UNIVERSIDADE FEDERAL DE SANTA CATARINA

Gabriel Ganzer

MONITORAMENTO DA ENTREGA DE PACOTES EM REDES INTRA-CHIP

Araranguá

2018

Gabriel Ganzer

MONITORAMENTO DA ENTREGA DE PACOTES EM REDES INTRA-CHIP

Trabalho de Conclusão de Curso submetido ao Bacharelado em Engenharia de Computação para a obtenção do grau de Bacharel em Engenharia de Computação.
Orientador: Prof. Marcelo Daniel Berejuck, Dr.

Araranguá

2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Ganzer, Gabriel
Monitoramento da entrega de pacotes em redes intra-chip / Gabriel Ganzer ; orientador, Marcelo Daniel Berejuck, 2018.
94 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Engenharia de Computação, Araranguá, 2018.

Inclui referências.

1. Engenharia de Computação. 2. redes intra-chip. 3. sistemas intra-chip. 4. observabilidade computacional. I. Berejuck, Marcelo Daniel. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. III. Título.

Gabriel Ganzer

MONITORAMENTO DA ENTREGA DE PACOTES EM REDES INTRA-CHIP

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do título de "Bacharel em Engenharia de Computação", e aprovado em sua forma final pelo Bacharelado em Engenharia de Computação.

Araranguá, 3 de julho 2018.

Profa. Eliane Pozzebon, Dra.
Coordenadora

Banca Examinadora:

Marcelo Daniel Berejuck
Assinado de forma digital por Marcelo Daniel Berejuck
Dados: 2018.07.06 14:24:48 -03'00'

Prof. Marcelo Daniel Berejuck, Dr.
Universidade Federal de Santa Catarina

Prof. Tiago Oliveira Weber, Dr.
Universidade Federal de Santa Catarina

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

Este trabalho é dedicado aos meus pais, João e Libera.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer aos meus pais João Ganzer e Libera Ganzer e à minha irmã Gracieli Ganzer, por todo o suporte oferecido em toda a minha trajetória. Serei eternamente grato por terem acreditado no meu potencial e pelos ensinamentos de meus pais, que sempre priorizaram a minha educação e a de minha irmã.

A Universidade Federal de Santa Catarina pela oportunidade de realizar este curso, em especial seu corpo docente, por terem expandido os meus horizontes e oferecido os conhecimentos necessários à minha formação.

Ao meu orientador, o Prof. Dr. Marcelo Daniel Berejuck, pela confiança e empenho dedicado ao desenvolvimento deste trabalho, pela oportunidade de integrar a equipe do seu laboratório de pesquisa, por acreditar nas minhas habilidades para a investigação científica e pelo auxílio oferecido para que eu pudesse alcançar meus objetivos.

A Biblioteca Setorial de Araranguá e sua equipe, por toda a assistência oferecida durante o desenvolvimento deste trabalho, em especial às bibliotecárias Thayse Hingst e Débora Maria Russiano Pereira.

Para finalizar, gostaria de agradecer à minha querida colega de turma e amiga Joice Preuss Cardoso, ao longo de nossas jornadas oferecemos suporte um ao outro e sou grato pela oportunidade que este curso nos deu de desenvolver uma amizade tão forte.

RESUMO

Sistemas intra-chip ou SoC (acrônimo de *System-on-Chip*) com múltiplas unidade de processamento heterogêneas têm sido usados pela indústria de silício como solução para disponibilizar o desempenho demandado pelas aplicações modernas. Entretanto, a integração de um grande número de tais unidades de processamento impõem um desafio aos mecanismos de interconexão. Para contornar esta situação, a indústria vem utilizando redes intra-chip ou NoC (acrônimo de *Networks-on-Chip*), as quais utilizam de roteadores e conceitos provenientes das redes computacionais convencionais para interconectar as unidades de processamento de um SoC. De forma geral, os fluxos dessas redes são heterogêneos, divididos basicamente em dois tipos: aqueles de tempo real, provenientes de tarefas que demandam garantia da entrega dos pacotes, qualidade de serviço e tempo de execução rígido; os de melhor esforço, provenientes de tarefas que exigem tempo de execução menos rígido. A grande maioria das redes intra-chip publicadas até o momento visam prover uma latência mínima e previsível para os fluxos de tempo real, como é o caso da rede RTSNoC (acrônimo de *Real-Time Network-on-Chip*), uma rede que oferece garantia de vazão às aplicações de tempo real através da previsibilidade da latência de pior caso. A RTSNoC não provê qualidade de serviço quando utilizada em sistemas com uma quantidade maior de fluxos de melhor esforço e apenas melhora a latência média de tais fluxos. O aperfeiçoamento do desempenho desta rede em aplicações onde a maioria dos fluxos são de melhor esforço pode ser atingida pelo uso de métodos otimizadores, algoritmos de roteamento adaptativos ou da computação reconfigurável atuando na redistribuição das unidades de processamento interconectadas aos roteadores. Tais métodos necessitam, inicialmente, serem alimentados constantemente com informações do tráfego da rede, obtidas por um sistema apto a monitorar a entrega dos pacotes, para só então operacionalizar alterações na rede em tempo de execução. Dito isso, o presente trabalho objetiva introduzir ao projeto da RTSNoC o conceito de observabilidade computacional através do desenvolvimento de um sistema de monitoramento da entrega dos pacotes para aquela rede.

Palavras-chave: Sistemas intra-chip. Redes intra-chip. Observabilidade computacional.

ABSTRACT

Systems with multiple heterogeneous processing units, also known as Systems-on-Chip (SoC), have been used by the silicon industry as a solution to deliver the high performance demanded by modern applications. However, the integration of a large number of such processing units poses a challenge to the interconnection mechanisms. The industry has been using Networks-on-Chip (NoC) to solve this problem, an approach that uses routers and concepts from conventional computational networks to interconnect the processing units of a SoC. In general, a flow of these networks can be divided into two types: those of real-time coming from tasks that demand assurance of package delivery, quality of service, and hard execution time; best-effort flows that come from soft tasks, which does not demand such requirements. So far, many of the published NoCs aim to provide a minimum and predictable latency for real-time flows, such as the RTSNoC, a network-on-chip that gives throughput guarantee to real-time systems by using the technique of worst-case latency predictability. The RTSNoC does not offer a quality of service when used in applications with a higher amount of best-effort flows and only improves the average latency of those flows. To achieve a level of performance improvement on applications that have a best-effort flows dominance that NoC could use optimization methods, adaptive routing algorithms or reconfigurable computing by redistributing the interconnected processing units at runtime. Such methods need to be continuously fed with information about the network traffic obtained by a system able to monitor the packet delivery. Within that said, the present work aims to introduce to the RTSNoC project the concept of computational observability by developing a system who would be able to monitor and get information about the delivery of packages on that network.

Keywords: System-on-Chip. Netwok-on-Chip. Computational observability.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SoC	*System-on-Chip*
NoC	*Network-on-Chip*
BE	*Best Effort*
QoS	*Quality of Service*
WCL	*Worst Case Latency*
2D	Duas dimensões.
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*
FPGA	*Field Programmable Gate Array*
NI	*Network Interface*
S	*Sniffer*
NM	*Network Manager*
TDM	*Time-Division Multiplexing*
GS	*Guaranteed Services*
IP	*Intellectual Property*
ACK	*Acknowledge*
NACK	*Acknowledge Negatively*
OCP	*Open Core Protocol*
DMA	*Direct Memory Access*
TRA	*Target Routing Address*
FP	*Full Packet*
EP	*End of Packet*
BDY	*Body*
CRT	*Current Routing Table*
NBS	*Next Buffer State*
CS	*Circuit Switching*
FP	*Fixed Priority*
DP	*Dynamic Priority*
RB	*Rate Based*
GT	*Guaranteed Throughput*
VCI	*Virtual Channel Interface*
RI	*Routing Information*
HLP	*Higher Level Protocol*
FIFO	*First In First Out*
RTL	*Register Transfer Level*
NI	*Network Interface*
OSI	*Open Systems Interconnection*
VCD	*Value Change Dump*
CSV	*Comma Separated Values*

SUMÁRIO

1 INTRODUÇÃO

Os avanços nas tecnologias para fabricação de chips têm permitido que um grande número de transistores sejam integrados em uma única pastilha de silício. Como consequência, dispositivos com sistemas mais complexos e robustos passaram a surgir, já que esses avanços permitiram aos projetistas integrarem em um único chip múltiplos núcleos com finalidades diversas, um conceito denominado na literatura como *System-on-Chip* (SoC).

Segundo Florez (2011), devido sua característica de ortogonalidade, um SoC pode ser dividido em duas estruturas: computação e comunicação. A estrutura de computação trata dos módulos responsáveis por processar e armazenar as informações do sistema. Já a estrutura de comunicação trata de interligar os núcleos do sistema e promover a troca de informações de forma coordenada. Em seu trabalho intitulado Redes-em-Chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho, Zeferino (2003) afirma que as abordagens de comunicação geralmente utilizadas nos sistemas antes do surgimento dos SoC's, como a interconexão dos núcleos através de canais ponto-a-ponto dedicados ou canais multipontos compartilhados não atendem os requisitos de desempenho e consumo de energia desse novo tipo de sistema. Além disso, foi previsto que esses sistemas poderiam tornar-se tão grandes e complexos que inviabilizariam seus usos.

Como alternativa aos problemas aqui descritos, os trabalhos científicos recentes propõem o uso de redes de interconexão chaveadas aplicadas à comunicação intra-chip, também chamadas de *Networks-on-Chip* (NoC). A terminologia foi proposta por Hemani et al. (2000) e em português são utilizados os termos Redes-em-Chip e Redes Intra-chip, ou simplesmente NoC. Uma NoC pode ser definida como um arranjo de canais ponto-a-ponto chaveados por roteadores e compartilhados pelos núcleos do sistema, onde os dados são transmitidos por meio de pacotes divididos em unidades atômicas chamadas *flits (Flow Control Unit)*, (COTA; AMORY; LUBASZEWSKI, 2011). Além do paralelismo nas comunicações, as NoC's proporcionam propriedades elétricas otimizadas e controladas ao permitirem a distribuição de sinais de relógio, também conhecidos como sinais de *clock*, de forma uniforme aos sistemas mais complexos.

Sistemas embarcados baseados em SoC vêm sendo cada vez mais usados em aplicações com requisitos temporais, denominadas de tempo real. Entre os sistemas computacionais mais simples, pode-se citar os controladores inteligentes embarcados em equipamentos de utilidade doméstica, como o forno de micro-ondas; entre os mais complexos, pode-se usar como exemplo os sistemas de controle utilizados em aeronaves (BEREJUCK, 2015).

Algumas aplicações de tempo real apresentam restrições de tempo mais rigorosas do que outras, impostas pelo ambiente no qual o sistema computacional está inserido. Ou seja, os sistemas computacionais usados em aplicações de tempo real interagem enviando continuamente respostas aos estímulos de entrada vindos dos seus ambientes. Um sistema de tempo real pode ser definido, portanto, como um sistema computacional que deve reagir a tais estímulos dentro de prazos conhecidos na literatura como *deadlines* (BODEMÜLLER, 2014).

Nesse contexto um sistema de tempo real pode, ainda, ser modelado como tendo uma quantidade de tarefas com tempo de execução menos rígidos, ditas como [1]*soft* que resultam num fluxo de melhor esforço (do inglês *Best Effort* — BE), em que não existe garantia de entrega dos pacotes e qualidade de serviço (QoS) além das tarefas que demandam por um tempo de execução rígido, ou [2]*hard*, que consequentemente resultam num fluxo com latência de pior caso (do inglês Worst Case Latency — WCL) que exige garantia de entrega dos dados e QoS. Assim, os requisitos de projeto tornam-se mais críticos e desenvolver estratégias de comunicação que reduzam a latência e garantam os requisitos de resposta em tempo real caracteriza uma etapa de suma importância para o projetista.

1.1 PROBLEMA

Atualmente, a grande maioria dos trabalhos científicos apresentam propostas que consideram o tráfego da rede previamente ao desenvolvimento do projeto para garantir as restrições temporais dos fluxos de tempo real. Assim, para que as restrições temporais das tarefas de tempo real crítico sejam garantidas é necessário que o SoC seja dimensionado pela sua maior restrição, o que acaba por aumentar o custo de silício, pois muitas vezes as restrições temporais críticas ocorrem esporadicamente na comunicação do SoC. Consequentemente, o fluxo se torna superestimado devido o planejamento do consumo dos recursos de silício ser baseado na previsibilidade da latência de uma rede.

Em *Evaluation of a Connectionless Technique for System-on-Chip Interconnection* os autores Berejuck e Fröhlich (2016) apresentam a rede [3]RTSNoC, uma rede de topologia em malha ortogonal 2D desenvolvida em VHDL e validada num dispositivo FPGA com previsão de latência e projetada para sistemas embarcados de tempo real. Essa rede garante QoS num fluxo WCL e melhora a latência média para fluxos BE sem comprometer

[1]Uma tarefa é dita como sendo *soft* se a perda do seu *deadline* causa apenas degradação do desempenho do sistema, sem impossibilitar o seu funcionamento normal. Para este tipo de tarefa, o seu resultado pode ou não ser útil após o seu *deadline* (BODEMÜLLER, 2014).

[2]Ao contrário de uma tarefa *soft*, neste tipo de tarefa a perda de seu *deadline* implica na inviabilidade do uso dos resultados obtidos.

[3]Acrônimo de *Real-Time Services Network-on-Chip* (BEREJUCK, 2015).

as garantias de WCL por não alocar os recursos da rede para a transmissão de pacotes. Além disso, a RTSNoC faz uso do algoritmo de roteamento estático XY, onde cada núcleo da rede pode simultaneamente enviar e receber pacotes no mesmo ciclo de *clock* (DING; HO; TSAY, 1994). A escolha desse algoritmo na RTSNoC ocorreu devido sua simplicidade de implementação e prevenção de *deadlock* [4].

O roteamento é o que define um caminho feito pelo pacote entre uma fonte e um destino, considerado por muitos projetistas o principal desafio no projeto da camada de rede em uma NoC, já que apresenta grande influência na média da taxa de envelhecimento dos pacotes que transitam pela rede. Essa taxa relaciona o período de injeção do pacote na rede com o seu *deadline*, portanto, sistemas de tempo real devem apresentar uma média relativamente baixa. Uma taxa de envelhecimento acelerada ocorre quando há atraso na entrega dos pacotes, congestionamento da rede e também pelo grande número de saltos que um pacote faz da origem até o destino. Além disso, um roteamento também pode ser classificado como determinístico ou adaptativo de acordo com o modo que os pacotes são transmitidos (ZHANG et al., 2009). No determinístico o caminho é unicamente definido pelo endereço de origem e destino do pacote e no adaptativo, o caminho é uma função das informações de endereçamento e das condições de tráfego da rede.

Nas redes NoC atuais existe uma tendência no uso de algoritmos adaptativos, como pode ser visto no trabalho dos autores Ghaderi, Alqahtani e Bagherzadeh (2017), que propõem uma rede NoC com roteamento adaptativo que visa diminuir a taxa de envelhecimento dos pacotes. Esta rede faz uso de um mecanismo integrado à porta de entrada dos seus roteadores para monitoramento do tráfego. Na RTSNoC o roteamento utilizado é determinístico, contudo, como cita Berejuck (2015) na sessão de trabalhos futuros, ao tratar de sistemas com uma dinâmica maior no comportamento dos fluxos a introdução da avaliação em tempo de execução das informações de tráfego da rede faz-se necessária para que só assim a rede possa adotar alguma técnica de adaptabilidade.

1.2 HIPÓTESES E JUSTIFICATIVA

Em seu trabalho intitulado *An overview about Networks-on-chip with multicast support* Berejuck (2016) apresenta a *Network Inteface* (NI), um componente localizado entre a conexão de um núcleo com a rede que atua como uma ponte para os pacotes. A primeira hipótese levantada seria introduzir junto a este componente um mecanismo de mensura das informações de tráfego da rede, denominado *Sniffer* (S). Esta abordagem, diferente da-

[4] Diz-se *deadlock* quando um núcleo da rede fica bloqueado e esperando que o canal esteja livre para poder transmitir (BEREJUCK, 2015).

quela adotada por Ghaderi, Alqahtani e Bagherzadeh (2017), considera também o atraso gerado pelas interfaces de rede, assim, as informações obtidas serão de fato mais acuradas. Este mecanismo, por premissa, deve observar a rede sem interferir no seu funcionamento.

Uma segunda hipótese seria a de conectar esses mecanismos a um dispositivo central por meio de barramentos dedicados sem fazer uso dos roteadores da rede ou dos seus canais de interconexão. Este dispositivo, denominado *Network Manager* (NM), seria responsável por armazenar os dados coletados pelos *Sniffers* e gerar relatórios que possam, futuramente, ser utilizados em algoritmos de reconfiguração da rede.

As hipóteses levantadas neste trabalho, se comprovadas, justificariam a introdução dos mecanismos de monitoramento do tráfego no projeto da RTSNoC com o intuito de torná-la uma rede adaptativa. Outra contribuição é no estudo de uma possível divisão das redes intra-chip em camadas de serviços, como nas redes computacionais convencionais. Uma terceira contribuição seria a avaliação do uso de técnicas não-intrusivas no monitoramento da entrega de pacotes dessas redes. A grande maioria dos trabalhos científicos atuais utiliza dos próprios roteadores na observação do tráfego, o que representa uma grande economia de silício, porém, acaba por aumentar a latência média da rede ou não apresentar uma melhoria significativa no roteamento, como pode ser visto em *Evaluation of pseudo adaptive XY routing using an object oriented model for NOC* dos autores Dehyadgari et al. (2005). A proposta apresentada neste trabalho exime esta função dos roteadores ou demais componentes da rede relacionados diretamente ao roteamento.

1.3 OBJETIVOS

Esta seção descreve o objetivo geral e os objetivos específicos deste trabalho de conclusão de curso.

1.3.1 Objetivo Geral

O principal objetivo deste trabalho é desenvolver um sistema capaz de monitorar a entrega de pacotes na rede RTSNoC. Cada elemento conectado à rede conterá um mecanismo que observará o tráfego de forma não-intrusiva. As informações obtidas por eles serão centralizadas em um segundo elemento que atuará como gerente da rede.

1.3.2 Objetivos Específicos

- Identificar as métricas que induzem o atraso na entrega de pacotes nas redes intra-chip.
- Aplicar técnicas provenientes do campo da observabilidade computacional às redes intra-chip.
- Simular o comportamento de uma aplicação real com base nos padrões de tráfego das redes de interconexão.
- Verificar por meio de um ambiente simulado o funcionamento do sistema de monitoramento introduzido à RTSNoC.

1.4 PROCEDIMENTOS METODOLÓGICOS

Para atingir os objetivos apresentados na Seção 1.3, foi definida a estratégia composta pelas seguintes etapas:

1. Revisar o estado da arte das redes intra-chip, com foco naquelas projetadas para SoC's de tempo real;
2. Portar a rede RTSNoC para a linguagem SystemC de síntese e descrição de hardware;
3. Projetar uma Interface de Rede para ser utilizada nos experimentos com a RTSNoC;
4. Desenvolver os componentes que atuarão na camada de serviço referente ao monitoramento da entrega de pacotes;
5. Verificar o funcionamento dos componentes desenvolvidos na etapa anterior com auxílio da ferramenta GTKWave;
6. Simular estímulos de aplicações reais através de geradores de tráfego baseados em padrões para redes de interconexões;
7. Validar o funcionamento do sistema de monitoramento por intermédio da produção de relatórios;
8. Avaliar os dados coletados no item anterior e apresentar as devidas considerações finais.

2 TRABALHOS RELACIONADOS

A grande maioria dos projetos de NoC's publicados tem como foco principal a minimização da latência média dos fluxos e a maximização da utilização da largura de banda, tipicamente oferecendo suporte aos fluxos de melhor esforço e de pior caso. O que pôde ser observado na literatura é o grande volume de NoC's que fazem uso de técnicas relacionadas a roteadores não bloqueantes com controle de fluxo, chaveamento de circuitos, muitas vezes adotando mecanismos de Multiplexação por Divisão de Tempo (*Time Division Multiplexing* - TDM) e, principalmente, no uso de roteamento estático. É possível perceber também que desde seu surgimento até o momento a maioria das NoC's são projetadas de acordo com sua latência de pior caso em tempo de execução, ou seja, todo o fluxo da rede é mapeado e conhecido durante a fase de desenvolvimento e assim a rede é projetada a partir do seu pior caso, característica essa que como dito nas seções prévias deste trabalho limita o dinamismo da rede e o reuso de componentes já consolidados no mercado.

O capítulo em questão apresenta os estudos feitos sobre sete NoC's voltadas para sistemas de tempo real, com destaque às abordagens utilizadas no controle do tráfego de pacotes. Este capítulo foi dividido em sete seções que apresentam aquelas NoC's, descrevendo a topologia de cada rede, os aspectos funcionais dos seus roteadores e as técnicas adotadas por cada uma delas para garantir a entrega dos pacotes. Além disso, subseções destacam duas propriedades de suma relevância a avaliação de cada rede: (i) o tempo de espera, que está relacionado ao tempo que algumas redes necessitam para requisitar ou liberar recursos da rede ($T_{wait;req}$, $T_{wait;reply}$); e (ii) a latência que um fluxo pode sofrer numa comunicação entre dois roteadores quaisquer ($T_{latency}$) (BEREJUCK, 2015). Por fim, a seção 2.8 finaliza este capítulo apresentando um resumo geral das redes verificadas.

2.1 ÆTHEREAL

A rede Æthereal é uma rede baseada em chaveamento de circuitos TDM desenvolvida por Goossens, Dielissen e Radulescu (2005) no *Philips Research Laboratories* com serviços de suporte GS, para tráfegos críticos e BE para flexíveis. Além disso, essa rede faz uso do conceito de alocação temporal onde a comunicação é dividida em *time slots* de duração fixa e igualitária para todos os roteadores da rede, que de modo síncrono propagam os dados. Portanto, cada roteador tem o mesmo período de tempo para encaminhar um bloco de dados para um canal de saída e a alocação geral dos *slots* é definida em tempo

de projeto.

Os roteadores desta rede possuem complexidade *N*, isto é, *N* entradas e *N* saídas, e fazem uso de uma Tabela de *Slots* no roteamento, onde cada *S time slot* é representado nas linhas desta tabela e as saídas são inseridas nas colunas. Em um *slot s* o nó da rede pode ler e escrever um único bloco de dados em cada entrada e saída, respectivamente. No próximo *slot*, (s + 1), o nó da rede escreve e lê blocos de dados em seus canais de saída apropriados. Assim, o blocos são propagados na rede em uma política *store-and-forward* no intuito de evitar *deadlock*. A latência sofrida por um bloco de dados se torna igual à duração do *slot* e a reserva de *slots* garante a largura de banda em múltiplos blocos de dados por *S slots*.

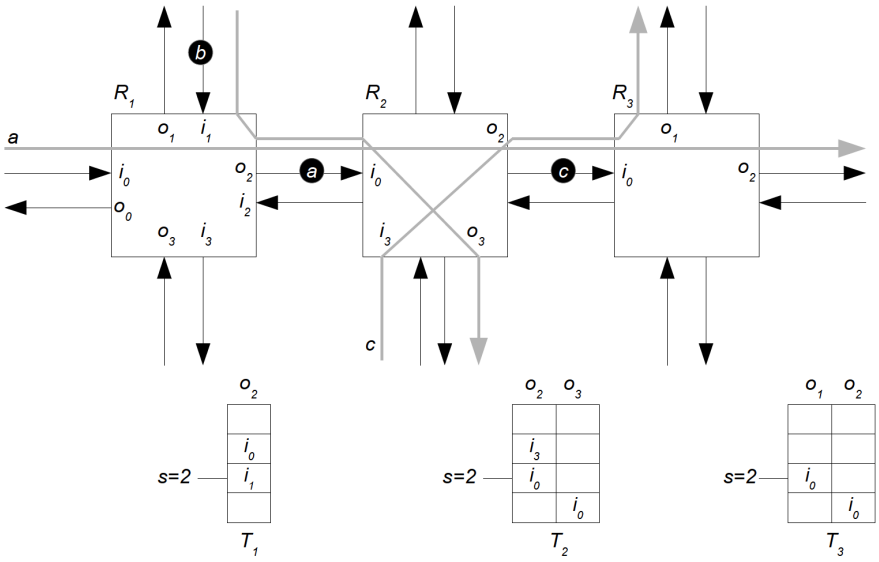
Na rede Æthereal a Tabela de *Slots* pode ser descrita como uma função que associa as entradas com as saídas do roteador: $T(s,o) = i$, ou seja, blocos de dados presentes em uma entrada i serão comutados para a saída o a cada $s + kS$ *slots*. Uma entrada está vazia quando não há reserva para uma saída qualquer em um determinado *slot* e não há contenção na rede pois existe pelo menos uma entrada para cada saída, para cada *slot*. Atribuir *slots* para conexões na rede é um problema de otimização e a escolha do algoritmo utilizado depende unicamente do tipo e requisitos da aplicação.

A Figura 1 demonstra o conceito de roteamento livre de contenção proposto pelos autores. Na figura, temos a representação de três roteadores, R_1, R_3 e R_2, no *time slot* $s = 2$, indicado na terceira entrada de cada tabela T. O tamanho das tabelas é $S = 4$, somente as colunas relevantes são representadas na figura. As três setas destacadas *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.

A sincronicidade é implementada com o uso de um único e centralizado sinal de *clock* distribuído através das técnicas provindas do modelo cascata de sincronização e design síncrono insensível à latência. Entretanto, sincronicidade distribuída também é implementável, para toda sincronização de *slot* o roteador e produz um *token* para cada saída antes de consumir um *token* para cada entrada, assim sincronizando cada *slot* com todos os de seus vizinhos. Com isso, todos os roteadores estão contidos sempre no mesmo *time slot* e a NoC num geral irá executar na mesma velocidade que o roteador mais lento.

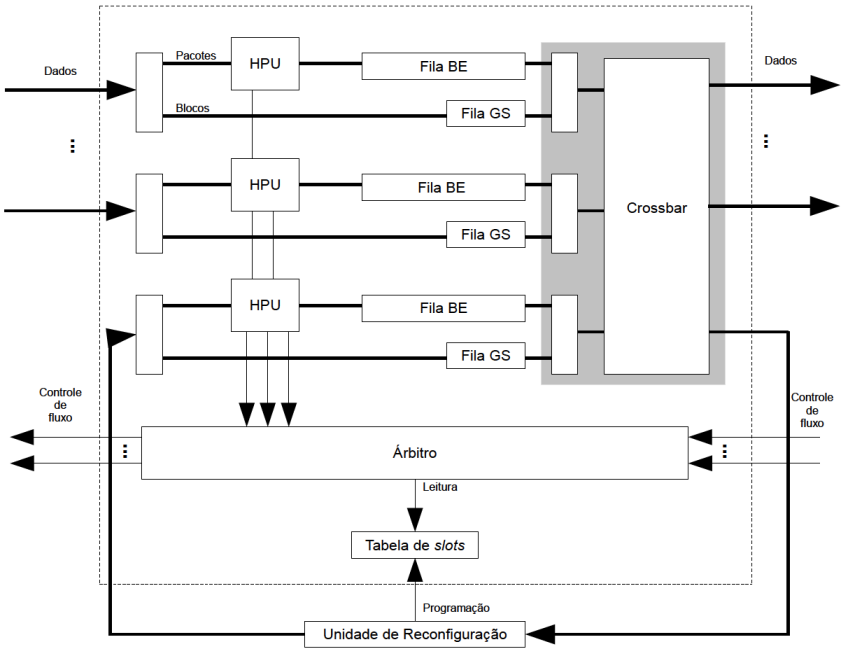
A Æthereal faz uso de uma arquitetura GS-BE combinada. Seus roteadores nada mais são que a junção de um roteador GS e um BE em paralelo como demonstra a Figura 2. O *flit* proveniente de um fluxo BE só pode fazer uso do canal quando não existir um bloco GS sendo transmitido, portanto, o roteador BE possui uma prioridade menor que o

Figura 1 – Roteamento livre de contenção.



