede RTSNoC, sem preocupações com questões relacionadas a aplicações em execução nos núcleos conectados nas rede. Os TGs geram quantidades aleatórias de *flits* por pacote a cada período de tempo. Os pacotes podem ter de três até trinta e seis *flits*, e cada pacote gerado possui diferentes quantidades de *flits*. Isto significa que quando a transmissão de um pacote termina, o próximo pacote gerado terá tamanho diferente.

5.4.2 Formato dos pacotes usados nos experimentos

A Figura 59 mostra um exemplo de pacote gerado por um TG. Naquela Figura, o TG gera um pacote `A` que requer um canal de comunicação por t_{packet} ciclos de *clock*. Após este tempo, o TG aguarda por t_{slack} ciclos de *clock* antes de gerar um novo pacote com tamanho diferente (pacote `B` naquela Figura).

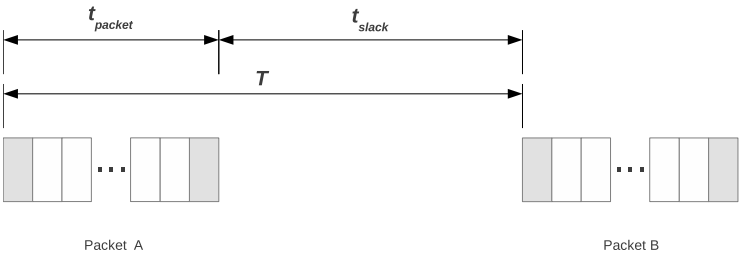


Figura 59 – Exemplo de pacotes gerados por um TG.

Foram estabelecidos cinco padrões para os TGs. Os padrões foram estabelecidos na seguinte relação entre t_{packet} e t_{slack}, apresentada na Figura 59:

$$IR = \frac{t_{packet}}{t_{packet} + t_{slack}} * 100(\%) \tag{5.3}$$

sendo que IR é chamado de taxa de injeção (em inglês *Injection Rate*) e os TGs mantém IR constante. Por exemplo, uma taxa de 20%

significa que os TGs geram pacotes e os enviam durante 20% de cada período T. Isto significa que, a cada novo pacote, o valor de t_{slack} precisa ser adaptado para manter a taxa de 20% do valor de t_{packet}. Nos experimentos foram feitas trinta medidas consecutivas de latência para diferentes valores de IR, nos dois cenários de rede mostrados na Figura 58 (20, 40, 60, 80 and 100%).

Este formato de pacotes foi escolhido por ser uma representação próxima da realidade de um sistema multimídia com taxa variável de bits. Neste ponto, é importante lembrar que uma das contribuições desta Tese, apresentada no Capítulo 1, é que todos os fluxos possíveis entre dois núcleos quaisquer da rede RTSNoC possuem valores de latência de pior caso fixos, e estes valores são independentes de possíveis variações na taxa de bits dos outros fluxos que competem pelos mesmos recursos da rede. Assim, buscou-se um modelo de gerador de tráfego que pudesse gerar tráfego com variações na taxa de bits.

O tráfego gerado pela Equação 5.3 é randômico. Isto significa que não há correlação entre amostras subsequentes geradas por aquela equação. Segundo os autores (GARRET, J., WITTEN, D., HASTIE, T., TIBSHIRANI, R., 2013), uma forma de determinar se as amostras de um conjunto de dados são correlacionadas é pela análise do "coeficiente de correlação", dado pela seguinte equação:

$$r = \frac{\frac{(\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y}))}{(N-1)}}{\sqrt{\frac{\sum_{i=1}^{N}(x_i-\overline{x})^2}{(N-1)}}\cdot\sqrt{\frac{\sum_{i=1}^{N}(y_i-\overline{y})^2}{(N-1)}}} \tag{5.4}$$

em que `N` é a quantidade de amostras do conjunto de dados, `x` e `y` são amostras subsequentes, $\overline{x}$ e $\overline{y}$ são os valores médios das amostras subsequentes e `r` é o coeficiente de correlação entre as amostras do conjunto de dados sob análise. Na Equação 5.4, a expressão do numerador da representa a "covariância" entre amostras subsequentes `x` e `y`, e o denominador é o produto do "desvio padrão" das amostras `x` e `y`. Para (GARRET, J., WITTEN, D., HASTIE, T., TIBSHIRANI, R., 2013), o coeficiente de correlação, `r`, é uma medida da intensidade da relação entre as amostras de um conjunto de dados, sendo que os valores de `r` podem variar entre `-1` e `+1`. O tipo de relação descrito por aqueles autores é representado pelo coeficiente de correlação da seguinte forma:

- Se (`r = +1`): correlação perfeitamente positiva;
- Se (`+1 < r < 0`): relação positiva;
- Se (`r = 0`): nenhuma relação;

- Se (`0 > r > -1`): relação negativa; e
- Se (`r = -1`): correlação perfeitamente negativa.

O coeficiente de correlação está limitado por `-1` ou `+1`. Quanto mais próximo o coeficiente estiver de `-1` ou `+1`, mais forte é a correlação. Pelo que foi exposto, pode-se dizer que um conjunto de dados randômico, sem correlação entre seus dados, precisa ter o valor do coeficiente de correlação (`r`) tendendo à zero. A Figura ilustra um histograma de um conjunto de dados obtidos por amostragem em um sinal cossenoidal com relação positiva, cujo coeficiente de correlação foi calculado pela Equação 5.4 como sendo `0,995`.

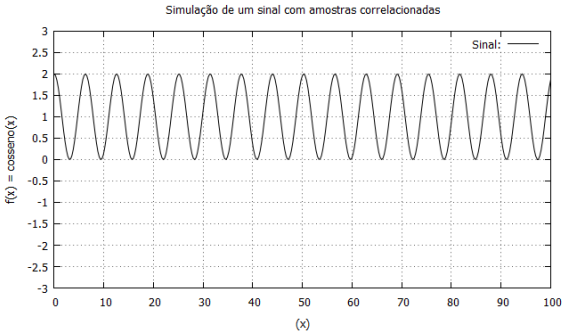
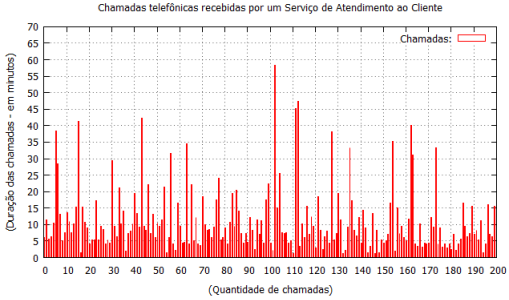


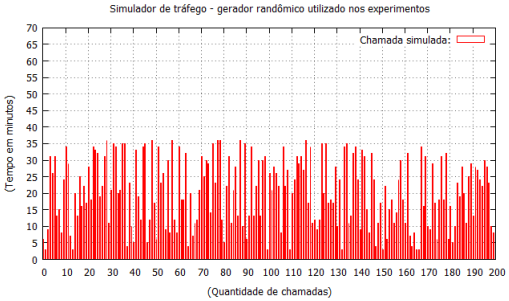
Figura 60 – Exemplo de um conjunto de dados relacionados obtidos de um sinal cossenoidal.

5.4.2.1 Comparação do gerador baseado em IR com um caso real

Para verificar se os dados gerados pela equação 5.3 podem ser utilizados para simular um tráfego multimídia real, foram feitas as avaliações analíticas em dois conjuntos de dados. O primeiro conjunto é referente ao relatório de chamadas telefônicas recebidas por um Serviço de Atendimento ao Cliente, do fabricante Intelbras S.A. O Serviço de Atendimento ao Cliente recebe diariamente cerca de 14 mil (14000) chamadas telefônicas, que são atendidas por cinquenta e seis (56) Pontos de Atendimento (PAs), em três turnos de trabalho consecutivos. As chamadas recebidas podem ter sido geradas por três diferentes tipos de compressão de voz: G711, G729 e G723. Além disso, o tempo de cada



(a) Histograma das chamadas recebidas por um Serviço de Atendimento ao Cliente, do fabricante Intelbras S.A.



