UNIVERSIDADE FEDERAL DO CEARÁ – UFC

CENTRO DE TECNOLOGIA – CT

DEPARTAMENTO DE ENGENHARIA DE TELEINFOMÁTICA – DETI

GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

Raul Holanda Cordeiro

HermesFT: Uma Versão da NoC Hermes Mais Robusta e Tolerante a Falhas

Fortaleza/CE

Dezembro 2011

Raul Holanda Cordeiro

HermesFT: Uma Versão da NoC Hermes Mais Robusta e Tolerante a Falhas

Monografia de Conclusão de Curso apresentada à Coordenação do Curso de Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará como parte dos requisitos para obtenção do grau de ***Engenheiro de Teleinformática****.*

Orientador: Prof. Msc. Jarbas Aryel Nunes Silveira

Fortaleza/CE

Dezembro 2011

Raul Holanda Cordeiro

HermesFT: Uma Versão da NoC Hermes Mais Robusta e Tolerante a Falhas

Monografia de Conclusão de Curso apresentada à Coordenação do Curso de Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará como parte dos requisitos para obtenção do grau de ***Engenheiro de Teleinformática****.*

Orientador: Prof. Msc. Jarbas Aryel Nunes Silveira

COMISSÃO EXAMINADORA

_____

Orientador – Prof. Msc. Jarbas Aryel Nunes Silveira

_____

Prof. Msc. Ricardo Jardel Nunes da Silveira

_____

Prof. Msc. Alexandre Augusto da Penha Coelho

Fortaleza/CE

Dezembro 2011

Resumo

Neste trabalho é realizada uma breve apresentação dos conceitos envolvidos no estudo de redes-em-chip e uma análise das fragilidades, frente a falhas transitórias e permanentes, da NoC Hermes. Após essa análise, são propostas algumas alterações que trazem maior tolerância dessa rede a esses tipos de falhas, gerando assim maior disponibilidade da arquitetura de comunicação ao sistema. Algumas dessas alterações são simples, porém bastante eficazes para alguns casos específicos. Outras são mais elaboradas, sem perder, contudo, a simplicidade necessária a qualquer mecanismo de detecção/recuperação de falhas. A detecção é basicamente temporal, através da utilização de um *watch dog timer*. Já a recuperação é baseada no reset (útil no caso de falhas transitórias) e no curto-circuito dos enlaces (útil no caso de falhas permanentes) do roteador em falha. Após a realização das alterações propostas, foram realizados testes em quatro cenários, inclusive com injeção de falhas do tipo preso-em-zero (do inglês, *stuck-at-zero*) e erro transitório de bit. Os resultados obtidos mostraram a eficiência, quando comparados aos resultados da rede original, e mostrou que a meta deste trabalho foi alcançada.

Palavras-chaves: Microeletrônica, Sistemas Integrados, Rede-em-Chip, *Watch Dog Timer*, Curto-Circuito.

Abstract

In this work a brief presentation of the concepts involved in the study of networks-on-chip and an analysis of the Hermes NoC weaknesses, about transient and permanent faults are done. After this analysis, it is proposed some changes that bring greater tolerance at presence of these types of network failures, thereby generating greater availability of communications architecture to the system. Some of these changes are simple but very effective for some specific cases. Others are more elaborate, without losing, however, the simplicity needed to any detection/recovery mechanism. The detection is made over time and implemented by a watch dog timer. Recovery is based on reset (useful in case of transient faults) and on the link's short-circuiting (useful for permanent faults) of router in failure. After the proposed changes are implemented, tests were performed in four scenarios, including with fault injection of type stuck-at-zero and transient bit error. The results showed the efficiency gain of the NoC HermesFT when compared to the results of the original NoC.

Keywords: Microeletronics, System on Chip, Network on Chip, Watch Dog Timer, Short-Circuiting.

Dedico este trabalho, primeiramente, a Deus por iluminar o caminho que segui até aqui. Em segundo lugar, aos meus pais, Rosa Maria Holanda Cordeiro e Sebastião Cordeiro de Vasconcelos, pelos valiosos e mais importantes ensinamentos que me foram passados. Em terceiro, mas não menos importante que os anteriores, a minha tia-mãe Maria Rosali Holanda pelo apoio dado nos momentos difíceis da vida estudantil deste aspirante a engenheiro.

Agradecimentos

À minha namorada, Camila, pela paciência, compreensão, ajuda e amor necessários para a realização desse trabalho,

Aos amigos extra-universidade pelo apoio e que, pelo convívio agradável, contribuíram indiretamente,

Aos companheiros do curso de graduação em Engenharia de Teleinformática, pela amizade, convivência e paciência (principalmente em momentos de estresse, como os finais de semestre) no ambiente universitário,

Aos colegas do Laboratório de Engenharia de Sistemas de Computação (LESC), junto do Prof. Msc. Alexandre Moreira (vulgo Sobral), pelas primeiras experiências profissionais que tanto valor tem na vida de um engenheiro,

Aos colegas de trabalho da empresa Trana Construções LTDA, pelo amadurecimento dos conceitos aprendidos na faculdade em um ambiente de trabalho desafiador e gratificante,

Ao Professor Jarbas Aryel Nunes Silveira, pela confiança, paciência, incentivo e dedicação durante toda a orientação acadêmica,

Aos demais professores do Departamento de Engenharia de Teleinformática, pelos conhecimentos transmitidos durante a graduação,

E à Universidade Federal do Ceará, pela formação de nível superior.

A educação tem raízes amargas,
mas seus frutos são doces.
Aristóteles

Sumário

Lista de Figuras ix
Lista de Tabelas xii
Lista de Siglas xiii
1 Introdução 1
1.1 Motivação 1
1.2 Revisão Bibliográfica 2
1.2.1 Pesquisas em SoC 2
1.2.2 Pesquisas em NoC 4
1.2.3 Pesquisas sobre Tolerância a Falhas em NoC 5
1.3 Objetivos 7
1.3.1 Objetivo Geral 7
1.3.2 Objetivos Específicos 7
1.4 Organização da Monografia 8
2 Conceitos de Redes Intrachip e de Tolerância a Falhas 9
2.1 Canais e Roteadores 9
2.2 Topologias e Mecanismos de Comunicação 11
2.2.1 Topologias de Rede de Interconexão 12
2.2.2 Mecanismos de Comunicação 13
2.2.2.1 Roteamento: Classificações e Problemas 14
2.2.2.2 Métodos, Modos e Algoritmos de Chaveamento 15
2.2.2.3 Exemplo de Algoritmo de Roteamento 16
2.2.2.4 Controle de Fluxo 17
2.2.2.5 Arbitragem 18
2.2.2.6 Memorização 18
2.2.2.7 Métricas de Desempenho da Rede 19
2.3 Tolerância a Falhas: Conceitos e Técnicas 20
2.3.1 Conceitos Introdutórios 20
2.3.2 Fases e Técnicas 23
2.3.2.1 Detecção de Erro 23
2.3.2.2 Confinamento do Dano 25
2.3.2.3 Recuperação do Erro 25

2.3.2.4 Tratamento da falha e continuidade do serviço 26
2.4 Conclusão 26
3 A NoC Hermes 27
3.1 A chave Hermes 27
3.1.1 Controle de Fluxo 27
3.1.2 Memorização 28
3.1.3 Algoritmos de Arbitragem e Chaveamento 28
3.1.4 Algoritmo de Roteamento 29
3.1.5 Exemplo de Funcionamento 30
3.1.6 Características Temporais 32
3.2 A Rede Hermes 33
3.3 Fragilidades 34
3.3.1 Controle de Timeout no Envio 34
3.3.2 Controle de Início e/ou Fim de Pacote 36
3.3.3 Falha Permanente 37
3.4 Conclusão 38
4 A NoC HermesFT 39
4.1 A Chave HermesFT 39
4.1.1 Controle de Fluxo 39
4.1.2 Memorização 40
4.1.3 Algoritmos de Arbitragem e Chaveamento 40
4.1.4 Algoritmo de Roteamento 40
4.1.5 Exemplo de Funcionamento 41
4.1.6 Características Temporais 41
4.2 O WDT 41
4.2.1 Eficácia do WDT 43
4.3 O Chaveador de Enlaces 44
4.4 A NoC HemesFT 46
4.4.1 Tipos de Curto-Circuito e suas Conseqüências 47
4.4.2 Parâmetros de *Timeout* 49
4.5 Fragilidades Vencidas e Robustez Adquirida 49
4.5.1 Controle de Timeout no Envio 49
4.5.2 Controle de Início e/ou Fim de Pacote 50
4.5.3 Falha Permanente 50

4.6 Conclusão 50
5 Análise dos Resultados 51
5.1 Metodologia Utilizada nos Testes 51
5.1.1 Criação da Rede 51
5.1.2 Geração do Padrão de Tráfego 52
5.1.3 Simulação 54
5.1.4 Análise de Tráfego 56
5.2 Cenários Utilizados nos Testes 57
5.3 Geração de Falhas 57
5.4 Apresentação dos Resultados 59
5.4.1 Cenário 1: Sem Falhas 59
5.4.2 Cenário 2: Roteador Não-Alvo com Falhas Transitórias 61
5.4.3 Cenário 3: Roteador Não-Alvo com Falha Permanente 65
5.4.4 Cenário 4: Roteador Alvo com Falha Permanente 68
5.5 Conclusões Sobre os Resultados 71
5.5.1 Cenário 1 71
5.5.2 Cenário 2 71
5.5.3 Cenário 3 73
5.5.4 Cenário 4 75
6 Conclusões 77
6.1 Perspectivas para Trabalhos Futuros 78
Referências Bibliográficas 79

Lista de Figuras

Figura 1 - Evolução do número de *cores* ao longo dos anos (GORDON, 2006). 3
Figura 2 - Arquiteturas de comunicação mais utilizadas: (a) ponto-a-ponto e (b) barramento (CARDOZO, 2005). 3
Figura 3 - Exemplo de SoC utilizando arquitetura de comunicação baseada em NoC (CARDOZO, 2005). 4
Figura 4 - Topologias mais utilizadas atualmente: (a) grelha e (b) torus (CARDOZO, 2005). . 5
Figura 5 - Representação do enlace de saída de um emissor ao seu receptor. 9
Figura 6 - Exemplo de dispersão espacial (grelha) de uma rede-em-chip..... 10
Figura 7 - Visão detalhada dos elementos internos do roteador de uma rede-em-chip (CARDOZO, 2005). 10
Figura 8 - Modelo de camadas OSI. 11
Figura 9 - Exemplo de rede direta, contendo um roteador para cada nodo do sistema. 12
Figura 10 - Topologias mais utilizadas atualmente: (a) grelha 2D, (b) torus e (c) cubo (ZEFERINO, 2003b). 13
Figura 11 - Rede crossbar 4 x 4. 13
Figura 12 - Classificações possíveis quanto ao algoritmo de roteamento utilizado (BEZERRA, 2009). 14
Figura 13 - Exemplo da trajetória dos *flits* de um pacote em uma rede malha 2D 3 x 3..... 17
Figura 14 - Diagrama de 3 Universos ilustrando a diferença entre falha, erro e defeito..... 22
Figura 15 - Representação temporal da falha e metodologia utilizada nas técnicas. 25
Figura 16 - Ligação dos sinais das portas de entrada e saída de duas chaves Hermes (BEZERRA, 2009). 28
Figura 17 - Diagrama de blocos destacando as unidades internas do roteador Hermes..... 28
Figura 18 - Exemplo de chave Hermes operando a máxima vazão..... 30
Figura 19 - Unidades internas de uma chave Hermes, os sinais que as interligam. Destaque para a seqüência em que o sinais são excitados (MORAES, 2003). 31
Figura 20 - Análise temporal dos eventos após a chegada de um pacote em uma porta de entrada da chave Hermes. 32
Figura 21 - Proposta inicial de topologia para a rede Hermes..... 33
Figura 22 - Opção de controle de timeout no recebimento de *flits* de um mesmo pacote..... 35
Figura 23 - Exemplo de formato de pacote mais robusto, com bits de sinalização para o início e fim de um pacote (ZEFERINO, 2003a). 36

Figura 24 - Cenário para apresentação das consequências de uma falha permanente em um roteador Hermes. 37
Figura 25 - Ligação dos sinais das portas de entrada e saída de duas chaves HermesFT. 40
Figura 26 - Representação em bloco, com entradas e saídas, do circuito WDT. 41
Figura 27 - Representação do circuito do roteador responsável por indicar existência de falha para o WDT. 43
Figura 28 - Bloco representativo do funcionamento de um multiplexador. 44
Figura 29 - Bloco representativo do funcionamento de um demultiplexador. 45
Figura 30 - Exemplo de implementação parcial do circuito chaveador entre duas portas de um roteador HermesFT. 46
Figura 31 - Rede HermesFT 3 x 3 e realce de um nodo. 47
Figura 32 - Formato de pacote escolhido para a rede HermesFT. 50
Figura 33 - Interface gráfica da ferramenta ATLAS. 51
Figura 34 - Geração da NoC Hermes com as características especificadas. 52
Figura 35 - Exemplo de configuração do tráfego gerado pelo core (ou *IP*) 00. 52
Figura 36 - Exemplo de arquivo gerado pelo padrão de tráfego especificado para o core (ou *IP*) 00. 53
Figura 37 - Exemplo de arquivo com os *flits* recebidos pelo core (ou *IP*) 22. 54
Figura 38 - Exemplo de arquivo gerado pela simulação com os *flits* enviados pela porta de saída *EAST* do roteador 00. 55
Figura 39 - Análise textual do número de *flits* enviados pelas portas de saída de todos os roteadores de uma rede Hermes 3 x 3. 56
Figura 40 - Análise gráfica do número de *flits* que percorreram cada enlace de uma rede Hermes 3 x 3. 57
Figura 41 - Exemplo de conteúdo do arquivo de configuração do módulo sabotador. 58
Figura 42 - Diagrama de blocos do processo de injeção de falhas no roteador. 59
Figura 43 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede Hermes no cenário 1. 60
Figura 44 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede Hermes no cenário 1. 60
Figura 45 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede HermesFT no cenário 1. 61
Figura 46 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede HermesFT no cenário 1. 61

Figura 47 - Arquivo com os *flits* de saída da porta *EAST* do roteador 00. 62
Figura 48 - Arquivo com os *flits* de saída da porta *EAST* do roteador 10. 62
Figura 49 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede Hermes no cenário 2. 63
Figura 50 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede Hermes no cenário 2. 63
Figura 51 - Arquivo com os *flits* de saída da porta *EAST* do roteador 00. 64
Figura 52 - Arquivo com os *flits* de saída da porta *EAST* do roteador 10. 64
Figura 53 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede HermesFT no cenário 2. 65
Figura 54 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede HermesFT no cenário 2. 65
Figura 55 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede Hermes no cenário 3. 66
Figura 56 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede Hermes no cenário 3. 66
Figura 57 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede HermesFT no cenário 3. 67
Figura 58 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede HermesFT no cenário 3. 67
Figura 59 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede Hermes no cenário 4. 68
Figura 60 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede Hermes no cenário 4. 68
Figura 61 - Arquivo com os *flits* enviados a porta GARBAGE do roteador 12. 69
Figura 62 - Arquivo com os *flits* enviados a porta GARBAGE do roteador 21. 69
Figura 63 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede HermesFT no cenário 4. 70
Figura 64 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede HermesFT no cenário 4. 70
Figura 65 - Arquivo com os *flits* recebidos pelo roteador 00. 72
Figura 66 - Forma de onda com os sinais do circuito detector de falha. 74

Lista de Tabelas

Tabela 1 - Comparativo entre arquitetura de comunicação intrachip (CARDOZO, 2005).....3
Tabela 2 - Características das técnicas de recuperação de falha. 25
Tabela 3 - Comparativo da disponibilidade das redes antes e depois da falha no cenário 2....73
Tabela 4 - Comparativo da disponibilidade das redes antes e depois da falha no cenário 3....75
Tabela 5 - Comparativo da disponibilidade das redes antes e depois da falha no cenário 4....76

Lista de Siglas

SoC System on Chip (Sistema em Chip)

WDT Watch Dog Timer (Cronômetro Cão de Guarda)

CMOS Complementary Metal-Oxide-Semiconductor (Semicondutor Metal-Óxido Complementar)

PUCRS Pontíficia Universidade Católica do Rio Grande do Sul

SPIN Scalable Programmable Integrated Network (Rede Integrada Programável e Escalável)

SoCIN System on Chip Interconnection Network (Rede de Interconexão de Sistema em Chip)

ACK Acknowledgement (Aceito)

NACK Not Acknowledgement (Não Aceito)

SEU Single Event Upset (Evento Simples de Inversão)

MAF Maximal Aggressor Fault (Falha de Agressor Máximo)

RG Random Generator (Gerador Randômico)

MG Mask Generator (Gerador de Máscara)

DBP Default Backup Path (Caminho Padrão de Backup)

CRC Cycle Redundancy Check (Verificação Cíclica Redundante)

VHDL Very High Speed Integrated Circuits Hardware Description Language (Linguagem de Descrição de Hardware de Circuitos Integrados de Altíssima Velocidade)

IP Intellectual Property (Propriedade Intelectual)

PHIT Phisical Unit (Unidade Física)

FLIT Flow Control Unit (Unidade com Controle de Fluxo)

FCFS First Come First Served (Primeiro que Chega, Primeiro a ser Servido)

LRS Least Recently Served (Último Recentemente Servido)

RR Round Robin (Volta Contínua)

FIFO First In First Out (Primeiro que Entra é o Primeiro que Sai)

SAFC Statically Allocated, Fully Connected (Estaticamente Alocado, Totalmente Conectado)

SAMQ Statically Allocated, Multi-Queue (Estaticamente Alocado, Múltiplas Filas)

DAMQ Dynamically-Allocated, Multi-Queue (Dinamicamente Alocado, Múltiplas Filas)

CBDA Centrally Buffered, Dynamically Allocated (Centralmente Armazenado,

Dinamicamente Alocado)

DSP Digital Signal Processor (Processador Digital de Sinais)

HOL Head of Line (Cabeça da Fila)

FPGA Field Programmable Gate Array (Arranjo de Portas Programável em Campo)

1 Introdução

Com a melhoria contínua dos processos submicrônicos de fabricação de circuitos integrados, possibilitando uma densidade cada vez maior de transistores em uma mesma pastilha de silício, vem-se cada vez mais falando no sistema-em-chip, ou SoC (do inglês, *System on Chip*). Esses sistemas – que já integram dezenas (e já centenas) de processadores, controladores e memórias em uma única pastilha – tornaram-se cada vez mais complexos e a comunicação entre suas várias partes tornou-se um ponto crítico do projeto.

O reuso tornou-se uma palavra-chave nesse contexto, pois é uma forma de atender mais rapidamente as pressões do mercado por novas tecnologias, diminuindo o *time-to-market* e também o custo final do produto. Assim, a arquitetura de comunicação é um ponto chave do projeto de um SoC, na medida em que são utilizados, cada vez mais, esses componentes pré-projetados e pré-verificados, denominados núcleos que devem comunicar-se entre si.

As arquiteturas de comunicação entre os núcleos desses sistemas dividem-se em duas: canais ponto-a-ponto e canais multiponto. A primeira oferece melhor desempenho, contudo é uma alternativa mais cara, pois necessita de um projeto específico, tornando sua reusabilidade limitada. A segunda, também conhecida como barramento, tem alto potencial de reusabilidade, porém tem seu desempenho degradado à medida que o número de núcleos interconectados pelo mesmo aumenta (devido ao aumento da capacitância parasita introduzida pelos circuitos conectados), sem falar da potência, que aumenta pela difusão de cada sinal a todos os pontos do barramento. Assim, não é recomendada a utilização de barramento como arquitetura de comunicação em sistemas que necessitem de alta escalabilidade (ZEFERINO, 2003a).

Nesse contexto, as redes de interconexão chaveada (como aquelas encontradas em computadores paralelos) tornaram-se a alternativa mais viável de arquitetura de comunicação dos SoCs. A aplicação desse conceito nesse novo mundo, levou ao termo rede-em-chip, ou NoC (do inglês, *Network on Chip*).

1.1 Motivação

Pela importância no projeto final de um SoC, as redes-em-chip tornaram-se objeto de muito estudo. Aumentar desempenho e confiabilidade, diminuir o consumo de potência, na medida em que a freqüência de operação dos circuitos aumenta, tornou-se um desafio aos projetistas responsáveis por essa parte crucial do sistema.

Nesse contexto, ter um sistema confiável e disponível implica ter uma arquitetura de comunicação confiável. Prover mecanismos de detecção/recuperação de falhas no sistema de comunicação é essencial para a confiabilidade e disponibilidade geral do circuito final. Essa não é mais uma opção, passou a ser uma exigência, na medida em que a taxa de falhas dos sistemas mais novos, desenvolvidos em escala nanométrica (ainda em freqüências altíssimas), já está na ordem de 10^{-1} – muito acima dos 10^{-9} da convencional tecnologia CMOS (NIKOLIC, 2001).

São esses assuntos que o presente trabalho se propõe a explorar e melhorar.

1.2 Revisão Bibliográfica

Esta seção tem como objetivo revisar os trabalhos correlatos às áreas dos estudos realizados nessa monografia.

1.2.1 Pesquisas em SoC

Os estudos e pesquisas sobre sistema-em-chip ou, simplesmente, SoC (do inglês, *System on Chip*) está em evidência hoje, bem como na última década. Esse interesse emergiu no início dos anos 2000, quando o paradigma de evolução da microeletrônica saiu um pouco do aumento da freqüência de operação dos circuitos e deslocou-se para o paralelismo real de processamento – tendo mais uma vez a natureza como fonte inspiradora para evolução, já que é sabido que o paralelismo é o segredo do poder do cérebro humano. A Figura 1 ilustra, quantitativamente, essa evolução que se manteve adormecida até o início dos anos 2000, quando a partir daí tomou forma exponencial.

Com esse novo paradigma, as arquiteturas de comunicação entre esses núcleos tornaram-se objeto de muito estudo. Dentre as mais utilizadas estão: as arquiteturas ponto-a-ponto e barramento. A Figura 2 ilustra cada uma delas.

Devido as limitações dessas arquiteturas surgiu o conceito de NoC, que pode ser definida como uma rede de interconexão chaveada dentro do chip. Com os mesmos conceitos de redes de computadores paralelos, logo surgiram as mesmas dificuldades enfrentadas nestas. Uma comparação entre as principais características dessas arquiteturas, revelando seus pontos fortes e fracos, é feita na Tabela 1. A análise atenta dessa tabela mostra, claramente, a superioridade da tecnologia de redes-em-chip: sobre o barramento, pelas características de paralelismo, consumo e escalabilidade; e sobre a arquitetura ponto-a-ponto, pelas características de escalabilidade e reusabilidade.