Rede com três roteadores (R_1, R_2 e R_3) no *time slot* $s = 2$, com as correspondentes tabelas de *slots* (T_1, T_2 e T_3). Fonte: (GOOSSENS; DIELISSEN; RADULESCU, 2005).

Figura 2 – Arquitetura interna dos roteadores da Æthereal.



Fonte: adaptada de Goossens, Dielissen e Radulescu (2005).

GS. A arquitetura GS é livre de contenção, cada entrada requer uma fila para um único bloco e cada fila é conectada a um *switch* que é configurada a cada *slot* s para ler $T(s, o) = i$ entradas provenientes da Tabela de *Slots*. Assim, os blocos GS nunca esperam e controle de fluxo em nível de canal entre os roteadores não é necessário. Já a arquitetura BE consiste de um roteamento *wormhole* convencional com arbitragem *Round-robin* nos

switchs. Controle de fluxo em nível de canal é utilizado nos roteadores BE para evitar congestionamentos de filas e o cabeçalhos dos pacotes transmitidos contém o caminho de origem até destino, ou seja, faz-se uso do conceito de roteamento de origem. No intuito de aumentar a reutilização de recursos, a rede Æthereal permite que fluxos BE utilizem, além dos *time slots* não reservados para os fluxos prioritários, os *time slots* reservados não utilizados momentaneamente pelos fluxos GS.

2.1.1 Propriedades para avaliação da entrega de pacotes

O conceito de escalonamento em *time slots* é a base da rede Æthereal. Isto implica que o período de um escalonamento é seguidamente repetido. Para um circuito virtual, considerando um período de P *time slots* e p como o número de *slots* que podem ser alocados para o circuito, as propriedades do tempo de execução dependem dos 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 um pacote (f).

Tempo de espera - para um circuito virtual que possui p *time slots* o tempo de espera é função apenas da requisição, já que após reservado o recurso a rede não espera nenhum tipo de resposta e está apto para transmitir. Assim, $T_{wait;req} = (P - p).s$ ciclos de relógio, sendo que p varia entre os limites $[1; P]$ e $T_{wait;reply} = 0$.

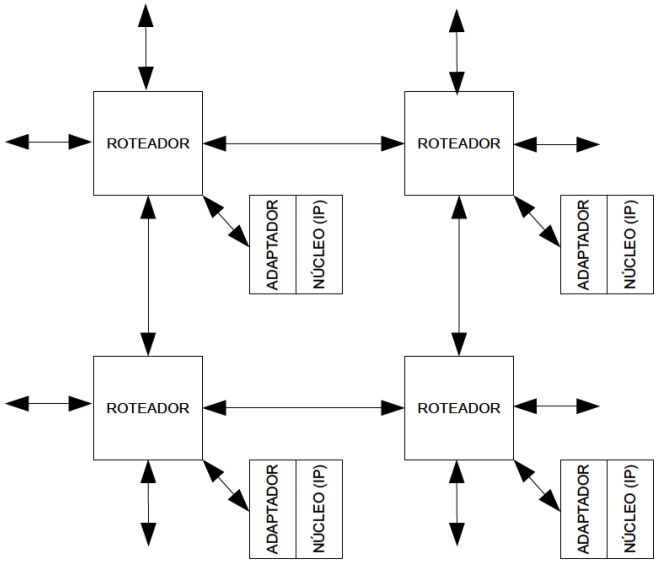
Latência - a latência de um pacote em cada roteador é função da profundidade do *pipeline (B)*, o qual é igual a duração de um *time slot* em ciclos de relógio (s), e do número de *flits* do pacote (f). Considerando h roteadores no caminho entre a origem do pacote até o seu destino a latência é expressa como $T_{latency} = h.B + f$ ciclos de relógio.

2.2 SOCBUS

A SoCBUS é uma rede desenvolvida por Sathe, Wiklund e Liu (2004) na *Linköping University* também baseada na técnica do chaveamento de circuitos. Os autores alcunham como *IP Cores* os blocos conectados à rede e como *switches* os nós responsáveis pelo roteamento e entrega dos pacotes, assim, as terminologias de núcleos e roteadores, respectivamente, serão adotadas durante a análise desta rede. A topologia adotada é a de malha ortogonal 2D, como pode ser vista na Figura 3, devido a sua facilidade de interconexão, largura de banda razoavelmente alta e por privilegiar a comunicação de núcleos com alta taxa de transmissão sem que esses consumam uma grande quantidade de recursos da rede.

Os roteadores apresentam uma configuração de cinco portas, quatro utilizadas para

Figura 3 – Configuração de uma rede 4x3 na arquitetura SoCBUS.



Fonte: adaptada de Wiklund e Liu (2003).

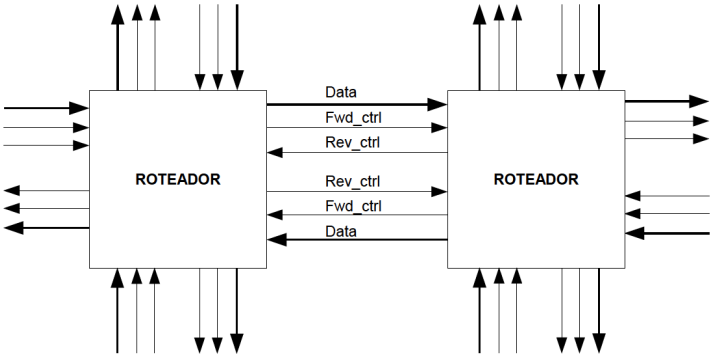
conexão com os roteadores adjacentes e uma porta para interconectar o núcleo à rede. A porta proveniente do roteador é conectada ao núcleo por meio de um adaptador (do inglês, *wrapper*), dispositivo responsável por preparar e armazenar os pacotes para a transmissão e também por lidar com possíveis diferenças de protocolo entre a rede e o núcleo.

A rede utiliza um procedimento baseado no envio de pacotes para estabelecer conexões ponto-a-ponto e 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 roteamento dos dados acontece em uma transação de quatro fases. Inicialmente, a conexão é estabelecida por um pacote enviado através da rede requisitando temporariamente os recursos necessários. Posteriormente, um pacote de aceite (ACK) ou recusa ($NACK$) é enviado de volta. Se a requisição for aceita, a conexão é mantida; caso contrário é desfeita. Na fase precedente à de *handshake* os dados podem ser transferidos utilizando os recursos reservados. Na última fase, após os dados terem chegado ao seu destino um pacote é enviado no sentido contrário da transmissão, confirmando o recebimento dos dados e cancelando a requisição dos circuitos ao mesmo tempo, quando necessário.

A interconexão dos roteadores na SoCBUS é composta por onze sinais em cada direção, onde oito deles carregam os dados e as requisições de roteamento dos pacotes, um sinal é usado para controle de encaminhamento, ou seja, para gerenciar a temporização das transmissões e os dois últimos sinais são usados para controle reverso e carregam os *acknowledges* de cada negociação. A Figura 4 mostra com detalhes a interconexão entre roteadores na SoCBUS. Naquela figura as linhas diagonais representam a interconexão

com os núcleos.

Figura 4 – Exemplo de interconexão entre roteadores na rede SoCBUS.



Fonte: adaptada de Wiklund e Liu (2003).

A sincronicidade é implementada com um sistema de relógio [1]mesócrono paralelamente à técnica de *retiming* dos sinais ao longo do sistema com o intuito de minimizar o *delay* gerado nas interconexões e o desvio de relógio. A adoção dessas técnicas traz algumas vantagens: não é necessário o uso de repetidores; o consumo de energia do sistema é reduzido; não há necessidade da rede ser inserida no chip de forma.

Durante a fase de requisição dos circuitos um algoritmo de busca por caminhos de menor tamanho é utilizado. A cada salto (do inglês *hop*) entre os roteadores, o pacote de requisição dos recurso é roteado para um dos roteadores disponíveis com base no endereço de destino, ou seja, o roteamento na SoCBUS é distribuído (ZHANG et al., 2009). Esta busca é feita de modo *round-robin* e se não houver roteadores disponíveis, é iniciado um protocolo de falha, no qual as conexões previamente estabelecidas são desfeitas e uma nova tentativa de estabelecimento da conexão é efetuada num período posterior. Após obter sucesso, os dados são encaminhados da origem ao destino através dos roteadores e enlaces reservados ocupando toda a largura de banda da conexão, sem postergação.

2.2.1 Propriedades para avaliação da entrega de pacotes

A SoCBUS não faz uso de componentes específicos para garantia de serviço já que a reserva de recursos na comunicação já é garantia de que os requisitos de tempo real serão atendidos. Contudo, o uso desta técnica pode resultar na incorreta utilização e gerenciamento dos recursos da rede, pois abre possibilidade a uma conexão não ser atendida devido falta de recursos e, como consequência, o desperdício da largura de banda com requisições de conexão repetidas e o comprometimento das propriedades de tempo real.

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

A seguir são apresentadas as avaliações feitas sobre a rede.

Tempo de espera - para um caminho de h saltos, são necessário quatro ciclos de relógio para estabelecimento da conexão e um ciclo para envio do pacote de aceite. Para cada transação é preciso considerar um *overhead* de h ciclos para que a conexão seja desfeita após o envio dos dados. Assim, o tempo de requisição é $T_{wait;req} = 4.h + h$. O tempo de resposta $T_{wait;reply} = 0$, já que pela reserva de recursos não existe espera de uma resposta para iniciar a transmissão.

Latência - uma vez estabelecida a conexão, é necessário um único ciclo de relógio para que um pacote atravesse o roteador. A latência total é descrita como $T_{latency} = h$, ou seja, é função unicamente do número de saltos realizados pelo pacote.

2.3 ×*PIPES*

A [2]×*pipes* é uma rede escalável, altamente parametrizável, insensível à latência voltada para ambas arquiteturas homogêneas e heterogêneas de SoC's. Esta rede foi desenvolvida inicialmente por Dall'Osso et al. (2003) na *Università di Bologna* e teve seu projeto continuado por Bertozzi e Benini (2004). A ×*pipes* oferece grande flexibilidade devido os seus parâmetros de projeto serem definidos em tempo de execução e o protocolo de comunicação ser totalmente independente dos núcleos conectados à rede, ou seja, é responsabilidade unicamente da rede e utiliza do conceito de conexão *end-to-endo OCP* que, basicamente, consiste na separação do canal de comunicação em *pipelines* de requisições e respostas.

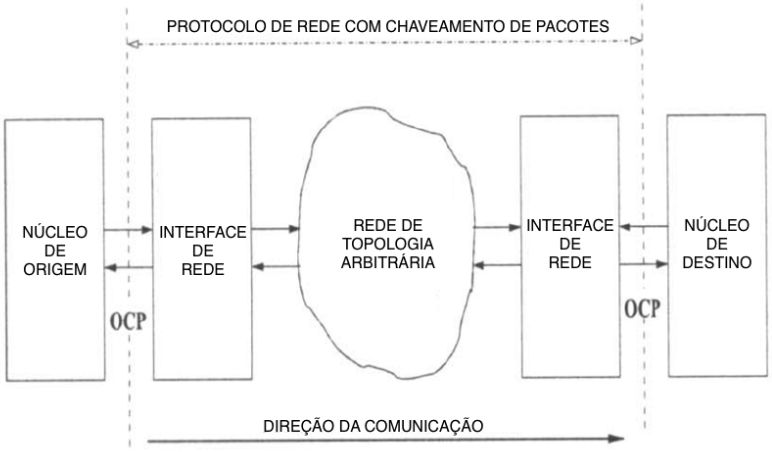
A metodologia de design proposta pela rede é realizada em duas fases. Na primeira, um *template* parametrizável de objetivo geral desenvolvido em *SystemC* contendo todas as informações da rede é disponibilizado ao projetista. Posteriormente, em tempo de execução, os componentes são conectados e configurados de acordo com o propósito do projeto através da especificação e instância dos parâmetros da rede. Não existe uma topologia pré-definida para uso na ×*pipes*, assim, os roteadores podem ser conectados de forma arbitrária.

Uma interface de rede (do inglês, *network interface*) interconecta os núcleos à rede e têm como objetivos principais inserir os dados em pacotes e converter o protocolo de comunicação de *end-to-end* para o utilizado na rede. É também nesse componente onde o cabeçalho do pacote contendo as informações de origem e destino a serem utilizadas no roteamento é construído. Cada pacote é quebrado em *flits*, que necessitam de apenas um ciclo de relógio para serem transmitidos. Com o intuito de manter a complexidade de

[2]Pronunciada como "*crosspipes*", termo derivado de "*crossing pipelines*"(DALL'OSSO et al., 2003).

projeto da rede baixa, a interface de rede atende somente uma requisição de leitura do núcleo por vez e múltiplas requisições de escrita. A Figura 5 demonstra um esquemático da arquitetura de comunicação da ×*pipes*.

Figura 5 – Arquitetura da rede ×*pipes*.



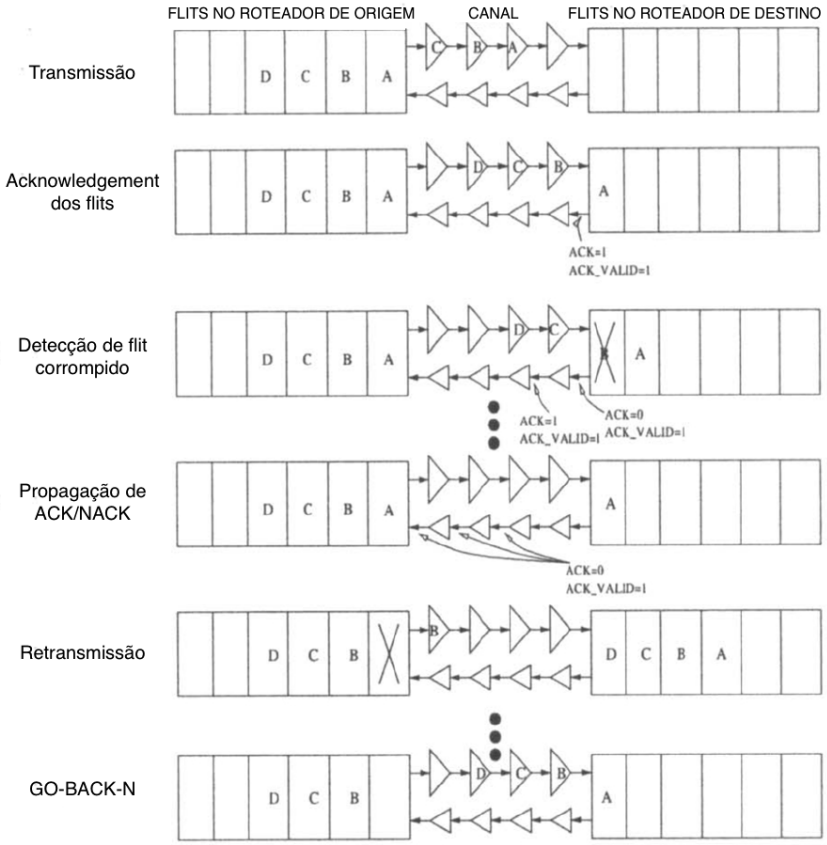
Fonte: adaptada de Dall'Osso et al. (2003).

O encaminhamento de pacotes na ×*pipes* é baseada na técnica *wormhole*, um *flit* só é transmitido para o próximo roteador se este conter recursos para armazená-lo. Os roteadores são configurados para apresentar múltiplos canais virtuais multiplexados em um único canal físico e a transmissão é feita *flit*-a-*flit*. Além disso, os canais podem apresentar diferentes tamanhos dentro da rede. A concepção do projeto tem como premissa a entrega ordenada dos *flits*, independente do seu tempo de residência. Os roteadores implementam a estratégia *GO-BACK-N* para retransmissão de pacotes corrompidos, como demonstra a Figura 6. Cabe citar, também, que o *pipelining* é utilizado para ambos os canais de dados e controle.

Na ×*pipes* foi adotado um algoritmo de roteamento estático denominado *street sign routing*. Neste algoritmo, os roteadores possuem acesso à uma tabela baseada nos endereços de destino que indica a saída pelos quais os *flits* de um determinado pacote devem ser direcionados. O armazenamento dos *flits* é feito nos *buffers* de saída dos roteadores. Um módulo de saída é responsável por gerar sinais de controle do fluxo baseados na política ACK/NACK de transação.

No intuito de otimizar a utilização dos ciclos de relógio, o roteamento é composto de sete estágios em *pipeline*. O primeiro estágio verifica o cabeçalho dos pacotes recebidos em diferentes portas de entrada para determinar se esses pacotes devem ser encaminhados através da porta de saída requisitada no momento. Somente os pacotes correspondentes seguem para o segundo estágio, que consiste de um árbitro que usa da política *round-robin* para resolver a contenção de pacotes. O árbitro gera os sinais NACK para os *flits* que

Figura 6 – Exemplo da estratégia usada na retransmissão de um *flit* corrompido.



Fonte: adaptada de Dall'Osso et al. (2003).

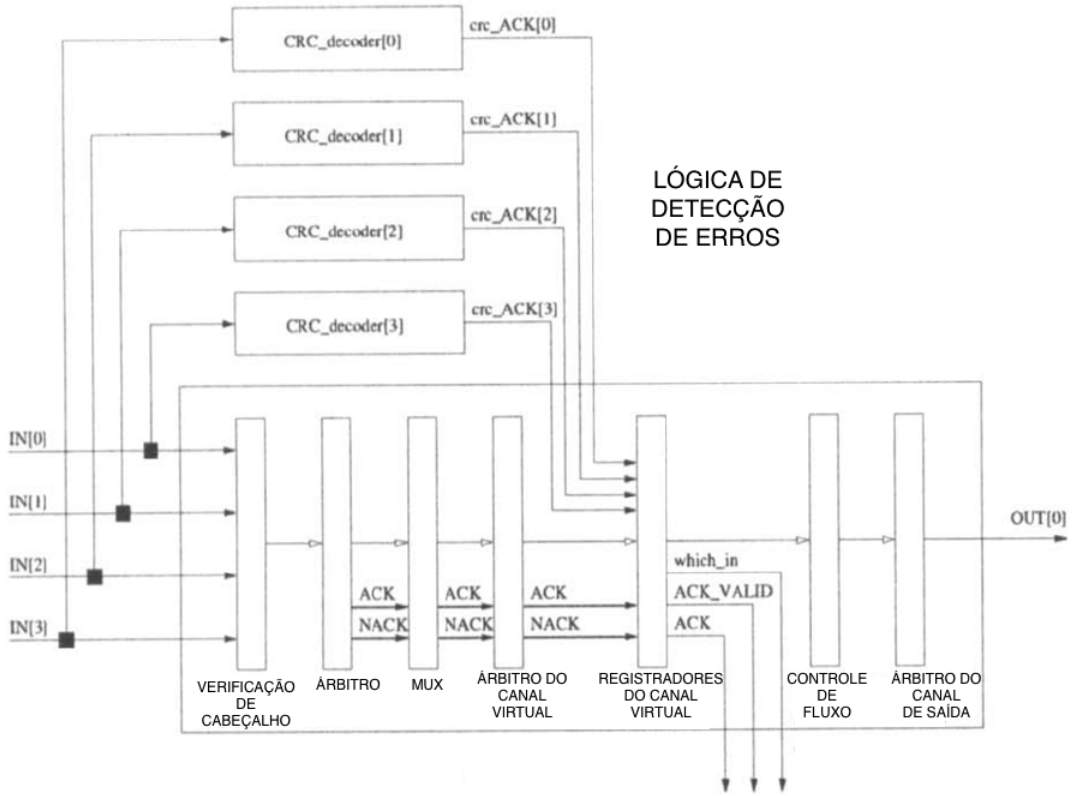
não são selecionados. O terceiro estágio é apenas um multiplexador que seleciona a porta de entrada priorizada. No quarto estágio é realizada a busca por um registrador livre para armazenar os *flits*. O quinto estágio consiste do armazenamento em si. O sexto e sétimo estágios consistem, respectivamente, no encaminhamento do *flit* para o próximo roteador e na multiplexação dos canais virtuais no canal físico. A Figura 7 demonstra essa arquitetura de comunicação em *pipeline*.

2.3.1 Propriedades para avaliação da entrega de pacotes

A ×*pipes* faz uso da técnica de *pipeline* no roteamento. Os autores propõe o uso de sete *pipelines* para um cenário livre de contenção, onde a taxa de transmissão alcançada foi de um *flit* por ciclo de relógio. O mesmo cenário será utilizado na análise desta rede.

O propósito da rede é ser capaz de atender qualquer arquitetura de SoC, tanto heterogênea quanto homogênea, objetivo este alcançado por meio da implementação das interfaces de rede e do uso de uma arquitetura altamente parametrizável. Seus componentes são insensíveis à latência, portanto, os dados fluem pela rede a taxas que não estão vinculadas pela latência de pior caso. Em contrapartida, o consumo dos ciclos de

Figura 7 – Estratégia de transmissão em *pipeline*.



Na figura, cada *pipeline* representa um dos estágios que envolvem a transmissão dos dados. Fonte: adaptada de Dall'Osso et al. (2003).

relógio relativos à sincronização entre o protocolo de comunicação da rede e do núcleo é imprevisível.

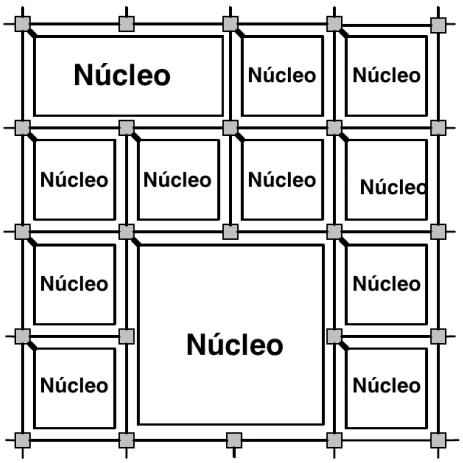
Tempo de espera - na ×*pipes* não há reserva de recurso e os *flits* são enviados mesmo sem recebimento do sinal de ACK, portanto, o tempo de requisição é nulo. Os registradores nos canais virtuais dos roteadores possem capacidade de armazenar $2n + m$ *flits*, sendo n a largura do canal físico e m a quantidade dos ciclos de relógio necessários para processar o roteamento. Um *flit* somente é transmitido adiante após o recebimento da mensagem de ACK, a qual necessita de $2n + m$ ciclos para transitar na rede. O tempo de resposta, portanto, é descrito como $T_{wait;reply} = 2n + m$.

Latência - para o envio de um *flit* é necessário considerar o número de saltos (h) realizados, a profundidade dos *pipelines* (B), a largura do canal físico (n) e os ciclos de relógio consumido no roteamento e conversão do protocolo de comunicação pelas interfaces de rede (m). Assim, a latência total na ×*pipes* é de $T_{latency} = h.B + 2n + m$.

2.4 QNOC

A rede QNoC[3] é uma rede escalável com garantia de QoS desenvolvida por Bolotin et al. (2004) no *Israel Institute of Technology*. Os autores propuseram uma topologia em malha irregular, como demonstra a Figura 8, por ser a melhor combinação à estrutura planar e irregular comumente utilizadas no design de SoC's. Cada núcleo do sistema é conectado ao roteador por meio de uma interface padrão (do inglês, *Standard Interface*), onde a largura de banda adapta-se às necessidades comunicativas do núcleo. De forma similar os *links* entre roteadores tem a largura de banda ajustada para acomodar o tráfego esperado e atender os requerimentos de QoS do sistema. Estes ajustes são realizados ao alterar tanto o número de conexões quanto a frequência dos dados.

Figura 8 – Topologia em malha irregular.



Fonte: adaptada de Bolotin et al. (2004).

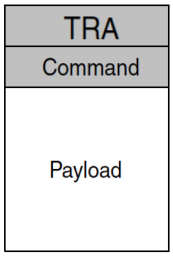
Os autores da QNoC identificaram quatro tipos diferentes de comunicação e definiram níveis de serviços adequados para suportá-los: o serviço *Signaling* atende os pacotes com prioridade mais alta para assegurar baixa latência, e, é utilizado por interrupções e sinais de controle, uma forma de evitar a reserva de recursos; *Real Time* é o serviço que garante um nível máximo de latência às aplicações de tempo real ao alocar para cada *real-time link* uma fração da largura de banda; o nível *Read/Write* é projetado para suportar acessos curtos às memórias e aos registradores; por fim, o nível de serviço *Block-Transfer* é usado na transferência de mensagens longas e grandes blocos de dados, tal como linhas de *cache* e transferências DMA (*Direct Memory Access*), portanto, é o nível de menor prioridade. Quando dois fluxos com níveis de serviços diferentes competirem por um mesmo recurso do roteador, o acesso será concedido àquele serviço de maior prioridade. Caso os

[3]Acrônimo de *Quality-of-Service Network-on-Chip* (BOLOTIN et al., 2004).

fluxos sejam da mesma classe de serviço a disputa é resolvida por meio de arbitragem *round-robin*.

Os pacotes da QNoC são formados por três campos: informações de roteamento, comandos e dados, como pode ser visto na Figura 9. Os autores também alcunham o campo das informações de roteamento como TRA (do inglês, *Target Routing Address*) onde estão contidas as informações de roteamento. O nível de serviço dos quais os dados provêm é especificado no campo de comandos. O campo de dados é onde o *payload*[4] trafega. Como comumente visto em NoC's, os pacotes são divididos em *flits*, estes classificados em três tipos básicos: *Full Packet* (FP), primeiro *flit* do pacote; *End of Packet* (EP), último *flit* do pacote; *Body* (BDY), os *flits* intermerdiários que trafegam entre FP e EP.

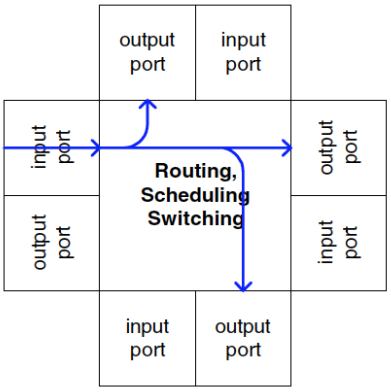
Figura 9 – Formato dos pacotes na QNoC.



Fonte: (BOLOTIN et al., 2004).

Os roteadores da rede QNoC possuem até cinco canais de comunicação, uma utilizada para conexão local com um núcleo e outras quatro para conexões com roteadores adjacentes. Quando os *flits* chegam em uma porta de entrada do roteador são, primeiramente, armazenados em um *buffer*, para só então serem encaminhados às portas de saída requisitadas, como pode ser visto na Figura 10. Cabe citar que existe um *buffer* para cada nível de serviço.

Figura 10 – Fluxo interno dos dados.

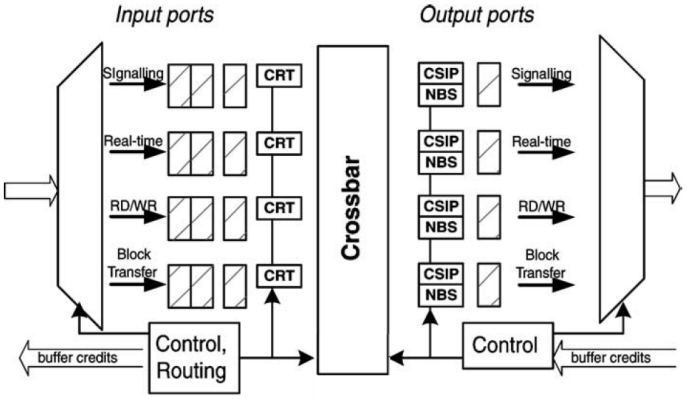


Fonte: (BOLOTIN et al., 2004).

[4] A informação transmitida de fato, subtraída daquelas geradas no roteamento (ZHANG et al., 2009).

O roteamento dos pacotes é feito por um algoritmo de caminho mais curto multi-classe do tipo *wormhole* baseado no algoritmo XY estático, com as informações de origem e destino já determinadas durante a injeção do pacote à rede. Até que o *flit* do tipo EP não seja recebido pelo roteador é mantida uma tabela, a CRT(do inglês, *Current Routing Table*), que contém as informações de roteamento para cada nível de serviço e para cada porta de entrada. Quando o *flit* é transmitido avante, o *buffer* torna-se livre para armazenamento e um crédito de *buffer* é retornado para o roteador adjacente àquela porta de entrada por um canal dedicado no intuito de informá-lo da viabilidade. Essa informação é armazenada no roteador em um campo denominado NBS (do inglês, *Next Buffer State*), como demonstra a Figura 11.