(b) Histograma do conjunto de dados fornecido pelo gerador de tráfego apresentado na Equação 5.3.

Figura 61 – Resultados da comparação entre o gerador de tráfego baseado em IR com um caso real.

chamada atendida varia consideravelmente. Para este primeiro conjunto de dados foram utilizadas duzentas (200) amostras, e a Figura 61(a) mostra um histograma com o resultado. O segundo conjunto de dados foi gerado pelo gerador de tráfego baseado na Equação 5.3 e também foi gerado com duzentas amostras. O histograma deste segundo conjunto de dados é mostrado na Figura 61(b).

Os coeficientes de correlação para os dois conjuntos de dados também foram calculados utilizando a Equação 5.4. Para o primeiro conjunto, relacionado às chamadas do Serviço de Atendimento ao Cliente, o resultado obtido foi de `0,05` e para o segundo conjunto de dados, relativos ao gerador de tráfego, o resultado obtido foi de `-0,01`. A Tabela 11 mostra os resultados obtidos nas avaliações analíticas com os

conjuntos de dados citados nesta subseção. Considerando os valores obtidos, pode-se dizer que os dois conjuntos de dados podem ser considerados randômicos. Além disso, pode-se dizer que o gerador de tráfego proposto é adequado para gerar fluxos de dados para os experimentos propostos nesta seção.

Tabela 11 – Coeficiente de correlação entre os três sinais avaliados.

Sinal	*Coeficiente de Correlação*	*Correlação*
Figura 60	0,995	Alta
Figura 61(a)	0,05	Baixa
Figura 61(b)	-0,01	Baixa

5.4.3 Cenário de testes

Foi considerado que um TG em cada uma das redes experimentais é um processador executando uma única tarefa de tempo real *hard* e gerando pacotes que precisam ser entregues ao seu destino (TM) antes de atingir o seu limite teórico de latência, de modo a não comprometer o *deadline* da tarefa. Tais TGs estão representados como círculos preenchidos na cor cinza na Figura 58. Todos os outros TGs foram considerados como processadores que também executam apenas uma tarefa, porém são tarefas de tempo real *soft*.

Todos os *flits* gerados pelos TGs são enviados para o núcleo medidor de tráfego (TM) o qual mede a latência dos pacotes gerados pelos TGs. Ele calcula a latência média de todos os pacotes recebidos dos TGs. Além disso, o TM possui um mecanismo que informa se o limite de latência foi excedido por algum pacote, usando como referência o máximo valor teórico para a rede RTSNoC sob teste.

A subseção seguinte apresenta os resultados obtidos com os experimentos nas redes mostradas na Figura 58 e discute tais resultados utilizando como referência os valores teóricos esperados para redes Æthereal com quantidades similares de núcleos à elas conectados.

5.4.4 Resultados de Latência

As Figuras 62(a) e 62(b) mostram os resultados medidos para as latências nas redes RTSNoC. A latência média medida foi plotada em relação aos valores teóricos de latência para as redes Æthereal com sete núcleos (Figura 62(a)) e vinte e três núcleos (Figura 62(b)). Os

valores máximos teóricos de latência esperados para as redes RTSNoC também foram plotados nas mesmas Figuras.

As redes Æthereal são baseadas no conceito de períodos de *time slots* e no escalonamento daqueles *time slots* para os fluxos que circulam nas redes. Para encontrar o valor teórico de latência para as redes Æthereal e usar estes valores como referência, foi aplicada a expressão de latência mostrada na Tabela 1, ao final do Capítulo 2. As curvas de máximos valores teóricos para as redes Æthereal ilustradas nas Figuras 62(a) e 62(b) foram baseadas nas seguintes premissas:

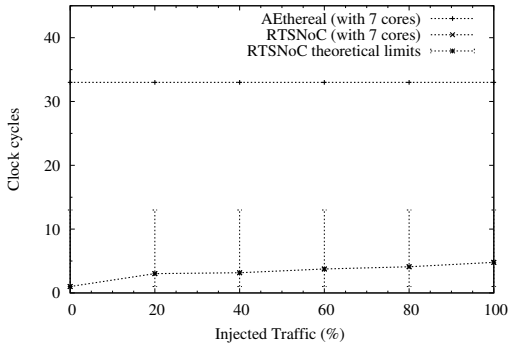
- as redes foram projetadas para oferecer 100% de garantias temporais; e
- existem *time slots* suficientes para todos os fluxos que trafegam naquelas redes.

É fato que redes com tais premissas possuem um custo de silício elevado, além de desperdiçarem largura de banda não utilizada (GOOSSENS; DIELISSEN; RADULESCU, 2005). No entanto, utilizando esta abordagem é possível considerar a latência previsível para todos os fluxos gerados por aplicações de tempo real *hard* e *soft* naquelas redes, e que é uma das propostas oferecidas pela rede RTSNoC discutida nesta Tese. Baseado nas premissas acima, assume-se aqui que os valores plotados para as redes Æthereal representam as máximas latências esperadas para os fluxos que circulam naquelas redes.

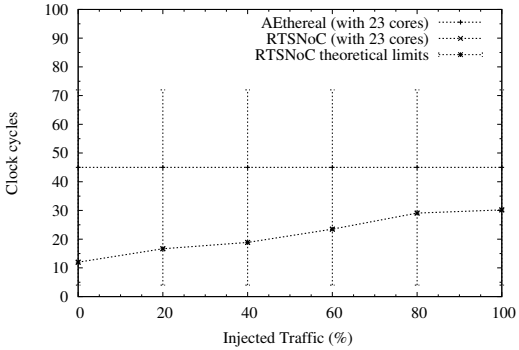
Além dos resultados obtidos nos experimentos, os valores teóricos de latência máxima e mínima esperados para as redes RTSNoC foram plotados como curvas de erro (*error-bar*). Os valores mínimos representam a latência direta (L_{direta}) e foram calculados considerando apenas os tráfegos gerados pelos TGs de referência, representados por círculos preenchidos na cor cinza nas redes mostradas na Figura 58. Os valores máximos representam o WCL e foram calculados assumindo o percentual de tráfego adicionado ao caminho entre os geradores de tráfego de referência e os medidores de tráfego (20, 40, 60, 80 and 100%).

É importante salientar que estes valores teóricos máximos e mínimos atribuídos à rede RTSNoC são pessimistas e não levam em consideração um dimensionamento de tráfego que considere limites de latência mais rígidos. Para que se possa atingir limites mais estreitos, uma análise estática pode ser feita antes da alocação dos núcleos na rede, de forma similar ao que é feito com as outras NoCs apresentadas na literatura.

Note que a latência não atingiu os limites máximos teóricos da RTSNoC. O único meio de atingir tais limites é considerando duas situações:



(a) Latência média *vs.* tráfego injetado para a rede RTSNoC com 7 núcleos.



(b) Latência média *vs.* tráfego injetado para a rede RTSNoC com 23 núcleos.

Figura 62 – Resultados das medidas de latência.

- todos os pacotes gerados por um TG possuem o mesmo tamanho; e
- um pacote precisa receber a menor prioridade em todos os roteadores no caminho, desde sua origem até seu destino.

Pelos valores expostos na Figura 62 é possível argumentar que a rede RTSNoC atingiu sua meta de oferecer um menor valor de latência média para todos os pacotes pertencentes a fluxos de tempo real *soft* que circulam pela rede. Além disso, não houve sinalização de pacotes que tenham ultrapassado os valores limites teóricos estabelecidos. Uma

NoC baseada na técnica TDM é inerentemente preditiva, mas o período dos *time slots* é maior para sistemas que apresentem previsibilidade de latência tanto para fluxos de tempo real *hard* como *soft*. Neste caso, para manter baixo o uso de recursos de silício, os projetista de redes TDM precisam reduzir a quantidade de *time slots* disponíveis para fluxos BE, o que implica no crescimento médio da latência para tais fluxos que passam a competir pelos mesmos recursos e precisam alocar e liberar os *time slots* constantemente.