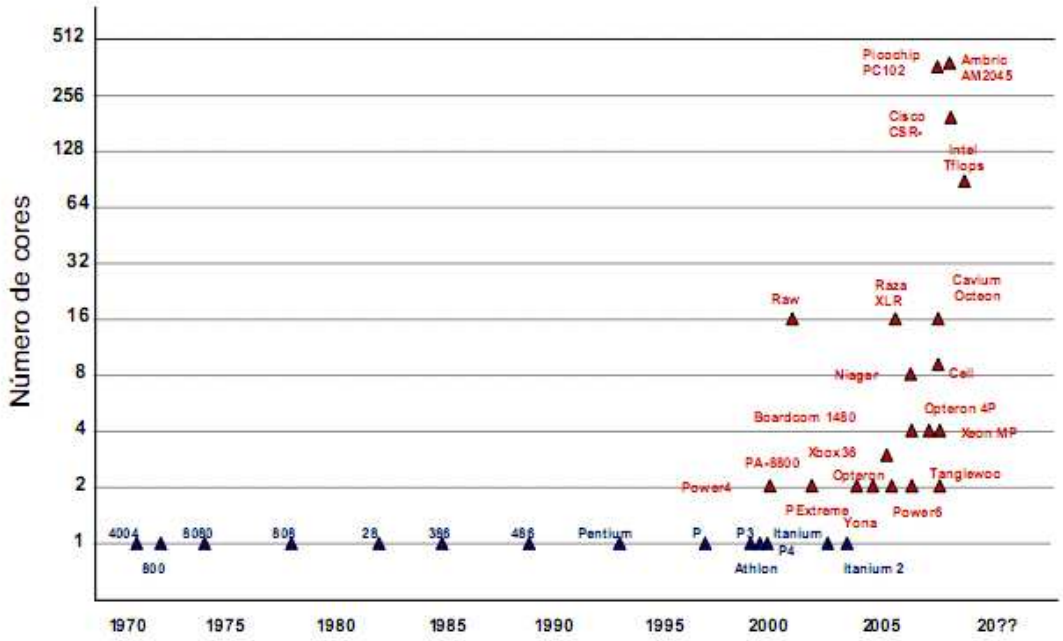

Figura 1 - Evolução do número de *cores* ao longo dos anos (GORDON, 2006).

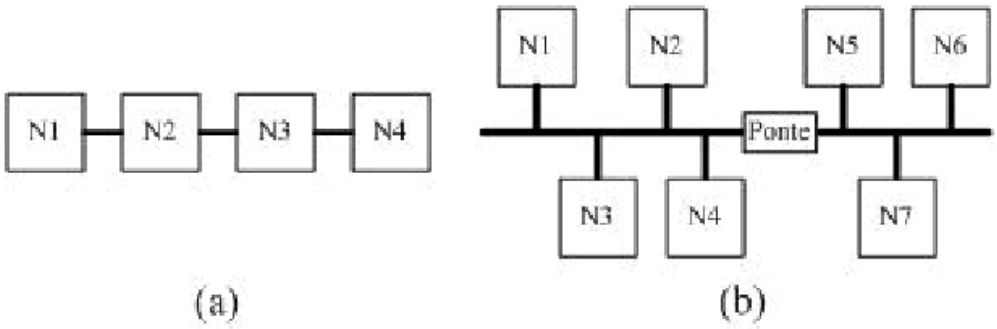

Figura 2 - Arquiteturas de comunicação mais utilizadas: (a) ponto-a-ponto e (b) barramento (CARDOZO, 2005).

	Paralelismo	Consumo	Escalabilidade	Reusabilidade
Arquitetura ponto-a-ponto	Possui paralelismo	Baixo consumo em relação ao barramento	Não oferece escalabilidade	Reuso restrito
Barramento	Não possui paralelismo	Maior consumo de energia	Escalabilidade limitada	Reutilizável
NoC	Possui paralelismo	Baixo consumo em relação ao barramento	Ilimitada, basta acrescentar mais roteadores	Reutilizável

Tabela 1 - Comparativo entre arquitetura de comunicação intrachip (CARDOZO, 2005).

Um exemplo de um sistema-em-chip utilizando uma arquitetura de comunicação do tipo NoC, com topologia grelha (ou *mesh*, do inglês) é mostrado na Figura 3.

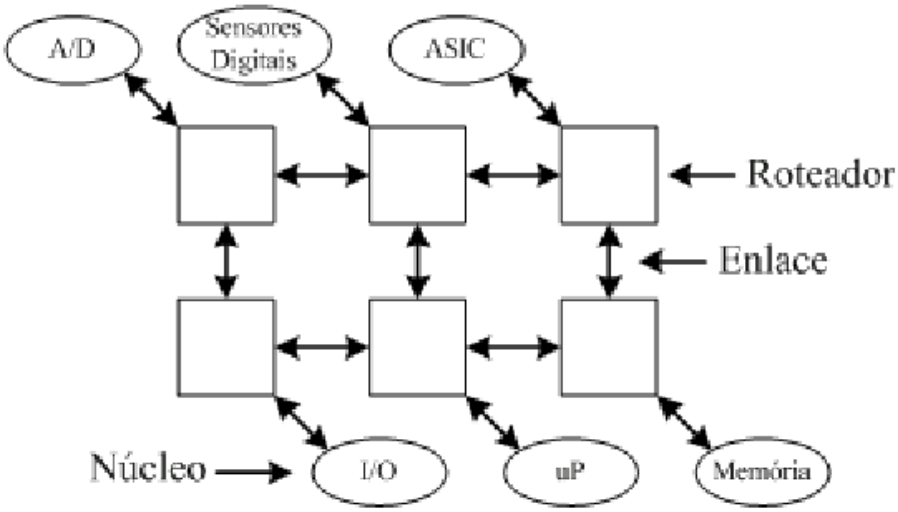

Figura 3 - Exemplo de SoC utilizando arquitetura de comunicação baseada em NoC (CARDOZO, 2005).

Nesse tipo de arquitetura de comunicação, trabalha-se geralmente com a troca de mensagens. Um dispositivo conectado à rede, caso necessite de uma operação de um dispositivo remoto, envia uma mensagem para o mesmo requisitando a operação.

Na próxima seção é feita uma revisão dos trabalhos relacionados às redes-em-chip.

1.2.2 Pesquisas em NoC

Extrapolando os conceitos do mundo das redes de computadores, surgiram as redes-em-chip. Dotada de roteadores e enlaces, que interligam os núcleos de um sistema computacional. Tem como ponto forte a eliminação da necessidade de sincronização global de todo o sistema, tornando-o localmente síncrono, porém globalmente assíncrono (*globally asynchronous locally synchronous* – GALS) (HEMAMI, 1999). Permite ainda o funcionamento dos diferentes núcleos com diferentes relógios.

A topologia da rede é determinada pela forma como os roteadores estão ligados entre si. As mais utilizadas tem sido a grelha (ou *mesh*, do inglês) e a torus. A Figura 4 ilustra a conexão entre os elementos da rede de cada topologia.

Conceitos que permeiam essa área de conhecimento são os mesmos aplicados às redes chaveadas de computadores. Latência, largura de banda, congestionamento, controle de fluxo, roteamento, arbitragem, chaveamento e memorização são conceitos importantes no projeto de uma rede-em-chip e também na medição/análise de seu desempenho.

O projeto ATLAS, da PUCRS, foi um dos pioneiros, no Brasil, no estudo das características e problemas das redes-em-chip. Neste projeto se desenvolveu a NoC Hermes (MORAES, 2003) e também um framework para parametrização e avaliação de desempenho

da mesma. Essa rede-em-chip é utilizada nesse trabalho como estudo de caso para implementação das técnicas propostas.

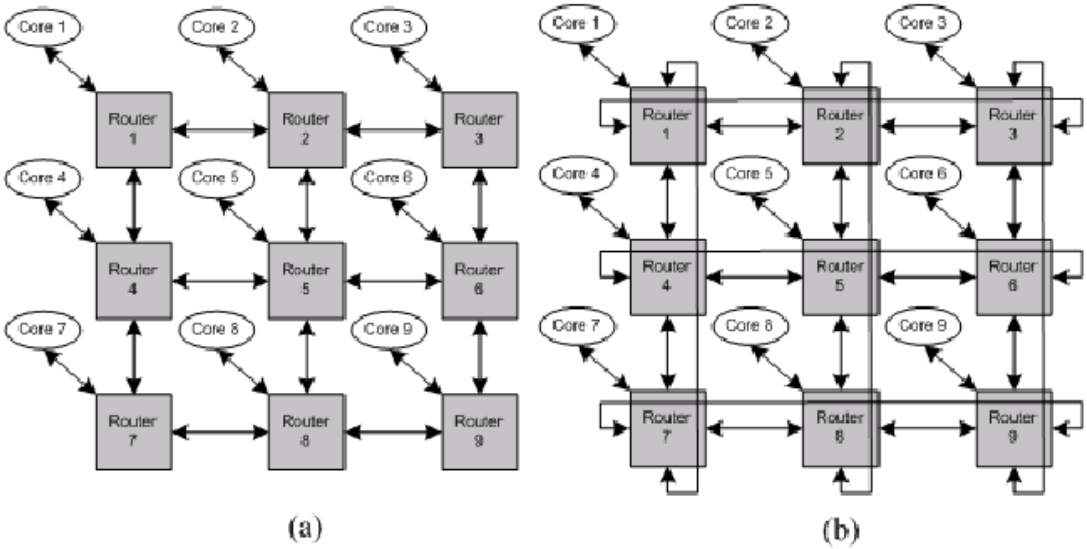

Figura 4 - Topologias mais utilizadas atualmente: (a) grelha e (b) torus (CARDOZO, 2005).

Outras redes também muito citadas na bibliografia são: SPIN (*Scalable Programmable Integrated Network*) (GUERRIER, 2000); aSoC (*Adaptative SoC*) (LIANG, 2000). No Brasil, existe ainda a SoCIN (ZEFERINO, 2003b).

1.2.3 Pesquisas sobre Tolerância a Falhas em NoC

Um modelo de falhas baseado em propriedades genéricas das funcionalidades do roteador de uma NoC foi proposta por (BENGTSSON, 2006). Esse modelo consiste de cinco tipos de falhas.

a) Falha de dado perdido: o roteador recebe o dado, mas nunca o envia à porta de destino.

b) Falha de dado corrompido: o roteador corrompe o dado e o envia à porta de destino.

c) Falha de roteamento: o roteador recebe o dado, mas o envia a uma porta errada.

d) Falha de múltiplas cópias no espaço: o roteador envia o pacote à porta correta, mas também às outras portas.

e) Falha de múltiplas cópias no tempo: o roteador envia o pacote à porta correta, mas várias vezes.

Pullini et al. (2005) verificaram o sobre custo de prover tolerância a falhas transitórias (*crosstalk*, ruído de fonte de alimentação, interferência eletromagnética e *soft-errors*) no

mecanismo de controle de fluxo de uma NoC. Os autores avaliaram três estratégias de controle de fluxo utilizadas em enlaces longos que requerem repetidores: STALL/GO; T-Error; e ACK/NACK. Com base nesses resultados, os autores concluíram que STALL/GO possui um sobre custo menor e um melhor desempenho, porém é mais suscetível a falhas. O T-Error pode ser implementado para melhorar o desempenho do enlace ou para melhorar a confiabilidade do sistema, porém não traz benefícios para o mesmo quando o número de estágios de *pipeline* é reduzido. A técnica de ACK/NACK oferece um suporte mais extensivo para manipulação de falhas, mas apresenta um maior sobre custo de potência e de área. Em outro cenário, quando o enlace é curto ou os congestionamentos são relativamente baixos, o aumento da latência percebida na aplicação tornou-se insignificante quando usado o ACK/NACK.

Frantz et al. (2006) avaliaram os efeitos e os impactos sobre falhas decorrentes do processo de fabricação de componentes em NoCs, do aumento da freqüência de operação do circuito e da radiação presente no espaço do ambiente no serviço de chaveamento. Os autores desenvolveram um mecanismo de injeção de falhas do tipo erro transitório de bit (para falhas de crosstalk) e de *soft-errors* (para falhas do tipo SEU – *Single Event Upset*). Um mecanismo simula os efeitos de SEU em componentes relacionados com a memória. Um segundo mecanismo simula os efeitos de crosstalk no enlace utilizando o modelo *Maximal Aggressor Fault* (MAF). Para avaliação, foi implementado um mecanismo de injeção de falhas constituído de dois blocos: Gerador Randômico (RG – *Random Generator*), que produz padrões pseudo-aleatórios que são decodificados pelo bloco Gerador de Máscara (MG – *Mask Generator*), responsável por gerar todos os sinais de falha habilitados. Com os resultados obtidos, os autores concluíram que a ocorrência de tais falhas pode afetar o funcionamento correto de uma NoC. No estudo, os autores não apresentaram nenhuma solução para prover tolerância a falhas.

Koibuchi et al. (2008) propuseram um mecanismo leve tolerante a falhas baseado em um caminho padrão de backup (DBP – *Default Backup Path*) para prover alta confiabilidade em uma NoC. A fim de manter a conectividade da rede, o mecanismo implementado fornece um caminho alternativo como backup entre determinadas portas do roteador e dos núcleos para contornar uma falha permanente no *crossbar* do roteador. Os experimentos de simulação realizados mostraram que o mecanismo assegura uma baixa degradação no desempenho da rede na medida em que aumenta o número de falhas nos roteadores.

Braga, Cota e Lubaszewski (2009) trataram as falhas ocorridas nos enlaces em uma arquitetura de NoC utilizando técnicas de codificação e retransmissão de dados para a

detecção e/ou correção de erros. Foram abordadas três técnicas de tolerância a falhas aplicadas no canal de comunicação: paridade única; retransmissão única; e transmissão utilizando codificação de Hamming. Os autores avaliaram o impacto das três técnicas no sobre custo de fios do canal do enlace e no atraso de propagação de sinais pelo enlace.

Lucas e Moraes (2009) propuseram um mecanismo de recuperação de erros devido a efeitos de *crosstalk* em nível de *flits* para aumentar a confiabilidade de uma NoC. Os autores trataram de falhas de *crosstalk* nos enlaces entre roteadores utilizando um mecanismo de codificação CRC (*Cycle Redundancy Check*) para detecção de falhas, retransmitindo *flits* quando da ocorrência de uma falha no próximo ciclo de clock. Para avaliação de desempenho, foi utilizado um simulador SystemC/VHDL da rede Hermes e os resultados mostraram que, num cenário de pior caso, o sobre custo na latência foi de 17%.

Nesse trabalho, é abordado apenas o primeiro tipo de falha (dados perdidos). O terceiro (falha de roteamento) é comentado no Capítulo 3, mas como efeito colateral do primeiro.

1.3 Objetivos

O objetivo geral desta monografia e os objetivos específicos são apresentados nesta seção.

1.3.1 Objetivo Geral

O principal objetivo desta monografia é propor e analisar o desempenho de um mecanismo de detecção/recuperação de falhas, tanto transitórias como permanentes, nos roteadores da NoC Hermes. Prezando pela simplicidade, foi escolhida uma técnica de detecção temporal de falha: *watch dog timer*. Para a recuperação, são utilizados: o reset do roteador em falha (útil no caso de falhas transitórias); ou curto-circuito dos enlaces (útil no caso de falhas permanentes).

A principal contribuição desse trabalho é a implementação das técnicas propostas na NoC Hermes e apresentar os impactos causados no sistema final.

1.3.2 Objetivos Específicos

Os objetivos específicos desta monografia estão listados a seguir.

a) Listar problemas causados por falhas transitórias e/ou permanentes no funcionamento da NoC Hermes;

b) Propor soluções que melhorem o desempenho da NoC Hermes frente a essas falhas;

c) Implementar as técnicas propostas;

d) Avaliar o desempenho das alterações, comparando os resultados antes e depois;

e) Propor trabalhos futuros para dar continuidade nas melhorias da NoC Hermes.

1.4 Organização da Monografia

Os estudos deste trabalho estão organizados da seguinte forma. No Capítulo 2 é feita uma breve apresentação dos conceitos envolvidos nos estudos de redes-em-chip e tolerância à falha. No Capítulo 3, já de posse dos conhecimentos básicos necessários, é feito um estudo das principais características da NoC Hermes, bem como de suas fragilidades, frente a falhas transitórias e permanentes. No Capítulo 4 é explicada a técnica utilizada para prover maior robustez à rede Hermes, resultando assim na NoC proposta, a HermesFT. No Capítulo 5 são apresentados e discutidos os resultados das simulações, em diversos cenários de falhas, realizadas com as duas redes-em-chip, para efeito comparativo. E finalmente, no Capítulo 6 são feitas as conclusões finais e apresentadas perspectivas para trabalhos posteriores.

2 Conceitos de Redes Intrachip e de Tolerância a Falhas

Este Capítulo apresenta os principais conceitos relacionados às redes intrachip e tolerância a falhas. Redes intrachip são arquiteturas de comunicação e assim, para o total entendimento, é necessária a familiaridade com uma quantidade de conceitos fundamentais de redes de comunicação. Dentre estes conceitos, destacam-se as estratégias de chaveamento, os algoritmos de roteamento, os métodos de armazenamento temporário de mensagens, entre outros.

Uma rede-em-chip pode ser definida como uma infra-estrutura que permite interconectar os núcleos de um sistema. Os núcleos podem ser processadores, memórias, controladores de E/S, ou mesmo um *IP Core* (do inglês, *Intellectual Property Core*) específico da aplicação. As duas principais figuras que compõem uma NoC são os roteadores e os enlaces.

2.1 Canais e Roteadores

Os enlaces são canais unidirecionais ponto-a-ponto que interligam um roteador a outro, ou a um núcleo. Para tornar possível a comunicação bidirecional simultânea, esses canais aparecem aos pares em cada roteador/núcleo, formando o que se chama enlace. O enlace é formado pela palavra de dados, pelos sinais de controle de fluxo, de sinalização de erro, paridade e enquadramento de mensagens. Chama-se *phit* (do inglês, *phisical unit*) o conjunto formado pela palavra de dados e pelos sinais de sinalização de erro, paridade e enquadramento de mensagens. A Figura 5 ilustra o que seria um phit.

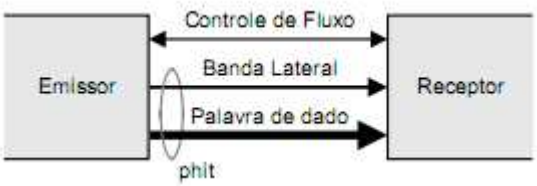

Figura 5 - Representação do enlace de saída de um emissor ao seu receptor.

O roteador é o conjunto de mecanismos que determinam o caminho da mensagem na rede e é constituído por um *crossbar*, lógicas de roteamento e chaveamento e das portas de comunicação que se conectam aos enlaces da rede. É a dispersão espacial desses elementos que define a topologia da rede.

A Figura 6 ilustra as entidades em uma rede-em-chip.

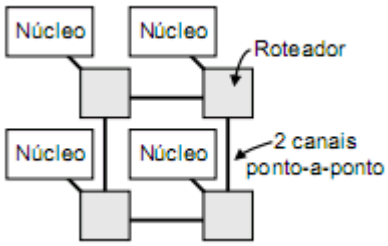

Figura 6 - Exemplo de dispersão espacial (grelha) de uma rede-em-chip.

Em uma visão mais detalhada, a Figura 7 ilustra os elementos de um roteador.

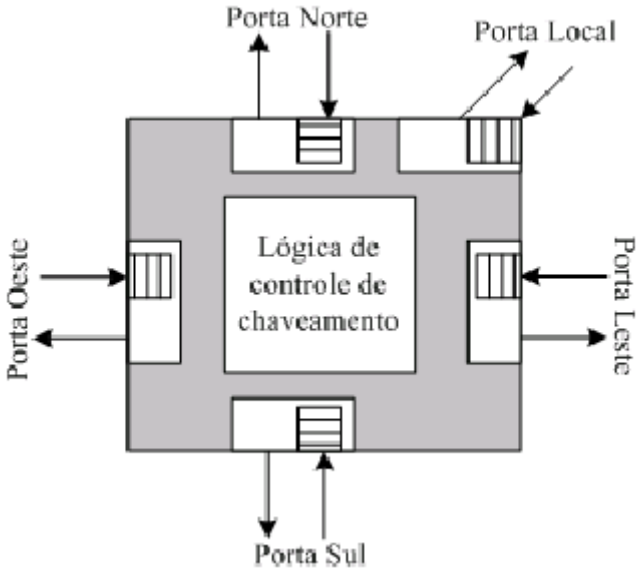

Figura 7 - Visão detalhada dos elementos internos do roteador de uma rede-em-chip (CARDOZO, 2005).

Para reger esses elementos da rede utiliza-se a metodologia de camadas, onde cada camada utiliza funções da camada inferior e provê funções de mais alto nível à camada superior. No modelo OSI são definidas sete camadas, das quais somente três são implementadas atualmente no universo das redes-em-chip, são elas: física, enlace e rede. A Figura 8 ilustra as camadas do modelo OSI, bem como suas principais funções.

A camada física é responsável por prover o meio físico e elétrico para a interconexão dos elementos da rede. Sendo assim, é composta pelos sinais utilizados pelos roteadores/núcleos para comunicar-se com seus vizinhos.

A camada de enlace é responsável por prover uma conexão lógica entre duas entidades da rede, tornando ainda confiável o meio físico não confiável. Para tanto, técnicas de controle de fluxo e detecção de erro são comumente utilizadas.

Figura 8 - Modelo de camadas OSI.

A camada de rede é responsável pela troca de pacotes entre as entidades da rede, desmontando-os, no remetente, para transmissão no enlace e montando-os novamente no destinatário.

A metodologia utilizada no projeto de um SoC que utiliza uma NoC consiste em duas fases (KUMAR, 2002). Na primeira fase, além da topologia da rede a ser utilizada, é definido também o número de roteadores, bem como os mecanismos de comunicação utilizados pelos mesmos para implementar as funcionalidades das camadas de comunicação do modelo utilizado.

A seleção dos mecanismos de comunicação e da topologia são decisões tomadas pelos projetistas levando em consideração fatores como relação custo/desempenho, escalabilidade e requisitos de performance exigida pela aplicação.

As topologias e mecanismos de comunicação utilizados no projeto de uma rede-em-chip são um subconjunto dos utilizados em uma rede de interconexão chaveada utilizada em computadores paralelos. Nesta seção seguinte são evidenciadas as principais alternativas utilizadas hoje no projeto de uma NoC.