Figura 11 – Arquitetura do roteador na QNoC.



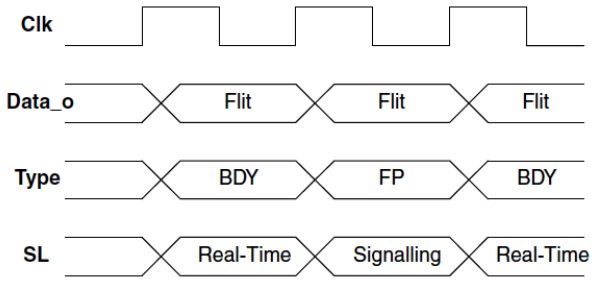
Fonte: adaptada de Bolotin et al. (2004).

A QNoC provê um método de escalonamento preemptivo, ou seja, se chegar um pacote de maior prioridade em uma das portas de entrada do roteador durante a transmissão do pacote em andamento, o roteamento deste é interrompido para que o de maior prioridade seja atendido. Um *flit* também só é transmitido para o canal de saída enquanto o *buffer* do canal de entrada do roteador seguinte conter espaço disponível para recebê-lo e não houver nenhum pacote com serviço de maior prioridade escalonado para o referido canal de saída. A Figura 12 demonstra como os sinais de controle dos canais de saída nos roteadores da rede QNoC estão estruturados, onde a Tabela 1 descreve as funcionalidades de cada um desses sinais.

A rede é projetada com requisitos de tráfego limitados nos níveis de serviço com prioridade mais alta, estratégia que visa evitar o *starvation*[5] dos serviços com prioridade mais baixa.

[5]Diz-se *starvation* quando um pacote fica permanentemente parado devido os recursos requisitados por este estarem em constante uso pelos demais pacotes que também os solicitam.(BEREJUCK, 2009).

Figura 12 – Sinais nos canais de saída dos roteadores da QNoC.



Ao observar as ondas na linha SL pode-se perceber a preempção de um pacote do tipo *Real-Time* por um *Signalling*, geralmente composto por um único *flit*. Fonte: (BOLOTIN et al., 2004).

Tabela 1 – Sinais nos canais de saída dos roteadores da QNoC.

Sinais	**Largura**	**Descrição**
Clk	1	Relógio que sincroniza a transmissão dos *flits*.
Data_o	Parametrizável	Dados saindo do roteador.
Type	2	Tipo do *flit*: 00: IDLE; 01: EP; 10: DBY; 11: FP.
SL	2	Nível de serviço do *flit*.

Fonte: adaptada de Bolotin et al. (2004).

2.4.1 Propriedades para avaliação da entrega de pacotes

A QNoC é uma rede baseada na premissa de que a topologia de uma rede pode variar de acordo com as necessidades do sistema. Com base nas diferenças de tamanhos dos núcleos e seus posicionamentos, os autores propuseram o uso de uma topologia em malha irregular combinada com uma arquitetura parametrizável.

Os autores da rede também identificaram quatro tipos de comunicação em um SoC com prioridades diferentes. Como forma de evitar o uso de técnicas baseadas na reserva de recursos, a rede utiliza do método de escalonamento preemptivo e de um roteamento do tipo *wormhole*. Possíveis conflitos de prioridade são resolvidos por uma política *Round-Robin* e um *flit* só é transmitido se o próximo roteador conter espaço livre em seu *buffer* do canal de entrada para armazená-lo e o canal de saída não estiver escalonado para um pacote de prioridade mais alta no momento da requisição.

Tempo de espera - a rede QNoC não faz uso de técnicas baseadas na reserva de recursos e no chaveamento de circuitos, ao invés disso, os fluxos de dados da comunicação são divididos em quatro níveis de serviço com prioridades diferentes, portanto, $T_{wait;req} = 0$. Em contrapartida, pacotes de fluxos com diferentes prioridades competem pelo mesmo canal de transmissão. Isso significa que independente de ser preemptivo ou não, pacotes de menores prioridades necessitam esperar que a transmissão de um com maior prioridade seja concluída. Portanto, a resposta do roteador para início de uma transmissão depende

da quantidade de *flits* que compõem o pacote daquele fluxo que ocupa o canal naquele instante. Assim, um pacote pode esperar por $T_{wait;reply} = f.m$ ciclos de relógio para obter uma resposta se pode ou não avançar.

Latência - ao chegar no canal de entrada do roteador, os *flits* do pacote são armazenados em *buffers* para posteriormente serem encaminhados ao canal de saída solicitado. Considerando que um pacote pode realizar vários saltos durante a transmissão e que o processo de roteamento e análise da disponibilidade do canal de saída é realizado a cada salto, a latência da rede é expressa por $T_{atency} = h.f.m + B$.

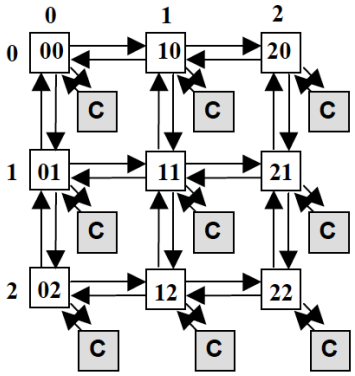
2.5 HERMES

A rede Hermes foi desenvolvida no Instituto de Informática da Universidade Federal do Rio Grande do Sul por Moraes et al. (2004). A rede foi inicialmente concebida para uso em fluxos BE e, posteriormente, foram propostas quatro variações com mecanismos de QoS por Mello (2007): rede Hermes com chaveamento de circuitos (Hermes-CS); rede Hermes com escalonamento baseado em prioridades fixas(Hermes-FP); rede Hermes com escalonamento baseado em prioridades dinâmicas (Hermes-DP) e a rede Hermes com escalonamento baseado em taxas (Hermes-RB). O termo infra-estrutura Hermes vêm, portanto, pelo fato de não existir apenas uma rede, o usuário define aquela que deseja implementar por meio de parâmetros configuráveis.

A topologia utilizada nas redes Hermes é a malha 2D e devido ao fato das redes serem escaláveis e parametrizáveis os roteadores podem apresentar diferentes números de portas de acordo com sua localização. Ao observar a Figura 13 pode-se perceber que o roteador localizado ao centro apresenta uma configuração de cinco portas, enquanto os roteadores localizados nas bordas da rede apresentam de duas à três. O uso da topologia em malha justifica-se na sua simplicidade de implementação do encaminhamento e roteamento dos pacotes. A infra-estrutura também admite as topologias torus, hipercubo ou malhas irregulares. Cabe citar que mudanças na topologia implicam em alterações também na conexão dos roteadores e algoritmo de roteamento.

A rede Hermes-CS oferece um serviço com vazão garantida ou GT(do inglês, *Guaranteed Throughput*) aos fluxos com requisitos de QoS e um serviço de BE àqueles serviços sem QoS. O encaminhamento dos pacotes é feito com o uso de dois canais virtuais, um deles transporta os dados através do chaveamento de circuitos, enquanto o outro transporta os dados por chaveamento dos pacotes. Como um fluxo GT têm prioridade maior sobre os fluxos BE, pacotes provenientes daquele fluxo contam com garantia da latência de fim-a-fim. Os fluxos BE só podem fazer uso do canal quando este não estiver ocupado

Figura 13 – Topologia utilizada na infra-estrutura Hermes.



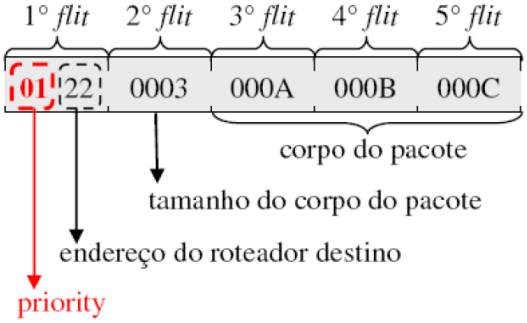
Fonte: (MORAES et al., 2004).

ou reservado para um fluxo GT.

O protocolo de transmissão dos dados provenientes de um fluxos GT é composto, basicamente, de cinco passos: inicialmente o núcleo de origem envia um pacote BE de controle, através do canal virtual para fluxos BE, ao núcleo de destino com o intuito de criar a conexão; em seguida, um sinal de ACK é enviado ao núcleo de origem quando o pacote de controle chega ao seu destino em h ciclos de relógio, onde h é o número de roteadores no caminho da conexão; ao receber o sinal de ACK, a conexão é efetivamente estabelecida e os dados podem ser transmitidos através do canal virtual para fluxos GT; ao término da transmissão um pacote BE é novamente enviado ao destino para solicitar a remoção da conexão; quando o pacote de controle chega ao núcleo de destino, novamente um sinal de ACK é propagado em h ciclos de relógio e só após o recebimento deste sinal pela origem é que a conexão será efetivamente desfeita.

A Hermes-FP, outra alternativa arquitetural, possui mecanismos de recursos baseados em prioridades fixas, o que permite a diferenciação dos fluxos pelas suas necessidades de desempenho. Cada canal virtual está associado a uma prioridade estática. Desse modo, esta versão da rede é capaz de diferenciar até n fluxos, com n sendo o número de canais virtuais por canal físico.

Figura 14 – Campos dos pacotes da Hermes-FP.



Fonte: (MELLO, 2007).

No escalonamento baseado em prioridades fixas, um campo denominado *priority* é inserido no início do primeiro *flit* de um pacote para fins de diferenciação dos fluxos e determinação de qual canal virtual será utilizado, como demonstra a Figura 14. O pacote com maior prioridade é atendido primeiro, mesmo se os outros pacotes estiverem esperando por mais tempo. Logo, a transmissão dos pacotes de menor prioridade depende da carga daqueles de maior, portanto, os limites de latência fim a fim só podem ser determinados para o canal virtual com prioridade mais alta. Quando os pacotes que competem pelo recurso apresentam a mesma prioridade não há garantia rígida a nenhum dos pacotes.

A Hermes-DP, por sua vez, trabalha com um mecanismo de alocação dos recursos baseado em prioridades dinâmicas, ou seja, a prioridade de cada canal virtual varia de acordo com a do pacote que o mesmo transmite. Assim, a transmissão dos pacotes é realizada por qualquer canal virtual livre. O campo *priority* é também incluído no cabeçalho dos pacotes dessa alternativa arquitetural, porém, o campo admite um valor entre zero e $(2t - 1)$, onde t é a largura em bits de um *flit*, diferente da rede Hermes-FP que admite valores entre zero e $(n - 1)$. Consequentemente, a Hermes-DP pode diferenciar um número maior de fluxos.

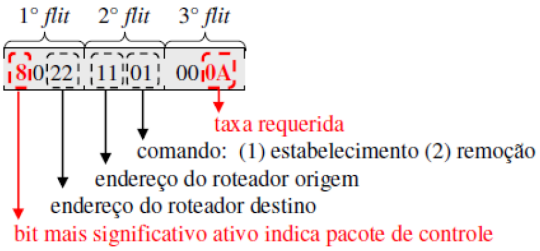
O uso de alocação dos recursos baseada em prioridades dinâmicas requer alterações na arbitragem, roteamento e escalonamento. Com relação ao escalonamento, o vetor *priority* é verificado para determinar qual canal virtual tem o pacote com maior prioridade. No caso de empate, ou seja, pacotes de mesma prioridade competirem por um mesmo canal, pode ser adotada uma política *Round-Robin*, onde os x pacotes com maior prioridade recebem $1/x$ da largura de banda do canal físico, ou, um dos pacotes de maior prioridade recebe toda a largura de banda. O uso de qualquer uma das abordagens citadas acima é definida durante a fase de projeto da rede.

O escalonamento baseado em taxas proposto na rede Hermes-RB é realizado em duas etapas: uma etapa composta pelo controle de admissão e outra que compreende um escalonamento dinâmico. A prioridade de cada fluxo é definida dinamicamente de acordo com as taxas de requisições e uso.

O controle de admissão determina se um novo fluxo pode ser admitido pela rede sem colocar em risco as garantias de desempenho dadas ao outros fluxos QoS. Essa etapa inicia com o envio de um pacote contendo a taxa requerida pelo núcleo do roteador de origem ao roteador de destino, como está indicado na Figura 15. O fluxo só é admitido se e somente se todos os roteadores no caminho podem transmitir a taxa requerida. Ao pacote de controle chegar no endereço de destino, um sinal de ACK é transmitido à origem. Se admitido, uma conexão virtual é estabelecida entre a origem e o destino. Os dados

dessa conexão, como endereço do roteador de origem, endereço do roteador de destino, taxa requerida e taxa usada, são inseridos numa tabela de fluxos para fins de roteamento. Completada a transmissão, a conexão é encerrada pelo envio de outro pacote de controle.

Figura 15 – Campos do pacote de controle da Hermes-RB.

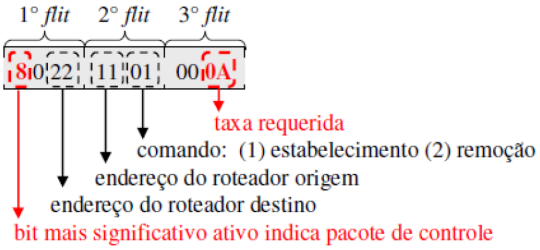


Fonte: (MELLO, 2007).

A tabela de fluxos é lida pelo escalonador com o intuito de encontrar a prioridade de cada fluxo QoS roteado para uma mesma porta de saída. A prioridade dos fluxos é atualizada periodicamente de acordo com a diferença entre a taxa requerida e a taxa usada. Uma prioridade positiva implica que o fluxo está usando menos largura de banda do que a requerida e caso negativa, o fluxo está violando a taxa admitida.

A Figura 16 ilustra os campos do pacote de dados na Hermes-RB. Quando os pacotes de dados chegam a uma porta de entrada do roteador são armazenados em *buffers*, arbitrados e roteados para uma porta de saída. Pacotes que precisam ser roteados para uma mesma porta de saída são atendidos de acordo com a política de escalonamento implementada.

Figura 16 – Campos dos pacotes de dados da Hermes-RB.



Fonte: (MELLO, 2007).

2.5.1 Propriedades para avaliação da entrega de pacotes

A rede Hermes é uma infra-estrutura de comunicação intra-chip que deriva-se em quatro redes diferentes: a Hermes-CS, baseada no escalonamento por chaveamento de circuitos; a Hermes-FP, que utiliza de escalonamento baseado em prioridades fixas; a Hermes-DP, que de forma similar à rede citada anteriormente gerencia o escalonamento

de acordo com a prioridade dos pacotes, porém nesta rede a atribuição das prioridades aos fluxos é realizada de forma dinâmica; e por fim, a rede Hermes-RB, que utiliza do escalonamento baseado em taxas.

Por tratar-se de uma infra-estrutura NoC parametrizável e escalável com diferentes versões propostas, será realizada a análise média da latência, já que as derivações da Hermes produzem diferentes resultados. O tempo de espera não apresenta variações significativas em número de ciclos de relógio entre uma versão e outra, pois as quatro versões fazem uso da política de *handshake* entre origem e destino.

Tempo de espera - como pôde ser visto durante a seção introdutória, as quatro variações da infra-estrutura Hermes usam do sinal de ACK entre os roteadores de origem e destino para iniciar ou finalizar uma conexão. Esse sinal precisa ser propagado por todo o caminho de comunicação, ou seja, está sujeito aos mesmos saltos entre roteadores que os pacotes de dados. Como esse sinal não é armazenado ou processado pelos roteadores durante a sua transmissão tanto para a requisição quanto na resposta, ou seja, é apenas um sinal de passagem, o tempo de resposta é expresso em $T_{wait;reply} = 2h$ ciclos de relógio. Por não haver reserva de recursos $T_{wait;req} = 0$.

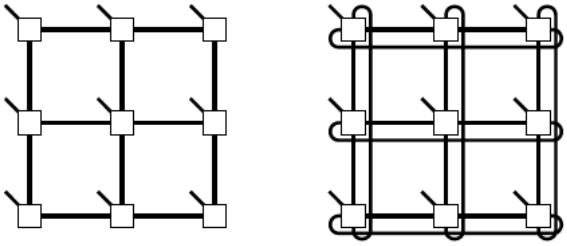
Latência - o tempo para entregar um pacote cresce linearmente na rede Hermes com o número de saltos realizados. Considerando que o *flit* de cabeçalho necessita estar armazenado no *buffer* para ser processado, esse *flit* em particular pode ser desconsiderado da análise de latência. Além disso, cada *flit* de um pacote gasta dois ciclos de relógio para ser transmitido ao próximo roteador e m ciclos durante o roteamento e arbitragem. Portanto, a latência média da infra-estrutura Hermes é dada pela equação $T_{latency} = \sum_{i=1}^{h} m_i + 2(f-1)$.

2.6 SOCIN

A rede SoCIN, desenvolvida por Zeferino e Susin (2003) no Instituto de Informática da Universidade Federal do Rio Grande do Sul e no Centro de Ciências Tecnológicas da Terra e do Mar da Universidade do Vale do Itajaí, caracteriza-se em uma arquitetura de comunicação intra-chip escalável com roteadores parametrizáveis e voltada para aqueles sistemas que apresentem maior número de tarefas com tempo real *soft*. A rede pode ser implementada em duas topologias 2D: malha ou torus, como é ilustrado na Figura 17. A topologia em malha apresenta menor custo de implementação e consumo de energia, enquanto a topologia torus reduz a latência da comunicação.

Os enlaces da SoCIN são compostos de dois canais opostos e unidirecionais. Cada canal é constituído por n bits de dados e dois bits de banda lateral: um utilizado para

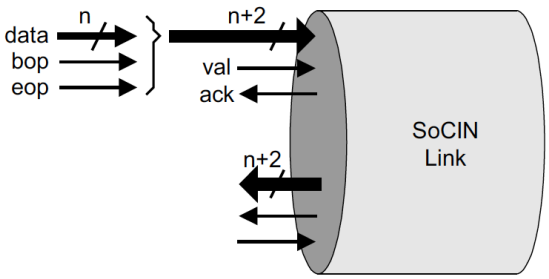
Figura 17 – Topologias implementáveis na rede SoCIN.



Fonte: (ZEFERINO; SUSIN, 2003).

caracterizar o ínicio de um pacote e outro para definir o seu fim (respectivamente, *begin-of-packet* e *end-of-packet*). Assim, a largura do canal físico é $n + 2$ e valores típicos para o parâmetro n são 8, 16 ou 32 bits, sem incluir os sinais dos protocolos de níveis mais altos, como controle de erros e paridade. Cada canal também inclui dois sinais para controle de fluxo: o sinal *val* utilizado na validação dos dados no canal e o sinal *ack* para confirmar o recebimento dos dados. Os bits para controle de fluxo não são incluídos no cálculo da largura física do canal, pois não atravessam os roteadores e não são armazenados em seus *buffers*. A Figura 18 ilustra um enlace físico na rede SoCIN.

Figura 18 – Enlace da rede SoCIN.

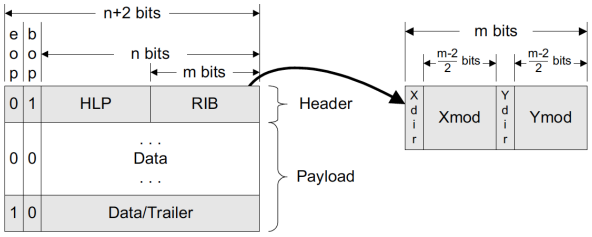


Fonte: (ZEFERINO; SUSIN, 2003).

A comunicação é baseada no modelo da passagem de mensagens, ou seja, os núcleos comunicam-se através do envio e recebimento de mensagens de solicitação e resposta. Neste modelo de comunicação existem dois tipos de núcleos: os iniciadores e alvos. Um iniciador (por exemplo, um processador) emite uma solicitação e um alvo (por exemplo, uma memória) responde. Um núcleo também pode implementar ambas as funcionalidades, ao ser um iniciador e alvo em diferentes momentos, como por exemplo, um co-processador. A conexão entre os núcleos e a rede é efetuada através da técnica de interface com canais virtuais, ou simplesmente VCI (do inglês, *Virtual Channel Interface*). A SoCIN implementa uma tabela de roteamento estático juntamente aos adaptadores de rede utilizada na configuração das informações incluídas no cabeçalho dos pacotes, ou RI (do inglês, *Routing Information*).

A comutação dos pacotes segue a abordagem *wormhole*. As mensagens são enviadas por meio de pacotes, divididos em *flits*. Um *flit* na SoCIN tem o mesmo tamanho que o canal físico, ou *phit* (do inglês, *physical unit*). O roteamento utilizado é o XY determinístico e com endereçamento na fonte. Cada remetente de pacote deve determinar o caminho a ser usado e incluir o RI correspondente no cabeçalho do pacote. O determinismo implica que dado um par remetente-receptor, o mesmo caminho será usado para os pacotes que fluem do remetente para o receptor.

Figura 19 – Formato do pacote da rede SoCIN e RI.



Fonte: (ZEFERINO; SUSIN, 2003).

O formato de um pacote da SoCIN é demonstrado na Figura 19. Como pode ser visto, o pacote inicia com um *flit* de cabeçalho e é seguido por um número ilimitado de *flits* do tipo *payload*. No cabeçalho, m bits são reservados para o RI e n - m bits podem ser usados para implementar HLP(*Higher Level Protocol*). O campo com os dados RI é especificado no protocolo da rede SoCIN e é composto pelos campos descritos na Tabela 2.

Tabela 2 – Campos reservados para o RI nos pacotes da SoCIN.

Campo	Significado
X_{dir}	0/1, pacotes devem ser roteados na direção Leste/Oeste.
X_{mod}	Número de enlaces remanescentes em X.
Y_{dir}	0/1, pacotes devem ser roteados na direção Norte/Sul.
X_{mod}	Número de enlaces remanescentes em Y.

Fonte: adaptada de Zeferino e Susin (2003).

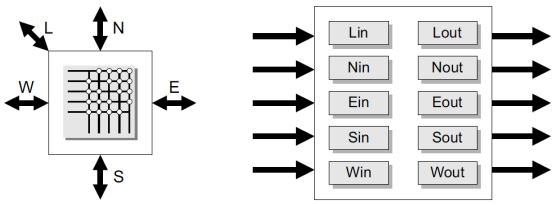
Ao injetar um pacote na rede, um núcleo iniciador obtêm as coordenadas do alvo ao verificar a tabela de roteamento. Para o caso da fonte e do destino estarem localizadas em diferentes colunas na rede, os pacotes devem sempre ser roteados primeiramente na direção X, e posteriormente na Y. No roteador, após escalonado o caminho do pacote, o circuito de roteamento atualiza o campo RI ao decrementar em uma unidade o campo *mod* (X ou Y) referente à direção selecionada. Quando o roteador recebe um pacote com X_{mod} nulo, significa que este precisa ser roteado na direção Y. Caso o campo Y_{mod} também seja nulo, então o pacote deve ser entregue ao núcleo conectado à porta local.

Na SoCIN é utilizada uma abordagem de arbitragem distribuída com um árbitro *Round-Robin* presente em cada canal de saída dos roteadores. Quando uma entrada recebe o sinal de consentimento para uso da saída, este canal recebe a menor prioridade no próximo ciclo de arbitragem. Essa abordagem visa a redução da latência com garantia de que nenhum pacote permaneça indefinidamente esperando por um canal de saída, ou seja, não ocorra casos de *starvation* na rede.

O controle de fluxo segue as premissas da técnica de *handshake* devido a sua simplicidade e baixo custo de implementação. Quando um emissor insere dados no enlace, o sinal *val* é ativado. Ao receptor estar apto para utilizar esses dados, o sinal de *ack* correspondente é ativado. Existe uma fila FIFO composta de *p-flit buffers* em cada canal de entrada para armazenamento dos *flits* que ainda não receberam o sinal de *ack*, onde p é um parâmetro que depende dos requerimentos da aplicação.

Os roteadores da SoCIN são denominados RASoC[6] e podem conter até cinco portas bidirecionais, como pode ser visto na Figura 20. As portas adjacentes são nomeadas com os pontos cardinais N (*North*), S (*South*), E (*East*) e W (*West*), além da porta diagonal L (*Local*). De acordo com a aplicação algumas portas não necessitam ser implementadas. Por exemplo, em uma rede 3x3 todas as portas do roteador central são implementadas, enquanto os roteadores nas bordas necessitam de apenas duas ou três portas. Internamente, o RASoC é composto por dois módulos: *input channel* (*in*) e *output channel* (*out*).

Figura 20 – Arquitetura do RASoC.



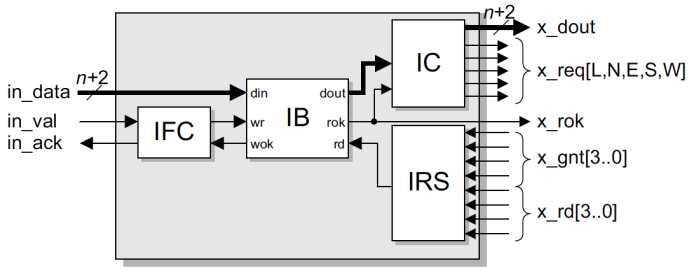
Fonte: (ZEFERINO; SUSIN, 2003).

A Figura 21 ilustra as estruturas internas do *input channel*. Esse módulo é formado por quatro componentes: o Controlador do Fluxo de Entrada (IFC), *Buffer* de Entrada (IB), Controlador de Entrada (IC) e Escalonador de Leitura da Entrada (IRS). O bloco IFC implementa a tradução entre o *handshake* e os protocolos para controle de fluxo das filas FIFO de entrada. O bloco IB é onde os *buffers* são de fato implementados. No bloco IC é performado o roteamento e, por fim, o bloco IRS faz a conexão entre o módulo *input* e *output*.

A estrutura interna do *output channel* é demonstrado na Figura 22. Esse módulo também é implementado por meio de quatro blocos lógicos: o Controlador de Saída

[6]Acrônimo para *Router Architecture for System-on-Chip* (ZEFERINO; SUSIN, 2003).

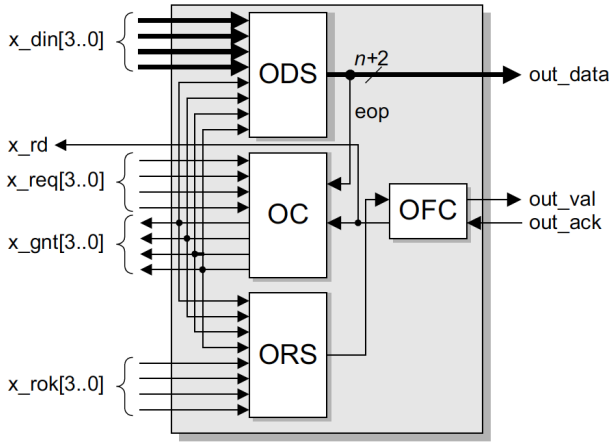
Figura 21 – Estrutura interna do *input channel*.



Fonte: (ZEFERINO; SUSIN, 2003).

(OC), Escalonador dos Dados na Saída (ODS), Escalonador de Leitura da Saída (ORS) e o Controlador do Fluxo de Saída (OFC). O bloco OC implementa o algoritmo *Round-Robin* para selecionar as requisições imitidas pelos *input channels*. Os blocos ODS e ORS trabalham em conjunto para fazer o chaveamento dos pacotes vindos da entrada aos canais de saída solicitados. O bloco OFC, por fim, é responsável por gerenciar o processo de transação entre roteador e destinatário.

Figura 22 – Estrutura interna do *output channel*.



Fonte: (ZEFERINO; SUSIN, 2003).

2.6.1 Propriedades para avaliação da entrega de pacotes

A SoCIN é uma rede escalável baseada na arquitetura de roteadores parametrizáveis para uso na síntese de NoC's customizáveis e de baixo custo. Os autores propõe o uso das topologias torus para as aplicações que necessitem de alta performance e malha 2D regular para aquelas que precisam manter o consumo de silício baixo.