5.5 CONSIDERAÇÕES

Neste capítulo foram apresentados e discutidos os custos de silício, devido a adoção de uma estrutura de *flit* na qual as informações de roteamento são anexadas a todos os *flits* e também os custos devido ao uso de um maior número de canais de comunicação por roteador. Conforme era esperado, a abordagem proposta para a rede RTSNoC apresentou um aumento no uso de recursos de silício, devido aos bits extras anexados aos *flits*. No entanto, a adoção de roteadores sem *buffers* minimiza este aumento, mesmo para redes compostas por roteadores com até oito canais de comunicação, apesar de isto ainda ser uma penalidade causada pela abordagem adotada.

Uma seção foi dedicada a avaliação da latência de pior caso e da vazão em uma rede RTSNoC com quatro roteadores e vinte e quatro núcleos. Os resultados das medições foram confrontados com os valores teóricos esperados para esta rede. O capítulo também apresentou resultados relativos a experimentos realizados com duas redes RTSNoC de tamanhos distintos. O objetivo dos experimentos foi realizar a medida da latência média apresentada por fluxos BE pertencentes a aplicações de tempo real *soft*, além de verificar se algum fluxo ultrapassaria os valores máximos teóricos esperados para cada uma das configurações de rede. Pelos resultados obtidos, a latência média ficou abaixo da latência esperada para uma rede de referência baseada na tecnologia TDM, em redes com um maior número de núcleos conectados. Além disso, todos os limites de latência máxima prevista nas duas redes experimentadas foram atendidos.

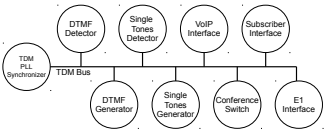
O próximo capítulo descreve a implementação de um SoC desenvolvido para uma aplicação industrial no qual foi utilizada a rede RTSNoC como mecanismo de comunicação entre os seus núcleos.

6 ESTUDO DE CASO

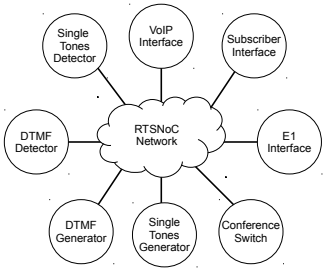
A rede RTSNoC foi avaliada experimentalmente em um SoC de tempo real originalmente desenvolvido para uma aplicação industrial de central telefônica tipo PABX (acrônimo de *Private Automatic Branch eXchange*), o qual foi denominado de SoC PABX. O objetivo foi avaliar o desempenho da rede RTSNoC em um caso de estudo usado na indústria.

A rede RTSNoC foi proposta endereçando sistemas de tempo real com pequenas quantidade de fluxos relacionados a aplicações de tempo real *hard* e muitos fluxos relacionados a aplicações de tempo real *soft*. Um PABX é um exemplo de sistema de tempo real que aceita algum grau de degradação de desempenho. No entanto, esta degradação pode afetar a qualidade de alguns serviços do equipamento, e muitas vezes a percepção de qualidade pode ser um requisito importante para alguns usuários deste tipo de sistema.

Segundo o seu fabricante, o sistema PABX utilizado neste experimento lança sessenta e quatro *threads* em uma operação regular do sistema, sendo que apenas 3 *threads* possuem deadline crítico e as demais podem ser tratadas como BE. A Figura 63 mostra dois diagramas de blocos relacionados ao experimento com o SoC PABX. A Figura 63(a) ilustra a comunicação original entre os núcleos do SoC, baseado em um barramento TDM, e a Figura 63(b) ilustra a comunicação daqueles núcleos sendo realizada através de uma rede RTSNoC.



(a) Diagrama de blocos mostrando a estrutura original do SoC PABX baseado na comunicação TDM entre os seus núcleos.



(b) Diagrama de blocos mostrando o SoC PABX utilizando a RTSNoC como meio de comunicação entre os núcleos.

Figura 63 – Visão geral do SoC PABX.

Este Capítulo explica, em linhas gerais, o funcionamento do SoC PABX e apresenta os resultados obtidos nos experimentos realizados.

6.1 ESTRUTURA DO SOC PABX

O SoC PABX é focado em sistemas de telecomunicações, baseados em tecnologia FPGA, e apresenta algumas funcionalidades necessárias para a implementação de equipamentos PABX digitais: um gerador de 425 H_z, um gerador de DTMF (acrônimo de *Dual-Tone Multi-Frequency*) implementado em um processador *soft core*, uma matriz digital para canais TDM que oferece o serviço de comutação de canais de voz, um detector de sinalização DTMF também implementado em um processador *soft core*, uma interface para enlace padrão E1, uma interface para oferecer acesso aos assinantes do sistema PABX e um processador *soft core* preparado para atuar como controlador geral, gerenciando as chamadas telefônicas em um PABX.

A implementação do SoC PABX é ilustrada na Figura 64. O Anexo 6.3 apresenta alguns detalhes do hardware utilizado na implementação do equipamento PABX de pequeno porte utilizando o SoC PABX, no qual os experimentos foram realizados. Naquela Figura, os quadrados na cor cinza são usados para representar os núcleos que possuem acesso externo ao SoC, enquanto que os círculos brancos representam núcleos sem acesso externo ao SoC. Outros equipamentos, como um gerador de chamadas telefônicas no padrão E1, aparelhos telefônicos e um hardware necessário para realizar a interface com os assinantes do equipamento foram usados mas não são detalhados neste documento.

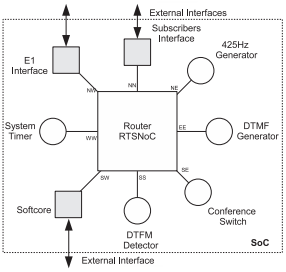


Figura 64 – Diagrama em blocos ilustrando a estrutura interna do SoC PABX implementado com o uso da RTSNoC.

6.2 APLICAÇÃO PABX

De acordo com o seu fabricante, o sistema PABX lança sessenta e quatro *threads* em uma operação regular do sistema. Destas *threads*, apenas três são consideradas de tempo real *hard* e seus *deadlines* não podem ser perdidos. Para um sistema PABX, a perda de *deadlines* não significa uma falha perigosa que pode causar danos ao equipamento, como teoricamente é considerado em sistemas de tempo real *hard*. Apesar disso, de acordo com o fabricante, a perda de *deadlines* daquelas três *threads* implica em uma falha de operação que pode gerar diversas conexões telefônicas previamente estabelecidas, podendo eventualmente causar o travamento do equipamento.

Outras *threads* no sistema PABX aceitam perdas em seus *deadlines*. Tais *threads* estão relacionadas à procedimentos no estabelecimento de chamadas telefônicas, para os quais a percepção humana permite alguns atrasos em diversas etapas no processamento da chamada telefônica. De acordo com o fabricante, se um equipamento PABX está sob alto tráfego no estabelecimento de chamadas telefônicas, a perda de *deadlines* afeta a qualidade geral do sistema, e os usuários do sistema percebem nitidamente tais perda na qualidade, como atraso para ocupação de linha telefônica, falhas na detecção de discagem, entre outros. Nos experimentos realizados com este SoC, estas sessenta e uma *threads* foram definidas como sendo de tempo real *soft*, enquanto que as outras três citadas anteriormente foram definidas como sendo de tempo real *hard* e foram consideradas prioritárias no sistema.

Uma das três *threads* prioritárias é executada no processador responsável pela detecção de sinalização DTMF no sistema e seu *deadline* é de 125 μs. Além desta *thread* prioritária, aquele processador executa outras quatro tarefas de tempo real *soft*. A segunda *thread* prioritária é executada no processador responsável pela geração de sinalização DTMF para todo o sistema e possui *deadline* de 125 μs, além de também possuir outras quatro tarefas de tempo real *soft* em execução. A última das três tarefas prioritárias é executada no processador de controle e possui um *deadline* de 5 ms. As outras cinquenta e três tarefas de tempo real *soft* também são executadas neste último processador.