2.2 Topologias e Mecanismos de Comunicação

Esta seção está subdividida em duas: a seção de topologias, que cita algumas das principais e mais utilizadas representantes, bem como suas características, pontos fortes e deficiências; e a seção de Mecanismos de Comunicação, que se prenderá aos mecanismos utilizados nesse trabalho.

2.2.1 Topologias de Rede de Interconexão

Uma rede de interconexão pode ser caracterizada pela estrutura de interligação de seus nodos. Essa estrutura pode ser estruturalmente representada por um grafo G(N,C) onde *N* representa o conjunto de nodos (de processamento e/ou de chaveamento) da rede e *C* representa o conjunto de canais de comunicação. Quanto a topologia, as redes de interconexão para multiprocessadores podem ser agrupadas em duas classes principais, as redes diretas e as redes indiretas (ZEFERINO, 2003a).

Nas redes *diretas*, cada nodo de chaveamento possui um nodo de processamento associado, e esse par pode ser visto como um elemento único dentro da máquina, tipicamente referenciado pela palavra nodo. Pelo fato de utilizarem nodos de chaveamento tipo roteador, as redes diretas são também chamadas de redes baseadas em roteadores (DUATO, 1997). A Figura 9 ilustra um exemplo de rede direta e destaca um de seus nodos.

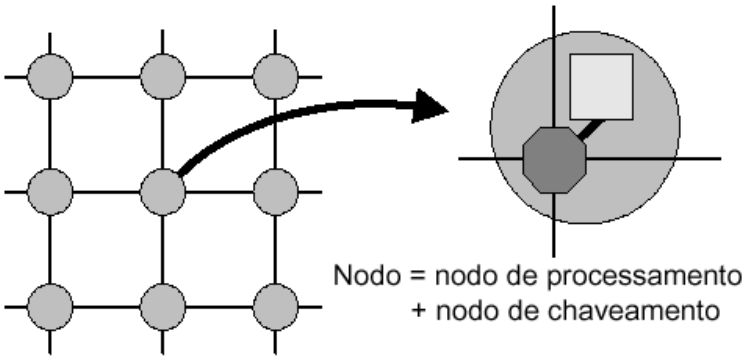

Figura 9 - Exemplo de rede direta, contendo um roteador para cada nodo do sistema.

As topologias de redes diretas estritamente ortogonais mais utilizadas são a grelha (ou malha) n-dimensional, o toróide e o hipercubo. A Figura 10 exemplifica cada uma dessas topologias.

Nas redes *indiretas*, o acoplamento entre os nodos de processamento e os nodos de chaveamento não ocorre no mesmo nível das redes diretas. Na visão unificada cada par nodo de processamento - nodo de chaveamento é visto como um elemento. Os nodos de processamento possuem uma interface para uma rede de nodos de chaveamento baseados em chaves. Cada chave possui um conjunto de portas bidirecionais para ligações com outras chaves e/ou com os nodos de processamento. Somente algumas chaves possuem conexões para nodos de processamento e apenas essas podem servir de fonte ou destino de uma mensagem. A topologia da rede é definida pela estrutura de interconexão dessas chaves.

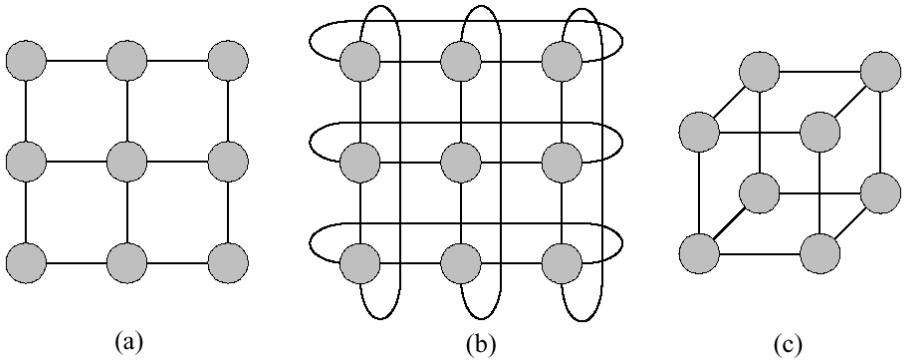

Figura 10 - Topologias mais utilizadas atualmente: (a) grelha 2D, (b) torus e (c) cubo (ZEFERINO, 2003b).

Duas topologias clássicas de redes indiretas se destacam: o *crossbar* e as redes multiestágio. Para conexão indireta de N nodos de processamento, o *crossbar* (Figura 11) é a topologia ideal, pois consiste de uma única chave $N \times N$ capaz de ligar qualquer entrada com qualquer saída. Embora seja mais econômico que uma rede direta completamente conectada (a qual necessitaria de N roteadores, cada um com um crossbar $N \times N$ interno), o crossbar possui uma complexidade $O(N^2)$, o que torna o seu custo proibitivo para redes grandes (ZEFERINO, 2003a). A Figura 11 ilustra uma rede *crossbar* de 4 x 4.

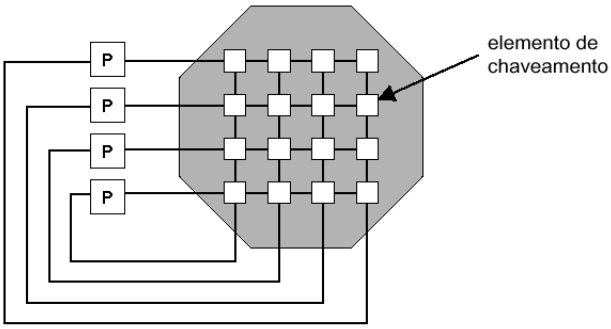

Figura 11 - Rede crossbar 4 x 4.

2.2.2 Mecanismos de Comunicação

Dos vários conceitos importantes de redes de comunicação, serão revisados nessa seção os de: algoritmos de roteamento, estratégias de chaveamento, métodos de armazenamento, controle de fluxo e arbitragem.

2.2.2.1 Roteamento: Classificações e Problemas

O primeiro conceito importante de mecanismos de comunicação é o algoritmo de roteamento. Usado para definir o caminho que um pacote deve utilizar a partir de um nodo origem até um nodo destino. Tais algoritmos podem ser classificados de acordo com diversos critérios. A Figura 12 ilustra algumas das diferentes classificações desses algoritmos, baseado nos critérios que a definem.

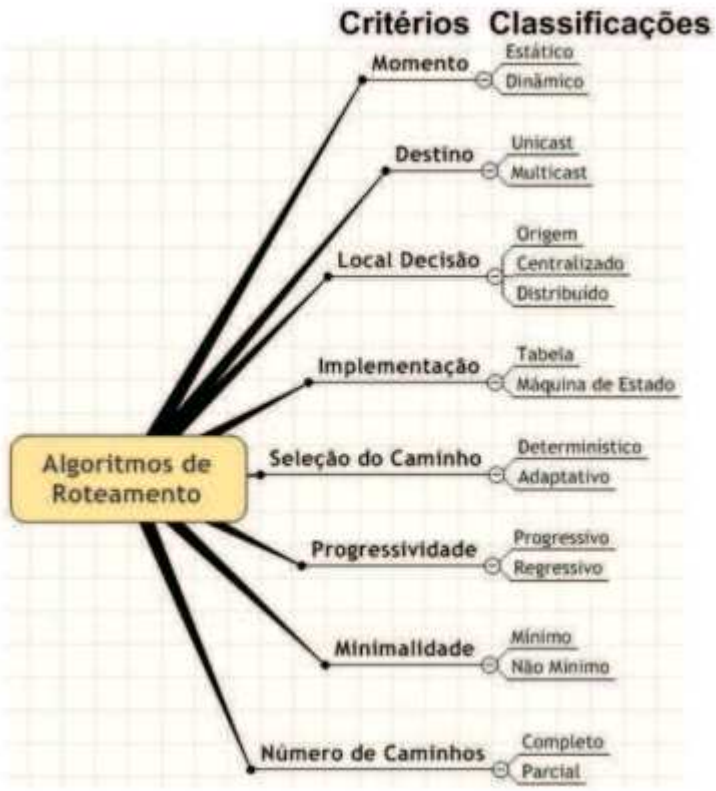

Figura 12 - Classificações possíveis quanto ao algoritmo de roteamento utilizado (BEZERRA, 2009).

Os critérios de classificação ilustrados na Figura 12 são descritos nos tópicos abaixo.

a) Quanto ao momento de realização do roteamento – algoritmos podem ser classificados segundo este critério em: estáticos, quando definidos em tempo de projeto da rede; ou dinâmicos, quando definidos durante a operação do sistema.

b) Quanto à quantidade de destinos das mensagens – algoritmos podem ser classificados segundo este critério em: *unicast*, quando existe apenas um destino; ou *multicast* quando, para pelo menos um pacote, existam pelo menos dois destinos. Um caso especial importante de algoritmo *multicast* são os algoritmos *broadcast*, onde um pacote gerado em algum ponto da rede é enviado para todos os demais pontos da rede.

c) Quanto ao local onde a decisão de roteamento é tomada – algoritmos podem ser classificados segundo este critério em: centralizado, quando um nodo define o caminho de todos os pacotes; ou origem quando o caminho ao longo da rede é definido na origem do pacote, fora da rede. O último tipo são os algoritmos distribuídos, quando o caminho do pacote é definido a cada roteador por onde este passa no seu percurso ao destino.

d) Quanto à forma de implementação – algoritmos podem ser classificados segundo este critério em: baseados em tabelas, quando o trajeto é definido a partir de consultas a tabelas que indicam o caminho ou parte deste; ou baseados em máquinas de estado, quando o roteamento é implementado usando um algoritmo (BARTIC, 2005).

e) Quanto ao processo de seleção do caminho – algoritmos podem ser classificados segundo este critério em: determinísticos quando, definidas origem e destino, o trajeto a ser realizado é sempre o mesmo. Outro tipo são os algoritmos adaptativos, onde o caminho da origem ao destino é determinado em tempo de execução, determinado pelas condições de tráfego, buscando o melhor caminho no instante do roteamento.

f) Quanto à progressividade – algoritmos podem ser classificados segundo este critério em: progressivos, se o pacote sempre avança reservando canais; ou regressivo, caso o mesmo possa liberar canais reservados previamente.

g) Quanto à minimalidade – algoritmos podem ser classificados segundo este critério em: mínimos, quando o nodo seguinte visitado por um pacote estiver sempre mais próximo do destino que o nodo anterior; ou não mínimo, quando se permite o afastamento temporário de um pacote do seu destino.

h) Quanto ao número de caminhos – algoritmos podem ser classificados segundo este critério em: completos, quando todos os caminhos possíveis de serem usados pelo pacote para ir de um nodo fonte a um nodo destino podem ser empregados pelo algoritmo; ou parciais, quando a quantidade de caminhos entre a fonte e um destino é limitada pelo algoritmo usado.

2.2.2.2 Métodos, Modos e Algoritmos de Chaveamento

A escolha de como se transfere pacotes entre origem e destino é o que determina o comportamento da lógica de chaveamento interna de cada roteador. É possível discernir dois

métodos de transferência, denominados chaveamento de circuito e chaveamento de pacotes, definidos a seguir.

No chaveamento de circuito (do inglês, *circuit switching*), o trajeto do nodo origem ao destino é inicialmente determinado e reservado, criando-se uma conexão (lógica e/ou física) entre origem e destino. Somente depois de estabelecida a conexão, inicia-se o envio da mensagem.

No chaveamento de pacotes (do inglês, *packet switching*), a mensagem é particionada ou agregada em pacotes, que são transmitidos sem uma determinação prévia do trajeto do nodo de origem ao destino. Isto, porém implica a escolha de um dentre três modos de chaveamento que estão listados logo abaixo.

a) *Store-and-forward*: Este modo se assume que cada pacote é armazenado inteiramente em um roteador antes de seguir caminho para o próximo roteador;

b) *Virtual-cut-through*: Neste modo, o roteador pode enviar um pacote adiante desde que o próximo roteador garanta a viabilidade de recebê-lo completamente;

c) *Wormhole*: Neste modo, bastante usual em redes intrachip, pacotes são divididos em partes menores denominadas *flits*, que constituem a unidade fundamental de controle de fluxo na rede. Cada *flit* pode ser enviado separadamente entre roteadores. Tipicamente, o primeiro *flit* constitui-se de um cabeçalho que sucessivamente reserva canais por onde todos os demais *flits* do pacote seguirão.

2.2.2.3 Exemplo de Algoritmo de Roteamento

Um algoritmo de roteamento simples e bem conhecido é denominado de roteamento XY (DUATO, 1997). Este é um algoritmo determinístico de operação mais simples em redes como malha 2D. Ele é naturalmente livre de *deadlock* e *livelock*.

A vantagem deste algoritmo em redes com topologia malha é que ele garante a liberdade de *deadlock* a um baixo custo de implementação. No entanto, ele restringe fortemente a forma de utilização dos recursos da rede. Essa restrição provém do fato do pacote ter que percorrer totalmente as abscissas até chegar à coluna da ordenada em que se situa o nodo destino, seguindo então por caminhos verticais até o destino.

A Figura 13 ilustra o trajeto de um pacote com quatro *flits* que penetra uma rede 3x3 com topologia malha 2D pelo roteador do canto inferior esquerdo e tem como destino o roteador superior central.

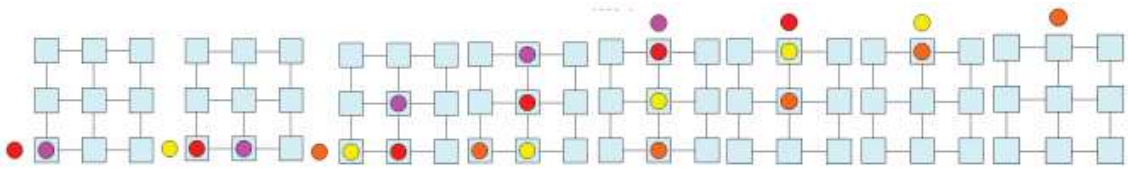

Figura 13 - Exemplo da trajetória dos *flits* de um pacote em uma rede malha 2D 3 x 3.

2.2.2.4 Controle de Fluxo

Quando algum recurso está alocado para um pacote de forma a impedir a sua alocação a outro pacote, ocorre um bloqueio de recursos. Nesta situação, é necessário que exista uma política determinando a melhor maneira de lidar com o pacote que não pode ser atendido. Dentre as políticas mais usadas destacam-se as listadas abaixo.

a) O descarte de pacotes que não têm acesso ao recurso necessário;

b) O bloqueio e armazenamento de pacote;

c) O desvio do caminho seguido pelo pacote.

Essa determinação é a política do controle de fluxo, definindo quem aloca os canais e as filas no momento em que o pacote trafega na rede (NI, 1993).

As estratégias de controle de fluxo utilizadas aqui pressupõem o bloqueio e armazenamento de pacotes. Neste sentido, duas estratégias são usadas para realizar o controle de fluxo: baseado em créditos e *handshake*. Abaixo estão as principais características de cada uma.

a) Controle de fluxo baseado em créditos – não deixa que pacotes sejam descartados, pois uma transmissão entre roteadores só inicia após a verificação junto ao receptor que os dados serão recebidos. Essa verificação é realizada da seguinte forma: o receptor envia ao transmissor um sinal confirmando a existência de créditos que o último possui para envio dos dados. Com esta informação, o transmissor envia dados somente quando existir crédito para o envio. Ao longo do processo de envio de dados do transmissor pode ou não ocorrer mudança no seu estado de créditos junto ao receptor, dependendo do espaço de armazenamento disponível no último.

b) Controle de fluxo *handshake* – o transmissor informa ao receptor a intenção de enviar um dado através de um sinal de solicitação. O receptor recebendo o sinal de requisição verifica a existência de espaço para recebimento. Existindo, o dado é lido e armazenado, e o receptor envia o sinal de reconhecimento de recepção (*ack*) ao transmissor. Na ausência desta possibilidade de recepção, o receptor pode enviar um

sinal de não reconhecimento (*nack*), fazendo com que o roteador transmissor necessite retransmitir o dado até o recebimento de um *ack*.

Uma observação quanto ao uso do controle de fluxo *handshake* é a perda de desempenho, devido ao tempo que os dados ficam armazenados nas filas até receberem o sinal de *ack*, ou quando não existe espaço em fila já que se torna necessário reenviar o dado (BEZERRA, 2009).

2.2.2.5 Arbitragem

O fato de ocorrerem conflitos de requisições simultâneas requer a determinação de procedimentos para resolver, através do gerenciamento de acesso a recursos compartilhados, o uso dos componentes de rede pelas mensagens que nesta trafegam. Por exemplo, ao receber requisições de roteamento simultâneas, um elemento de arbitragem pode atribuir prioridade a portas de um roteador e com base nestas encaminha pacotes para uma unidade única de roteamento conforme esta escala de prioridades.

À arbitragem está relacionado ao problema de *starvation*, que pode ser amenizado e até mesmo resolvido de acordo com o critério de arbitragem adotado. Existem diversas políticas propostas, tais como prioridade estática, prioridade dinâmica, escalonamento por idade, FCFS (*First Come First Served*), LRS (*Least Recently Served*) e RR (*Round-Robin*). As vantagens e inconvenientes de cada uma destas são amplamente discutidos na literatura. Ver por exemplo (SILBERSCHATZ, 1997) para uma discussão mais detalhada.

2.2.2.6 Memorização

Nas redes que utilizam chaveamento por pacotes do tipo *wormhole*, os roteadores devem armazenar os *flits* dos pacotes com destino às saídas que estejam indisponíveis, realizando o controle de fluxo para evitar a perda de dados. Para oferecer essa característica é necessário usar um esquema de armazenamento temporário dos pacotes bloqueados no roteador. Tal armazenamento, conhecido pelo termo técnico “bufferização” será citado no restante do trabalho como memorização. Claramente não é possível disponibilizar uma capacidade de armazenamento infinita para garantir armazenamento na rede de todo e qualquer pacote em qualquer situação. Assim um processo crítico é o de dimensionar os meios de armazenamento temporário dentro dos roteadores, de forma a reduzir tanto a perda de desempenho como o desperdício de área de silício.

Existem três opções básicas de memorização, listadas abaixo com seus prós e contras.

a) Memorização na entrada – a memorização temporária de dados na entrada pressupõe a existência de filas independentes em cada uma das portas de entrada do roteador. As filas podem ser implementadas de várias formas, destacando-se as estratégias FIFO (*First In First Out*), SAFC (*Statically Allocated, Fully Connected*), SAMQ (*Statically Allocated, Multi-Queue*) e DAMQ (*Dynamically-Allocated, Multi-Queue*) (CARARA, 2007). A estratégia FIFO é usada, por exemplo, na NoC Hermes (MORAES, 2004).

b) Memorização na saída – a memorização temporária de dados na saída implica a inserção de filas nas portas de saída do roteador. O problema desta estratégia é que cada fila deve ser capaz de receber simultaneamente dados das N entradas, implicando que a fila de saída possua N portas de entrada ou que opere a uma velocidade N vezes maior do que as entradas. O uso de armazenamento temporário de saída exige a implementação de um controle de fluxo entre a porta de entrada e de saída, aumentando assim a complexidade do roteador.

c) Memorização centralizada compartilhada – a memorização centralizada compartilhada, denominado CBDA (*Centrally Buffered, Dynamically Allocated*), utiliza filas para armazenamento de pacotes de todas as portas de entrada do roteador. O espaço de memória disponível a ser utilizado é dividido de forma dinâmica entre os pacotes de diferentes entradas. Esta estratégia é usada, por exemplo, na NoC Mercury (BASTOS, 2005).

O armazenamento temporário centralizado compartilhado oferece uma melhor utilização de memória do que aquelas proporcionadas pelas abordagens onde este espaço é prévia e estaticamente alocado a portas de entrada. Segundo Zeferino (ZEFERINO, 2003a), o CBDA deve oferecer no mínimo uma largura de banda igual à soma das larguras de banda de todas as portas, fazendo com que em um roteador NxN, a fila possua 2N portas de acesso, de modo a permitir N acessos simultâneos de leitura e N acessos simultâneos de escrita.

2.2.2.7 Métricas de Desempenho da Rede

Um aspecto importante do mecanismo de comunicação que precisa ser levado em consideração são as métricas de desempenho, cujas principais representantes são a largura de banda, a vazão e a latência.

A largura de banda (do inglês, *bandwidth*) refere-se à taxa máxima com a qual a rede de interconexão pode propagar as informações, uma vez que uma mensagem entra na rede. No cálculo da largura de banda costumam serem contabilizados os bits do cabeçalho, da carga

útil e do terminador da mensagem (caso se aplique), sendo que a unidade de medida utilizada é “bit por segundo” (ou bps).

A vazão (*throughput*) é definida em (DUATO, 1997) como o tráfego máximo aceito pela rede ou, em outras palavras, a quantidade máxima de informação entregue na unidade de tempo. Para que a vazão seja independente do tamanho das mensagens e da rede, seu valor pode ser normalizado, dividindo-o pelo tamanho das mensagens e pelo tamanho da rede.

Já a latência é o tempo envolvido desde o início da transmissão de uma mensagem até o momento em que ela é completamente recebida. É medida em unidades de tempo ou em ciclos de relógio.

2.3 Tolerância a Falhas: Conceitos e Técnicas

Nessa seção serão vistos os principais conceitos de tolerância a falhas, bem como suas fases, classificação e modelos. Em seguida, serão apresentadas as principais técnicas utilizadas para provimento de tolerância a falhas em sistemas distribuídos – já que as redes-em-chip são um sistema desse tipo – e as técnicas já utilizadas em NoCs.

2.3.1 Conceitos Introdutórios

Em muitas áreas, circuitos eletrônicos realizam tarefas críticas e que envolvem risco de vidas humanas, tais como sistemas de controle de vôo, monitoramento de pacientes, sistemas de controle de tráfego aéreo, dentre outros. Outras áreas de aplicação, mesmo sem tal importância, dependem fortemente de circuitos eletrônicos, pois estando esses em estado de falha, podem gerar grandes perdas financeiras ou de oportunidades. Todas estas áreas requerem sistemas altamente confiáveis, dado sua dependência dos mesmos.

