PUCRS

FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

WILLIAM SCHNEIDER

AVALIAÇÃO SISTEMÁTICA DE REDES *INTRACHIP*

Porto Alegre
2014

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

AVALIAÇÃO SISTEMÁTICA DE REDES *INTRACHIP*

WILLIAM SCHNEIDER

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

Orientador: Prof. Dr. Ney Laert Vilar Calazans

Porto Alegre

2014

Dados Internacionais de Catalogação na Publicação (CIP)

S359a Schneider, William

Análise sistemática de redes intrachip / William Schneider. – Porto Alegre, 2014.
152 f.

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

1. Informática. 2. Redes de Computadores. 3. Arquitetura de Redes. 4. Multiprocessadores. I. Calazans, Ney Laert Vilar. II. Título.

CDD 004.6

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

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

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "Avaliação Sistemática de Redes *Intrachip*" apresentada por William Schneider como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, aprovada em 13/03/2014 pela Comissão Examinadora:

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

Prof. Dr. César Augusto Missio Marcon – PPGCC/PUCRS

Prof. Dr. Fernando Gehm Moraes – PPGCC/PUCRS

Prof. Dr. Sandro Rigo – UNICAMP

Homologada em 23/04/2014, conforme Ata No. 006 pela Comissão Coordenadora.

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

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

AGRADECIMENTOS

Agradeço primeiramente à minha família, que foi de fundamental importância para que eu chegasse até aqui. Assim, muito obrigado Pai, Mãe, Mano e Mana.

Cabe também um agradecimento aos meus grandes amigos: Eduardo, Lucas e Ricardo (Grandes histórias desde 1994 :D). Ao pessoal do GAPH, especialmente o Madalozzo, o Ruaro, o GHeck, LHeck e Bruno pela parceria e ajuda no desenvolvimento deste trabalho, Foi uma honra fazer parte desse grupo.

Gostaria de agradecer também aos Professores Sandro e Marcon que aceitaram avaliar este trabalho e principalmente ao Professor Moraes que foi de extrema importância para o desenvolvimento do mesmo.

Por fim, um agradecimento especial ao meu orientador, pelos ensinamentos, pela ajuda, parceria e por ter aceitado me orientar em um trabalho de sistemas síncronos. Sei que o senhor prefere não-síncronos :-). Sou muito grato por ter sido seu orientando.

AVALIAÇÃO SISTEMÁTICA DE REDES *INTRACHIP*

RESUMO

O aumento no número de núcleos presentes em Sistemas Integrados em *Chip* tem proporcionado o projeto de circuitos com especificações cada vez mais agressivas. Arquiteturas de interconexão eficientes tais como as redes intrachip são fundamentais para a viabilidade destes projetos. Entretanto, medir e comparar o desempenho destas redesainda é uma tarefa desafiadora, resultado: (*i*) da complexidade imposta pela abundância de opções disponíveis no espaço de projeto destas redes; (*ii*) da atual não adoção de uma mesma plataforma de avaliação para a comparação de diferentes propostas de redes; (*iii*) e do fato de o tráfego de rede exercer uma influência muito maior do que qualquer característica de projeto no desempenho destas.

Este trabalho tem como principal objetivo estratégico a avaliação e comparação de diferentes arquiteturas de redes *intrachip* através de uma plataforma de avaliação unificada. Adota-se Nocbench, uma plataforma recente, já validada em alguns contextos e proposta como um padrão para a avaliação de redes *intrachip*. O método de avaliação empregado baseia-se na simulação de redes e utiliza como entrada modelos de tráfego e de computação descritos sob a forma de *traces*, ambos extraídos de aplicações reais. As principais contribuições do trabalho residem: (*i*) na proposta de diversas melhorias para a plataforma escolhida; (*ii*) no desenvolvimento de módulos para a integração das redes *Hermes HS*, *Hermes OO*, *Hermes TB*, *Hermes VC* e *YeaH*do grupo de pesquisa do Autor à plataforma em questão; (*iii*) no aprimoramento do processo de avaliação de desempenho da plataforma, através da inclusão de métricas comumente utilizadas para comparar redes *intrachip*, incluindo: latência, vazãoe *jitter*. Um conjunto de experimentos valida as contribuições e demonstra o uso da plataforma Nocbench como uma ferramenta útil na comparação de redes *intrachip* de origens diversas.

Palavras Chave: redes *intrachip*, NoC (*Networkon Chip*), Sistemas Integrados em Chip, avaliação de desempenho, *benchmarking*.

SYSTEMATIC EVALUATION OF INTRACHIP NETWORKS

ABSTRACT

The increase in the number of cores available in Systems on a Chip has enabled the design of circuits with increasingly aggressive specifications. Efficient interconnection architectures such as intrachip networks are critical to the viability of these projects. However, measuring and comparing performanceof these networks for a given system is still a challenging task, which results from: (*i*) the complexity imposed by the abundance of available options in the design space of these networks; (*ii*) the current non-adoption of a unique evaluation platform to compare different networks proposals; (*iii*) the fact that the network traffic has a greater influence on the performance of such networks than any other design characteristic.

This work has as main strategic goal the evaluation and comparison of different intrachip network architectures through the use of a unified evaluation platform. It adopts Nocbench, a recent platform, already validated in some contexts and proposed as a standard for the evaluation of intrachip networks. The employed evaluation method is based on the simulation of networks and uses as input traffic and computation models described in the form of traces, both extracted from real application. The main contributions of this work reside in: (*i*) the proposal of several enhancements to the chosen platform; (*ii*) the development of modules added to integrate the networks *Hermes HS*, *Hermes OO*, *Hermes TB*, *Hermes VC*, and *YeaH* from the author´s research group to the platform; (*iii*) the enhancement of the platform performance evaluation process, through the inclusion of metrics usually employed to compare intrachip networks, including: latency, throughput and jitter. A set of experiments validates the contributions and demonstrate the use the Nocbench platform as a useful tool in the comparison of intrachip networks of diverse origins.

Keywords: intrachip networks, NoC (Networks on Chip), SoC (Systems on a Chip), performance evaluation, benchmarking,

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API	*Application Programming Interface*
AV	*Audio Video*
CAD	*Computer Aided Design*
CPU	*Central Processing Unit*
ENSTA	*École Nationale Supérieure de Techniques Avancées*
EP	*Elemento de Processamento*
FDM	*Frequency-Division Multiplexing*
FPGA	*Field Programmable Gate Array*
GAPH	*Grupo de Apoio ao Projeto de Hardware*
HS	*Handshake*
ISS	*Instruction Set Simulator*
KTH	*Kungliga Tekniska Högskolan*
MBps	*Megabytes por segundo*
MCSL	*Mobile Computing System Lab*
MPEG4	*Moving Picture Experts Group 4*
MPSoC	*Multiprocessor System on Chip*
MWD	*Multi-Window Display*
NI	*Network Interface*
NoC	*Network-on-Chip*
OCP-IP	*Open Core Protocol International Partnership*
OFDM	*Orthogonal Frequency-Division Multiplexing*
OO	*On-Off*
OSCI	*Open SystemC Initiative*
PARSEC	*Princeton Application Repository for Shared-Memory Computers*
PDG	*Packet Dependency Graph*
PDG_GEN	*Packet Dependency Graph Generator*
RTL	*Register Transfer Level*

SoC	*System on Chip*
SPARCV8	*Scalable Processor ARChitecture V8*
SPLASH-2	*Stanford ParalleL Applications for SHared memory*
TB	*Torus Bidirectional*
TG2	*Transaction Generator 2*
TLM	*Transaction-Level Modeling*
UMTS	*Universal Mobile Telecommunication System*
VC	*Virtual Channel*
VHDL	*Very High Speed Integrated Circuit Hardware Description Language*
VOPD	*Video Object Plane Decoder*
XML	*eXtensible Markup Language*
YeaH	*Yet Another Hermes*

LISTA DE SÍMBOLOS

μt	*Média*
ud	*Média*
σt	*Desvio Padrão*
od	*Desvio Padrão*
λi	*Intervalo de Geração de Pacotes*

Sumário

1. Introdução

Com o constante crescimento da quantidade de núcleos presentes em Sistemas Integrados em *Chip* (do inglês, *Systems on Chip - SoCs*), estruturas de interconexão *intrachip* eficientes tornam-se cada vez mais importantes. Exemplos destas estruturas são as redes *intrachip*. As *Networks-on-Chip* ou NoCscomo também são conhecidas, têm recebido atenção substancial da comunidade científica nos últimos anos, devido as suas vantagens em relação aos tradicionais barramentos compartilhados, incluindo: escalabilidade, eficiência energética, e paralelismo de comunicação.

O projeto de uma rede intrachip requer a seleção de diversas características, incluindo topologia da rede, algoritmo de roteamento, esquema de controle de fluxo, estratégia de chaveamento, esquema de arbitragem, tamanhos de *flits* e *buffers* entre outras. Este vasto espaço de alternativas de projeto tem proporcionado uma abundância considerável de projetos de arquiteturas de NoCs distintas. No entanto, apesar do grande número de NoCs propostas [SAL07], medir e comparar o desempenho destas continua sendo um desafio [GRE07].

Este trabalho utiliza o termo *benchmark* para descrever o conjunto de aplicações, técnicas e dados associados, utilizado para avaliar o desempenho de um determinado recurso. Tradicionalmente, *benchmarks* têm sido empregados para medir o desempenho de CPUs, compiladores e ferramentas CAD (do inglês, *Computer-Aided Design*). Atualmente, estes vêm sendo propostos para avaliar o projeto de NoCs [SAL08]. Segundo Weiss [WEI02]*benchmarks* podem ser divididos em duas classes principais: sintéticos, e baseados em aplicações. Benchmarks sintéticos têm por objetivo medir apenas determinadas características de um sistema, processador etc., podendo imitar o comportamento de aplicações reais ou serem totalmente artificiais. São úteis na depuração de características específicas, mas não podem ser facilmente relacionados ao comportamento destas características em uma aplicação. Além disso, por visarem a avaliação de apenas um subconjunto de características, tendem a ser pequenos. *Benchmarks*

baseados em aplicações, por sua vez, são programas ou modelos que se assemelham a aplicações reais ou são baseados em tais aplicações e visam avaliar o sistema como um todo. Em virtude disso, geralmente possuem uma grande quantidade de código e de dados associados.

A utilização de *benchmarks* para a avaliação de NoCs pode ser realizada através de diversas abordagens. Segundo Hestness et al. [HES10], existem cinco métodos principais para atingir avaliações relevantes destes componentes: modelagem analítica (do inglês, *analytical modeling*), cargas de trabalho sintéticas (do inglês, *synthetic workloads*), simulação completa do sistema (do inglês, *full system simulation*), simulação da rede orientada a traces (do inglês, *trace-driven NoC simulation*) e simulação da rede orientada a dependências (do inglês, *dependency-driven NoC simulation*). A modelagem analítica possibilita uma avaliação rápida da rede, através do uso de modelos que abrangem desde energia [GUI08] e área, até padrões de tráfego de pior caso [TOW02]. No entanto, com o crescente dinamismo e complexidade dos sistemas com múltiplos processadores, a modelagem destes sistemas tem se tornado cada vez mais difícil. A metodologia mais utilizada para avaliação de NoCs é baseada em cargas de trabalhos sintéticas. Nesta abordagem, utiliza-se um simulador de rede em conjunto com padrões de tráfego sintéticos, tais como, aleatório-uniforme, matriz transposta, entre outros. Esta solução possibilita a obtenção de informações a respeito de métricas como vazão e latência com taxas de injeção variadas. Estes padrões, contudo, não correspondem diretamente ao comportamento de tráfego gerado por aplicações reais. Um método bastante preciso é a simulação completa do sistema. Esta abordagem utiliza aplicações reais e possibilita a identificação de gargalos na rede, bem como o impacto que possíveis otimizações de rede podem causar no desempenho das aplicações. Suas principais desvantagens estão no alto tempo de simulação e na variabilidade, uma vez que pequenas modificações na rede podem levar a resultados muito diferentes. Como alternativa à simulação completa do sistema, podem ser utilizados simuladores de rede baseados em *traces* de tráfego coletados a partir da execução de aplicações reais. Esta solução apresenta velocidades até 50 vezes superiores em comparação à simulação completa do sistema. O problema desta abordagem está no fato de ela não considerar as dependências entre as

mensagens da rede, o que acaba acarretando em intercalação de mensagens que jamais ocorreriam em uma simulação completa do sistema, o que pode levar a resultados enganosos. De modo a solucionar este problema, existe a abordagem de simulação baseada em dependências, que é semelhante à abordagem anterior, porém com o diferencial de considerar as dependências entre as mensagens da rede. Esta metodologia aumenta consideravelmente a precisão da avaliação da rede com pouco impacto na velocidade da simulação.

De modo a suprir a falta de benchmarks voltados para NoCs, a OCP-IP (do inglês, *Open Core Protocol International Partnership*) [OCP13] criou um grupo de trabalho voltado ao desenvolvimento de *benchmarks* para NoCs. Composto por representantes da academia e da indústria, incluindo Carnegie Mellon University e Sonics Inc. dos Estados Unidos, Tampere University of Technology da Finlândia, ENSTA da França, e University of British Columbia do Canadá, este grupo defende o desenvolvimento de *benchmarks* cujas aplicações sejam descritas de um modo mais abstrato, como por exemplo, sob a forma de grafos de tarefas com tempos e cargas de comunicação e computação, ao invés de códigos de aplicações [GRE07].

Apesar desta iniciativa, Salminen et al. [SAL09] após a análise de 140 publicações sobre NoCs em duas revisões do estado da arte [SAL07][SAL08] afirmam que a maior parte dos *benchmarks* utilizados em NoCs ainda são proprietários, pequenos e com pouca ou nenhuma documentação. Ao mesmo tempo, dependendo dos *benchmarks* e da configuração de rede utilizados, os resultados apresentam discrepâncias muito grandes entre si. Além disso, comparações entre redes de diferentes autores são praticamente inexistentes.

Com o intuito de atender a esses problemas, foi lançada em 2011 a plataforma Nocbench. Esta plataforma vem sendo desenvolvida na Universidade de Tampere e incorpora as ideias defendidas pela OCP-IP, tendo sido inclusive adotada pela mesma como plataforma padrão para o processo de *benchmarking* de NoCs. Ainda no mesmo ano foi lançada a suíte MCSL, um conjunto de benchmarks para NoCs desenvolvido pela Universidade de Hong Kong e contando com a participação de membros da Intel, e que acabou sendo integrada à plataforma Nocbench. Assim, com o intuito de obter avaliações justas, realistas e rápidas a respeito das NoCs desenvolvidas pelo grupo de pesquisa do qual o Autor deste

trabalho faz parte, bem como comparações destas com NoCs projetadas por outros grupos de pesquisa, optou-se pela adoção e consequente expansão desta plataforma.

1.1 Objetivo

O objetivo específico principal deste trabalho foi a expansão da plataforma de avaliação de NoCs Nocbench, de modo a possibilitar a avaliação de desempenho das redes *intrachip* desenvolvidas pelo grupo de pesquisa do Autor incluindo as redes: Hermes HS (do inglês, *Handshake*) [MOR04], Hermes OO (do inglês, *On-Off*),Hermes TB (do inglês, *Torus Bidirectional*) [SCH07],Hermes VC (do inglês, Virtual Channel) e YeaH (do inglês, *Yet another Hermes*), bem como a comparação destas com as demais NoCs já integradas à plataforma Nocbench. Três parâmetros de desempenho são considerados na avaliação: latência fim a fim, jitter de latência e vazão de aplicação.

1.2 Contribuições

As principais contribuições deste trabalho são as seguintes:

- Otimização e adaptação das descrições VHDL dasNoCs*Hermes HS*, *Hermes OO*, *Hermes TB*, *Hermes VC*, e *YeaH*, visando a inclusão das mesmas na plataforma *Nocbench*.
- Desenvolvimento de módulos para a integração dasNoCsH*ermes HS*, *Hermes OO*, *Hermes TB*, *Hermes VC*, e *YeaH*descritos em VHDL na plataforma.
- Automatização do processo de especificação da NoC a ser utilizada em uma simulação.
- Aprimoramento do processo de avaliação de desempenho da plataforma através da inclusão de parâmetros comumente utilizados no processo de avaliação de NoCs: latência fim a fim, *jitter de latência* e vazão de aplicação.

1.3 Organização do restante do documento

O Capítulo 2 apresenta uma discussão sobre trabalhos relacionados ao processo de *benchmarking* de NoCs. Isto inclui trabalhos que propõem conjuntos de *benchmarks* para NoCs, trabalhos sobre plataformas de *benchmarking* para NoCs e trabalhos que descrevem métodos para adaptar *benchmarks* clássicos, tais como PARSEC [BIE08] e SPLASH-2 [WOO05] para as estruturas de interconexão em questão. No Capítulo 3 introduz-se a arquitetura alvo deste trabalho. No Capítulo 4 apresentam-se as duas contribuições iniciais do trabalho, a automação do processo de gerar *wrappers* para NoCs e a adaptação das descrições de NoCs alvo do trabalho para facilitar sua integração à Nocbench. O Capítulo 5 descreve o processo de integração das NoCs alvo do trabalho ao ambiente Nocbench. O Capítulo 6aborda a validação do processo de integração, mostrando exemplos do processo de validação por simulação funcional. Em seguida, o Capítulo 7 mostra mais um conjunto de contribuições do trabalho, que consiste no uso da Nocbench para comparar o desempenho do novo conjunto de NoCs integradas à plataforma usando diversas métricas. Por fim, o Capítulo 8 apresenta um conjunto de conclusões sobre o tema principal desta Dissertação.

2. *Trabalhos Relacionados*

A metodologia mais precisa para avaliar projetos de NoCs utilizando aplicações é a simulação completa de um sistema [MAC13]. Utilizando essa metodologia, caso seja necessário comparar duas NoCs distintas, duas simulações completas do sistema devem ser executadas, uma para cada rede. Estas simulações incluem não apenas os modelos dos componentes da NoC em si, mas também o de cada elemento de processamento (tais como processadores, memórias compartilhadas, dispositivos de entrada e saída, etc.). A desvantagem é a lentidão, uma vez que esse processo pode levar semanas ou até mesmo meses, dependendo do nível de detalhe da simulação e do tempo de execução da aplicação, impossibilitando o projetista de explorar o espaço de projeto adequadamente [HES10].

Um método amplamente utilizado para contornar esse problema é a utilização dessa mesma abordagem apenas uma vez para obter traces do comportamento da aplicação e aplicá-los como entrada do modelo da NoC, usando um simulador de rede. Contudo, uma vez que os traces capturados por essa abordagem contêm apenas informações a respeito da ordem e dos tempos de transmissão dos pacotes [MAC11] (sem, no entanto considerar as dependências entre esses pacotes), essa solução pode acabar resultando em valores enganosos [HES10].

Em acordo com a ideia exposta nos parágrafos anteriores, este Capítulo aborda somente trabalhos que levam em consideração as dependências entre pacotes citadas.

2.1 Conjunto de *Benchmarks* MCSL

Liu et al. [LIU11] propõem um conjunto de benchmarks para a avaliação de NoCs denominado MCSL (do inglês, Mobile Computing System Lab). O MCSL consiste em padrões de tráfego e computação extraídos de aplicações reais. A primeira versão pública (1.1) desta suíte lançada em maio de 2011, conta com 8 aplicações para três topologias de rede: malha 2D, toro 2D, e árvore gorda, e

três tamanhos de rede: 4x4, 4x8, e 8x8. A segunda versão (1.5) lançada em janeiro de 2013 manteve três das oito aplicações da versão anterior e incorporou outras três. Além disso, os formatos de arquivos (ver Tabela 4 e Tabela 5) utilizados para a distribuição dos padrões de tráfego e computação que compõem o MCSL foi modificado. A terceira versão (1.6) foi lançada em janeiro de 2014 e trouxe duas novas aplicações em relação à versão 1.5. Os tamanhos de rede suportados pelas aplicações das versões 1.5 e 1.6 são apresentados na Tabela 1. As aplicações presentes na versão 1.1 e nas versões 1.5 e 1.6 são apresentadas respectivamente na Tabela 2 e na

A Figura 1 apresenta o fluxograma do processo de geração dos padrões de tráfego que compõem o MCSL. O conjunto de *benchmarks* em questão contém não só a direção e a quantidade de dados trocados entre as tarefas, como também as dependências temporais entre estas. O método para geração dos padrões utilizado tem como base um modelo de aplicação representado por um grafo de comunicação de tarefas, e um modelo de arquitetura, que captura os recursos de *hardware* de um MPSoC, tais como os elementos de processamento e a NoC.

Tabela 3.

Tabela 1 - Tamanhos de rede suportadas em cada topologia para as versões 1.5 e 1.6 do conjunto de benchmarks MCSL [MCS14].

Topologia	Tamanho
Malha 2D	2x2, 2x4, 3x3, 4x4, 5x5, 4x8, 6x6, 7x7, 8x8, 9x9, 10x10, 11x11, 8x16, 12x12,13x13, 14x14, 15x15, 16x16
Toro 2D	2x2, 2x4, 3x3, 4x4, 5x5, 4x8, 6x6, 7x7, 8x8, 9x9, 10x10, 11x11, 8x16, 12x12,13x13, 14x14, 15x15, 16x16
Árvore Gorda	4, 8, 16, 32, 64, 128, 256

Tabela 2 - Aplicações que compõem a versão 1.1 do conjunto de benchmarks MCSL [LIU11].

Aplicação	Descrição	Número de Tarefas
SAMPLE	Conversor de taxa de amostragem	612
H263E	Codificador H.263	201
H264DH	Decodificador com alta resolução H. 264	6343
H264DL	Decodificador com baixa resolução H. 264	403
ROBOT	Cálculo de controle dinâmico Newton-Euler para o manipulador Stanford com 6 graus de liberdade	88
FPPPP	Aplicação fpppp da suíte SPEC 95 - um programa de aplicação em química que executa derivadas integrais multi-electron	334

SATELL	Receptor de satélite	4515
SPARSE	Solucionador de matriz esparsa para simuladores de circuitos eletrônicos	96

A Figura 1 apresenta o fluxograma do processo de geração dos padrões de tráfego que compõem o MCSL. O conjunto de *benchmarks* em questão contém não só a direção e a quantidade de dados trocados entre as tarefas, como também as dependências temporais entre estas. O método para geração dos padrões utilizado tem como base um modelo de aplicação representado por um grafo de comunicação de tarefas, e um modelo de arquitetura, que captura os recursos de *hardware* de um MPSoC, tais como os elementos de processamento e a NoC.

Tabela 3 - Aplicações que compõem as versões 1.5 e 1.6 do conjunto de benchmarks MCSL [MCS14]. Apenas três das oito aplicações da versão 1.1 estão presentes na versão atual. As demais devem ser incluídas em versões posteriores.

Aplicação	**Descrição**	**Número de Tarefas**
RS-32_28_8_enc	Codificador de código Reed-Solomon com formato de palavra RS(32, 8, 8)	248
RS-32_28_8_dec	Decodificador de código Reed-Solomon com formato de palavra RS(32, 8, 8)	278
H264-720p_dec*	Decodificador de vídeo H.264 com resolução de 720p	2311
H264-1080p_dec*	Decodificador de vídeo H.264 com resolução de 1080p	5191
ROBOT	Cálculo de controle dinâmico Newton-Euler para o manipulador Stanford com 6 graus de liberdade	88
FPPPP	Aplicação fpppp da suíte SPEC 95 - programa de aplicação em química que executa derivadas integrais multi-electron	334
FFT-1024_complex	Transformada Rápida de Fourier com 1024 entradas de números complexos	16384
SPARSE	Solucionador de matriz esparsa para simuladores de circuitos eletrônicos	96

**Aplicações presentes apenas na versão 1.6.*

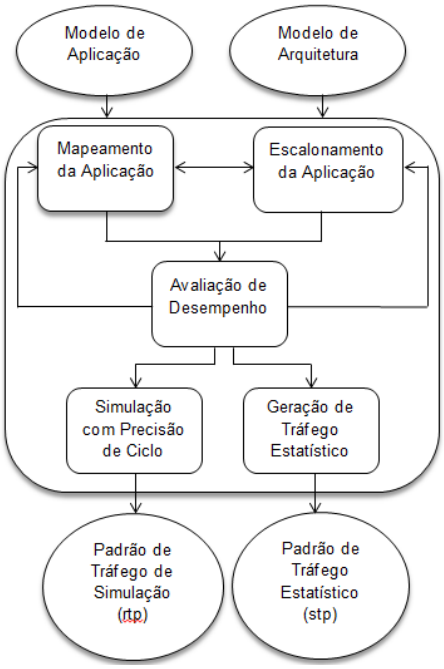

Figura 1 – Processo de geração de tráfego usado no conjunto de benchmarks MCSL [LIU11].

A geração de tráfego é realizada em cinco passos: (*i*) mapeamento, (*ii*) escalonamento da aplicação, (*iii*) avaliação de desempenho, (*iv*) simulação precisa em nível de ciclo e (*v*) geração de tráfego estatístico. Um desempenho não satisfatório na etapa (*iii*) pode levar o processo de geração de volta às etapas de mapeamento ou escalonamento, até que o requisito de desempenho esperado seja cumprido.

A estratégia de mapeamento de tarefas implementada utiliza uma política de balanceamento de carga e consiste na atribuição das tarefas da aplicação aos Elementos de Processamento (EPs) respeitando a ordem definida pelas relações de dependência do grafo de comunicação de tarefas da aplicação. O escalonamento é estático, isto é, as tarefas mapeadas em um determinado processador são escalonadas seguindo a ordem do mapeamento.

O resultado do processo de geração são duas versões de padrão de tráfego. A primeira denominada *recorded traffic pattern (rtp)*, obtida através de simulações com precisão de ciclo em SystemC, que contém *traces* de comunicação e computação precisos. E a outra, denominada *statistical traffic pattern* (*stp*), que utiliza fórmulas estatísticas para o cálculo dos tempos de execução, dos tamanhos dos dados produzidos pelas tarefas e dos intervalos de tem-

po relativos em que estes dados são agrupados em pacotes. Ambas as versões podem ser reutilizadas em NoCs com diferentes configurações mas com a mesma topologia.

O tempo de execução de cada tarefa corresponde ao pior caso e foi obtido com base no *profiling online* utilizando o simulador SimpleScalar.As latências dos pacotes trocados entre processadores variam de acordo com a configuração da rede utilizada. Uma vez que o padrão de tráfego *rtp* mantém as dependências entre os pacotes ao invés dos tempos exatos, *traces* deste padrão podem ser utilizados em NoCs com diferentes configurações mas com a mesma topologia, visto que todas as relações temporais podem ser reconstruídas corretamente.

Os resultados experimentais mostram que os padrões que compõem o MCSL apresentam uma média de 87,3% de diferença na latência dos pacotes em relação ao tráfego uniforme. A justificativa para essa grande diferença reside no fato de aplicações reais frequentemente possuírem concentrações locais de tráfego em certos períodos de tempo, o que não é observado em padrões de tráfego uniforme, que geram tráfego igualmente distribuído através da rede.

Cada padrão de tráfego (estatístico e obtido por simulação) de uma aplicação é distribuído sob a forma de um arquivo de *traces*. Existe um destes arquivos para cada conjunto composto por uma tupla *<p, a, t, d>*, onde *p* é o padrão de tráfego, *a* é a aplicação, *t* a topologia e *d* a dimensão da rede. Os formatos destes arquivos são apresentados na Tabela 4 e na Tabela 5.

Tanto o arquivo de *traces*estatístico quanto o arquivo de *traces* obtido por simulação são divididos em três partes. A primeira, que ocupa as primeiras cinco linhas de cada arquivo, contém o tipo do arquivo de *traces* (*rtp* ou *stp*), o número de EPs, linhas, colunas, tarefas, canais de comunicação da aplicação, tarefas iniciais e tarefas finais, além de listas de tarefas iniciais e de tarefas finais. No caso do arquivo *rtp*, existe ainda um campo denominado número de iterações, que representa o número de vezes que a aplicação será executada. Cabe ressaltar que o formato da versão 1.1 utiliza apenas os seguintes campos: *(i)* tipo de *trace*, *(ii)* número de EPs, *(iii)* número de tarefas e *(iv)* número de canais de aplicação e *(v)* número de iterações.

Tabela 4 - Formato de um arquivo de traces estatístico (stp) [MCS14].

Bloco de cabeçalho (5 linhas)							
Tipo de traço							
Topologia		Número de EPs		Número de linhas			Número de colunas
Número de tarefas		Número de canais de comunicação					
Número de tarefas iniciais		Lista de tarefas iniciais					
Número de tarefas finais		Lista de tarefas finais					
Bloco de execução da tarefa (uma linha para cada tarefa, cada uma das quais como a seguir)							
ID da tarefa		EP mapeado (ID/coordenada)		Número de sequência do escalonamento	*Média (μt)*		Desvio padrão (σt)
Bloco de comunicação da tarefa (uma linha para cada canal de comunicação, cada uma das quais como a seguir)							
ID do canal de comunicação	ID da tarefa fonte	ID da tarefa destino	Endereços iniciais de memória	Tamanho da memória	*Média (μd)*	Desvio padrão (σd)	Intervalo de geração de pacotes (λi)

A segunda parte dos dois formatos representa o bloco de execução de cada tarefa. Uma vez que o bloco de cada uma destas é descrito em uma linha, o número de linhas que compõem esta parte é igual ao número de tarefas. Os três primeiros campos são iguais em ambos os tipos de arquivos, e correspondem respectivamente ao identificador da tarefa, ao identificador do EP no qual esta tarefa foi mapeada, e ao número de sequência, que determina após que tarefa a mesma será escalonada no EP. Observe que no caso do arquivo rtp, há uma lista de números de sequência, uma vez que a aplicação pode ser configurada para ser executada mais de uma vez. Já o último campo presente no arquivo rtp corresponde a uma lista com os tempos de execução da tarefa em ciclos de relógio para cada uma das vezes em que ela será executada. No caso do tráfego estatístico, o tempo de execução segue uma distribuição Gaussiana. No arquivo stp, os dois últimos campos representam a média (μt) e o desvio padrão (σt) da Gaussiana.

Tabela 5 - Formato de um arquivo de traces obtido de uma simulação (rtp) [MAC13].

Bloco de cabeçalho (5 linhas)			
Tipo de traço			
Topologia	Número de EPs	Número de linhas	Número de colunas
Número de tarefas	Número de canais de comunicação	Número de iterações	
Número de tarefas iniciais	Lista de tarefas iniciais		
Número de tarefas finais	Lista de tarefas finais		
Bloco de execução da tarefa (uma linha para cada tarefa, cada uma das quais como a seguir)			
ID da tarefa	EP mapeado (ID/coordenada)	Número de sequência do escalonamento	Tempos de execução
Bloco de comunicação da tarefa (uma linha para cada canal de comunicação, cada uma das quais como a seguir)			
ID do canal de comunicação	ID da tarefa fonte	ID da tarefa destino	Endereços de memória	Tamanho das mensagens

A última parte contém o bloco de comunicação de cada tarefa. O número de linhas que compõem esta parte é igual ao número de canais de comunicação. Os três primeiros campos são semelhantes para os dois tipos de padrões de tráfego, e representam respectivamente os identificadores do canal de comunicação, da tarefa fonte e da tarefa destino. Na versão 1.1 há ainda mais dois campos: os identificadores do EP fonte e do EP destino. No arquivo rtp os dois últimos campos representam respectivamente os endereços de memória e o tamanho das mensagens em bytes. Já no tráfego estatístico, em vez de três, são utilizados cinco campos. Os dois primeiros contêm respectivamente o endereço inicial e o tamanho da memória. Já os últimos três campos correspondem à média (μd), ao desvio padrão (σd) e ao intervalo de geração de pacotes (λi) da distribuição Gaussiana.

Os anexos O e O apresentam respectivamente os arquivos de *traces* estatístico e obtido por simulação (de 20 execuções) para a aplicação Sparse em uma topologia malha e tamanho de rede 2x2.

2.2 Plataforma Padrão de Avaliação de NoCs Nocbench

A plataforma Nocbench [NOC13a] visa a avaliação de NoCs através da utilização de modelos de tráfego e computação de aplicações reais. Ela é composta pelos *traces* da versão 1.1 (*parser* da versão 1.5 encontra-se em fase de

desenvolvimento) da suíte MCSL descrita na Seção anterior e de um conjunto de 8 outros modelos de tráfego e computação derivados de aplicações reais [PEK11] apresentadas na Tabela 6, além de um simulador de rede e uma ferramenta para o monitoramento da simulação, denominadas respectivamente Transaction Generator 2 (TG2) e Execution Monitor [NOC13b].

Como se pode observar na Tabela 6, todas as aplicações pertencem às áreas de Multimídia e Telecomunicações. As aplicações MWD, VOPD e MPEG4 Decoder são aplicações de processamento de vídeo. A aplicação AV é composta por um codificador e decodificador de vídeo e um codificador e decodificador de áudio. A aplicação Equalizador de Canal é utilizada em receptores de rádio para corrigir distorções que podem ser causadas, por exemplo, pela reflexão do sinal de edifícios e veículos. Esta aplicação é muito usada em sistemas portáteis. Receptores UMTS (do inglês, Universal Mobile Telecommunications System) estão presentes na maioria dos celulares. A aplicação UMTS basicamente divide uma cadeia de dados serializada de entrada em cadeias paralelas. A aplicação OFDM (do inglês, Orthogonal Frequency Division Modulation) é utilizada, por exemplo, em transmissões de rádio e televisão e realiza multiplexação por divisão de frequência (FDM), ou seja, múltiplos sinais são enviados em diferentes frequências.

Tabela 6 - Aplicações adicionais da plataforma Nocbench.

Aplicação	Categoria	Número de Tarefas
MWD (Multiwindow Display)	Multimídia	12
VOPD (Video Object Plane Decoder)	Multimídia	12
Decodificador MPEG4	Multimídia	12
AV (Audio Video)	Multimídia	40
Equalizador de Canal	Telecomunicações	12
UMTS (Universal Mobile Telecommunications System) Receiver	Telecomunicações	7
Receptor OFDM (Orthogonal Frequency Division Modulation)	Telecomunicações	7
Ericsson Radio System	Telecomunicações	16

O número médio de tarefas presentes nas aplicações desta suíte é 15. Em média, cada tarefa comunica-se apenas com outras duas, sendo que em seis aplicações existem tarefas que se comunicam com no mínimo quatro outras. Quatro aplicações contêm uma tarefa enviando grande parte de todo o tráfego.

Segundo os autores, isso mostra que tráfego real está longe de ser uniforme e uma grande variação pode ser vista dentro de uma aplicação e entre diferentes aplicações.

Os modelos propostos foram derivados de experimentos presentes em trabalhos publicados na literatura. Estes trabalhos são devidamente referenciados no material de distribuição do Nocbench. As aplicações nem sempre apresentam um nível de detalhamento adequado. Assim, os proponentes do Nocbench estabelecem os seguintes pressupostos: *(i)* tarefas com mais de um canal de entrada são executadas quando ambos os canais tiverem recebido dados; *(ii)* canais bidirecionais são divididos em dois canais unidirecionais, com metade da largura de banda para cada; *(iii)* tarefas sem canais de entrada (também chaadas de tarefas iniciais) são disparadas por temporizadores.

As aplicações que compõem a suíte Nocbench originalmente propostas por Pekkarinen et al. [PEK11] foram especificadas em descrições XML, seguindo o estilo de modelagem proposto por Salminen et al. [SAL09]. Neste estilo, especifica-se em um arquivo XML tanto o modelo de aplicação, quanto a plataforma em que esta aplicação será executada, o mapeamento das tarefas que compõem a aplicação, os parâmetros de avaliação, além de detalhes relativos à simulação. O arquivo XML em questão divide-se em quatro Seções: *(i)* Aplicação; *(ii)* Mapeamento; *(iii)* Plataforma; e *(iv)* Restrições.

O simulador TG2 foi escrito em C++ e possibilita avaliar NoCs descritas nas linguagens SystemC e VHDL, em diferentes níveis de abstração. O TG2 não é um simulador de conjunto de instruções ou um simulador com precisão em nível de ciclo mas, fornece precisão suficiente para uma análise de desempenho inicial de NoCs. O modelo de computação do simulador assemelha-se a redes de processos Kahn, mas pode ser estendido. O simulador também permite a modelagem de cache misses e de comunicação por memória compartilhada, uma vez que conta com um modelo preciso de memória, desenvolvido no KTH [KTH13].

Por ser usada no presente trabalho, a plataforma Nocbench será abordada em mais detalhe no Capítulo 3.

2.3 Plataforma de Avaliação de NoCs NoCBench

Mandal et. al [MAN09] propõem uma plataforma semelhante à descrita na Seção 2.2, inclusive no nome. NoCBench é a denominação desta plataforma, que consiste de um simulador de NoCs com precisão de *flit* denominado NoCSim e um conjunto de benchmarks sintéticos e baseados em aplicações.

O simulador NoCSim foi escrito em SystemC e projetado para a simulação de NoCs descritas em nível comportamental. Quatro unidades compõem o simulador: *(i)* gerador de NoC, que realiza a leitura de um arquivo de configuração especificado em XML e gera a NoC, utilizando os módulos com as descrições SystemC da rede; *(ii)* biblioteca de componentes da rede, que contém as descrições da NoC em SystemC; *(iii)* biblioteca de núcleos configuráveis, que provê modelos de EPs descritos em SystemC e *(iv)* motor de simulação, que é o motor OSCI SystemC.