6.3 ESTRUTURA DE HARDWARE USADA NA VALIDAÇÃO DO SOC PABX

Para a validação física do SoC PABX foi montado um hardware capaz de oferecer uma infra-estrutura básica de um pequeno sistema PABX, ilustrado na Figura 65. Este sistema é composto de uma placa de ramais, responsável por alimentar os aparelhos telefônicos que serão conectados ao PABX e uma placa de troncos analógicos, onde são conectadas as linhas telefônicas fornecidas por empresas Operadoras de Telefonia Fixa em Telecomunicações.

Naquela Figura, a placa de ramais é representada na pela letra B, na cor vermelha, e permite à conexão de até quatro aparelhos telefônicos em seus conectores, representados pela letra C. A placa de troncos analógicos necessita de codec externo para a digitalização dos sinais de voz. Este codec deve ser sintetizado em um FPGA externo e os sinais de controle e áudio devem ser fornecidos através dos conectores na Figura 65 pela letra F. Ao contrário desta placa de Troncos Analógicos, a placa de Ramais Analógicos já possui um codec incorporado e pode ser conectada diretamente a um FPGA através dos sinais disponíveis no conector apresentado naquela Figura com a letra E.

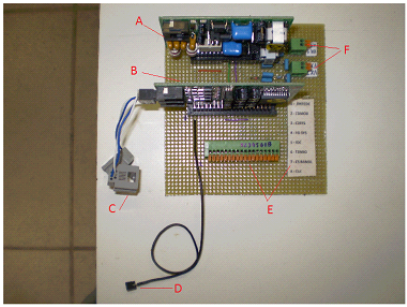


Figura 65 – Imagem do hardware implementado para uso como PABX Digital.

O controle do protótipo PABX é feito por um FPGA externo, no qual foi implementado o SoC TDM. Os sinais do conector apresentado na Figura 65 pela letra E e o sinal de GND (Figura 65 letra D) foram conectados a um kit de FPGA, modelo ML605 do fabricante Xilinx® e que contém um Virtex6®, ilustrado na Figura 66.

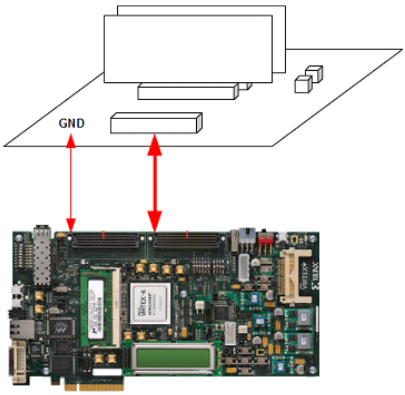


Figura 66 – Representação da conexão do protótipo PABX com o kit ML605.

6.4 OPERAÇÃO DO SOC PABX E RESULTADOS OBTIDOS

O SoC PABX foi sintetizado através da ferramenta do fabricante Xilinx®, versão 13.1, para um dispositivo FPGA da família Virtex 6®. Para analisar o sistema em sua carga de operação máxima, foram geradas trinta chamadas telefônicas no enlace E1, e comutadas com as interfaces de assinantes do sistema PABX, controlado pelo SoC. As chamadas foram geradas aleatoriamente com curta duração de tempo (com até três segundos de duração por chamada) Para cada chamada recebida, os núcleos conectados na rede RTSNoC trocam mensagens relacionadas a identificação das chamadas, geração de tons, detecção de sinalizações e procedimentos de encaminhamento e comutação das chamadas. A rede RTSNoC foi sintetizada para operar com um *clock* de 100 MHz e oferecer vazão de $133,312$ $Mbps$. os núcleos geram uma taxa de $16Mbps$.

Apesar de o tráfego geral neste sistema PABX ser moderado em relação a capacidade total da rede, as mensagens de tratamento de chamadas ocorrem em meio a rajadas de mensagens que são geradas pelos núcleos, e que ocorrem logo após eles terem recebido uma mensagem de sincronização de um núcleo gerador de base de tempo (*System Timer* no canal WW da Figura 64). Portanto, existe uma intensa troca de mensagens de curta duração ocorrendo num curto período de tempo, gerando um *jitter* para os pacotes de sincronização, conforme mostrado na Figura 67. Tomando como base as características de tráfego do sis-

tema PABX, foi calculado a WCL teórica para os pacotes gerados na rede entre o núcleo gerador de base de tempo do sistema e o processador de gerência do sistema, o que resultou num valor de 420 ns como latência de pior caso. As medidas realizadas apontaram que a latência daqueles pacotes variou entre $288 - 317$ ns, o que é menor do que o WCL de 420 ns previsto com a utilização da equação 4.6, apresentada na Seção 4.5.

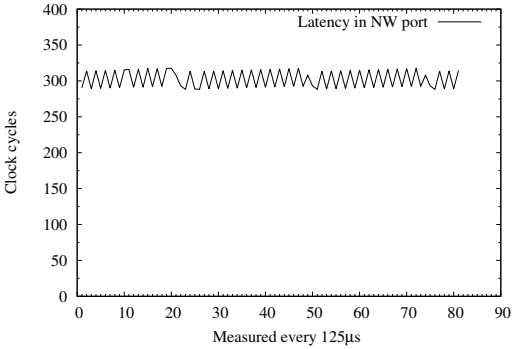


Figura 67 – Valores de latência do pacote de sincronização medidos no canal NW.

6.5 CONSIDERAÇÕES

O objetivo deste capítulo foi a implementação da rede em um SoC de tempo real utilizado em um equipamento de telecomunicações. O SoC utilizado possuía núcleos previamente testados e validados pelo seu fabricante. A comunicação original entre os diversos núcleos do SoC era feita através de conexões ponto-a-ponto entre os núcleos que necessitavam trocar informações. Esta abordagem atendia aos limites de *deadline*, porém a reusabilidade dos núcleos exigia um trabalho de engenharia maior para implementações de novos produtos que necessitassem daqueles núcleos funcionais. A rede RTSNoC foi implementada como mecanismo de comunicação entre tais núcleos, que além de atender os requisitos temporais do sistema, expressos pelos *deadlines* das diversas tarefas em execução no sistema, também ofereceu escalabilidade e reusabilidade aos núcleos do SoC.

PARTE IV

CONCLUSÃO

7 CONSIDERAÇÕES FINAIS E PERSPECTIVAS

Neste documento foram relatadas as atividades de pesquisa relacionadas ao uso de uma rede intra-chip como mecanismo de comunicação em SoCs com características de tempo real. Foi feito um estudo sobre as soluções que foram apresentadas até o momento e que oferecem previsibilidade à sistemas de tempo real que utilizam NoC como solução de interconexão intra-chip, as quais resolvem o problema da previsibilidade de latência para os fluxos na rede, seja utilizando roteadores não-bloqueantes com controle de taxa de transmissão ou chaveamento de circuitos. Porém, o custo de silício para garantir tal previsibilidade a todos os fluxos da rede em ambas as técnicas é considerado alto por diversos autores, o que limita o conhecimento das latências apenas para fluxos prioritários. Devido ao alto custo de recursos, fluxos que não estão relacionados à aplicações de tempo real críticas são tratados em segundo plano, geralmente adotando-se uma política de melhor esforço (BE). Além disso, todas as técnicas citadas precisam mapear os requisitos de comunicação das tarefas de tempo real críticas para os recursos de rede disponíveis em fases iniciais do projeto. Este mapeamento, no entanto, muitas vezes é realizado considerando um cenário de pior caso, resultando em reserva de recursos que poderiam ser dinamicamente realocados para outros fluxos. Embora adequado para aplicações críticas de tempo real, esta estratégia resulta na má utilização de silício para aplicações multimídia com taxa de bits variável.