Nesse contexto, tem-se o primeiro e principal conceito: a dependabilidade (do inglês, *dependable*) que é a propriedade de um sistema que justifica a confiança posta nos serviços que ele provê, isto é, o quão o seu sistema/serviço/funcionalidade pode seguramente depender deste sistema (AVIZIENIS, 2001). Abaixo estão listados os atributos de um sistema confiável (*dependable*).

a) Disponibilidade: propriedade de um sistema estar pronto para ser utilizado imediatamente;

b) Confiabilidade: propriedade de um sistema continuar em execução contínua sem falhas. Se um sistema fica indisponível por um milissegundo a cada hora, terá uma disponibilidade de 99,9999% mas é altamente não confiável. De forma semelhante, se

nunca falha, mas é desligado duas semanas todo mês de agosto, possui alta confiabilidade, mas apenas 96 % de disponibilidade;

c) Segurança: é a probabilidade de o sistema estar operacional, e exucutar sua função corretamente, ou descontinuar suas funções de forma a não provocar dano a outros sistemas ou pessoas que dele dependam. Pode ser interpretada também como uma medida da capacidade do sistema de se comportar de forma livre de falhas (*fail-safe*);

d) Mantenabilidade: define a facilidade de se reparar sistemas que falharam. Um sistema de fácil manutenção pode também apresentar alta disponibilidade, especialmente se falhas puderem ser detectadas e reparadas automaticamente;

e) Integridade: prevenir o acesso e manipulação não autorizada de informações.

Atrelado ao conceito de dependabilidade estão os conceitos de falha, erro e defeito – considerados impedimentos à dependabilidade. Em sistemas computacionais essas palavras possuem significados distintos. A falha acontece no mundo físico dos sistemas computacionais ou por um erro no algoritmo. Por exemplo: caso um roteador, que conecta um DSP (do inglês, *Digital Signal Processor*) a uma rede-em-chip de um SoC, tenha sua tensão de alimentação atenuada, nesse caso há uma falha que precisa ser corrigida, mas a princípio ela não se transformou em erro ou defeito. Se essa atenuação de tensão fizer com que algum *flipflop* (que é um circuito biestável) saia de um dos seus estados para o outro (digamos, de 1 para 0), a falha se transformou em erro. Se o erro anterior não for tratado e o roteador travar ou alterar alguma informação de um fluxo de dados, por exemplo, o erro se transformou em defeito. Se o usuário (no caso, o núcleo que necessita daquele roteador para que seu pacote chegue ao destino) foi afetado pelo problema, este não é mais erro e sim defeito. A Figura 14 ilustra os três conceitos através do chamado Modelo de 3 Universos (uma simplificação do Modelo de 4 Universos de Avizienis).

As falhas podem ser classificadas de acordo com a freqüência em que ocorrem no sistema. A lista abaixo trás a classificação, bem como os conceitos e um exemplo.

a) Transientes: ocorrem uma vez e desaparecem. Exemplo: a inversão de um bit devido a uma radiação externa incomum ao ambiente do circuito;

b) Intermitentes: ocorrem, desaparecem e reaparecem. Exemplo: a inversão de um bit devido a uma radiação externa comum ao ambiente do circuito;

c) Permanentes: ocorrem continuamente até que o componente defeituoso seja reparado. Exemplo: falhas de projeto.

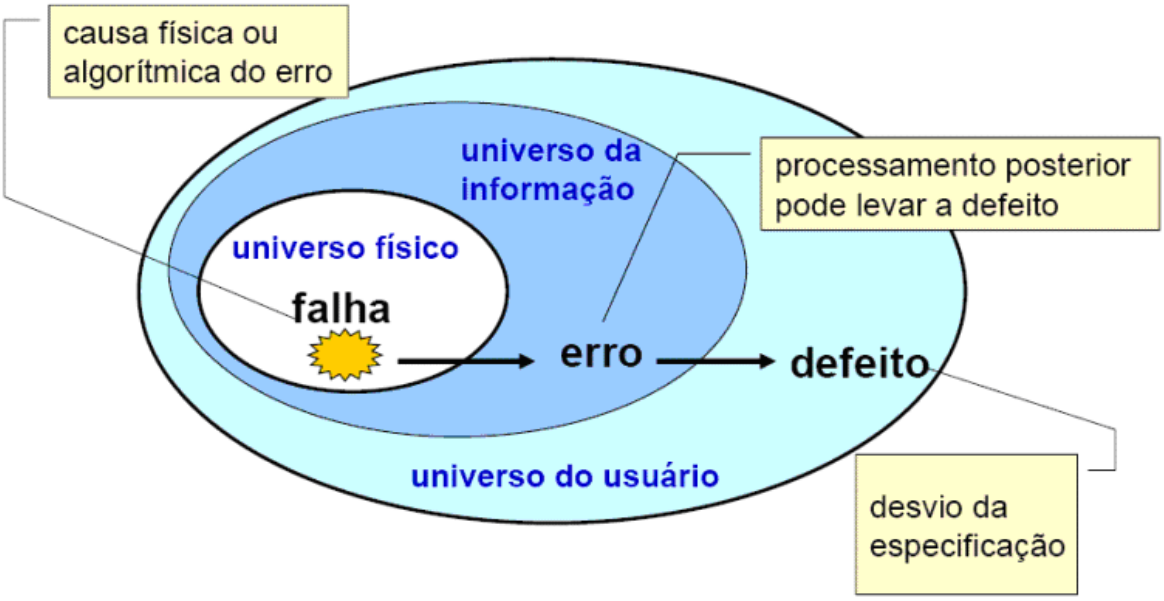

Figura 14 - Diagrama de 3 Universos ilustrando a diferença entre falha, erro e defeito.

As medidas para atingir a dependabilidade podem ser classificadas em quatro tipos, que estão listados a seguir.

a) Prevenção de Falhas: evitar a ocorrência ou introdução de falhas;

b) Remoção de Falhas: visa reduzir o número de falhas no sistema;

c) Previsão de Falhas: estimar o número, a ocorrência e as conseqüências das falhas;

d) Tolerância a Falhas: visa manter o correto serviço mesmo com a presença de falhas.

A Prevenção de falhas procura evitar a introdução de falhas durante as fases de especificação, projeto e implementação do sistema. Para tanto, a especificação deve ser rigorosa. A aplicação de técnicas de garantia da qualidade, bem como a utilização de metodologias de desenvolvimento, ajuda nesse sentido. Apesar de todos os esforços, falhas de projeto sempre ocorrem e sempre ocorrerão (PRADHAN, 1996).

A Remoção de falhas pode ser feita com inspeções de código, teste e verificação. Testes exaustivos são normalmente proibitivos devido à complexidade dos sistemas. O teste pode ser usado para comprovar a presença de falhas e não a sua ausência. A validação como forma de verificar se as especificações estão sendo atingidas também é uma forma de remover falhas, bem como: testes formais (que buscam prova matemática do correto funcionamento) e injeção de falhas (para verificar capacidade de tolerância a falhas). Deve-se salientar, contudo, que as técnicas de remoção de falhas não conseguem eliminar todas as falhas de projeto.

A Previsão de falhas é uma medida importante para indicar a necessidade da implementação de mecanismos de tolerância a falhas no sistema projetado. Pode ser feita de duas formas: qualitativa, levantando os tipos de falhas previstas, classificando essas falhas, a contribuição para possíveis defeitos e suas conseqüências; quantitativa, fazendo a predição da fiabilidade e disponibilidade do sistema (PRADHAN, 1996).

A Tolerância a falhas é uma medida que tem como objetivo tornar o sistema capaz de fornecer o serviço esperado mesmo na presença de falhas. Pode ser classificada em níveis, como listado abaixo.

a) Tolerância a falhas total – o sistema continua a operar com toda a sua funcionalidade mesmo na presença de falhas;
b) Degradação consentida (do inglês, *graceful degradation*) – o sistema continua a operar com uma degradação parcial em funcionalidade ou performance;
c) Paragem segura (do inglês, *fail-safe*) – o sistema deixa de operar, porém mantém uma resposta segura.

Uma característica que distingue sistemas distribuídos de sistemas centralizados é a noção de falha parcial. Uma falha parcial ocorre quando um componente no sistema distribuído falha. Um objetivo importante do projeto de um sistema assim é construí-lo de forma que possa se recuperar automaticamente de falhas parciais sem afetar dramaticamente o desempenho geral (JALOTE, 1994).

As técnicas de tolerância a falha demandam recursos adicionais, na forma de algoritmos, redundância ou diversidade. Assim, devem ser bem conhecidas para que a funcionalidade traga o melhor custo/benefício ao sistema.

2.3.2 Fases e Técnicas

Podem-se enumerar quatro fases distintas, desde a ocorrência de uma falha até a recuperação do sistema: detecção de erro; confinamento do dano; recuperação do erro; e tratamento da falha e continuidade do serviço. Nas seções seguintes, cada uma dessas fases será detalhada e técnicas utilizadas em cada uma serão discutidas.

2.3.2.1 Detecção de Erro

Falhas e defeitos não podem ser diretamente observados, mas podem ser deduzidos pela presença de erros. Como erros são definidos pelo estado do sistema, verificações podem

ser realizadas para verificar a existência dos mesmos. Mecanismos de detecção de erros muitas vezes são chamados de “detecção de defeitos/falhas”.

Um detector ideal deve ser: determinado somente a partir das especificações do sistema e não devem ser influenciados pelo projeto interno do mesmo; completo e correto, detectando todos os erros possíveis no comportamento do sistema que podem ocorrer na presença das falhas que se deseja tolerar, nunca informando um erro quando ele não estiver presente; independente do sistema com relação a sua susceptibilidade de falha. Se o detector falhar sempre que o sistema falhar, ele não será útil.

Em sistemas reais os critérios anteriores raramente podem ser completamente satisfeitos. Tipicamente não é viável realizar uma detecção completa devido a custos financeiros ou de desempenho, além de verificações muito complexas poderem falhar. Independência completa normalmente não pode ser obtida, já que o sistema e o detector usualmente compartilham algum ambiente (fonte de energia, por exemplo). Portanto, o objetivo é manter o custo da detecção de erro baixo, maximizando os erros que são detectados. Não há garantias de que todos os erros serão detectados, mas sim que a maioria daqueles de interesse, particularmente os que possuam maior probabilidade de ocorrer, o serão.

Algumas técnicas de detectores merecem ser destacadas nesse ponto: a de detectores replicados; detectores temporais; e detectores estruturais e de código.

Na técnica de detectores replicados, como o próprio nome diz, replica-se algum componente do sistema e os resultados de cada um são comparados, ou votados, para a detecção de erros. É um método caro, usualmente empregado para componentes de hardware utilizados em áreas medidas, militar e espacial. É preciso salientar que, caso haja falha de projeto, essa técnica é obsoleta (PRADHAN, 1996).

A técnica de detectores temporais é utilizada se a especificação de um componente incluir restrições de tempo, para verificar se essas restrições estão sendo cumpridas. Uma violação de tempo indica que o componente está se comportando incorretamente.

Na técnica de detectores estruturais e de código são feitas verificações semântica – que tenta garantir se o valor do dado é consistente com o restante do sistema – e estrutural – que verifica se a estrutura do dado é consistente com o que deveria ser. Como forma mais usual de verificação estrutural é a codificação, na qual bits são adicionados aos dados e estão relacionados a estes (WEBER, 2001).

2.3.2.2 Confinamento do Dano

Devido a interações entre os componentes, um erro pode ser propagado para outras partes do sistema. Portanto, após detectar um erro e antes de recuperar o sistema do estado de erro necessitamos determinar os limites do estado corrompido. O objetivo é identificar o limite no estado além do qual nenhuma troca de informação ocorreu. O dano estará então confinado a este limite. Os limites podem ser identificados dinamicamente, registrando-se e examinando o fluxo de informação entre componentes. Pode-se ainda projetar o sistema de forma que portas anti-incêndio são estaticamente incorporadas ao sistema (AVIZIENIS, 2001).

2.3.2.3 Recuperação do Erro

A recuperação de erros ocorre após a detecção e envolve a troca do estado atual incorreto para um estado livre de falhas. Pode ser de duas formas: técnicas de recuperação por retorno (do inglês, *backward error recovery*) e técnicas de recuperação por avanço (*forward error recovery*). A Tabela 2 detalha as duas técnicas e a Figura 15 ilustra como a técnica age.

Técnica	Definição	Características	Implementação
forward error recovery	condução a novo estado ainda não ocorrido desde a última manifestação de erro	eficiente, mas danos devem ser previstos acuradamente	específica a cada sistema
backward error recovery	condução a estado anterior	alto custo, mas de aplicação genérica	pontos de verificação (checkpoints), pistas de auditoria (audit trails), ...

Tabela 2 - Características das técnicas de recuperação de falha.

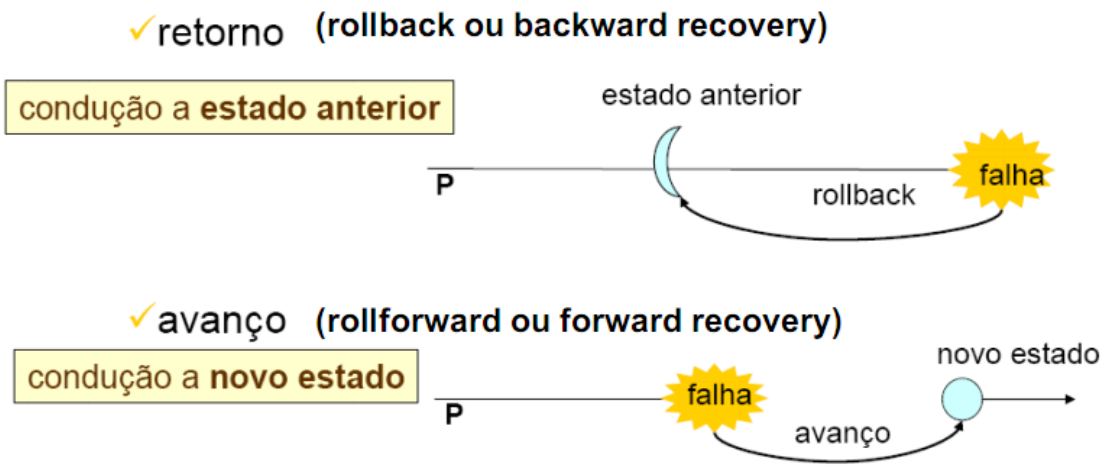

Figura 15 - Representação temporal da falha e metodologia utilizada nas técnicas.

2.3.2.4 Tratamento da falha e continuidade do serviço

Se a falha é permanente, o componente com problema deve ser identificado e não mais utilizado após a recuperação, que compreende uma reconfiguração dinâmica do sistema, sem intervenções manuais. Uma estratégia simples para se atingir isso é o uso de *stanby spare*: quando o componente principal falhar, um componente redundante passa a ser utilizado.

Nessa fase geralmente é considerada a hipótese de falha única, ou seja, uma única falha ocorrendo a cada vez (WEBER, 2001).

2.4 Conclusão

De posse dos conceitos envolvidos no estudo de redes-em-chip e tolerância a falhas, é possível agora realizar uma análise detalhada das características de funcionamento da NoC Hermes, buscar fragilidades e propor melhorias e técnicas de tolerância a falhas que melhorem o desempenho da rede.

3 A NoC Hermes

Valendo-se dos conceitos introduzidos no Capítulo 2, será apresentada agora a NoC Hermes. Começando pelo seu elemento principal, o roteador (também comumente chamado de chave) Hermes, passando pela topologia adotada e parâmetros utilizados nesse trabalho. Em seguida, é feita uma análise crítica das fragilidades dessa rede-em-chip.

3.1 A chave Hermes

O principal objetivo da chave em uma NoC é proporcionar um mecanismo de transferência de mensagens confiável entre núcleos *IP*. Para tanto, chaves são formadas por lógica de arbitragem, um algoritmo de roteamento, portas e armazenamento de entrada e/ou saída. As portas conectam-se a núcleos *IP* e/ou a outras chaves para formar a rede de interconexão.

A chave Hermes contém um bloco de controle centralizado e cinco portas bidirecionais denominadas *NORTH, SOUTH, EAST, WEST, LOCAL*. A porta *LOCAL* é dedicada ao núcleo *IP* local e as demais portas são utilizadas para conexão a outras chaves, de acordo com a topologia adotada.

3.1.1 Controle de Fluxo

O controle de fluxo é do tipo *handshake*. Para implementação do mesmo, a porta de saída é composta pelos sinais listados abaixo:

a) tx: indica que existe um dado a ser enviado;

b) data_out: contém o dado (16 bits) a ser enviado;

c) ack_tx: indica que o dado foi enviado.

E a porta de entrada, pelos sinais listados abaixo:

a) rx: indica que existe um dado a ser recebido;

b) data_in: contém o dado (16 bits) a ser recebido;

c) ack_rx: indica que o dado foi recebido.

Na Figura 16 pode ser visualizada a forma de ligação desses sinais em uma rede.

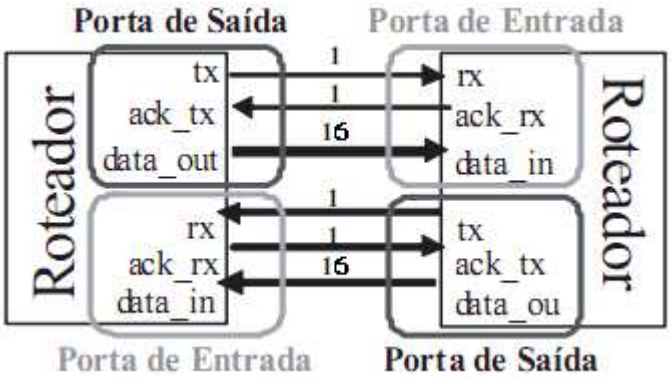

Figura 16 - Ligação dos sinais das portas de entrada e saída de duas chaves Hermes (BEZERRA, 2009).

3.1.2 Memorização

A memorização é implementada através da inclusão de filas de armazenamento temporário em cada porta de entrada, com tamanho parametrizável. Esta abordagem contribui para a redução de chaves que ficam bloqueadas para um único pacote que está sendo transmitido pela rede, o que permite maior velocidade de transmissão de dados.

3.1.3 Algoritmos de Arbitragem e Chaveamento

Na lógica de controle são implementados o algoritmo de arbitragem (centralizado) e o algoritmo de chaveamento de pacotes. O diagrama de blocos da chave Hermes é mostrado na Figura 17.

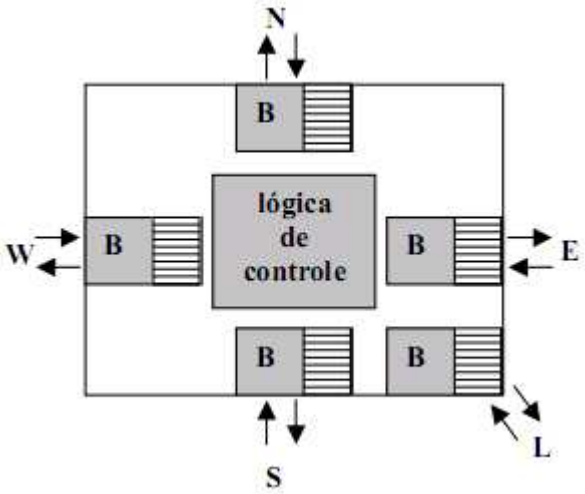

Figura 17 - Diagrama de blocos destacando as unidades internas do roteador Hermes.

O modo de chaveamento utilizado é *wormhole*, por proporcionar baixa latência e menor consumo de área, quando comparado aos demais modos de chaveamento por pacote (MORAES, 2003).

Para ser transmitido pela rede de interconexão, um pacote deve ser dividido em *flits*, que são as menores unidades de transmissão. O tamanho do *flit* é parametrizável, todavia a alteração de tamanho representa necessidades de mudança na forma como os dados devem ser avaliados para roteamento. Atualmente os tamanhos de *flit* com que se trabalha são 8, 16, 32, e 64 bits. A organização do pacote é definida abaixo, tendo em vista *flits* de tamanho igual a 16 bits.

a) O primeiro *flit* do pacote armazena o endereço de destino do pacote;

b) O segundo *flit* contém o tamanho do corpo da mensagem, expresso em *flits*;

c) Os demais *flits* representam o corpo da mensagem.

Uma relação importante existe entre o tamanho do *flit* e o tamanho máximo da rede. Por exemplo, se o *flit* tiver tamanho de 16 bits significa que o tamanho máximo da rede será 256 x 256 roteadores, pois metade do primeiro *flit* especifica a abscissa do destino e a outra metade especifica a ordenada deste.

A lógica de controle é composta por dois módulos: arbitragem, que é centralizada; e roteamento. Quando um novo pacote chega em uma chave, o cabeçalho da mensagem é armazenado e uma requisição de roteamento é realizada. A arbitragem é definida com prioridade rotativa (*round robin*) para atendimento das requisições de roteamento. Assim, de acordo com a prioridade das portas que estão requisitando o roteamento seu pedido é atendido.

3.1.4 Algoritmo de Roteamento

O algoritmo de roteamento é o XY. O endereçamento na rede é dado pelas coordenadas XY de cada chave. O cabeçalho do pacote possui o endereço XY^{alvo} do destino e a chave onde se encontra o cabeçalho do pacote contém o endereço XY^{local}. Assim, o roteamento com o algoritmo XY é realizado através do algoritmo apresentado abaixo.

a) se $X^{alvo} > X^{local}$ então reserva porta *WEST*;

b) se $X^{alvo} < X^{local}$ então reserva porta *EAST*;

c) se $X^{alvo} = X^{local}$ e $Y^{alvo} > Y^{local}$ então reserva porta *SOUTH*;

d) se $X^{alvo} = X^{local}$ e $Y^{alvo} < Y^{local}$ então reserva porta *NORTH*;

e) se $XY^{alvo} = XY^{local}$ então reserva porta *LOCAL*.

Depois de concluído o roteamento, o header é enviado ao destino (sejam eles a porta local ou um roteador vizinho). Após o envio do header, o próximo *flit*, com o tamanho do pacote, é carregado em um contador – individual para cada porta – que é decrementado, à medida que os *flits* daquele pacote são enviados, até chegar a zero e assim a conexão ser terminada.

Depois de terminada a transmissão da mensagem, a lógica de controle desfaz a conexão entre portas de entrada e de saída.

Ainda referente ao algoritmo de roteamento, a reserva de portas somente é possível se a porta destino estiver livre, ou seja, se a porta pela qual o pacote será enviado não estiver reservada. Por serem portas bidirecionais, estas podem receber e enviar *flits* ao mesmo tempo, conforme mostra a Figura 18, na qual a chave está transmitindo com vazão máxima.

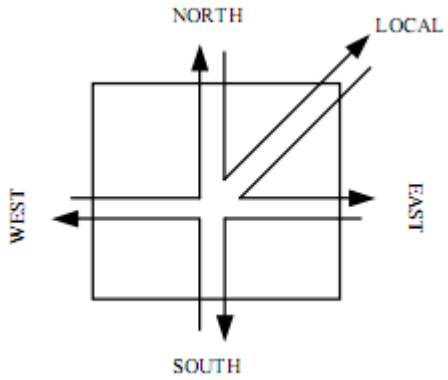

Figura 18 - Exemplo de chave Hermes operando a máxima vazão.