Dois tipos de núcleos são fornecidos: *(a)* núcleos sintéticos, que geram dados com base em distribuições estatísticas de tráfego ou em grafos de comunicação de tarefas; *(b)* núcleos reais, tipicamente implementados em C/C++ com wrappers SystemC. Exemplos destes núcleos são ISSs de diferentes processadores. Atualmente, o simulador conta com o ISS SPARCV8 descrito em ArchC e modelos soft de memória e cache L1 e L2. Através destes núcleos é possível executar aplicações reais. Segundo os autores, a plataforma foi testada com aplicações das suítes MiBench e MediaBench.

2.4 Biblioteca para Captura de Dependências entre Mensagens de Rede Netrace

Hestness et al. [HES10] propõem uma biblioteca que captura as dependências entre mensagens de rede obtidas em simulações completas de sistema de aplicações multithread denominada Netrace. A abordagem utilizada consiste em construir um grafo acíclico dirigido entre mensagens de rede, baseado no ordenamento e nas dependências entre transações de memória gravadas durante uma simulação completa do sistema. A informação de dependência é armazenada junto com os dados do pacote no trace de rede.

Netrace é composto por um conjunto de funções para manipulação de arquivos de tráfego (traces), extração de pacotes destes arquivos, e rastreamento das dependências destes pacotes. Para a obtenção destes traces, as aplicações da suíte PARSEC v2.1 foram executadas no simulador M5 [BIN06] sob uma versão modificada do Linux 2.6.27. As mesmas foram modeladas para representar um sistema com 64núcleos,cuja configuração é apresentada na Tabela 7.

Tabela 7 - Configuração do sistema alvo para a coleta de traces do Netrace [HES10].

Cores	64 on-chip, in-order, Alpha ISA, 2GHz
L1 cache	32KB instruction/32KB data, 4-way associative, 64B lines, 3 cycle access time, MESI coherence protocol
L2 cache	64 bank fully shared S-NUCA, 16MB, 64B lines, 8-way associative, 8 cycle bank access time
Memory	150 cycle access time, 8 on-chip memory controllers

O sistema simulado possui a mesma estrutura denúcleos e *caches* do sistema alvo (Tabela 7). Porém, os núcleos foram conectados através de barramentos sem contenção. O objetivo com isso é evitar qualquer dependência da realização física do processador e eliminar qualquer atraso ou contenção artificiais nos *traces* de rede. A fim de modelar esse comportamento, o sistema simulado utiliza uma *cache* L2 unificada, diferentemente do sistema alvo, que utiliza um banco de *caches* distribuídas. Decisão semelhante foi utilizada na modelagem da memória, que também foi unificada. Na etapa de pós-processamento, fontes e destinos dos pacotes são mapeados para os bancos de *cache* ou controladores de memória em bases de endereços intercalados. Esse mapeamento pode ser modificado pelo simulador de rede. Para manter a abstração do sistema simulado, define-se um único valor para modelar a latência de acesso entre as *caches*. A latência de rede também foi fixada. Este valor foi computado através da utilização de um modelo analítico que define o número médio de saltos, que gera a latência média do pacote. Obtém-se isto através de uma NoC com uma topologia malha, utilizando a distância entre dois pontos da rede (*taxi-cab distance*) e assumindo não haver contenção. Fixou-se o tempo que um roteador leva para processar um pacote em 4 ciclos de relógio.

Após a coleta da comunicação *cache-to-cache*, realiza-se a etapa de pós-processamento, responsável por interpretar os *traces* e construir pacotes de rede a partir deles. Em seguida, realiza-se a detecção e rastreamento das dependências entre estes pacotes. Um exemplo do processo é apresentado na Figura 2 e na Figura 3.

```
Format:
<inject_cycle>: <bus>: src <port_id>
          dst <port_id> <type> <addr>

A 36: sys.tol2bus: src 96 dst -1 ReadReq  0xd040
B 50: sys.tol2bus: src 46 dst -1 ReadReq  0xdb40
C 60: sys.tol2bus: src 16 dst 96 ReadResp 0xd040
D 74: sys.tol2bus: src 16 dst 46 ReadResp 0xdb40
```

Figura 2 – Exemplo de traço coletado [HES10].

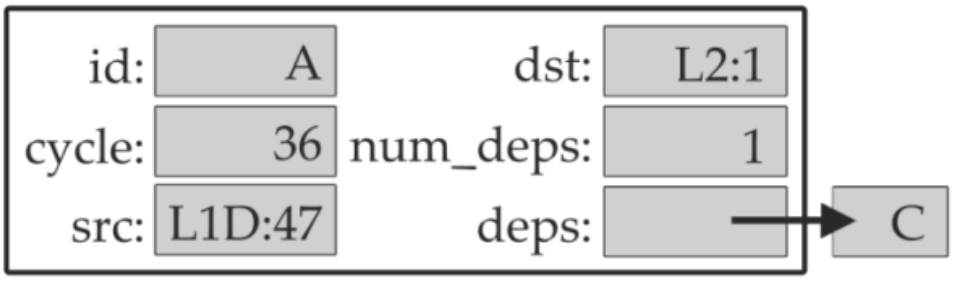

Figura 3 – Exemplo de pacote construído a partir do traço da Figura 2[HES10].

Observe que o ciclo do pacote é obtido diretamente do traço. A fonte e o destino são traduzidos utilizando o mapeamento definido na simulação, que nesse caso vinculou a fonte 96 à *cache* de dados L1 no nodo 47. O destino -1 indica que o pacote foi enviado em modo *broadcast*. O traçoC contém a reposta ao traçoA, que é originada na fonte 16, *cache* L2.

Quando o traçoC é interpretado, a aplicação Netrace pesquisa os *trace-s*interpretados anteriormente por pacotes de que C depende. No exemplo, C depende de A; assim o número de dependências do pacote A é incrementado e C é adicionado à lista de dependências de A.

Qualquer parâmetro de configuração da rede pode ser modificado, sem que para isso seja necessário alterar os *traces*utilizados. A alteração da frequência dos processadores pode ser realizada através do ajuste dos *timestamps* das requisições de acesso a memória.

2.5 Técnica para Identificação e Inclusão de Dependências entre Mensagens de Rede PDG_GEN

Macdonald et al. [MAC13] propõem PDG_GEN, uma técnica baseada em inferências para identificar e incluir dependências em traces. Estes tracessão obtidos através de múltiplas simulações sistêmicas para uma dada aplicação.

Para cada simulação sistêmica utiliza-se uma configuração de rede diferente. Cada simulação apresenta informações sobre relações de dependências entre diferentes pacotes, que podem ser extraídas através da comparação dos traces. Com essa abordagem cria-se um grafo de dependências de pacotes (PDG - Packet Dependency Graph), que é essencialmente um conjunto de traces estendido com informações de dependência. Esse PDG pode ser utilizado como entrada para simuladores de rede.

O primeiro traço gerado, denominado traço base, é obtido através da execução da aplicação em uma rede totalmente conectada com um único ciclo de latência. Com base nesse traço, particionam-se os nodos da rede no número de conjuntos desejado, sendo que os pares de nodos com a maior quantidade de comunicação entre si são colocados em conjuntos diferentes. Por exemplo, considerando uma rede com quatro nodos: A, B, C, D, com um volume de comunicação muito grande entre A e B e entre C e D, e com um valor de particionamento igual a 2, os conjuntos gerados serão obrigatoriamente: conjunto 0 = {A, C}, conjunto 1 = {B, D}. Com estes dois conjuntos, duas simulações adicionais são executadas. Na primeira, a rede é modelada com todos os canais de saída do conjunto 0 (A->B, A->D, C->B, C->D) tendo uma latência alta (representada por um valor maior que 1), e com os demais canais deste conjunto tendo uma latência baixa (representada por um valor igual a 1). O mesmo ocorre para o conjunto 1. Essas latências servem para expor informações sobre dependências entre pacotes. Alguns pacotes serão atrasados enquanto aguardam pelas dependências dos nodos mais lentos, enquanto outros, que não possuem tais dependências, não serão retardados.

O algoritmo PDG_GEN utiliza o traço base e os traces adicionais para gerar um conjunto de dependências e um tempo de computação para cada pacote. O

algoritmo é dividido em três etapas. Na primeira, para cada evento de transmissão em cada um dos traces, todas as dependências potenciais são adicionadas a esse evento. Na etapa seguinte, removem-se todas as dependências que violarem o tempo de transmissão de um evento, ou seja, todas as possíveis dependências que chegarem após um evento de transmissão ter acontecido em qualquer um dos traces. Na última etapa, o algoritmo determina o tempo de computação associado a cada evento de transmissão. Mais detalhes a respeito desta proposta podem ser encontrados em [MAC11].

2.6 Discussão dos Trabalhos Apresentados

As técnicas propostas por Hestness [HES10] e Macdonald [MAC13] apresentam muitas semelhanças. No entanto, uma vez que a primeira utiliza traces provenientes de apenas uma única simulação, não se pode nesta detectar dependências complexas entre referências de memória, só identificáveis através de múltiplas simulações completas de sistema [MAC11].

Apesar de uma simulação baseada em traces ser muito mais rápida do que uma simulação completa de sistema, a primeira também apresenta algumas desvantagens, dependendo da abordagem de obtenção destes traces. Os traces obtidos nas propostas de Hestness e Macdonald, por exemplo, foram derivados de simulações completas de sistema, e por isso possuem um nível de detalhamento muito grande, o que acaba refletindo em seu tamanho, que em ambas as abordagens chega à ordem de Gigabytes, dificultando desse modo, a distribuição destas soluções.

Os traces que compõem a proposta de Liu et al. [LIU11] foram obtidos de simulações com precisão de ciclo em SystemC utilizando uma plataforma de simulação de NoCs ao invés de um simulador completo de sistema. O resultado disso é um conjunto de traces com um tamanho consideravelmente menor, cerca de 100 MB, porém ainda assim, significativamente volumoso.

A proposta de Salminen et al. [SAL09] e Pekkarinen et al. [PEK11] oferece flexibilidade quanto ao mapeamento das tarefas, uma vez que a mesma consiste de um conjunto de modelos de computação e comunicação e não de traces

obtidos para um determinado mapeamento, como ocorre nas três propostas discutidas anteriormente.

Ambas as plataformas propostas por Salminen et al. [SAL09] e Mandal et al. [MAN09] baseiam-se na iniciativa de padronização do processo de benchmarking de NoCs proposto pela OCP-IP [GRE07]. As principais vantagens da primeira residem: *(i)* na quantidade e representatividade das aplicações, uma vez que tanto a suíte proposta por Liu et al. quanto a suíte proposta por Pekkarinen et al. encontram-se integradas a esta plataforma; e *(ii)* no estilo de modelagem adotado, visto que, o próprio Malave [MAL10] um dos autores da proposta de Mandal et al. reconhece a necessidade de especificações mais robustas de dependências entre comunicações, assim como modelos de aplicações mais representativas, já que grande parte das aplicações da última não possui mais do que quatro tarefas. Além disso, ainda segundo Malave [MAL10], na abordagem de Mandal et al. não há a possibilidade de se especificar cargas de processamento em nível de tarefa como dependência para que a comunicação ocorra. Em contrapartida, a abordagem proposta por Mandal et al. [MAN09] oferece a possibilidade de avaliar NoCs através da execução de aplicações reais, tais como as presentes nas suítes MiBench [GUT01] e MediaBench [LEE97]. Apesar desta possibilidade, o uso de modelos permite uma avaliação de desempenho mais rápida e segundo Pekkarinen et al., com uma taxa de erro aceitável, abaixo de 10%, tendo como base a avaliação de NoCs em FPGAs através da ferramenta Traffic Generator [NOC13c].

Em [MAL10] Malave afirma que diversos mapeamentos de uma mesma aplicação resultam em sistemas com diferentes características, criando assim diferentes benchmarks. Com base nesta ideia, a adesão no presente trabalho à proposta de Salminen et al. [SAL09] torna-se ainda mais interessante, devido à presença da suíte proposta por Pekkarinen et al. [PEK11].

Por fim, a adoção da plataforma Nocbench pela OCP-IP como ferramenta padrão para o processo de benchmarks de NoCs, aparece como outro aspecto decisivo para a escolha desta plataforma como base para a realização do trabalho aqui proposto.

3. Arquitetura Alvo

Conforme mencionado anteriormente, Salminen et al.propõem um estilo de modelagem para o processo de avaliação de NoCs, que juntamente com o simulador TG2 e os *benchmarks* propostos por Liu et al. e Pekkarinen et al., formam a plataforma Nocbench. Este Capítulo aborda esta plataforma em mais detalhes. Primeiro apresenta-se o estilo de modelagem proposto, utilizando para isso a aplicação AV (ver descrição na Tabela 6) e em seguida o simulador TG2, responsável pela interpretação de e por exercitar esta modelagem. Cabe salientar que as alterações realizadas sobre a plataforma durante este trabalho não são abordadas neste Capítulo, sendo alvo do Capítulo 4

3.1 Estilo de Modelagem

Seis itens da modelagem são especificados em um mesmo arquivo XML: *(i)* a descrição da aplicação, *(ii)* a plataforma de processamento, *(iii)* o mapeamento das tarefas da aplicação, *(iv)* os parâmetros de avaliação, *(v)* os detalhes relativos à simulação, e a *(vi)* NoC. Os três primeiros itens são descritos respectivamente nas Seções: Aplicação, Mapeamento e Plataforma, e os três últimos, na Seção Restrições. Cada aplicação é descrita por um arquivo XML específico. A seguir detalha-se as quatro Seções que compõem este arquivo.

3.1.1 Seção Aplicação

A Figura 4 mostra a estrutura básica da descrição XML da Seção Aplicação. Toda aplicação é descrita dentro da *tag application*. A *tag task_graph* contém as tarefas da aplicação. Uma tarefa é composta por dois atributos básicos: nome e identificador. Após a descrição das tarefas, declaram-se as conexões entre elas, através da *tag port_connection.* Por fim, a *tag event_list* contém os eventos responsáveis pela a inicialização da aplicação.

```
<application>
        <task_graph>

                <task name = "nome_da_tarefa" id = "id_da_tarefa" >
                </task>

                ...

        </task_graph>

        <port_connection src = "porta_de_saida_da_tarefa_fonte"
                         dst = "porta_de_entrada_da_tarefa_destino" />

        ...

        <event_list>
                <event out_port_id = "id_da_porta_de_saida_do_evento"
                amount = "quantidade_de_bytes_a_serem_enviados"
                name = "nome_do_evento"
                id = "id_do_evento" />

                ...

        </event_list>
</application>
```

Figura 4 - Estrutura básica da Seção Aplicação do arquivo de modelagem XML do Nocbench.

A Figura 5mostra o grafo de comunicação de tarefas da aplicação AV.

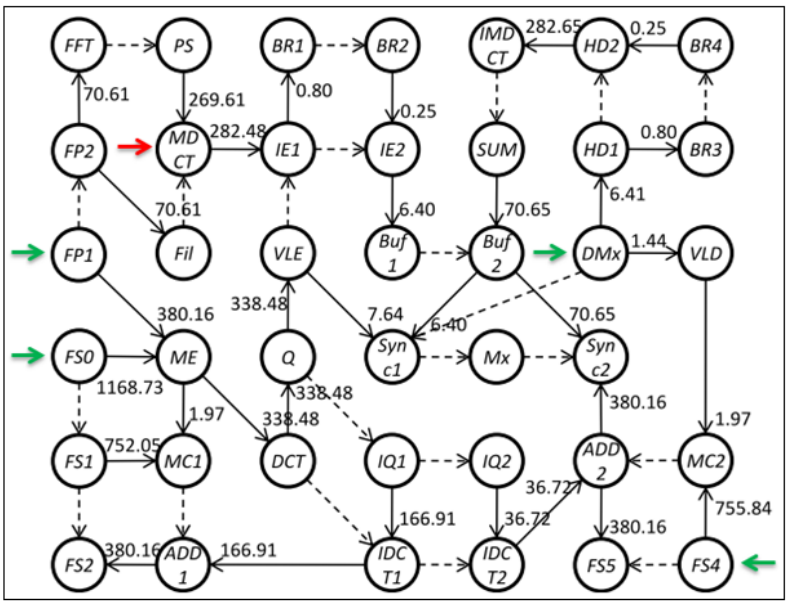

Figura 5 - Grafo de comunicação de tarefas da Aplicação AV [PEK11]. A seta vermelha indica a localização da Tarefa MDCT no grafo. Observe que esta tarefa contém duas portas de entrada, pelas quais se recebe de dados enviados pelas Tarefas PS e Fil; e uma porta de saída, pela qual se enviam dados à Tarefa IE1. As setas verdes indicam a localização dos eventos (FP1, FS0, DMx e FS4) responsáveis pela inicialização da aplicação. Os arcos tracejados representam a comunicação entre tarefas presentes em um mesmo EP, enquanto que os demais representam a comunicação entre tarefas localiza-

das em EPs distintos. Os valores nos arcos do grafo representam a taxa de envio de dados das tarefas, em Megabytes/s (MBps).

Em virtude da descrição desta aplicação ser bastante extensa, usa-se como ilustração apenas a tarefa MDCT desta aplicação, cujo XML aparece na Figura 6. Após a definição do nome (name) e do identificador (id) da tarefa, apresenta-se a interface de entrada e saída desta. Conforme a Figura 5, a Tarefa MDCT possui duas portas de entrada (in_port) e uma porta de saída (out_port), conectadas respectivamente às portas de saída das Tarefas PS e Fil e à porta de entrada da Tarefa IE1.

```
<task name = "MDCT" id = "39" >
    <in_port id = "3936" />
    <in_port id = "3938" />
    <out_port id = "3913" />
    <trigger dependence_type = "and" >
        <in_port id = "3936" />
        <in_port id = "3938" />
        <exec_count>
            <op_count>
                <int_ops>
                    <polynomial>
                        <param value = "1" exp = "0" />
                    </polynomial>
                </int_ops>
            </op_count>
            <send out_id = "3913" prob = "1" >
                <byte_amount>
                    <polynomial>
                        <param value = "8828" exp = "0" />
                    </polynomial>
                </byte_amount>
            </send>
            <next_state value = "READY" />
        </exec_count>
    </trigger>
</task>
```

Figura 6 – Descrição XML da tarefa MDCT da aplicação AV, na Seção Aplicação do arquivo.

Uma tarefa pode conter vários fluxos de execução. Condiciona-se o início da execução de um determinado fluxo ao recebimento de uma mensagem. Na modelagem proposta, um fluxo de execução é modelado através de uma *tag trigger*. O atributo *dependence_type* desta *tag* possibilita definir se a execução de um determinado fluxo depende do recebimento de uma mensagem em todas, algumas ou em apenas uma das portas de entrada especificadas logo após a *tag trigger*. Na tarefa MDCT foi especificado o valor "and" para este atributo. Deste modo, a execução desta tarefa só é iniciada após o recebimento de uma mensagem em cada uma de suas portas de entrada. Caso o valor do atributo *dependence_type*

fosse “or”, a execução teria início assim que uma das portas recebe uma mensagem.

Especificam-se a quantidade de operações executadas e o número de *bytes* enviados em um determinado fluxo de execução dentro da *tag exec_count* através das *tags op_count* e *send*. Na primeira (*op_count*) se definem *os tipos* (com números inteiros *<int_ops>*, com números em ponto flutuante *<float_ops>*ou coma memória <mem_ops>) e *a quantidade de operações* a serem executadas pelo EP ao qual esta tarefa será mapeada. Na outra (*send*), definem-se *os tipos* e a quantidade de dados a serem enviados. Tanto a quantidade de operações quanto a quantidade de dados podem ser definidas com um valor fixo, através da *tag polynomial* (isto ocorre na Tarefa MDCT), ou com um valor aleatório dentro de um intervalo estipulado, como a Figura 7 ilustra. Neste caso, este valor é gerado com base no atributo *value* da *tag rng_seed* da Seção Restrições (*Constraints*).

Conforme a Figura 6, a Tarefa MDCT realiza uma operação com números inteiros e em seguida envia 8828 *bytes* para a porta de saída 3913, com uma probabilidade de envio igual a 100%. Caso a probabilidade especificada seja menor que 100%, usa-se a *tag rng_seed* para a geração de um número aleatório; caso este seja menor que a probabilidade, o envio é realizado. Por exemplo, considerando que uma tarefa seja executada 4 vezes e sua probabilidade de envio seja de 50%, o envio de dados será realizado em duas execuções. O intervalo entre estes envios é determinado pelo número aleatório gerado, e afeta diretamente a latência de rede (apesar de a quantidade de dados enviados após a simulação ser a mesma).

```
<distribution>
        <uniform min = "30" max = "90" />
<distribution>

<distribution>
        <normal mean = "50" standard_deviation = "5" />
<distribution>
```

Figura 7 – Sintaxe utilizada para especificar a quantidade de operações a serem executadas. Os dois conjuntos de tags distribution apresentam respectivamente a sintaxe utilizada para a geração de um valor randômico entre 30 e 90 com distribuição uniforme e um valor randômico com base em uma distribuição Gaussiana.

A Figura 5 mostra que a taxa de envio de dados da tarefa MDCT é igual a 282,48 MB/s. Até agora vimos que esta aplicação envia apenas 8828 *bytes*. No entanto, ainda na Seção Aplicação é possível definir o número de vezes que cada tarefa será executada. Para isto, utiliza-se a *tag event_list*.

O bloco de eventos da aplicação AV aparece na Figura 8. Através de eventos é possível regular a taxa de injeção da aplicação. Observe-se que foram definidos eventos para quatro tarefas do grafo (indicadas por setas verdes) que ativam a execução de todas as tarefas da aplicação. O atributo *period* representa o intervalo em segundos entre disparos de eventos sucessivos do mesmo tipo. Nesse exemplo, foi definido que cada tarefa será executada a cada 0,00003125 segundos, ou seja, 32000 vezes por segundo (1 / 0,00003125). Também foi definido que cada vez que um evento for disparado, um *byte* (especificado pelo atributo *amount* = "1") será enviado à tarefa conectada a esse evento. Multiplicando-se os 8828 *bytes* enviados pela tarefa a cada execução por 32 kHz obtêm-se a taxa de 282 MB/s.

Um atributo útil na caracterização da taxa de injeção de uma determinada aplicação é o atributo *count*. Através dele pode-se especificar a quantidade de vezes que uma aplicação executada. Para impedir que uma aplicação seja executada mais do que uma vez, por exemplo, basta inserir este atributo com o valor 1 em todos os eventos presentes na aplicação.

```
<event_list>
    <event out_port_id = "1000" amount = "1" name = "FP0_trigger"
    id = "1" period = "0.00003125" prob = "1" />

    <event out_port_id = "90" amount = "1" name = "FS0_trigger"
    id = "2" period = "0.00003125" prob = "1" />

    <event out_port_id = "210" amount = "1" name = "DMx_trigger"
    id = "3" period = "0.00003125" prob = "1" />

    <event out_port_id = "240" amount = "1" name = "FS4_trigger"
    id = "4" period = "0.00003125" prob = "1" />
</event_list>
```

Figura 8 – Lista de eventos da aplicação AV.

Por fim, ainda na Seção Aplicação é realizada a conexão entre as tarefas. Para isso, utiliza-se a *tag port_connection*. As ligações das tarefas Fil e PS com a tarefa MDCTsão feitas respectivamente através das *tags*:

<port_connection src = "3639" dst= "3936"/> e *<port_connection src="3839" dst="3938"/>*. Já a ligação da tarefa MDCTcom a tarefa IE1é realizada através *da tag:* <port_connection src = "3913" dst = "1339"/>. Os atributos *src* e *dst* de cada *tag* referem-se respectivamente à porta de saída e à porta de entrada das tarefas comunicantes.

3.1.2 Seção Mapeamento

A Seção Mapeamento liga tarefas a grupos de tarefas ou diretamente a recursos de *hardware*. Os atributos *content* e *position* são utilizados na versão gráfica do simulador para determinar se grupos podem ser movidos para diferentes recursos ou para alterar os conteúdos dos grupos através da movimentação das tarefas de um grupo a outro. O código XML que especifica o mapeamento da tarefa MDCT é apresentado naFigura 9.

```
<mapping>
    <resource name = "DSP6" id = "2" contents = "mutable" >
        <group position = "movable" name = "g3" id = "2"
        contents = "mutable" >
            <task position = "movable" name = "Filter"
            id = "36" />
            <task position = "movable" name = "MDCT"
             id = "39" />
            ...
        </group>
        ...
    </resource>
    ...
</mapping>
```

Figura 9 - Código XML que explicita o mapeamento da tarefa MDCT da aplicação AV. Apresenta-se apenas a parte do mapeamento em que a tarefa MDCT encontra-se inserida.

Observe-se que nesse exemplo de mapeamento pode-se alterar o conteúdo do recurso e do grupo. Também se pode mover o grupo para outro recurso, a tarefa para outro grupo ou até mesmo manter uma tarefa sem grupo.

3.1.3 Seção Plataforma

Na Seção Plataforma definem-se os tipos de recursos de *hardware*. A Figura 10 apresenta como exemplo a descrição do recurso *DSP6*. Além do identifica-

dor, nome e frequência de operação do recurso (em MHz), há também um atributo *type*, que mapeia este recurso para uma biblioteca externa, também descrita em XML e apresentada na Figura 11. Esta Figura define características como o custo da comunicação entre tarefas do mesmo grupo, de grupos diferentes e entre tarefas de diferentes EPs. Também define a quantidade de operações por ciclo para aquelas que o recurso é capaz de executar tais como: operações sobre números de ponto flutuante, sobre números inteiros, e sobre memórias. Além disto, ela também informa dados de consumo de energia, área, além das frequências de operação mínima e máxima para o recurso. Ainda na descrição do recurso definem-se os tamanhos dos *buffers* de entrada *(rx_buffer)* e de saída(*tx_buffer*)e o tamanho do pacote utilizado pelo recurso.

```
<platform>
    <resource_list>
        </resource>
            <resource id = "2" name = "DSP6"
            frequency = "500" type = "CPU_TYPE_1"
            rx_buffer_size = "0" tx_buffer_size = "0"
            packet_size = "8" >
            ...
        </resource>
    </resource_list>
</platform>
```

Figura 10 - Exemplo de Seção Plataforma. Mostra-se apenas a descrição do recurso no qual a tarefa MDCT foi mapeada.

As configurações do EP do tipo CPU_TYPE_3 apresentadas na Figura 11 mostram que não há nenhum custo adicional para o envio de uma mensagem a uma tarefa do mesmo grupo, mas são gastos 1675 ciclos adicionais para o recebimento.

```
<processing_element type= "CPU_TYPE_3" class="general" int_ops="2" float_ops="2"
mem_ops="1" power="1"  area="15000" min_freq="100" max_freq="1000">

 <communication_costs>

    <intragroup>
       <send> <polynomial> <param exp="0" value="0"/> </polynomial> </send>
       <receive> <polynomial> <param exp="0" value="1675"/> </polynomial> </receive>
    </intragroup>

    <intergroup>
       <send> <polynomial> <param exp="0" value="1956"/> </polynomial> </send>
       <receive> <polynomial> <param exp="0" value="8693"/> </polynomial> </receive>
    </intergroup>

    <interpe>
      <send> <polynomial> <param exp="0" value="7404"/> </polynomial> </send>
      <receive>  <polynomial> <param exp="0" value="15285"/> </polynomial> </receive>
    </interpe>

 </communication_costs>
</processing_element>
```

Figura 11 – Biblioteca externa ao modelo. Define informações adicionais relativas aos diferentes tipos de EPs.

Na comunicação entre tarefas de grupos diferentes há tanto uma sobrecarga de processamento (*overhead*) para o envio quanto para o recebimento. No caso de comunicação entre tarefas de diferentes EPs, o número de ciclos adicionais que o processador gasta é ainda maior. Estes custos adicionais visam representar o tempo que o sistema operacional gasta com o chaveamento de contextos e as chamadas de função necessárias para que a comunicação ocorra.

3.1.4 Seção Restrições

A última Seção define os detalhes da simulação, tais como a unidade utilizada nos arquivos de log, o tempo de simulação, o intervalo entre medições, o local onde os arquivos de log serão armazenados e os parâmetros de avaliação para os quais se deseja resultados. Aqui também se especifica a NoC a ser utilizada. A Figura 12 apresenta um exemplo de uma descrição desta Seção.

```
<constraints>

    <noc class = "mesh_2d" type = "vhd" subtype =
    "4x4" />
     <pe_lib file= "examples/pe_lib.xml" />

    <rng_seed  value = "42" />

     <sim_resolution  time = "1.0"   unit = "ns" />
     <sim_length      time = "150"  unit = "ms" />
     <measurements  time = "1.0"   unit "ms" />

    <log_exec_mon  file = "log_execmon.txt" />
    <log_packet      file = "log_packet.csv" />
    <log_token       file = "log_token.csv" />
    <log_summary   file = "log_summary.csv" />
    <log_pe          file = "log_pe.csv" />
    <log_app         file = "log_app.csv" />

    <cost_function   func = "parametro_de_avaliacao " />

</constraints>
```

Figura 12 – Exemplo de Seção Restrições para o estilo de modelagem proposto.

Note que a especificação da NoC sobre o qual a aplicação executa é feita via três atributos: *(i) class*, que é utilizado para a seleção de uma dentre as diferentes redes integradas à plataforma; *(ii) type*, uma vez que podem existir várias redes de uma mesma classe, este atributo é utilizado para a seleção de uma dentre estas redes *(iii) subtype*, especifica a organização dos roteadores que compõem a NoC. Em topologias malha 2D e toro 2D utiliza-se o formato *<número_de_roteadores_no_ei-xo_x>X<número_de_roteadores_no_eixo_y>*. Em NoCs com topologia árvore gorda, por exemplo, esse formato pode ser substituído pela lista dos números dos roteadores. A Tabela 8 apresenta uma lista completa dos parâmetros aos quais a plataforma dá suporte.

Tabela 8 – Parâmetros de avaliação aos quais a plataforma Nocbench dá suporte.

Parâmetro	**Descrição**
pu_[n]	Utilização média do EP com id *n*
pu_[x]	Utilização média do EP com nome *x*
pu_avg	Utilização média de todos os processadores
tc_[n]	Número de vezes que a tarefa com id *n* foi disparada
tc_[x]	Número de vezes que a tarefa com nome *x* foi disparada
tc_tot	Soma do número de vezes que todas as tarefas foram disparadas
tt_[n]_[y]	Instante que a tarefa com id *n* foi disparada pela *y* vez

ec_[n]	Número de vezes que o evento *n* foi disparado
ec_tot	Soma do número de vezes que todos os eventos foram disparados
lat_[src]_[dst]_min	Latência mínima das mensagens enviadas da porta de saída *src* à porta de entrada *dst*
lat_[src]_[dst]_avg	Latência média das mensagens enviadas da porta de saída *src* à porta de entrada *dst*
lat_[src]_[dst]_max	Latência máxima das mensagens enviadas da porta de saída *src* à porta de entrada *dst*
latf_[src]_[dst]_min	Latência mínima das mensagens enviadas da porta de saída *src*àporta de entrada *dt* (considerando apenas mensagens totalmente enviadas)
latf_[src]_[dst]_avg	Latência média das mensagens enviadas da porta de saída *src*àporta de entrada *dst* (considerando apenas mensagens totalmente enviadas)
latf_[src]_[dst]_max	Latência máxima das mensagens enviadas da porta de saídas*rc* à porta de entrada *dst* (considerando apenas mensagens totalmente enviadas)
path_[src]_[dst]_min	Latência mínima de todas as mensagens que trafegaram pelo caminho entre a porta de saída *src* e a porta de entrada *dst*
path_[src]_[dst]_avg	Latência média de todas as mensagens que trafegaram pelo caminho entre a porta de saída *src* e a porta de entrada *dst*
path_[src]_[dst]_max	Latência máxima de todas as mensagens que trafegaram pelo caminho entre a porta de saída *src* e a porta de entrada *dst*
path_[src]_[dst]_count	Número de mensagens que completaram o caminho entre a porta de saída *src* e a porta de entrada *dst*

A *tag pe_lib* define o local onde se encontra armazenada a biblioteca externa(apresentada na Figura 11), através do atributo *file.* A biblioteca contém as características de cada recurso.

A *tagrng_seed* é utilizada para a geração de um número aleatório e afeta diretamente o intervalo entre o envio de dados quando a probabilidade for menor que 100%. Ela também especifica a quantidade de operações executadas e a quantidade de dados enviados quando um valor randômico for especificado.

As *tags sim_resolution, sim_length* e *measurements* definem respectivamente a unidade (*fs, ps*, *ns*, *us*, *ms* ou *s*) utilizada nos arquivos de *log*, o tempo de simulação e o intervalo em que as avaliações são realizadas. Este intervalo determina a frequência com que alguns arquivos de *log* são atualizados.

As *tags <log_nome_do_log>*especificam o nome dos *logs*, bem como o local em que eles serão armazenados. Por fim, ainda podem-se especificar funções custo. Estas apresentam resultados de diversos parâmetros de avaliação.

3.2 Transaction Generator 2 (TG2)

O núcleo do simulador TG2 é codificado em C++ e possui mais de 12 mil linhas de código, sendo que boa parte das classes que compõem a implementação atual correspondem a *tags* do estilo de modelagem apresentado na Seção 3.1, como se observa no diagrama de classes da Figura 13.

A primeira classe executada é a classe *Main*. Esta é responsável pela extração das informações contidas no modelo descrito no arquivo XML, cujo formato a Seção 3.1 discutiu. O modelo é selecionado pelo usuário. *Main* utiliza estas informações para criar objetos que formam o ambiente a ser simulado, tais como: eventos (*Event*), tarefas (*Task*), EPs (*ProcessingElement*), filas (*Buffer*), funções de custo (*CostFunction*), etc.

A classe *Main* também é responsável pela criação de dois objetos auxiliares de extrema importância na implementação atual do simulador: *Configuration* e *NocFactory*. O primeiro objeto é utilizado como um objeto de armazenamento, e é utilizado pelos demais para obter informações sobre o modelo, armazenadas em um objeto *PropertyTree*. O outro objeto (*NocFactory*) é responsável pela instanciação da NoC a ser utilizada na simulação.

O simulador permite a modelagem de sistemas com memória compartilhada e *caches*, incluindo a possibilidade de modelar *cache misses*. Para isso, são utilizadas as classes *MemoryModel* e *MemArea*. *MemoryModel* atua apenas como um *wrapper*, uma vez que a maior parte da modelagem de memória é realizada com base em desenvolvimento realizados pelo KTH, não apresentado no diagrama de classes da Figura 13.

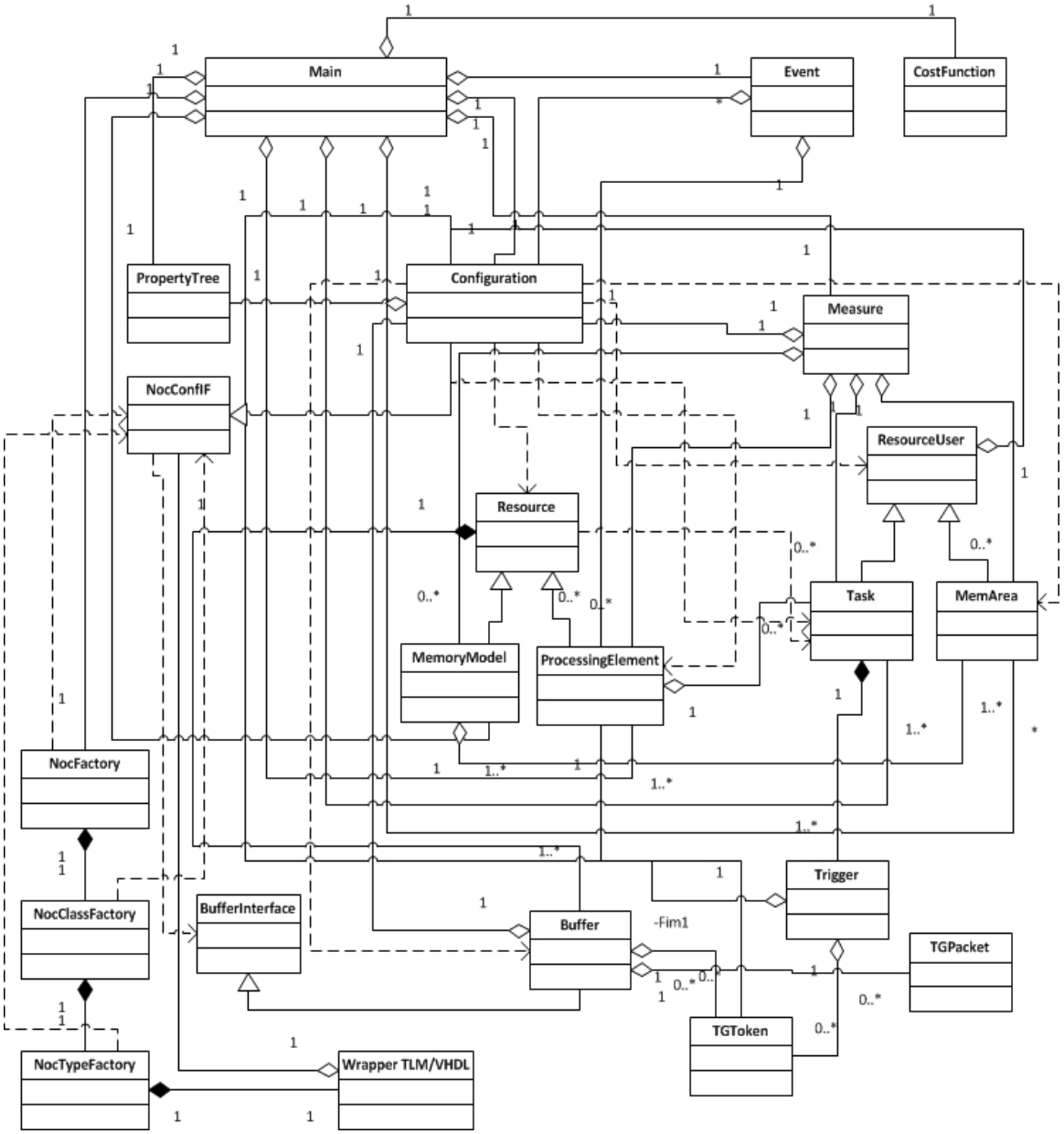

Figura 13 - Diagrama de classes do simulador TG2.