A hipótese levantada nesta Tese foi afirmar que era possível estabelecer uma estrutura de comunicação para SoCs de tempo real, baseada no conceito de *Network-on-Chip*, sem reserva de recursos, capaz de oferecer previsibilidade na latência de pior caso na comunicação entre dois pontos quaisquer da rede. Além disso, o valor médio da latência para os fluxos tratados como melhor esforço deveria ser menor quando comparado com uma rede com reserva de recursos que também ofereça tratamento de melhor esforço para fluxos sem restrições de tempo real. Como solução ao problema de pesquisa, foi proposta uma rede intra-chip, chamada de RTSNoC, e que é baseada na técnica de intercalação de *flits* em um mesmo canal de comunicação entre roteadores da rede.

Foram realizados experimentos com a rede proposta e feitas comparações analíticas com outra NoC para validar a rede proposta nesta Tese. Pelos resultados obtidos, foi possível comprovar que a RTSNoC oferece a previsibilidade da latência de pior caso para fluxos de tempo real *hard* e apresenta um valor médio de WCL menor para fluxos BE

do que a rede com reserva de recursos Æthereal, para um número maior de núcleos por rede. Tais comparações analíticas foram feitas com base nas equações de latência de pior caso (WCL) de ambas as redes, RTSNoC e Æthereal. Além disso, outras avaliações teóricas apresentadas na Seção 3.2 demonstram que a rede RTSNoC apresenta uma latência média menor que outras redes BE quando o tráfego oferecido é maior do que 70% da capacidade de tráfego da rede.

A Intercalação de *flits* exigiu a adição de informações de roteamento para todos os *flits* que trafegam pela rede RTSNoC. Como consequência, era esperado um aumento do uso de recursos de silício devido a adoção desta técnica. Para resolver este problema, os *buffers* foram alocados apenas nas interfaces de rede, minimizando assim a quantidade de memória nos roteadores. Paralelo a isso, os roteadores da rede permitem mais de uma conexão de unidades de processamento por roteador. Esta abordagem foi adotada para explorar o senso de localidade, minimizando o tráfego entre roteadores. Como esperado, as informações extras de roteamento aumentaram o uso de recursos de silício. No entanto, o consumo é mais relevante quando o tamanho do campo de dados cresce, do que devido a adição de bits extras em cada *flit*. Por exemplo, o aumento no uso de recursos de silício devido ao incremento no número de bits usados para endereçar mais roteadores, e consequentemente mais unidades de processamento conectados na rede, foi em média de 0,3%; enquanto isso o consumo cresce em média até 4,0% quando o tamanho do campo de dados aumenta de 16 bits para 256 bits.

Pelo que foi exposto neste documento, conclui-se que a rede proposta (RTSNoC) valida a hipótese apresentada nesta Tese e apresenta diferentes contribuições. A primeira contribuição foi apresentar uma menor latência média para fluxos BE relacionados a aplicações multimídia com taxa de bits variável. O experimento apresentado na Seção 5.4 foi baseado em um gerador randômico de fluxos de dados que eram injetados na rede RTSNoC. O valor médio de latência encontrado foi comparado com os valores máximos esperados para as redes RTSNoC e Æthereal. Naqueles experimentos, a rede RTSNoC teve um valor médio de latência melhor quando a rede era composta por um número maior de núcleos, assumindo que a rede Æthereal teria sido configurada para oferecer *time slots* suficientes para os fluxos BE.

Uma segunda contribuição está relacionada ao valor de latência de pior caso (WCL). Na rede RTSNoC todos os fluxos possíveis entre dois núcleos quaisquer da rede possuem valores de WCL fixos, e estes valores são independentes de possíveis variações na taxa de bits dos

outros fluxos que competem pelos mesmos recursos da rede. Isto se deve ao algoritmo de arbitragem baseado na intercalação de *flits* em um mesmo canal de comunicação. Desta forma, independente do tamanho dos pacotes que outros fluxos possam gerar e injetar na rede, o valor de WCL é sempre o mesmo, pois o núcleo de origem não tem que aguardar a liberação de recursos da rede para iniciar sua transmissão, visto que todos os *flits* que concorrem por um mesmo canal de comunicação da rede são intercalados. É importante lembrar que a RTSNoC possui *buffers* apenas nas interfaces de rede, e estas memórias devem ser devidamente dimensionadas para que os núcleos tenham sempre a sua disposição espaço de armazenamento para enviar dados pela rede.

Uma terceira contribuição está relacionada ao fato da rede RTSNoC manter o valor médio da latência baixo para fluxos BE, considerando que a rede opera com um tráfego oferecido superior a 80%. Isto acontece porque na RTSNoC os fluxos BE também estão sujeitos aos valores de WCL que podem ser calculados para estes fluxos, pois o algoritmo de arbitragem não faz distinção entre os *flits*; ele leva em consideração apenas a distância dos nodos de origem para realizar a arbitragem de modo igualitário entre todos os fluxos.

Pelo que foi exposto, conclui-se que a rede RTSNoC é uma alternativa de interconexão entre elementos de processamento e que oferece previsibilidade para aplicações de tempo real rígidas, melhorando o valor da latência média para fluxos relacionados à aplicações passíveis de tratamento com técnicas de melhor esforço. Três pontos devem ser considerados pelo projetista antes de decidir pelo uso da rede RTSNoC. O primeiro deles está relacionado ao *deadline* das tarefas em aplicações de tempo real rígido. Neste caso, o valor da latência de pior caso oferecido pela rede para os fluxos relacionados a tais tarefas tem que estar dentro dos limites toleráveis pela aplicação. Outro ponto está relacionado ao tipo de aplicação a que se destina. O método de intercalação de *flits* oferece uma latência menor para pacotes menores. Por este motivo, aplicações como multimídia com taxa variável de bits podem obter melhores resultados de latência média na rede. Por fim, o senso de localidade deve ser explorado para minimizar a comunicação através da rede, colocando em um mesmo roteador, ou em roteadores próximos, os núcleos que trocam com maior frequência mensagens entre si. Para finalizar este Capítulo, é apresentada a Tabela 12, relacionando a RTSNoC com as outras redes estudadas. A contribuição da RTSNoC apresentada naquela tabela está relacionada ao fato de o tempo de pior caso ser independente do tráfego da rede (ver linha WCL $\times$ Tráfego na tabela). Este fato abre perspectivas de trabalhos futuros relacionados a

computação reconfigurável e ao escalonamento de tarefas em sistemas com múltiplos processadores.

7.1 PUBLICAÇÕES

O resultados parciais e finais deste trabalho foram submetidos a três jornais científicos e um pedido de registro de patente. Um dos jornais científicos (IJASCSE) já aceitou e publicou um dos artigos, e o pedido de patente foi depositado e publicado. Os outros dois jornais científicos ainda estão em fase de revisão dos artigos à eles submetidos.

- "Evaluation of silicon consumption for a connectionless Network-on-Chip". Em: International Journal of Advanced Studies in Computer Science & Engineering (IJASCSE), 2014. Disponível em: http://arxiv.org/find/all/1/all:+berejuck/0/1/0/all/0/1
- "Rede Intra-Chip preditiva para aplicações de tempo real". Em: Instituto Nacional de Propriedade Intelectual (INPI), protocolo: BR 10 2013 026770 8, depósito realizado no ano de 2013.