3.1.5 Exemplo de Funcionamento

A Figura 19 tem um diagrama de blocos parcial do roteador xLyL=0x0000, mostrando duas de suas portas (*EAST* e *LOCAL*), o árbitro e a lógica de roteamento. Os números representam a seqüência temporal em que os eventos ocorrem dentro da chave.

As etapas descritas abaixo são da chegada de um pacote na porta local (simbolizada na figura com um L, do inglês *LOCAL*), com destino ao roteador com endereço xLyL=0x0022).

Assim, de acordo com o algoritmo de roteamento XY, o mesmo deverá seguir pela porta *EAST*, já que deverá primeiro ter sua coordenada X alinhada. Para ilustrar como a máquina de estados da chave funciona, cada passo foi descrito com detalhes.

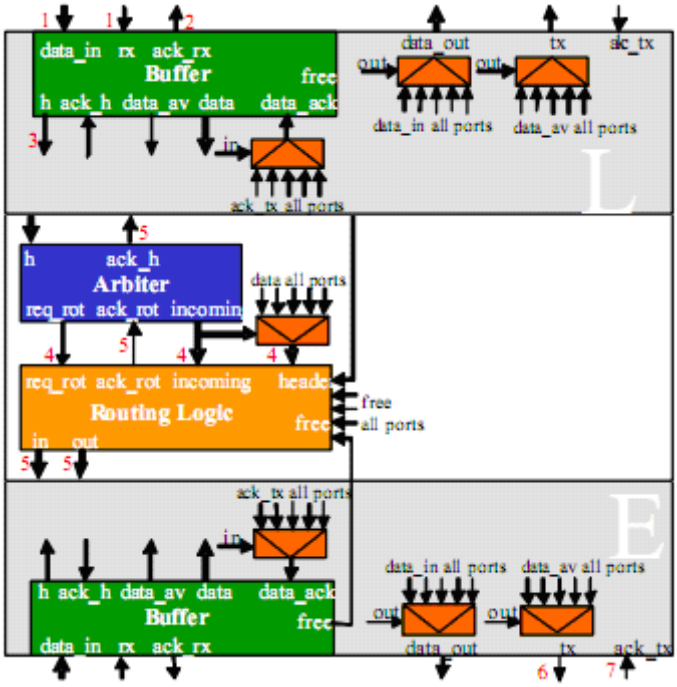

Figura 19 - Unidades internas de uma chave Hermes, os sinais que as interligam. Destaque para a seqüência em que o sinais são excitados (MORAES, 2003).

a) No instante 1, a chave (xLyL=0x0000) recebe o pacote pela porta *LOCAL*. O sinal rx da porta é ativado e o sinal data_in da mesma contém o primeiro *flit* do pacote.

b) No instante 2, a porta de entrada *LOCAL*, já após verificar que existe lugar na sua fila, sinaliza que o *flit* foi recebido, através do sinal ack_rx.

c) No instante 3, a porta solicita chaveamento ao árbitro, ativando seu sinal h.

d) Ainda no instante 3, o árbitro, que recebe o pedido de chaveamento de todas as portas, seleciona a de maior prioridade. A porta *LOCAL* é selecionada porque, neste momento, assume-se que não há nenhuma outra requisição (e mesmo que houvesse, o exemplo continua válido, pois podia se assumir que a porta *LOCAL* seria a de maior prioridade).

e) No instante 4, o árbitro solicita o chaveamento à lógica de chaveamento enviando o *flit* da porta *LOCAL*, que é o endereço da chave destino, e a porta de origem da requisição (sinal incoming, valor 4, representando a porta *LOCAL*).

f) No instante 5, o algoritmo de chaveamento XY é executado, as tabelas de chaveamento são preenchidas e o sinal ack_rot é ativado indicando que a conexão foi estabelecida.

g) Ainda no instante 5, o árbitro informa à porta *LOCAL* que a conexão foi estabelecida e que o *flit* pode ser transmitido.

h) No instante 6, a chave ativa o sinal tx da porta de saída selecionada (no caso, a porta *EAST*) e coloca o *flit* em data_out desta mesma porta.

i) No instante 7, a chave aguarda o sinal ack_tx da porta de saída selecionada (no caso, a porta *EAST*) sinalizar o recebimento do *flit* enviado.

j) A partir daí o pacote será enviado até o seu último *flit*, seguindo o protocolo *handshake*.

3.1.6 Características Temporais

A chave Hermes tem características temporais determinísticas, que valem ser ressaltadas. Através da análise da chegada do primeiro *flit* de um pacote a uma das portas da chave, até o início do seu envio através da porta de saída, podem-se apresentar todas essas características.

Para tanto, foi capturada a forma de onda, que é apresentada na Figura 20, dos principais sinais da chave utilizada no exemplo de funcionamento da seção anterior. Além disso, cursores foram inseridos e numerados de acordo com os instantes em que cada um acontece, destacando ainda o instante temporal.

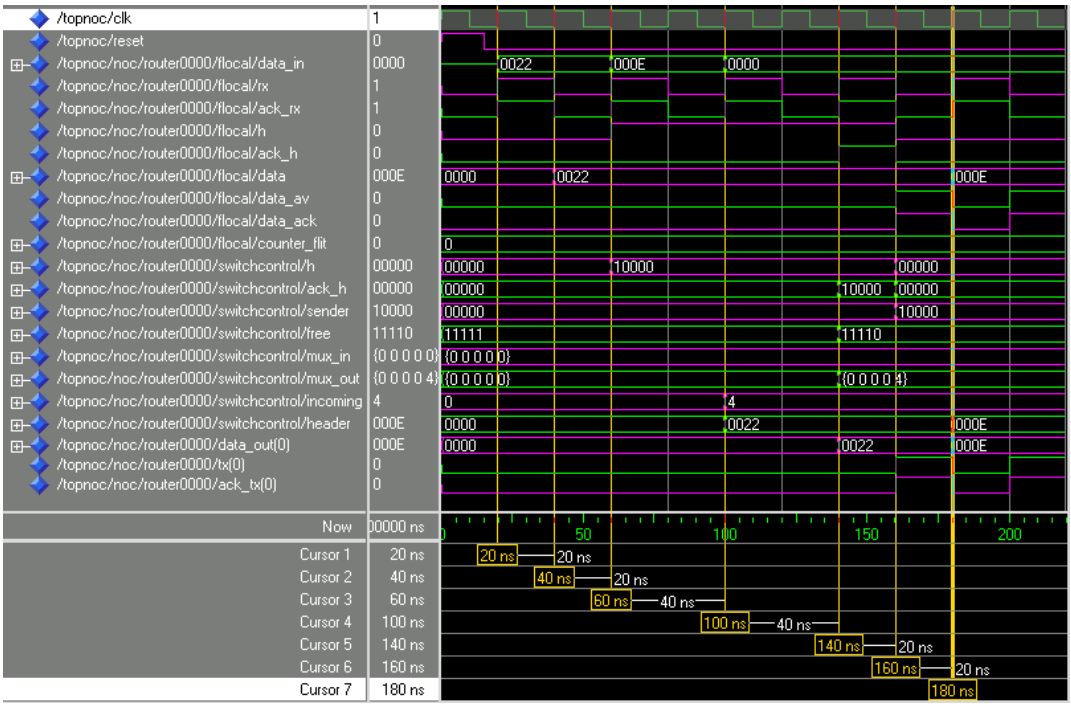

Figura 20 - Análise temporal dos eventos após a chegada de um pacote em uma porta de entrada da chave Hermes.

Da forma de onda, podem-se checar as conclusões listadas abaixo.

a) O *clock* do circuito tem período de 20ns. Sua freqüência de operação é, então, de 50MHz.

b) Desde a confirmação do recebimento do primeiro *flit* do pacote (marcado pelo cursor 2, em t = 40ns) até o envio do mesmo pela porta de saída (marcado pelo cursor 6, em t = 160ns), tem-se t = 120ns de diferença, ou 6 ciclos de *clock*. Esse tempo pode variar de acordo com a disposição (ocupada ou não) da porta de saída.

c) São necessários dois ciclos de *clock* para o envio de um *flit* (como era esperado, devido ao protocolo *handshake*).

3.2 A Rede Hermes

A forma como as chaves se interconectam define a topologia da rede de interconexão. Para a proposta da NoC Hermes original foi utilizada a topologia malha 2D, onde cada chave está ligada a um núcleo *IP* pela porta *LOCAL* e, dependendo da posição da chave na rede, a outras chaves com as demais portas, conforme a Figura 21. Apesar de esta topologia ter sido definida, a chave Hermes permite que outras topologias também sejam implementadas tais como toro e hipercubo. Para tanto, as chaves têm de ser interconectadas de forma a representar tais topologias e o algoritmo de roteamento tem de ser adequado.

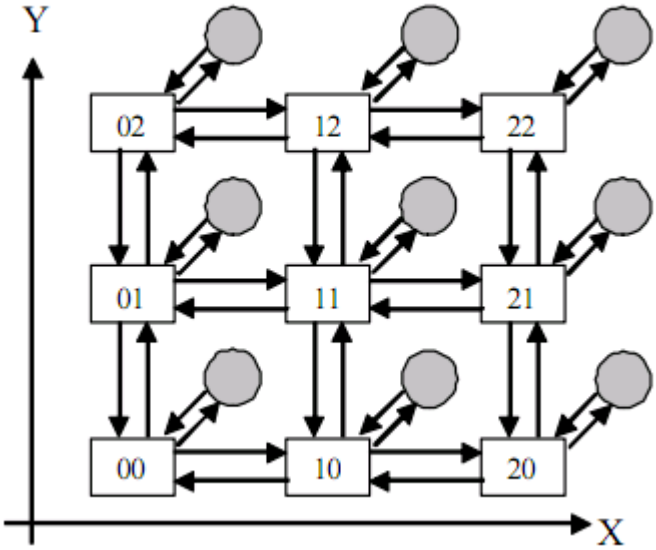

Figura 21 - Proposta inicial de topologia para a rede Hermes.

3.3 Fragilidades

Já de posse dos princípios de funcionamento da chave Hermes, será feita, nessa seção, uma lista das fragilidades encontradas na rede Hermes. Tais pontos justificam as propostas de melhoria desse trabalho, bem como o provimento de um mecanismo de tolerância a falhas.

3.3.1 Controle de Timeout no Envio

A primeira e principal fragilidade da chave Hermes é a falta de um controle individual de timeout de suas operações. A rede é formada por elementos independes que, idealmente, devem ser robustos e ter autonomia para criticar se a entrada detectada (ou pior, não detectada) era a esperada ou não.

A análise temporal é uma das formas de se realizar esse tipo de verificação. Se uma tarefa tem um tempo para ser executada e não o foi, algo de errado aconteceu (a chave vizinha travou ou sofreu um *reset*, por exemplo). Para explorar tal técnica basta o pré-conhecimento das regras temporais que devem ser obedecidas.

Como exemplo, pode-se tomar o algoritmo de chaveamento por pacotes do tipo *wormhole* utilizado em conjunto com o controle de fluxo por *handshake*. Por estabelecer uma conexão virtual entre um roteador e seu vizinho – no sentido que nenhum dado, o qual não seja daquele pacote, possa ser transferido até que todo o pacote já o tenha – pode-se concluir que após enviar o cabeçalho de um pacote, dois ciclos de *clock* depois, o segundo *flit* daquele pacote deve já estar disponível para envio. Isso porque, no primeiro ciclo, o emissor disponibiliza o dado em data_out e sinaliza o destinatário com o sinal tx. No segundo ciclo, o emissor recebe a confirmação, pelo sinal ack_tx de que o destinatário recebeu o dado e assim o emissor enviará o próximo dado já no próximo ciclo. Tal regra deve valer para todos os *flits* do pacote: após enviar o segundo *flit*, contendo o tamanho daquele pacote, enquanto houver *flits* desse para serem transmitidos (ou seja, enquanto o contador regressivo não chegar à zero), não pode haver um atraso maior que dois ciclos de *clock* entre um *flit* e outro. A máquina de estados da Figura 22 mostra uma alternativa para a implementação de um algoritmo contemplando tais regras.

Veja que, sem esse controle de *timeout*, existe o risco de a conexão entre as portas de entrada e saída jamais ser desfeita, fazendo assim com que essa porta de saída jamais fique disponível para outra porta de entrada. Para tanto, basta que não chegue mais nenhum outro pacote na porta de entrada já conectada. Caso chegue, será ainda pior: os *flits* desse segundo pacote serão interpretados como parte do primeiro, podendo fazer com que a chave jamais

entre em funcionamento normal novamente e envie falsos pacotes a destinos aleatórios (já que interpretará o *header* do segundo pacote como *flit* do *payload* do primeiro, *flit* do *payload* do segundo como *header* e assim sucessivamente).

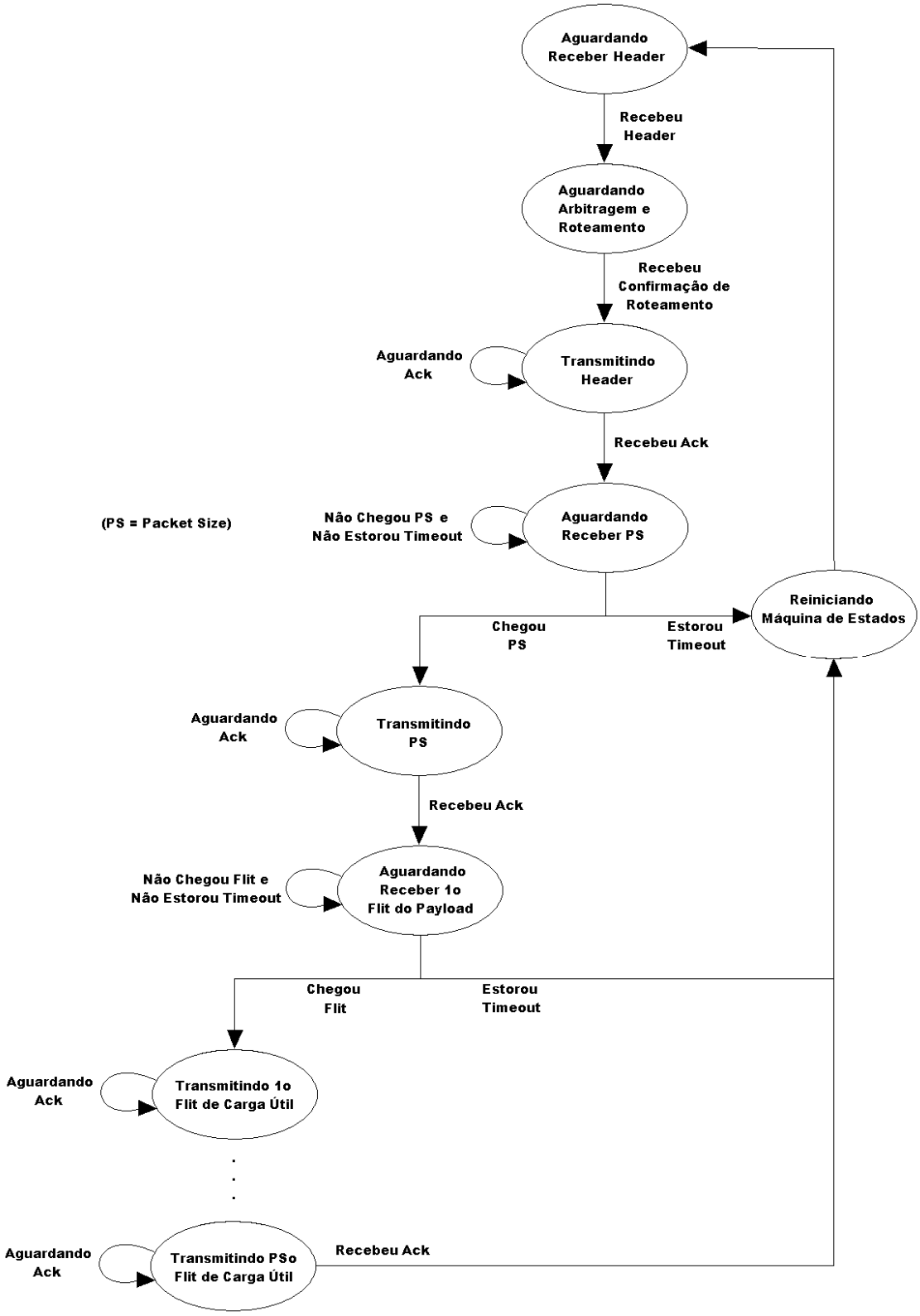

Figura 22 - Opção de controle de timeout no recebimento de *flits* de um mesmo pacote.

Assim, além de causar uma grande contenção em toda rede, o simples reset de um dos roteadores pode fazer com que o mesmo nunca mais volte ao seu estado normal de funcionamento causando o colapso de toda a rede (devido ao efeito dominó do bloqueio de diversas portas de saída).

3.3.2 Controle de Início e/ou Fim de Pacote

O problema descrito na seção anterior acabou revelando um grave problema da NoC Hermes: se a comunicação entre dois nós sair de sincronismo, não há como restabelecê-lo. Isso se dá devido à falta de uma marca de início e/ou fim de pacote, técnica já utilizada em outras redes-em-chip, como na SoCIN (ZEFERINO, 2003b). A Figura 23 mostra o formato de pacote utilizado nessa rede.

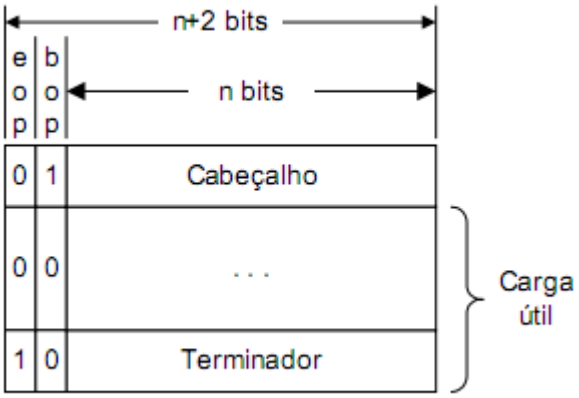

Figura 23 - Exemplo de formato de pacote mais robusto, com bits de sinalização para o início e fim de um pacote (ZEFERINO, 2003a).

Assim, se por conta de alguma falha, for perdido o sincronismo entre a máquina de estados do roteador e os *flits* do fluxo, o funcionamento voltará ao normal já no próximo cabeçalho detectado.

Outras técnicas como a utilização de *flits* especiais para marcar o início e fim de pacote, sem a necessidade de linhas adicionais na comunicação também é uma opção. Contudo, torna a tarefa muito mais complexa, pois necessita de constante processamento a fim de encontrar um desses *flits* especiais dentro de um fluxo não sincronizado. Mais ainda, a necessidade da utilização de *ESCAPE* torna esse método muito dispendioso tanto em termos de memória como de processamento (e, indiretamente, de consumo de potência do circuito).

3.3.3 Falha Permanente

Em caso de falha permanente em uma chave, não só a comunicação do *IP* conectado a ela é afetada, mas pode-se dizer toda a rede. A NoC, cedo ou tarde, entrará em colapso. Para esclarecer como isso pode acontecer será utilizado um exemplo. A Figura 24 ilustra uma rede-em-chip 3 x 3 de roteadores Hermes que servirá de cenário para o exemplo.

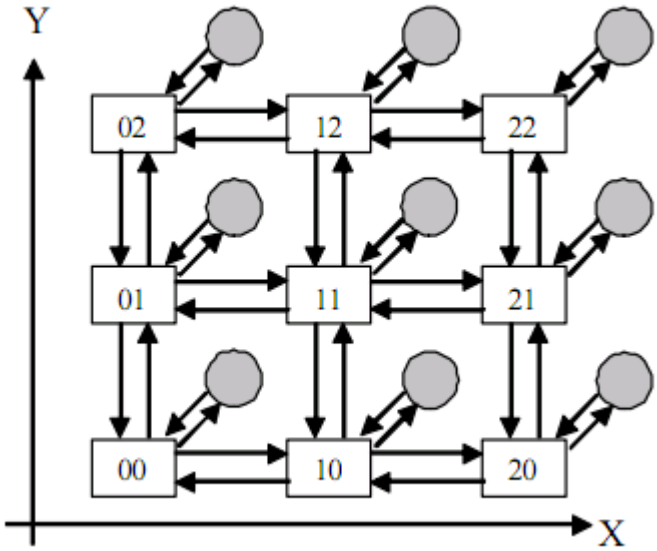

Figura 24 - Cenário para apresentação das consequências de uma falha permanente em um roteador Hermes.

Para o exemplo, será considerado que as filas de entrada dos roteadores têm tamanhos 16 e que esse seja também o tamanho do pacote (apesar desses valores influenciarem na ocorrência do problema, eles não o impedem). Será considerada a seqüência de eventos apresentada abaixo.

a) A chave 00 começa a enviar um pacote à chave 10 com destino à chave 21;

b) A chave 00 termina de enviar o pacote para a chave 10, mas essa ainda envia os últimos *flits* à chave 20;

c) Nesse momento, a chave 20 trava e a chave 10 aguardará indefinidamente pela confirmação do recebimento do *flit* atual;

d) A chave 00 precisa enviar um pacote à chave 11, e então inicia o envio dos *flits* à chave 10 que vai enfileirando os mesmos até que a sua fila encha;

e) Pela contenção causada pelo primeiro pacote, que nunca será entregue e assim nunca sairá da fila, o segundo pacote – com destino à chave 11 – nunca será entregue.

Conhecido como efeito HOL (do inglês, *Head of Line*), pelo *flit* da cabeça da fila bloquear a entrega dos demais, esse efeito é discutido na literatura e pode ser amenizado com a utilização de canais virtuais. Essa solução, contudo, traz muitas complicações ao projeto da chave, tornado-a ainda mais passível de falhas (BEZERRA, 2009). Duas soluções, bem mais simples, para esse problema serão apresentadas e discutidas no Capítulo 4.

3.4 Conclusão

Após o estudo detalhado das características de funcionamento da NoC Hermes e de suas fragilidades, foram incorporadas melhorias e novas funcionalidades ao projeto da mesma, resultando assim na NoC HermesFT, assunto do Capítulo 4.

4 A NoC HermesFT

Nesse Capítulo serão discutidas as alterações e técnicas utilizadas na NoC Hermes que resultaram em uma rede-em-chip mais robusta e tolerante a falhas: a NoC HermesFT. Para tanto, foram exploradas as fragilidades expostas no Capítulo 3 e implementadas técnicas de tolerância a falhas.