Apesar da possibilidade de uso do paradigma de comunicação por memória compartilhada, a distribuição atual do simulador conta apenas com modelos de aplicações que utilizam troca de mensagens para comunicação entre tarefas.

Tarefas podem ser mapeadas em um mesmo EP ou em EPs diferentes. No último caso, quando uma tarefa (classe *Task*) deseja enviar uma mensagem a outra, o EP (classe *Processing Element*) responsável pela execução da mesma divide esta mensagem (classe *TGToken*) em um ou vários pacotes (classe *TGPacket*) e os armazena em sua fila de saída (classe *Buffer*). O EP também armazena esta mensagem no objeto *Configuration*. Quando um pacote chega à fila de entrada de um

determinado EP, compara-se o primeiro dado do *payload* com o ID das mensagens armazenadas em *Configuration*. No TG2 este primeiro dado sempre corresponde ao ID da mensagem ao qual o pacote pertence. Caso a comparação seja válida, o pacote é recebido. Caso contrário, uma exceção é lançada e a simulação é abortada.

Um *Wrapper* TLM/VHDL (classe *Wrapper*) realiza o processo de obtenção dos pacotes armazenados na fila de saída dos EPs, criação de pacotes de rede e repasse destes a Interface de Rede da NoC. Este mesmo Wrapper também realiza a recepção dos dados ejetados pela NoC, o agrupamento destes em pacotes, e a inserção dos mesmos na fila de entrada dos EPs. A seleção da NoC e do respectivo *Wrapper* que serão utilizados em uma determinada simulação é papel de uma hierarquia de objetos *Factory*: *NocFactory*, *NocClassFactory* e *NocTypeFactory*, onde cada objeto desta lista é responsável pela instanciação do objeto seguinte.

A decisão de quais classes *Factory* devem ser instanciados dá-se em tempo de execução, através do acesso aos atributos *class*, *type* e *subtype* armazenados na classe base de *Configuration*, *NocConfIF*. O atributo*class* refere-se ao autor ou nome da NoC; *type*-corresponde ao nível de abstração ou a alguma característica que distingue NoCs de uma mesma classe; e *subtype* representa a organização dos roteadores na rede. Por exemplo, considere as configurações de NoCs apresentadas na Tabela 9, onde se têm duas classes de NoCs, cada uma com dois tipos. Assumindo que a primeira configuração tenha sido especificada, o objeto *NocFactory* realizará a instanciação do objeto *NocClassFactory* da NoC *Hermes*. *NocClassFactoryHermes*, por sua vez, criará um objeto *NocTypeFactory* do tipo "On-Off". E por fim, *NocTypeFactoryOn-Off* criará um objeto *Wrapper* para o tamanho 8x8 da rede escolhida.

Além do tamanho da NoC, a classe NocTypeFactory do tipo escolhido também é responsável pela passagem de outras informações necessárias para a criação do objeto Wrapper desejado. Estes parâmetros variam de acordo com as características da rede e da maneira como a mesma encontra-se descrita.

Tabela 9 - Exemplos de especificações de NoCs utilizando o estilo de modelagem aceito pelo TG2.

Configuração	Classe	Tipo	Subtipo

1	Hermes	On-Off	8x8
2		Handshake	
3	Ase Mesh	VHD	8x8
4		SystemC	

4. Contribuições Iniciais

A revisão do estado da arte que o Capítulo 2 apresenta justificou a escolha da plataforma Nocbench como base para realizar o presente trabalho. Contudo, deficiências nesta plataforma, que constitui um ambiente em constante evolução pelos seus proponentes levaram o autor a propor melhorias nesta. Um objetivo destas propostas foi aumentar o grau de automatização das tarefas propensas a erro da plataforma. Um segundo objetivo visou extrair funcionalidade do interior do código de programas da plataforma, de forma a tornar a mesma mais flexível e evitar a necessidade de programação da plataforma de avaliação de NoCs por usuários finais desta. A Seção 4.1 descreve uma melhoria relevante proposta pelo autor e aceita para compor uma nova versão da plataforma Nocbench, a geração automatizada de *wrappers* VHDL/SystemC TLM. Além desta melhoria, foi avaliada a adequabilidade das NoCs do grupo GAPH para integração à plataforma Nocbench. O resultado foi descobrir que o estado atual dos modelos de NoC do GAPH poderiam ser melhorados e ao mesmo tempo ter seu processo de integração ao Nocbench facilitado por estas mudanças. Assim, a Seção 4.2 descreve um conjunto de adaptações realizadas sobre os modelos das NoCs do GAPH.

4.1 Geração Automatizada de *Wrappers*

A classe Wrapper converte pacotes TLM gerados pelos EPs em pacotes RTL e vice-versa. A passagem dos parâmetros da classe *NocTypeFactory* à classe *Wrapper*ocorre em tempo de compilação, através de *templates* C++. Inicialmente, o TG2 permitia a utilização de apenas um conjunto de tamanhos de NoCs, especificados no código fonte da classe *NocTypeFactory*. Caso fosse de interesse a utilização de um tamanho não pertencente a este conjunto, o código fonte desta classe precisaria ser modificado, e todas as classes do sistema recompiladas. Uma primeira contribuição original deste trabalho foi remover esta limitação, automatizando o processo de criação do *Wrapper* TLM/VHDL para qualquer NoC.A Figura 14ilustra como seria o código fonte da classe *NocTypeFactory* da

NoC *Hermes OO* para três tamanhos de redes (2x2. 4x4. 8x8) sem a automatização do processo de criação. A Figura 15 apresenta os modelos desenvolvidos para a geração do código fonte desta classe para qualquer tamanho de rede.

```
1.
// Declaração do Wrapper TLM/VHDL para a NoC Hermes On-Off
// Arquivo noc_type_factory_hermes_on_off.hh

wrapper_hermes_on_off <  4, 2, 2, 32, 32, 16, 500000000 >* ni_2x2;
wrapper_hermes_on_off < 16, 4, 4, 32, 32, 16, 500000000 >* ni_4x4;
wrapper_hermes_on_off < 64, 8, 8, 32, 32, 16, 500000000 >* ni_8x8;
```

```
2.
// Instanciação do Wrapper TLM/VHDL apropriado para o tamanho de rede especificado
// Arquivo noc_type_factory_hermes_on_off.cc

std::string subtype = nocConfIf->getNocSubType();
if(subtype == "2x2")
{
        wrapper_2x2 = new wrapper_hermes_on_off<4, 2, 2, 32, 32, 16, 500000000>
        ("hermes_on_off", nocConfIf);
}
else if(subtype == "4x4")
{
        wrapper_4x4 = new wrapper _hermes_on_off<16, 4, 4, 32, 32, 16, 500000000>
        ("hermes_on_off", nocConfIf);
}
else if(subtype == "8x8")
{
        wrapper_8x8 = new wrapper_hermes_on_off<64, 8, 8, 32, 32, 16, 500000000>
        ("hermes_on_off", nocConfIf);
}
else
{
        std::ostringstream oss;
        oss << "NocFactory (VHDL) unsupported subtype \""
        << subtype << "\"";
        throw std::runtime_error(oss.str().c_str());
}
```

Figura 14 - Extrato do código fonte da classe NocTypeFactoryOnOff. 1 - arquivo de cabeçalho da classe (".hh") e 2 - arquivo de implementação da mesma (".cc").

Para permitir a utilização de qualquer tamanho de NoC e de concentrar todas as modificações do sistema a ser simulado no arquivo XML do modelo, como proposto inicialmente por Salminen, os arquivos da classe *NocTypeFactory* de todas as NoCs presentes no simulador passaram a ser criados com base em arquivos modelos. Para isso, foram inseridos dois arquivos adicionais para cada um dos tipos de redes integradas ao simulador, um para o arquivo de ca-

beçalho (extensão ".hh") e outro para o arquivo de implementação (extensão .cc). A estrutura e o conteúdo dos mesmos são semelhantes a dos arquivos originais, mas os valores literais dos últimos foram substituídos por símbolos nos primeiros. Com isso, as características do *Wrapper* passam a ser informadas através de atributos adicionais na *tagnoc* do modelo (discutido na Seção3.1.4). No momento da simulação, executa-se um *script* responsável pela obtenção dos valores destes atributos e pela criação dos arquivos de código fonte da classe *NoCTypeFactory* para o tipo de rede utilizada na simulação. Este *script* tem como base os arquivos modelos do tipo de rede em questão. Após executar o *script*, apenas os arquivos necessários são recompilados e a execução da simulação efetivamente tem início.

```
1.
// Declaração do Wrapper TLM/VHDL para a NoC Hermes On-Off
// Arquivo noc_type_factory_hermes_on_off_template.hh

wrapper_hermes_on_off < $numero_de_wrappers$,
                        $numero_de_linhas$,
                        $numero_de_colunas$,
                        $largura_do_barramento_de_dados$,
                        $largura_do_barramento_de_enderecos$,
                        $largura_do_barramento_de_tamanho_de_payload$,
                        $frequencia_de_operacao$,
                      >* wrapper_hermes_on_off;
```

```
2.
// Instanciação do Wrapper TLM/VHDL apropriado para o tamanho de rede especificado
// Arquivo noc_type_factory_hermes_on_off_template.cc

wrapper = wrapper_hermes_on_off < $numero_de_wrappers$,
                                  $numero_de_linhas$,
                                  $numero_de_colunas$,
                                  $largura_do_barramento_de_dados$,
                                  $largura_do_barramento_de_enderecos$,
                                  $largura_do_barramento_de_tamanho_de_payload$,
                                  $frequencia_de_operacao$,
                                >* wrapper;
```

Figura 15 – Extrato do código fonte dos arquivos modelos da classe NocTypeFactoryOnOff. 1 – modelo do arquivo de cabeçalho da classe (".hh*"). 2 – modelo do arquivo de implementação da classe ("*.cc*"). Estes modelos são parametrizados antes do início da simulação.*

Os parâmetros entre "<" ">" representam respectivamente o número de *wrappers*, de linhas e de colunas, a largura dos barramentos de dados e de endereços, o tamanho de *payload* e a frequência de operação. Este conjunto de parâmetros é utilizado para a instanciação de qualquer um dos *Wrappers* das

NoCs integradas ao longo deste trabalho. A largura dos barramentos de dados, de endereços e do tamanho de *payload* do *Wrapper* são iguais ao tamanho do *flit* da NoC.

O número e o tipo de símbolospresentes variam de acordo com o *Wrapper* e a descrição da NoC. No caso das descrições VHDL das NoCs *Hermes* integradas ao simulador, especificam-se informações tais como: largura dos barramentos, profundidade dos *buffers* de entrada, algoritmo de roteamento, tamanho da rede, entre outras. Estas aparecem no arquivo *package* de cada rede. Isto se distingue do que ocorre nas NoCs nativas do simulador, que recebem estas informações através das declarações *"generic"* diretamente de suas respectivas Interfaces de Rede.

4.2 Adaptação de NoCs

Uma segunda contribuição deste trabalho foi a adaptação e otimização das descrições das NoCs *Hermes HS*, *Hermes OO*, *Hermes TB*, *Hermes VC*, e *YeaH*, visando seu uso junto com o simulador TG2 e os conjuntos de *benchmarks* que este aceita (Nocbench e MCSL). Para tanto foram utilizados os modelos das descrições originais destas redes formados pelos conjuntos de arquivos que aparecem na Tabela 10, disponíveis no ambiente ATLAS de geração de NoCs [MEL05a].

Tabela 10 - Arquivos que compõem as descrições das NoCs integradas à plataforma Nocbench.

HERMES HANDSHAKE	**HERMES ON-OFF**
Hermes_package.vhd Hermes_buffer.vhd Hermes_switchcontrol.vhd Hermes_router.vhd	Hermes_package.vhd Hermes_buffer.vhd Hermes_crossbar.vhd Hermes_switchcontrol.vhd Hermes_router.vhd
HERMES TB	**HERMES VC**
HermesTB_package.vhd HermesTB_buffer.vhd HermesTB_crossbar.vhd HermesTB_switchcontrol.vhd RouterCC.vhd	Hermes_package.vhd Hermes_buffer.vhd Hermes_inport.vhd Hermes_outport.vhd Hermes_switchcontrol.vhd Hermes_router.vhd
YEAH	

hermes_pld_package.vhd input_buffer.vhd arbiter.vhd routing_unit.vhd input_control.vhd input_interface.vhd	output_control.vhd output_interface.vhd hermes_pld_port.vhd hermes_pld.vhd noc_hermes_pld.vhd

Na ATLAS os modelos formados pelos arquivos apresentados na Tabela 10 servem de base para a geração de NoCs via parâmetros especificados pelo usuário. Uma exceção é a NoC *YeaH*, ainda não integrada ao ambiente. Além dos arquivos de modelo parametrizados, o projeto de uma NoC *Hermes HS*, *Hermes OO* ou *Hermes VC* pode conter até dez arquivos adicionais, sendo um deles a interface de entrada e saída da rede, e os demais, os nove tipos de roteadores existentes, considerando uma organização com no mínimo três linhas e três colunas de roteadores. No caso da NoC *Hermes TB*, apenas dois arquivos adicionais são gerados, uma vez que todos os roteadores da rede são do mesmo tipo.

Um requisito importante considerado durante a realização deste trabalho foi a manutenção do padrão utilizado para a integração das demais redes já presentes na plataforma. Assim, ao invés de utilizar as descrições geradas pela ATLAS, optou-se pela adaptação e otimização das mesmas.

O processo de otimização consistiu basicamente na reescrita das descrições dos arquivos de projeto de uma NoC 3X3 gerada pela ATLAS para as redes *Hermes HS*, *Hermes OO*, *Hermes TB* e *Hermes VC*, utilizando sempre que possível, construções "*for-generate*" da linguagem VHDL. A partir destas descrições derivaram-se descrições parametrizáveis para qualquer tamanho de rede. Com isso a quantidade de linhas de código presentes na interface de entrada e saída destas NoCs, que era de aproximadamente 450 linhas de código passou a ser de apenas 150 linhas. Além disso, as nove descrições de roteadores existentes que totalizavam cerca de 1100 linhas de código, foram substituídas por apenas uma com aproximadamente 250 linhas. Isto para uma rede 3X3. Em NoCs com um número maior de roteadores, está diferença é ainda mais acentuada. Em uma NoC 8X8, por exemplo, a economia de código com a interface de entrada e saída foi de 2800 linhas. Já, para as descrições dos roteadores, esse número é o mesmo para qualquer número de roteadores.

Com o intuito de manter o padrão utilizado nas outras NoCs já integradas à plataforma Nocbench, as seguintes modificações foram realizadas nas redes Hermes:

- O endereçamento [coluna][linha] (Figura 16a) utilizado nas redes desenvolvidas pelo GAPH, foi substituído pelo endereçamento [linha][coluna] (Figura 16b);

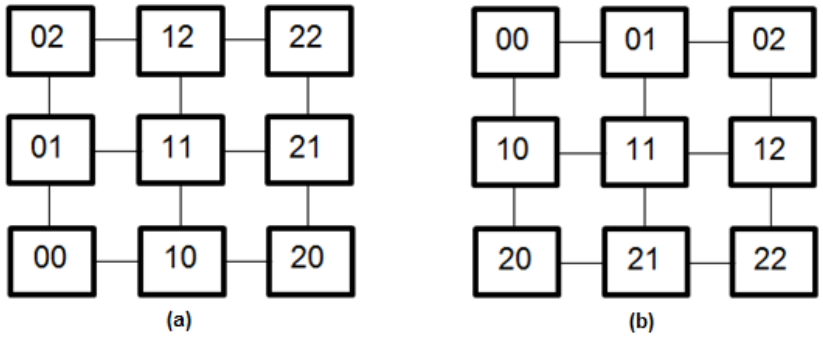

Figura 16 – Endereçamentos: (a) [coluna][linha] de baixo pra cima – endereçamento padrão das NoCs presentes no ambiente ATLAS, (b) [linha][coluna] de cima para baixo – endereçamento padrão das NoCs presentes na plataforma Nocbench.

- O algoritmo XY foi substituído pelo algoritmo YX (exceto na NoC *Hermes TB* cujo algoritmo de roteamento é próprio para a Topologia Torus). Isto exigiu a modificação dos *crossbars* das redes *Hermes VC* e *YeaH*, que até então, não continham as ligações das portas de entrada Norte e Sul com as portas de Saída Leste e Oeste.

Além destas modificações, o modo como os dados de entrada e saída passaram a ser injetados na rede e ejetados da rede foi modificado, o que exigiu a modificação dos tipos de dados utilizados para representar os barramentos envolvidos. A Figura 17 apresenta a definição dos tipos e a declaração dos barramentos de entrada e saída de dados que utilizam os mesmos, nas descrições das NoCs geradas pela ATLAS. A Figura 18 mostra como isso é feito nas descrições integradas ao Nocbench.

```
1. -- Definição dos tipos no arquivo package

2. subtype regflit is std_logic_vector ( (TAM_FLIT-1) downto 0);
3. type arrayNrot_regflit is array ( (NROT-1) downto 0) of regflit;

4. -- Sinais de entrada e saída de dados declarados na interface de entrada e saída
   das redes geradas pelo ambiente ATLAS

5. data_inLocal   : in arrayNrot_regflit;
6. data_outLocal  : out arrayNrot_regflit;
```

Figura 17 – Definição dos tipos e declaração dos barramentos de entrada e saída de dados que os utilizam nas NoCs geradas pela ATLAS.

```
1. -- Sinais de entrada e saída de dados declarados na interface de entrada e saída
   das redes integradas ao Nocbench

2. tx_data_in   :  in  std_logic_vector  ( TAM_FLIT  *  NROT-1  downto  0);
3. rx_data_out  :  out std_logic_vector  ( TAM_FLIT  *  NROT-1  downto  0);
```

Figura 18 – Declaração dos barramentos de entrada e saída de dados nas NoCs integradas ao Nocbench

Note que no Nocbench a passagem dos dados à NoC, assim como o recebimento dos dados desta é realizada com o auxílio de um único vetor com largura igual à largura do barramento de dados da NoC (TAM_FLIT) multiplicada pelo número de roteadores. Na *ATLAS* esse processo é realizado através de um vetor de vetores com largura igual do tamanho do *flit* da NoC. Percebe-se ainda que a nomenclatura adotada para os sinais de entrada e saída da NoC também foi modificada.

Todas as otimizações e adaptações mencionadas anteriormente, com exceção da mudança de endereçamento dos roteadores e substituição do algoritmo de roteamento, não se aplicam à NoC *YeaH*. Apesar da mesma, não possuir uma interface de entrada e saída compatível com o padrão utilizado no *Nocbench*, o *wrapper* que acompanha a distribuição pode ser utilizado para solucionar este problema.

Visando utilizar a mesma interface de rede desenvolvida para as NoCs *Hermes HS*, *Hermes OO*, *Hermes TB* e *Hermes VC*, os parâmetros que antes eram passados para a NoC *YeaH* através da declaração *"generics"* passaram a ser especificados no *package* da descrição.

O processo de geração da NoC a ser utilizada na simulação é semelhante ao realizado para a geração dos *Wrappers*. O *package* de cada rede é gerado com

base em um modelo parametrizado do arquivo original no momento da execução do simulador. Os parâmetros que caracterizam as NoCs integradas neste trabalho são os seguintes: número de roteadores, número de linhas, número de colunas, largura do barramento de dados, profundidade do *buffer* e algoritmo de roteamento. Os quatro primeiros são utilizados tanto para a criação da NoC quanto para a criação do *Wrapper*.

5. Integração das NoCs do GAPH à plataforma Nocbench

Este Capítulo apresenta a terceira contribuição da Dissertação: a integração das NoCs *Hermes HS*, *Hermes OO*, *Hermes TB*, *Hermes VC*, e *YeaH* à Plataforma *Nocbench*. O projeto de integração destas redes foi baseado no modelo de integração das NoCs já presentes na plataforma. Neste modelo, a classe Wrapper, responsável pela conversão de pacotes TLM gerados pelos EPs em pacotes RTL e vice-versa não se comunica diretamente com a NoC, e sim com a NI da mesma. Para isto, ela conta com dois processos: um de envio e outro de recebimento de dados. Estes processos controlam o fluxo de dados das filas (classe *Buffer*) dos EPs. Em um sistema 4x4 existem 32 destes processos, um de envio e outro de recebimento para cada EP.

A NI controla o fluxo dos dados que passa pela NoC e assim como o *Wrapper*, possui um processo de envio denominado *Sender* e outro de recebimento de dados denominado *Receiver* para cada EP do sistema. A Figura 19 ilustra o fluxo de dados entre EPs e NoC.

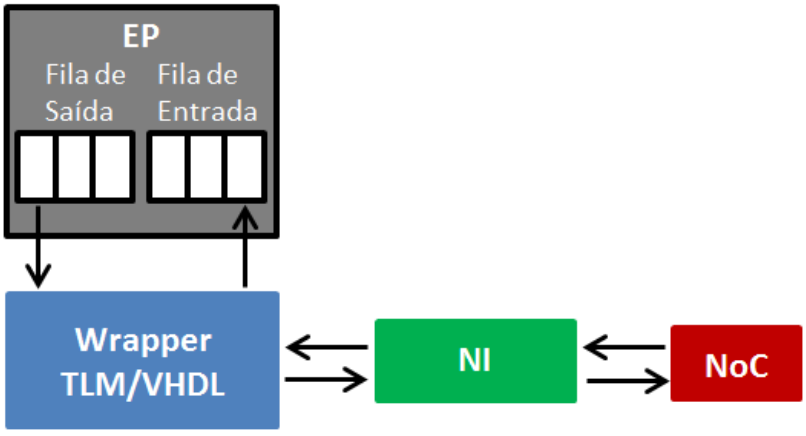

Figura 19 – Esquema mostrando o fluxo de dados entre as filas dos EPs e a NoC.

Toda a lógica de comunicação entre EPs e NoC é implementada nos módulos *Wrapper* e NI através de dois processos: um de envio (processo *Sender*) e outro de recebimento (processo *Receiver*) de dados. Estes processos comunicam-se

entre si de modo a respectivamente transferir os dados das filas de saída dos EPs à NoC e desta às filas de entrada dos EPs.

Uma vez que os processos *Sender* e *Receiver* do módulo *Wrapper* comunicam-se exclusivamente com os processos homônimos do módulo NI, sempre que houver uma referência a um destes módulos no texto que segue, esta estará referindo-se ao processo do módulo referenciado.

A seguir serão apresentados os pares de processos *Sender* e *Receiver* desenvolvidos para a comunicação dos EPs com a NoC. De modo a facilitar a compreensão destes, o protocolo de comunicação implementado em cada um destes é apresentado sob a forma de um diagrama de transição de estados. O Capítulo seguinte apresenta a validação destes protocolos.

O processo *Sender* de ambos os módulos é o mesmo para todas as redes, exceto para a rede *Hermes HS, que diferentemente das demais utiliza o esquema de controle de fluxo handshake ao invés do esquema on-off*. Os processos *Receiver* de ambos os módulos são semelhantes para todas as redes, exceto para a rede *YeaH*, que ao contrário das demais não exige um intervalo de no mínimo um ciclo entre o recebimento de dois pacotes

5.1 Processo *Sender*

O processo *Sender* do módulo *Wrapper* utiliza os métodos `txPacketAvailable()` e `txGetPacket()` da API de comunicação da classe *Buffer* para respectivamente verificar se existem pacotes na fila de saída do EP e obter os mesmos. Em seguida, as informações necessárias destes pacotes, são extraídas e repassadas ao processo homônimo do módulo NI. Note que estas informações variam de acordo com a rede. Por exemplo, as NoCs integradas neste trabalho utilizam a informação de tamanho de *payload* para determinar se o dado recebido pertence ao pacote atual ou é o primeiro dado de um novo pacote. Uma das NoCs nativas da plataforma, por outro lado, não utiliza esta informação e sim dois *bits* adicionais em cada pacote de rede, um antes do primeiro dado de *payload* e outro após o último dado de *payload.*

O processo *Sender* do módulo NI implementa o controle do fluxo de entrada de dados na NoC. Os dados recebidos do *Wrapper* são armazenados em registradores locais e caso exista espaço na fila do roteador destino, são repassados ao mesmo. Além disso, neste módulo é realizada a conversão dos dados recebidos do *Wrapper* para o formato utilizado pela NoC.

5.1.1 NoC *HermesHS*

O protocolo implementado no processo *Sender* do módulo *Wrapper* para a NoC *Hermes HS* pode ser representado através de uma máquina de três estados: *Aguarda Pacote*, *Envia Cabeçalho* e *Envia Payload* como a Figura 20 ilustra.

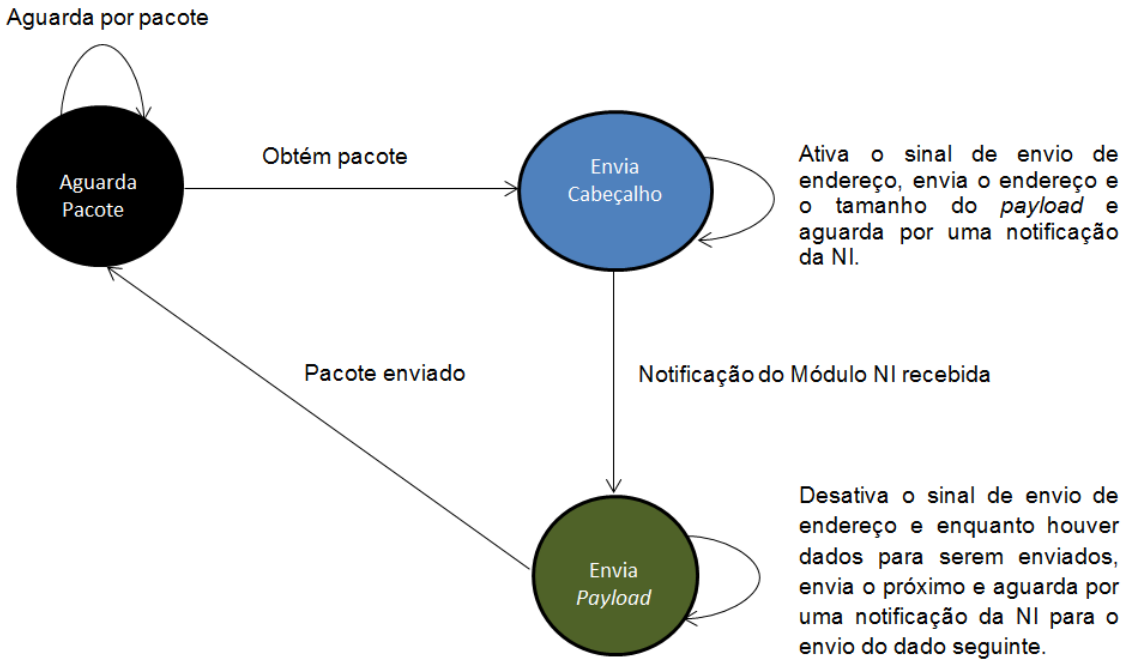

Figura 20 – Diagrama de transição de estados do processo Sender do módulo Wrapper para a NoC Hermes HS.

No estado inicial (*Aguarda Pacote*) realiza-se o monitoramento da fila de saída do EP em busca de pacotes a serem transmitidos. Assim que houver um pacote disponível, o mesmo é obtido e a máquina avança para o próximo estado.

No estado *Envia Cabeçalho*, o sinal de envio de endereço é ativado, e o endereço e o tamanho do *payload* do pacote são enviados à NI. A máquina permanece neste estado até o recebimento de uma notificação da NI, quando então avança ao estado *EnviaPayload*.

No estado *Envia Payload,* o sinal de envio de endereço é desativado e enquanto houver dados, enviam-se estes. Sempre após o envio de cada dado de *payload*, aguarda-se por uma notificação da NI, para que o próximo dado seja enviado. Finalmente, quando não houver mais dados, a máquina volta ao estado inicial e aguarda um novo pacote. Observe que no estado *Envia Payload,* antes do recebimento da primeira notificação da NI, o primeiro dado de *payload* já foi enviado. Assim, após a primeira notificação ter sido recebida, envia-se o segundo dado do *payload*, após a segunda notificação ter sido recebida o terceiro dado do *payload* será enviado, e assim sucessivamente. Em seguida, a Figura 21 mostra o protocolo de envio de dados do processo *Sender* do módulo NI

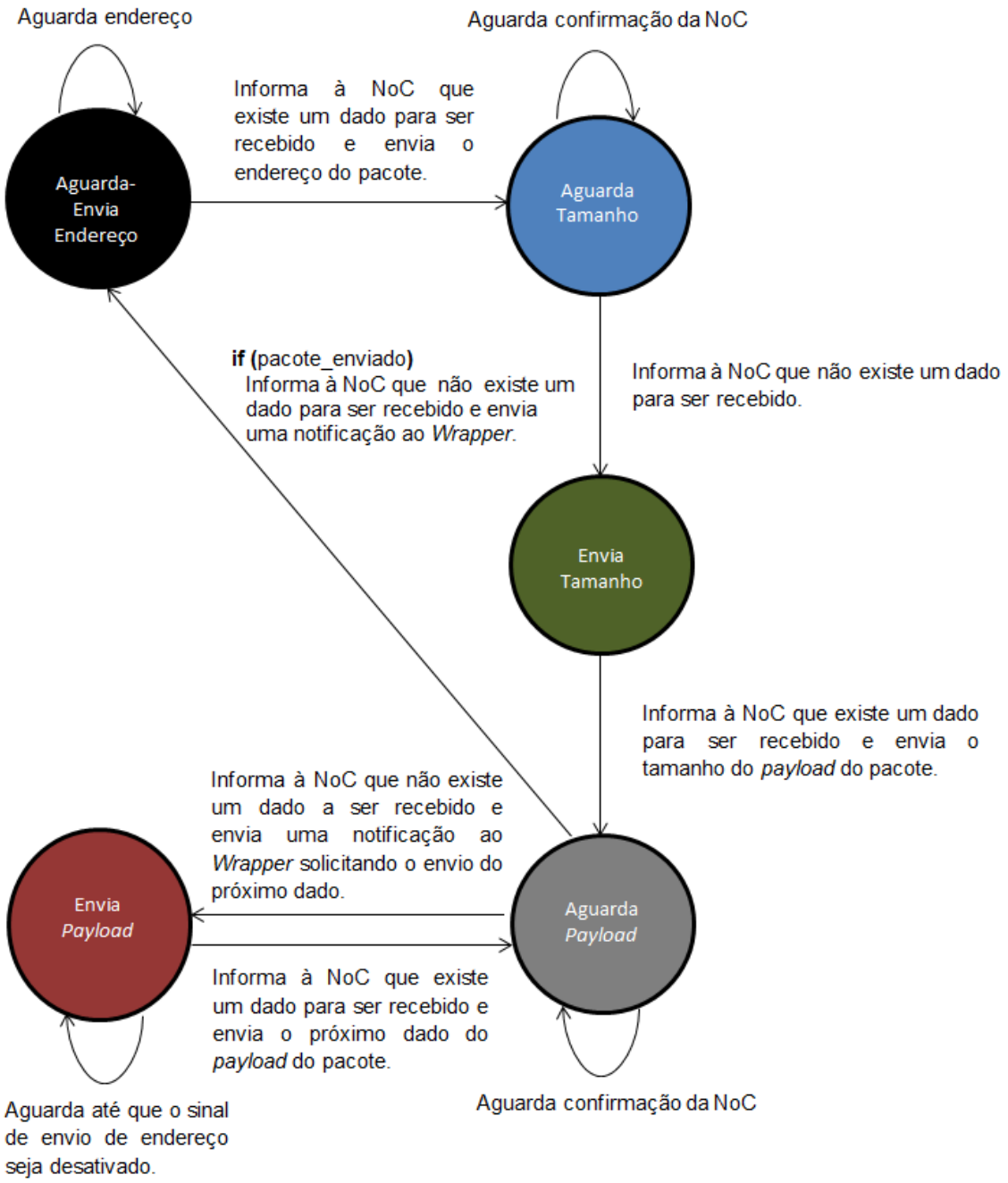

Figura 21 – Diagrama de transição de estados do processo Sender do módulo NI para a NoC Hermes HS.

A Figura representa o diagrama de transição de estados de uma máquina de cinco estados: *Aguarda-Envia Endereço, Aguarda Tamanho, Envia Tamanho, Aguarda Payload* e *Envia Payload.*

No estado inicial (*Aguarda-Envia Endereço*) a máquina aguarda pela ativação do sinal de envio de endereço. Assim que este sinal é ativado, o endereço recebido (ID do EP destino) é convertido no endereço do Roteador conectado a

este EP (coordenadas YX). Em seguida, este dado, bem como uma notificação informando que existe um dado válido para ser recebido, são enviados a NoC. Após isso, a máquina avança para o estado *Aguarda Tamanho*.

No estado *Aguarda Tamanho* a máquina aguarda pela confirmação do recebimento do endereço do pacote, por parte da NoC. Assim que esta confirmação é recebida, uma notificação informando que não existe nenhum dado válido para ser recebido é enviada à mesma e a máquina avança para o estado *Envia Tamanho*.

No estado *Envia Tamanho*, uma notificação informando que existe um dado válido para ser recebido, bem como o tamanho do *payload* do pacote, são enviados a NoC. Após isso, a máquina avança ao estado *Aguarda Payload*.

No estado *Aguarda Payload,* caso não haja mais dados para serem enviados, a máquina volta ao estado inicial. Caso contrário, aguarda-se pela confirmação do recebimento do último dado enviado por parte da NoC. Assim que esta confirmação é recebida, duas notificações são enviadas, uma para NoC, informando que não existe nenhum dado válido para ser recebido, e uma para o *Wrapper*, solicitando o envio do próximo dado. Em seguida, a máquina avança ao estado *Envia Payload*.

No estado *Envia Payload*, ocorre o envio dos dados de *payload* do pacote. O envio só tem início após a desativação do sinal de envio de endereço pelo *Wrapper*. Após o envio de cada dado, a máquina volta ao estado *Aguarda Payload*.

Observe-se que quando a máquina chega ao estado Envia Payload, o próximo dado já está sendo disponibilizado pelo Wrapper. Assim, quando a máquina volta ao estado Aguarda Payload, o dado que será enviado à NoC no próximo estado (Envia Payload), já se encontra disponível.

5.1.2 NoC *Hermes OO*, *Hermes TB*, *Hermes VC* e *YeaH*

Enquanto que na rede *Hermes HS* a troca de um *flit* entre um EP e seu roteador, ou entre dois roteadores vizinhos requer dois ciclos de relógio, nas demais redes integradas, apenas um ciclo é necessário. Assim, nestas últimas,

caso exista espaço no *buffer* do roteador destino, um *flit* é enviado a cada ciclo. Isto acrescenta certo nível de complexidade ao processo *Sender*, uma vez que o módulo NIconsome um ciclo de relógio adicional, dificultando o envio de um dado a cada ciclo. A solução adotada para o cumprimento deste comportamento utiliza um segundo barramento de dados. Desse modo, cada processo *Sender* do módulo *Wrapper* pode repassar dois dados de *payload* a cada ciclo ao processo *Sender* do módulo NI.

Conforme mostra a Figura 22, o protocolo implementado no processo *Sender* do módulo *Wrapper* para as NoCs *Hermes OO*, *Hermes TB*, *Hermes VC* e *YeaH* é semelhante ao utilizada para a NoC *Hermes HS* (ver Figura 20). As diferenças residem no estado *Envia Cabeçalho*, onde além de serem enviadas informações relativas ao cabeçalho do pacote, também são enviados os dois primeiros dados do *payload*, eno estado *Envia Payload*, onde ao invés de um, dois dados são enviados por ciclo.