	Network-on-Chip						
	SoCBUS	**Nostrum**	**Æthereal/Aelite**	**MANGO**	**4S Project**	**dAElite**	**RTSNoC**
Ano	2004	2004	2005/2010	2005	2006	2012	2014
Técnica	CC*	TDM	TDM	RNB‡	CC*	TDM	Intercalação de *flits*
WCL	TP§	TP§	TP§	TP§	TP§	TP§	TP§
WCL × Tráfego	Dependente.	Dependente.	Dependente.	Dependente.	Dependente.	Dependente.	Independente.
Wait time							
$(T_{wait;req})$	$4.h + h$	$T - t$	$(P - p).s$	0	0	0	0
$(T_{wait;reply})$	0	$T - t$	$(P - p).s$	0	0	0	0
Latência							
$(T_{latency})$	h	$h.B$	$h.B + f$	$(h + f).(C + B)$	$h.B + b/(L.l)$	$h.B + f$	$\sum_{i=1}^{h}(2N_i) + 2k(f-1) + 2B$
Vazão	$1packet/cycle$	$(n.T)/t$	$(p.n)/(P.s)$	$n.f.C$	$(n.L)/(b.l)$	$(p.n)/(P.s)$	$\frac{1}{2.k}$

Tabela 12 – Relação entre a RTSNoC e as outras redes avaliadas nesta Tese.

Legenda:

∗ CC - Chaveamento de Circuitos.
‡ RNB - Roteador sem bloqueio.
§ TP - Definido durante o projeto da rede.

h - número de saltos (*hops*) entre roteadores em um caminho na rede.
B - tamanho do *buffer* nos roteadores.
n - número de pacotes em uma transação.
f - número de *flits* em um pacote.
b - número de bits em um pacote.
l - pistas (*lanes*) alocadas para um circuito virtual.
L - tamanho das *lanes* em bits.
t - *containers* alocados para um canal virtual.
T - tamanho máximo de um caminho, ida e volta (*loop*).
p - *time slots* alocados para um circuito virtual.
P - período dos *time slots* em um escalonamento.
s - duração de um *time slot* em ciclos de *clock*.
C - número de canais virtuais em um enlace.

8 TRABALHOS FUTUROS

Durante o desenvolvimento desta Tese foram observadas algumas questões que abrem possibilidades de novas pesquisas envolvendo a rede proposta. Uma delas está relacionada a ferramentas para alocação de núcleos na rede. A troca de dados em uma rede entre dois pontos tem influência na latência na rede. Esta latência tem influência na posição relativa dos núcleos na rede. Este passo é conhecido como alocação na rede (ou *Network Assignment*), e normalmente é feita estaticamente, ou em outras palavras, em tempo de projeto. Assim, o desenvolvimento de ferramentas para auxiliar a alocação do núcleos na rede é uma questão a ser explorada.

Para tratar de sistemas com uma dinâmica maior no comportamento dos fluxos da rede, especialmente redes conectando múltiplos processadores, um tema a ser pesquisado é a análise em tempo de execução e alteração de rotas, feitas dinamicamente no sistema. Para isso seria necessário implementar um mecanismo que avalie as condições dos caminhos de roteamento e a alterem o caminho no caso de uma latência superior a certos limites pré-estabelecidos. Isto implicaria na adoção de algoritmos de roteamento adaptativos, que seriam realimentados com as informações de tráfego da rede.

A análise dinâmica de fluxos na rede abre oportunidades de pesquisa relacionadas a computação reconfigurável. O uso de computação reconfigurável seria uma alternativa interessante a ser pesquisada, com a troca de algoritmos de roteamento sendo feitas através de reconfigurações parciais do dispositivo FPGA. Neste ponto, o uso de componentes tanto em software quanto em hardware são fundamentais. O tempo de reconfiguração de partes de um FPGA podem ser superiores aos limites aceitáveis pelo sistema. Deste como, uma metodologia poderia ser a utilização de componentes híbridos para resolver este problema, como a ADESD (acrônimo de *Application-Driven Embedded System Design* (POLPETA; FRöHLICH, 2004). O uso da ADESD vem se mostrando uma boa opção para guiar o desenvolvimento de sistemas embarcados e seus componentes híbridos são passíveis de reutilização e reconfiguração dinâmica.

Um trabalho de investigação científica relacionado a computação reconfigurável está em andamento, no qual a rede RTSNoC tem os seus roteadores e unidades de processamento reconfigurados em tempo de execução. Naquele trabalho, a gestão da reconfiguração do FPGA é feita pelo sistema EPOS. O Projeto EPOS (acrônimo de *Embedded*

Parallel Operating System) visa automatizar o desenvolvimento de sistemas embarcados para que os desenvolvedores possam se concentrar apenas no desenvolvimento dos aplicativos. O EPOS é baseado na metodologia ADESD para orientar o desenvolvimento de ambos os componentes de software e hardware que podem ser automaticamente adaptados para satisfazer os requisitos de aplicações específicas. O EPOS oferece um conjunto de ferramentas para apoiar os desenvolvedores na seleção, configuração e na conexão de componentes em seu *framework*. A combinação de metodologia, componentes, estruturas e ferramentas permitem a geração automática de instâncias e um sistema embarcado de aplicação específica.

Outro trabalho de investigação em Computação Reconfigurável está associado ao fato de a latência de pior caso entre dois pontos quaisquer da rede ser sempre o mesmo. É possível pensar em reconfigurar o conjunto de tarefas de alguns processadores conectados na rede RTSNoC tendo a garantia de que os processadores que não foram reconfigurados manterão seus WCL atendidos. Isso significa dizer que a análise de escalonabilidade de tarefas pode ser feita por processador e não para o conjunto de tarefas em todos os processadores.

REFERÊNCIAS

ATIENZA, D., ANGLIOLINI, F., MURALI, S., PULLINI, A., BENINI, L., DE MICHELI, G. Network-on-Chip design and synthesis outlook. *VLSI journal*, Elsevier, v. 41, pp. 340-359, 2008.

ADRIAHANTENAINA, A. e GREINER, A. Micro-network for SoC: implementation of a 32-port SPIN network. *Design, Automation and Test in Europe Conference and Exhibition*, p. 1128-1129, 2003.

BANERJEE, A., WOLKOTTE, P.T., MULLINS, R.D., MOORE, S.W. e SMIT, G. An Energy and Performance Exploration of Network-on-Chip Architectures. *Transactions on Very Large Scale Integration (VLSI) Systems*, p. 319-329, 2009.

BENINI, Luca; DE MICHELI, Giovani. Networks on Chip. 1.ed. [S.l.]: Morgan Kaufmann, 2006.

BEREJUCK; ZEFERINO. Adding Mechanisms for QoS to a Network-on-chip. In: Proceedings of the 22Nd Annual Symposium on Integrated Circuits and System Design: Chip on the Dunes, pp.25:1-25:6, 2009.

BERTOZZI et a. Xpipes: a latency insensitive parameterized network-on-chip architecture for multiprocessor SoCs. Computer Design, 2003. Proceedings. 21st International Conference on, p. 536-539, 2003.

BJERREGAARD e SPARSO. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. Proceedings, Design, Automation and Test in Europe, v. 2, p. 1226-1231, 2005.

BJERREGAARD e SPARSO. Scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip. Proceedings. 11th IEEE International Symposium on Asynchronous Circuits and Systems, p. 34-43, 2005.

BJERREGAARD; MAHADEVAN; OLSEN; SPARSOE. An OCP Compliant Network Adapter for GALS-based SoC Design Using the MANGO Network-on-Chip. In: Proceedings: International Symposium on System-on-Chip, pp. 171-174, 2005.

BJERREGAARD e SPARSO. Implementation of guaranteed services in the MANGO clockless network-on-chip. Proceedings. Computers and Digital Techniques, v. 153, p. 217-229, 2006.

BJERREGAARD; SPARSO. Packetizing OCP Transactions in the MANGO Network-on-Chip. In: 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools-DSD, pp. 657-664, 2006.

BJERREGAARD; STENSGAARD; SPARSOE. A Scalable, Timing-Safe, Network-on-Chip Architecture with an Integrated Clock Distribution Method. In: Design, Automation Test in Europe Conference Exhibition, DATE '07, pp. 1-6, 2007.

BODEN, N. et al. Myrinet: a gigabit-per-second local area network. Micro, IEEE, v. 15, n. 1, p. 29–36, 1995. ISSN 0272-1732.