Os elementos utilizados foram: a chave HermesFT, que é uma versão melhorada da chave Hermes; o WDT, que é um circuito detector de falhas da chave; e o circuito chaveador, que será o responsável por realizar o curto-circuito dos enlaces de uma chave em falha permanente. A seguir, cada elemento terá seu princípio de funcionamento detalhado e, em seguida, um esboço da interconexão dos mesmos será apresentado.

4.1 A Chave HermesFT

A chave HermesFT é similar à chave Hermes, com algumas melhorias, que serão citadas nessa seção. Além das cinco portas da sua antecessora, foi adicionada uma porta unidirecional denominada *GARBAGE*, como opção para a coleta de pacotes inválidos, aqueles que são enviados a um roteador em falha.

4.1.1 Controle de Fluxo

O controle de fluxo é também do tipo *handshake*. Para implementação do mesmo, a porta de saída é composta pelos sinais listados abaixo.

a) tx: indica que existe um dado a ser enviado.

b) data_out: contém o dado (16 bits) a ser enviado.

c) txh: indica que o *flit* presente em data_out é o cabeçalho de um pacote.

d) ack_tx: indica que o dado foi enviado.

E a porta de entrada, pelos sinais abaixo.

a) rx: indica que existe um dado a ser recebido.

b) data_in: contém o dado (16 bits) a ser recebido.

c) rxh: indica que o *flit* presente em data_in é o cabeçalho de um pacote.

d) ack_rx: indica que o dado foi recebido.

A diferença para a chave Hermes foi a inclusão dos sinais txh e rxh. Tal diferença pode ser notada ao se comparar a Figura 25 (que ilustra a forma de ligação dos sinais de duas chaves HermesFT) à Figura 16 (que ilustra a forma de ligação dos sinais de duas chaves Hermes).

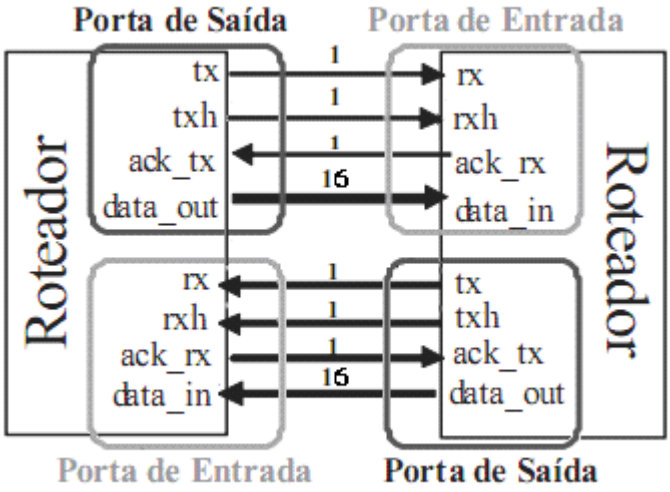

Figura 25 - Ligação dos sinais das portas de entrada e saída de duas chaves HermesFT.

4.1.2 Memorização

A memorização é implementada também através de filas de armazenamento temporário em cada porta de entrada, com tamanho parametrizável.

4.1.3 Algoritmos de Arbitragem e Chaveamento

O modo de chaveamento utilizado também é *wormhole*. Todas as características da lógica de controle da chave Hermes foram conservadas. A única observação é quanto ao primeiro *flit* do pacote, que terá agora o sinal txh e rxh em nível alto, no momento da transmissão e recepção, respectivamente.

4.1.4 Algoritmo de Roteamento

O algoritmo de roteamento é também o XY. Todas as características do algoritmo de roteamento da chave Hermes foram conservadas.

4.1.5 Exemplo de Funcionamento

O exemplo de funcionamento da chave é equivalente ao da chave Hermes, apresentado no Capítulo 3. A única diferença é a excitação dos sinais adicionados de sinalização de cabeçalho.

4.1.6 Características Temporais

As características temporais da chave são iguais às da chave Hermes.

4.2 O WDT

O circuito denominado WDT (do inglês, *Watch Dog Timer*) realiza a detecção de falha da chave a qual esteja monitorando. A técnica utilizada foi a mesma que deu o nome à unidade e tem por objetivo realizar a detecção temporal de falhas, qual quer que sejam (conhecidas ou não).

A implementação desse componente previu seis entradas e duas saídas, tornando-o bastante personalizável com relação a seu comportamento. A Figura 26 tem o esboço do circuito, com suas entradas e saídas. Em seguida é feita a descrição e a utilidade de cada pino.

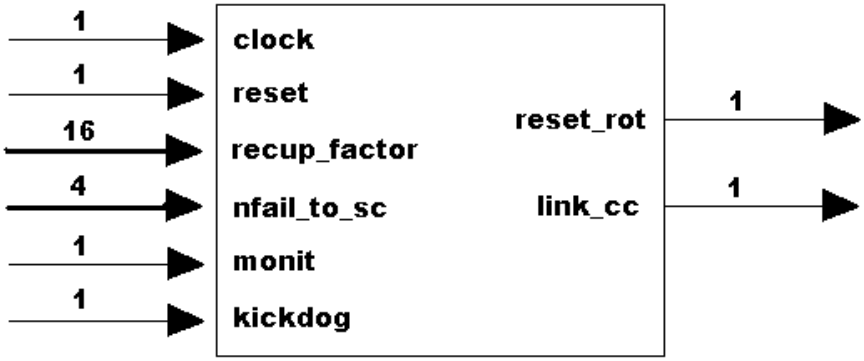

Figura 26 - Representação em bloco, com entradas e saídas, do circuito WDT.

a) clock: sinal de sincronismo com o resto do circuito;

b) reset: sinal com pulso de inicialização do sistema;

c) recup_factor: registrador de 16 bits com fator de recuperação de falha do sistema em monitoramento;

d) nfail_to_sc: registrador de 4 bits com o número de falhas para considerar o circuito monitorado em falha permanente;

e) monit: sinal para iniciar/parar o funcionamento do circuito detector;

f) kickdog: sinal de indicação que o circuito monitorado está funcionando corretamente.

Para o melhor entendimento dos sinais do circuito apresentado, abaixo se tem uma explicação mais detalhada do seu funcionamento.

O sinal monit habilita a detecção de falha do circuito sendo monitorado. Esse sinal é importante por um motivo: se um circuito não estiver sendo utilizado, não há porque ser monitorado (isso só levaria a um maior consumo de potência). Em outras palavras, nos termos utilizados em tolerância a falhas: neste caso, a falha e o erro no roteador são toleráveis caso o mesmo não esteja sendo utilizado, já o defeito, não o é.

O monitoramento realizado pelo circuito detector é feito através de um contador regressivo. Partindo de um valor (determinado pelas regras temporais do circuito, as quais devem ser atendidas), aqui chamado de *timeout*, o mesmo é decrementado a cada borda de subida do sinal de clock. Caso chegue a zero, é inferido que o circuito monitorado tem uma falha, que se transformou em erro e, em seguida, em defeito. Para evitar que isso ocorra, o circuito monitorado, em funcionamento normal, deve fixar o sinal kickdog em nível alto a cada borda de subida do sinal de clock.

Na inicialização do sistema, são armazenados os valores dos sinais recup_factor e nfail_to_sc.

Após a detecção de uma falha, um contador interno de falhas é incrementado e comparado ao valor de nfail_to_sc, caso o primeiro seja maior ou igual ao segundo, o sinal link_cc é ativado. Caso a condição anterior não seja satisfeita, o sinal reset_rot é pulsado.

No caso de funcionamento normal, o circuito monitorado ganha e acumula créditos. Esses, por sua vez, quando chegam a um limite superior igual a recup_factor vezes o *timeout*, podem anular até uma falha anteriormente contabilizada. O significado final desse mecanismo é o seguinte: o circuito monitorado deve funcionar corretamente *n* vezes mais tempo que estar em falha. Assim, o fator de recuperação entra no funcionamento do detector no sentido de assegurar a disponibilidade de um circuito que necessite de um nível de confiabilidade menor.

Pode-se notar que os dois sinais de saída do detector agem em situações diferentes de falha. O primeiro age na situação em que é inferido que há uma falha transitória do circuito monitorado. Assim, a ação tomada é dar um reset naquele elemento para que o mesmo volte ao seu funcionamento normal. Essa medida é muito útil no caso, por exemplo, da mudança do valor de um bit por efeito de uma alteração do nível de tensão em certa parte do circuito. O segundo age na situação em que a primeira não foi eficaz. Como exemplo, pode-se citar o

rompimento do dielétrico que isola a base de um MOSFET do circuito de um roteador, por se tratar de uma falha permanente.

4.2.1 Eficácia do WDT

A eficácia do WDT em termos de detecção da falha depende de um segundo módulo inserido no roteador HermesFT, que dá ao mesmo a condição de um roteador monitorável. Tal módulo age no sentido de monitorar o circuito dos 6 elementos básicos do roteador: as 5 portas e a lógica central de controle que realiza a arbitragem e o roteamento. Na Figura 27 é apresentada a implementação desse módulo e, em seguida, é explicado seu funcionamento.

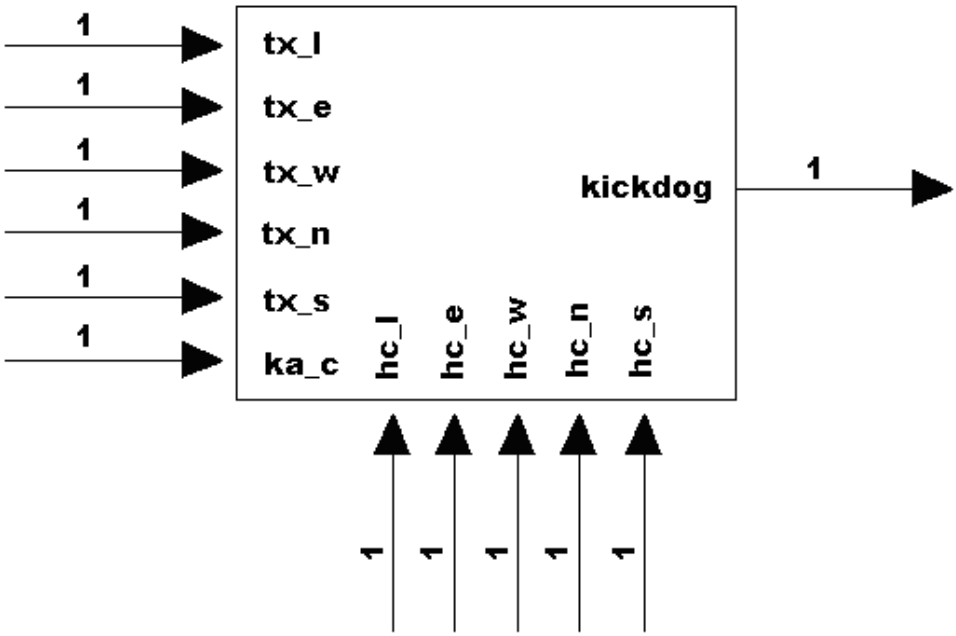

Figura 27 - Representação do circuito do roteador responsável por indicar existência de falha para o WDT.

O circuito tem como entrada os sinais tx e hc de cada porta de saída – LOCAL (_l), *EAST* (_e), *WEST* (_w), *NORTH* (_n) e *SOUTH* (_s) – além de um sinal que indica o funcionamento da máquina de estados da lógica central do roteador (ka_c) que, quanto em funcionamento normal, pulsará tal sinal em cada transição de estado. O sinal tx indica que há dados sendo enviados pela porta. Já o sinal hc indica que a porta está ocupada por uma conexão com uma das porta de entrada do roteador. O sinal kickdog vai ser conectado diretamente ao sinal, com mesmo nome, do circuito WDT.

Caso haja dados em qualquer porta de entrada do roteador, o circuito WDT iniciará o monitoramento. Caso não haja nenhuma conexão estabelecida entre portas de entrada e portas de saída (ou seja, todos os sinais hc_* em nível zero), a máquina de estados da lógica central

deve estar progredindo e o sinal ka_c será diretamente conectado ao sinal kickdog. Caso contrário, haverá alguma conexão estabelecida e, pelo menos um sinal tx deverá estar pulsando. Como há cinco sinais tx, um para cada porta, é feita a combinação de todos em uma lógica OU.

Tal funcionamento garante assim o correto monitoramento do roteador nos dois momentos de seu funcionamento: no primeiro momento, no qual é realizada a arbitragem e roteamento; e no segundo, no qual é realizado o envio dos *flits*, um a um, de um pacote através da conexão já estabelecida entre a porta de entrada e a porta de saída.

4.3 O Chaveador de Enlaces

Apesar de ser um elemento simples, por se tratar basicamente de um conjunto de multiplexadores e demultiplexadores, há nele algumas regras importantes no funcionamento correto da rede com elemento(s) em falha.

O mecanismo de funcionamento desses dois componentes é ilustrado na Figura 28 e na Figura 29, respectivamente.

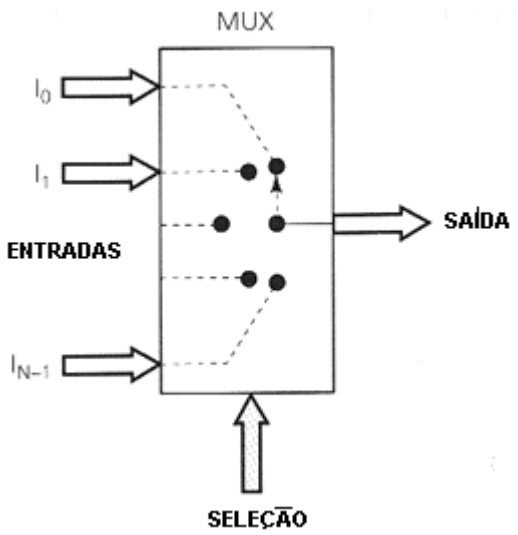

Figura 28 - Bloco representativo do funcionamento de um multiplexador.

Em cada porta de saída do roteador é posto um mux com uma saída (o enlace de saída) e duas entradas (um enlace alternativo de entrada de outra porta, com o qual o enlace de saída será curto-circuitado caso seja detectado uma **falha permanente**, e a porta de saída do roteador).

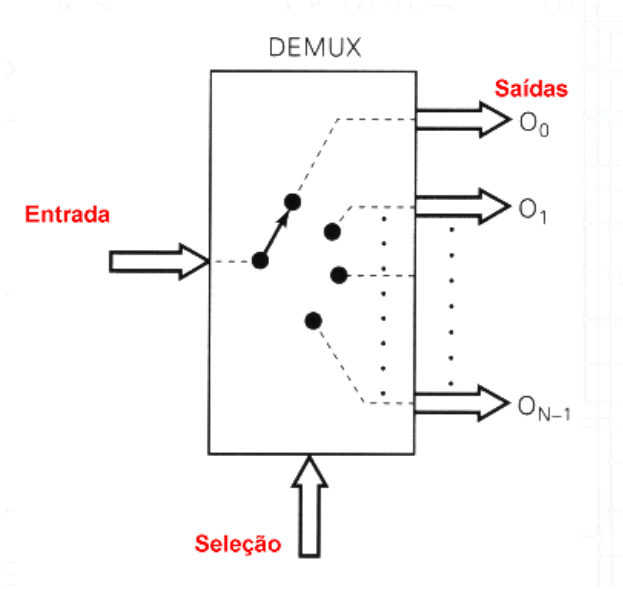

Figura 29 - Bloco representativo do funcionamento de um demultiplexador.

Em cada porta de entrada do roteador é posto um demux com uma entrada (o enlace de entrada) e duas saídas (um enlace alternativo de saída de outra porta, com o qual o enlace de entrada será curto-circuitado caso seja detectado uma **falha permanente**, e a porta de entrada do roteador).

Em ambos os componentes, a seleção é feita através do sinal vindo do circuito detector de falhas, chamado link_sc.

A Figura 30 apresenta a implementação parcial de um chaveador, tomando como exemplo o curto circuito das portas *NORTH* e *SOUTH* de um roteador em falha.

Para roteadores cujo número de portas utilizadas é par (como é o caso dos roteadores dos vértices e do centro de uma topologia de rede em malha), não há problema enquanto ao curto circuito.

Contudo, para roteadores cujo número de portas é ímpar (como é o caso dos roteadores das arestas de uma topologia de rede em malha), sempre haverá um dos enlaces sem par para ser curto-circuitado. A solução proposta para esse impasse foi atuar nos sinais que realizam o controle de fluxo: aterrar o tx do enlace, para que o roteador vizinho não recebesse dados inválidos espuriamente; e referenciar em nível lógico alto o rx do enlace, para não causar contenção no roteador vizinho (efeito HOL em uma porta de entrada que se conecte a porta de saída desse enlace) e, conseqüentemente, na rede como um todo.

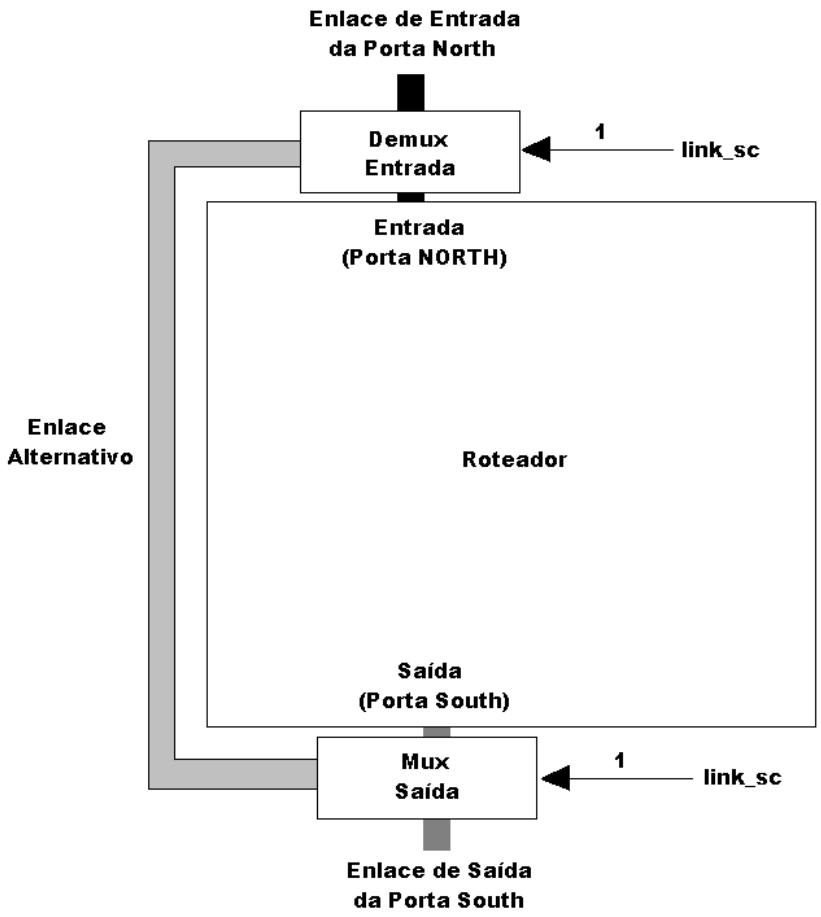

Figura 30 - Exemplo de implementação parcial do circuito chaveador entre duas portas de um roteador HermesFT.

4.4 A NoC HemesFT

Para essa proposta da NoC HermesFT foi também utilizada a topologia malha 2D, onde cada chave está ligada a um núcleo *IP* pela porta *LOCAL* e, dependendo da posição da chave na rede, a outras chaves com as demais portas, conforme a Figura 31. Apesar de esta topologia ter sido definida, a chave HermesFT, assim como sua antecessora, permite que outras topologias também sejam implementadas tais como toro e hipercubo. Para tanto, as chaves têm de ser interconectadas de forma a representar tais topologias e o algoritmo de roteamento tem de ser adequado.

A Figura 31 mostra a topologia adotada, bem como o realce de um dos nós da rede que agora são formados não só pelos roteadores, mas também pelos componentes discutidos nas seções anteriores.

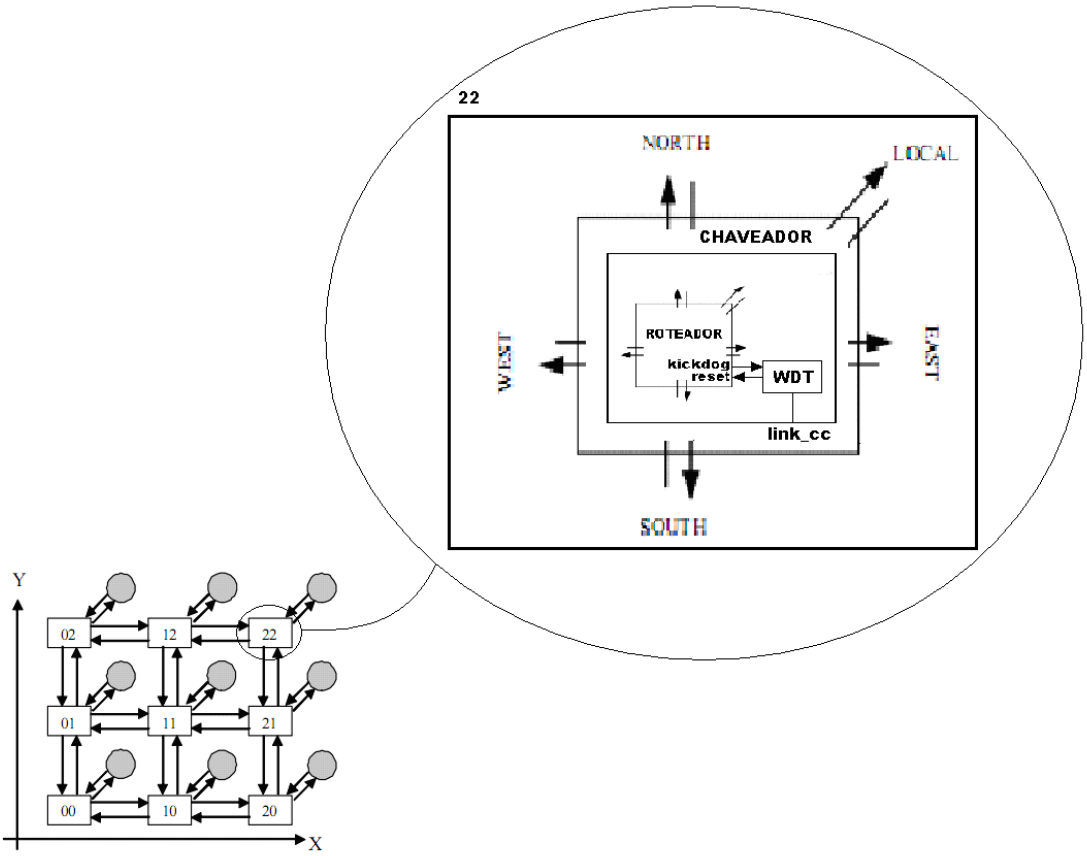

Figura 31 - Rede HermesFT 3 x 3 e realce de um nodo.