Essa rede, assim como a grande maioria das NoC's aqui apresentadas, faz uso do roteamento XY estático. Zeferino e Susin (2003) apresentam o RASoC, um roteador parametrizável de cinco portas que implementa uma metodologia distribuída de arbitragem

Round-Robin, onde cada canal de saída implementa o seu próprio árbitro. Na SoCIN não existe diferenciação entre um fluxo de melhor esforço daquele que necessite de QoS, portanto, a rede não implementa nenhuma técnica para reserva de recursos ou garantia de vazão.

Tempo de espera - como não há reserva de recursos o tempo de requisição na SoCIN é nulo. Entretanto, a rede ainda faz uso da técnica de *handshake* para validação dos dados enviados. Um roteador necessita receber um ACK para efetivar a transação. O sinal de ACK não transita no mesmo nível que os dados, ou seja, é um sinal de passagem. Portanto, $T_{wait;reply} = m$, onde m é o tempo necessário para que o roteador verifique o sinal de ACK e valide a transmissão.

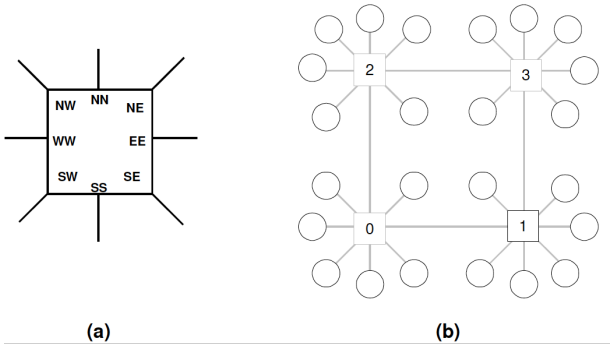
Latência - na SoCIN não há suporte QoS e fluxos de prioridades diferentes competem pelos recursos de forma igualitária. A vazão da rede não é garantida, portanto a latência aumenta proporcionalmente ao número de *flits* que compõem um pacote. Ademais, os pacotes realizam um determinado número de saltos (h) na rede. A cada salto é efetuada a verificação de disponibilidade do canal de comunicação anteriormente ao envio do pacote. O processo de chaveamento da entrada com a saída é realizado em dois ciclos de relógio. Devido a arquitetura dos roteadores da SoCIN o *flit* de cabeçalho é desconsiderado na análise. Assim, a rede SoCIN tem sua latência expressa por $T_{latency} = \sum_{i=1}^{h} m_i 2(f-1)$.

2.7 RTSNOC

A rede RTSNoC é uma rede com previsão de latência para uso em sistemas de tempo real desenvolvida por Berejuck (2015). O foco da rede são os sistemas embarcados nos quais existem mais aplicações de tempo real *soft* do que *hard* e o sistema demanda um certo grau de qualidade nos serviços dos fluxos de tempo real *soft*. A NoC proposta é baseada na intercalação dos *flits* provenientes de diferentes fluxos no mesmo meio de comunicação entre roteadores, ou seja, cada *flit* do pacote carrega as informações de roteamento. A topologia utilizada é a malha direta ortogonal 2D, como demonstra a Figura 23-b, devido sua simplicidade de implementação e consumo de silício reduzido.

Os roteadores da RTSNoC são configurados em tempo de execução para conter de cinco à oito canais para comunicação, nomeados como pontos cardinais. A Figura 23-a ilustra um roteador com oito portas. Qualquer canal de comunicação na rede RTSNoC pode ser utilizado para interconectar roteadores ou conectar o roteador a um núcleo. Sabe-se que a complexidade da rede cresce exponencialmente com o aumento do número dos canais de comunicação, porém, a abordagem proposta permite que o projetista mantenha

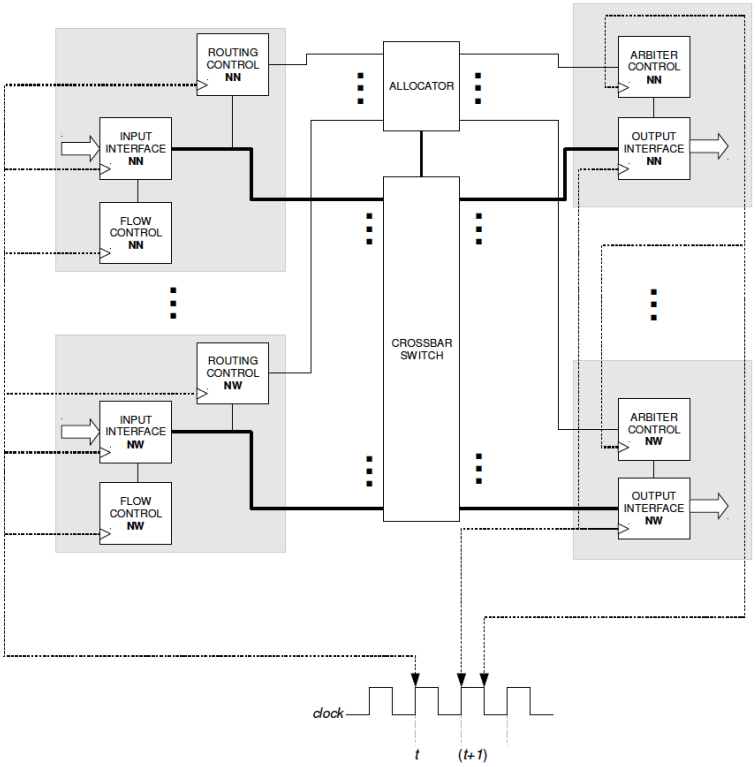
Figura 23 – Topologia da RTSNoC.



(a) Exemplo de roteador com oito portas; (b) Rede 2x2 com quatro roteadores e vinte e quatro núcleos; Fonte: (BEREJUCK; FRÖHLICH, 2016).

os núcleos que se comunicam com maior frequência no mesmo roteador. A implementação dos roteadores é concretizada por meio de sete blocos lógicos: interface de entrada, controlador de fluxo, controlador de roteamento, árbitro, alocador, matriz *crossbar* e interface de saída. O roteamento é realizado *flit-by-flit* e de forma justa entre os fluxos que competem por um recurso. Os árbitros priorizam *flits* provenientes de roteadores mais distantes e os *buffers* estão localizados apenas na interface de saída, abordagem que visa minimizar o consumo de silício. A Figura 24 mostra o diagrama de blocos que constitui um roteador da RTSNoC.

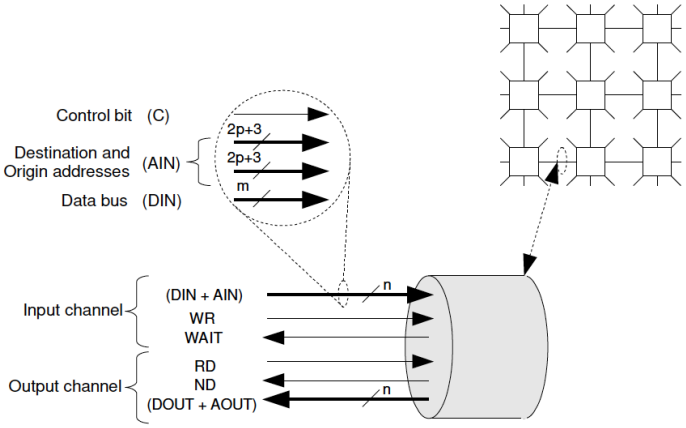
Figura 24 – Estrutura interna dos roteadores da RTSNoC.



Fonte: (BEREJUCK, 2015).

A comunicação é realizada por meio de dois canais unidirecionais, um para entrada e outro para saída de dados. Além disso, os canais também transportam os sinais de endereçamento e sincronização da transferência. O tamanho de cada canal também é configurado em tempo de execução com base nas necessidades da aplicação. O campo de dados m pode variar entre a faixa de 8-64 bits. Já o endereçamento necessita de $2p + 3$ bits, onde p determina a dimensão da rede (por exemplo, para uma rede 2x2 $p = 1$) e os três bits restantes são utilizados para a porta local. Os demais sinais apresentados na Figura 25 são usados no controle de fluxo e sincronismo da comunicação.

Figura 25 – Canais de comunicação da rede RTSNoC.

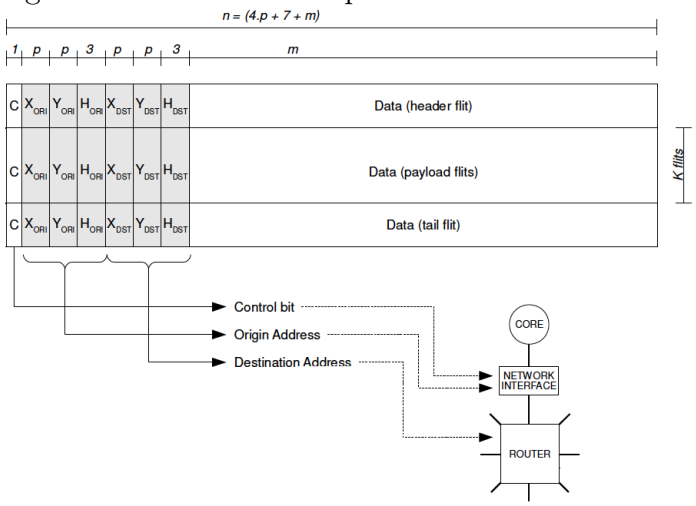


Fonte: (BEREJUCK, 2015).

O formato dos pacotes é inerente ao canal de comunicação, como ilustra a Figura 26. Cada *flit* contêm $1 + 2(2p + 3) + d$ bits, onde p é a dimensão da rede e d é o tamanho do campo de dados. O bit C é utilizado para diferenciação dos *flits* de dados daqueles que caracterizam cabeçalho ou fim de pacote. Os bits adjacentes, assim como na SoCIN, caracterizam o endereçamento: X_{ORI} e Y_{ORI} descrevem as coordenadas do roteador de origem e H_{ORI} designa a porta do roteador pela qual o *flit* foi injetado na rede. Consequentemente, X_{DST}, Y_{DST} e H_{DST} correspondem às informações de destino. Os bits remanescentes são utilizados para dados. Nos *flits* de cabeçalho ou fim de pacote o campo dos dados está a disposição para uso em HLP.

O roteamento é efetuado por meio de um algoritmo XY estático, assim como a SoCIN, devido a topologia 2D ortogonal da rede garantir que o uso deste específico algoritmo proporciona entrega ordenada e livre de *deadlock*. Como o caminho de comunicação entre dois núcleos é sempre o mesmo, os *flits* de um pacote são entregues na mesma ordem que são injetados na rede. Pacotes de diferentes fluxos que competem pelo mesmo canal têm os seus *flits* intercalados na saída correspondente por meio de um árbitro que controla o acesso ao canal. A Figura 27 compara as técnicas de intercalação dos *flits* e *wormhole*.

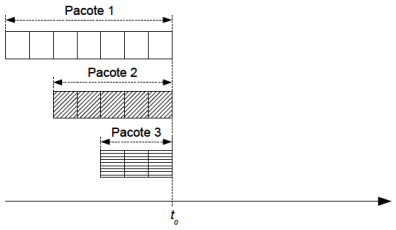
Figura 26 – Formato dos pacotes da rede RTSNoC.



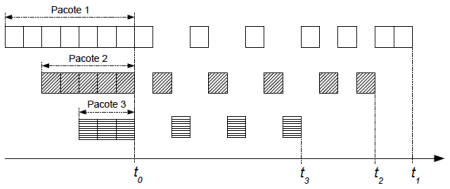
Fonte: (BEREJUCK; FRÖHLICH, 2016).

É importante notar, na seção *(b)* daquela figura, que o pacote de três *flits* é entregue antes que os demais. Esta característica da técnica de intercalação dos *flits* é o que traz garantia de QoS à RTSNoC, pois em geral, fluxos *real-time* são compostos de pacotes com poucos *flits*. De forma contrária, a técnica *wormhole* faz com que a saída permaneça escalonada enquanto o *flit* que caracteriza fim de pacote não seja recebido. Como os árbitros permitem a transmissão de apenas um *flit* a cada ciclo de relógio na RTSNoC, toda porta de saída implementa um *buffer* capaz de armazenar um único *flit*.

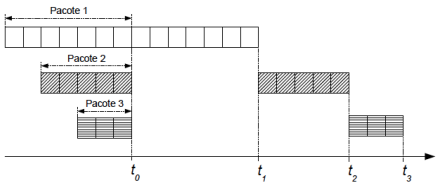
Figura 27 – Demonstração da técnica de intercalação dos *flits*.



(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.



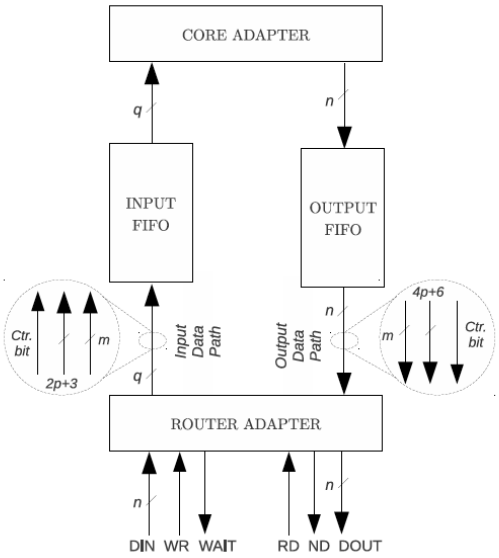
(c) Exemplo de um chaveamento *wormhole* para os mesmos três pacotes.

Em (a) são ilustrados três pacotes esperando para ser roteados. Em (b) é demonstrado o método de intercalação dos *flits* e em (c) o método *wormhole*. Fonte: (BEREJUCK; FRÖHLICH, 2016).

Os árbitros implementam um algoritmo de escalonamento. Na inicialização da rede os canais recebem diferentes níveis de prioridade. As maiores prioridades são atribuídas aos canais *NN, SS, EE* e *WW*, pois são utilizados na interconexão de roteadores em uma malha regular 2D. Além disso, esses canais podem enviar mais de um *flit* a cada sinal de *grant*. A quantidade de *flits* que um canal pode enviar a cada *grant* depende do número de requisições ocorridas ao mesmo tempo nos roteadores anteriores ao caminho solicitado. Qualquer *flit* tem sua requisição de roteamento atendida se tiver a maior prioridade ou se não houver outros pedidos no árbitro. Atendida a solicitação, o canal recebe um nível de prioridade mais baixo e pode enviar apenas outros *flits* se não houver outro *flit* na espera para ser roteado.

A Figura 28 ilustra os blocos lógicos que compõe a Interface de Rede. Essa estrutura é responsável pelo interfaceamento do núcleo com o roteador. A funcionalidade do *Router Adapter* prepara os dados provenientes da rede ao remover as informações de endereçamento do *flit*. Em contrapartida, o *Core Adapter* prepara os dados provenientes do núcleo ao concatenar os bits de controle e endereçamento com o campo de dados. Além disso, a interface implementa duas filas FIFO, uma para os dados recebidos e outra para aqueles que esperam para serem entregues. O tamanho dessas filas é estabelecida pelo projetista em tempo de execução e a vazão da rede é função direta desse parâmetro. Os roteadores necessitam que pelo menos um *flit* esteja armazenado na fila e pronto para ser transferido. As FIFO's contam com sinais de *handshake* para alertar os núcleos quando as memórias encontram-se vazias ou lotadas.

Figura 28 – Estrutura interna da Interface de Rede.



Fonte: (BEREJUCK, 2015).

2.7.1 Propriedades para avaliação da entrega de pacotes

A rede RTSNoC é uma rede voltada para sistemas de tempo real, sem reservas de recursos e com previsibilidade da latência de pior caso. Os autores adotaram como topologia a malha 2D regular e roteadores de até oito portas para comunicação e roteamento XY determinístico. A abordagem proposta é da utilização das conexões adjacentes dos roteadores não apenas para interconexão, mas também na conexão de núcleos à rede. Além disso, a arquitetura da rede não trata os protocolos de alto níveis relacionados ao interfaceamento dos núcleos, o que torna esse processo customizável e dependente da aplicação onde a rede será utilizada.

Os fluxos de diferentes prioridades que circulam na RTSNoC competem pelos mesmos recursos. Para garantir uma latência de pior caso conhecida e a vazão de fluxos que exigem QoS a rede faz uso da técnica de arbitragem distribuída, assim como na rede SoCIN, combinada com a técnica de intercalação dos *flits* no canal de saída. Ou seja, o roteamento é realizado *flit-by-flit*, o que garante a não ocorrência de *starvation*. Além disso, devido o determinismo proporcionado pelo algoritmo utilizado, *flits* de diferentes fluxos são entregues na mesma ordem de injeção à rede livres de *deadlock*.

Tempo de espera - como a rede não faz uso da reserva de recursos, ou seja, os pacotes não precisam fazer nenhum tipo de requerimento para serem transmitidos, o tempo de requisição é nulo. O controle de fluxo da RTSNoC não interfere na transmissão dos dados e os pacotes são enviados sem o uso de mensagens ACK, portanto, o tempo de resposta também é nulo para esta rede.

Latência - na RTSNoC a vazão da rede é garantida por meio da intercalação de *flits*. Como não há contenção de recursos, se N fluxos competem pelo mesmo canal de saída em um roteador, cada um de seus *flits* é enviado pelo canal solicitado a cada ciclo da arbitragem de acordo com a prioridade dada pelo árbitro. Apenas os *flits* de carga útil são considerado na análise da latência quando k pacotes competem pelo mesmo nodo de destino. Além disso, existe a possibilidade de as duas filas FIFO da interface de rede estarem completamente ocupadas tanto na origem quanto destino, portanto, o tamanho dos *buffers* dessas filas (B) é multiplicado por dois e somado à latência, o que resulta em $T_{latency} = \sum_{i=1}^{h} 2N_i + 2k(f-1) + 2B$. Cabe citar que essa equação é equivalente a latência de pior caso da RTSNoC, já que devido as características arquiteturais da rede, a latência total não é constante.

2.8 RESUMO DOS TRABALHOS RELACIONADOS

A Tabela 3 apresenta um resumo das redes estudadas no capítulo 2. Percebe-se que em sua maioria as redes fazem uso de algoritmos estáticos, com exceção da SoCBUS. Naquela rede há reserva de recursos para a entrega dos pacotes, portanto, o algoritmo apresenta um leve nível de adaptabilidade. A topologia utilizada é a malha 2D e durante a fase de reserva do canal o roteador verifica aqueles que estão livres para o estabelecimento da comunicação. O roteamento não segue premissas pré-estabelecidas o que torna-o semi-estático, já que após reservado o canal, os pacotes transitam de forma similar ao algoritmo XY puramente estático.

As redes Æthereal e SoCBUS oferecem suporte QoS através da reserva de recursos, portanto, essas redes também apresentam um tempo de espera associado à requisição destes recursos. A rede QNoC, por sua vez, utiliza uma técnica de garantia da vazão no suporte à QoS sem reservar recursos, porém mensagens de *acknowledgment* são utilizadas no roteamento e isso reflete-se na existência de um tempo para resposta associado. A rede RTSNoC, objeto de estudo deste trabalho, utiliza da intercalação de *flits* para garantir suporte à QoS sem utilizar da reserva de recursos ou mensagens de *acknowledgment*, assim, não há tempo de resposta ou requisição associados. As demais redes não oferecem garantias de suporte à QoS.

A latência de todas as redes estudadas depende de alguma forma do número de saltos realizados pelos pacotes entre a origem e o destino. Na RTSNoC em particular, a latência depende também da quantidade de fluxos competindo por um mesmo canal, o tamanho desses pacotes e a vazão na fase de interfaceamento. Em condições ideais onde os núcleos conectados à rede operam nas mesmas taxas que os roteadores pode-se atingir uma vazão de um *flit* por ciclo de *clock*, ou seja, o aumento da latência nesta rede depende de fatores externos aos componentes diretamente ligados ao roteamento.

O árbitro da RTSNoC apenas prioriza os canais de interconexão com roteadores e usa da intercalação de *flits* para controle de fluxo. Devido a arquitetura da rede os núcleos necessitam que todos os *flits* pertencentes a um pacote sejam entregues para poder iniciar as operações sobre ele. Se um pacote de tempo real tiver de disputar com outros pacotes de melhor esforço pelo mesmo canal ele não será priorizado e terá seus *flits* intercalados com os demais igualitariamente. Quanto maior for a quantidade de fluxos que competem por um mesmo canal, maior será o tempo gasto na entrega efetiva de um pacote.

O interfaceamento na maioria das redes é customizável ou segue um protocolo de comunicação *open-source*. A rede Æthereal é a única dentre as redes estudadas que usa de um protocolo estabelecido pela empresa *Philips*. Ademais, nenhuma das redes conta

com alguma forma de observação dos seus comportamentos ou mecanismos integrados que possibilitem a verificação de informações relativas à entrega de pacotes.

Legenda para a Tabela:

DTL – *Philips Device Transaction Level Protocol*

OCP – *Open-Core Protocol*

VCI – *Virtual Channel Interface*

CS – *Circuit Switching*

GT – *Guaranteed throughput*

IF – *Interleave of flits*

B – Profundidade do pipeline e tamanho dos *buffers*

f – Número de *flits*

h – Número de saltos

k – Quantidade de pacotes que competem pelos mesmos recursos da rede

N – Fluxos que competem pelos mesmos recursos da rede

n – Largura do canal físico

m – Quantidade dos ciclos de *clock* utilizados no processamento

s – Duração do *time slot*

P – Período do *time slot*

p – Número de *time slots* disponíveis para alocação

Tabela 3 – Propriedades e características das redes estudadas.

	Æthereal	**SoCBUS**	**×*pipes***	**QNoC**	**HERMES**	**SoCIN**	**RTSNoC**
Ano	2005	2004	2003	2004	2004	2003	2015
Topologia	Malha 2D	Malha 2D	Arbitrária	Malha 2D regular ou irregular	Malha 2D	Malha 2D	Malha 2D
Roteamento	*Source*	XY semi-estático	*Street Sign*	XY estático	XY estático	XY estático	XY estático
Tamanho do *flit*	32 bits	16 bits de dados + 3 bits de controle	32, 64 ou 128 bits	16 bits de dados + 10 bits de controle parametrizáveis	8 de bits dados + 2 bits de controle parametrizáveis	n de bits dados + 4 bits de controle parametrizáveis	n bits de dados + 7 bits de controle parametrizáveis
Buffering	Fila de entrada	Entrada de posição única e buffers de saída	Fila de saída virtual	Fila de entrada parametrizável e buffer de posição única na saída	Fila de entrada parametrizável	Fila de entrada parametrizável	Fila de entrada parametrizável e buffer de posição única na saída
Interface	DTL	Customizável	OCP	Customizável	OCP	VCI	Customizável
Suporte QoS	CS	CS	Não	GT	Não	Não	IF
$\boldsymbol{T_{wait;req}}$	$(P-p)s$	$4h+h$	0	0	0	0	0
$\boldsymbol{T_{wait;reply}}$	0	0	$2n+m$	fm	$2h$	m	0
$\boldsymbol{T_{latency}}$	$hB+f$	h	$hB+2n+m$	$hfm+B$	$\sum_{i=1}^{h} m_i+2(f-1)$	$\sum_{i=1}^{h} m_i 2(f-1)$	$\sum_{i=1}^{h} 2N_i+2k(f-1)+2B$

Fonte: elaborada pelo autor.

3 PROJETO LÓGICO DO MONITORAMENTO DA ENTREGA DE PACOTES

Este capítulo tratará da arquitetura adotada para monitoramento da entrega de pacotes na rede RTSNoC, uma *Network-on-Chip* com previsão de latências projetada para sistemas embarcados de tempo real que garante uma latência de pior caso para fluxos de tempo real *hard* e melhora a latência média dos fluxos de tempo real *soft*. O capítulo inicia com a implementação da rede em SystemC RTL, já que esta foi inicialmente desenvolvida em VHDL e uma das propostas deste trabalho é a de portá-la para aquela linguagem. A seção seguinte trata do desenvolvimento de uma interface para a RTSNoC, dispositivo crucial para a etapa de simulação. Em seguida, o desenvolvimento do mecanismo que irá efetuar o monitoramento da entrega de pacotes em si é tratado. O capítulo finaliza, então, com uma seção destinada ao desenvolvimento do Gerente de Rede que atuará conjuntamente com o mecanismo descrito na seção prévia.

3.1 IMPLEMENTAÇÃO DA RTSNOC EM SYSTEMC

O SystemC, segundo Bhasker (2002), é uma ferramenta baseada em C++ que estende a capacidade desta linguagem de programação orientada a objetos ao habilitá-la para modelagem e descrição de hardware. O SystemC pode ser efetivamente utilizado para descrever modelos com precisão em ciclos de relógio e proporciona o uso de única linguagem na integração de software e hardware. Ademais, cabe citar que esta é uma ferramenta *open-source* mantida pela organização OSCI (IEEE et al., 2012).

A avaliação de desempenho das Redes Intra-Chip é uma ramo que começou a ser explorado no momento em que esta arquitetura consolidou-se em meados de 2003. Inicialmente, os experimentos relativos à avaliação de desempenho eram feitos através de simulações com base em modelos VHDL, como pode ser visto em Moraes et al. (2004). Posteriormente, alguns trabalhos passaram a utilizar os ambientes em nível de sistema, com o uso de simuladores dedicados ou não-dedicados (BOLOTIN et al., 2004). A partir de 2007, o SystemC passou a ser adotado como ferramenta padrão de fato neste tipo de investigação, com redes inteiramente projetadas nesta linguagem, como pode-se observar nos trabalhos de Dehyadgari et al. (2005) e Ghaderi, Alqahtani e Bagherzadeh (2017).

Apesar do modelo VHDL da rede RTSNoC permitir simulação e síntese no mesmo ambiente, os sistemas descritos nesta linguagem podem apresentar elevado custo computacional no que diz respeito ao tempo de simulação conforme sua complexidade. Zeferino

et al. (2007) afirma que a construção de geradores e analisadores de tráfego voltados à execução nos mesmos ambientes de simulação e síntese da rede é limitada ao subconjunto do VHDL suportado pelo ambiente de desenvolvimento utilizado e, assim, restringe a generalidade dos modelos.

A necessidade de ferramentas para modelagem que permitem caracterizar o desempenho de diferentes configurações da rede em um menor tempo computacional surgiu de forma atrelada ao crescimento destas redes. Como mencionado no capítulo anterior, a RTSNoC caracteriza-se por uma rede em malha 2D com roteadores capazes de comportar até oito interconexões, o que a difere das demais redes de mesma topologia que comportam até quatro ou cinco interconexões. Esta característica obviamente aumenta sua complexidade de forma considerável. A fim de facilitar a exploração do projeto, e, ainda assim manter a precisão em nível de ciclos como também assegurar a possibilidade de síntese já provida pelo VHDL, foi realizada a modelagem do roteador da rede RTSNoC em SystemC RTL. A verificação e validação deste componente serão detalhadas no Capítulo 5 deste trabalho.

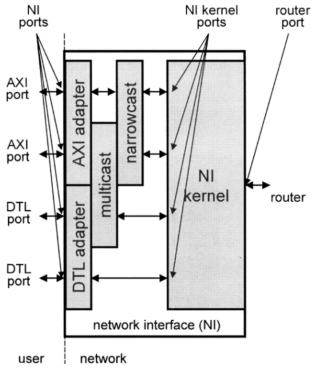
3.2 INTERFACES EM REDES INTRA-CHIP

Um componente crítico no projeto de uma Rede Intra-Chip é a Interface de Rede (do inglês *Network Interface* — NI). Este dispositivo atua como o *link* entre os núcleos e roteadores tanto do ponto de vista do *hardware* quanto de protocolo. Existe uma deficiência na literatura no que diz respeito ao projeto de uma NI, devido a necessidade de uma especificação dos núcleos que serão interconectados à rede. Contudo, dada a crescente difusão das Redes Intra-Chip e a reusabilidade de arquiteturas para núcleos já existentes, faz-se necessário um estudo mais aprofundado.