BOLOTIN EVGENY; CIDON, I. G. R. K. A. Qnoc: Qos architecture and design process for network on chip. Journal of Systems Architecture, v. 50, n. 2:3, p. 105 – 128, 2004.

DAVIS, ROBERT I. and BURNS, ALAN. A Survey of Hard Real-Time Scheduling for Multiprocessor Systems. ACM Comput. Surv., v.43, n.4, p. 1–44, 2011.

BROWN, S.; VRANESIC, Z. Fundamentals of digital logic with VHDL design. [S.l.]: MacGraw-Hill, 2005.

BUTTAZZO, Giorgio. Hard Real-Time Computing Systems: Predictable Scheduling Algorithms an Applications. 2. ed. [S.l.]: Springer, 2005.

CIDON, I. Zooming in on network-on-chip architectures. Proceedings of the 16th international conference on Structural Information and Communication Complexity. Berlin, Heidelberg: Springer-Verlag, 2010. (SIROCCO'09), p. 1–1. ISBN 3-642-11475-X, 978-3-642-11475-5. $< http://dx.doi.org/10.1007/978-3-642-11476-2_1 >$.

DALLY, W.; TOWLES, B. Route packets, not wires: on-chip interconnection networks. Design Automation Conference, 2001. Proceedings. [S.l.: s.n.], 2001. p. 684–689. ISSN 0738-100X.

DAS et al. A low latency router supporting adaptivity for on-chip interconnects. Design Automation Conference, 2005.

DALLY, William J. e SEITZ, Charlles L. *The Torus Routing Chip.* Journal of Distributed Computing, 1986.

DEHON, A. Nanowire-based programmable architectures. J. Emerg. Technol. Comput. Syst., ACM, New York, NY, USA, v. 1, n. 2, p. 109–162, jul. 2005. ISSN 1550-4832. <http://doi.acm.org/10.1145/1084748.1084750>.

DALLY WILLIAM. J.; TOWLES, B. Principles and practices of interconnection networks 1.ed. [S.l.]: Morgan Kaufmann, 2001.

DUATO, J.; YALAMANCHILI, S.; NI, L. Interconnection Networks. [S.l.]: Morgan-Kaufmann, 2002.

FENQXIANG; BURNS. Schedulability Analysis for Real-Time Systems with EDF Scheduling. Computers, IEEE Transactions, v.58,n. 9, pp. 1250-1258, 2009.

FOROUZAN, B. A.; MOSHARRAF, F. Data Communications and Networking. [S.l.]: Bookman, 2012.

GOOSSENS, DIELISSEN e RADULESCU. AEthereal network on chip: concepts, architectures, and implementations. Design Test of Computers, IEEE, v. 22, n. 5, p. 414-421, 2005.

GOSSENS; BENNEBROEK; HUR; WAHLAH. Hardwired Networks on Chip in FPGAs to Unify Functional and Configuration Interconnects. In: Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip, pp. 45-54, 2008.

GOOSSENS; MHAMDI; SENIN. Internet-Router Buffered Crossbars Based on Networks on Chip. In: 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools, DSD, pp 365-374, 2009.

GOOSSENS, K. e STEFAN, A. The aethereal network on chip after ten years: goals, evolution, lessons, and future. Proceedings of the 47th Design Automation Conference (DAC), p. 306-311, 2010.

GUPTA, Rajesh K.; ZORIAN, Yervant. Introducing core-based system design. IEEE Design and Test of Computers, v. 14, n. 4, p.15-25, Oct.-Dec. 1997.

GARRET, J., WITTEN, D., HASTIE, T., TIBSHIRANI, R. An Introduction to Statistical Learning: with Applications in R. 1. ed. [S.l.]: Springer, 2013.

HANSSON; GOOSSENS. Trade-offs in the Configuration of a Network on Chip for Multiple Use-Cases. In: First International Symposium on Networks-on-Chip, NOCS, pp. 233-242, 2007.

HANSSON; GOOSENS. A Quantitative Evaluation of a Network on Chip Design Flow for Multi-core Consumer Multimedia Applications. In: Journal of Des. Autom. Embedded Syst.,v. 15(2), pp. 159-190,2011.

HANSSON; GOOSSENS; RADULESCU. A Unified Approach to Mapping and Routing on a Network-on-Chip for Both Best-Effort and Guaranteed Service Traffic. In: VLSI Design, v. 2007, 16 pp., 2007.

HAUCK SCOTT.; DEHON, A. Reconfigurable Computing: the theory and practice of FPGA-based computing. [S.l.]: Morgan Kaufmann, 2008.

HANSSON et al. Aelite: A flit-synchronous network on chip with composable and predictable services. Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE), 2009.

HANSSON e GOOSSENS. On-Chip Interconnect with aelite: Composable and Predictable Systems. Embedded Systems Series. Springer, 2010.

HELMY, H., PIERRE, L., JANTSCH, A. Theorem proving techniques for the formal verification of NoC communications with non-minimal adaptive routing. In: IEEE 13th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), pp. 221-224, 2010.

JAN, M.; RABAEY ANANTHA, P.; CHANDRAKASAN, B.N. Digital Integrated Circuits.. [S.l.]: Prentice-Hall, 2002.

JANTSCH, A. Models of Computation for Networks on Chip. In: Sixth International Conference on Application of Concurrency to System Design, ACSD 2006, pp 165-178, 2006.

KARIN, NGUYEN e DEY. Network on a Chip: an architecture for billion transistor era. Proceedings: NORCHIP, 2002.

KARIN, NGUYEN e DEY. An interconnect architecture for networking systems on chips. Micro, IEEE, p. 36-45, 2002.

KAVALDJIEV, SMIT, JANSEN e WOLKOTTE. A virtual channel network-on-chip for GT and BE traffic. Emerging VLSI Technologies and Architectures. IEEE Computer Society Annual Symposium on, 6 pp., 2006.

KIM, B. et al. A real-time communication method for wormhole switching networks. Parallel Processing, 1998. Proceedings. 1998 International Conference on. [S.l.: s.n.], 1998. p. 527–534. ISSN 0190-3918.

KOPETZ. Real-Time Systems: Design Principles for Distributed Embedded Applications. [S.2.]: Springer, 2011.

KUON, I.; ROSE, J. Measuring the gap between fpgas and asics. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, v. 26, n. 2, p. 203–215, 2007. ISSN 0278-0070.

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

Lucas e Moraes. Crosstalk fault tolerant NoC - Design and evaluation. Very Large Scale Integration (VLSI-SoC), 2009 17th IFIP International Conference on, p. 115-120, 2009.

MCDYSAN, D. E.; SPOHN, D. ATM: Theory and Application. [S.l.]: McGraw-Hill, 1994.

MARCULESCU e JINGCAO. SoCIN: a parametric and scalable network-on-chip. SBCCI 2003. Proceedings. 16th Symposium on, p. 169-174, 2003.

MARK, D.; FAN, J. Localizing open interconnect defects using targeted routing in fpga's. Test Conference, 2004. Proceedings. ITC 2004. International. [S.l.: s.n.], 2004. p. 627–634.

MELLO. Qualidade de Serviço em Rede Intra-chip. Implementação e avaliação sobre a Rede Hermes. Dissertação (Mestrado), Pontifícia Universidade Católica do Rio Grande do Sul. Programa de Pós Graduação em Computação, Porto Alegre, 129 p., 2006.

MILLBERG, NILSSON, THID e JANTSCH. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip. *Proceedings. Design, Automation and Test in Europe Conference and Exhibition*, p. 890-895, 2004.

MILLBERG, NILSSON, THID, KUMAR e JANTSCH. The Nostrum backbone-a communication protocol stack for Networks on Chip. Proceedings. 17th International Conference on VLSI Design, p. 693-696, 2004.

MILLBERG, M., JANTSCH, A. Priority based forced requeue to reduce worst-case latencies for bursty traffic. In: Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09, pp. 1070-1075, 2009.