4.4.1 Tipos de Curto-Circuito e suas Conseqüências

A rede da Figura 31 é suficiente para representar o funcionamento de todos os números de roteadores em uma rede-em-chip HermesFT, pois possui os nove tipos diferentes de configurações de nós (quanto a localização, o que determina onde estão e quantos são seus vizinhos). Qualquer rede com mais ou menos nós terá um subconjunto desses nove tipos. Assim, será discutido, em seguida, como é realizado o curto-circuito dos enlaces para cada tipo.

CHAVEBL (BOTTOM-LEFT)

Similar a chave 00 da Figura 31, que será utilizada nos exemplos dados a seguir. Nesse caso, os enlaces das portas *NORTH* e *EAST* são curto-circuitados em caso de falha. Caso um roteador vizinho, digamos o 10, queira enviar um pacote ao roteador em falha, quem receberá o pacote é o 01. Quando receber o pacote e executar o algoritmo de roteamento, ocorrerá um conflito: a porta de saída será a mesma da porta de entrada. O fato anterior é a forma como as chaves vizinhas da chave em falha detectam que a mesma falhou. Isso porque

o algoritmo XY é dito estritamente progressivo, nunca podendo assim um pacote retornar pela mesma porta que entrou. Assim, o pacote em questão será descarregado não pela mesma porta que entrou, mas pela porta *GARBAGE*, apresentada anteriormente.

Tal fenômeno ocorrerá com todos os roteadores dos vértices, sendo assim não será apresentado nas seções dos mesmos, adiante.

CHAVEBC (BOTTOM-CENTER)

Similar a chave 10 da Figura 31, que será utilizada nos exemplos dados a seguir. Nesse caso, os enlaces das portas *WEST* e *EAST* são curto-circuitados em caso de falha. Tal decisão foi tomada após uma análise, caso a caso, de qual curto-circuito dos três possíveis (*WEST-EAST*, *WEST-NORTH*, *EAST-NORTH*) levaria a uma menor perda de rotas. O fato é que, além do *IP* do roteador em falha nunca receber pacotes, alguns outros *IPs* nunca receberão pacotes de outros (mas não de todos). Como exemplo, pode-se verificar que uma mensagem do roteador 20 nunca chegará ao roteador 11, mesmo que ambos não estejam em falha. A alternativa encontrada foi minimizar a quantidade de rotas perdidas.

Tal fenômeno ocorrerá com todos os roteadores de arestas, sendo assim não será apresentado nas seções dos mesmos, adiante.

CHAVEBR (BOTTOM-RIGHT)

Similar a chave 20 da Figura 31. Nesse caso, os enlaces das portas *WEST* e *NORTH* são curto-circuitados em caso de falha.

CHAVECL (CENTER-LEFT)

Similar a chave 01 da Figura 31. Nesse caso, os enlaces das portas *SOUTH* e *NORTH* são curto-circuitados em caso de falha.

CHAVECC (CENTER-CENTER)

Similar a chave 11 da Figura 31. Nesse caso, os enlaces das portas *SOUTH* e *NORTH*, são curto-circuitados em caso de falha. A mesma coisa é feita com os enlaces das portas *EAST* e *WEST*.

CHAVECR (CENTER-RIGHT)

Similar a chave 21 da Figura 31. Nesse caso, os enlaces das portas *SOUTH* e *NORTH* são curto-circuitados em caso de falha.

CHAVETL (TOP-LEFT)

Similar a chave 02 da Figura 31. Nesse caso, os enlaces das portas *SOUTH* e *EAST* são curto-circuitados em caso de falha.

CHAVETC (TOP-CENTER)

Similar a chave 12 da Figura 31. Nesse caso, os enlaces das portas *WEST* e *EAST* são curto-circuitados em caso de falha.

CHAVETR (TOP-RIGHT)

Similar a chave 22 da Figura 31. Nesse caso, os enlaces das portas *SOUTH* e *WEST* são curto-circuitados em caso de falha.

4.4.2 Parâmetros de *Timeout*

O timeout utilizado para a função de *watch dog timer* foi de cinco ciclos de *clock*. Tal valor foi obtido através de uma simples análise: o número de ciclos de *clock* que são necessários para que uma chave inicie o processo de requisitar o roteamento para envio de um pacote a partir da chegada de seu primeiro *flit* no buffer de entrada é um. A arbitragem pode durar até quatro ciclos, totalizando assim os cinco ciclos necessários.

Minimizando assim o tempo necessário para a detecção da falha, evita-se ao máximo a propagação da mesma pela rede.

4.5 Fragilidades Vencidas e Robustez Adquirida

Já de posse dos princípios de funcionamento da rede-em-chip HermesFT, será feita, nessa seção, uma lista das fragilidades (relacionadas à sua antecessora) que foram eliminadas e uma discussão da robustez adquirida com as mudanças já apresentadas.

4.5.1 Controle de Timeout no Envio

Foi implementado o controle de timeout sugerido no Capítulo 3, quando foi exposta tal fragilidade da NoC Hermes, resolvendo os problemas (comentados nessa ocasião) que poderiam ser causados por ela.

4.5.2 Controle de Início e/ou Fim de Pacote

Foi implementada, parcialmente, a técnica utilizada na rede-em-chip SoCIN (ZEFERINO, 2003b), isso porque apenas um dos bits utilizados, o bop, foi acrescentado. De qualquer forma, o problema citado no Capítulo 3 que ocorreria, pela falta de um sinalizador de início/fim do pacote não mais ocorrerá. Assim, o formato do pacote agora será como o apresentado na Figura 32.

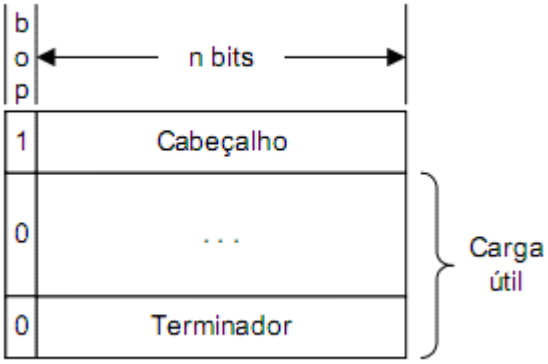

Figura 32 - Formato de pacote escolhido para a rede HermesFT.

Assim, se por conta de alguma falha, for perdido o sincronismo entre a máquina de estados do roteador e os *flits* do fluxo, o funcionamento voltará ao normal já no próximo cabeçalho detectado.

4.5.3 Falha Permanente

Em caso de falha permanente em uma chave, não mais ocorrerá os problemas expostos no Capítulo 3. Isso porque, após a detecção da falha, um número pré-definido de tentativas de *reset* será realizado no roteador. Caso a falha seja permanente, essas tentativas serão inúteis, fazendo assim com que os enlaces sejam curto-circuitados, isolando a chave em falha do restante da rede. Dessa forma, apenas o *IP* conectado à chave em falha será perdido da rede, sem onerar a comunicação do resto da mesma.

4.6 Conclusão

Após a implementação das alterações sugeridas, visando a melhoria do serviço de comunicação desempenhado pela NoC Hermes em cenários com falha, foram realizados testes em diferentes cenários para certificar a evolução proposta. Os resultados obtidos, bem como uma análise crítica dos mesmos, são apresentados no Capítulo 5.

5 Análise dos Resultados

Neste Capítulo é apresentada a metodologia utilizada para a realização dos testes, bem como os cenários utilizados para a realização dos mesmos e a comparação dos resultados obtidos com a rede-em-chip original Hermes e com a HermesFT, proposta nesse trabalho. Ao final do Capítulo são apresentadas conclusões que podem ser verificadas a partir dos resultados obtidos.

5.1 Metodologia Utilizada nos Testes

Para a realização dos testes foi utilizada a ferramenta ATLAS, cujo nome foi herdado de um projeto da PUCRS, também responsável pelo desenvolvimento da NoC Hermes. Tal ferramenta constitui-se de um framework (utilizando SystemC e VHDL) para parametrização e avaliação de desempenho de uma NoC Hermes. Como a interface da NoC HermesFT é idêntica à da sua antecessora, a ferramenta foi utilizada sem problemas para as simulações realizadas nesse trabalho.

Para um melhor entendimento do procedimento realizado nos testes, serão explicadas a seguir as etapas necessárias para a obtenção de resultados úteis (a partir da ferramenta ATLAS) para serem analisados e criticados.

5.1.1 Criação da Rede

A Figura 33 apresenta a interface gráfica da ferramenta ATLAS.

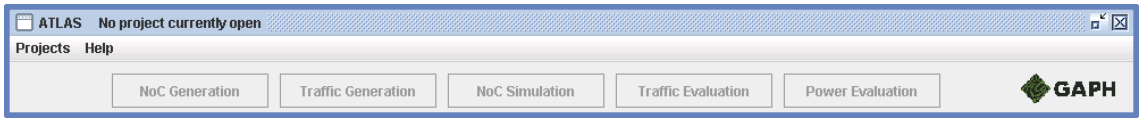

Figura 33 - Interface gráfica da ferramenta ATLAS.

Na criação do projeto, especificam-se as características da rede-em-chip desejada. A Figura 34 apresenta uma rede-em-chip Hermes 3 x 3, com *flits* de 16 bits, filas de entrada de 16 *flits*, algoritmo de roteamento XY e controle de fluxo *handshake*.

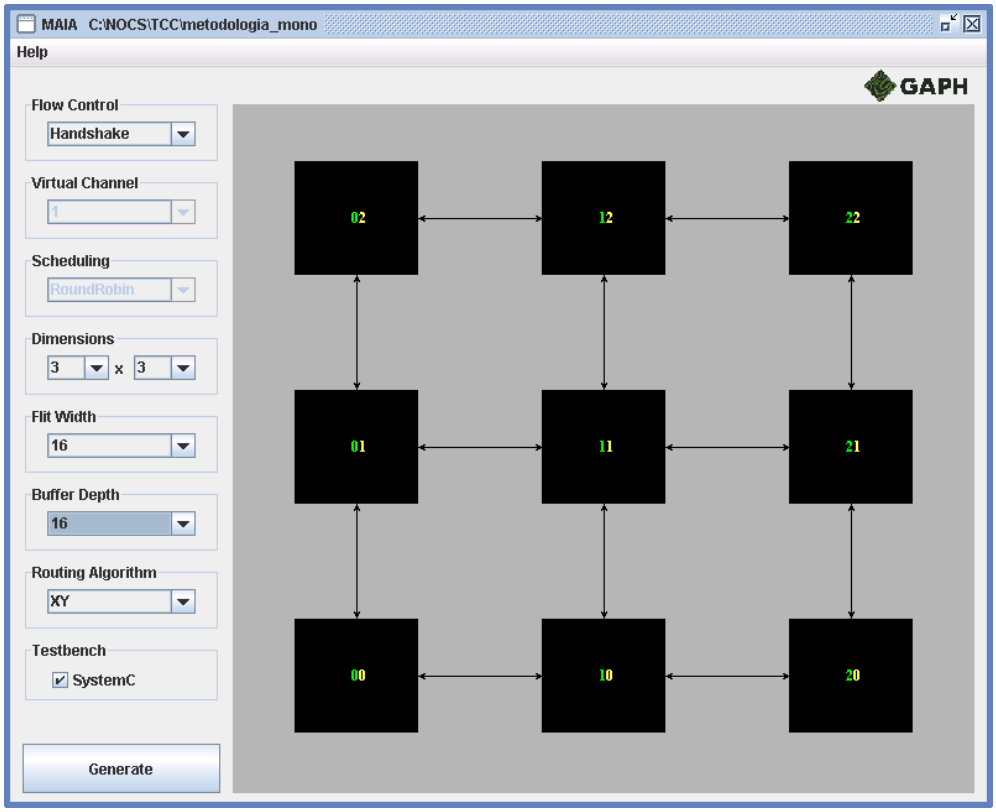

Figura 34 - Geração da NoC Hermes com as características especificadas.

5.1.2 Geração do Padrão de Tráfego

Na geração de tráfego é possível configurar o tráfego injetado por cada núcleo que está conectado às chaves. Na Figura 35 está sendo configurado o tráfego injetado pelo núcleo conectado à chave 00.

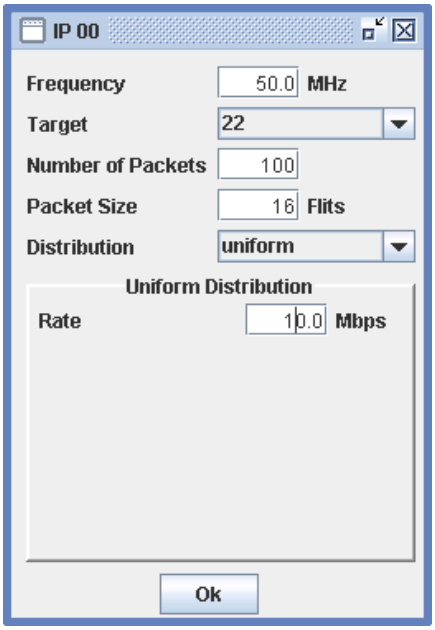

Figura 35 - Exemplo de configuração do tráfego gerado pelo core (ou *IP*) 00.

A freqüência de operação do núcleo é de 50MHz, e ele irá injetar 100 pacotes de 16 *flits* cada um, distribuídos uniformemente no tempo a uma taxa de 10Mbps com destino ao roteador 22.

Como resultados, são gerados arquivos como o apresentado na Figura 36.

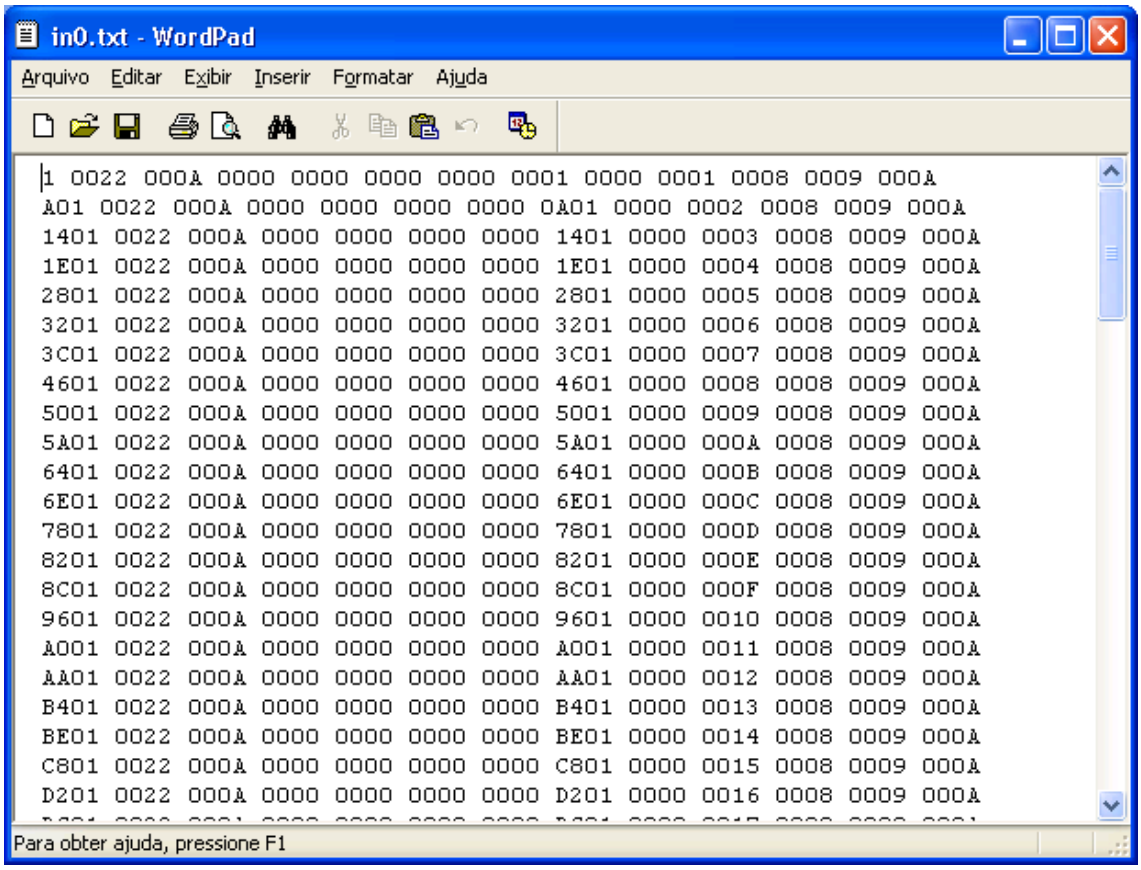

Figura 36 - Exemplo de arquivo gerado pelo padrão de tráfego especificado para o core (ou *IP*) 00.

Como se pode verificar, cada linha corresponde a um pacote. No início de cada linha há um *flit* adicional que é o instante em que o pacote deve ser inserido na rede (para obedecer à taxa requerida), em números de *clock*.

O segundo *flit* é o alvo do pacote, que no exemplo é o roteador 22. O terceiro *flit* é o tamanho do resto do pacote. O quarto *flit* é a fonte do pacote, que no exemplo é o 00. Os próximos 4 *flits* são os *flits* com o instante no qual o pacote deveria entrar na rede. Os próximos dois *flits* (nono e décimo) formam um número de seqüência, tornando possível assim a identificação de perdas de pacotes. Os três últimos *flits* não são utilizados para armazenar nenhuma informação importante e sempre contêm os valores 0008, 0009 e 000A, respectivamente.

Somando a quantidade de *flits* de uma linha pertencentes ao pacote, chega-se ao número 12. Como o pacote foi configurado para ter tamanho 16, faltam 4. Esses quatro *flits*

que não estão no arquivo serão inseridos pelo núcleo com o instante no qual o pacote realmente entrou na rede (possivelmente diferente do instante no qual o pacote deveria entrar por conta de contenção).

5.1.3 Simulação

A simulação é realizada através do software ModelSim®. Como resultados, são gerados arquivos de saída com os pacotes recebidos por cada núcleo. Além disso, são gerados, opcionalmente, para análise interna do fluxo dos pacotes na rede, arquivos para cada uma das portas de saída dos roteadores contendo os *flits* que passaram por ela.

Um exemplo de um arquivo com os pacotes recebidos por um núcleo é apresentado abaixo.

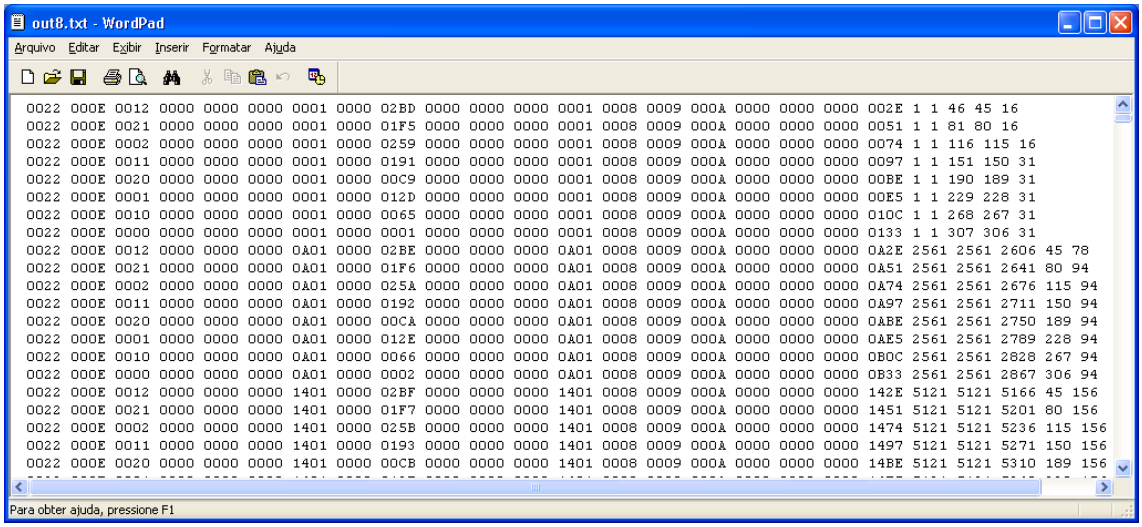

Figura 37 - Exemplo de arquivo com os *flits* recebidos pelo core (ou *IP*) 22.

O exemplo da Figura 37 é do núcleo 22 e cada linha corresponde a um pacote recebido pelo mesmo. Abaixo é explicado o significado de cada valor da linha.

a) 1º *flit*: Núcleo destino do pacote.

b) 2º *flit*: Tamanho do restante do pacote.

c) 3º *flit*: Núcleo emissor do pacote.

d) 4º-7º *flits*: Instante no qual o pacote deveria entrar na rede.

e) 8º e 9º *flits*: Número de sequência do pacote.

f) 10º-13º *flits*: Instante no qual o pacote efetivamente entrou na rede.

g) 14º-16º *flits*: Dados não utilizados.

h) 17º-20º *flits*: Instante no qual o pacote saiu da rede.

i) 1º decimal: Instante no qual o pacote deveria entrar na rede, em decimal.

j) 2º decimal: Instante no qual o pacote deveria entrou na rede, em decimal.

k) 3º decimal: Instante no qual o pacote saiu da rede.

l) 4º decimal: Latência do pacote (instante de saída – instante de entrada).

m) 5º decimal: Instante, em ms, de simulação.

Um exemplo de um arquivo com os pacotes enviado por uma das portas de um roteador é apresentado abaixo.