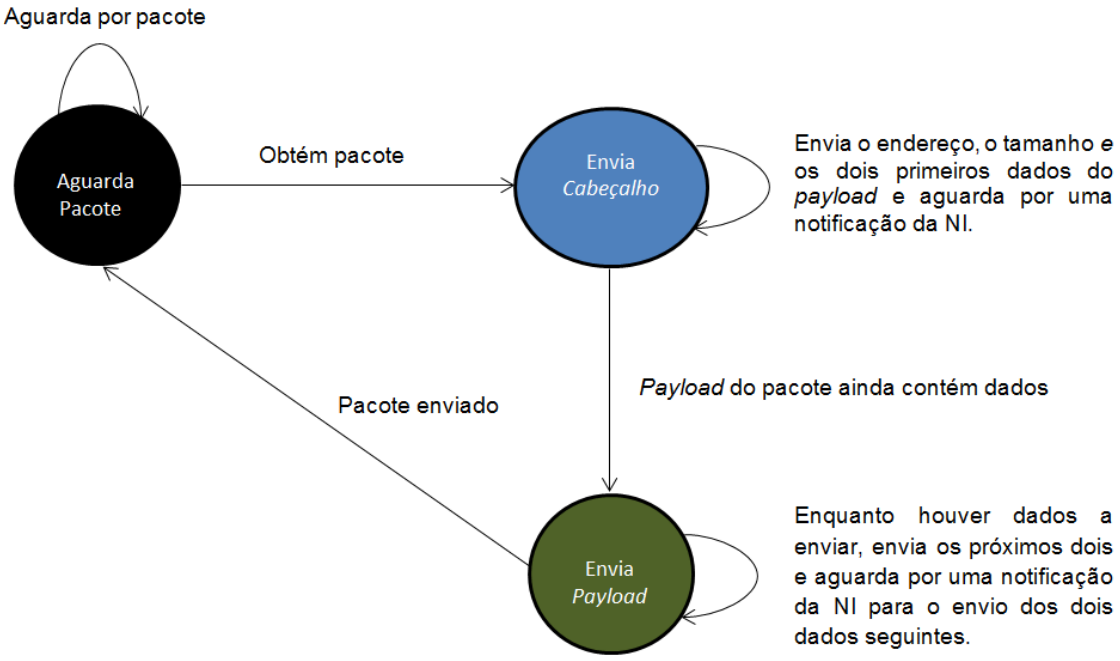

Figura 22 – Diagrama de transição de estados do processo Sender do módulo Wrapper para as NoCs Hermes OO, Hermes TB, Hermes VC e YeaH.

Como já mencionado, o valor do primeiro dado do *payload* de um pacote será sempre o identificador da mensagem ao qual ele pertence. Os valores dos demais dados não são considerados relevantes, e por isso, podem ser configurados com qualquer valor que caiba no espaço de representação do tamanho de *flit* da NoC. Caso a quantidade de dados do *payload* de um pacote for ímpar, o

processo *Sender* do módulo *Wrapper* para as NoCs *Hermes 00*, *Hermes TB*, *Hermes VC* e *YeaH repassará* uma quantidade de dados par à NI, e esta através da consulta ao tamanho do *payload* do pacote, descartará o último dado recebido. Para a comparação das NoCs, utilizou-se o identificador do EP que originou o pacote como valor dos dados de *payload*.

A máquina de estados utilizada para representar o protocolo de envio de dados do processo *Sender* do módulo NI é composta por quatro estados: *Aguarda-Envia Endereço*, *Envia Tamanho*, *Envia Payload* e *Envia Payload_2*, como ilustra a Figura 23.

No estado Inicial (*Aguarda-Envia Endereço*) aguarda-se pelo recebimento do endereço do pacote e por espaço disponível na fila do roteador destino. Assim que ambas as condições são cumpridas simultaneamente, uma notificação informando que existe um dado para ser recebido é enviada à NoC e o dado é repassado a ela. Após isso, a máquina avança ao estado *Envia Tamanho.* Neste estado, aguarda-se até que exista espaço disponível na fila do roteador destino. Após isto, o tamanho do *payload* do pacote é enviado à NoC e a máquina avança ao estado *Envia Payload*.

No estado *Envia Payload*, os dados presentes nos dois barramentos de dados são armazenados em barramentos auxiliares (aux e aux2). Caso exista espaço na fila de entrada do roteador destino, o dado presente no primeiro barramento de dados é enviado à NoC. Caso contrário, a máquina aguarda até que exista espaço e envia o dado presente no barramento auxiliar aux. Em seguida, uma notificação solicitando o envio do próximo par de dados é enviada ao módulo *Wrapper* e a máquina avança para o estado *Envia Payload_2*.

Antes de a máquina avançar ao estado *Envia Payload_2*, ela notifica o módulo *Wrapper*. Quando a máquina chega ao estado *Envia Payload_2*, esta notificação é recebida por este módulo, que envia o próximo par de dados. Estes dados apenas passam a estar disponíveis nos barramentos de dados do módulo *Wrapper* no próximo ciclo. Por isso, caso exista espaço no roteador destino na primeira tentativa de envio, o dado presente no barramento de dados é enviado. Caso contrário, o dado presente no barramento auxiliar é enviado (uma vez que o dado presente no barramento principal foi sobrescrito).

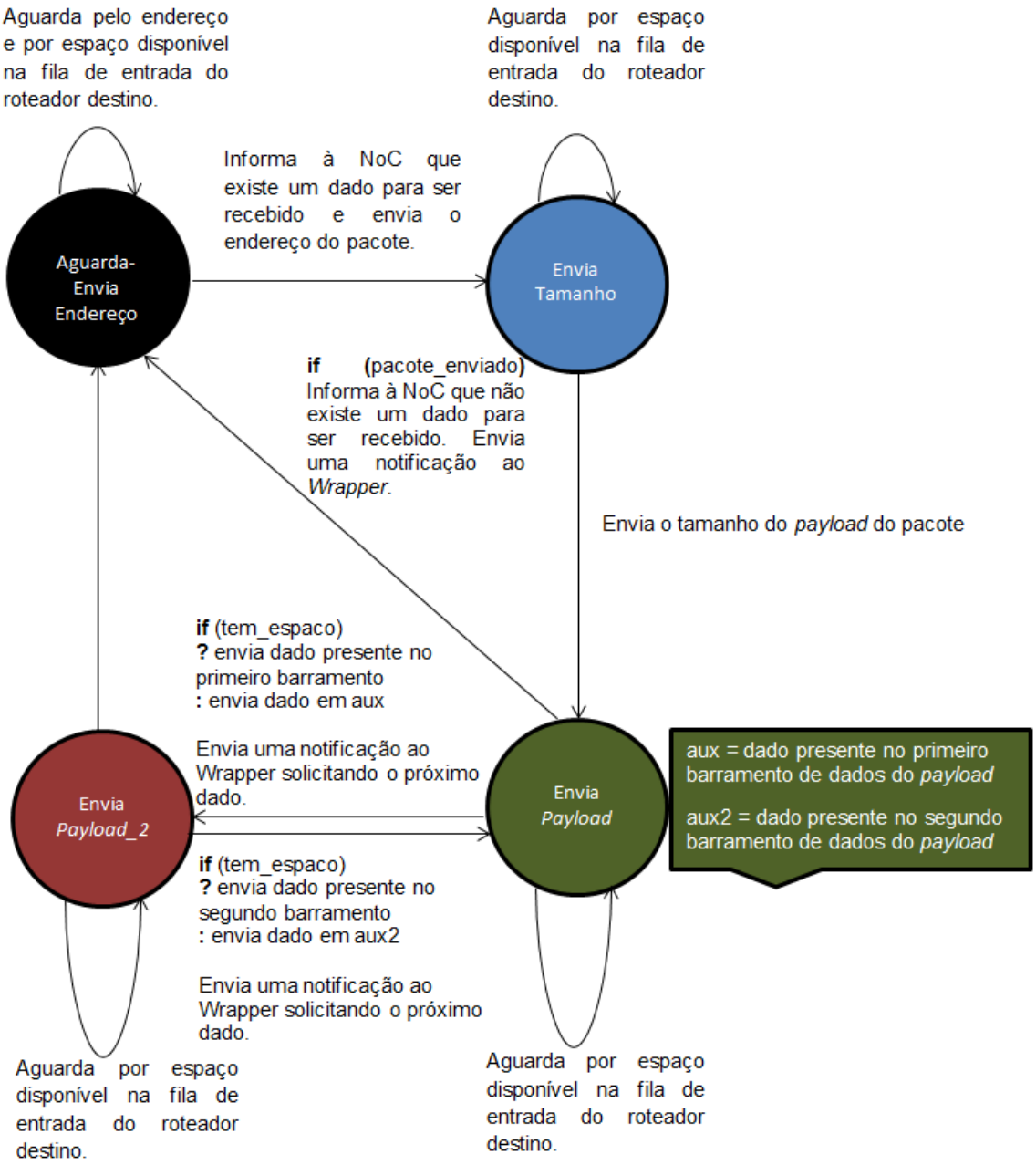

Figura 23 – Diagrama de transição de estados do processo Sender do módulo NI para as NoCs Hermes OO, Hermes TB, Hermes VC e YeaH.

No *estado Envia Payload_2*, de maneira similar ao que ocorre no estado anterior, caso exista espaço na fila de entrada do roteador destino o dado presente no segundo barramento de dados é enviado à NoC. Caso contrário, a máquina aguarda até que exista espaço, e envia o dado presente no barramento auxiliar aux2. Em seguida, uma notificação solicitando o envio do próximo par de dados é enviada ao módulo *Wrapper* e a máquina volta para o estado *Envia Payload*.

Nos dois últimos estados, caso o pacote tenha sido enviado, a NoC é informada de que não existe nenhum dado válido para ser recebido, envia-se uma notificação ao processo *Sender* do módulo *Wrapper* e a máquina volta ao estado inicial.

5.2 Processo *Receiver*

O processo *Receiver* do módulo NI implementa o controle do fluxo de saída de dados da NoC. Os dados recebidos desta são repassados ao processo homônimo do módulo *Wrapper,* onde os mesmos são agrupados em pacotes e armazenados nas filas de entrada dos EPs através do método `rxPutPacket()`.

Para a explicação do processo de envio de dados à NoC, seguiu-se a ordem do fluxo de dados, isto é, EP - Wrapper - NI - NoC. A ordem do processo de recebimento de dados é exatamente a inversa. Assim, apresenta-se primeiro o processo *Receiver* do módulo NI.

5.2.1 NoC *HermesHS*

O protocolo de recebimento de dados do processo *Receiver* do módulo NI para a NoC *Hermes HS* pode ser representado através de uma máquina de cinco estados: *Aguarda-Recebe Endereço, Aguarda Tamanho, Recebe Tamanho, Aguarda Payload* e *Recebe Payload.* A Figura 24 mostra o comportamento deste processo através de um diagrama de transição de estados.

No estado inicial (*Aguarda-Recebe Endereço)*, aguarda-se pelo recebimento do endereço do pacote. Após este dado ser recebido, o mesmo é convertido no ID do EP destino. Em seguida, envia-se este dado e uma notificação informando que o endereço do pacote está disponível ao *Wrapper*. Outra notificação informando que o endereço foi recebido é enviada à NoC e a máquina avança para o estado *Aguarda Tamanho*. Neste estado, o sinal utilizado para informar a NoC que o dado foi recebido é desativado e a máquina avança ao próximo estado.

No estado *Recebe Tamanho*, aguarda-se pelo recebimento do tamanho de *payload* do pacote. Após este dado ser recebido, uma notificação informando isto é enviada à NoC, e o mesmo é repassado ao *Wrapper.* Em seguida, a máquina a-

vança para o estado *Aguarda Payload*. Em *Aguarda Payload*, o sinal utilizado para informar a NoC que o dado foi recebido é desativado e a máquina avança ao próximo estado.

No estado *Recebe Payload*, aguarda-se pelo recebimento do próximo dado de *payload*, após o que se envia este ao *Wrapper*. Ao mesmo tempo, duas notificações são enviadas, uma para a NoC, informando que o dado foi recebido, e uma para o *Wrapper*, informando que o dado pode ser recebido. Feito isso, a máquina volta ao estado *Aguarda Payload*.

Durante o recebimento dos dados de *payload*, a máquina alterna entre os estados *Aguarda Payload* e *Envia Payload*. Após o pacote ter sido recebido, o sinal utilizado para informar a NoC que o dado foi recebido é desativado e uma notificação que não existem dados para serem recebidos é enviada ao *Wrapper*.

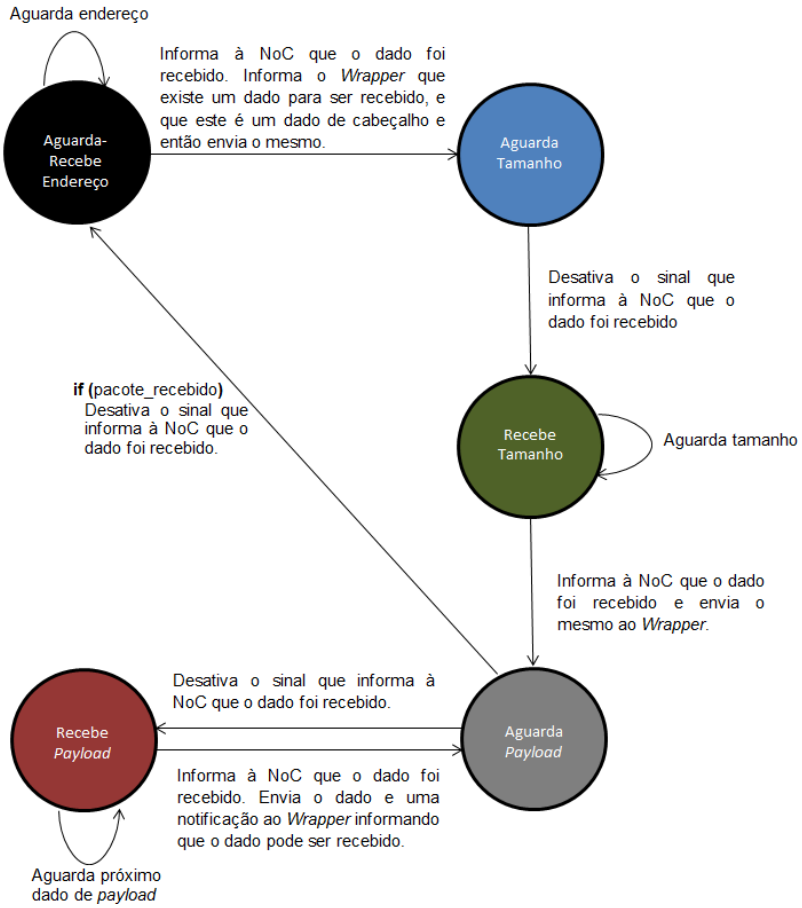

Figura 24 – Diagrama de transição de estados do processo Receiver do módulo NI para a NoC Hermes HS.

O protocolo implementado no processo *Receiver* do módulo *Wrapper* pode ser representado através de uma máquina de três estados: *Aguarda-Recebe Endereço, Recebe Tamanho* e *Recebe Payload* como ilustra a Figura 25.

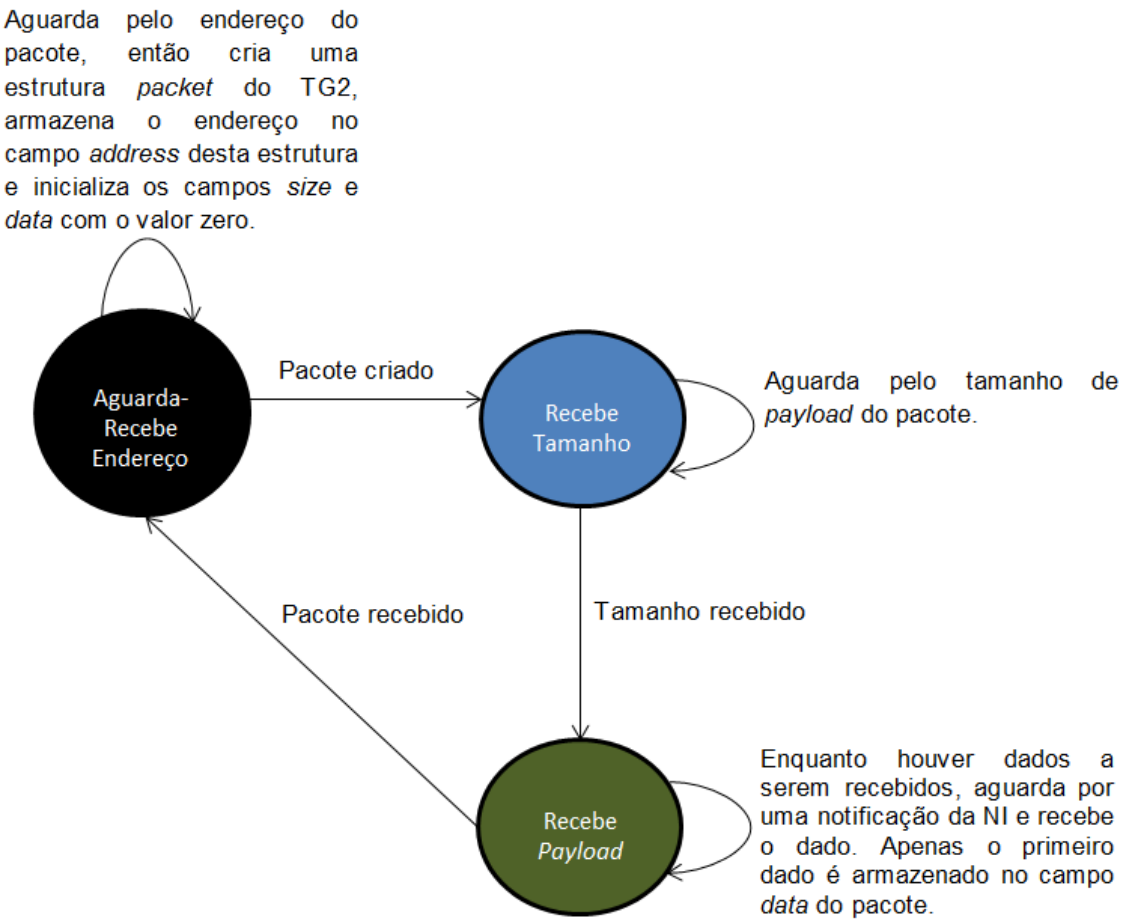

Figura 25 – Diagrama de transição de estados do processo Receiver do módulo Wrapper para a NoC Hermes HS.

A máquina permanece no estado inicial (*Aguarda-Recebe Endereço*) até o recebimento do endereço do pacote. Após isso, ela cria uma estrutura *packet* do TG2, armazena o endereço no campo *address* desta estrutura, inicializa os campos *size* e *data* com o valor zero e avança para o estado *Recebe Tamanho*.

Em *RecebeTamanho,* a máquina aguarda até o recebimento do tamanho de *payload* do pacote, então avança para o estado *Recebe Payload.*

Neste estado, enquanto houver dados a receber, a máquina aguarda por uma notificação da NIe então recebe o dado. Após o recebimento de cada dado, o campo *size* do pacote é incrementado em (largura_do_barramento_de_dados/8)*bytes*, uma vez que os tamanhos dos pacotes

gerados pelos EPs no TG2 são representados em *bytes* e não em *flits* como ocorre na NoC. Quando o pacote é totalmente recebido, a máquina volta ao estado inicial.

5.2.2 NoC *Hermes OO*, *Hermes TB*, *Hermes VC*

O protocolo de recebimento de dados do processo *Receiver* do módulo NIpara as NoCs *Hermes OO, Hermes TB, Hermes VC* pode ser representado por uma máquina de apenas três estados: *Aguarda-Recebe Endereço, Recebe Tamanho* e *Recebe Payload*. A simplicidade do protocolo implementado, deve-se ao fato das filas dos EPs não terem um tamanho limite. A Figura 26 ilustra o diagrama de transição de estados deste processo.

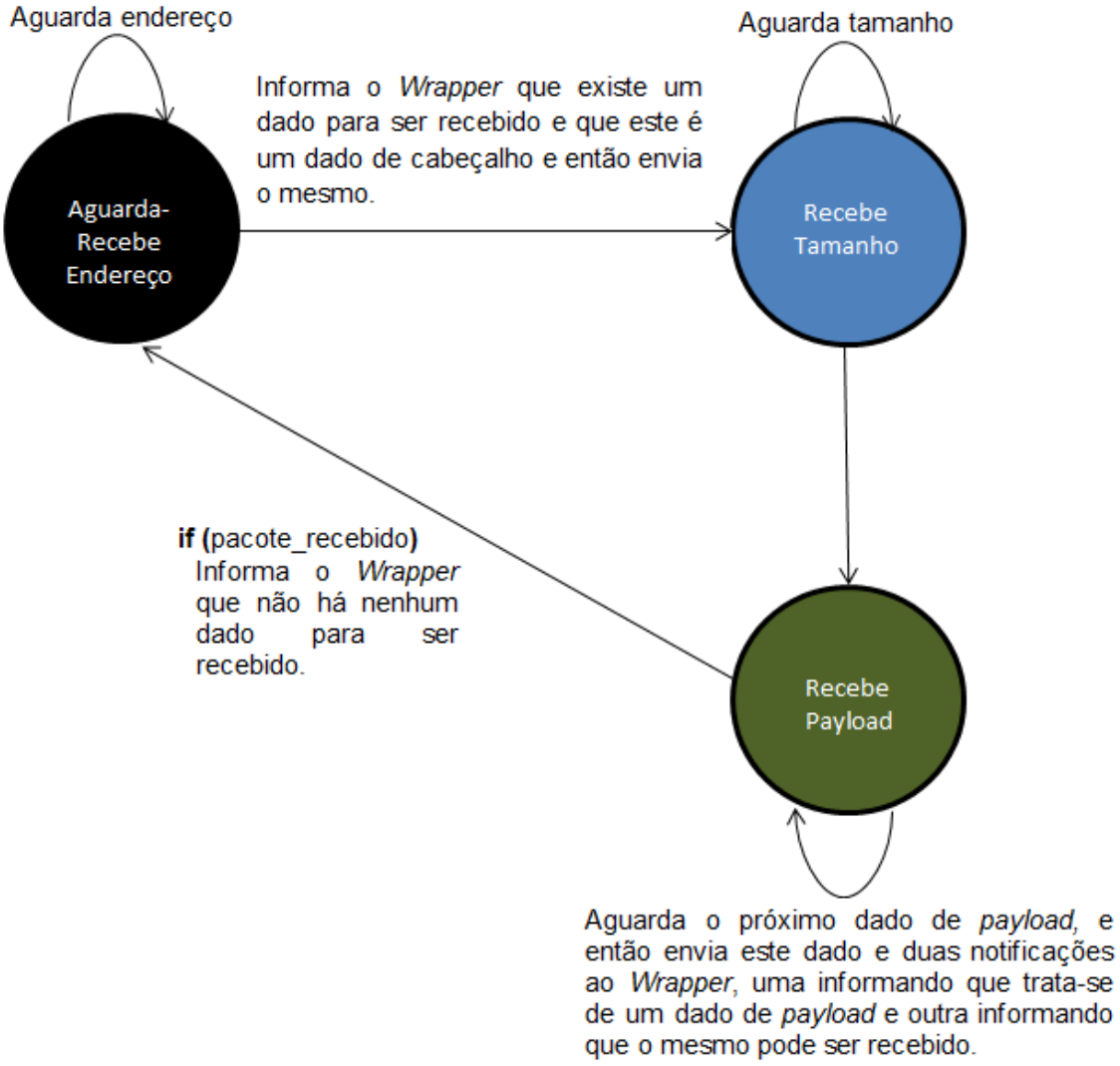

Figura 26 – Diagrama de transição de estados do processo Receiver do módulo NI para as NoCs Hermes OO, Hermes TB, Hermes VC.

No estado inicial (*Aguarda-Recebe Endereço)*, aguarda-se pelo recebimento do endereço do pacote. Após isto, o endereço é convertido no ID do EP destino. Em seguida, este dado e uma notificação informando que o endereço do pacote está disponível, são enviados ao *Wrapper*. Envia-se à NoC outra notificação informando que o endereço foi recebido e a máquina avança para o estado *Recebe Tamanho*.

No estado *Recebe Tamanho*, aguarda-se pelo recebimento do tamanho de *payload* do pacote. Após este dado ser recebido, a máquina avança para o estado *Recebe Payload*. Neste, aguarda-se pelo recebimento dos dados de *payload*. Quando o primeiro destes é recebido, ele e uma notificação informando que existem dados de *payload* a receber são enviados ao *Wrapper*. A cada ciclo, um dado é recebido e enviado ao *Wrapper*.

Caso não haja dados para receber, uma notificação solicitando que o *Wrapper* aguarde é enviada a este módulo.

Após o pacote ter sido recebido, uma notificação informando isto é enviada ao *Wrapper*.

O protocolo de recebimento de dados implementado no processo *Receiver* do módulo *Wrapper* é semelhante ao implementado para a NoC *Hermes HS*. Porém, durante o recebimento dos dados de *payload*, a máquina aguarda apenas pela notificação de recebimento do primeiro dado. Após isso, recebe-se um dado a cada ciclo.

5.2.3 NoC *YeaH*

Enquanto que nas NoCs *Hermes* o intervalo entre o recebimento de dois pacotes é sempre de no mínimo um ciclo, o mesmo não acontece na NoC *YeaH*. Nesta, caso existam pacotes a receber, os mesmos são obtidos sem qualquer intervalo. Assim, foi necessário alterar o protocolo de recebimento de dados para esta rede. O processo *Receiver* do módulo NI é ilustrado na Figura 27. Os mesmos três estados utilizados para representar a máquina das NoCs *Hermes 00, Hermes TB* e *Hermes VC* são utilizados para representar a máquina da NoC *YeaH*.

Após o pacote ter sido recebido, a máquina não volta ao estado inicial para a espera do endereço do próximo pacote. O processo de recebimento deste dado, bem como o envio do mesmo ao *Wrapper*, é realizado no estado *Recebe Payload*.

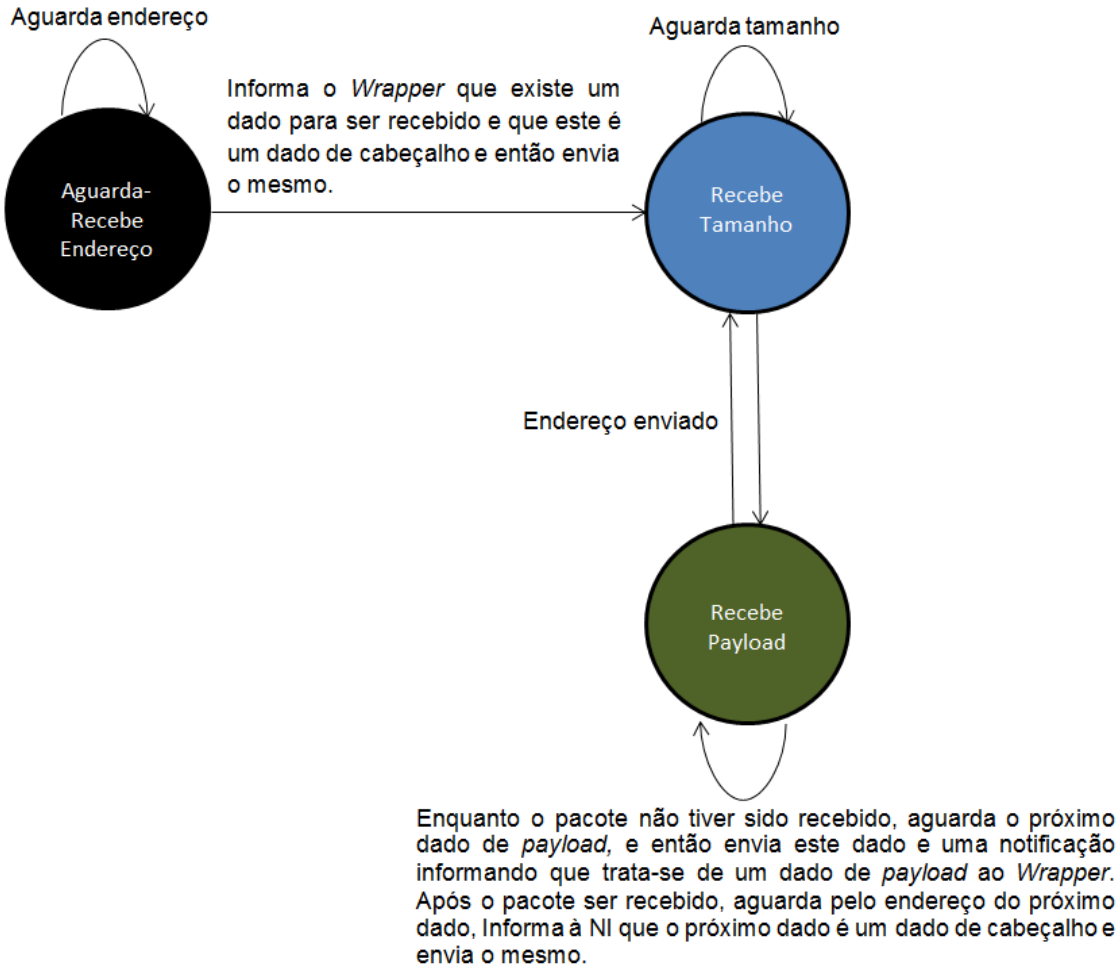

Figura 27 – Diagrama de transição de estados do processo Receiver do módulo NI para a NoC YeaH.

O protocolo de recebimento de dados implementado no processo Receiver do módulo *Wrapper* é semelhante ao implementado paras as NoCs *Hermes OO, Hermes TB* e *Hermes VC*. Porém, após o pacote ter sido recebido, de maneira similar ao que ocorre na NI, a máquina não volta ao estado inicial para a espera do endereço do próximo pacote. Ela aguarda por este dado no estado atual *Recebe Payload*.

6. Validação da Integração

A comunicação de cada uma das NoCs integradas à plataforma Nocbench durante este trabalho (conforme descrito no Capítulo 5)com o Simulador TG2 foi validada através de simulação funcional. A Figura 28 ilustra a topologia do cenário simulado, que envolve uma instância de uma NoC 2x2 (com quatro roteadores) omitindo enlaces, roteadores e interfaces externas que não são relevantes para o experimento. AFigura 29ilustra a transmissão de um dado da NI 0a NI 5 assumindo a NoC *Hermes HS* como meio de interconexão.

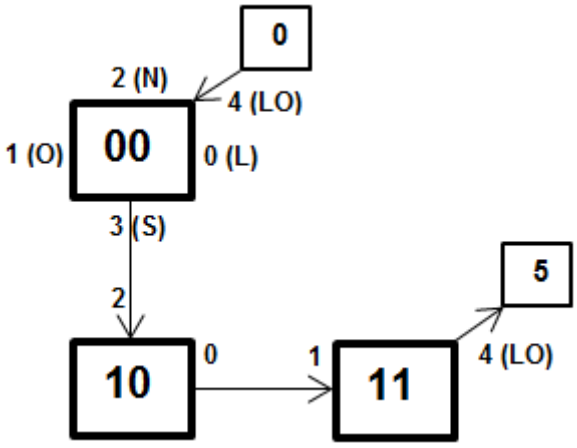

Figura 28 - NI e Roteadores envolvidos na validação básica comunicação entre NoCs do GAPH integradas e o simulador TG2.

Discute-se a seguir as etapas da simulação. Observe-se que a ordem utilizada para a descrição destas não necessariamente corresponde à ordem em que as mesmas foram executadas.

1. A NI0 aguarda pela ativação do sinal de envio de endereço sndr_av_in e então ativa o sinal de dado válido data_valid_out e copia o endereço do roteador destino (5) presente no barramento sndr_add_in para o barramento de dados conectado ao Roteador 00, net_data_out (257). O valor 5, que corresponde ao endereço do EP destino, é convertido para o valor 257na NI, que é equivalente ao valor binário: 00000000 00000000 00000001 00000001. Os 8 *bits* menos significativos e os 8 *bits* subsequentes deste valor,

correspondem respectivamente a coluna e a linha do roteador destino.

2. NIO ativa o sinal de dado válido data_valid_out e copia o tamanho de *payload* do pacote (4) presente no barramento sndr_tx_len_in para o barramento de dados net_data_out.
3. Após a desativação do sinal envio de endereço sndr_av_in, NI 0 inicia o envio dos dados de *payload* do pacote (0, 47, 6, 11). Para isso, ele copia os mesmos do barramento sndr_data_in para o barramento net_data_out.
4. Sempre no ciclo seguinte ao envio de um dado, a NI 0 aguarda por uma confirmação de recebimento do dado por parte do roteador 00 e então desativa o sinal de dado válido data_valid_out. Apenas no envio do primeiro pacote, um ciclo adicional é gasto antes do envio dos dados de *payload.* Isto é uma característica de projeto da NI, não estando relacionada à NoC.
5. O sinal sndr_full_out controla o envio dos dados de *payload* do pacote do *Wrapper* a NI. Toda vez que este sinal é desativado, o *Wrapper* disponibiliza o próximo dado do *payload* no barramento sndr_data_in.
6. Da porta Local (4) do Roteador 00 os dados seguem pela porta Sul (3) deste, e então pelas portas Norte (2) e Leste (0) do Roteador 10 e Oeste (1) e Local (4) do Roteador 11, e finalmente chegam à NI 5.
7. NI 5 aguarda pela ativação do sinal de dado válido net_data_valid_in e então desativa o sinal rdr_empty_out, de modo a informar ao *Wrapper* que existem dados a receber. Após isto, ativa o sinal de envio de endereço rdr_av_out, converte o endereço do roteador destino (257) para o endereço do EP destino (5) e disponibiliza o resultadono barramento de dados (rdr_data_out) conectado ao *Wrapper* do EP5.

8. A NI *5* aguarda pela ativação do sinal de dado válido net_data_valid_in e então copia o tamanho do *payload* (4) presente no barramento net_data_in para o barramento de dados rdr_data_out.

9. Após o recebimento do tamanho de *payload*, a NI 5 desativa o sinal de envio de endereço e inicia o recebimento dos dados de *payload* (0, 47, 6, 11). Para cada dado, ela aguarda a ativação do sinal net_data_valid_in e então copia o mesmo do barramento net_data_in para o barramento de dados rdr_data_out.

10. No ciclo seguinte ao recebimento de um dado, a NI 5 envia uma confirmação de recebimento do dado ao roteador 11 (sinal net_re_out).

11. O sinal rdr_free_out controla o envio dos dados de *payload* do pacote, da *NI ao Wrapper*. Toda vez que este sinal é ativado, a NI disponibiliza o próximo dado do *payload* no barramento rdr_data_out.

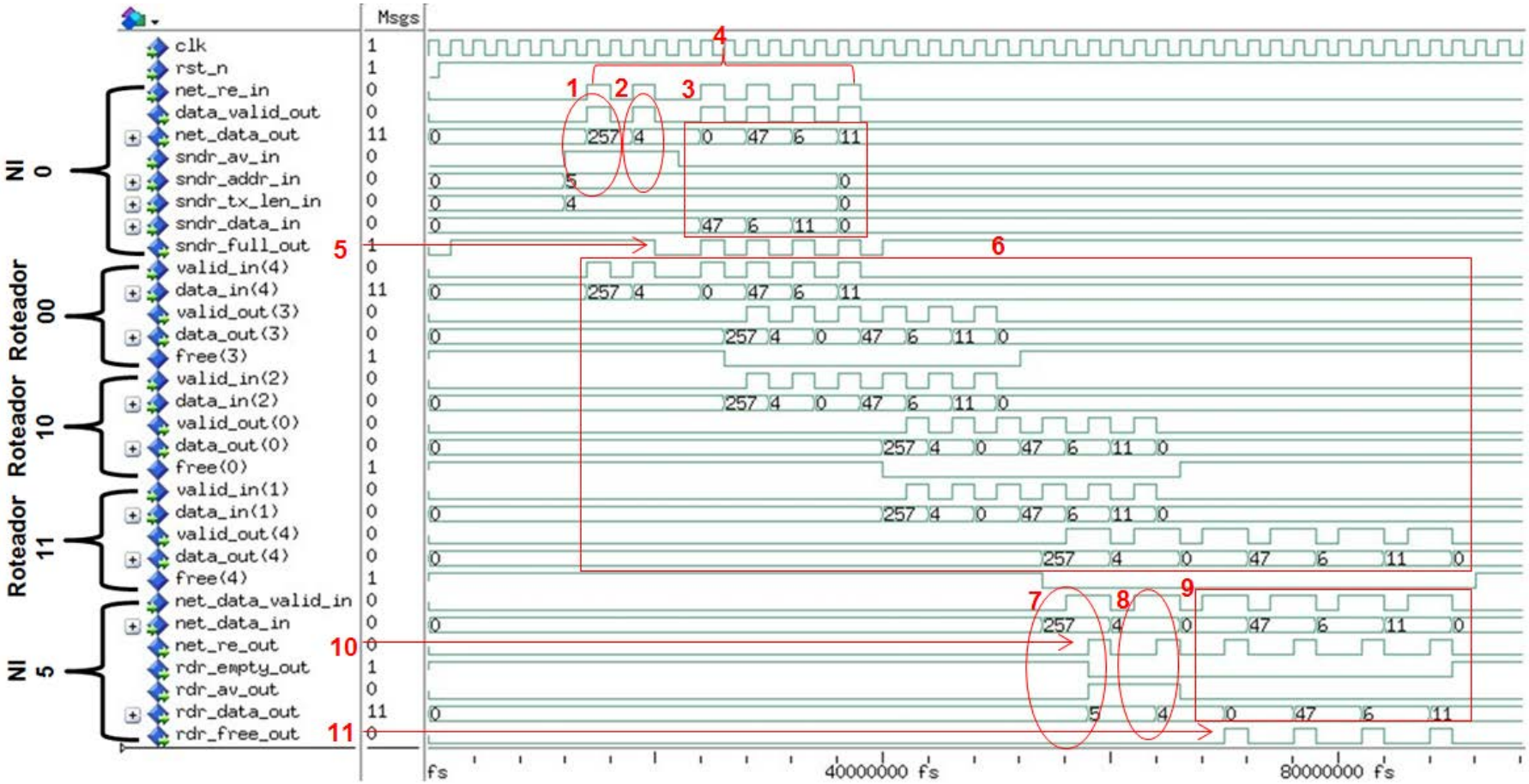

Figura 29 - Simulação usando o TG2 de uma transmissão de dados entre a NI do roteador 00 e a NI do roteador 11 para a NoC Hermes HS.