MLA, T.-C. p. Survey of time-predictable NOCs and their WCET analysis. 2012. <http://www.t-crest.org/page/results>. Acessado em 20/01/2013.

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

NAEEM, A., JANTSCH, A., ZHONGHAI LU. Scalability Analysis of Memory Consistency Models in NoC-Based Distributed Shared Memory SoCs. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, (32), n.5, pp. 760-773, 2013.

NAVABI et al. Evaluation of pseudo adaptive XY routing using an object oriented model for NOC. Microelectronics, ICM, 2005.

NI, L. Issues in designing truly scalable interconnection networks. Parallel Processing, 1996. Proceedings of the 1996 ICPP Workshop on Challenges for. [S.l.: s.n.], 1996. p. 74–83. ISSN 1530-2016.

NI, L.; MCKINLEY, P. A survey of wormhole routing techniques in direct networks. Computer, v. 26, n. 2, p. 62–76, 1993. ISSN 0018-9162.

NOLLET, V. et al. Operating-system controlled network on chip. Proceedings of the 41st annual Design Automation Conference. New York, NY, USA: ACM, 2004. (DAC '04), p. 256–259. ISBN 1-58113-828-8. <http://doi.acm.org/10.1145/996566.996637>.

PENOLAZZI; JANTSCH. A High Level Power Model for the Nostrum NoC. In: 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools, DSD, pp. 673-676, 2006.

PETERSON LARRY L.; DAVIE, B. S. Computer networks: a system approach. 4.ed. [S.l.]: Morgan Kaufmann, 2007.

Palopoli, L. and Abeni, L. and Buttazzo, G. and Conticelli, F. and Di Natale, M., Real-time control system analysis: an integrated approach Real-Time Systems Symposium, 2000. Proceedings. The 21st IEEE, pp. 131-140, 2000.

POLPETA, F. V.; FRöHLICH, A. A. Hardware mediators: A portability artifact for component-based systems. EUC. [S.l.: s.n.], 2004. v. 3207, p. 271–280.

RADU, MOLNOS, E GOOSSENS, K. dAElite: A TDM NoC Supporting QoS, Multicast, and Fast Connection Set-up. IEEE Transactions on Computers, v. 99, s.n., 2012.

RADULESCU et al. An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, v. 24, n. 1, pp. 4-17, 2005.

RIJPKEMA et al. Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip. In: IEE Proceedings - Computers and Digital Techniques, v. 150(5), pp. 294-302, 2003.

ROYAL, CHEUNG. Globally Asynchronous Locally Synchronous FPGA Architectures. Springer Berlin Heidelberg, v.2778, p.355-364, 2003

SATHE, WIKLUND, e LIU. Design of a guaranteed throughput router for on-chip networks. International Symposium on System-on-Chip, p. 25-28, 2004.

STENSGAARD; BJERREGAARD, SPARSOE; PEDERSEN. A Simple Clockless Network-on-Chip for a Commercial Audio DSP Chip. In: 9th EUROMICRO Conference on Architectures, Methods and Tools, DSD, pp. 641-648, 2006.

SHI, Z.; BURNS, A. Real-time communication analysis for on-chip networks with wormhole switching. In:
hskip 1em plus 0.5em minus 0.4emProceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip. Washington, DC, USA: IEEE Computer Society, 2008. (NOCS '08), p. 161–170. ISBN 978-0-7695-3098-7. <http://dl.acm.org/citation.cfm?id=1397757.1397996>.

SMIT et al. Multi-core Architectures and Streaming Applications. In: Proceedings of the International Workshop on System Level Interconnect Prediction, pp. 35-42, 2008.

SMITH, M. J. S. Application-Specific Integrated Circuits. [S.l.]: Addison-Wesley, 1997.

SODERQUIST, I. Globally updated mesochronous design style (gum amp;8211;design amp;8211;style). In:
hskip 1em plus 0.5em minus 0.4emSolid-State Circuits Conference, 2002. ESSCIRC 2002. Proceedings of the 28th European. [S.l.: s.n.], 2002. p. 603–606.

STEFAN e GOOSSENS. A TDM slot allocation flow based on multipath routing in NoCs. In
hskip 1em plus 0.5em minus 0.4emMicroprocess. Microsyst., v. 35, n. 2, p. 130-138, 2011.

SHELKE e PATIL. Low-Latency, Low-Area Overhead and High Throughput NoC Architecture for FPGA Based Computing System. In: International Conference on Electronic Systems, Signal Processing and Computing Technologies (ICESC), pp. 53-57, 2014.

TAMIR e FRAZIER. Dynamically-allocated multi-queue buffers for vlsi communication switches. In: Computers, IEEE Transactions on, v.41, n.6, p. 725-737, 1992.

UG364, Xilinx. Virtex 6 Configurable Logic Block, 2010. Disponível em *www.xilinx.com/support/documentation/user_guides/ug*360.*pdf*. Acessado em 22/05/2012.

UG872, Xilinx. Large FPGA methodology guide, 2012. Disponível em *www.xilinx.com/support/documentation/sw_manuals/xilinx*13_4/*ug*872_*largefpga.pdf*. Acessado em 02/03/2014.

VITKOVSKI, A., JANTSCH, A., LAUTER, R., HAUKILAHTI, R., NILSSON, E. Low-power and error protection coding for network-on-chip traffic. In: Computers Digital Techniques, IET, (2), n. 6, pp. 483-492, 2008.

YE, T. T.; BENINI, L.; MICHELI, G. D. Packetization and Routing Analysis of On-Chip Multiprocessor Networks. Journal of Systems Architecture, 2003.

WIKLUND, LIU. SoCBUS: switched network on chip for hard real time embedded systems. International Parallel and Distributed Processing Symposium, 8 pp., 2003.

WIKLUND, SATHE, LIU. System-on-Chip for Real-Time Applications. Proceedings. 4th IEEE International Workshop on, pp. 269-274, 2004.

WIKLUND, EHLIAR, LIU. Design of an Internet core router using the SoCBUS network on chip. International Symposium on Signals, Circuits and Systems, pp. 513-516, (2), 2005.

WOLKOTTE, RAUWERDA e SMIT. An Energy-Efficient Reconfigurable Circuit-Switched Network-on-Chip. Proceedings. 19th IEEE International Parallel and Distributed Processing Symposium., p. 155a-155a, 2005.

WOLKOTTE, KAVALDJIEV, BECKER e BECKER. Energy Model of Networks-on-Chip and a Bus. In: Proceedings. International Symposium on System-on-Chip, p. 82-85, 2005.

WOLKOTE, HÖLZENSPIES; SMIT. Fast, Accurate and Detailed NoC Simulations. In: First International Symposium on Networks-on-Chip., pp. 323-332, 2007.

VAN DEN BRAND, J. W., CIORDAS, C., GOOSSENS, K. e BASTEN, T. Congestion-Controlled Best-Effort Communication for Networks-on-Chip. Design, Automation Test in Europe Conference Exhibition, pp. 1 - 6, 2007.

XIAOWEN CHEN, ZHONGHAI LU, JANTSCH A., SHUMING CHEN. Run-Time Partitioning of Hybrid Distributed Shared Memory on Multi-core Network-on-Chips. In: Third International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), pp. 39-46, 2010.

ZEFERINO e SUSIN. SoCIN: a parametric and scalable network-on-chip. Proceedings. 16th Symposium on Integrated Circuits and Systems Design, SBCCI, p. 169-174, 2003.

ZIMMER e JANTSCH. A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip. Hardware/Software Codesign and System Synthesis, 2003. First IEEE/ACM/IFIP International Conference on, p. 188-193, 2003.

APÊNDICE A – Emenda

O sexto membro da Banca Examinadora desta Tese foi o Professor Dr. Carlos Barros Montez. Assim, onde estiver escrito nesta Tese "Prof. Eduardo Barros Montez, Dr.", leia-se: "Prof. Carlos Barros Montez, Dr.".