O projeto apresentado por Radulescu et al. (2005) consiste de uma NI modular que faz uso de diversos protocolos para prover reusabilidade e flexibilidade dos núcleos conectados à rede Æthereal, a qual utiliza o protocolo DTL. Esta NI é dividida em duas parte: o *kernel*, que encapsula e escalona as mensagens para posterior envio ao roteador, assim como também implementa quando necessário um sistema de cruzamento entre domínios de ciclos de relógio; o *shell*, que implementa as conexões, um sistema de controle das transações e demais operações de alto-nível específicas do protocolo que rege o núcleo. Uma biblioteca pode ser desenvolvida para realizar operações no *shell* durante a fase de síntese no chip e facilitar alterações. A Figura 29 ilustra a NI utilizada pela rede Æthereal.

Em Chouchene et al. (2011) é implementada uma técnica de disparo de relógio para reduzir o consumo de energia pela NI. Este componente é projetado para atuar como

Figura 29 – Interface da rede Æthereal.



Fonte: (RADULESCU et al., 2005).

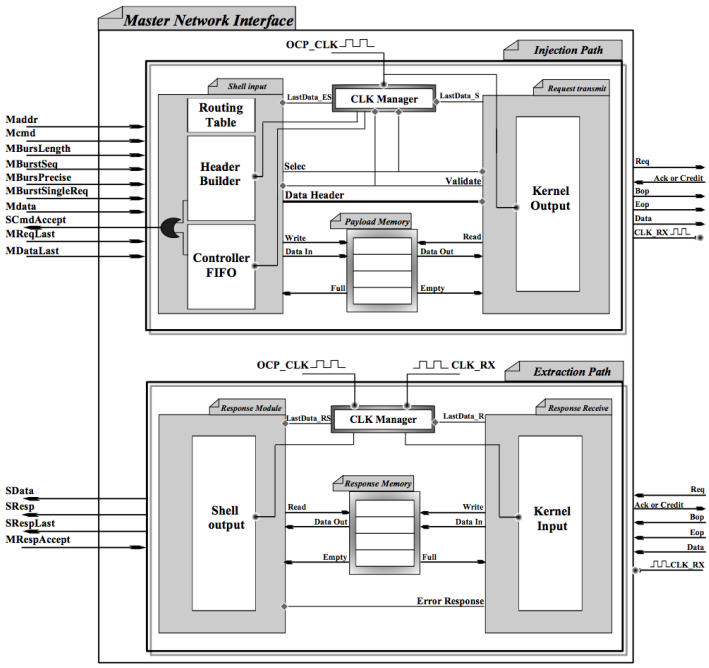
ponte entre uma interface OCP e a NoC, como é o caso da rede ×*pipes*. Dois sub-módulos compõe a NI, como demonstra a Figura 30, um para as requisições e outro para respostas. No módulo de requisição existe um *shell* de entrada, o qual constrói o cabeçalho do pacote e a tabela de roteamento naquela NoC. A saída, por sua vez, apresenta um *kernel* que injeta os *flits* na rede. Já o módulo de resposta apresenta um *kernel* na entrada que recebe os *flits* e remonta o pacote em um *shell* na saída. Além destes módulos, existe um terceiro módulo capaz de transferir ou interromper o sinal de relógio das entradas, a fim de desligar parte da NI e economizar energia. Como resultado, a arquitetura adotada de fato reduz significativamente o consumo de energia, porém, o sistema consome uma maior área de silício e as frequências alcançáveis são baixas.

O dispositivo proposto por Swaminathan et al. (2013) trata-se uma arquitetura que visa o baixo consumo de energia por meio de um barramento *WISHBONE* e apto a operar entre diferentes domínios de clock sem perda de informações. Os pacotes provenientes do roteador são armazenados em uma fila FIFO assíncrona, desempacotados e transmitidos através do barramento *WISHBONE* para os núcleos. A injeção de pacotes na rede segue o mesmo processo de forma inversa. A economia de energia se dá por um controlador de configuração, como ilustra a Figura 31, que ativa e desativa as FIFO's com base no tráfego da rede e os requisitos de consumo de energia.

Em Bhojwani e Mahapatra (2003) é apresentado um estudo de esquemas para empacotamento em duas diferentes formas: uma via software; uma em nível de hardware no núcleo. O esquema via software introduziu uma alta latência ao NoC, porém sem sobrecarga da área de silício. O esquema em nível de hardware apresentou alto consumo da área de silício e baixa latência, contudo, alterações nos parâmetros do núcleo inviabilizam seu reuso.

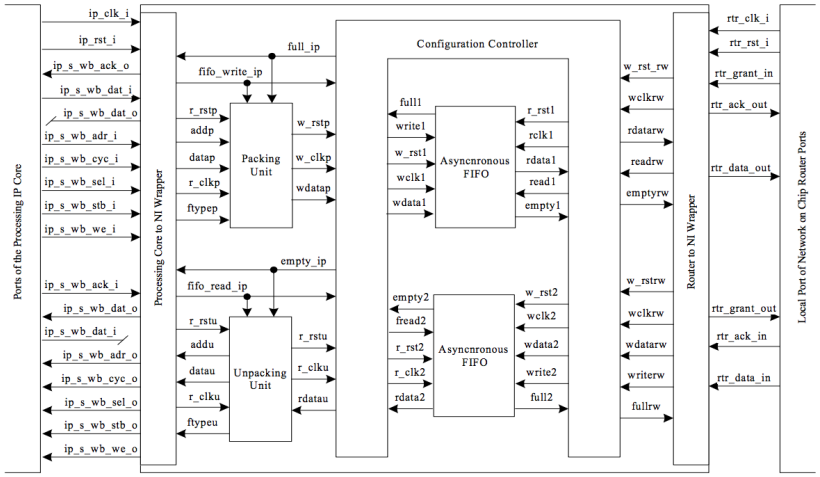
A interface proposta por Berejuck (2015) para a RTSNoC é composta por quatro

Figura 30 – Interface de rede para núcleos com base no protocolo OCP.



Fonte: (CHOUCHENE et al., 2011).

Figura 31 – Interface de rede com a arquitetura WISHBONE.

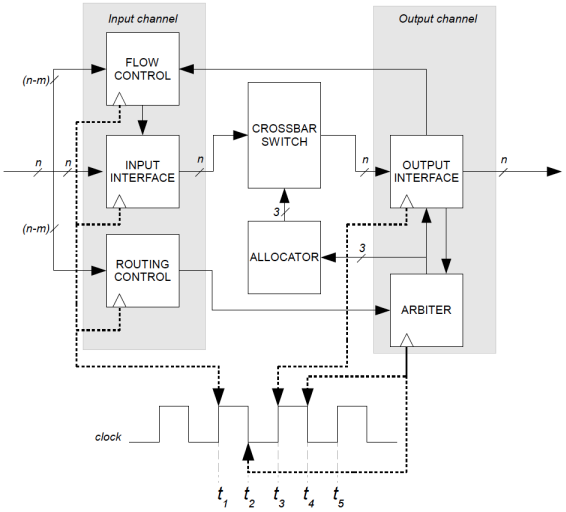


Fonte: (SWAMINATHAN et al., 2013).

componentes: um adaptador de núcleo, uma fila FIFO para o canal de entrada, uma para o canal de saída e um adaptador de roteador, conforme ilustra a Figura 28 da Seção 2.7. Esta foi a arquitetura adotada para as interfaces de rede utilizadas nos experimentos deste trabalho. Além daqueles componentes, no desenvolvimento deste projeto verificou-se a necessidade de inserir um registrador ativado por uma borda negativa do sinal de *clock* entre o canal de saída da interface e a entrada no roteador, devido o método de sincronização utilizado pelos componentes internos dos roteadores da rede RTSNoC, ilustrados

pela Figura 32.

Figura 32 – Sincronização dos componentes internos do roteador RTSNoC com o sinal de *clock*.

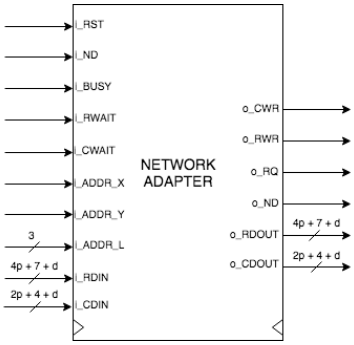


Fonte: (BEREJUCK, 2015).

Um sinal de *clock* foi representado na base da Figura 32. No instante t_1 todos os três componentes do canal de entrada são ativados pela borda positiva do *clock*. No próximo instante, t_2, o árbitro recebe as requisições dos canais de entrada e verifica se o canal de saída está disponível para receber um novo *flit*. Se ele estiver livre, o árbitro aplica as prioridades vigentes naquele momento, envia o sinal de controle para o Alocador da *Crossbar* e informa o canal de saída que um dado está pronto para ser entregue a ele. Percebe-se que os sinais de alerta são emitidos pelo árbitro nas bordas negativas do sinal de *clock*, portanto, para que exista sincronicidade entre as operações do roteador e seu adaptador, o *flit* é armazenado no registrador ativado por uma borda negativa do sinal de *clock* antes de ser encaminhado ao roteador. 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 a ele que um novo *flit* está disponível para entrega. Por fim, no instante t_4 o árbitro faz a atualização dos níveis de prioridade (BEREJUCK, 2015). O sinal de leitura, descrito no instante t_5 da Figura 32, dá-se em uma borda positiva do sinal de *clock*, o que justifica a não inserção de um registrador no canal de saída do roteador e entrada em um Adaptador de Núcleo.

A Figura 28 não ilustra nenhum sinal de entrada ou saída do Adaptador de Núcleo no que diz respeito à conexão e interface com o núcleo propriamente dito, pois aquele componente necessita das especificações do protocolo da aplicação. Para o desenvolvimento deste trabalho, assumiu-se que os núcleos já preparam os *flits* com o endereço de destino, os dados, e a sinalização de início e fim de pacote, sendo papel do Adaptador de Núcleo

Figura 33 – Diagrama de blocos da *Network Interface* para a RTSNoC.



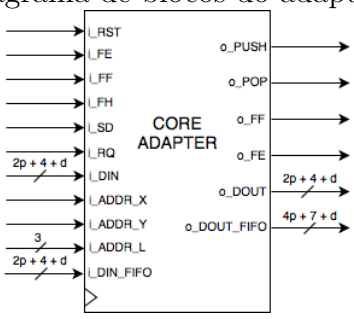
Fonte: elaborada pelo autor.

concatenar estas informações com o endereço de origem e inserir os dados referentes ao protocolo da rede no campo de dados dos *flits* de cabeçalho. A sincronização de leitura e escrita com o núcleo segue os mesmos padrões e utiliza de sinalização similar àquela adotada pelos roteadores, conforme ilustra a Figura 33. O sinal de entrada *i_ADDR_L* possui tamanho 3 naquela figura, pois representa o endereço local da interface. O campo p nos sinais de dados refere-se às dimensões da rede e o campo d a largura do campo de dados. Os valores 4 e 7, também nos sinais de entrada e saída de dados, representam a soma do bit de controle com o endereço local. As seções seguintes discutem com maiores detalhes o desenvolvimento de cada componente que compõe a interface de rede.

3.2.1 Adaptador de Núcleo

O Adaptador de Núcleo é um bloco lógico que prepara os dados vindos do núcleo para serem escritos na rede ao concatenar os campos referentes ao bit de controle, endereço de origem, destino e os dados em si para gerar o *flit* que será transmitido à rede. Além disso, este bloco lógico lê da FIFO de entrada o *flit* para aquele núcleo ao qual está conectado e escreve-o no canal de saída.

Figura 34 – Diagrama de blocos do adaptador de núcleo.



Fonte: elaborada pelo autor.

O sinal *i_SD*, ilustrado na Figura 34, indica para o Adaptador de Núcleo quando um

flit está pronto para ser enviado. Se ativo, o adaptador então verifica o estado do sinal *i_FF* proveniente da FIFO de saída, o qual indica se a fila está cheia ou se pode receber aqueles dados. Caso o sinal *i_FF* esteja de fato ativo, o sinal *o_FF* também é ativado, indicando ao núcleo que a fila está cheia e é necessário aguardar a mudança de estado. Caso contrário, o adaptador recebe aquele *flit* e procede com a operação de concatenação e escrita no canal de saída para a FIFO. Após, o sinal *o_PUSH* é ativado por um ciclo de *clock* para indicar à FIFO a escrita dos dados. O sinal *i_FH* indica quando a fila atinge um limiar, para o desenvolvimento deste trabalho este sinal não será utilizado, porém há a possibilidade de utilizá-lo conjuntamente com o sinal *i_FF* ou até mesmo substituir este sinal.

O processo de leitura ocorre similarmente à escrita, com o sinal *i_RQ* da Figura 34 indicando quando o núcleo está pronto para receber um *flit*. O estado da FIFO é apontado pelo sinal *i_FE*, assim como no processo de escrita este sinal também pode ser utilizado de forma conjunta ou substituído pelo sinal *i_FH*. Caso a FIFO esteja vazia, *o_FE* sinaliza ao núcleo que não há dados para serem encaminhados. Se *i_FE* está desativado, isso indica que existem dados e, portanto, o adaptador ativa *i_POP* por um ciclo de *clock* para ler da FIFO e faz o chaveamento do canal de entrada da FIFO com o de saída para o núcleo.

Os sinais *i_DIN* e *o_DOUT* são referentes aos canais de entrada e saída do núcleo, respectivamente. Estes sinais são de tamanho $(2p + 4 + d)$, pois o adaptador é quem escreve no *flit* o endereço de origem lido dos sinais *i_ADDR_X*, *i_ADDR_Y* e *i_ADDR_L*. Por este mesmo motivo, o sinal *o_DOUT_FIFO* tem tamanho $(4p + 7 + d)$. Já o sinal *i_DIN_FIFO* é de largura reduzida, pois o endereço de destino é removido do *flit* ao deixar a rede.

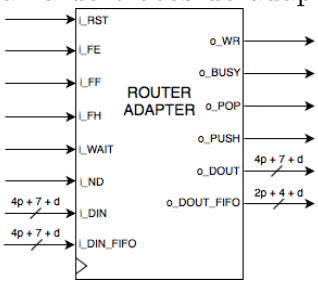
3.2.2 Adaptador de Roteador

O Adaptador de Roteador é o bloco que interage com a rede e trabalha com os sinais do canal de comunicação físico apresentados na Figura 25 da Seção 2.7. Ele prepara os dados vindos da rede para serem entregues ao núcleo e remove do *flit* os campos de endereço de destino, pois esta é uma informação utilizada apenas pelos roteadores.

A Figura 35 ilustra o diagrama de blocos deste componente. Assim como no Adaptador de Núcleo, os sinais *i_FF*, *i_FE*, *i_FH*, *o_PUSH* e *o_POP* são utilizados para indicar o estados das filas FIFO e operacionalizar suas leituras e escritas. Também de forma similar os sinais *i_DIN*, *o_DOUT*, *i_DIN_FIFO* e *o_DOUT_FIFO* são os canais de entrada e saída do roteador e das FIFO's.

O sinal *i_WAIT* indica quando o roteador pode ou não receber um *flit*. Se ativo, o

Figura 35 – Diagrama de blocos do adaptador de roteador.



Fonte: elaborada pelo autor.

roteador está ocupado e, portanto, o sinal *o_POP* permanece desativado. Caso contrário, o Adaptador de Roteador verifica se *i_FE* também está desativado, o qual indica que a FIFO de saída não está vazia. Consequentemente, se há um *flit* para ser encaminhado e o roteador não está ocupado, este é lido da fila, escrito no canal de saída e o sinal *o_WR* é ativado por um ciclo de *clock*.

A leitura de um *flit* dá-se no momento que ocorre uma transição de negativo para positivo no sinal *i_ND*. Neste momento, o adaptador verifica se a FIFO de saída não está cheia através do sinal *i_FF*. Caso a FIFO não possa receber o *flit*, *o_BUSY* é ativado. Caso contrário, os campos referentes ao endereço de destino são removidos e o *flit* é escrito na fila ao ativar o sinal *o_POP* por um ciclo de *clock*. No Adaptador de Roteador o canal de saída para a FIFO é o único que apresenta largura reduzida dos demais.

3.2.3 Filas de Entrada, Saída e Registrador

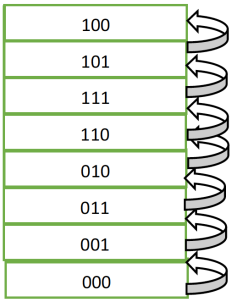
A escolha da arquitetura adequada das memórias para interfaces de redes ou demais sistemas de comunicação são pontos críticos do projeto. Em relação às filas FIFO, estas podem ser síncronas ou assíncronas. Segundo Cummings (2001), diz-se FIFO síncrona quando as operações de leitura e escrita são realizadas sobre o mesmo domínio de *clock* e assíncrona quando em domínios distintos.

As redes NoC são sistemas que trabalham com sinais de *clock* em altas frequências. Os núcleos, em contrapartida, geralmente trabalham com sinais de frequências menores, o que caracteriza domínios de *clock* distintos, porém ocasiona um problema de compatibilidade na rede. Portanto, as filas FIFO implementadas neste trabalho são assíncronas.

O modelo apresentado por Prashant (2013) para o projeto de uma FIFO assíncrona utiliza um componente denominado *Gray-Counter* para controle do incremento e decremento dos endereços da memória, assim como na lógica dos indicadores de estado da fila. Este componente foi projetado por Frank (1953) e trata-se de uma máquina de estados do tipo Moore que transita sequencialmente entre os códigos de Gray, conforme ilustra a

Figura 36. Embora seja a abordagem mais utilizada no projeto de memórias deste tipo, o número de posições disponíveis para escrita na memória é diretamente proporcional à profundidade deste contador, que devido suas características arquiteturais não ultrapassa vinte e três posições. Outra desvantagem é que são necessários dois ciclos de *clock* para realizar uma operação e atualizar os sinalizadores de estado da memória. Além disso, uma FIFO que utiliza esta abordagem está restrita às aplicações puramente assíncronas.

Figura 36 – Sequência dos códigos de Gray.

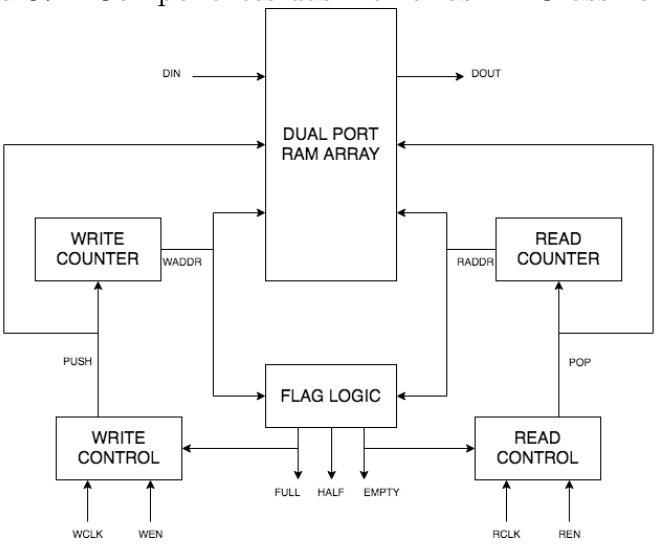


Fonte: (PRASHANT, 2013).

O modelo descrito por Cummings (2001) trata-se de uma FIFO que pode ser utilizada tanto no modo síncrono como assíncrono sem a necessidade de alterar parâmetros ou configurações, pois a lógica utilizada torna a FIFO auto-adaptativa. O conceito principal desta arquitetura é tratar as operações de leitura e escrita de forma paralela. Os contadores utilizados para incrementar o endereço de memória são síncronos, entretanto, o componente responsável por verificar e atribuir o estado da FIFO não depende dos sinais de *clock* inseridos. A Figura 37 ilustra os componentes internos de uma FIFO que utiliza esta abordagem. Naquela figura, o componente *Flag Logic* efetua a diferença entre os endereços de leitura e escrita da memória e ativa ou desativa os sinais para indicar os estados: vazio quando o resultado da diferença é nulo; cheio se igual à profundidade da memória; quase cheio/vazio se igual ao limiar pré-estabelecido. As vantagens desta abordagem é que não há restrições quanto à profundidade da fila, a atualização dos sinais é concorrente à operação efetuada e, por fim, há flexibilidade para transitar entre as formas síncrona e assíncrona. Contudo, a enésima posição e a de endereço zero são considerados valores proibidos, portanto, não são utilizados nas operações de leitura ou escrita, detalhe o qual deve ser levado em consideração no dimensionamento do projeto.

As filas FIFO implementadas neste trabalho são ilustradas na Figura 38 e seguem a abordagem apresentada na Figura 37, devido as vantagens citadas anteriormente. É importante notar que a largura dos canais da *Input FIFO* é reduzida, pois esta é conectada ao canal de saída do Adaptador de Roteador, componente que remove do *flit* os campos referentes ao endereço de destino. A *Output FIFO*, por sua vez, têm canais com largura

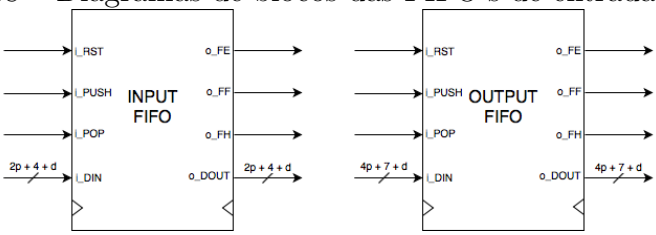
Figura 37 – Componentes das memórias FIFO assíncronas.



Fonte: adaptada de (CUMMINGS, 2001).

de um *flit*.

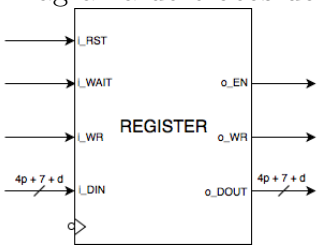
Figura 38 – Diagramas de blocos das FIFO's de entrada e saída.



Fonte: elaborada pelo autor.

O Registrador citado na Seção 3.2 é ilustrado na Figura 39. Como mencionado naquela seção, este registrador é ativado pela borda negativa do sinal de *clock* e está localizado entre o canal de saída do Adaptador de Roteador e o canal de entrada do roteador em si. Portanto, a largura dos canais de dados deste componente é igual ao tamanho de um *flit*.

Figura 39 – Diagrama de blocos do registrador.



Fonte: elaborada pelo autor.

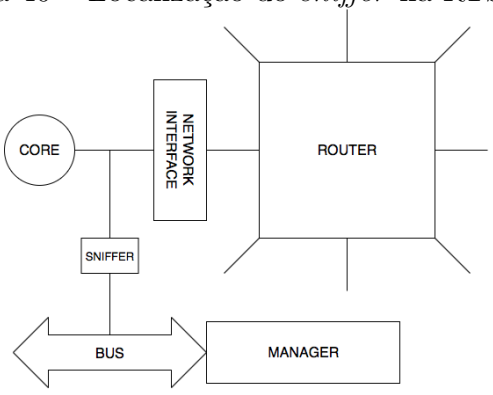
3.3 MECANISMO DE MONITORAMENTO

A principal contribuição deste trabalho é introduzir à RTSNoC um sistema apto a observar o comportamento da rede e verificar a entrega de pacotes em tempo de execução, sem reduzir a performance. Ciordas et al. (2004) afirma que a observabilidade computacional é um campo de grande relevância no meio científico, entretanto, as pesquisas atuais sobre as redes intra-chip tendem a focar apenas no design, análise e uso destas redes. Ademais, os trabalhos que exploram algum tipo de sistema para monitoramento de uma NoC utilizam, em sua maioria, abordagens intrusivas que resultam na queda da performance.

A presente seção descreve o desenvolvimento do mecanismo de monitoramento denominado como *sniffer* no Capítulo 1 deste trabalho. Este dispositivo é responsável por escutar o canal dos dados que são injetados pela rede no núcleo, através de meios não-intrusivos como os apresentados por Vermeulen, Oostdijk e Bouwman (2001).

Os núcleos de um SoC podem ser classificados em três categorias: emissores, receptores e híbridos (ZEFERINO, 2003). Ao tratar de um sistema que irá monitorar a entrega de pacotes em uma rede, é intuitivo afirmar que serão seus alvos aqueles núcleos receptores ou então híbridos, que tanto emitem quanto recebem informações da rede. Na Figura 40 é explicitada a conexão de um *sniffer*. Este dispositivo faz a captura dos sinais do canal de entrada do núcleo e saída da Interface de Rede, abordagem que visa garantir acuracidade na mensura dos parâmetros.

Figura 40 – Localização do *sniffer* na RTSNoC.

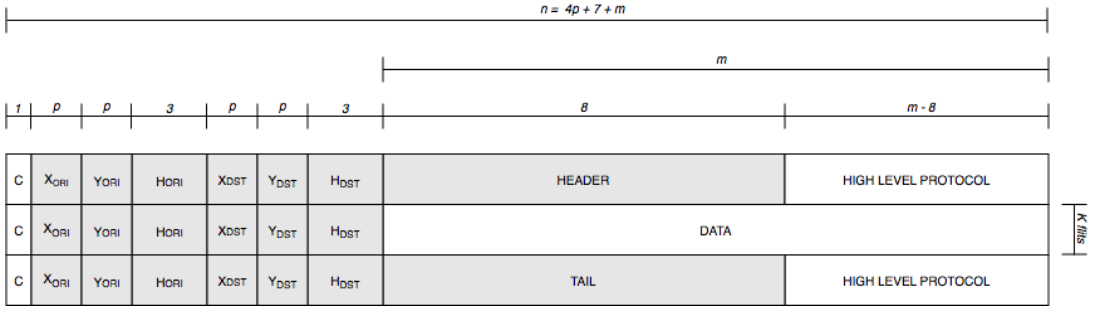


Fonte: elaborada pelo autor.

Os *flits* de cabeçalho da RTSNoC não são considerados de carga útil para o núcleo e podem ter o seu campo de dados utilizado por protocolos de alto nível. Assim, para caracterizar o início e fim de um pacote, além do bit de controle ativo foi introduzida a técnica de *byte stuffing*, que consiste em inserir palavras reservadas ou valores ilegais em uma sequência longa de bits para atuarem como delimitadores (CHESHIRE; BAKER,

1999). Nos pacotes da RTSNoC, esses valores são inseridos no *byte* mais significativo do campo de dados dos *flits* de cabeçalho, onde o valor $0xF0$ em hexadecimal corresponde ao início do pacote e o valor $0xFF$ ao fim de pacote, ou *header* e *tail*, como demonstra a Figura 41. Nas aplicações onde o campo de *payload* é maior que um *byte*, os demais bits não utilizados pelo protocolo de monitoramento da rede podem ainda serem destinados a outros protocolos de alto nível.

Figura 41 – Modelo dos pacotes da RTSNoC após introdução da técnica *byte stuffing*.



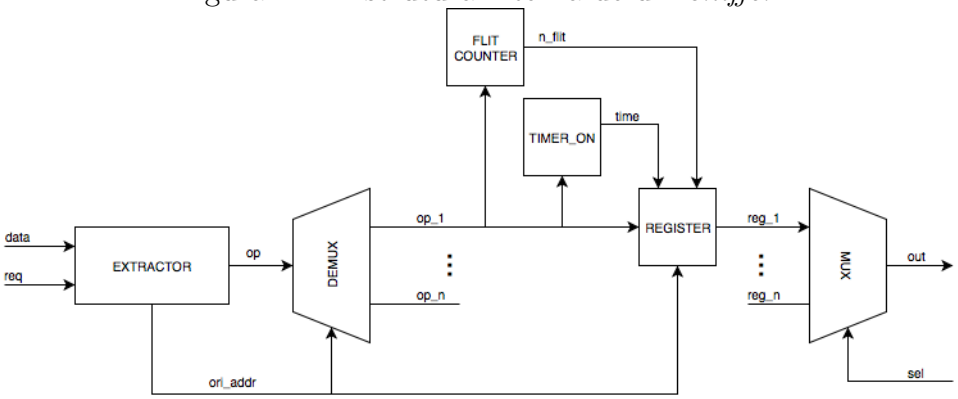
Fonte: elaborada pelo autor.