A Figura 30ilustra a simulação do mesmo cenário anterior, mas utilizando agora a NoC *Hermes 00*. As etapas 1 (Envio do endereço da NI 0 ao Roteador 00), 2 (Envio do tamanho de *payload* da NI 0 ao Roteador 00), 6 (Fluxo dos dados pela NoC) e 7 (Recebimento do endereço pela NI 5) do cenário anterior também são válidas para este.

Na etapa 3, o sinal de envio de endereço sndr_av_in, não é utilizado como referência para o início da transmissão dos dados de *payload.* Além disso, estes se encontram divididos nos barramentossndr_data_in e sndr_data2_in.

Na etapa 4, diferentemente do que ocorre na simulação da NoC *Hermes HS*, um dado só é enviado ao Roteador 00, após a NI 0 certificar-se de que existe espaço na fila de entrada deste roteador através da verificação do sinal net_re_in. Enquanto houver espaço, um dado é enviado a cada ciclo e o sinal de dado válido é mantido ativo durante todo o período de transmissão do pacote.

Em relação à etapa 5, assim como ocorre na simulação da NoC *Hermes HS*, o sinal sndr_full_out controla o envio dos dados de *payload* do pacote do *Wrapper* a NI. Porém, isto acontece apenas a partir do 3° dado.

Nas etapas 8 e 9 (Recebimento do tamanho de *payload* e dos dados deste respectivamente) a NI não aguarda pela ativação do sinal de dado válido net_data_valid_in, uma vez que o mesmo permanece ativo durante todo o período de transmissão do pacote.

Na etapa 10, o sinal net_re_out é mantido ativo durante todo o processo de recebimento de dados, de modo a indicar que existe espaço na fila do EP destino.

Por fim, na etapa 11, enquanto houver dados de *payload* a receber, o sinal rdr_free_out é mantido ativo. Assim, a cada ciclo um dado é recebido pelo *Wrapper*.

AFigura 31, a Figura 32e a Figura 33ilustram a simulação com as NoCs *Hermes VC*, *YeaH* e *Hermes TB*. Para a última foi utilizado o cenário apresenta-

do na Figura 30. A mesmas etapas utilizadas na simulação da NoC *Hermes OO* são realizadas para estas redes.

Na *Hermes VC* foi estabelecido que o envio dos dados a NoC dê-se pelo primeiro canal virtual (net_lane_out(0)). Uma vez que o envio de um único pacote não causa qualquer congestionamento na rede, os dados seguem por este mesmo canal até o destino.

Enquanto na NoC *YeaH* o roteamento (armazenamento na fila, arbitragem, roteamento) do endereço do pacote leva apenas 2 ciclos, nas demais redes são necessários 7 ciclos. Os dados seguintes passam pelo roteador com uma latência de um ciclo cada, exceto na NoC *Hermes HS*, onde dois ciclos são necessários.

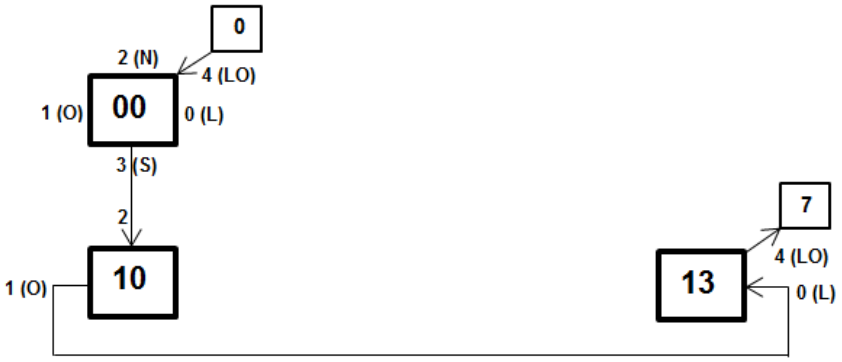

Figura 30 – NI e Roteadores envolvidos na validação básica da comunicação para a NoC Hermes TB do GAPH com topologia toro 2D e o simulador TG2.

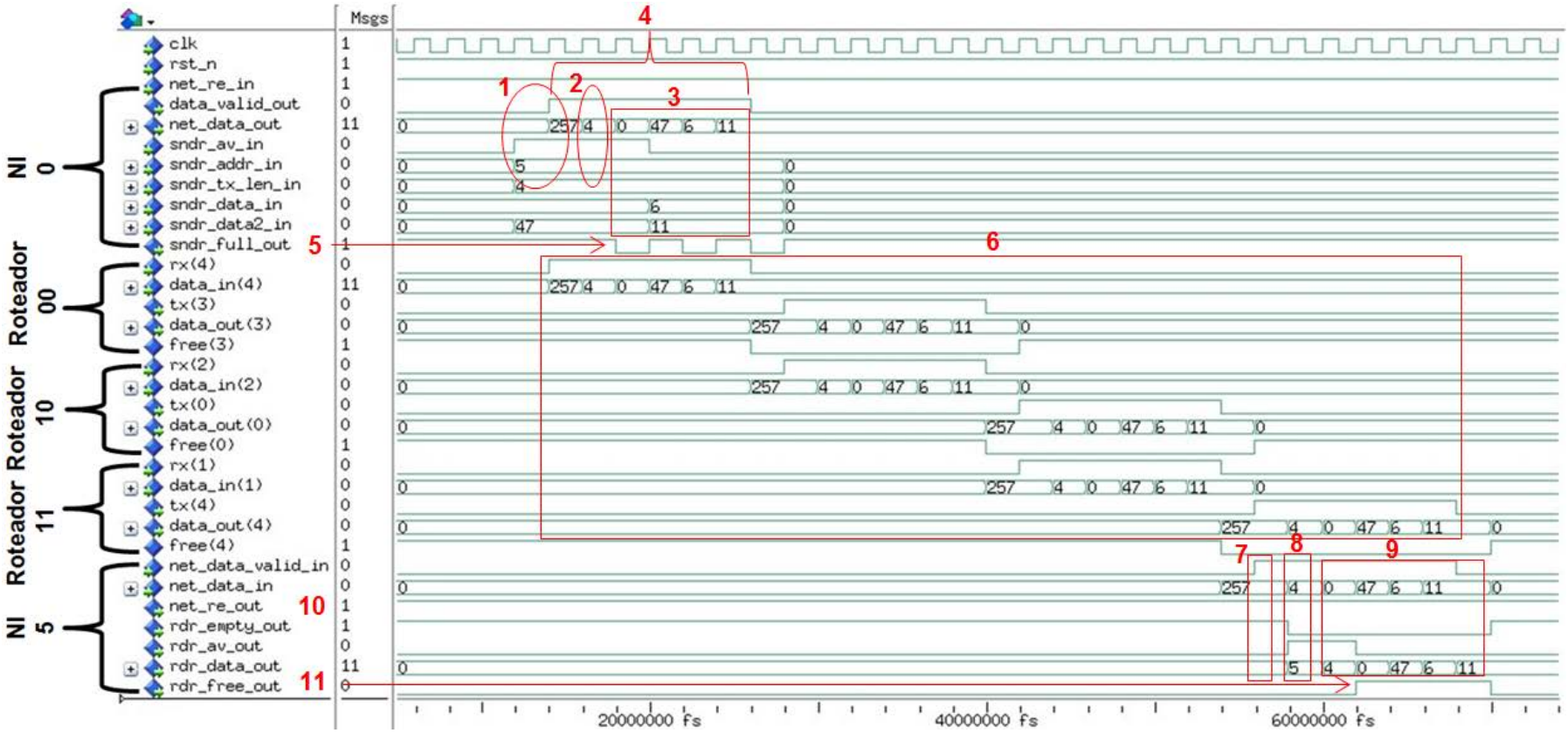

Figura 31 - Simulação usando o TG2 de uma transmissão de dados entre a NI do roteador 00 e a NI do roteador 11 com a NoC Hermes 00.

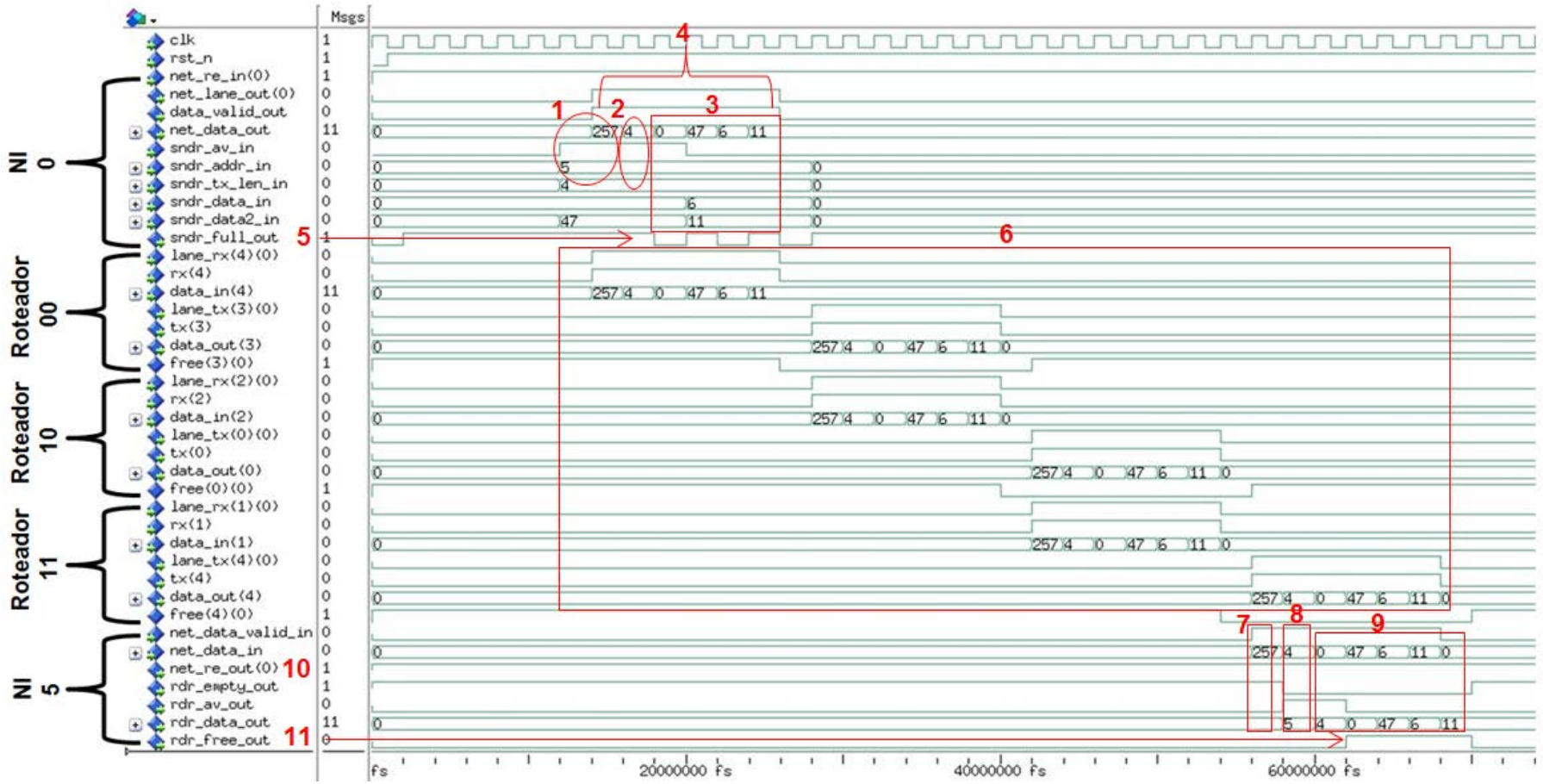

Figura 32 - Simulação usando o TG2 de uma transmissão de dados entre a NI do roteador 00 e a NI do roteador 11 com a NoC Hermes VC.

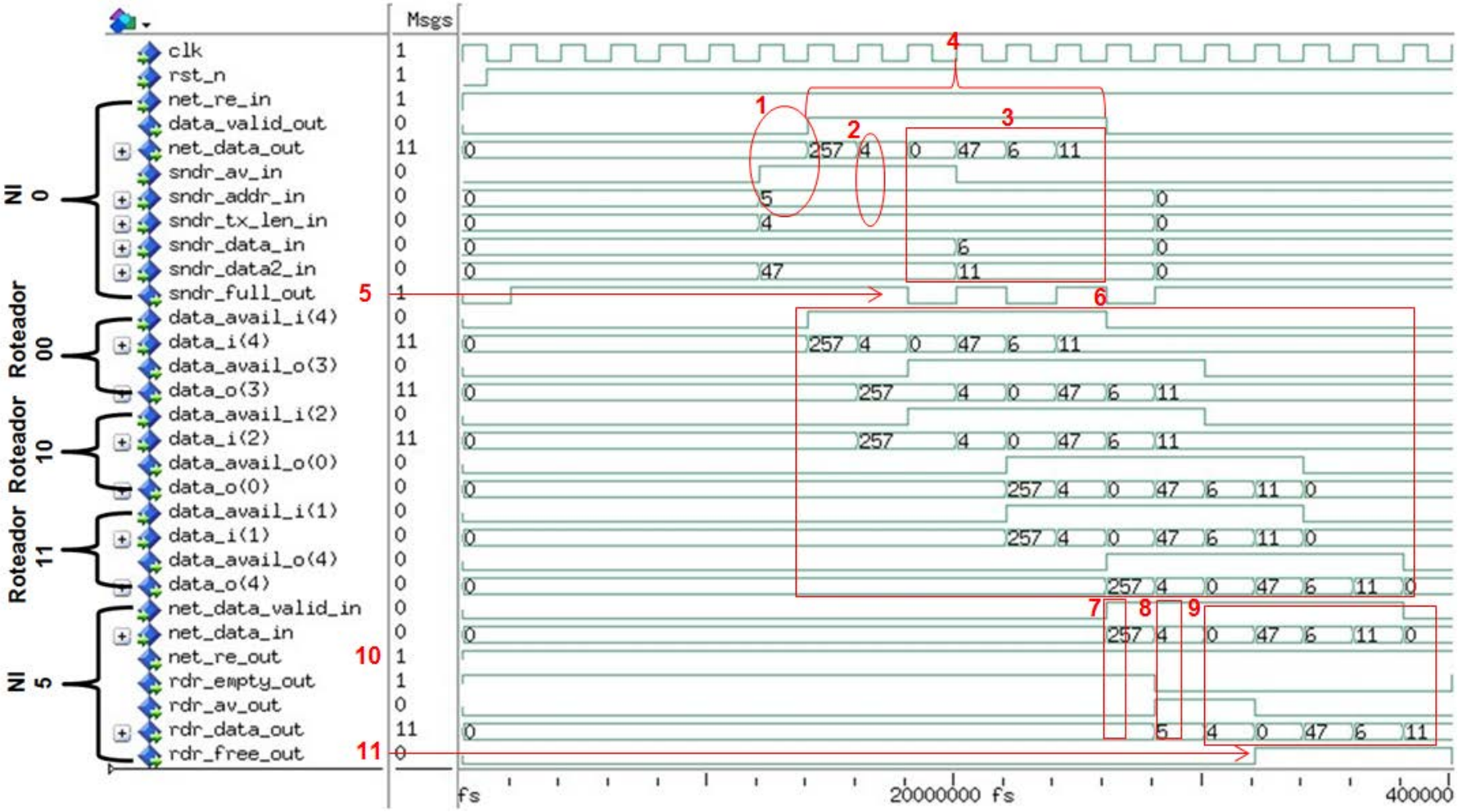

Figura 33 - Simulação usando o TG2 de uma transmissão de dados entre a NI do roteador 00 e a NI do roteador 11 com a NoC YeaH.

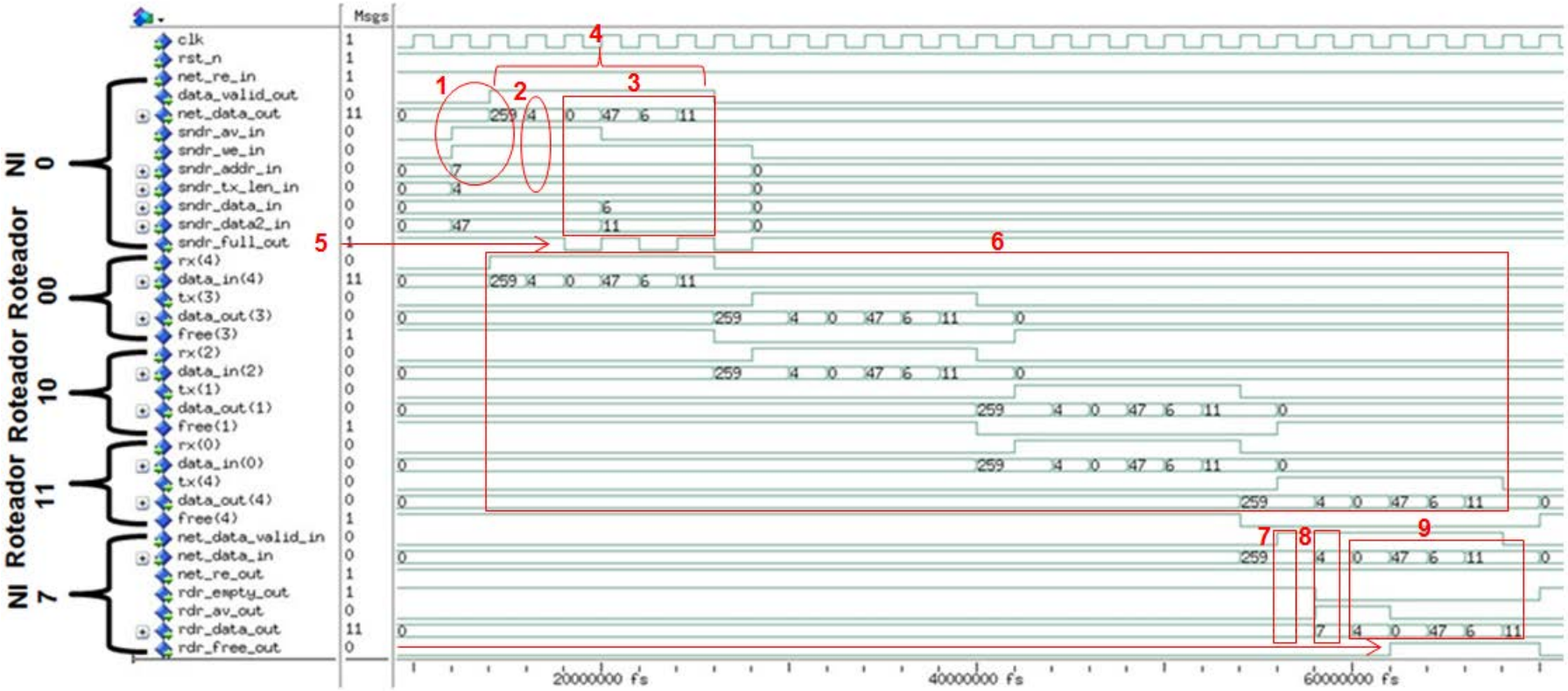

Figura 34 – Simulação usando o TG2 de uma transmissão de dados entre a NI do roteador 00 e a NI do roteador 13 com a NoC Hermes TB.

7. Comparação de Desempenho na Plataforma Nocbench

Este Capítulo divide-se em oito Seções. A Seção 7.1 descreve o refinamento do processo de avaliação da plataforma. As Seções 7.2 e 7.3 apresentam respectivamente as características das NoCs nativas da plataforma Nocbench e das NoCs integradas durante o presente trabalho. A Seção 7.4 descreve a configuração de benchmarks empregada nos experimentos. Por fim, as Seções 7.5 a 7.8apresentamos experimentos.

7.1 Refinamento do Processo de Avaliação de Desempenho da Plataforma

A plataforma Nocbench coleta e disponibiliza durante a simulação diversos valores básicos, conforme detalha a Tabela 8. No entanto, estes não podem ser facilmente empregados de forma direta na avaliação global de NoCs. Isto costuma ser uma meta importante para a maioria dos projetistas de sistemas que desejam comparar NoCs distintas a selecionar como arquitetura de interconexão *intrachip*. Três parâmetros importantes considerados neste tipo de avaliação são a latência média global, vazão média globale o*jitter* global. Apesar de a plataforma conter parâmetros de cálculo de latência, estes consideram mensagens trocadas entre tarefas de um mesmo EP e que, portanto, não passam pela NoC. Assim, estas medidas são inadequadas para a avaliação de NoCs. Assim, uma quarta contribuição deste trabalho foi a adição da capacidade de computar os parâmetros de latência, vazão e *jitter descritos acima* à plataforma Nocbench.

De acordo com Salminen [SAL07], a definição de latência de rede é ambígua. Dependendo do trabalho, a medição deste parâmetro é realizada com base no tempo que uma palavra, cabeçalho do pacote, pacote ou mensagem leva para transitar entre dois pontos do sistema considerado. Além disso, o tempo em

que os dados permanecem na fila de entrada do sistema pode ou não ser considerado.

O cálculo de latência que se considera aqui corresponde ao tempo decorrido desde o momento em que a primeira tentativa de injeção do primeiro *flit* de um pacote na rede é realizada, até o momento em que o primeiro *flit* de *payload* deste pacote é armazenado na fila de entrada do EP destino(Equação 1). Para a comparação dos diferentes projetos de NoCs considerou-se a média destas latências(Equação 2). Os cálculos de latência máxima e de latência mínima (maior e menor latência dentre todos os pacotes que trafegaram pela NoC durante a execução da aplicação) também foram incluídos na plataforma. A avaliação da latência máxima é de extrema importância em aplicações do tipo *real-time,* que exigem que as latências de todos os pacotes estejam abaixo de certo valor.

Como já mencionado na Seção 3.2, no TG2 o primeiro dado de *payload* de um pacote corresponde ao identificador da mensagem ao qual este pacote pertence. Uma vez que os pacotes não contêm dados de controle como o número de sequência, este identificador é utilizado para associar um pacote a uma mensagem e com isso determinar a latência do mesmo. Isto justifica o porquê da latência implementada considerar o momento em que o primeiro *flit* de *payload* e não o último *flit* do cabeçalho é armazenado na fila de entrada do EP destino.

Para cada pacote calcula-se a latência do cabeçalho do mesmo:

$$latencia_{cabecal\ ho} = tfc - tic \qquad \text{(Equação 1)}$$

Onde:

tic: momento em que a primeira tentativa de injeção do primeiro *flit* de um pacote na rede é realizada.

tfc: momento em que o primeiro *flit* de *payload* deste pacote é armazenado na fila de entrada do EP destino.

$$latencia_{media} = \frac{\sum latencia_{cabecal\ ho}}{np} \qquad \text{(Equação 2)}$$

Onde:

np: número de pacotes que trafegaram pela rede.

O *jitter* corresponde ao desvio padrão da latência média. Um valor alto deste parâmetro geralmente indica que algumas mensagens permaneceram bloqueadas na rede por um longo período de tempo [DUA03], o que compromete a execução de certas classes de aplicações tais como multimídia [TED05].

Para cada pacote, a diferença da latência média da latência do cabeçalho é primeiro elevada ao quadrado:

$$diferenca = (latencia_{cabecal\ ho} - latencia_{media})^2 \qquad \text{(Equação 3)}$$

Em seguida calcula-se o *jitter* com a (Equação 4)

$$jitter = \sqrt{\frac{\sum diferenca}{np}} \qquad \text{(Equação 4)}$$

Onde:

np: número de pacotes que trafegaram pela rede.

A vazão corresponde à quantidade máxima de dados entregues por unidade de tempo. Ela também pode ser definida como a quantidade máxima de dados aceitos pela rede [DUA03]. Uma vez que este parâmetro está diretamente relacionado ao tamanho das mensagens que trafegam pela rede, o cálculo de vazão implementado considera a média ponderada (Equação 7) da vazão de todos os pares de tarefas comunicantes de uma determinada aplicação(Equação 6). O cálculo da vazão destes pares utiliza o cálculo de latência expresso pela(Equação 5).

Para cada pacote de um par de tarefas comunicantes a latência do mesmo é calculada por:

$$latencia_{pacote} = tfp - tip \qquad \text{(Equação 5)}$$

Onde:

tip: momento em que a primeira tentativa de injeção do primeiro *flit* de um pacote na rede é realizada.

tfp: momento em que o último *flit* de *payload* deste pacote é armazenado na fila de entrada do EP destino. Por outro lado, a vazão é calculada por:

$$vazao_{par} = \frac{bits\_transmitidos\_par}{\sum latencia_{pacote}} \qquad \text{(Equação 6)}$$

Onde:

bits_transmitidos_par: número de bits transmitidos da tarefa fonte a tarefa destino.

$$vazao_{medi\ a_{ponderada}} = \frac{\sum(vazao_{par} * bits\_transmitidos\_par)}{\sum bits\_transmitidos\_par} \qquad \text{(Equação 7)}$$

Os parâmetros de avaliação da plataforma consideram apenas pacotes recebidos em ordem. Assim, na avaliação de desempenho de uma determinada NoC, apenas algoritmos determinísticos podem ser utilizados. A abordagem de inserir dados de controle, como o tempo inicial e o número de sequência no *payload* do pacote foi descartada, uma vez que, isto acarretaria a alteração das características de tráfego dos modelos de aplicação, o que não condiz com o comportamento da maioria das aplicações, que utilizam pacotes pequenos para tarefas de sinalização e controle, por exemplo.

7.2 NoCs FH Mesh e Ase Mesh

Como não existem publicações a respeito destas NoCs, procurou-se obter informações a respeito das mesmas com os Autores da plataforma Nocbench. Segundo os mesmos, tanto FH Mesh quanto Ase Mesh possuem:

- Topologia malha 2D com cada roteador conectado aos demais por até quatro portas bidirecionais (Leste, Oeste, Sul e Norte) e ao EP por uma porta bidirecional (Local);
- Lógica de controle (arbitragem e roteamento) centralizada;

- Esquema de arbitragem baseado em prioridade fixa, sendo que a ordem das portas da maior para a menor prioridade é a seguinte: Norte, Oeste, Sul, Leste e Local.
- Algoritmo de roteamento YX;
- Suporte a estratégias *wormhole* e *store-and-forward.*

Enquanto que a NoC FH Mesh contém um *buffer* em cada canal de saída, a NoC Ase Mesh possui um *buffer* em cada canal de entrada. Além disso, diferentemente da FH Mesh (que utiliza a informação de tamanho de *payload* para determinar se o dado recebido pertence ao pacote atual ou é o primeiro dado de um novo pacote),a NoC Ase Mesh faz uso de dois *bits* adicionais em cada pacote de rede, um antes do primeiro dado de *payload* e outro após o último dado de *payload*.

7.3 NoCs *Hermes e YeaH*

Nas NoCs Hermes, cada roteador possui um conjunto de até quatro portas bidirecionais (Leste, Oeste, Sul e Norte) para a interconexão com os demais roteadores e uma porta bidirecional para a conexão com um EP (Local). O roteador é constituído por lógica de controle e um conjunto de portas de comunicação. A lógica de controle é composta por uma lógica de roteamento e por uma lógica de arbitragem. As portas de comunicação são compostas por canais de entrada e saída, sendo que cada canal de entrada contém ainda um *buffer* para armazenamento temporário.

Os dois primeiros *flits* de qualquer pacote em redes Hermes constituem o cabeçalho do pacote e contêm respectivamente as seguintes informações: endereço do roteador destino, na metade inferior do primeiro *flit*, e o número de *flits* de *payload*, no segundo. A estratégia de chaveamento empregada é a *wormhole*. Nas versões com arbitragem e roteamento centralizado, quando um roteador recebe o primeiro *flit* de cabeçalho, a lógica de arbitragem é executada e se a requisição desse pacote for aceita, a lógica de roteamento é executada para conectar a porta de entrada à porta de saída determinada pelo algoritmo, caso esta esteja disponível. Os algoritmos de roteamento disponíveis são ba-

seados no modelo *turn mode*/[GLA92]e a disponibilidade de cada um destes varia de acordo com a versão da NoC. A lógica de arbitragem centralizada (presente em todas as NoCs Hermes, exceto na Yeah) utiliza o esquema *round robin,* onde a prioridade de uma porta depende da última porta a ter uma solicitação atendida. Por exemplo, caso a porta de entrada com índice 4 (Local) tenha sido a última a ser atendida, a porta 0 (Leste) terá prioridade maior seguida, pelas portas 1 (Oeste), 2 (Norte), e 3 (Sul), atribuindo certo grau de justiça no processo de seleção de requisições de aceso ao roteamento. Se uma porta de entrada tiver sua solicitação de arbitragem atendida, ela sempre receberá a seguir a menor prioridade, mesmo que não consiga a conexão à porta (por exemplo, se a porta está ocupada com uma conexão a outra porta de entrada) de saída desejada. Esta última característica faz com que uma porta de entrada possa sofrer postergação indefinida, em condições muito desfavoráveis de concorrência entre as portas de entrada da NoC.

A NoC *Hermes* dá suporte a duas topologias de rede: *(i)* malha 2D e *(ii)* toro 2D. A primeira conta com dois esquemas de controle de fluxo: *Handshake* (*Hermes HS* [MOR04]) e *On-Off* (*HermesOO*). A segundadá suporte apenas ao esquema *On-Off* (*Hermes TB*[SCH07]). Existe ainda uma versão da rede *Hermes OO*, que utiliza canais virtuais (2 ou 4), denominada *Hermes VC*[MEL05b].A NoC *YeaH* é uma versão da NoC *HermesOn-Off*recente, com arbitragem e roteamento distribuído e foi desenvolvida durante o segundo semestre de 2013.

7.4 *Benchmarks* Utilizados nos Experimentos

Foram selecionadas oito aplicações para a realização dos experimentos, sendo quatro da suíte MCSL 1.1: H263e, H264dl, Sample e Sparse; e quatro aplicações adicionais da plataforma Nocbench: AV, MPEG4 Decoder, Radio Sys e VOPD. Os traces das aplicações do MCSL correspondem a 10 execuções de cada aplicação. Já as aplicações adicionais foram limitadas a uma única execução. O mapeamento não foi alterado. Em todos os experimentos, foram utilizados EPs com frequência de operação de 500 MHz, capazes de realizar uma operação (sobre números de ponto flutuante, sobre números inteiros ou sobre memórias) por ciclo.

A Tabela 11 apresenta o tamanho máximo de mensagem presente nas aplicações. Observe que este valor determina o tamanho máximo de pacote de rede para cada aplicação. A Tabela 12 apresenta o total de dados que trafegaram pela NoC para cada par *<aplicação, tamanho de pacote>*. Observe que nas aplicações Sparse e H264dl, e na aplicação Sample, o total de dados que trafegaram pela rede foi igual, para pacotes com tamanho superior a 34 e 18 *flits* (*de 32 bits*) respectivamente. No TG2, caso o tamanho de uma mensagem seja menor do que o tamanho de pacote especificado, o pacote enviado será do tamanho desta mensagem.

Tabela 11 - Tamanho Máximo de Mensagem (em flits de 32 bits) presente em cada uma das aplicações utilizadas nos experimentos.

Aplicação	**Tamanho Máximo de Mensagem (em *flits* de 32 *bits*)**	**Tamanho Máximo de Pacote**
Radio Sys	320.000	258
H263e	277	258
Mpeg4 Decoder	116.480	258
VOPD	128.000	258
Sample	29	18
AV	9.131	258
Sparse	58	34
H264dl	58	34

Tabela 12 - Número total de dados que trafegaram pela NoC (em flits de 32 bits), para os tamanhos de pacotes especificados na segunda linha da tabela, em cada uma das aplicações utilizadas nos experimentos.

Aplicação	**Tamanho do Pacote (em *flits* de 32 *bits*)**							
	4	**6**	**10**	**18**	**34**	**66**	**130**	**258**
Radio Sys	6.169.996	4.627.500	3.856.250	3.470.626	3.277.814	3.181.408	3.133.204	3.109.108
MPEG4 Decoder	1.774.592	1.330.944	1.109.120	998.208	942.752	915.024	901.164	894.236
VOPD	1.758.211	1.318.659	1.098.883	988.995	934.051	906.579	892.843	885.975
AV	114.716	86.048	71.714	64.554	60.968	59.180	58.292	57.848
H263e	1.844.680	1.384.780	1.154.820	1.039.940	982.480	952.940	937.820	930.260
Sample	466.110	353.130	298.530	266.310	248.490	248.490	248.490	248.490
Sparse	57.030	42.990	35.970	32.650	30.610	29.510	29.510	29.510
H264dl	32.780	24.740	20.640	18.700	17.620	16.980	16.980	16.980

A Tabela 13 mostra que se têm sete NoCs diferentes, considerando-se os seguintes parâmetros: (i) o projeto do buffer: otimizado (sem intervalo entre o envio de pacotes subsequentes) ou não otimizado (com intervalo de um ciclo entre o envio de pacotes subsequentes); (ii) a arbitragem e o roteamento: centralizado ou distribuído; (iii) o esquema de arbitragem: *round robin* ou com prioridade fixa; (iv) a topologia de rede: malha ou toro; (v) a quanti-

dade de canais virtuais: dois ou nenhum; o (vi) o esquema de controle de fluxo: *handshake* ou *on-off*; e (vii) a localização dos *buffers*: portas de entrada ou saída. Os experimentos apresentados a seguir buscam mostrar o impacto destas características no desempenho das NoCs.

Tabela 13 - Características das NoCs avaliadas: (i) projeto do buffer; (ii) arbitragem e roteamento; (iii) esquema de arbitragem; (iv) topologia de rede; (v) quantidade de canais virtuais; (vi) esquema de controle de fluxo; (vii) localização dos buffers.

	(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)
YeaH	otimizado	distribuído	round-robin	malha	nenhum	oo	entrada
Ase Mesh	otimizado	centralizado	prioridade fixa	malha	nenhum	oo	entrada
Hermes OO	não otimizado	centralizado	round-robin	malha	nenhum	oo	entrada
Hermes TB	não otimizado	centralizado	round-robin	toro	nenhum	oo	entrada
Hermes VC	não otimizado	centralizado	round-robin	malha	dois	oo	entrada
Hermes HS	não otimizado	centralizado	round-robin	malha	nenhum	hs	entrada
FH Mesh	não otimizado	centralizado	prioridade fixa	malha	nenhum	oo	saída

7.5 Experimento 1: Impacto do Projeto do *Buffer* e da Arbitragem e Roteamento Distribuído na Vazão e Latência Média

Neste experimento, procurou-se observar o quanto o projeto do *buffer* e a arbitragem e o roteamento distribuídos afetam a vazão e a latência média das NoCs. Para isso, foram considerados pacotes de 4, 6, 10, 18, 40, 66, 130 e 258 *flits*. Estes valores incluem os dois *flits* de cabeçalho(endereço do roteador destino e tamanho do *payload* do pacote).As NoCs foram configuradas com tamanho 4x4, largura de *flit* de 32 *bits* e profundidade de *buffer* de 4 *flits*.

7.5.1 Vazão Média

Em teoria, o pico de vazão de um roteador da NoC *Hermes HS* é de 4 Gbps ((500 MHz/2) * 5 portas * 32 bits). Para as NoCs *Hermes OO*, *Hermes VC*, *Hermes TB*, *YeaH*, *FH Mesh* e *Ase Mesh* este valor dobra (8 Gbps (500 MHz * 5 portas * 32 bits)), visto que apenas um ciclo é necessário para o envio de um *flit* a um roteador vizinho. Assim, para as NoCs 4x4 consideradas neste experimento, o pico teórico de vazão é de 64 (16 * 4) e 128 (16 * 8) Gbps respectivamente.

Em todas as aplicações analisadas a NoC *YeaH* ofereceu uma vazão média superior as demais, como ilustra a Figura 35 e a Figura 36. Isto se deve ao

fato da mesma possibilitar a arbitragem e o roteamento de até 5 flits (1 para cada porta) simultaneamente, considerando um caso ótimo, onde o destino de cada um dos *flits* presente em uma porta de entrada seja uma porta de saída distinta dos demais.

Entre as soluções com controle centralizado (arbitragem e roteamento) destaca-se a NoC *Ase Mesh*. Esta, assim como a NoC *YeaH* possui *buffers* cuja máquina de escrita é otimizada, de modo a não ser necessário aguardar um intervalo de um ciclo entre o envio de pacotes subsequentes.