O *sniffer* é composto por cinco componentes: Extrator, Demultiplexador, Contadores de *Flits*, *Timers*, Registradores e um Multiplexador, ilustrados na Figura 42. Quando o núcleo faz uma nova requisição à Interface de rede, o Extrator também recebe este sinal e passa a "escutar" o canal de dados e lê do *flit* seu endereço de origem e o bit de controle. Caso seja um *flit* de cabeçalho o Extrator verifica, também, se é início ou fim de pacote pelo campo de dados. O endereço de origem, internamente, passa a ser o sinal de seleção do Demultiplexador e uma das entradas do Registrador.

Os pacotes na RTSNoC não são entregues de forma sequencial, como discutido na Seção 2.7. Desta forma, cada fluxo da rede necessita ser tratado separadamente, ou seja, o *sniffer* conectado à um núcleo destinatário qualquer deve levar em consideração que *flits* com diferentes endereços de origem podem ser intercalados, mesmo que o núcleo não tenha recebido aquele que sinaliza fim do pacote. Portanto, o total de saídas do Demultiplexador e consequentemente de entradas no Multiplexador, assim como a quantidade de Contadores de *Flits*, *Timers* e Registradores necessários é diretamente proporcional ao total de endereços de origem válidos na rede. Os núcleos que são exclusivamente emissores não são considerados no dimensionamento do *sniffer*, pois estes indivíduos compõem o conjunto de endereço não válidos.

Os valores lidos no sinal de requisição, no bit de controle do *flit* e no campo de *byte stuffing* são a base do Extrator para inferir a entrada do Demultiplexador. Este sinal caracteriza a operação que será realizada pelo Contador de *Flits*, *Timer* e Registrador daqueles fluxo. A tabela 4 relaciona os estados de cada um desses componentes com

Figura 42 – Estrutura interna de um *sniffer*.



Fonte: elaborada pelo autor.

o valor da entrada *op*: *0b00* é atribuído quando o sinal de requisição está desativado, independente do que for lido do canal de dados; *0b01* é atribuído quando o sinal de requisição está ativo e o bit de controle é nulo; *0b10* é atribuído quando o sinal de requisição está ativo, o bit de controle é um e o campo *byte stuffing* caracteriza início de pacote; por fim, *0b11* é o valor atribuído para *op* quando o sinal de requisição está ativo, o bit de controle é um e o campo *byte stuffing* caracteriza fim de pacote.

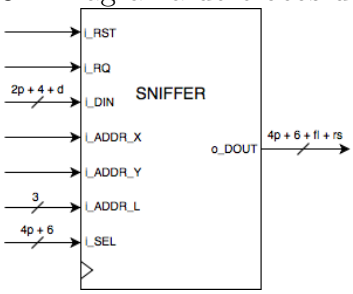
Tabela 4 – Estado dos componentes internos de um *sniffer* com relação à entrada *op*.

Op	*Flit counter*	*Timer*	*Register*
0b00	*Idle*	*Increment*	*Idle*
0b01	*Increment*	*Increment*	*Idle*
0b10	*Reset*	*Reset*	*Idle*
0b11	*Register*	*Register*	*Save*

Fonte: elaborada pelo autor.

O diagrama de blocos do *sniffer* é demonstrado na Figura 43. Além dos sinais já citados anteriormente, este mecanismo recebe o mesmo endereço que o núcleo ao qual está conectado. O sinal *i_SEL* representado naquela figura com largura $4p + 6$, trata-se do sinal de seleção da saída do Multiplexador somado ao endereço do próprio *sniffer*, já que este responde ao *Manager* ilustrado na Figura 40 por demanda. A saída *o_DOUT* é de tamanho $4p + 6 + fl + rs$, onde fl é o número de *flits* com carga útil do último pacote lido e rs o tempo gasto em ciclos de *clock* para efetuar a entrega. O tamanho de ambos os campos citados anteriormente, fl e rs, são parâmetros definidos de acordo com as características da rede.

Figura 43 – Diagrama de blocos do *sniffer*.

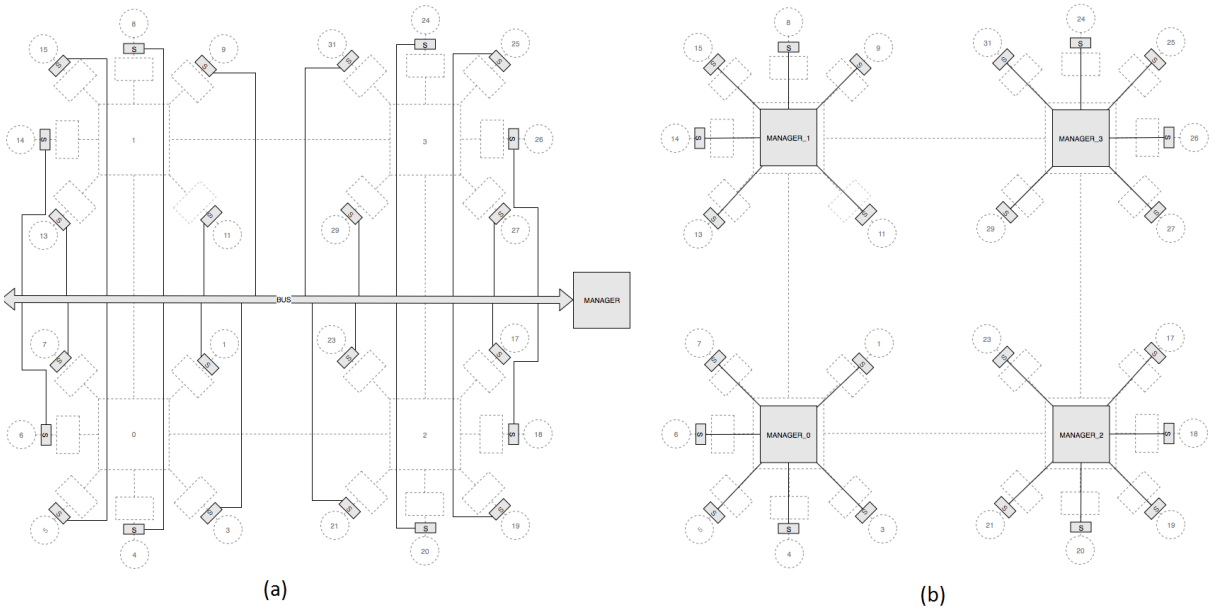


Fonte: elaborada pelo autor.

3.4 GERENTE DE REDE

Os sistemas de monitoramento atuais utilizam de recursos da própria NoC para transporte e avaliação das informações obtidas. Este trabalho procura uma solução que exima esta função dos roteadores ao dividir a rede em camadas de serviços, similarmente ao modelo OSI . O *Sniffer*, apresentado na seção anterior, é responsável por observar o comportamento da rede e alimentar um segundo componente denominado Gerente de Rede, ou *Manager*, com informações sobre o tráfego que a circula.

Figura 44 – Topologia de comunicação entre *sniffers* e *manager*

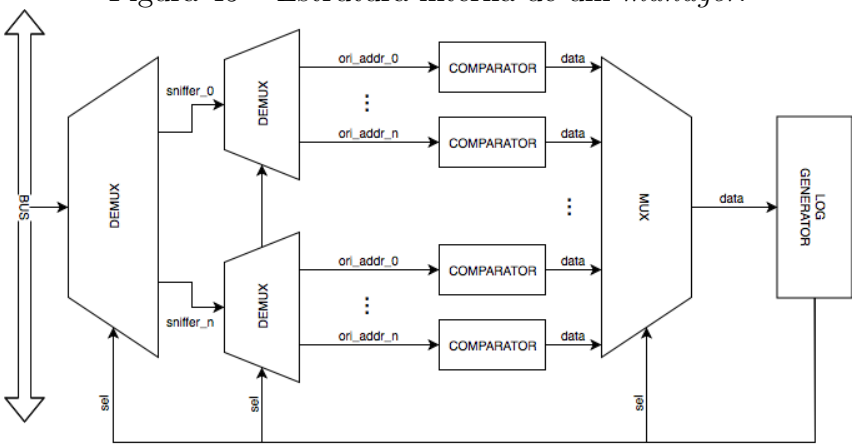


Em (a) é exemplificado o modelo centralizado e em (b) o descentralizado. Fonte: elaborada pelo autor.

O *Manager* comunica-se com os *sniffers* por meio de barramentos dedicados. Essa abordagem visa transportar através de caminhos externos as informações geradas por aqueles componentes. Assim, o serviço de monitoramento não utiliza dos recursos da rede e suas propriedades são mantidas inalteradas. A topologia de comunicação entre *sniffer* e *manager* pode ser centralizada, quando todos os *sniffers* da rede comunicam-se

com um único *manager*, ou descentralizada (CIORDAS et al., 2004). A Figura 44 fornece um exemplo das diferenças entre ambas as topologias. É importante destacar que no caso particular da topologia centralizada que utiliza de um barramento único para comunicação a complexidade do sistema aumenta de forma proporcional ao tamanho da rede, ou seja, esta topologia é indicada apenas para redes pequenas, com poucos núcleos. Na topologia descentralizada cada *manager* é responsável pelos *sniffers* de um único roteador. Ambas as topologias apresentadas podem, ainda, ser combinadas de acordo com as necessidades da aplicação e formar diversas outras topologias.

Figura 45 – Estrutura interna de um *manager*.

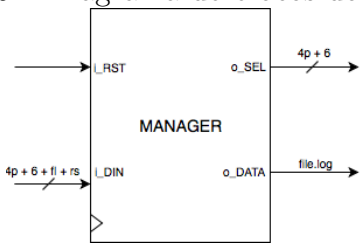


Fonte: elaborada pelo autor.

A Figura 45 ilustra os componentes internos de um *manager*, composto basicamente por quatro diferentes estruturas: comparadores, demultiplexadores, um gerador de *log* e um multiplexador. Naquela figura, o demultiplexador principal é conectado diretamente ao barramento de comunicação com o *Manager* e faz o chaveamento dos dados provenientes de cada *sniffer*. Este demultiplexador só faz-se necessário quando a topologia de comunicação não é descentralizada. Um segundo nível de demultiplexadores faz o chaveamento dos dados do *sniffer* de acordo com o endereço de origem daquele fluxo. Novamente, a inserção ou não destes demultiplexadores, assim como a quantidade de saídas é parametrizável de acordo com a aplicação. Os dados passam por um comparador antes de serem encaminhados ao gerador de *log*, para evistar a repetição de informações em fluxos homogêneos. Por fim, gerador de *log* é responsável por receber as informações e armazená-las. Este último componente pode ser tanto on-chip quanto off-chip.

O diagrama de blocos do *Manager* é demonstrado na Figura 46. O sinal *i_DIN* trata-se da porta de entrada dos dados transmitidos pelos *sniffers*. Para o caso de uma topologia descentralizada, o número de entradas é igual a quantidade de *sniffers* que irão comunicar-se com aquele *manager*. Como a entrega de dados pelo *sniffer* é via demanda, o

Figura 46 – Diagrama de blocos do *manager*.



Fonte: elaborada pelo autor.

sinal *o_SEL* é utilizado para seleção do mecanismo que irá comunicar-se com o *manager* e sua largura varia de acordo com a aplicação. Naquela figura, o sinal *o_DOUT* foi inserido apenas para representar o arquivo *log* gerado.

4 GERADORES DE TRÁFEGO

A geração de tráfego objetiva definir estruturas coordenadas para transmissão de dados entre iniciadores e destinos. Através destas estruturas é que são caracterizadas as aplicações que venham fazer uso de uma determinada rede. Este Capítulo visa introduzir conceitos relativos aos padrões utilizados nos experimentos. A primeira seção discute os conceitos relativos à distribuição espacial dos geradores de tráfego. A seção seguinte trata da carga oferecida, ou seja, a taxa de injeção dos dados. Por fim, uma terceira seção aborda a modelagem dos geradores de tráfego na rede RTSNoC.

4.1 DISTRIBUIÇÃO ESPACIAL

Em padrões de tráfego são especificados os pontos da rede que irão efetuar uma comunicação, ou seja, a fonte e o destino. Dito isso, Tedesco (2005) apresenta dois diferentes tipos de localidade: espacial e temporal. Quando um nodo apresenta maior probabilidade de enviar menagens àqueles que já foram escolhidos como destinatários do que os demais, independentemente de esses nodos estarem ou não próximos, diz-se que a rede apresenta localidade temporal. Em contrapartida, se a distância média entre nodos é menor do que a observada em um tráfego uniforme em que todos os nodos têm a mesma probabilidade de ser destino, ou seja, um nodo fonte provavelmente irá se comunicar com um de seus vizinhos, diz-se localidade espacial.

Segundo Zeferino et al. (2007), os padrões para distribuição de tráfego existentes podem ser divididos, ainda, em dois subconjuntos: os padrões com múltiplos destinatários definidos ao longo da execução e os de único destinatário já definidos previamente ao início da execução. Ambos os subconjuntos citados serão discutidos nas subseções seguintes.

4.1.1 Padrões com múltiplos destinatários

Os padrões de múltiplos destinatários são aqueles onde não existe um destino previamente especificado para cada pacote gerado por um núcleo, logo, esses padrões exibem um baixo grau de localidade espacial e uma localidade temporal variável ao decorrer da execução. Segundo Duato, Yalamanchili e Ni (2003), Tedesco (2005) e Zeferino et al. (2007) são eles:

- Uniforme: todos os nodos têm a mesma probabilidade de serem destinos. Cada fonte emite o mesmo número de pacotes a todos os demais nodos.

- Não-Uniforme: a probabilidade de um nodo enviar pacotes para seus vizinhos é o dobro em relação aos nodos restantes, ou seja, a probabilidade diminui com a distância entre eles.

- Local: todos os nodos vizinhos tem a mesma probabilidade de serem destinatários e recebem o mesmo número de pacotes. Contudo, não há comunicação entre nodos adjacentes, ou seja, em uma rede intra-chip que utiliza este padrão os núcleos conectados a um roteador somente trocarão informações entre si.

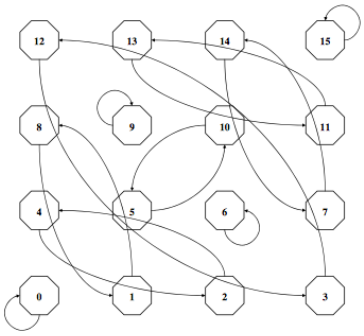
4.1.2 Padrões com único destinatário

Os padrões com único destinatário são aqueles em que os núcleos enviam pacotes para um endereço específico e sem alterá-lo ao longo da execução, portanto, estes padrões exibem alta localidade temporal e a localidade espacial varia de acordo com a distribuição utilizada. Estes padrões são largamente visualizados em arquiteturas paralelas e podem ser aplicados às NoC devido suas similaridades. As subseções seguintes descrevem as distribuições apresentadas no trabalho de Tedesco (2005) e pelos autores Duato, Yalamanchili e Ni (2003).

4.1.2.1 *Bit-reversal*

Na distribuição *bit-reversal*, o nodo que possui coordenadas no formato binário $a_{n-1}, a_{n-2}, ..., a_1, a_0$, comunica-se com o nodo $a_0, a_1, ..., a_{n-2}, a_{n-1}$, conforme demonstra a Figura 47.

Figura 47 – Distribuição espacial *bit-reversal*.

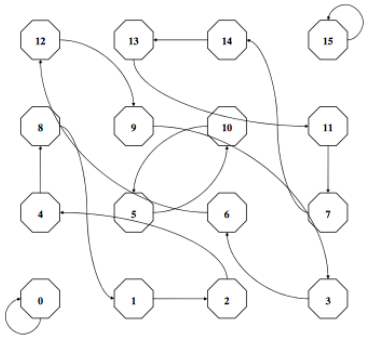


Fonte: adaptada de (TEDESCO, 2005).

4.1.2.2 *Perfect Shuffle*

Na distribuição *perfect shuffle*, o nodo que possui coordenadas no formato binário $a_{n-1}, a_{n-2}, ..., a_1, a_0$, comunica-se com o nodo $a_{n-2}, a_{n-3}, ..., a_0, a_{n-1}$, ou seja, ocorre a rotação de 1 bit para a esquerda. A Figura 48 ilustra o diagrama de fluxo deste padrão.

Figura 48 – Distribuição espacial *perfect shuffle*.

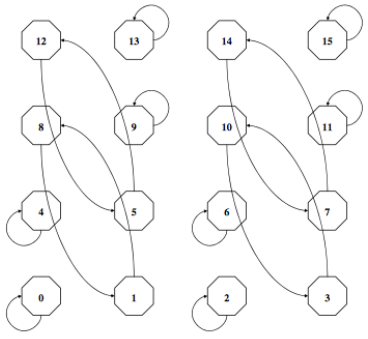


Fonte: adaptada de (TEDESCO, 2005).

4.1.2.3 *Butterfly*

Na distribuição *butterfly*, o nodo com coordenadas $a_{n-1}, a_{n-2}, ..., a_1, a_0$, comunica-se com o nodo $a_0, a_{n-2}, ..., a_1, a_{n-1}$, ou seja, há a troca dos bits mais e menos significativos, como mostra a Figura 49.

Figura 49 – Distribuição espacial *butterfly*.



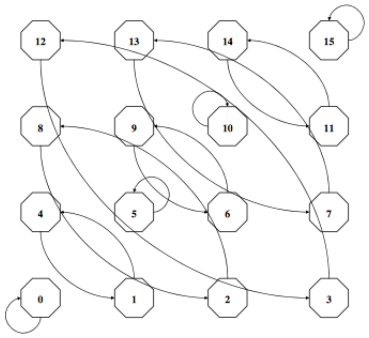
Fonte: adaptada de (TEDESCO, 2005).

4.1.2.4 *Matrix Transpose*

Na distribuição *matrix transpose*, o nodo que possui coordenadas no formato binário $a_{n-1}, a_{n-2}, ..., a_1, a_0$, comunica-se com o nodo $a_{\frac{n}{2}-1}, ..., a_0, a_{n-1}, a_{\frac{n}{2}}$, ou seja, há a rotação

de $\frac{n}{2}$ bits para a esquerda, onde n é o número de bits que identificam o nodo. A Figura 50 ilustra este padrão.

Figura 50 – Distribuição espacial *matrix transpose*.

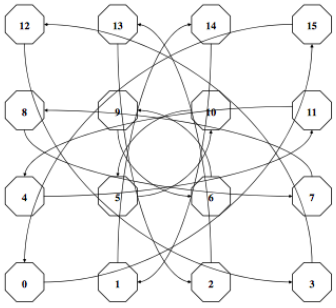


Fonte: adaptada de (TEDESCO, 2005).

4.1.2.5 Complemento

Na distribuição complemento, o nodo que possui coordenadas no formato binário $a_{n-1}, a_{n-2}, ..., a_1, a_0$, comunica-se com o nodo $a_0, a_1, ..., a_{n-2}, a_{n-1},$, ou seja, todos os bits tem seus valores invertidos. A Figura 51 ilustra o diagrama de fluxo deste padrão.

Figura 51 – Distribuição espacial complemento.



Fonte: adaptada de (TEDESCO, 2005).

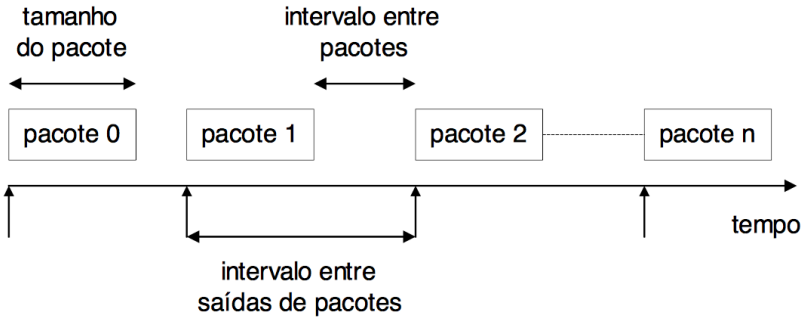
4.2 CARGA OFERECIDA

A carga oferecida corresponde ao percentual de ocupação do canal e relaciona a taxa de injeção dos dados por um determinado núcleo com a capacidade total do canal que conecta o mesmo núcleo à rede (TEDESCO, 2005).

A taxa de injeção dos dados pode ser estabelecida dentro de um quadro com as seguintes possibilidades: (i) o tamanho dos pacotes é fixo e o intervalo de tempo entre o término do envio de um pacote e o início do próximo é variável; (ii) variar o tamanho

dos pacotes e fixar o intervalo entre eles; (*iii*) o tamanho dos pacotes é fixo e varia-se o intervalo de saída entre eles. A Figura 52 ilustra os parâmetros relativos à taxa de injeção.

Figura 52 – Parâmetros relativos à taxa de injeção.



Fonte: (TEDESCO, 2005).

No que diz respeito às NoC's, o parâmetro que define o tamanho dos pacotes está diretamente ligado à quantidade de *flits* que compõem estes pacotes. Os autores Bolotin et al. (2004) classificam o tráfego da rede QNoC dentro de quatro grupos com base na quantidade média de *flits* dos pacotes, como demonstra a Tabela 5. Assim, a carga oferecida pode, ainda, variar entre um destes grupos, de acordo com a aplicação e o tamanho dos pacotes.

Tabela 5 – Classificação dos tráfegos na rede QNoC.

Tipo	Tamanho médio (*flits*)
Signaling	2
Real-Time	40
RD/WR	4
Block-Transfer	2000

Fonte: adaptada de Bolotin et al. (2004).

4.3 MODELAGEM DE TRÁFEGO

A modelagem de tráfego objetiva introduzir propriedades probabilísticas de aplicações reais à geração de tráfego. Para atingir tal objetivo, alteram-se os parâmetros da carga oferecida para atingir certo grau de realismo às simulações (TEDESCO, 2005).

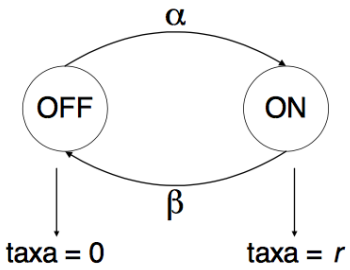
No modelo de tráfego constante, os pacotes são gerados a uma taxa fixa. Este é o modelo mais utilizado nas avaliações das redes de interconexão que simulam as aplicações com tráfego de sinais telefônicos digitalizados não-compactados e *streams* de áudio e vídeo.

O modelo *ON-OFF* implementa períodos de atividade e inatividade. Nos períodos de atividade, o núcleo fonte gera pacotes de tamanho fixo em intervalos regulares, enquanto

que nos períodos de inatividade não são injetados pacotes. De acordo com Pande et al. (2005), esta modelagem pode ser combinada com a distribuição de probabilidade Pareto e ser utilizada para simular aplicações de vídeo e redes computacionais.

O modelo *Markov ON-OFF* segue as mesma premissas do modelo *ON-OFF* convencional, porém, os dados são gerados a uma taxa r nos períodos de atividades, esta especificada pelo estado corrente da fonte de tráfego em uma cadeia de Markov, ilustrada pela Figura 53. Os parâmetros α e β indicam as probabilidades de mudança do estado da fonte.

Figura 53 – Cadeia de Markov.



Fonte: (TEDESCO, 2005).

Uma Cadeia de Markov é um processo em que a probabilidade de estar em um certo estado em um tempo futuro pode depender do estado atual do sistema, mas não dos estados em tempos passados (GRIGOLETTI, 2011). Em outras palavras, dado o estado atual do sistema da Cadeia de Markov, o próximo estado é independe do passado, mas pode ou não depender do estado presente.

Uma das mais importantes características exibidas por muitas cadeias de Markov é um comportamento de equilíbrio em longo prazo. Esta propriedade, denominada memória Markoviana, faz com que a distribuição da cadeia de Markov permaneça aproximadamente a mesma em um certo período, após um longo tempo de execução (NEVES, 2014). Isso significa que, em longo prazo, as probabilidades de o sistema estar em cada um dos vários estados pouco ou nada variam à medida que o tempo passa.

O modelo para geração de tráfegos pode, ainda, fazer uso de um tipo especial da Cadeia de Markov em que nem todos os estados são perfeitamente conhecidos. Os denominados modelos de Markov escondidos surgiram originalmente no domínio de reconhecimento da fala e atualmente têm sido empregados em modelos de computação natural, em trabalhos sobre visão computacional e reconhecimento de manuscritos, formas, gestos e expressões faciais, como também na simulação de aplicações do campo da biologia computacional.

5 AVALIAÇÃO EXPERIMENTAL

Este capítulo apresenta os resultados obtidos em experimentos realizados com a rede RTSNoC e é dividido em duas seções. A primeira seção discute o funcionamento da rede implementada em SystemC, da interface de rede e do sistema de monitoramento apresentados no Capítulo 3 deste trabalho durante a fase de validação e verificação do projeto. A segunda seção apresenta os resultados de experimentos com uma rede RTSNoC simulada, composta por quatro roteadores de oito conexões, ou seja, a rede utilizada interconecta até vinte e quatro núcleos. O objetivo dos experimentos foi obter o tempo médio de entrega dos pacotes em diferentes configurações de tráfego para, assim, avaliar o comportamento da rede.

5.1 VALIDAÇÃO E VERIFICAÇÃO

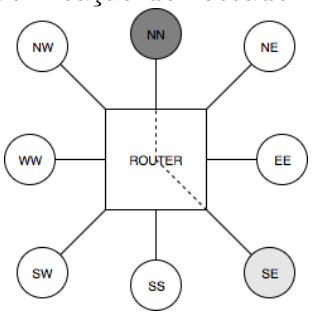
Para que experimentos com os modelos de tráfego apresentados no Capítulo 4 sejam efetuados, é necessário que todos os componentes da rede tenham seu funcionamento verificado e validado. Para tal, utilizou-se da ferramenta *GTKWave*, um analisador de ondas que lê arquivos do tipo VCD (DELANGE; HUGUES; DISSAUX, 2012). Este formato é um padrão na indústria para verificação de projetos desenvolvidos em *Verilog*, especificado pela normativa IEEE-1364 e com suporte também no SystemC. As subseções a seguir discutem os resultados obtidos na fase de verificação e validação de cada componente.

5.1.1 Roteadores da RTSNoC

O primeiro componente a ser validado é o roteador da rede RTSNoC. Para isso, após ser implementado em SystemC, um gerador de estímulos foi conectado à porta norte do roteador para simular a produção de pacotes. Na porta sudeste, outro gerador de estímulos simula a requisição destes pacotes. A Figura 54 demonstra o ambiente de verificação utilizado.

De acordo com o que foi apresentado na Seção 2.7 do Capítulo 2, 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. Assim, a latência máxima esperada para o *flit* de cabeçalho L_{header}, que pertence a um pacote do fluxo σ_i é dada pela Equação 5.1.

Figura 54 – Ambiente de verificação do roteador da RTSNoC em SystemC.



Fonte: elaborada pelo autor.

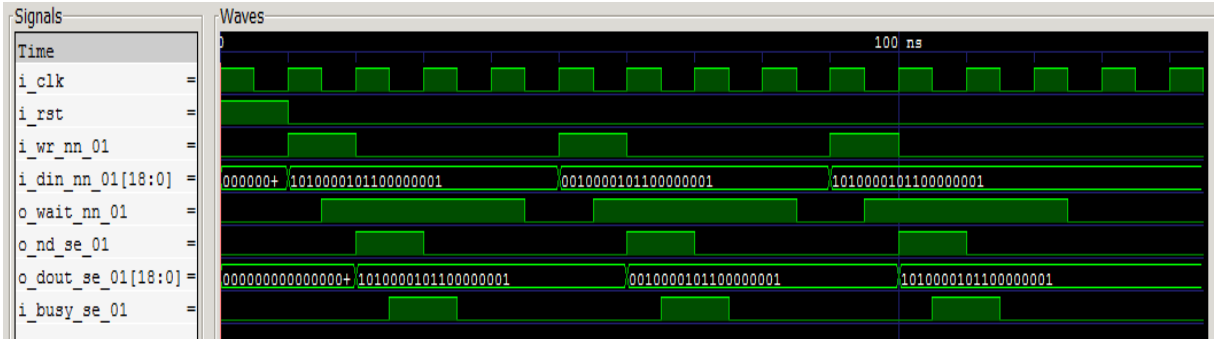
$$L_{header(i)} = \sum_{j=1}^{h} 2N_j \tag{5.1}$$

A vazão na rede RTSNoC para um determinado fluxo é dada pela expressão $B_i = \frac{1}{2k}$, então a latência da carga útil e do *flit* terminador é dada pela Equação 5.2, onde 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.

$$L_{payload;tail(i)} = \frac{(f-1)}{\frac{1}{2k}} = 2k(f-1) \tag{5.2}$$

A Figura 55 demonstra o gráfico obtido na ferramenta *GTKWave* ao simular o roteador da RTSNoC. Foi explicitado o roteamento de um pacote contendo três *flits* da porta norte para a porta sudeste. Como não há mais nenhum pacote competindo por aquele nodo destino e a quantidade de *flits* que compõe este pacote é igual ao valor mínimo, considera-se que as condições de roteamento são ideais. Portanto, como pode ser observado naquele gráfico, é gasto apenas um ciclo de *clock* para rotear um *flit*. Este é comportamento esperado e o roteador é validado pelas equações 5.1 e 5.2.