Analisando-se as diferenças entre os resultados de vazão das NoCs se pode ter uma ideia do impacto aproximado causado por certas características de projeto. Sabendo-se que *(i) YeaH* e *Ase Mesh* possuem *buffers* otimizados; *(ii)Ase Mesh* e *Hermes OO* possuem arbitragem e roteamento centralizado; e *(iii)YeaH* e *Hermes OO* não possuem nenhuma das características citadas nos dois itens anteriores em comum; deduz-se então que a diferença entre as vazões das NoCs *YeaH* e *Ase Mesh* representa o impacto aproximado do controle distribuído em relação ao controle centralizado, e a diferença entre as vazões das NoCs *Ase Mesh* e *Hermes OO* representa o impacto aproximado da otimização do *buffer*. Assim, o impacto da arbitragem e do roteamento distribuído é inferior ao impacto da otimização do *buffer*. No entanto, é preciso ressaltar que isto é uma aproximação, visto que, a NoC *Ase Mesh* utiliza um esquema de arbitragem diferente das demais, e que os projetos destas redes foram propostos e desenvolvidos por diferentes Autores em diferentes contextos.

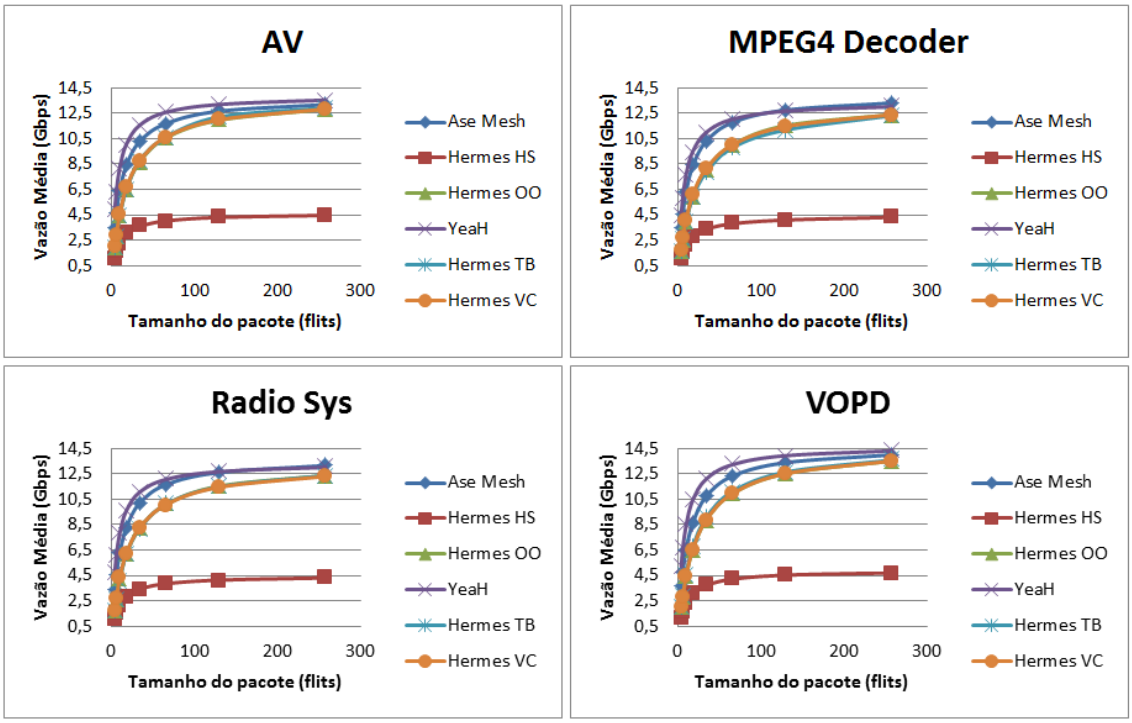

Figura 35 - Vazão Média (em Gbps) para aplicações adicionais do Nocbench.

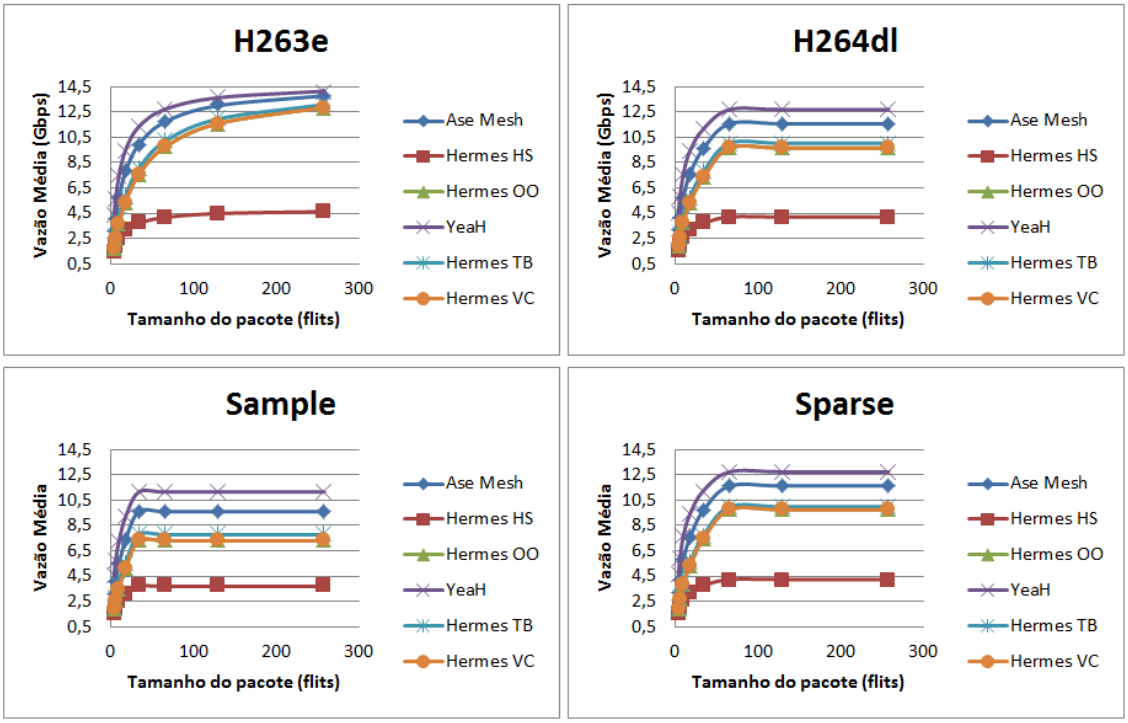

Figura 36 - Vazão Média (em Gbps) para aplicações do MCSL.

A Tabela 14 e aTabela 15 apresentam respectivamente a diferença de vazão média das NoCs *YeaH* e Ase *Mesh* em relação às demais.A diferença média da vazão da NoC YeaH em relação as NoCs Hermes OO e Ase Mesh, e desta em relação à anterior é de respectivamente 36.07%, 15.57% e 25.69%.

Tabela 14 - Diferença percentual de vazão média da NoC YeaH em relação relação as demais redes, considerando todos os tamanhos de pacotes e todas as aplicações.

	Ase Mesh	**Hermes TB**	**Hermes OO**	**Hermes VC**	**Hermes HS**
AV	14,77	31,35	31,96	30,83	70,81
H263e	15,83	32,52	36,45	36,33	67,35
H264dl	18,00	35,44	39,32	39,25	66,91
Mpeg4 Decoder	10,01	33,75	32,83	31,75	71,03
Radio Sys	12,19	32,33	32,28	31,91	70,86
Sample	20,15	39,22	43,64	43,62	66,87
Sparse	18,28	36,82	39,02	39,01	66,79
VOPD	15,33	31,77	33,05	32,89	71,08
Média	**15,57**	**34,15**	**36,07**	**35,70**	**68,96**

Sabendo-se que: *(i)YeaH* e *Ase Mesh* possuem *buffer* otimizados; *(ii)Ase Mesh* e *Hermes OO* possuem arbitragem e roteamento centralizado; e *(iii)YeaH* e *Hermes OO* não possuem nenhuma das características citadas nos dois itens anteriores em comum; deduz-se então que a diferença entre as vazões das NoCs *YeaH* e *Ase Mesh* representa o impacto aproximado do controle distribuído em relação ao controle centralizado, e a diferença entre as vazões das NoCs *Ase Mesh* e *Hermes OO* representa o impacto aproximado da otimização do *buffer*.

Tabela 15 - Diferença percentual de vazão média da NoC Ase Mesh em relação as demais redes (exceto a NoC YeaH), considerando todos os tamanhos de pacotes e todas as aplicações.

	Hermes TB	**Hermes OO**	**Hermes VC**	**Hermes HS**
AV	21,18	21,86	20,39	65,78
H263e	21,12	25,99	25,83	60,67
H264dl	22,11	26,98	26,88	59,17
Mpeg4 Decoder	27,92	26,99	25,63	67,86
Radio Sys	24,68	24,64	24,11	66,83
Sample	24,40	29,98	29,97	58,17
Sparse	23,63	26,39	26,38	58,84
VOPD	21,16	22,72	22,51	65,88
Média	**23,27**	**25,69**	**25,21**	**62,90**

Assim, o impacto da arbitragem e do roteamento distribuído é inferior ao impacto da otimização do *buffer*. No entanto, é preciso ressaltar que isto é uma aproximação, visto que, a NoC *Ase Mesh* utiliza um esquema de arbitragem diferente das demais, e que os projetos destas redes foram propostos e desenvolvidos por diferentes Autores em diferentes contextos.

7.5.2 Latência Média

A Figura 37 e a Figura 38ilustram a variação da latência média das NoCs em função de diferentes tamanhos de pacotes de rede. Observa-se, para as aplicações adicionais da plataforma Nocbench, que o aumento da latência acompanha o aumento do tamanho do pacote. Nas aplicações do MCSL o que ocorre é diferente, a latência é constante.

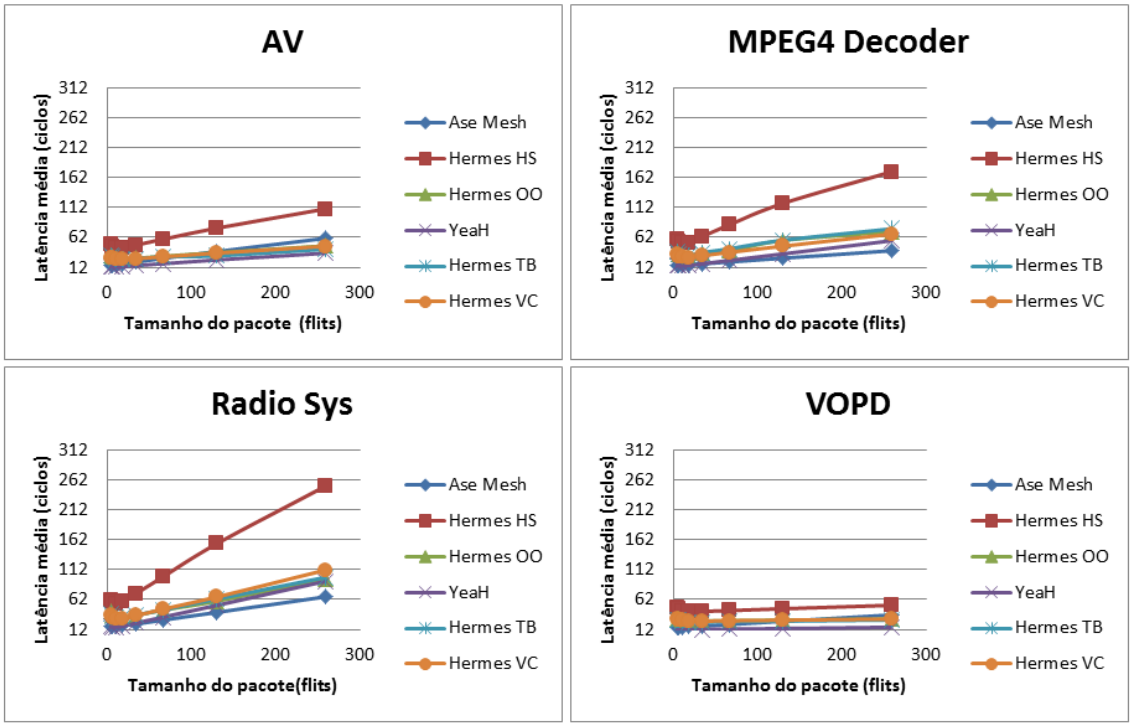

Figura 37 – Latência Média (em ciclos) para as aplicações adicionais do Nocbench.

Considerando-se a média de todas as aplicações apresentada na

Tabela16, a NoC YeaH apresentou uma latência inferior às demais. Contudo, através da análise individual de cada aplicação, se pode perceber que nas aplicações *MPEG4 Decoder* e *Radio Sys*, a NoC *Ase Mesh* obteve melhores resultados. Analisando a diferença de latência para cada tamanho de pacote de rede nestas aplicações (

Tabela17), observa-se que a NoC *Ase Mesh* supera a NoC YeaH quando utilizados pacotes com tamanho igual ou superior a 34 *flits*.

Os resultados de latência obtidos com as aplicações *MPEG4 Decoder* e *Radio Sys* corroboram a afirmação de Duato [DUA03] de que a latência média dos

pacotes em uma NoC com chaveamento *wormhole* sofre uma influência muito maior do tráfego do que de qualquer outro parâmetro de projeto.

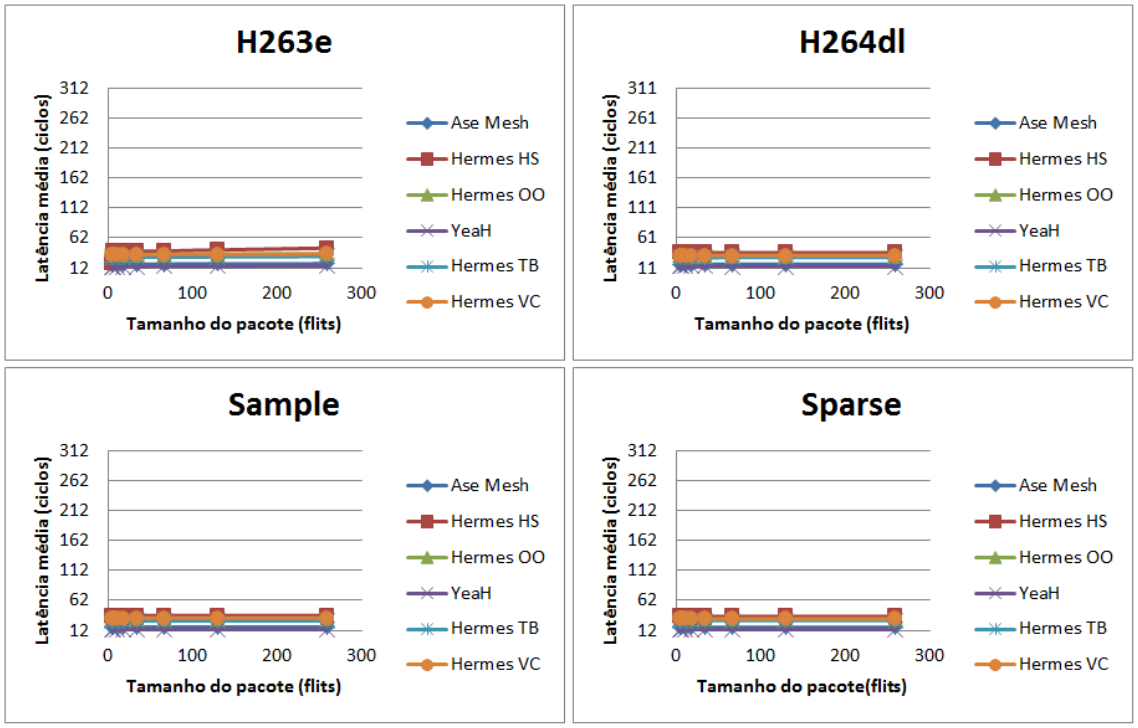

Figura 38 – Latência Média (em ciclos) para as aplicações adicionais do Nocbench

Tabela 16 – Diferença percentual de latência média das NoCs Ase Mesh, Hermes TB, Hermes OO, Hermes VC e Hermes HS em relação a NoC YeaH, considerando todos os tamanhos de pacotes e todas as aplicações.

	Ase Mesh	Hermes TB	Hermes OO	Hermes VC	Hermes HS
AV	40,12	82,65	88,43	79,66	239,33
H263e	27,28	110,33	141,16	139,60	193,15
H264dl	29,62	111,07	137,04	136,62	170,30
Mpeg4 Decoder	-4,31	99,88	95,75	76,68	253,64
Radio Sys	-7,70	67,04	67,22	62,75	220,29
Sample	29,41	107,67	139,89	139,75	169,99
Sparse	30,24	116,99	136,30	136,25	166,73
VOPD	57,26	107,63	121,05	119,17	259,49
Média	**25,24**	**100,41**	**115,85**	**111,31**	**209,12**

Assim como na avaliação de vazão apresentada na Seção anterior, o impacto aproximado do controle distribuído em relação ao controle centralizado e dos *buffers* otimizados em relação aos não otimizados na latência média correspondem respectivamente às diferenças entre os valores de latência das NoCs *Ase Mesh, YeaH*, *Hermes OO* e *Ase Mesh.*

Tabela 17 – Diferença percentual de latência média da NoC YeaH em relação à NoC Ase Mesh, para cada tamanho de pacote de rede.

	MPEG4 Decoder	Radio Sys
4	3,34	-1,76
6	8,65	3,66
10	9,41	5,20
18	6,49	3,93
34	-1,10	-6,80
66	-10,78	-15,27
130	-21,04	-22,48
258	-29,47	-28,11

Tabela 18 – Diferença percentual de latência média das NoCs Hermes TB, Hermes OO, Hermes VC e Hermes HS em relação a Ase Mesh, considerando todos os tamanhos de pacotes e todas as aplicações.

	Hermes TB	Hermes OO	Hermes VC	Hermes HS
AV	35,31	38,93	32,12	147,08
H263e	65,25	89,47	88,24	130,42
H264dl	62,83	82,87	82,55	108,53
Mpeg4 Decoder	108,26	103,32	83,50	274,77
Radio Sys	78,68	78,45	75,46	249,67
Sample	60,48	85,37	85,26	108,63
Sparse	66,61	81,43	81,39	104,80
VOPD	39,25	48,37	47,03	139,84
Média	**64,59**	**76,03**	**71,94**	**157,97**

7.6 Experimento 2: Impacto da Topologia de Rede na Vazão e Latência Média

Neste experimento foram consideradas apenas as aplicações da suíte MCSL, uma vez que a mesma fornece um conjunto de *traces* distinto para cada topologia de rede. A diferença destes *traces* é resultado do mapeamento de tarefas adotado, que busca maximizar o desempenho para uma determinada topologia de rede. Comparou-se a vazão e a latência média das NoCs *Hermes OO* e *Hermes TB* cujas diferenças residem, na topologia de rede e no algoritmo de roteamento.

Em virtude de o algoritmo YX poder levar a situações de *deadlock* em topologias *torus*, a NoC *Hermes TB* utiliza o algoritmo TRANC [RAH11]. A Tabela 19 e a

Tabela 17 apresentam a diferença de vazão (em Gbps) e latência (em ciclos) entre as duas NoCs citadas.

Boa parte das tomadas de decisões do algoritmo TRANC é baseada em comparações com valores constantes. Em NoCs com tamanhos maiores esta característica pode resultar no encaminhamento dos pacotes por caminhos mais longos, levando a uma queda de desempenho. No entanto, no caso de NoCs com tamanhos relativamente menores, como o utilizado neste experimento, o comportamento do mesmo assemelha-se ao comportamento do algoritmo YX. Aliando-se a isto o fato da descrição da NoC *Hermes TB* ser praticamente igual a da NoC *Hermes* OO, sendo ambas desenvolvidas pelo mesmo grupo de pesquisa, pode-se afirmar que a diferença dos resultados entre estas redes, corresponde ao impacto imposto pela topologia de rede.

Tabela 19- Diferença percentual de vazão média da NoC Hermes TB em relação a NoC Hermes OO para as aplicações do MCSL, considerando todos os tamanhos de pacotes.

	Diferença Vazão
H263e	6,60
H264dl	6,49
Sample	7,49
Sparse	3,75
Média	**6,09**

Tabela 20 – Diferença percentual de latência média da NoC Hermes OO em relação a NoC Hermes TB para as aplicações do MCSL, considerando todos os tamanhos de pacotes.

	Diferença Latência
H263e	14,66
H264dl	12,31
Sample	15,51
Sparse	8,90
Média	**12,85**

7.7 Experimento 3: Impacto da Localização do *Buffer* na NoC na Vazão e Latência Média

Com este experimento buscou-se avaliar o que é mais vantajoso:*buffers* nas portas de entrada ou nas portas de saída dos roteadores. Para isso, comparou-se a vazão e a latência média das NoCs *Hermes OO, Hermes VC e FH Mesh.*

A primeira possui *buffers* nas portas de entrada, a outra nas portas de saída.Utilizou-se apenas pacotes de 4 e 6 *flits*, devido a limitações da NoC *FH Mesh*. A Tabela 21 e a Tabela 22 apresentam respectivamente a diferença da vazão e da latência média da NoC FH Mesh em relação as NoCs *Hermes OO* e *Hermes VC*.

Conforme a Tabela 21, a NoC *FH Mesh* obteve uma vazão média superior a 500 Mbps em relação a NoC *Hermes OO* considerando-se a média de todas as aplicações. Nas aplicações da suíte MCSL esse valor foi superior a 1 Gbps. No entanto, nas demais aplicações não se observoudiferença significativa entre as duas abordagens.

Considerando-se a latência média, novamente a NoC *FH Mesh* obteve melhores resultados nas aplicações do MCSL. Nestas aplicações ela apresentou uma latência média entre 11 e 14 ciclos inferior a NoC *Hermes OO*. Já nas aplicações AV e MPEG4 Decoder a NoC *Hermes OO* obteve uma latência menor.

Porém, diferentemente das NIs das NoCs *Ase Mesh*, *Hermes* e *YeaH*, que repassam um *flit*à NoC a cada ciclo, a NI da NoC *FH Mesh*armazena os dados recebidos do módulo *Wrapper* em um *buffer*. O repasse destes dados à NoC é realizado quando: *(i)* nenhum dado foi recebido em um intervalo de X ciclos (utilizou-se X=5); *(ii)* um novo dado de endereço foi recebido (ou seja,um pacote foi completamente recebido); ou *(iii)*não há mais espaçoem*buffer*. Assim, o desempenho desta NoC é ainda melhor.

O melhor desempenho da NoC *FH Mesh* pode ser justificado pelo fato de sistemas com *bufferização* nas portas de saída apresentarem uma profundidade média de*buffers*menor do que sistemas equivalentes com *bufferização* nas portas de entrada. Isto acontece porque com *buffers* nas portas de entrada, pacotes destinados a portas de saída ociosas são enfileirados atrás de pacotes cujas portas de saída estão ocupadas, não podendo assim ser transmitidos[KAR87]. Uma maneira de superar este problema é através da utilização de canais virtuais, entretanto, como nos mostra aTabela 21 e a Tabela 22, esta abordagem não apresentou uma melhora significativa, pelo menos para esta implementação de canais virtuais.

Cabe salientar que a NoC *FH Mesh* utiliza um esquema de arbitragem diferente das NoCs *Hermes*. Assim, ainda é necessário calcular a relevância desta característica para uma avaliação mais precisa do impacto que a localização dos *buffers* representa.

Tabela 21 – Diferença percentual de vazão média da NoC FH Mesh em relação às NoC Hermes OO e Hermes VC para todas as aplicações e tamanhos de pacotes.

	Hermes OO	**Hermes VC**
AV	-1,10	-8,10
H263e	49,25	32,56
H264dl	45,55	30,99
Mpeg4 Decoder	8,31	0,44
Radio Sys	2,92	-0,52
Sample	46,29	31,60
Sparse	44,87	30,95
VOPD	-0,92	-1,93
Média	**24,40**	**14,50**

Tabela 22 – Diferença percentual de latência média da NoC Hermes OO e Hermes VC em relação à NoC FH Mesh para todas as aplicações e tamanhos de pacotes.

	Hermes OO	**Hermes VC**
AV	-13,09	-23,70
H263e	38,44	38,00
H264dl	36,67	36,42
Mpeg4 Decoder	-8,41	-23,41
Radio Sys	11,93	-0,26
Sample	37,52	37,47
Sparse	36,59	36,54
VOPD	2,80	1,25
Média	**17,81**	**12,79**

7.8 Experimento 4: Latência Máxima e *Jitter*

A Figura 39e a Figura 40apresentam respectivamente a latência máxima e o *jitter* para as aplicações adicionais da plataforma Nocbench. Nestes resultados se nota que a opção por um esquema de arbitragem baseado em prioridades, como o utilizado na NoC *Ase Mesh*, em detrimento de um esquema de arbitragem justo, tal como o esquema *round robin* utilizado nas demais NoCs, pode aumentar consideravelmente a latência máxima e o *jitter* da NoC. Percebe-se isto nas aplicações AV, Radio Sys e VOPD.

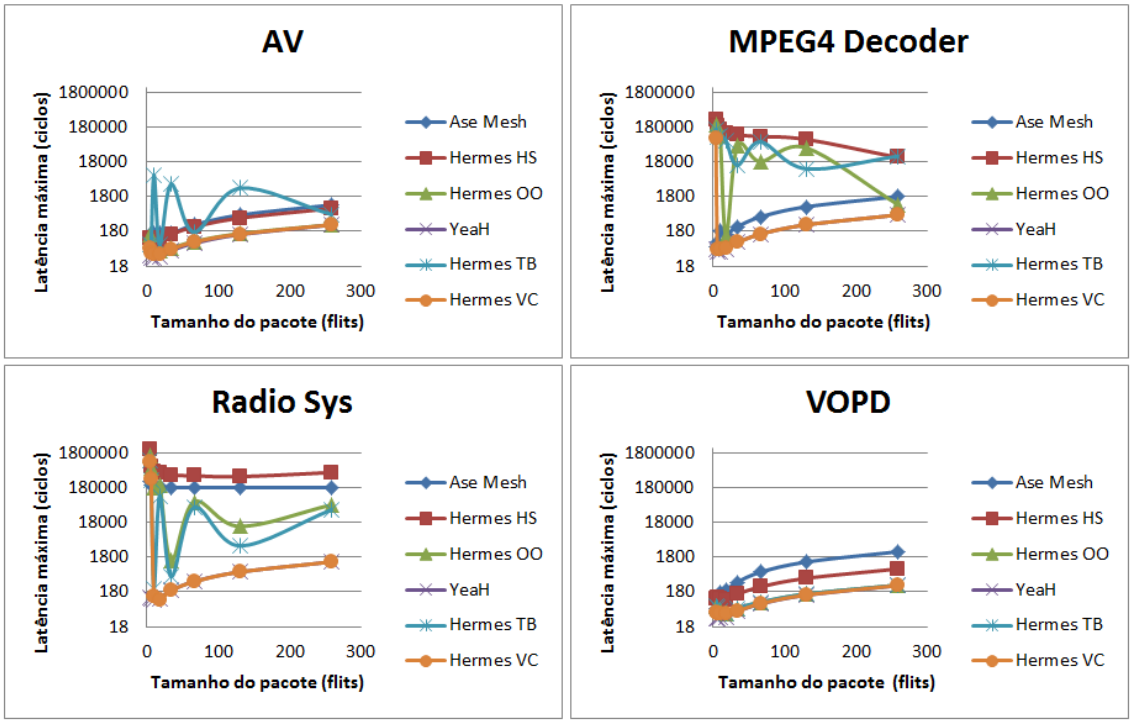

Figura 39 - Latência máxima (em ciclos) para as aplicações adicionais do Nocbench.

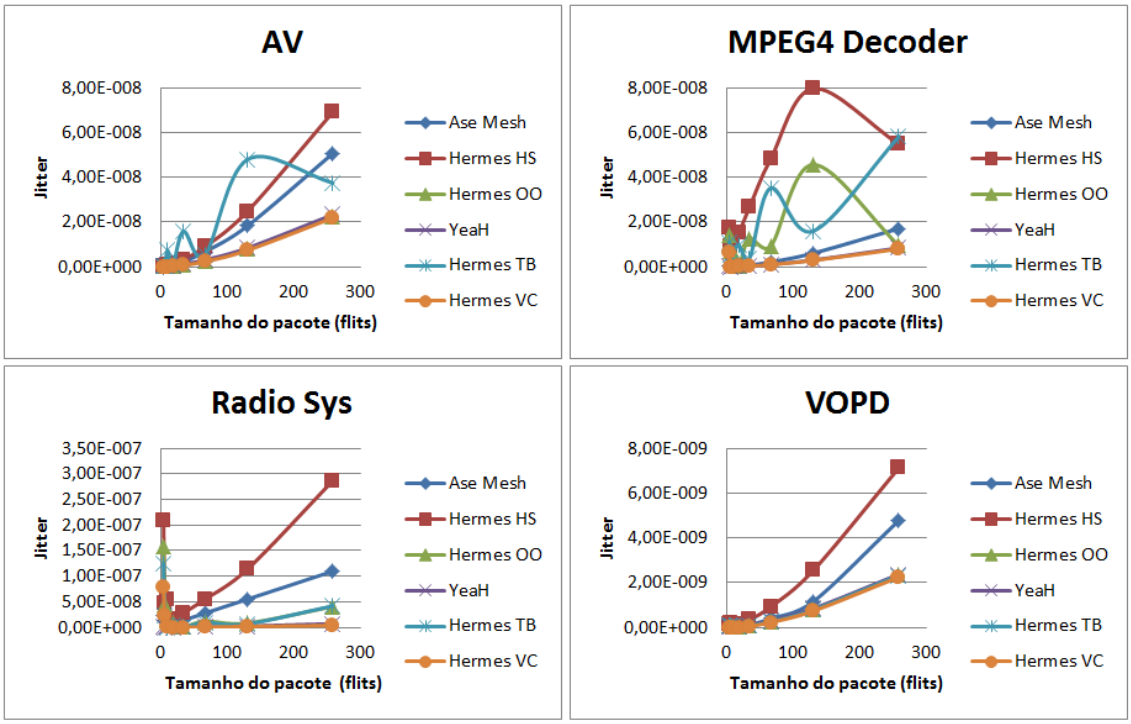

Figura 40 - Jitter para as aplicações adicionais do Nocbench.

Nestas, tanto a latência máxima quanto o *jitter* da NoC *Ase Mesh* são semelhantes à latência máxima e ao *jitter* da NoC *Hermes HS*. Já o péssimo desempenho da NoC *Hermes TB* nas aplicações AV e MPEG4 Decoder é justificado pelo mapeamento de tarefas adotado, que como se sabe é o mesmo utilizado nas NoCs com

topologia malha. Observou-se também que o uso de canais virtuais é uma boa opção para a redução do *jitter* da NoC.

8. Conclusão

Este trabalho apresentou um conjunto de contribuições para fazer evoluir o processo sistemático de avaliação de redes intrachip. Dentre as contribuições específicas mostrou-se como se pode facilitar a integração, avaliação e comparação de diferentes arquiteturas de redes intrachip em uma plataforma já proposta como um padrão para *benchmarking* de NoCs. Sete NoCs foram consideradas nos experimentos descritos no Capítulo 7, sendo cinco destas integradas à plataforma Nocbench.

A integração de uma nova NoC a plataforma mostrou-se bastante intuitiva depois de conduzidas as modificações no ambiente descritas no Capítulo 4. O processo de especificação e seleção do projeto de rede a ser avaliado, que antes exigia que o usuário de Nocbench se envolvesse com detalhes de programação da plataforma, foi automatizado e nenhuma programação é mais necessária para tal ação. As descrições das redes integradas foram otimizadas, o que proporcionou uma redução considerável no tempo de compilação destas e facilitou sua integração. O tamanho das descrições de rede também foi otimizado, e agora é o mesmo para qualquer rede, sem depender diretamente das dimensões destas.

Com os experimentos realizados, observou-se que a arbitragem e o roteamento distribuído são uma boa opção para o aumento da vazão média, especialmente em sistemas onde a carga é distribuída igualitariamente entre os nós da rede, como é o caso do conjunto de *benchmarks* da suíte MCSL, com um custo adicional em área do roteador. O mesmo não pode ser afirmado a respeito de canais virtuais. Nenhuma melhora significativa na vazão média foi observada com a utilização desta abordagem. Contudo, ainda seria necessário investigar a utilização desta solução em tamanhos de *buffer* maiores, o que ainda não foi realizado por limitações de tempo. Cabe salientar, no entanto, que o uso de canais virtuais pode diminuir o *jitter*, levando a resultados semelhantes ao apresentado pelo uso de arbitragem e roteamento distribuído.

Referências

[BIE08] Bienia, C.; Kumar, S.; Singh, J. P.; Li, K. “The PARSEC Benchmark Suite: Characterization and Architectural Implications”. In: 17th International Conference on Parallel Architectures and Compilation Techniques (PACT'08), 2008, pp. 72-81.

[BIN06] Binkert, N. L.; Dreslinski, R. G.; Hsu, L. R.; Lim, K. T.; Saidi, A. G.; Reinhardt, S. K. “The M5 Simulator: Modeling Networked Systems”. IEEE Micro, 26, July-August 2006, pp. 52-60.

[DUA03] Duato, J.; Yalamanchili, S.; Ni, L. “Interconnection Networks – An Engineering Approach”. Elsevier, San Francisco. 2003, 600p.

[GLA92] Glass, C.; Ni, L. “The Turn Model for Adaptive Routing”. In: 19th Annual International Symposium on Computer Architecture (ISCA), 1992, pp. 278-287.

[GRE07] Grecu, C.; Ivanov, A.; Pande, R.; Jantsch, A.; Salminen, E.; Ogras, U.; Marculescu, R. *“*Towards Open Network-on-Chip Benchmarks*”.* In: 1st International Symposium on Networks-on-Chip (NOCS'07), 2007, pp. 205-212.

[GUI08] Guindani, G.; Reinbrecht, C.; Raupp, T.; Calazans, N.; Moraes, F. G. “NoC Power Estimation at the RTL Abstraction Level”. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI'08), 2008, pp. 475-478.

[GUT01] Guthaus, M. R.; Ringenberg, J. S.; Ernst, D.; Austin, T. M.; Mudge, T.; Brown, R. B. “MiBench: A Free, Commercially Representative Embedded *Benchmark* Suite”. In: IEEE International Workshop on Characterization (WWC4'01), 2001, pp. 3-14.

[HES10] Hestness, J.; Grot, B.; Keckler, S. W. “Netrace: Dependency-Driven Trace Based Network-on-Chip Simulation”. In: Third International Workshop on Network on Chip Architectures (NoCArc'10), 2010, pp. 31-36.

[KAR87] Karol, M.J.; Hluchyj, M.G.; Morgan, S.P. "Input Versus Output Queueing on a Space-Division Packet Switch".IEEE Transactions on Communications, COM-35(12), 1987, pp. 1347-1356.

[KTH13] KTH. “KTH Royal Institute of Technology”. Captured in Dec. 2013. Available at: http://www.kth.se/en.

[LEE97] Lee, C; Potkonjak, M; Mangione-Smith, W. “MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems”. In: Thirtieth Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'97), 1997, pp. 330-335.

[LIU11] Liu, W.; Xu, J.; Wu, X.; Ye, Y.; Wang, X.; Zhang, W.; Nikdast, M.; Wang, Z. “A NoC Traffic Suite Based on Real Applications”. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI'11), 2011, pp. 66-71.

[MAC11] Macdonald, K. “Inferring Packet Dependencies to Improve Trace-based Simulation of On-chip Networks”. MSc Dissertation, University of California, Davis, 2011.

[MAC13] Macdonald, K.; Nitta, C.; Farrens, M.; Akella, V. “PDG_GEN: A Methodology for Fast and Accurate Simulation of On-chip Networks”. IEEE Transactions on Computers, 63(3), March 2014, pp. 650-663.

[MAL10] Malave, J. “A Benchmarking Platform for Network-on-Chip (NOC) Multiprocessor System-on-Chips”. MSc Dissertation, Texas A&M University, Texas, December 2010. 82p.

[MAN09] Mandal, S; Gupta, N.; Mandal, A.; Malave, J.; Lee, J.; Mahapatra, R. “NoC-Bench: a Benchmarking Platform for Network on Chip”. In: Workshop on Unique Chips and Systems (UCAS'09), 2009.

[MCS14] MCSL. “MCSL Network-on-Chip Traffic Suite User Manual”. User Manual, V1.5, January 2014, 10 p.

[MEL05a] Mello, A.; Amory, A.; Calazans, N.; Moraes, F. “Atlas: Network-on-Chip Generation and Evaluation Network”. Captured in December 2013. Available at: https://corfu.pucrs.br/ redmine projects/atlas.

[MEL05b] Mello, A.; Tedesco, L.; Calazans, N.; Moraes, F. "Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC". In: 18th Symposium on Integrated Circuits and Systems Desig (SBCCI), 2005, pp.178-183.

[MOR04] Moraes, F. G.; Calazans, N. L. V.; Mello, A. V.; Moller, L. H.; Ost, L. C. “HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip”. Integration the VLSI Journal, 38(1), October 2004, p. 69-93.

[NOC13a] Nocbench. “ Standardization of Benchmarking Methodology for Network-on-Chip”. Captured in December 2013. Available at: http://www.tkt.cs.tut.fi/research/nocbench/.

[NOC13b] Nocbench. “Transaction Generator 2”. Captured in Dec. 2013. Available at: http://www.tkt.cs.tut.fi/research/nocbench/data/sctg_2013_01_23.zip.