Figura 55 – Gráfico obtido na validação do roteador da RTSNoC.

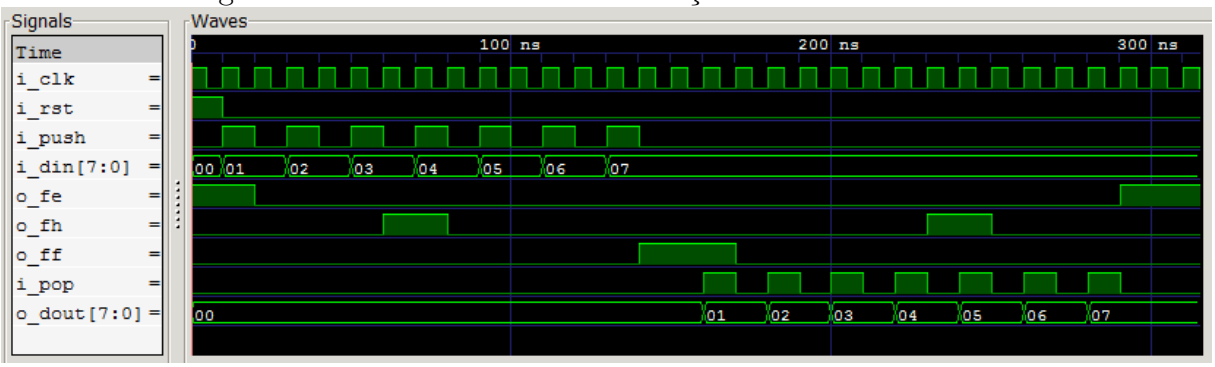


Fonte: elaborada pelo autor.

5.1.2 Memórias FIFO

As memórias FIFO utilizadas pela Interface de Rede discutidas na Subseção 3.2.3 também precisaram ter seu comportamento validado antes de serem introduzidas ao projeto. De forma similar ao roteador da Figura 54, geradores de estímulos simularam operações de leitura e escrita de uma memória com profundidade de oito posições.

Figura 56 – Gráfico obtido na validação das memórias FIFO.



Fonte: elaborada pelo autor.

O gráfico gerado pelo *GTKWave* é apresentado na Figura 56. A memória inicia com o *flag* que indica estado vazio ativo. Ao realizar uma operação de escrita o *flag* é desativado. Quando a memória atinge metade da sua capacidade, após três operações de escrita, o *flag* que indica memória parcialmente cheia é ativado. Após sete operações de escrita o *flag* de memória cheia é ativado e permanece assim até que uma operação de leitura seja feita. Nota-se que o dado apresentado na saída da primeira leitura é igual ao primeiro dado escrito.Após quatro operações de leitura a FIFO atinge novamente metade da sua capacidade e após sete leituras, retorna ao estado vazio. A implementação da memória é, portanto, validada pelo seu comportamento.

5.1.3 Interface de Rede

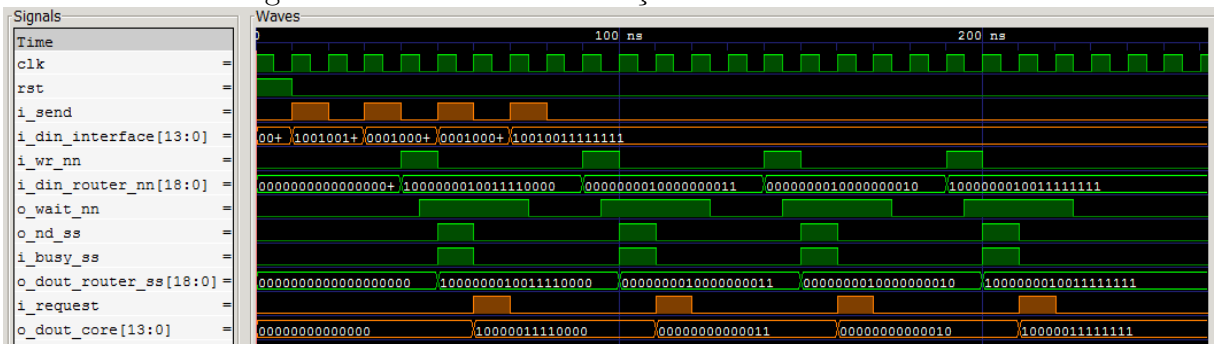
A verificação da Interface de Rede ocorreu no mesmo ambiente da Figura 54, com o nodo conectado à porta sul do roteador como destinatário. A latência de pior caso da RTSNoC é expressa na Equação 5.3, onde B é a profundidade das memórias FIFO utilizadas na Interface de Rede. Este coeficiente é multiplicado por dois, pois são necessárias duas filas FIFO.

$$WCL_{packet(i)} = \sum_{j=1}^{h} 2N_j + 2k(f-1) + 2B \tag{5.3}$$

A adição do coeficiente referente às memórias FIFO no cálculo da latência é jus-

tificada pelo gráfico da Figura 57. Os canais de entrada e saída da Interface de Rede foram destacados em laranja naquele gráfico. Nota-se que após inserir este componente ao roteador da RTSNoC houve um incremento de quatro ciclos de *clock* para a entrega de um *flit*, valor previsto pela Equação 5.3 e referente às operações de leitura e escrita das filas. Além disso, a Interface de Rede eficientemente concatenou os dados inseridos na rede pelo núcleo de orgiem com o seu respectivo endereço, assim como removeu dos dados entregues ao destinatário os campos relativos ao endereço de destino.

Figura 57 – Gráfico de validação da Interface de Rede.

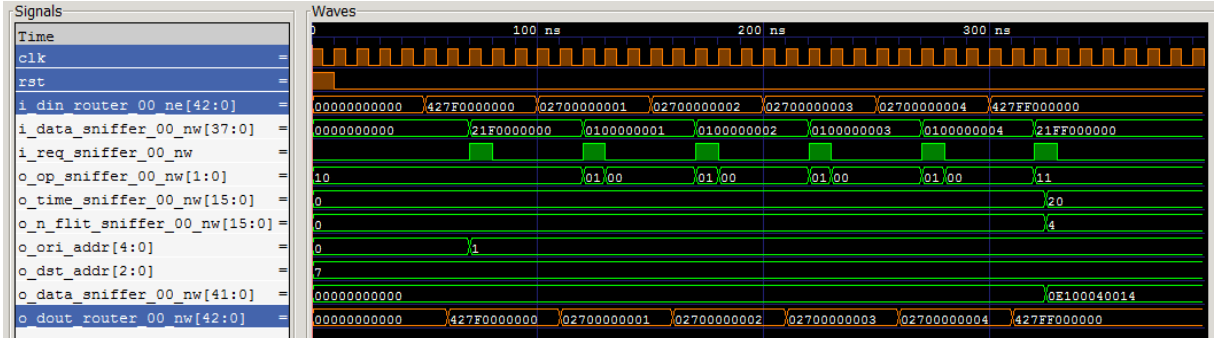


Fonte: elaborada pelo autor.

5.1.4 *Sniffer* e *Manager*

O processo de validação e verificação de ambos *Sniffer* e *Manager* transcorreu de forma simultânea. Um pacote com quatro *flits* de carga útil foi enviado da porta nordeste à noroeste do mesmo roteador, representado pelas ondas de cor laranja na Figura 54. O sinal *o_op_sniffer_00nw* diz respeito à entrada descrita na Tabela 4. Percebe-se também que o *sniffer* contabilizou corretamente a quantidade de *flits* e ciclos de *clock* gastos na entrega do pacote, o que valida seu funcionamento. Ademais, como previsto o mecanismo não alterou a taxa de entrega dos *flits* ou interferiu no funcionamento do roteador.

Figura 58 – Gráfico de validação do *Sniffer* e *Manager*.



Fonte: elaborada pelo autor.

5.2 AVALIAÇÃO DO SISTEMA DE MONITORAMENTO

A presente seção descreve a avaliação do sistema de monitoramento. A versão da rede RTSNoC utilizada nos experimentos é demonstrada na Figura 46(a). Esta versão é composta de quatro roteadores conectados em malha ortogonal, aptos a realizar até oito interconexões, com topologia de monitoramento centralizada e todos os vinte e quatro núcleos híbridos, ou seja, tanto enviam quanto recebem dados. Os demais parâmetros são especificados na Tabela 6.

Tabela 6 – Parâmetros da RTSNoC utilizada nos experimentos

Parâmetro	Especificação
Frequência do sinal de *clock*	1GHz
Tamanho da rede	2x2
Quantidade de núcleos	24
Largura do campo de *payload*	32 bits
Roteamento	XY estático
Arbitragem	Intercalação de *flits*
Técnica adotada no controle de fluxo	Baseada em créditos
Profundidade das filas na Interface de Rede	32 posições
Resolução dos contadores de *flits*	11 bits
Resolução dos *timers*	17 bits

Fonte: elaborada pelo autor.

Os padrões utilizados para distribuição espacial dos núcleos nos experimentos são: complemento, local, uniforme e não-uniforme. Os padrões *bit-reversal*, *perfect shuffle*, *butterfly* e *matrix transpose* não são considerados nos experimentos, pois, como demonstrado na Subseção 4.1.2, nestes padrões nem todos os nodos da rede injetam pacotes, o que induz uma largura de banda efetiva menor do que a especificada pela taxa de injeção e dificulta a avaliação dos resultados.

As subseções seguintes apresentam os cenários utilizados nos experimentos para avaliação do sistema de monitoramento: o primeiro que segue o modelo de tráfego constante e injeção de pacotes com tamanho fixo; o segundo cenário, baseado no modelo *ON-OFF* de injeção do tráfego, com pacotes de tamanho randomicamente variado.

5.2.1 Modelo Constante

O primeiro cenário de avaliação visa identificar o comportamento da rede nas diferentes distribuições espaciais. Os pacotes injetados seguem a mesma classificação da rede QNoC apresentada na Tabela 5. O arquivo *log* gerado pelo *manager* é exemplificado na Figura 59 e segue o padrão CSV para facilitar o tratamento dos dados. A primeira

coluna deste arquivo é referente ao identificador do pacote, seguido do seu fluxo. A terceira coluna é onde está impressa a quantidade de *flits* com carga útil daquele pacote, seguido do tempo de entrega em nanossegundos. Por fim, são impressos o tipo do pacote e o seu tempo de chegada em nanossegundos. Cabe citar que o *manager* registra as informações no arquivo *log* ordenadamente com base no tempo de chegada dos pacotes. Como previsto, devido à técnica de intercalação dos *flits*, os pacotes menores provenientes de fluxos críticas foram entregues antes e em um tempo menor que os demais. Além das informações geradas pelos *sniffers*, são impressos no início do arquivo o padrão de distribuição avaliado e o tempo total de simulação ao fim.

Figura 59 – Exemplo de um arquivo log gerado pelo *Manager*.

```
──────────────────────────────────── RTSNoC ────────────────────────────────────

1st Environment: Complement

packet, ori_addr -> dst_addr , n_flit , delivery_time , type,           arrival_time

  0,       9     ->    21,       2,     84 ns,          signaling,      163 ns
  0,       5     ->    26,       2,     84 ns,          signaling,      167 ns
  0,       21    ->    9,        2,     84 ns,          signaling,      178 ns
  0,       20    ->    11,       2,     84 ns,          signaling,      179 ns
  0,       24    ->    7,        4,     168 ns,         rd-wr,          224 ns
  0,       8     ->    23,       4,     168 ns,         rd-wr,          236 ns
  0,       4     ->    27,       4,     168 ns,         rd-wr,          240 ns
  0,       31    ->    1,        4,     168 ns,         rd-wr,          267 ns
  0,       29    ->    3,        4,     168 ns,         rd-wr,          268 ns
  0,       23    ->    8,        4,     168 ns,         rd-wr,          273 ns
  0,       25    ->    6,        40,    1064 ns,        real-time,      1135 ns
  0,       3     ->    29,       40,    1246 ns,        real-time,      1321 ns
  0,       15    ->    17,       40,    1232 ns,        real-time,      1335 ns
  0,       14    ->    18,       40,    1232 ns,        real-time,      1336 ns
  0,       7     ->    24,       40,    1232 ns,        real-time,      1341 ns
  0,       27    ->    4,        2000,  28798 ns,       block-transfer, 28877 ns
  0,       26    ->    5,        2000,  28798 ns,       block-transfer, 28878 ns
  0,       11    ->    20,       2000,  29274 ns,       block-transfer, 29346 ns
  0,       6     ->    25,       2000,  29267 ns,       block-transfer, 29350 ns
  0,       1     ->    31,       2000,  29288 ns,       block-transfer, 29354 ns
  0,       13    ->    19,       2000,  29274 ns,       block-transfer, 29369 ns
  0,       17    ->    15,       2000,  42175 ns,       block-transfer, 42230 ns
  0,       19    ->    13,       2000,  42175 ns,       block-transfer, 42252 ns
  0,       18    ->    14,       2000,  42175 ns,       block-transfer, 42253 ns

Simulation time: 42253 ns

──────────────────────────────────────────────────────────────────────────────
```

Fonte: elaborada pelo autor.

Como citado ao longo do desenvolvimento deste trabalho, a rede RTSNoC possui previsibilidade de latência, esta calculada por uma equação de pior caso. Portanto, previamente à execução dos testes, obteve-se os valores teóricos para cada um dos quatro tipos de fluxo.

$$L_{header(i)} = \sum_{j=1}^{h} 2N_j = (2 \times 8) + (2 \times 8) + (2 \times 8) = 48 \tag{5.4}$$

Na rede utilizada nos testes, um pacote pode atravessar até três roteadores, ou seja, realizar três saltos. A Equação 5.4 descreve a quantidade de ciclos que um *flit* de cabeçalho necessita para atravessar, por exemplo, os roteadores 0→1→3. Naquela

equação, N representa a quantidade de fluxos que competem pelos mesmos recursos, que num pior caso é igual à oito.

$$L_{payload;tail(i)} = 2k(f-1) = 2 \times 8(2002-1) = 32016 \tag{5.5}$$

A Equação 5.5, por sua vez, descreve a quantidade de ciclos que os *flits* de carga útil do tipo *block-transfer* necessitam para atravessar o mesmo caminho em uma situação de pior caso. O fator k daquela equação descreve a quantidade de pacotes que competem pelos mesmos recursos do roteador, ou seja, para a rede com roteadores que operacionalizam até oito interconexões, este fator também será oito no pior caso.

$$B = 2K(f-p) = 2 \times 8 \times (2000-32) = 31488 \tag{5.6}$$

Em condições ideais, onde os núcleos leem os *flits* das Interfaces de Rede imediatamente após a indicação da sua chegada e sem a possibilidade de encher os *buffers* de memória das filas FIFO, o termo B da Equação 5.6 é nulo. Caso contrário, verificou-se que este termo depende da quantidade de *flits* do pacote (f), da profundidade das filas FIFO (p) e de um fator K, que diz respeito aos ciclos de *clock* necessários para a interface escrever e o núcleo ler um próximo *flit*, e, também por tal motivo que este fator é multiplicado por 2. Assim, é descrito na Equação 5.6 o total de ciclos de *clock* gastos pelos geradores de tráfego do fluxo *block-transfer*.

$$WCL_{packet(i)} = \sum_{j=1}^{h} 2N_j + 2k(f-1) + 2B = 48 + 32016 + (2 \times 31488) = 95040 \tag{5.7}$$

Por fim, a Equação 5.7 descreve o valor total da latência de pior caso para os fluxos do tipo *block-transfer*. O mesmo raciocínio foi utilizados para o cálculo do valor teórico dos demais fluxos, detalhados na Tabela 7. Cabe citar que estes valores são referentes apenas à rede utilizada nos experimentos e pode variar de acordo com a topologia utilizada, geradores de tráfego utilizados e demais especificações. O valor teórico para a latência de melhor caso é obtida quando um fluxo entre os núcleos mais distantes trafega na rede sem competição, como demonstra a Figura 63.

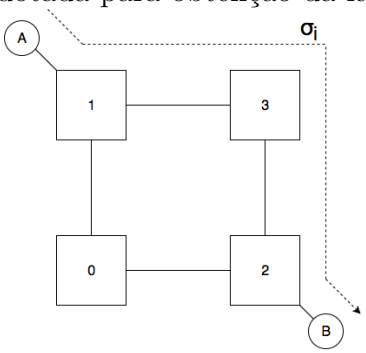
A taxa de injeção (do inglês, *injection rate*) utilizada nos experimentos é demonstrada na Equação 5.8. Essa taxa relaciona a quantidade de *flits* injetadas pelos geradores de acordo com o seu tipo. Uma taxa igual à 100% significa que todos os vinte e quatro núcleos conetados à rede são geradores de fluxos do tipo *block-transfer*, a base do cálculo por se aquele que demanda mais recursos da rede.

Tabela 7 – Valores teóricos de pior caso para cada tipo de fluxo.

Fluxo	Quantidade de ciclos de *clock*
Signaling	128
Real-Time	2046
RD/WR	256
Block-Transfer	95040

Fonte: elaborada pelo autor.

Figura 60 – Técnica adotada para obtenção da latência de melhor caso.

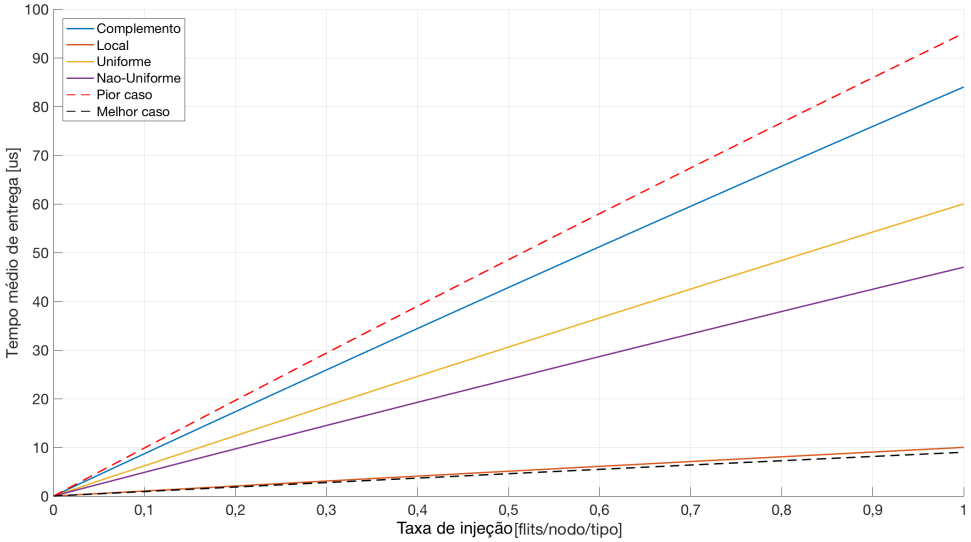


Fonte: elaborada pelo autor.

$$IR(\%) = \frac{2000x_1 + 40x_2 + 4x_3 + 2x_4}{24 * 2000} * 100 \tag{5.8}$$

O gráfico da Figura 61 ilustra as curvas para o tempo médio de entrega obtidos nos experimentos com cada padrão de distribuição espacial dos geradores de tráfego. Como o objetivo deste experimento é observar o comportamento da rede em diferentes taxas de injeção, alterou-se a proporção dos núcleos geradores de pacotes grandes e pequenos. Os valores teóricos de melhor e pior caso também estão ilustrados naquela figura.

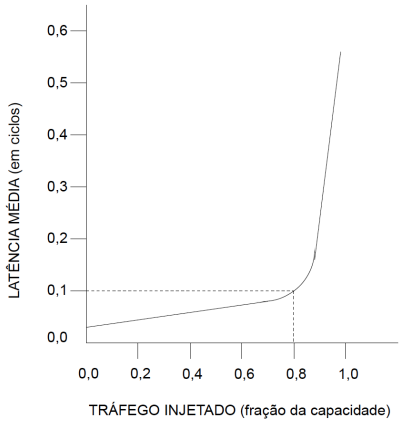
Figura 61 – Tempo médio de entrega obtido em cada distribuição x Taxa de injeção.



Fonte: elaborada pelo autor.

O primeiro ponto a ser observado naquele gráfico, é a clara linearidade das curvas, característica esperada de uma rede que utiliza da técnica de intercalação dos *flits*, diferente de outras redes que tendem a apresentar curvas exponenciais com valores mais acentuados em taxas de injeção maiores que 80%, como demonstra a Figura 62.

Figura 62 – Latência de uma rede genérica.

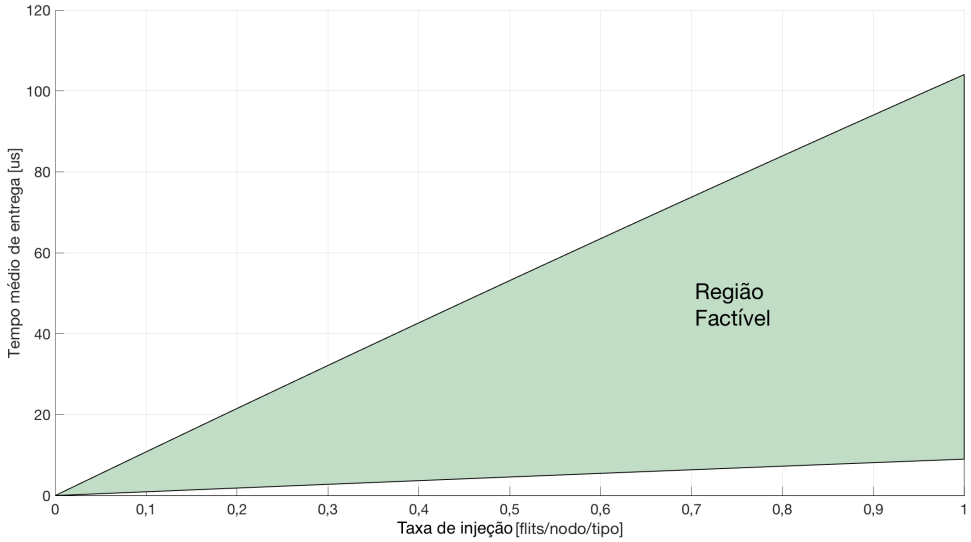


Fonte: adaptada de (DALLY; TOWLES, 2001).

A distribuição espacial que apresentou melhores resultados foi a Local, já que neste padrão não há comunicação entre roteadores adjacentes. A distribuição Complemento apresentou valores próximos ao valor teórico de pior caso. As outras duas distribuições, Uniforme e Não-Uniforme, tiveram performance mediana. É importante destacar que a distribuição Não-Uniforme contam com um senso de localidade maior que a Uniforme, ou seja, naquela distribuição há probabilidade maior de um núcleo trocar informações com seus vizinhos do que com os núcleos conetados em roteadores mais distantes, o que justifica sua maior proximidade com a curva da distribuição Local.

O ramo da programação linear consiste em solucionar os problemas de maximização ou minimização de uma função objetivo que satisfaz um certo número de restrições (NEVES, 2014). A linearidade do tempo de entrega médio possibilita a aplicação de métodos provenientes da programação linear para otimização dos resultados. Os valores teóricos de pior e melhor caso atuam como as restrições do problema. Graficamente, a região delimitada por aqueles limiares é denominada factível, como demonstra a Figura 63, pois um algoritmo de otimização linear busca dentro daquela região um valor ótimo para a função objetivo. Em uma aplicação real, as informações obtidas pelo sistema de monitoramento aliadas aos valores teóricos calculadores podem ser utilizadas por métodos que visem otimizar a distribuição espacial dos núcleos e, consequentemente, melhorar o tempo médio de entrega dos pacotes. A computação reconfigurável pode ser utilizada em conjunto à programação linear para operacionalizar a redistribuição dos núcleos na rede.

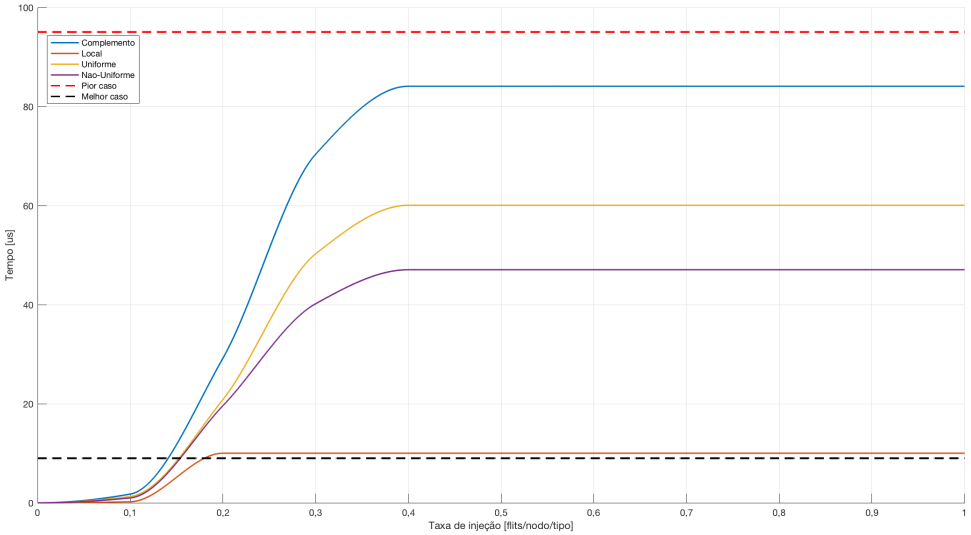
Figura 63 – Demonstração da região de valores factíveis.



Fonte: elaborada pelo autor.

O modelo constante de geração do tráfego possibilita a mensura do tempo total de simulação dos experimentos para cada taxa de injeção e cada padrão de distribuição, expresso na Figura 64. Nota-se que as curvas de todos os padrões convergem para um valor constante a partir de uma determinada taxa de injeção. Como esperado, a distribuição Local demandou menor tempo de simulação e convergiu em taxas menores que os demais.

Figura 64 – Tempo de simulação obtido em cada distribuição x Taxa de injeção.

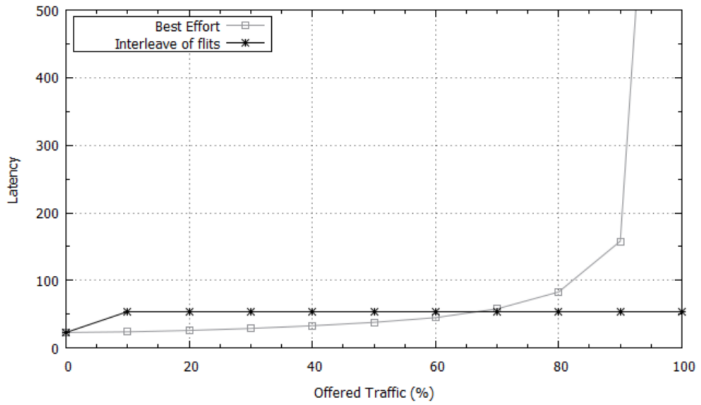


Fonte: elaborada pelo autor.

Os resultados do tempo total de simulação podem ser confrontados com a latência esperada de uma rede que utiliza da técnica de intercalação dos *flits*. Na Figura 69 são apresentadas as curvas esperadas da latência de uma rede que utiliza alguma técnica de melhor esforço e de uma rede com intercalação de *flits*. Aquela de melhor esforço apresenta

resultados exponenciais com o aumento da taxa de injeção, já a de intercalação dos *flits* tenta amenizar a latência e convergir os resultados para um valor constante.