[NOC13c] Nocbench. “Traffic Generator”. Captured in Dec. 2013. Available at: http://www.tkt.cs.tut.fi/research/nocbench/data/traffic_generator_20091201.zip.

[OCP13] OCP-IP. “OCP International Partnership”. Captured in Dec. 2013. Available at: http://www.ocpip.org.

[PEK11] Pekkarinen, E.; Lehtonen, L.; Salminen, E.; Hämäläinen, T. D. “A Set of Traffic Models for Network-on-Chip Benchmarking”. In: International Symposium on System on Chip (SoC'11), 2011, pp. 78-81.

[RAH11] Rahmati, D.; Sarbazi-Azad, H.; Hessabi, S.; Kiasary, E. “Power Efficient deterministic and adaptive routing in torus networks-on-chip”. Microprocessors and Microsystems 36(7), October 2011, pp. 571-585.

[SAL07] Salminen, E.; Kulmala, A.; Hämäläinen, T. D. “On Network-on-Chip Comparison”. In: 10th Euromicro Conference on Digital System Design Architectures Methods and Tools (DSD'07), 2007, pp. 503-510.

[SAL08] Salminen, E.; Kangas, T.; Riihimäki, J.; Hämäläinen, T. D. “Requirements for Network-on-Chip Benchmarking”. In: NORCHIP Conference, 2005, pp. 82-85.

[SAL08] Salminen, E.; Kulmala, A.; Hämäläinen, T. D. “Survey of Network-on-chip Proposals”. White Paper, OCP-IP, 2008.

[SAL09] Salminen, E.; Grecu, C.; Hämäläinen, T. D.; Ivanov, A. “Application Modelling and Hardware Description for Network-on-Chip Benchmarking”. IET Computers & Digital Techniques, 3(5), September 2009, pp. 539-550.

[SCH07] Scherer Jr, C. A. “Torus Topology Wormhole Intra-chip Networks-on-Chip: Design, Generation and Evaluation”. MSc Dissertation, PPGCC - FACIN - PUCRS, Porto Alegre, Brazil. March 2007. 101p.

[TAM88] Tamir, Y.; Frazier, G.L. "High-performance multiqueue buffers for VLSI communication switches". In: 15th Annual International Symposium on Computer Architecture, 1988, pp. 343-354.

[TED05] Tedesco, L. “Uma proposta para Geração de Tráfego e Avaliação de Desempenho de NoCs”. MSc Dissertation, PPGCC - FACIN - PUCRS, Porto Alegre, Brazil, November 2005, 108p.

[TOW02] Towles, B.; Dally, W. J. “Worst-case Traffic for Oblivious Routing Functions". Computer Architecture Letters, 1(1), January-December 2002, 4 p.

[WEI02] Weiss, A. R. “Dhrystone Benchmark – History, Analysis, “Scores” and Recommendations”. White Paper, 2002.

[WOO05] Woo, S. C.; Ohara, M.; Torrie, E.; Singh, J. P.; Gupta, A. “The SPLASH-2 Programs – Characterization and Methodological Considerations”. In: 22nd Annual International Symposium on Computer Architecture (ISCA'95), 1995, pp. 24-36.

Apêndice A – Formato de um Arquivo de Traces Estatísticos

Este Apêndice apresenta o formato de um arquivo de *traces* do padrão estatístico (*stp*) do MCSL (Aplicação Sparse para topologia malha e tamanho 2x2).

```
/*********************************************************************
*
* File name: Sparse_mesh_2x2.stp
* Version: 1.6
*
* Package name: MCSL NoC traffic pattern suite
*Author: Zhe Wang (HKUST), Jiang Xu (HKUST), Xiaowen Wu (HKUST),
*Xuan Wang (HKUST), Zhehui Wang (HKUST), Duong Luan (HKUST), Peng
*Yang (HKUST), Wei Zhang (HKUST), Bin Li (Intel), Ravi Lyer (Intel),
*Ramesh Illikkal (Intel)
* Past members: Weichen Liu, Yaoyao Ye
* Website:  http://www.ece.ust.hk/~eexu
*
* The copyright information of this program can be found in the file COPY-
RIGHT.
*
*********************************************************************/
0       0
1       0       4       2       2
2       96      67
3       62      0       1       2       3       4       5       6       7       8
        9       10      11      14      15      17      19      21      24      25
        26      27      28      29      30      31      32      33      34      35
        36      37      38      39      40      42      43      44      45      46
        48      49      50      51      52      56      57      60      62      65
        66      67      68      69      71      72      73      76      81      83
        84      85      95
4       48      0       3       4       5       7       8       9       14      15
        21      33      37      38      39      41      43      46      47      48
        49      51      52      53      54      56      59      61      64      65
```

	66	67	68	74	75	76	83	84	85	86
	87	88	89	90	91	92	93	94	95	
0	(0,0)	0	3520		440					
1	(0,1)	0	2880		360					
2	(1,0)	0	3520		440					
3	(1,1)	0	3520		440					
4	(0,1)	1	2880		360					
5	(0,0)	1	3520		440					
6	(1,0)	1	3520		440					
7	(1,1)	1	3520		440					
8	(0,1)	2	3520		440					
9	(0,0)	2	3520		440					
10	(1,0)	2	3520		440					
11	(1,1)	2	2880		360					
12	(1,0)	15	1600		200					
13	(1,0)	20	2880		360					
14	(0,1)	3	3520		440					
15	(1,1)	3	3520		440					
16	(1,1)	15	2240		280					
17	(0,0)	3	2240		280					
18	(0,0)	16	2880		360					
19	(1,0)	3	3520		440					
20	(1,0)	16	2240		280					
21	(0,0)	4	3520		440					
22	(0,0)	20	1600		200					
23	(0,1)	16	1600		200					
24	(0,1)	4	3520		440					
25	(1,1)	4	2880		360					
26	(1,0)	4	3520		440					
27	(0,0)	5	3520		440					
28	(0,1)	5	2880		360					
29	(1,1)	5	3520		440					
30	(1,0)	5	3520		440					
31	(0,1)	6	2880		360					

32	(0,0)	6	2880	360
33	(1,1)	6	3520	440
34	(1,0)	6	3520	440
35	(0,1)	7	3520	440
36	(0,0)	7	2880	360
37	(1,1)	7	3520	440
38	(1,0)	7	3520	440
39	(0,0)	8	3520	440
40	(0,1)	8	3520	440
41	(1,1)	16	4480	560
42	(1,1)	8	2880	360
43	(1,0)	8	3520	440
44	(0,0)	9	3520	440
45	(0,1)	9	3520	440
46	(1,1)	9	3520	440
47	(0,1)	17	4480	560
48	(1,0)	9	2880	360
49	(0,0)	10	3520	440
50	(0,1)	10	3520	440
51	(1,1)	10	3520	440
52	(1,0)	10	2880	360
53	(1,0)	17	4480	560
54	(0,0)	17	4480	560
55	(1,1)	17	5440	680
56	(0,0)	11	2880	360
57	(0,1)	11	2880	360
58	(0,1)	18	4160	520
59	(1,0)	18	3840	480
60	(1,1)	11	3520	440
61	(0,1)	21	4480	560
62	(1,0)	11	2880	360
63	(0,0)	18	3200	400
64	(1,0)	19	4480	560

65	(0,0)	12	3520	440		
66	(0,1)	12	3520	440		
67	(1,0)	12	3520	440		
68	(1,1)	12	3520	440		
69	(0,0)	13	2880	360		
70	(0,1)	19	4480	560		
71	(0,1)	13	3520	440		
72	(1,0)	13	3520	440		
73	(1,1)	13	3520	440		
74	(0,0)	19	4480	560		
75	(1,1)	20	4800	600		
76	(0,0)	14	3520	440		
77	(0,0)	21	4160	520		
78	(0,0)	22	3840	480		
79	(0,1)	23	4800	600		
80	(0,0)	23	4480	560		
81	(0,1)	14	3520	440		
82	(1,1)	18	4480	560		
83	(1,0)	14	3520	440		
84	(1,1)	14	3520	440		
85	(0,0)	15	3520	440		
86	(0,1)	24	1280	160		
87	(0,1)	20	960	120		
88	(1,0)	22	960	120		
89	(1,0)	21	960	120		
90	(1,0)	24	1280	160		
91	(1,1)	21	960	120		
92	(1,1)	19	960	120		
93	(1,0)	23	960	120		
94	(0,1)	22	960	120		
95	(0,1)	15	2560	320		
0	2	12	0xf400	0x400	204.80	25.60
	0.007386					

1	12	13	0x15000	0x400	204.80	25.60	0.016250
2	10	16	0x10000	0x800	204.80	25.60	0.007386
3	17	18	0x0	0x400	204.80	25.60	0.011607
4	19	20	0x11400	0x800	204.80	25.60	0.007386
5	16	22	0x19000	0x400	204.80	25.60	0.011607
6	18	22	0x3000	0x400	204.80	25.60	0.009028
7	11	23	0x16000	0x400	204.80	25.60	0.009028
8	26	41	0x12000	0x800	204.80	25.60	0.007386
9	45	47	0xa800	0x800	204.80	25.60	0.007386
10	44	53	0x1c00	0x800	204.80	25.60	0.007386
11	40	54	0xa000	0x800	204.80	25.60	0.007386
12	29	55	0x17000	0x800	204.80	25.60	0.007386
13	35	55	0x9800	0x800	204.80	25.60	0.007386
14	27	58	0x400	0x800	204.80	25.60	0.007386
15	31	58	0x8c00	0x800	204.80	25.60	0.009028
16	32	59	0xc00	0x800	204.80	25.60	0.009028
17	55	61	0x19c00	0x800	204.80	25.60	0.004779
18	30	63	0x12800	0x800	204.80	25.60	0.007386
19	34	64	0x13800	0x800	204.80	25.60	0.007386
20	62	70	0x14000	0x800	204.80	25.60	0.009028

21	72	74	0x14800	0x800	204.80	25.60
	0.007386					
22	60	75	0x17c00	0x800	204.80	25.60
	0.007386					
23	70	75	0xe400	0x800	204.80	25.60
	0.005804					
24	20	77	0x15400	0x800	204.80	25.60
	0.011607					
25	69	77	0x2400	0x800	204.80	25.60
	0.009028					
26	71	77	0xcc00	0x800	204.80	25.60
	0.007386					
27	16	78	0x19400	0x800	204.80	25.60
	0.011607					
28	22	78	0x4c00	0x400	204.80	25.60
	0.016250					
29	23	78	0xdc00	0x400	204.80	25.60
	0.016250					
30	57	78	0xb800	0xc00	204.80	25.60
	0.009028					
31	63	78	0x3800	0x800	204.80	25.60
	0.008125					
32	77	78	0x5000	0x800	204.80	25.60
	0.006250					
33	36	79	0x1400	0x800	204.80	25.60
	0.009028					
34	77	79	0x5800	0x800	204.80	25.60
	0.006250					
35	78	80	0x6800	0x800	204.80	25.60
	0.006771					
36	50	82	0xb000	0x800	204.80	25.60
	0.007386					
37	24	86	0x8400	0x400	204.80	25.60
	0.007386					
38	25	86	0x16c00	0x400	204.80	25.60
	0.009028					
39	79	86	0xec00	0x800	204.80	25.60
	0.005417					

40	80	86	0x7800	0x800	204.80	25.60	0.005804
41	81	86	0xd800	0x400	204.80	25.60	0.007386
42	19	87	0x11c00	0x400	204.80	25.60	0.007386
43	69	87	0x2c00	0x400	204.80	25.60	0.009028
44	71	87	0xd400	0x400	204.80	25.60	0.007386
45	18	88	0x3400	0x400	204.80	25.60	0.009028
46	63	88	0x4000	0x400	204.80	25.60	0.008125
47	77	88	0x6000	0x400	204.80	25.60	0.006250
48	13	89	0x15c00	0x400	204.80	25.60	0.009028
49	58	89	0xe000	0x400	204.80	25.60	0.006250
50	73	89	0x18400	0x800	204.80	25.60	0.007386
51	10	90	0x10800	0xc00	204.80	25.60	0.007386
52	11	90	0x16400	0x800	204.80	25.60	0.009028
53	57	90	0xc400	0x800	204.80	25.60	0.009028
54	63	90	0x4400	0x800	204.80	25.60	0.008125
55	78	90	0x7000	0x400	204.80	25.60	0.006771
56	6	91	0xfc00	0x400	204.80	25.60	0.007386
57	42	91	0x17800	0x400	204.80	25.60	0.009028
58	78	91	0x7400	0x400	204.80	25.60	0.006771
59	2	92	0xf800	0x400	204.80	25.60	0.007386

60	31	92	0x9400	0x400	204.80	25.60
	0.009028					
61	73	92	0x18c00	0x400	204.80	25.60
	0.007386					
62	30	93	0x13000	0x800	204.80	25.60
	0.007386					
63	77	93	0x6400	0x400	204.80	25.60
	0.006250					
64	1	94	0x8000	0x400	204.80	25.60
	0.009028					
65	28	94	0x8800	0x400	204.80	25.60
	0.009028					
66	82	94	0x1a400	0x400	204.80	25.60
	0.005804					

Apêndice B – Formato de um Arquivo de Traces Obtidos por Simulação

Este Apêndice apresenta o formato de um arquivo de *traces* do padrão obtido por simulação (r*tp*) do MCSL (Aplicação Sparse para topologia malha e tamanho 2x2para 20 execuções).

```
/*************************************************************************
*******
*
* File name: Sparse_mesh_2x2.rtp
* Version: 1.6
*
* Package name: MCSL NoC traffic pattern suite
*Author: Zhe Wang (HKUST), Jiang Xu (HKUST), Xiaowen Wu (HKUST),
*Xuan Wang (HKUST),Zhehui Wang (HKUST), Duong Luan (HKUST), Peng
*Yang (HKUST), Wei Zhang (HKUST), Bin Li (Intel), Ravi Lyer (Intel),
*Ramesh Illikkal (Intel)
* Past members: Weichen Liu, Yaoyao Ye
* Website: http://www.ece.ust.hk/~eexu
*
* The copyright information of this program can be found in the file COPY-
RIGHT.
*
*************************************************************************
*****/
0  1
1  0   4      2      2
2  96  67     20
3  62  0      1      2      3      4      5      6      7      8      9
       10     11     14     15     17     19     21     24     25     26
       27     28     29     30     31     32     33     34     35     36
       37     38     39     40     42     43     44     45     46     48
       49     50     51     52     56     57     60     62     65     66
       67     68     69     71     72     73     76     81     83     84
       85     95
```

4	48	0	3	4	5	7	8	9	14	15	21
		33	37	38	39	41	43	46	47	48	49
		51	52	53	54	56	59	61	64	65	66
		67	68	74	75	76	83	84	85	86	87
		88	89	90	91	92	93	94	95		
0		(0,0)	0	1	3	6	10	14	18	22	26
		30	34	38	42	46	50	54	58	62	67
		72	3068	4178	3610	3349	3549	3606	4353	3731	4280
		3264	2997	3368	4034	3680	3299	3325	3351	3786	3544
		3442									
1		(0,1)	0	23	48	73	98	123	148	173	198
		223	248	273	298	323	348	373	398	423	448
		473	3113	3497	2932	2324	3211	3042	3600	3064	2779
		2510	3059	3394	2832	3407	2837	3183	2219	2852	2880
		3133									
2		(1,0)	0	22	46	74	100	122	146	171	200
		225	248	275	296	325	350	375	400	423	446
		475	4043	3368	2976	3977	3472	3116	2865	3986	3186
		3516	4158	2972	2633	4363	3228	3835	3773	3511	3247
		3135									
3		(1,1)	0	1	3	6	9	12	15	18	22
		26	29	32	35	38	41	44	47	50	53
		56	2677	3564	2896	3664	3385	4140	2928	4400	4400
		3653	3651	3382	3988	3033	3902	2996	4058	4297	4400
		4029									
4		(0,1)	1	24	50	75	100	125	150	175	200
		225	250	275	300	325	350	375	400	425	450
		475	3351	3284	3135	3161	2941	2550	2820	3111	3372
		2737	1894	2632	3177	3213	2630	2989	2591	2499	2560
		2767									
5		(0,0)	2	4	7	11	15	19	23	27	31
		35	39	43	47	51	55	59	63	68	73
		77	2731	4305	4016	2950	3116	2559	3218	3247	3508
		3787	3379	3831	4242	3694	3181	3961	2803	3513	3751
		3577									
6		(1,0)	1	26	48	76	101	126	151	176	201
		226	251	276	301	326	351	376	401	426	451
		476	3949	3903	3124	3680	3421	3169	2899	4268	3910
		3457	2989	3550	3389	3280	4124	3510	3328	4229	3224
		4400									
7		(1,1)	2	4	7	10	13	16	19	23	27
		30	33	36	39	42	45	48	51	54	57
		59	2998	3541	2462	2543	3338	3420	3167	3271	3112

	2642	2757	3421	3064	3787	2940	3905	3457	2960	3417
	4151									
8	(0,1)	2	25	52	77	102	127	152	177	202
	227	252	277	302	327	352	377	402	427	452
	477	3723	3763	3815	3215	3942	2992	2967	4290	3411
	3546	3605	3818	3630	2425	3178	4059	3165	3571	3197
	3114									
9	(0,0)	5	8	12	16	20	24	28	32	36
	40	44	48	52	56	60	64	69	74	78
	81	3213	3170	3058	2974	3715	3418	3842	2636	3226
	3216	3810	4346	4400	3447	3673	3692	2853	4066	3333
	3963									
10	(1,0)	2	27	50	77	102	127	152	177	202
	227	252	277	302	327	352	377	402	427	452
	477	2942	3959	3572	3727	2431	3106	3677	3527	3763
	3906	3725	3604	3742	3579	3348	3260	3400	3443	3481
	4069									
11	(1,1)	5	20	76	96	115	137	157	177	195
	216	236	256	277	296	316	336	356	376	397
	416	2647	2543	2238	3024	2556	2503	2791	2590	3600
	2687	2876	2405	2068	3278	2951	3254	3203	2720	2888
	2837									
12	(1,0)	15	40	65	90	115	140	165	190	215
	240	265	290	315	340	365	390	415	440	465
	490	1854	1630	1557	1506	1250	1597	1686	1503	1502
	1832	2000	1293	1599	1419	1852	1549	1941	1383	1711
	1613									
13	(1,0)	20	45	70	95	120	145	170	195	220
	245	270	295	320	345	370	395	420	445	470
	495	2817	3010	3467	3362	2612	2940	2323	2818	2418
	2886	2896	2953	3272	2654	2716	3227	3531	2963	2643
	3019									
14	(0,1)	3	26	53	78	103	128	153	178	203
	228	253	278	303	328	353	378	403	428	453
	478	3267	4400	3774	2954	3438	3371	3823	3228	2614
	4236	3087	3650	3532	3158	3602	3027	3921	3201	3148
	2966									
15	(1,1)	8	24	78	98	117	139	159	179	197
	218	238	258	279	298	318	338	358	378	399
	418	2659	3185	3783	4059	3488	2641	2966	4158	3583
	3834	3499	3751	3882	3191	3736	3543	3823	3677	3286
	3774									

16	(1,1)	46	73	93	113	133	153	173	193	213
	233	253	273	293	313	333	353	373	393	413
	433	2430	2533	2085	2553	2114	2441	2163	2448	2152
	2197	2292	2170	1930	2262	2365	2576	2173	2176	2187
	1979									
17	(0,0)	9	65	95	116	137	158	179	200	221
	242	263	284	305	326	347	368	389	410	431
	452	2334	2240	2475	2078	2124	2388	1929	2212	2066
	2372	2800	2206	2124	2490	2073	2517	2407	2147	2432
	2617									
18	(0,0)	61	94	115	136	157	178	199	220	241
	262	283	304	325	346	367	388	409	430	451
	472	3600	3357	2961	2740	2868	3251	3294	3319	2619
	3600	2688	3218	3129	3158	3358	3431	3159	3469	3600
	3419									
19	(1,0)	3	28	52	78	103	128	153	178	203
	228	253	278	303	328	353	378	403	428	453
	478	3075	3727	3656	3803	3788	3079	4203	3514	4128
	3773	2795	3283	3777	4400	4008	2786	2785	3888	3095
	2772									
20	(1,0)	16	41	66	91	116	141	166	191	216
	241	266	291	316	341	366	391	416	441	466
	491	2518	2575	1327	2226	2176	2234	1861	2103	2279
	2191	2317	2211	2226	2181	2616	2728	2589	2381	2489
	1993									
21	(0,0)	13	70	97	117	138	159	180	201	222
	243	264	285	306	327	348	369	390	411	432
	453	3439	3351	3452	3524	3247	3726	3665	3772	3954
	3533	2737	4400	3583	4245	3377	2963	3478	3566	3783
	3962									
22	(0,0)	80	102	124	147	170	192	213	234	255
	276	297	318	339	360	381	402	423	444	465
	476	1598	1225	1716	1623	1876	1590	1847	1741	1501
	1242	1484	1448	1638	1503	1699	1691	1437	1990	1624
	1417									
23	(0,1)	16	41	66	91	116	141	166	191	216
	241	266	291	316	341	366	391	416	441	466
	491	1751	1796	1588	1253	1701	1736	1548	1568	1811
	1568	1657	1588	1793	1393	1407	1823	1441	1652	1460
	1554									
24	(0,1)	4	29	54	79	104	129	154	179	204
	229	254	279	304	329	354	379	404	429	454
	479	3804	2854	3077	2876	3888	4251	4104	3257	3417

	3493	2446	3480	4248	3438	2940	4060	3179	3342	4258
	3897									
25	(1,1)	11	62	82	102	122	142	162	182	202
	222	242	262	282	302	322	342	362	382	402
	422	2816	2572	2383	2536	3041	3597	2954	3597	2869
	2832	3134	2404	3327	3137	2842	2668	2566	2986	3079
	2383									
26	(1,0)	4	29	54	79	104	129	154	179	204
	229	254	279	304	329	354	379	404	429	454
	479	3447	3223	3858	4090	3294	4360	3004	3215	3364
	3605	3549	3542	3478	3580	3130	3909	3507	3496	2971
	3707									
27	(0,0)	17	75	99	119	139	160	181	202	223
	244	265	286	307	328	349	370	391	412	433
	454	3673	3470	2468	3998	3309	3325	3697	3507	3822
	3356	3090	3546	3488	3334	3008	3456	3547	3535	2585
	2927									
28	(0,1)	5	30	55	80	105	130	155	180	205
	230	255	280	305	330	355	380	405	430	455
	480	2816	3101	2586	3002	3239	2751	3148	3012	2726
	3116	2705	2714	2555	2522	3241	2595	2647	2671	2809
	2807									
29	(1,1)	14	63	83	103	123	143	163	183	203
	223	243	263	283	303	323	343	363	383	403
	423	2502	3134	2901	4027	3934	4261	3425	3518	3581
	3417	3466	3881	3487	3362	3955	3866	3173	3265	3648
	4082									
30	(1,0)	5	30	55	80	105	130	155	180	205
	230	255	280	305	330	355	380	405	430	455
	480	3515	3235	3294	3358	3883	3167	3216	3023	3817
	3502	3985	3285	3257	4171	3758	3281	4400	3544	3494
	3558									
31	(0,1)	6	31	56	81	106	131	156	181	206
	231	256	281	306	331	356	381	406	431	456
	481	2949	3569	2999	3152	2965	3204	2610	2142	2925
	2218	3058	3114	3014	2281	3314	2450	2890	3066	2974
	2763									
32	(0,0)	21	79	101	121	140	161	182	203	224
	245	266	287	308	329	350	371	392	413	434
	455	2890	3340	2458	3417	2947	3410	2352	2233	3039
	3414	3217	3266	3128	2797	2787	2605	3065	2903	3213
	3173									

33	(1,1)	17	64	84	104	124	144	164	184	204
	224	244	264	284	304	324	344	364	384	404
	424	3894	3570	3450	3363	3367	3997	3698	4139	3014
	3900	3838	3568	3969	3341	3286	3795	3232	4372	3878
	3938									
34	(1,0)	6	31	56	81	106	131	156	181	206
	231	256	281	306	331	356	381	406	431	456
	481	3730	3465	3584	3406	3964	4146	3677	3983	3194
	3587	3450	2660	3107	3598	3485	3500	3968	3126	3386
	3675									
35	(0,1)	7	32	57	82	107	132	157	182	207
	232	257	282	307	332	357	382	407	432	457
	482	3290	2964	3812	3466	3454	4215	3422	3026	3339
	3759	3857	4192	4161	3697	3324	3358	3447	3278	3078
	4322									
36	(0,0)	25	82	103	123	142	162	183	204	225
	246	267	288	309	330	351	372	393	414	435
	456	2497	2381	2702	2737	2643	2983	2833	3365	3416
	3431	3530	2461	2827	3600	2259	2633	3164	2700	2896
	2900									
37	(1,1)	21	65	85	105	125	145	165	185	205
	225	245	265	285	305	325	345	365	385	405
	425	3696	2664	4149	3277	3449	3629	3321	3969	3105
	3847	3620	3369	3980	2949	2892	4400	3888	3614	3812
	3497									
38	(1,0)	7	32	57	82	107	132	157	182	207
	232	257	282	307	332	357	382	407	432	457
	482	3328	3593	2697	3122	3371	4046	2638	3715	3405
	3029	2469	3636	4136	3151	2956	2700	4084	3777	2737
	3413									
39	(0,0)	29	84	105	125	144	163	184	205	226
	247	268	289	310	331	352	373	394	415	436
	457	4249	4047	3266	3846	4066	3242	3731	4060	4050
	3548	3272	3716	3831	3636	3581	2760	3977	4400	2743
	3251									
40	(0,1)	8	33	58	83	108	133	158	183	208
	233	258	283	308	333	358	383	408	433	458
	483	4089	4285	3601	3399	3491	4120	3004	3162	4286
	4122	3836	3332	3443	4259	3640	3418	2869	2763	3375
	3594									
41	(1,1)	49	74	94	114	134	154	174	194	214
	234	254	274	294	314	334	354	374	394	414
	434	4587	4200	5594	5600	4659	3790	4415	4875	5425

	4763	5233	4362	3997	4290	5041	5134	3871	4237	5346
	5124									
42	(1,1)	25	66	86	106	126	146	166	186	206
	226	246	266	286	306	326	346	366	386	406
	426	3371	2605	2403	2487	2346	2968	2816	2693	3165
	2830	3267	2493	2548	2485	3228	2294	3031	2436	2838
	2575									
43	(1,0)	8	33	58	83	108	133	158	183	208
	233	258	283	308	333	358	383	408	433	458
	483	3717	3857	2686	3703	2952	3830	4120	2609	3479
	3998	3957	3877	2812	3590	3745	3003	3606	3229	4400
	4095									
44	(0,0)	33	86	107	127	146	165	185	206	227
	248	269	290	311	332	353	374	395	416	437
	458	3537	3419	3294	3959	4173	3455	3465	3755	3700
	3542	3083	3567	3170	3830	3833	3713	3673	3701	3976
	3603									
45	(0,1)	9	34	59	84	109	134	159	184	209
	234	259	284	309	334	359	384	409	434	459
	484	2900	3379	3297	3805	2859	3863	3090	3388	4144
	3672	3642	3799	2931	3355	3586	3459	3868	3525	3011
	3623									
46	(1,1)	28	67	87	107	127	147	167	187	207
	227	247	267	287	307	327	347	367	387	407
	427	3699	2885	3947	3664	2917	3568	2838	3791	3709
	3278	3880	3308	3683	3191	3972	3114	2991	2718	2943
	3235									
47	(0,1)	17	42	67	92	117	142	167	192	217
	242	267	292	317	342	367	392	417	442	467
	492	4547	4669	4638	4475	3923	4587	5288	4046	3321
	3722	3995	5360	4219	3756	4914	4516	4686	4208	4733
	4621									
48	(1,0)	9	34	59	84	109	134	159	184	209
	234	259	284	309	334	359	384	409	434	459
	484	3323	2791	2909	2742	2821	3014	2872	2859	3044
	2768	2639	2376	3072	2520	2435	2632	3250	2714	2447
	2585									
49	(0,0)	37	88	109	129	148	167	186	207	228
	249	270	291	312	333	354	375	396	417	438
	459	3192	3265	3642	3820	3205	2902	3711	3292	3721
	3342	3608	3921	2951	3537	3424	2600	3778	2855	2965
	3902									

50	(0,1)	10	35	60	85	110	135	160	185	210
	235	260	285	310	335	360	385	410	435	460
	485	2931	3960	4172	3461	4123	3719	2778	3773	3808
	3116	3947	4088	4400	3254	3971	3245	3445	3679	3439
	3490									
51	(1,1)	31	68	88	108	128	148	168	188	208
	228	248	268	288	308	328	348	368	388	408
	428	4400	3256	2505	3800	3754	3160	2559	3589	3273
	3924	3273	3336	3102	3375	3431	3180	3723	2858	3458
	3686									
52	(1,0)	10	35	60	85	110	135	160	185	210
	235	260	285	310	335	360	385	410	435	460
	485	3283	2994	2686	2396	2527	3319	2465	3346	2862
	2800	3109	2998	2687	2773	3008	3600	2543	2853	2976
	3259									
53	(1,0)	17	42	67	92	117	142	167	192	217
	242	267	292	317	342	367	392	417	442	467
	492	4157	4099	3811	4920	4684	4836	4668	4511	2954
	5051	4156	3922	3423	4838	4702	4639	4758	4692	3673
	4466									
54	(0,0)	66	96	118	141	164	188	210	230	250
	273	292	314	334	357	378	398	418	440	461
	473	4113	5068	4561	4951	5482	5036	4079	5205	4611
	5387	3957	3712	4962	5413	4530	3637	4586	4298	5084
	5095									
55	(1,1)	52	75	95	116	135	155	175	196	215
	235	255	275	295	315	335	355	375	395	415
	435	5770	4518	4308	6800	5600	6048	4463	3828	5126
	4849	5190	3713	6028	5246	5794	5330	4765	3851	4836
	5901									
56	(0,0)	41	89	110	131	150	169	187	208	229
	251	271	293	313	335	355	376	397	419	439
	460	3590	2808	3120	3350	2805	2718	2183	1854	3046
	3405	2706	2834	2459	2609	2678	2526	2964	3083	2744
	3045									
57	(0,1)	11	36	61	86	111	136	161	186	211
	236	261	286	311	336	361	386	411	436	461
	486	2820	2931	3412	3409	3079	2710	3158	2682	2376
	2547	2374	3131	2739	3460	2542	2875	2311	3057	2563
	2989									
58	(0,1)	18	43	68	93	118	143	168	193	218
	243	268	293	318	343	368	393	418	443	468
	493	4150	3923	3725	3515	4708	4192	4452	4052	3645

	5197	3587	4383	5200	3659	3793	3660	4712	4706	5200
	3853									
59	(1,0)	18	43	68	93	118	143	168	193	218
	243	268	293	318	343	368	393	418	443	468
	493	3485	3032	3430	4204	3676	3774	3485	3194	4292
	3072	4800	4486	2888	4353	4060	3954	3444	3303	3779
	3684									
60	(1,1)	34	69	89	109	129	149	169	189	209
	229	249	269	289	309	329	349	369	389	409
	429	2847	3413	4230	3310	3778	2881	3378	4040	3472
	3917	3402	3621	4171	3166	3405	4400	4226	2956	3809
	3521									
61	(0,1)	21	46	71	96	121	146	171	196	221
	246	271	296	321	346	371	396	421	446	471
	496	4522	3481	5600	4804	3653	4596	4714	4452	4341
	3899	4363	3459	5494	4728	3683	4768	5103	4650	4599
	4486									
62	(1,0)	11	36	61	86	111	136	161	186	211
	236	261	286	311	336	361	386	411	436	461
	486	3357	3129	2134	3600	2615	2972	2616	3088	2825
	2781	3395	3021	3118	3177	2906	2708	3185	3207	3161
	2424									
63	(0,0)	71	98	120	143	166	190	211	232	252
	274	294	316	336	358	379	400	420	442	463
	474	3420	3346	3739	3015	2774	3318	3343	2691	2718
	3198	3132	3173	3359	3207	2404	3719	3372	3438	3311
	2948									
64	(1,0)	19	44	69	94	119	144	169	194	219
	244	269	294	319	344	369	394	419	444	469
	494	3798	4681	4199	4372	4662	4510	5527	5600	3416
	4312	4647	3179	4003	4822	3986	4220	4232	3540	4163
	4830									
65	(0,0)	45	90	111	132	152	171	189	209	231
	253	272	295	315	337	356	377	399	421	441
	462	3886	3427	3522	3643	4400	3869	3320	3258	3493
	3365	3729	3340	3750	4400	3543	2689	4400	3196	3265
	3454									
66	(0,1)	12	37	62	87	112	137	162	187	212
	237	262	287	312	337	362	387	412	437	462
	487	3782	3700	4160	2843	3603	4400	4281	3644	3360
	3439	4164	3447	3258	3811	3173	3608	3785	3891	3284
	4055									

67	(1,0)	12	37	62	87	112	137	162	187	212
	237	262	287	312	337	362	387	412	437	462
	487	2722	2528	3382	3572	3385	3465	3757	3351	3133
	3296	2852	2966	3543	3368	2590	2861	3795	3774	3170
	4400									
68	(1,1)	37	70	90	110	130	150	170	190	210
	230	250	270	290	310	330	350	370	390	410
	430	2740	3883	3917	3611	3605	3469	3571	3983	4400
	3558	3544	3996	2578	3455	3572	3690	3909	3405	3331
	3618									
69	(0,0)	49	91	112	133	154	175	196	217	238
	259	280	301	322	343	364	385	406	427	448
	469	2885	2699	3276	2777	3095	2022	2610	2755	3472
	2699	2025	2968	2439	2692	2702	2514	2641	2659	2484
	2466									
70	(0,1)	19	44	69	94	119	144	169	194	219
	244	269	294	319	344	369	394	419	444	469
	494	4384	3793	3955	5069	5377	4407	4746	5288	4226
	5185	4725	4923	4535	5008	4542	4087	4647	4836	4548
	4012									
71	(0,1)	13	38	63	88	113	138	163	188	213
	238	263	288	313	338	363	388	413	438	463
	488	3511	3616	3867	2764	3751	3283	3402	3661	3503
	4214	3801	3695	2896	3490	4070	3195	3052	2551	3451
	3505									
72	(1,0)	13	38	63	88	113	138	163	188	213
	238	263	288	313	338	363	388	413	438	463
	488	3711	3783	3304	3391	4291	3705	3866	3874	4103
	3581	3598	3545	3840	3206	3455	4329	2786	4038	3230
	3364									
73	(1,1)	40	71	91	111	131	151	171	191	211
	231	251	271	291	311	331	351	371	391	411
	431	4205	2757	4031	3634	4244	2601	3015	4400	3360
	3435	2938	2382	2701	3680	3020	3007	3613	3189	3521
	3249									
74	(0,0)	76	100	122	145	168	191	212	233	254
	275	296	317	338	359	380	401	422	443	464
	475	4923	4844	3845	4400	4848	4224	4279	5017	4637
	4348	4899	4591	4577	3955	4152	4730	4428	4920	4368
	4764									
75	(1,1)	60	80	100	120	140	160	180	200	220
	240	260	280	300	320	340	360	380	400	420
	438	5798	4911	5450	4715	4686	4457	5339	4013	5689

	4890	5057	4123	4555	6000	5045	5027	5118	4237	4541
	5421									
76	(0,0)	53	92	113	134	155	176	197	218	239
	260	281	302	323	344	365	386	407	428	449
	470	4305	4108	3066	3187	3784	3301	3051	3380	3899
	3168	3440	4061	3967	4144	3765	4025	4100	2725	3085
	3267									
77	(0,0)	83	104	126	149	172	193	214	235	256
	277	298	319	340	361	382	403	424	445	466
	477	4142	3929	3976	3306	4101	4320	4279	4211	3488
	4898	3665	4562	3449	4074	3676	4875	4580	4374	4401
	4078									
78	(0,0)	85	106	128	151	173	194	215	236	257
	278	299	320	341	362	383	404	425	446	467
	478	3753	3519	3996	4298	3879	3385	3136	2892	4667
	3730	3427	4110	3798	3291	3956	3535	3581	3953	3269
	4514									
79	(0,1)	27	49	74	99	124	149	174	199	224
	249	274	299	324	349	374	399	424	449	474
	498	4716	5059	5040	4955	4821	4913	5179	5228	5370
	3474	3821	4194	5000	5784	5628	4171	4881	4355	4372
	4731									
80	(0,0)	87	108	130	153	174	195	216	237	258
	279	300	321	342	363	384	405	426	447	468
	479	4145	4510	3950	4276	4777	4336	5282	4212	4485
	4484	3325	3771	3605	4601	3991	4697	3552	5314	3731
	4607									
81	(0,1)	14	39	64	89	114	139	164	189	214
	239	264	289	314	339	364	389	414	439	464
	489	4081	3682	3458	2709	3833	3689	3711	3410	3385
	3471	3441	3673	3741	3086	3201	3737	3821	3902	3738
	3438									
82	(1,1)	55	77	97	118	136	156	176	198	217
	237	257	276	297	317	337	357	377	396	417
	436	4495	4329	4766	4224	5388	4846	3830	5413	5489
	3176	4735	4835	4835	4227	4752	2846	4737	4134	4611
	4266									
83	(1,0)	14	39	64	89	114	139	164	189	214
	239	264	289	314	339	364	389	414	439	464
	489	3176	2709	4400	4164	3625	3919	3145	3689	3418
	3706	3506	3014	2804	3209	3716	4400	3733	4069	3179
	3995									