Figura 65 – Latência de uma rede puramente *best-effort* x Latência de uma rede com intercalação de *flits*.



Fonte: (BEREJUCK, 2015).

5.2.2 Modelo *ON-OFF*

O segundo cenário de avaliação objetiva avaliar se as mesmas propriedades verificadas no modelo constante de injeção do tráfego são observáveis em ambientes mais dinâmicos. O modelo *ON-OFF* foi escolhido por ser uma representação dos sistemas multimídia reais com pacotes de tamanhos variados ao longo da execução.

A geração dos pacotes de forma randômica significa que não existe correlação entre amostras subsequentes. Os autores James et al. (2013) demonstram que uma forma de determinar se as amostras de um conjunto de dados são correlacionados é pela análise do "coeficiente de correlação", dado pela equação:

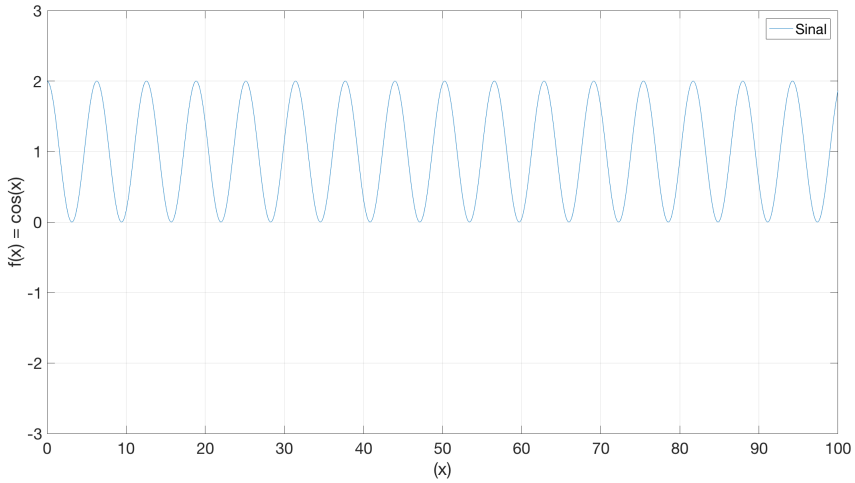
$$r = \frac{\frac{\sum_{j=1}^{N}(x_i-\bar{x})(y_i-\bar{y})}{N-1}}{\sqrt{\frac{\sum_{j=1}^{N}(x_i-\bar{x})^2}{(N-1)}}\cdot\sqrt{\frac{\sum_{j=1}^{N}(y_i-\bar{y})^2}{(N-1)}}} \tag{5.9}$$

em que N é a quantidade de amostras do conjunto de dados, x e y são as amostras subsequentes, $\bar{x}$ e $\bar{y}$ são os valores médios das amostras. A expressão do numerador da Equação 5.9 é a covariância entre amostras subsequentes x e y, e o denominador é o produto do desvio padrão dessas amostras. O coeficiente de correlação mede a intensidade da relação entre as amostras de um conjunto de dados e pode variar entre -1 e $+1$. Os autores James et al. (2013) dividem o tipo de relação 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;
- Se $(r = -1)$: correlação perfeitamente negativa.

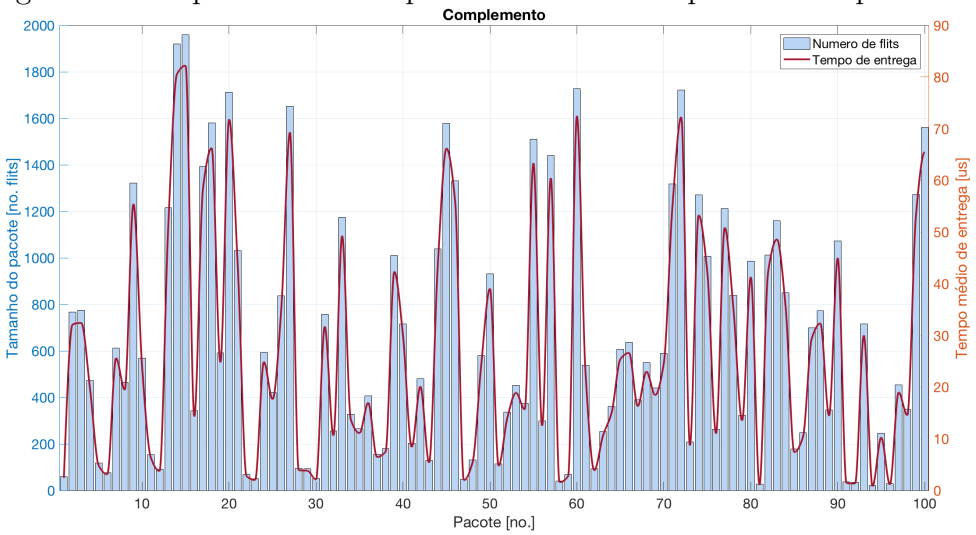
Figura 66 – Sinal amostrado com coeficiente de correlação alto.



Fonte: elaborada pelo autor.

Na Figura 66 é apresentado o histograma de um conjunto de dados obtidos por amostragem em um sinal cossenoidal com relação positiva. O coeficiente de correlação resultante deste sinal é de $0,995$, ou seja, alta.

Figura 67 – Experimento com pacotes variáveis no padrão Complemento.

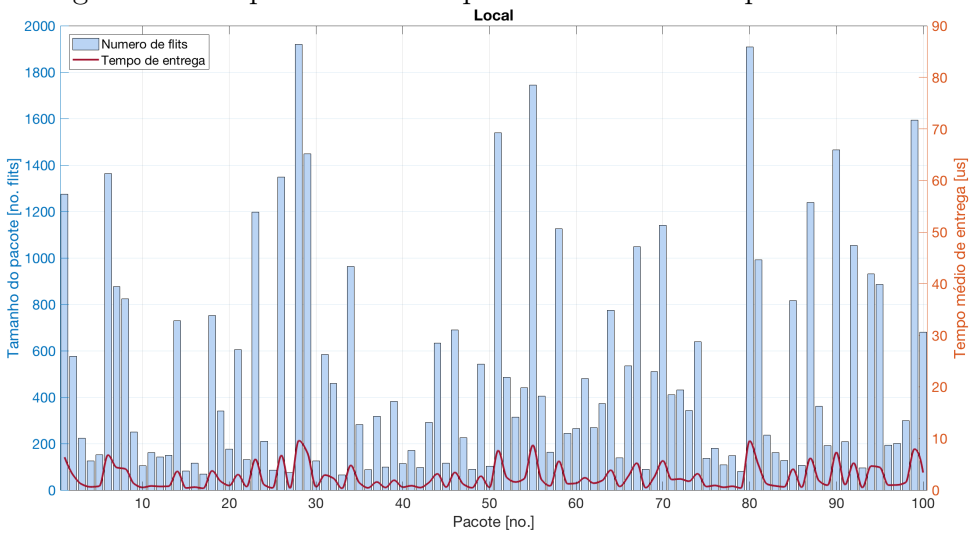


Fonte: elaborada pelo autor.

O gráfico ilustrado na Figura 67 relaciona o tamanho dos pacotes gerados no experimento com a distribuição Complemento e a curva do tempo médio de entrega obtida.

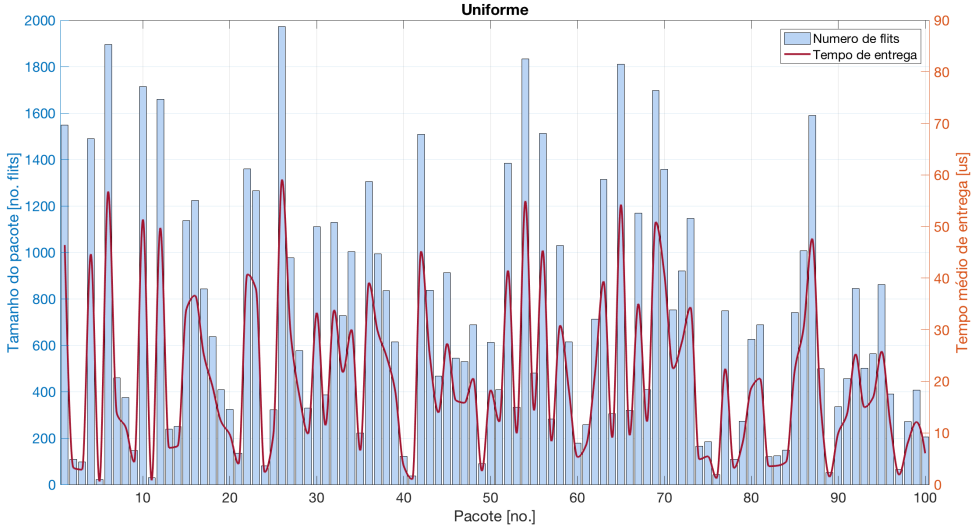
Ao observar este gráfico, percebe-se que de fato da RTSNoC é linear e proporcional aos tamanhos do pacotes, com a curva do tempo médio da entrega daqueles pacotes atingindo picos similares ao histograma dos seus tamanhos. O gráfico da Figura 68 relaciona da mesma forma os resultados obtidos com a distribuição Local. Como esperado, esse padrão apresentou um tempo médio de entrega muito menor que a distribuição Complemento.

Figura 68 – Experimento com pacotes variáveis no padrão Local.



Fonte: elaborada pelo autor.

Figura 69 – Experimento com pacotes variáveis no padrão Uniforme.

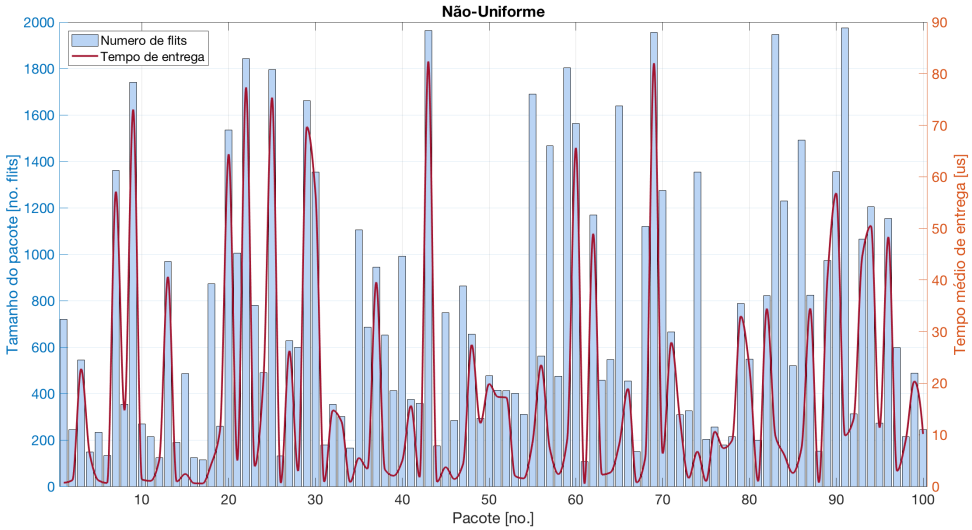


Fonte: elaborada pelo autor.

A Figura 69 e a Figura 70 demonstram os gráficos das distribuições Uniforme e Não-Uniforme, respectivamente, que apresentaram performance mediana nos experimentos com o modelo de tráfego constante e comportamento similar nos experimentos com o modelo *ON-OFF*. É importante notar que a curva do tempo médio de entrega não teve o

mesma forma que o histograma dos pacotes em determinados momentos no padrão Não-Uniforme. Isso comprova de forma visual o comportamento desta distribuição, em que as probabilidades de ser destinatário dos núcleos varia ao longo da execução com momentos em que a localidade é mais forte, o que acarreta na queda da curva do tempo de entrega.

Figura 70 – Experimento com pacotes variáveis no padrão Não-Uniforme.



Fonte: elaborada pelo autor.

A Tabela 8 mostra os resultados obtidos nas avaliações analíticas com os conjuntos de dados citados nesta subseção. Ao observar os valores obtidos, pode-se dizer que os quatro conjuntos de dados podem ser considerados randômicos. Além disso, pode-se afirmar que o gerador de tráfego proposto é adequado para gerar fluxos de dados similares aos sistemas multimídias.

Tabela 8 – Coeficiente de correlação entre os quatro sinais avaliados.

Sinal	Coeficiente de correlação	Correlação
Figura 66	0,995	Alta
Figura 67	-0,042	Baixa
Figura 68	0,081	Baixa
Figura 69	-0,164	Baixa
Figura 70	0,025	Baixa

Fonte: elaborada pelo autor.

Em um cenário com geradores de tráfego randômicos a forma de adaptabilidade dos roteadores pode ser feita estatisticamente, através do controle de processos estocásticos, com base nas informações obtidas pelo sistema de monitoramento. Um processo estocástico é uma sequência de variáveis aleatórias indexadas por algum elemento T, ou seja, um processo estocástico nada mais é que uma coleção de variáveis aleatórias que descrevem o comportamento de algum processo com o passar do tempo (SOUZA, 2013).

Algoritmos adaptativos aproveitam-se de duas características dos processos estocásticos para controlá-los: a sua periodicidade em tempos regulares e discretos; o fato que um processo estocástico evolui de um estado para outro dependentemente apenas do seu último estado, e consequentemente independentemente dos demais. O autor Souza (2013) cita, ainda, que os denominados processos Markovianos, aqueles modelados pela cadeia de Markov da Figura 53, possuem a peculiaridade de serem irrelevantes os estados anteriores para a predição do estado seguinte, desde que o estado atual seja conhecido. Esta propriedade é denominada memória Markoviana e pode, também, ser utilizada por algoritmos adaptativos no controle dos estados de tráfegos randômicos, pois após um longo período de execução os resultados pouco se alteram.

6 CONSIDERAÇÕES FINAIS

Este Trabalho de Conclusão de Curso relatou as atividades de pesquisa relacionadas ao uso de um sistema não-intrusivo para monitoramento da entrega de pacotes em redes intra-chip, baseado nos conceitos da observabilidade computacional. Foi feito um estudo sobre as principais redes publicadas até o momento voltadas para sistemas de tempo real ou que oferecem alguma técnica de garantia ao QoS.

A rede RTSNoC foi o objeto de pesquisa deste trabalho, por esta ser uma rede com previsibilidade da latência de pior caso para fluxos de tempo real *hard*. Esta rede garante um tempo de entrega baixo para fluxos de pequenos pacotes, entretanto, aplicações como multimídia onde os fluxos são em maioria caracterizados como de melhor esforço a rede oferece uma latência média. Um fator que aperfeiçoaria os resultados apresentados pela RTSNoC é a exploração em tempo de execução do senso de localidade, ao reconfigurar o posicionamento dos núcleos que comunicam-se com maior frequência no mesmo roteador ou roteadores próximos.

O conceito de computação reconfigurável faz-se necessário na rede RTSNoC. Todo mecanismo auto-adaptativo ou sistema reconfigurável precisa de uma base para a tomada de decisões. Assim, propôs-se a introdução de técnicas provenientes do campo da observabilidade computacional como solução para tal problem. As hipóteses levantadas afirmaram que era possível integrar mecanismos à rede capazes de monitorar a entrega de pacotes e fornecer informações relativas ao tráfego para um gerenciador ou algoritmo adaptativo.

A primeira contribuição deste trabalho é ter apresentado um sistema de monitoramento completamente não-intrusivo que utiliza de recursos próprios para transporte, armazenamento e avaliação das informações obtidas, sem interferir no funcionamento dos componentes ligados ao serviço de roteamento. A rede RTSNoC exigia a integração de tal sistema para que só então comportasse técnicas de adaptabilidade.

Através de experimentos realizados com diferentes distribuições espaciais de núcleos conectados à rede e modelos para geração de tráfego, foi validado o funcionamento do sistema de monitoramento e identificado alguns padrões de comportamento da RTSNoC. A segunda contribuição deste trabalho foi, portanto, comprovar pelos resultados obtidos com tais experimentos que de fato a distribuição espacial dos núcleos e o tamanho dos pacotes de seus fluxos são as métricas de maior influência no aumento da latência na RTSNoC. Além disso, o relatório gerado pelo Gerente de Rede também demonstrou que o algoritmo de intercalação dos *flits* adotado na arbitragem garante o suporte à QoS.

Uma terceira contribuição está relacionada ao fato da RTSNoC ter sido portada da linguagem VHDL de síntese e descrição de hardware para o SystemC, uma linguagem mais dinâmica e largamente utilizada na avaliação do desempenho de redes para interconexão.

Pelo que foi exposto, conclui-se que a rede RTSNoC é uma alternativa eficaz de interconexão entre elementos de processamento, que oferece previsibilidade de pior caso para aplicações de tempo real e que o sistema de monitoramento apresentado neste trabalho pode fornecer o suporte necessário para introduzir métodos de otimização, técnicas de computação reconfigurável, escalonamento de tarefas, redistribuição de carga, e controle de processos estocásticos, todos os quais visam aperfeiçoar o valor da latência média dos fluxos de melhor esforço.

6.1 TRABALHOS FUTUROS

Durante o desenvolvimento deste Trabalho de Conclusão de Curso foram observadas algumas questões que abrem possibilidades de novas pesquisas envolvendo a RTSNoC e o sistema de monitoramento integrado.

O tempo médio de entrega dos pacotes provou ser linear na rede utilizada nos experimentos, o que abre margem para a exploração de métodos otimizadores provenientes do ramo da programação linear. O uso de computação reconfigurável pode, agora, ser uma alternativa interessante a ser verificada junto à otimização, ao operacionalizar a redistribuição dos núcleos nos roteadores da rede em tempo de execução.

O armazenamento e análise *on-chip* dos dados fornecidos pelos mecanismos de monitoramento é outra questão que também necessita ser verificada com maior atenção. Esta é uma técnica largamente adotada em algoritmos de roteamento adaptativos, outro tema relevante para um trabalho de investigação científica agora que a rede conta com um sistema de realimentação das informações de tráfego.

A investigação do controle de processos estocásticos é outro tema a ser pesquisado em redes com tráfegos randômicos. Em aplicações com fluxos caracterizados por períodos de atividade curtos e constantes, procedidos por um longo período de inatividade também bem definido, é importante investigar o escalonamento de tarefas ou a distribuição da carga de pontos congestionados a estes núcleos durante os seus períodos de inatividade e, assim, melhorar a latência média de melhor esforço.

REFERÊNCIAS

BEREJUCK, M. D. **Rede intra-chip com qualidade de serviços para uso em telecomunicações**. Dissertação (Mestrado) — Universidade do Vale do Itajaí. Mestrado em Computação Aplicada, 2009.

BEREJUCK, M. D. **Rede intra-chip com previsibilidade de latência para uso em sistemas de tempo real**. Tese (Doutorado) — Universidade Federal de Santa Catarina. Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas, 2015.

BEREJUCK, M. D. An overview about networks-on-chip with multicast support. **International Journal of Advanced Studies in Computers, Science and Engineering**, 2016.

BEREJUCK, M. D.; FRÖHLICH, A. A. Evaluation of a connectionless technique for system-on-chip interconnection. **Journal of Circuits, Systems and Computers 25.10**, 2016.

BERTOZZI, D.; BENINI, L. Xpipes: a network-on-chip architecture for gigascale systems-on-chip. **IEEE Circuits and Systems Magazine**, v. 4, n. 2, p. 18–31, 2004. ISSN 1531-636X.

BHASKER, J. **A SystemCTM Primer**. [S.l.]: Star Galaxy Publishing, 2002.

BHOJWANI, P.; MAHAPATRA, R. Interfacing cores with on-chip packet-switched networks. In: **16th International Conference on VLSI Design, 2003. Proceedings.** [S.l.: s.n.], 2003. p. 382–387. ISSN 1063-9667.

BODEMÜLLER, R. **Mecanismos de previsão de perda de deadline para tratadores de eventos RTSJ**. Dissertação (Mestrado) — Universidade Federal de Santa Catarina. Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação, 2014.

BOLOTIN, E. et al. Qnoc: Qos architecture and design process for network on chip. **Journal of systems architecture**, Elsevier, v. 50, n. 2, p. 105–128, 2004.

CHESHIRE, S.; BAKER, M. Consistent overhead byte stuffing. **IEEE/ACM Transactions on networking**, IEEE, v. 7, n. 2, p. 159–172, 1999.

CHOUCHENE, W. et al. A low power network interface for network on chip. In: **Eighth International Multi-Conference on Systems, Signals Devices**. [S.l.: s.n.], 2011. p. 1–6.

CIORDAS, C. et al. An event-based network-on-chip monitoring service. In: IEEE. **High-Level Design Validation and Test Workshop, 2004. Ninth IEEE International**. [S.l.], 2004. p. 149–154.

COTA, E.; AMORY, A. de M.; LUBASZEWSKI, M. S. **Reliability, Availability and Serviceability of Networks-on-Chip**. [S.l.]: Springer, 2011. ISBN 978-1461407904.

CUMMINGS, C. E. Synthesis and scripting techniques for designing multi-asynchronous clock designs. In: **SNUG 2001 (Synopsys Users Group Conference, San Jose, CA, 2001) User Papers**. [S.l.: s.n.], 2001.

DALL'OSSO, M. et al. Xpipes: a latency insensitive parameterized network-on-chip architecture for multiprocessor socs. In: **Proceedings 21st International Conference on Computer Design**. [S.l.: s.n.], 2003. p. 536–539. ISSN 1063-6404.

DALLY, W. J.; TOWLES, B. Route packets, not wires: On-chip interconnection networks. In: IEEE. **Design Automation Conference, 2001. Proceedings**. [S.l.], 2001. p. 684–689.

DEHYADGARI, M. et al. Evaluation of pseudo adaptive xy routing using an object oriented model for noc. In: **2005 International Conference on Microelectronics**. [S.l.: s.n.], 2005. ISSN 2159-1660.

DELANGE, J.; HUGUES, J.; DISSAUX, P. Validate implementation correctness using simulation: the taste approach. 2012.

DING, K.-S.; HO, C.-T.; TSAY, J.-J. Matrix transpose on meshes with wormhole and xy routing. In: **Proceedings of 1994 6th IEEE Symposium on Parallel and Distributed Processing**. [S.l.: s.n.], 1994. p. 656–663.

DUATO, J.; YALAMANCHILI, S.; NI, L. M. **Interconnection networks: an engineering approach**. [S.l.]: Morgan Kaufmann, 2003.

FLOREZ, M. J. S. **Projeto de estruturas de comunicação intrachip baseadas em NoC que implementam serviços de QoS e segurança.** Tese (Doutorado) — Universidade de SÃ£o Paulo. Escola Politécnica. Doutorado em Microeletrônica., 2011.

FRANK, G. **Pulse code communication**. [S.l.]: Google Patents, mar. 17 1953. US Patent 2,632,058.

GHADERI, Z.; ALQAHTANI, A.; BAGHERZADEH, N. Online monitoring and adaptive routing for aging mitigation in nocs. In: **Design, Automation Test in Europe Conference Exhibition (DATE), 2017**. [S.l.: s.n.], 2017. p. 67–72.

GOOSSENS, K.; DIELISSEN, J.; RADULESCU, A. Aethereal network on chip: concepts, architectures, and implementations. **IEEE Design Test of Computers**, v. 22, n. 5, p. 414–421, Sept 2005. ISSN 0740-7475.

GRIGOLETTI, P. S. Cadeias de markov. **Recuperado em**, v. 19, n. 10, p. 2014, 2011.

HEMANI, A. et al. Network on chip: An architecture for billion transistor era. **IEEE NorChip Conference**, 2000.

IEEE, A. S. et al. Ieee standard for standard systemc language reference manual. **IEEE Computer Society**, 2012.

JAMES, G. et al. **An introduction to statistical learning**. [S.l.]: Springer, 2013.

MELLO, A. V. de. **Qualidade de serviço em redes intra-chip: implementação e avaliação sobre a rede Hermes**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul. Programa de Pós Graduação em Ciências da Computação, 2007.

MELO, D. R. de. **Desenvolvimento de aplicação com requisitos de qualidade de serviço para sistema integrado baseado em rede-em-chip**. 2008. Trabalho de Conclusão de Curso (Graduação) — Universidade do Vale do Itajaí. Ciência da Computação.

MORAES, F. et al. Hermes: an infrastructure for low area overhead packet-switching networks on chip. **INTEGRATION, the VLSI journal**, Elsevier, v. 38, n. 1, p. 69–93, 2004.

NEVES, R. A. O. D. **Programação linear e algumas extensões**. Dissertação (Mestrado) — Universidade do Porto. Faculdade de Ciências. Mestrado em Matemática para Professores, 2014.

PANDE, P. P. et al. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. **IEEE Transactions on Computers**, v. 54, n. 8, p. 1025–1040, Aug 2005. ISSN 0018-9340.

PEREIRA, T. F. **Gerador de tráfego sintetizável para redes-em-chip**. 2008. Trabalho de Conclusão de Curso (Graduação) — Universidade do Vale do Itajaí. Ciência da Computação.

PRASHANT, C. D. Designing asynchronous fifo. **Journal Of Information, Knowledge and Reseaarch In Electronics and communication Engineering**, v. 2, n. 2, 2013.

RADULESCU, A. et al. An efficient on-chip ni offering guaranteed services, shared-memory abstraction, and flexible network configuration. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 24, n. 1, p. 4–17, Jan 2005. ISSN 0278-0070.

SATHE, S.; WIKLUND, D.; LIU, D. Design of a guaranteed throughput router for on-chip networks. In: **2004 International Symposium on System-on-Chip, 2004. Proceedings.** [S.l.: s.n.], 2004. p. 25–28.

SODERQUIST, I. Globally updated mesochronous design style (gum-design-style). In: **Proceedings of the 28th European Solid-State Circuits Conference**. [S.l.: s.n.], 2002. p. 603–606.

SOUZA, D. M. de. **Modelos ocultos de Markov: uma Abordagem em Controle de Processos**. 2013. Trabalho de Conclusão de Curso (Graduação) — Universidade Federal de Minas Gerais. Estatística.

SUTHERLAND, S. The ieee verilog 1364-2001 standard what's new, and why you need it. In: **9th Internatioinal HDL Conference (HDLCon)**. [S.l.: s.n.], 2000.

SWAMINATHAN, K. et al. Design of a low power network interface for network on chip. In: **2013 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)**. [S.l.: s.n.], 2013. p. 1–4. ISSN 0840-7789.

TEDESCO, L. **Uma proposta para Geração de Tráfego e Avaliação de Desempenho para NoCs**. 125 p. Dissertação (Mestrado) — Pontifícia Universidade Católica Do Rio Grande Do Sul. Faculdade de Informática. Programa de Pós-Graduação em Ciência da Computação, 2005.

VERMEULEN, B.; OOSTDIJK, S.; BOUWMAN, F. Test and debug strategy of the pnx8525 nexperia/sup tm/digital video platform system chip. In: IEEE. **Test Conference, 2001. Proceedings. International**. [S.l.], 2001. p. 121–130.

WIKLUND, D.; LIU, D. Socbus: switched network on chip for hard real time embedded systems. In: **Proceedings International Parallel and Distributed Processing Symposium**. [S.l.: s.n.], 2003. p. 8 pp.–. ISSN 1530-2075.

ZEFERINO, C. A. **Redes-em-Chip : arquiteturas e modelos para avaliação de área e desempenho**. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação, 2003.

ZEFERINO, C. A. et al. Avaliação de desempenho de rede-em-chip modelada em systemc. In: **Proceedings of the 27rd Congress of Brazilian Computer Society-WPerformance**. [S.l.: s.n.], 2007. p. 559–578.

ZEFERINO, C. A.; SUSIN, A. A. Socin: a parametric and scalable network-on-chip. In: **16th Symposium on Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings.** [S.l.: s.n.], 2003. p. 169–174.

ZHANG, W. et al. Comparison research between xy and odd-even routing algorithm of a 2-dimension 3x3 mesh topology network-on-chip. In: **2009 WRI Global Congress on Intelligent Systems**. [S.l.: s.n.], 2009. v. 3, p. 329–333.