84	(1,1)	43	72	92	112	132	152	172	192	212
	232	252	272	292	312	332	352	372	392	412
	432	3214	3292	3982	4267	3107	3503	3461	4035	3544
	4139	2865	4028	4027	3535	2993	2851	3419	2838	3710
	3600									
85	(0,0)	57	93	114	135	156	177	198	219	240
	261	282	303	324	345	366	387	408	429	450
	471	3411	3499	4294	4369	3966	3688	3303	2689	3948
	3398	3146	3017	3697	3659	3614	3495	3595	3581	3786
	2736									
86	(0,1)	28	51	76	101	126	151	176	201	226
	251	276	301	326	351	376	401	426	451	476
	499	1266	1134	1096	1290	1209	1093	1040	1279	1358
	1097	1125	1407	1385	1234	1405	1494	1158	1099	1197
	1395									
87	(0,1)	20	45	70	95	120	145	170	195	220
	245	270	295	320	345	370	395	420	445	470
	495	899	1072	993	914	960	1042	925	869	1049
	1010	865	1126	1082	787	971	956	1020	1200	863
	1066									
88	(1,0)	23	49	72	97	123	148	173	197	222
	247	272	298	322	347	372	397	422	448	472
	497	961	992	1056	893	951	997	859	1163	777
	957	1068	837	1120	958	453	1026	918	1116	915
	924									
89	(1,0)	21	47	71	96	121	147	172	196	221
	246	271	297	321	346	371	396	421	447	471
	496	1001	894	795	831	897	939	1053	1093	1121
	1065	1083	890	912	856	880	941	911	1006	1021
	978									
90	(1,0)	25	53	75	99	125	150	175	199	224
	250	274	300	324	349	374	399	425	450	474
	499	1193	1517	1101	1376	1505	1273	1192	1330	1385
	1080	1560	972	1253	1233	832	1182	1009	1452	1287
	1335									
91	(1,1)	61	81	101	121	141	161	181	201	221
	241	261	281	301	321	341	361	381	401	421
	439	859	1096	968	1031	1120	811	923	1100	1026
	931	1141	1064	1020	960	893	1114	909	1038	975
	754									
92	(1,1)	58	79	99	119	138	158	178	199	219
	239	259	278	299	319	339	359	379	398	419
	437	1017	1069	794	898	868	940	1200	886	897

	774	938	988	1045	1200	1005	867	1021	1176	943
	775									
93	(1,0)	24	51	73	98	124	149	174	198	223
	249	273	299	323	348	373	398	424	449	473
	498	720	1092	937	1035	1040	998	905	831	945
	954	752	1037	939	1090	810	964	764	861	936
	897									
94	(0,1)	22	47	72	97	122	147	172	197	222
	247	272	297	322	347	372	397	422	447	472
	497	936	1037	1159	886	1025	1066	789	1077	1121
	972	922	827	684	942	966	996	1059	1045	1029
	769									
95	(0,1)	15	40	65	90	115	140	165	190	215
	240	265	290	315	340	365	390	415	440	465
	490	2789	2704	2258	2469	2790	2376	2795	2282	2548
	2678	2583	2584	2690	2722	2805	2455	2119	2853	2612
	2148									
0	2	12	0xe680	0xe680	0xe680	0xe680				
	0xe680		0xe680	0xe680	0xe680	0xe680				
	0xe680		0xe680	0xe680	0xe680	0xe680				
	0xe680		0xe680	0xe680	0xe680	0xe680				
	0xe680		153.00	248.00	183.50	178.00				
	210.00		169.75	182.50	193.75	204.00				
	216.50		221.50	181.75	246.75	215.75				
	233.50		186.75	256.00	223.50	218.25				
	187.75									
1	12	13	0x13c8c	0x13c8c	0x13c8c	0x13c8c				
	0x13c8c		0x13c8c	0x13c8c	0x13c8c	0x13c8c				
	0x13c8c		0x13c8c	0x13c8c	0x13c8c	0x13c8c				
	0x13c8c		0x13c8c	0x13c8c	0x13c8c	0x13c8c				
	0x13c8c		220.25	147.75	256.00	212.25				
	225.25		146.75	179.00	211.75	203.50				
	254.75		247.25	187.25	169.75	211.75				
	179.25		231.75	230.25	228.00	183.25				
	178.25									
2	10	16	0xf1c0	0xf5ac	0xf1c0	0xf5ac				
	0xf1c0		0xf5ac	0xf1c0	0xf5ac	0xf1c0				
	0xf5ac		0xf1c0	0xf5ac	0xf1c0	0xf5ac				
	0xf1c0		0xf5ac	0xf1c0	0xf5ac	0xf1c0				
	0xf5ac		193.25	206.75	190.00	175.25				
	239.50		212.50	184.50	200.25	229.50				
	202.75		250.25	230.75	221.75	256.00				
	184.50		256.00	197.25	222.50	213.50				
	218.25									

3	17 18	0x0	0x0	0x0	0x0
	0x0	0x0	0x0	0x0	0x0
	0x0	0x0	0x0	0x0	0x0
	0x0	0x0	0x0	0x0	0x0
	0x0	239.50	166.25	206.50	165.25
	208.75	194.00	196.75	169.00	220.00
	204.75	207.50	165.75	208.50	228.75
	187.25	223.25	172.00	242.75	244.75
	256.00				
4	19 20	0x104d8	0x108b4	0x104d8	0x108b4
	0x104d8	0x108b4	0x104d8	0x108b4	0x104d8
	0x108b4	0x104d8	0x108b4	0x104d8	0x108b4
	0x104d8	0x108b4	0x104d8	0x108b4	0x104d8
	0x108b4	212.50	214.00	204.50	199.00
	177.00	221.00	208.25	204.50	187.50
	218.00	173.00	198.00	148.00	211.00
	189.75	206.75	183.50	211.50	246.50
	194.00				
5	16 22	0x17984	0x17984	0x17984	0x17984
	0x17984	0x17984	0x17984	0x17984	0x17984
	0x17984	0x17984	0x17984	0x17984	0x17984
	0x17984	0x17984	0x17984	0x17984	0x17984
	0x17984	207.25	201.25	196.50	222.25
	207.25	203.50	219.75	211.75	226.00
	239.25	185.25	209.00	208.75	210.00
	239.75	208.00	213.75	188.00	220.25
	225.25				
6	18 22	0x2dc0	0x2dc0	0x2dc0	0x2dc0
	0x2dc0	0x2dc0	0x2dc0	0x2dc0	0x2dc0
	0x2dc0	0x2dc0	0x2dc0	0x2dc0	0x2dc0
	0x2dc0	0x2dc0	0x2dc0	0x2dc0	0x2dc0
	0x2dc0	194.25	185.25	218.00	196.50
	176.75	180.00	199.25	227.25	212.00
	190.00	229.75	229.50	208.50	186.00
	184.50	186.50	164.00	215.00	199.75
	210.50				
7	11 23	0x14c64	0x14c64	0x14c64	0x14c64
	0x14c64	0x14c64	0x14c64	0x14c64	0x14c64
	0x14c64	0x14c64	0x14c64	0x14c64	0x14c64
	0x14c64	0x14c64	0x14c64	0x14c64	0x14c64
	0x14c64	217.75	161.50	193.00	232.25
	201.50	212.75	208.75	207.00	225.50
	216.50	224.50	223.50	185.50	196.75
	217.50	186.75	170.50	221.00	242.25
	241.50				

8	26 41	0x11020	0x113f0	0x11020	0x113f0
	0x11020	0x113f0	0x11020	0x113f0	0x11020
	0x113f0	0x11020	0x113f0	0x11020	0x113f0
	0x11020	0x113f0	0x11020	0x113f0	0x11020
	0x113f0	177.50	236.00	196.75	191.00
	202.50	208.00	240.00	192.00	218.75
	183.25	160.00	189.00	187.00	201.50
	225.25	161.25	243.75	224.25	175.00
	175.75				
9	45 47	0x9f00	0xa284	0x9f00	0xa284
	0x9f00	0xa284	0x9f00	0xa284	0x9f00
	0xa284	0x9f00	0xa284	0x9f00	0xa284
	0x9f00	0xa284	0x9f00	0xa284	0x9f00
	0xa284	219.00	199.75	155.00	196.00
	216.75	179.00	208.25	230.75	214.75
	238.00	217.00	221.50	214.75	238.50
	186.00	204.50	199.00	218.50	224.25
	200.50				
10	44 53	0x1af8	0x1e88	0x1af8	0x1e88
	0x1af8	0x1e88	0x1af8	0x1e88	0x1af8
	0x1e88	0x1af8	0x1e88	0x1af8	0x1e88
	0x1af8	0x1e88	0x1af8	0x1e88	0x1af8
	0x1e88	159.50	185.00	198.75	179.50
	228.00	241.50	218.25	167.00	202.50
	189.75	212.75	191.75	197.75	212.25
	202.25	188.25	184.25	199.00	201.25
	244.75				
11	40 54	0x97b4	0x9b24	0x97b4	0x9b24
	0x97b4	0x9b24	0x97b4	0x9b24	0x97b4
	0x9b24	0x97b4	0x9b24	0x97b4	0x9b24
	0x97b4	0x9b24	0x97b4	0x9b24	0x97b4
	0x9b24	163.75	236.25	181.50	186.00
	211.50	188.25	210.75	207.25	182.25
	222.25	181.75	200.75	219.50	246.25
	188.50	191.50	167.00	189.25	192.50
	235.50				
12	29 55	0x15bc0	0x15f98	0x15bc0	0x15f98
	0x15bc0	0x15f98	0x15bc0	0x15f98	0x15bc0
	0x15f98	0x15bc0	0x15f98	0x15bc0	0x15f98
	0x15bc0	0x15f98	0x15bc0	0x15f98	0x15bc0
	0x15f98	230.25	196.75	217.50	224.50
	185.75	178.50	135.75	189.25	217.00
	146.50	165.50	187.75	190.50	206.25
	195.25	181.75	245.25	183.00	187.50
	228.25				

13	35 55	0x9080	0x9418	0x9080	0x9418
	0x9080	0x9418	0x9080	0x9418	0x9080
	0x9418	0x9080	0x9418	0x9080	0x9418
	0x9080	0x9418	0x9080	0x9418	0x9080
	0x9418	214.25	204.25	187.50	162.00
	209.25	224.00	124.75	215.75	182.25
	222.75	210.25	202.00	223.25	220.25
	168.25	195.00	229.75	173.50	217.00
	231.00				
14	27 58	0x400	0x800	0x400	0x800
	0x400	0x800	0x400	0x800	0x400
	0x800	0x400	0x800	0x400	0x800
	0x400	0x800	0x400	0x800	0x400
	0x800	251.25	218.50	256.00	212.25
	212.25	214.25	237.75	218.75	198.50
	207.00	186.50	171.25	251.25	213.75
	227.50	208.75	208.25	218.00	160.50
	192.25				
15	31 58	0x84c8	0x888c	0x84c8	0x888c
	0x84c8	0x888c	0x84c8	0x888c	0x84c8
	0x888c	0x84c8	0x888c	0x84c8	0x888c
	0x84c8	0x888c	0x84c8	0x888c	0x84c8
	0x888c	178.25	226.00	137.00	239.75
	205.00	220.50	240.25	228.75	160.00
	252.75	182.00	151.50	212.75	226.25
	205.00	249.50	210.25	199.75	233.50
	208.00				
16	32 59	0xb6c	0xf1c	0xb6c	0xf1c
	0xb6c	0xf1c	0xb6c	0xf1c	0xb6c
	0xf1c	0xb6c	0xf1c	0xb6c	0xf1c
	0xb6c	0xf1c	0xb6c	0xf1c	0xb6c
	0xf1c	202.50	256.00	199.75	190.50
	204.50	153.25	170.00	212.50	136.50
	207.75	203.50	236.50	187.50	209.75
	235.25	192.75	190.25	256.00	222.75
	167.75				
17	55 61	0x184e8	0x18870	0x184e8	0x18870
	0x184e8	0x18870	0x184e8	0x18870	0x184e8
	0x18870	0x184e8	0x18870	0x184e8	0x18870
	0x184e8	0x18870	0x184e8	0x18870	0x184e8
	0x18870	191.25	219.50	184.00	206.25
	168.00	177.50	214.25	212.00	195.25
	216.25	216.25	208.50	208.00	194.50
	196.75	206.25	195.00	200.25	225.50
	175.50				

18	30 63	0x117a0	0x11b7c	0x117a0	0x11b7c
	0x117a0	0x11b7c	0x117a0	0x11b7c	0x117a0
	0x11b7c	0x117a0	0x11b7c	0x117a0	0x11b7c
	0x117a0	0x11b7c	0x117a0	0x11b7c	0x117a0
	0x11b7c	145.25	237.00	179.50	195.25
	203.50	202.50	185.50	184.75	157.25
	217.00	247.00	180.50	183.75	175.50
	145.75	195.50	192.75	222.75	192.00
	188.75				
19	34 64	0x12624	0x129b4	0x12624	0x129b4
	0x12624	0x129b4	0x12624	0x129b4	0x12624
	0x129b4	0x12624	0x129b4	0x12624	0x129b4
	0x12624	0x129b4	0x12624	0x129b4	0x12624
	0x129b4	204.50	214.75	179.50	196.75
	181.00	218.25	219.25	226.75	215.75
	222.50	228.00	209.50	211.00	198.75
	214.50	214.50	160.75	220.25	199.25
	164.75				
20	62 70	0x12d40	0x13140	0x12d40	0x13140
	0x12d40	0x13140	0x12d40	0x13140	0x12d40
	0x13140	0x12d40	0x13140	0x12d40	0x13140
	0x12d40	0x13140	0x12d40	0x13140	0x12d40
	0x13140	199.75	211.00	189.00	205.00
	200.25	189.00	219.00	172.75	220.00
	190.25	190.00	200.50	256.00	208.50
	241.25	208.75	176.75	204.75	190.00
	221.75				
21	72 74	0x134b8	0x138b4	0x134b8	0x138b4
	0x134b8	0x138b4	0x134b8	0x138b4	0x134b8
	0x138b4	0x134b8	0x138b4	0x134b8	0x138b4
	0x134b8	0x138b4	0x134b8	0x138b4	0x134b8
	0x138b4	230.50	233.50	203.75	216.75
	180.25	212.75	255.00	200.00	220.00
	220.00	162.00	225.25	208.75	156.50
	246.00	214.50	225.00	245.25	187.00
	222.50				
22	60 75	0x166e0	0x16ab4	0x166e0	0x16ab4
	0x166e0	0x16ab4	0x166e0	0x16ab4	0x166e0
	0x16ab4	0x166e0	0x16ab4	0x166e0	0x16ab4
	0x166e0	0x16ab4	0x166e0	0x16ab4	0x166e0
	0x16ab4	194.25	207.50	215.25	191.75
	172.50	145.25	193.75	221.00	245.00
	201.75	178.75	167.75	207.00	176.50
	208.00	201.75	208.75	224.75	192.50
	224.50				

23	70 75	0xd7d0	0xdb5c	0xd7d0	0xdb5c
	0xd7d0	0xdb5c	0xd7d0	0xdb5c	0xd7d0
	0xdb5c	0xd7d0	0xdb5c	0xd7d0	0xdb5c
	0xd7d0	0xdb5c	0xd7d0	0xdb5c	0xd7d0
	0xdb5c	217.25	196.50	173.75	231.75
	206.75	192.00	224.50	208.50	177.00
	214.00	226.25	227.50	208.00	221.75
	209.00	197.75	171.50	173.75	195.75
	219.25				
24	20 77	0x1408c	0x1448c	0x1408c	0x1448c
	0x1408c	0x1448c	0x1408c	0x1448c	0x1408c
	0x1448c	0x1408c	0x1448c	0x1408c	0x1448c
	0x1408c	0x1448c	0x1408c	0x1448c	0x1408c
	0x1448c	256.00	198.00	222.50	196.75
	191.00	199.00	167.50	223.50	203.75
	193.75	205.25	170.00	220.00	198.25
	217.00	248.50	208.00	214.00	234.50
	240.75				
25	69 77	0x225c	0x2600	0x225c	0x2600
	0x225c	0x2600	0x225c	0x2600	0x225c
	0x2600	0x225c	0x2600	0x225c	0x2600
	0x225c	0x2600	0x225c	0x2600	0x225c
	0x2600	199.00	183.75	208.50	181.00
	191.25	191.25	193.00	242.25	232.50
	227.50	192.25	227.00	194.25	235.50
	163.50	162.75	206.25	163.25	214.00
	222.50				
26	71 77	0xc070	0xc470	0xc070	0xc470
	0xc070	0xc470	0xc070	0xc470	0xc070
	0xc470	0xc070	0xc470	0xc070	0xc470
	0xc070	0xc470	0xc070	0xc470	0xc070
	0xc470	165.00	169.25	203.25	211.75
	199.25	201.50	210.50	254.50	256.00
	224.25	179.25	162.50	197.25	196.75
	217.25	214.00	183.00	202.75	175.25
	256.00				
27	16 78	0x17d44	0x18128	0x17d44	0x18128
	0x17d44	0x18128	0x17d44	0x18128	0x17d44
	0x18128	0x17d44	0x18128	0x17d44	0x18128
	0x17d44	0x18128	0x17d44	0x18128	0x17d44
	0x18128	209.00	239.75	240.25	211.50
	202.75	216.25	203.50	177.50	189.75
	236.00	184.25	207.00	184.25	216.50
	210.00	201.50	214.75	223.50	249.00
	162.50				

28	22 78	0x47b0	0x47b0	0x47b0	0x47b0
	0x47b0	0x47b0	0x47b0	0x47b0	0x47b0
	0x47b0	0x47b0	0x47b0	0x47b0	0x47b0
	0x47b0	0x47b0	0x47b0	0x47b0	0x47b0
	0x47b0	210.25	185.00	164.00	234.00
	171.75	172.25	198.50	213.50	211.75
	191.75	185.00	204.50	201.00	233.25
	220.25	228.75	158.50	204.50	207.25
	184.00				
29	23 78	0xd024	0xd024	0xd024	0xd024
	0xd024	0xd024	0xd024	0xd024	0xd024
	0xd024	0xd024	0xd024	0xd024	0xd024
	0xd024	0xd024	0xd024	0xd024	0xd024
	0xd024	215.00	234.75	209.75	225.75
	176.50	189.50	209.00	198.00	216.00
	187.75	157.25	190.50	177.25	216.75
	221.75	179.50	215.50	220.50	173.75
	194.25				
30	57 78	0xadcc	0xb160	0xb510	0xadcc
	0xb160	0xb510	0xadcc	0xb160	0xb510
	0xadcc	0xb160	0xb510	0xadcc	0xb160
	0xb510	0xadcc	0xb160	0xb510	0xadcc
	0xb160	182.50	235.50	225.00	166.25
	207.75	207.00	185.25	191.00	174.75
	223.75	216.00	212.25	228.25	208.00
	224.25	225.75	185.75	233.50	205.25
	191.00				
31	63 78	0x3500	0x389c	0x3500	0x389c
	0x3500	0x389c	0x3500	0x389c	0x3500
	0x389c	0x3500	0x389c	0x3500	0x389c
	0x3500	0x389c	0x3500	0x389c	0x3500
	0x389c	188.50	186.75	203.75	195.25
	194.25	172.50	204.75	241.25	141.50
	190.00	228.75	173.00	174.50	182.25
	221.75	185.50	230.50	160.50	189.00
	214.50				
32	77 78	0x4b58	0x4f14	0x4b58	0x4f14
	0x4b58	0x4f14	0x4b58	0x4f14	0x4b58
	0x4f14	0x4b58	0x4f14	0x4b58	0x4f14
	0x4b58	0x4f14	0x4b58	0x4f14	0x4b58
	0x4f14	198.50	167.25	198.25	182.00
	179.50	172.50	238.50	193.00	191.25
	169.25	224.00	173.00	164.00	214.75
	209.00	206.50	189.25	251.50	228.00
	194.25				

33	36 79	0x131c	0x171c	0x131c	0x171c
	0x131c	0x171c	0x131c	0x171c	0x131c
	0x171c	0x131c	0x171c	0x131c	0x171c
	0x131c	0x171c	0x131c	0x171c	0x131c
	0x171c	242.75	196.00	208.25	192.50
	212.00	234.25	256.00	184.50	179.75
	234.00	212.50	246.75	228.50	183.25
	221.00	178.50	194.50	179.00	237.00
	199.25				
34	77 79	0x5304	0x5704	0x5304	0x5704
	0x5304	0x5704	0x5304	0x5704	0x5304
	0x5704	0x5304	0x5704	0x5304	0x5704
	0x5304	0x5704	0x5304	0x5704	0x5304
	0x5704	214.25	223.00	183.00	182.00
	256.00	225.25	179.50	185.25	226.75
	195.25	184.00	256.00	194.25	194.00
	203.00	240.25	248.00	194.00	203.75
	173.00				
35	78 80	0x6304	0x668c	0x6304	0x668c
	0x6304	0x668c	0x6304	0x668c	0x6304
	0x668c	0x6304	0x668c	0x6304	0x668c
	0x6304	0x668c	0x6304	0x668c	0x6304
	0x668c	203.25	185.75	218.75	232.25
	207.00	195.50	158.25	218.50	220.50
	193.75	199.25	196.25	226.00	178.50
	213.25	175.00	206.50	239.75	181.00
	175.00				
36	50 82	0xa640	0xa9e8	0xa640	0xa9e8
	0xa640	0xa9e8	0xa640	0xa9e8	0xa640
	0xa9e8	0xa640	0xa9e8	0xa640	0xa9e8
	0xa640	0xa9e8	0xa640	0xa9e8	0xa640
	0xa9e8	214.75	166.25	201.25	249.00
	183.25	170.50	204.50	234.00	204.00
	185.75	225.00	218.75	214.75	215.50
	233.75	173.75	223.50	218.25	228.50
	176.00				
37	24 86	0x7da0	0x7da0	0x7da0	0x7da0
	0x7da0	0x7da0	0x7da0	0x7da0	0x7da0
	0x7da0	0x7da0	0x7da0	0x7da0	0x7da0
	0x7da0	0x7da0	0x7da0	0x7da0	0x7da0
	0x7da0	175.75	146.75	212.25	217.50
	214.00	159.50	221.50	171.00	204.50
	224.00	192.75	205.75	223.50	176.50
	239.50	219.50	202.75	234.25	220.50
	200.50				

38 25 86 0x15810 0x15810 0x15810 0x15810
0x15810 0x15810 0x15810 0x15810 0x15810
0x15810 0x15810 0x15810 0x15810 0x15810
0x15810 0x15810 0x15810 0x15810 0x15810
0x15810 233.25 213.50 189.00 198.75
231.00 217.00 174.50 208.50 224.50
176.50 225.00 204.75 205.75 217.00
203.25 182.75 154.25 236.00 224.50
175.25

39 79 86 0xdefc 0xe2fc 0xdefc 0xe2fc
0xdefc 0xe2fc 0xdefc 0xe2fc 0xdefc
0xe2fc 0xdefc 0xe2fc 0xdefc 0xe2fc
0xdefc 0xe2fc 0xdefc 0xe2fc 0xdefc
0xe2fc 196.75 151.75 211.25 222.75
237.50 166.25 165.00 224.75 228.75
214.75 220.50 192.75 205.75 171.25
229.50 207.00 153.25 199.50 256.00
202.25

40 80 86 0x7230 0x75d8 0x7230 0x75d8
0x7230 0x75d8 0x7230 0x75d8 0x7230
0x75d8 0x7230 0x75d8 0x7230 0x75d8
0x7230 0x75d8 0x7230 0x75d8 0x7230
0x75d8 205.00 214.25 208.00 230.00
226.25 176.75 176.00 232.00 163.00
187.50 231.25 223.75 179.50 194.50
184.00 241.50 233.75 209.25 163.00
214.50

41 81 86 0xcc24 0xcc24 0xcc24 0xcc24
0xcc24 0xcc24 0xcc24 0xcc24 0xcc24
0xcc24 0xcc24 0xcc24 0xcc24
0xcc24 0xcc24 0xcc24 0xcc24 0xcc24
0xcc24 0xcc24 182.00 156.50 217.75
243.25 211.50 188.50 224.25 208.75
200.50 177.25 161.50 200.00 232.50
210.25 189.75 213.00 200.75 256.00
196.75 221.75

42 19 87 0x10c28 0x10c28 0x10c28 0x10c28
0x10c28 0x10c28 0x10c28 0x10c28 0x10c28
0x10c28 0x10c28 0x10c28 0x10c28 0x10c28
0x10c28 0x10c28 0x10c28 0x10c28 0x10c28
0x10c28 192.25 182.25 217.75 233.50
243.75 211.25 246.00 231.00 181.50
220.50 219.25 145.50 219.75 189.50
240.50 208.75 254.00 195.75 239.25
227.25

43	69 87	0x29cc	0x29cc	0x29cc	0x29cc
	0x29cc	0x29cc	0x29cc	0x29cc	0x29cc
	0x29cc	0x29cc	0x29cc	0x29cc	0x29cc
	0x29cc	0x29cc	0x29cc	0x29cc	0x29cc
	0x29cc	199.50	182.50	230.00	236.75
	210.50	196.75	208.50	223.25	218.50
	243.50	222.25	234.75	155.50	211.00
	189.00	201.50	221.25	179.50	188.50
	252.25				
44	71 87	0xc870	0xc870	0xc870	0xc870
	0xc870	0xc870	0xc870	0xc870	0xc870
	0xc870	0xc870	0xc870	0xc870	0xc870
	0xc870	0xc870	0xc870	0xc870	0xc870
	0xc870	216.25	235.00	217.00	197.50
	198.50	173.75	235.75	216.50	166.25
	174.00	200.25	232.75	185.25	236.75
	182.50	215.50	187.25	197.25	217.50
	197.50				
45	18 88	0x3158	0x3158	0x3158	0x3158
	0x3158	0x3158	0x3158	0x3158	0x3158
	0x3158	0x3158	0x3158	0x3158	0x3158
	0x3158	0x3158	0x3158	0x3158	0x3158
	0x3158	181.00	168.00	175.25	198.50
	210.50	203.00	174.00	212.00	196.75
	156.00	228.25	217.25	185.25	201.25
	222.25	205.00	233.75	193.25	190.25
	142.25				
46	63 88	0x3c64	0x3c64	0x3c64	0x3c64
	0x3c64	0x3c64	0x3c64	0x3c64	0x3c64
	0x3c64	0x3c64	0x3c64	0x3c64	0x3c64
	0x3c64	0x3c64	0x3c64	0x3c64	0x3c64
	0x3c64	225.75	171.25	201.75	220.75
	162.75	213.00	228.50	227.50	224.25
	175.75	179.00	176.75	208.25	231.75
	191.50	205.00	192.25	230.00	178.50
	164.50				
47	77 88	0x5b04	0x5b04	0x5b04	0x5b04
	0x5b04	0x5b04	0x5b04	0x5b04	0x5b04
	0x5b04	0x5b04	0x5b04	0x5b04	0x5b04
	0x5b04	0x5b04	0x5b04	0x5b04	0x5b04
	0x5b04	256.00	239.50	199.75	236.25
	200.50	232.25	256.00	196.50	201.00
	210.00	196.50	209.00	195.50	210.25
	196.50	208.25	213.75	191.00	169.75
	161.00				

48	13 89	0x14870	0x14870	0x14870	0x14870
	0x14870	0x14870	0x14870	0x14870	0x14870
	0x14870	0x14870	0x14870	0x14870	0x14870
	0x14870	0x14870	0x14870	0x14870	0x14870
	0x14870	174.25	232.75	212.50	210.50
	227.00	203.50	165.75	216.25	221.00
	252.50	240.00	174.75	226.00	196.25
	243.50	221.00	213.00	164.25	201.25
	169.00				
49	58 89	0xd3d0	0xd3d0	0xd3d0	0xd3d0
	0xd3d0	0xd3d0	0xd3d0	0xd3d0	0xd3d0
	0xd3d0	0xd3d0	0xd3d0	0xd3d0	0xd3d0
	0xd3d0	0xd3d0	0xd3d0	0xd3d0	0xd3d0
	0xd3d0	244.75	237.75	206.50	202.75
	164.50	180.75	229.00	229.00	203.50
	221.25	184.25	183.00	256.00	189.00
	225.75	216.00	250.25	237.75	216.75
	203.50				
50	73 89	0x16e38	0x17238	0x16e38	0x17238
	0x16e38	0x17238	0x16e38	0x17238	0x16e38
	0x17238	0x16e38	0x17238	0x16e38	0x17238
	0x16e38	0x17238	0x16e38	0x17238	0x16e38
	0x17238	183.25	200.50	247.75	181.50
	203.75	212.00	224.25	207.25	224.75
	196.75	190.75	172.75	190.25	228.75
	256.00	225.00	205.75	225.75	249.25
	168.25				
51	10 90	0xf9ac	0xfdac	0x10144	0xf9ac
	0xfdac	0x10144	0xf9ac	0xfdac	0x10144
	0xf9ac	0xfdac	0x10144	0xf9ac	0xfdac
	0x10144	0xf9ac	0xfdac	0x10144	0xf9ac
	0xfdac	177.00	188.75	189.75	181.25
	195.75	171.25	201.00	220.75	183.50
	229.50	229.50	222.25	202.50	196.50
	229.00	256.00	212.75	226.75	239.00
	207.50				
52	11 90	0x15030	0x15430	0x15030	0x15430
	0x15030	0x15430	0x15030	0x15430	0x15030
	0x15430	0x15030	0x15430	0x15030	0x15430
	0x15030	0x15430	0x15030	0x15430	0x15030
	0x15430	202.75	244.00	177.50	234.25
	254.25	222.50	188.25	162.75	204.75
	184.25	197.25	186.25	209.25	184.75
	191.75	205.50	184.50	186.25	256.00
	247.25				

53	57 90	0xb8b8	0xbca0	0xb8b8	0xbca0
	0xb8b8	0xbca0	0xb8b8	0xbca0	0xb8b8
	0xbca0	0xb8b8	0xbca0	0xb8b8	0xbca0
	0xb8b8	0xbca0	0xb8b8	0xbca0	0xb8b8
	0xbca0	222.00	244.00	199.00	232.25
	249.50	222.25	247.75	177.50	169.25
	189.75	149.25	198.25	213.75	188.00
	215.00	224.25	158.00	226.75	200.00
	168.25				
54	63 90	0x4004	0x43cc	0x4004	0x43cc
	0x4004	0x43cc	0x4004	0x43cc	0x4004
	0x43cc	0x4004	0x43cc	0x4004	0x43cc
	0x4004	0x43cc	0x4004	0x43cc	0x4004
	0x43cc	209.00	185.25	203.50	248.25
	200.00	215.75	200.00	183.00	214.50
	219.25	163.25	194.75	241.50	232.75
	207.75	186.75	189.25	234.00	174.25
	184.00				
55	78 90	0x6a4c	0x6a4c	0x6a4c	0x6a4c
	0x6a4c	0x6a4c	0x6a4c	0x6a4c	0x6a4c
	0x6a4c	0x6a4c	0x6a4c	0x6a4c	0x6a4c
	0x6a4c	0x6a4c	0x6a4c	0x6a4c	0x6a4c
	0x6a4c	210.50	181.75	218.75	211.75
	194.00	222.00	202.25	230.75	248.75
	228.50	205.50	205.25	225.25	218.25
	187.00	239.25	195.00	183.50	152.75
	181.75				
56	6 91	0xee10	0xee10	0xee10	0xee10
	0xee10	0xee10	0xee10	0xee10	0xee10
	0xee10	0xee10	0xee10	0xee10	0xee10
	0xee10	0xee10	0xee10	0xee10	0xee10
	0xee10	207.25	198.75	203.75	164.50
	214.75	235.75	191.25	212.50	206.25
	223.00	210.00	193.75	228.75	231.75
	208.25	201.00	212.50	200.75	229.25
	191.25				
57	42 91	0x1632c	0x1632c	0x1632c	0x1632c
	0x1632c	0x1632c	0x1632c	0x1632c	0x1632c
	0x1632c	0x1632c	0x1632c	0x1632c	0x1632c
	0x1632c	0x1632c	0x1632c	0x1632c	0x1632c
	0x1632c	203.50	189.00	193.25	194.50
	188.75	223.25	182.75	177.00	199.75
	216.75	194.50	151.25	190.75	236.25
	215.25	178.75	236.00	209.00	159.75
	235.25				

58	78 91	0x6e30	0x6e30	0x6e30	0x6e30
	0x6e30	0x6e30	0x6e30	0x6e30	0x6e30
	0x6e30	0x6e30	0x6e30	0x6e30	0x6e30
	0x6e30	0x6e30	0x6e30	0x6e30	0x6e30
	0x6e30	219.50	216.00	215.00	195.75
	177.00	256.00	206.75	157.00	205.75
	213.50	198.25	198.00	168.00	168.50
	199.50	240.50	204.50	224.25	147.00
	228.50				
59	2 92	0xea80	0xea80	0xea80	0xea80
	0xea80	0xea80	0xea80	0xea80	0xea80
	0xea80	0xea80	0xea80	0xea80	0xea80
	0xea80	0xea80	0xea80	0xea80	0xea80
	0xea80	219.25	208.00	197.25	212.25
	215.00	191.50	209.00	189.25	190.50
	203.25	200.25	182.75	217.50	206.25
	199.75	213.00	227.75	178.25	204.00
	211.25				
60	31 92	0x8c80	0x8c80	0x8c80	0x8c80
	0x8c80	0x8c80	0x8c80	0x8c80	0x8c80
	0x8c80	0x8c80	0x8c80	0x8c80	0x8c80
	0x8c80	0x8c80	0x8c80	0x8c80	0x8c80
	0x8c80	220.25	194.25	226.50	181.50
	220.00	160.75	211.25	226.50	206.75
	208.00	175.25	225.25	256.00	207.50
	217.00	232.75	214.75	214.75	196.75
	188.75				
61	73 92	0x175cc	0x175cc	0x175cc	0x175cc
	0x175cc	0x175cc	0x175cc	0x175cc	0x175cc
	0x175cc	0x175cc	0x175cc	0x175cc	0x175cc
	0x175cc	0x175cc	0x175cc	0x175cc	0x175cc
	0x175cc	219.00	185.50	226.50	216.50
	189.00	212.00	197.75	218.75	199.25
	206.00	217.75	224.00	236.00	212.00
	159.50	223.75	237.50	203.00	166.25
	229.00				
62	30 93	0x11f30	0x12288	0x11f30	0x12288
	0x11f30	0x12288	0x11f30	0x12288	0x11f30
	0x12288	0x11f30	0x12288	0x11f30	0x12288
	0x11f30	0x12288	0x11f30	0x12288	0x11f30
	0x12288	191.00	225.25	203.25	196.50
	186.25	193.75	162.00	204.75	213.75
	177.75	201.00	208.50	195.00	184.25
	208.50	185.25	187.75	230.75	192.25
	220.25				

63	77 93	0x5f04	0x5f04	0x5f04	0x5f04
	0x5f04	0x5f04	0x5f04	0x5f04	0x5f04
	0x5f04	0x5f04	0x5f04	0x5f04	0x5f04
	0x5f04	0x5f04	0x5f04	0x5f04	0x5f04
	0x5f04	198.25	173.00	191.00	201.50
	164.75	179.00	255.75	219.25	207.25
	155.00	189.25	172.50	219.75	225.25
	202.50	245.25	185.00	209.75	217.50
	212.25				
64	1 94	0x79a0	0x79a0	0x79a0	0x79a0
	0x79a0	0x79a0	0x79a0	0x79a0	0x79a0
	0x79a0	0x79a0	0x79a0	0x79a0	0x79a0
	0x79a0	0x79a0	0x79a0	0x79a0	0x79a0
	0x79a0	227.50	214.00	173.50	188.75
	170.25	211.25	220.25	234.00	172.75
	190.00	255.50	193.00	240.00	191.50
	250.00	216.50	153.50	195.25	189.25
	175.00				
65	28 94	0x8160	0x8160	0x8160	0x8160
	0x8160	0x8160	0x8160	0x8160	0x8160
	0x8160	0x8160	0x8160	0x8160	0x8160
	0x8160	0x8160	0x8160	0x8160	0x8160
	0x8160	191.75	197.25	217.75	206.50
	207.00	215.00	195.50	202.25	205.25
	193.50	171.00	208.25	216.25	212.50
	173.25	202.25	204.75	152.50	203.00
	186.75				
66	82 94	0x18be0	0x18be0	0x18be0	0x18be0
	0x18be0	0x18be0	0x18be0	0x18be0	0x18be0
	0x18be0	0x18be0	0x18be0	0x18be0	0x18be0
	0x18be0	0x18be0	0x18be0	0x18be0	0x18be0
	0x18be0	219.75	211.75	191.50	239.00
	179.00	204.25	218.25	159.25	206.00
	185.50	168.75	242.00	200.00	199.00
	175.50	192.75	188.25	222.50	211.75
	220.75				

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br