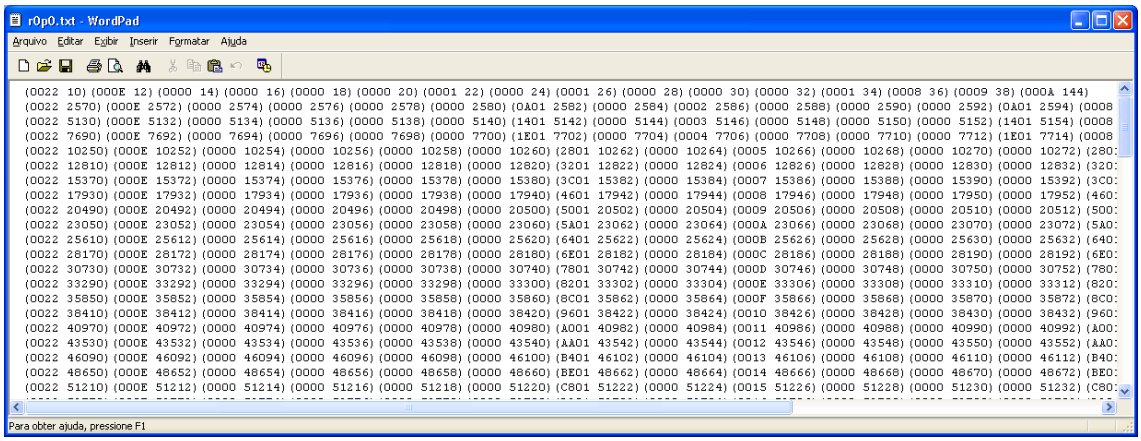

Figura 38 - Exemplo de arquivo gerado pela simulação com os *flits* enviados pela porta de saída *EAST* do roteador 00.

O exemplo da Figura 38 é da porta *EAST*, do roteador 00. Abaixo é explicado o conteúdo do arquivo.

Nesse tipo de arquivo é armazenado cada *flit* enviado pela porta, junto do instante em que o mesmo foi enviado. O formato utilizado para armazenar as duas informações juntas é (FFFF d), com FFFF representando os quatro *nibbles* de cada *flit*, seguido de um espaço em branco e d, um valor decimal representando o instante no qual aquele *flit* foi enviado ao próximo roteador.

De posse de todas essas informações é possível realizar várias análises, como cálculo de vazão e latência de um núcleo a outro. Para poupar tempo dos interessados, foi criada, pela equipe do GAPH da PUCRS, uma ferramenta chamada ATLAS, que automatiza esse trabalho. Tal ferramenta será apresentada na seção seguinte.

5.1.4 Análise de Tráfego

De posse de toda essa informação contida nos arquivos citados anteriormente, a ferramenta de análise de tráfego calcula informações como vazão e latência de cada pacote de um núcleo a outro. Além disso, calcula também: a quantidade de *flits* transmitidos em cada porta de saída de cada roteador; o número de ciclos médio por *flit*; taxa de utilização de canal; e a vazão entre dois roteadores.

O exemplo apresentado na Figura 39 ilustra a análise da quantidade de *flits* enviados por cada porta de saída. Para tal análise, a ferramenta disponibiliza uma análise textual e uma gráfica.

A Figura 39 apresenta a análise textual.

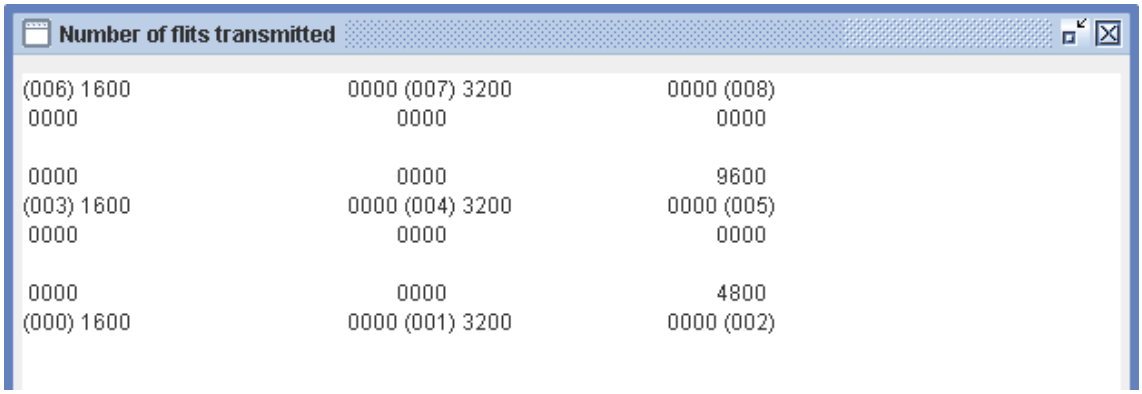

Figura 39 - Análise textual do número de *flits* enviados pelas portas de saída de todos os roteadores de uma rede Hermes 3 x 3.

A regra para a identificação dos roteadores muda nessa representação. Entre parênteses é posto o resultado da operação x + 3y (onde x é o valor da ordenada da posição do roteador e y o valor da abscissa). Assim, o roteador 22 é representado como 2 + 3.2 = 8.

Já a Figura 40 apresenta a análise gráfica dos mesmos dados. Nessa análise, os roteadores estão nas posições 2x e 2y. Por exemplo, as coordenadas, no gráfico, do roteador 11 é o par ordenado $(x_{11}, y_{11}) = (2.1, 2.1) = (2, 2)$.

O diferencial entre a análise gráfica e a textual é que na primeira os fluxos de entrada e saída são somados para representar a quantidade de *flits* injetados no enlace (de entrada e saída). Na segunda forma, essas informações são apresentadas separadamente.

Esse trabalho utilizará somente a análise do número de *flits* transmitidos, mostrada como exemplo acima, pois seu foco é a analisar a confiabilidade e disponibilidade da arquitetura de comunicação.

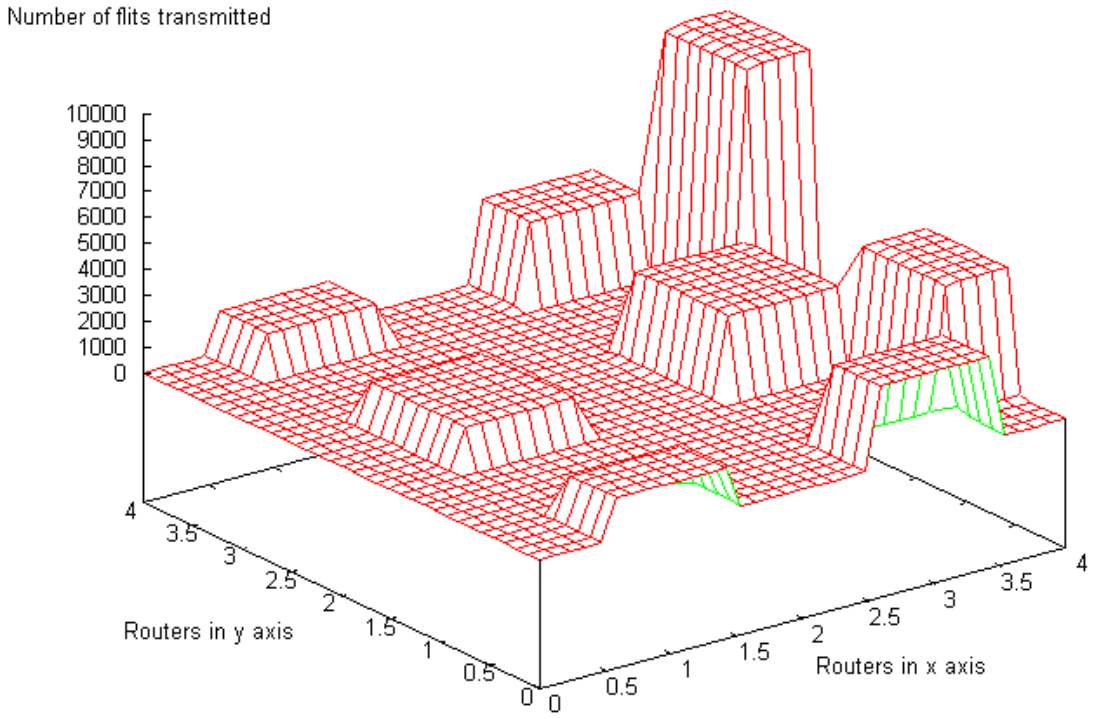

Figura 40 - Análise gráfica do número de *flits* que percorreram cada enlace de uma rede Hermes 3 x 3.

5.2 Cenários Utilizados nos Testes

Todos os cenários escolhidos têm o mesmo padrão de tráfego (controlado, de forma a facilitar o entendimento dos comportamentos), com todos os núcleos enviando o mesmo número de pacotes ao roteador alvo.

O tráfego de todos os núcleos tem 100 pacotes (de 16 *flits* cada) os quais o alvo é o núcleo 22. A taxa de bits é de 10 Mbps e também igual para todos.

Os cenários escolhidos para serem utilizados nos testes foram quatro: a rede com nenhum roteador em falha; a rede com um roteador (que não seja o alvo) com falhas transitórias; a rede com um roteador (que não seja o alvo) com falha permanente; e a rede com o roteador alvo em falha permanente. Todos os cenários serão simulados nas duas redes-em-chip – a Hermes e a HermesFT e os resultados comparados em seguida.

5.3 Geração de Falhas

O mecanismo de geração de falhas foi implementado dentro das próprias chaves com um módulo sabotador acoplado ao circuito das mesmas. O mesmo é configurável e pode gerar

até três tipos pré-definidos de falha. O arquivo apresentado abaixo, no formato INI é utilizado como entrada para o módulo sabotador.

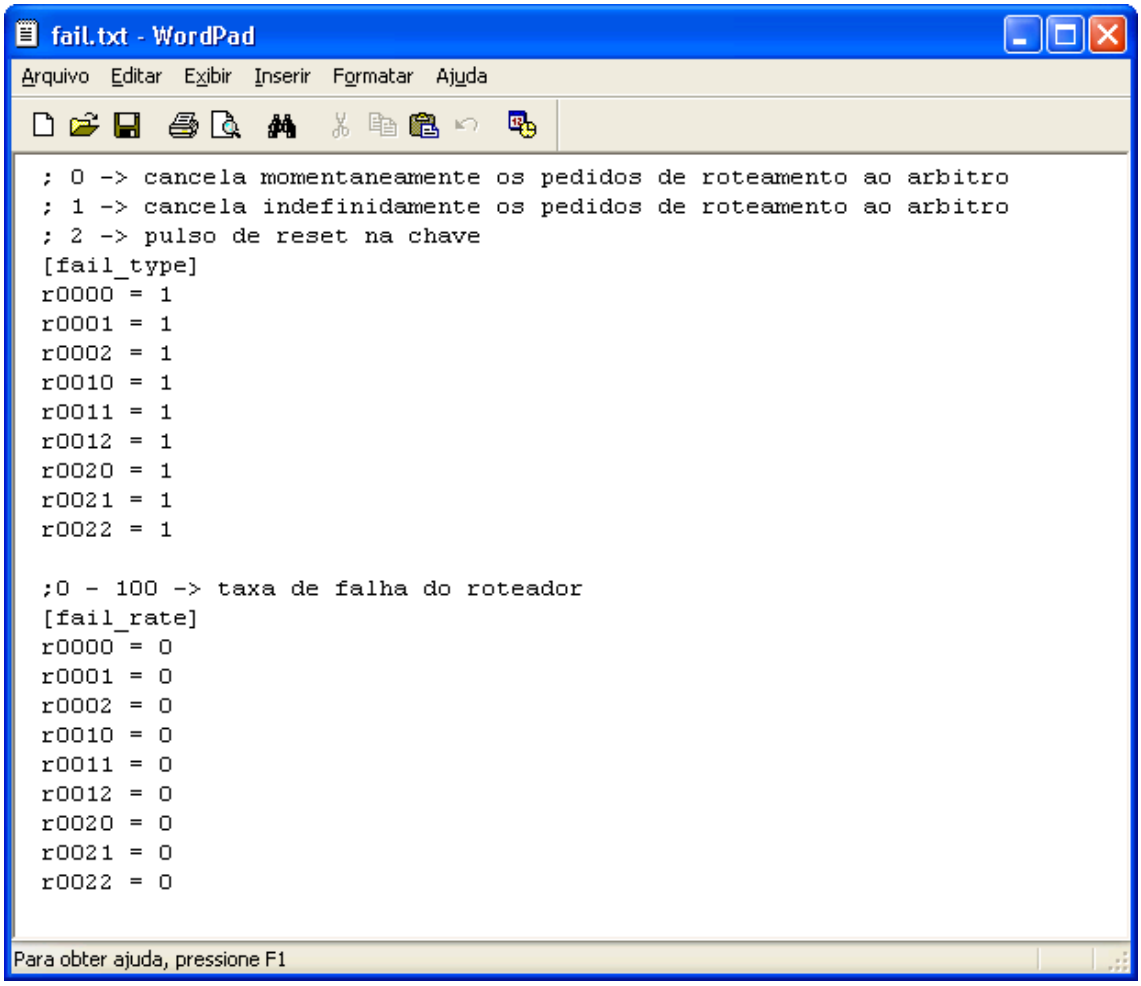

Figura 41 - Exemplo de conteúdo do arquivo de configuração do módulo sabotador.

Como se pode verificar no arquivo mostrado na Figura 41, a taxa de falha de cada roteador é também um item configurável. No exemplo, por as taxas de falha de todos os roteadores estarem em zero, nenhuma falha ocorrerá. Caso estivesse diferente de zero, a falha seria gerada aleatoriamente com uma probabilidade de acontecer igual à taxa configurada.

Na Figura 42 é apresentado o diagrama de blocos dos módulos envolvidos no processo de injeção de falhas no roteador.

O módulo VHDL é inserido no circuito de cada roteador. Recebe os sinais reset e h do roteador e, de acordo com os valores dos sinais force_reset e force_hstuck, modifica ou não o valor dessas entradas e disponibiliza o resultado nas saídas reset_r e h_r, respectivamente. Essas saídas são utilizadas para a injeção da falha transitória e permanente, respectivamente.

O módulo Saboteur, implementado em SystemC, tem como entrada o arquivo fail.txt apresentado anteriormente e gera as falhas aleatoriamente, no tempo, de acordo com os parâmetros configurados no arquivo.

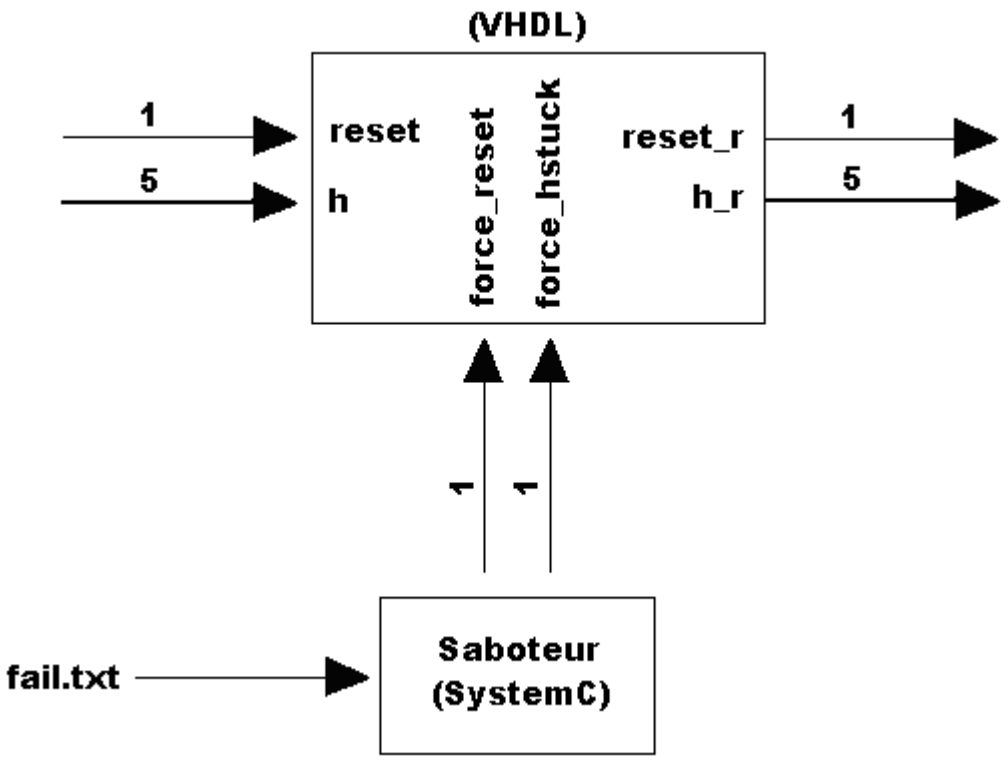

Figura 42 - Diagrama de blocos do processo de injeção de falhas no roteador.

5.4 Apresentação dos Resultados

Os resultados obtidos para cada cenário serão apresentados nas seções dedicadas a seguir. Nenhuma análise crítica é realizada. Tal tarefa foi postergada para a Seção 5.5 desse capítulo, onde são apresentadas as conclusões retiradas dos resultados obtidos em cada cenário.

5.4.1 Cenário 1: Sem Falhas

REDE HERMES

A Figura 43 e a Figura 44 têm a representação textual e gráfica, respectivamente, da contabilização dos pacotes que cada roteador enviou por cada porta.

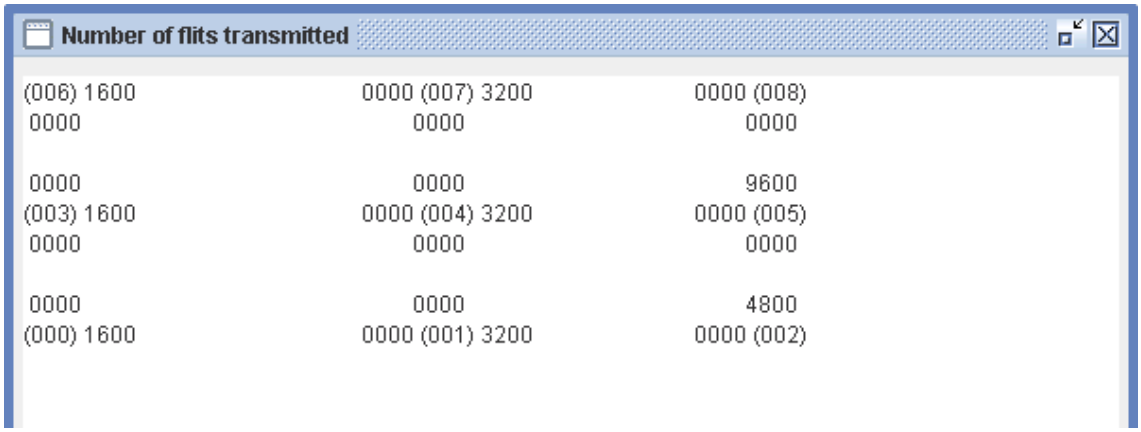

Figura 43 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede Hermes no cenário 1.

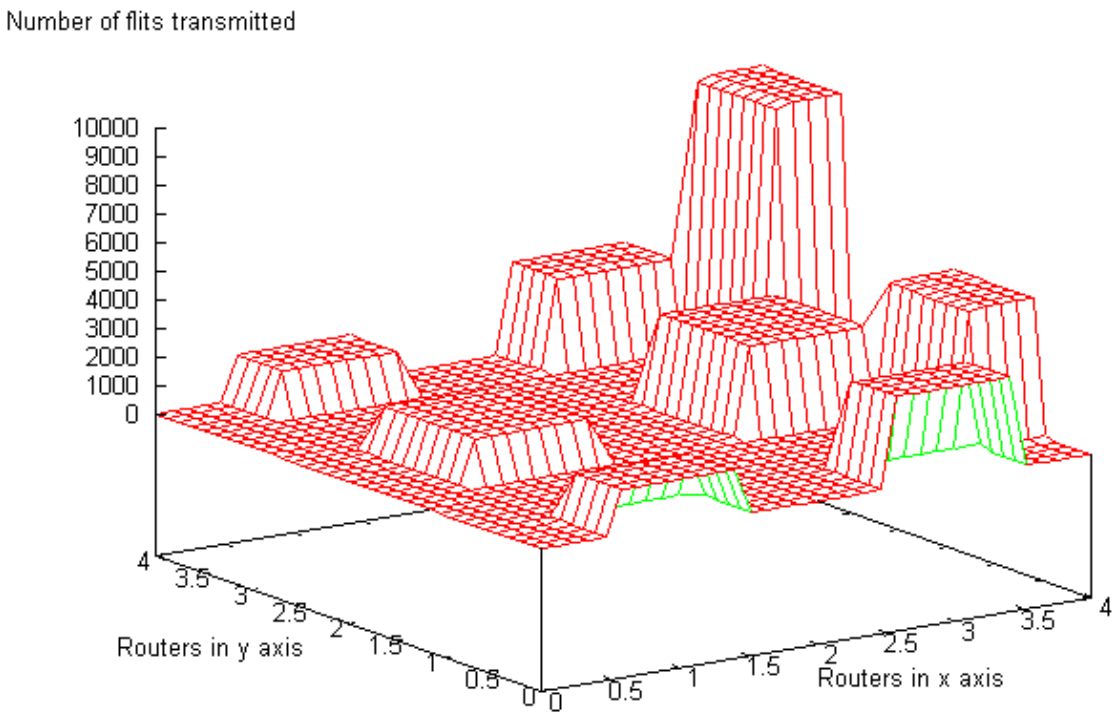

Figura 44 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede Hermes no cenário 1.

REDE HERMESFT

A Figura 45 e a Figura 46 têm a representação textual e gráfica, respectivamente, da contabilização dos pacotes que cada roteador enviou por cada porta.

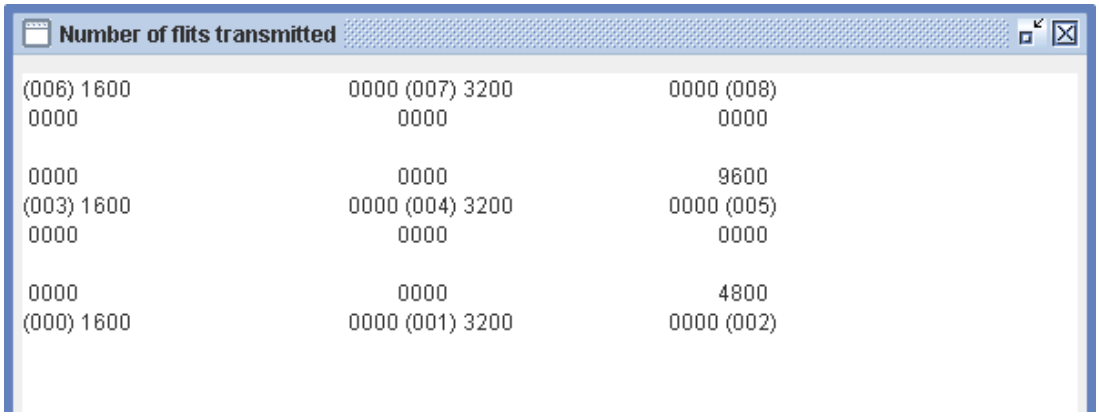

Figura 45 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede HermesFT no cenário 1.

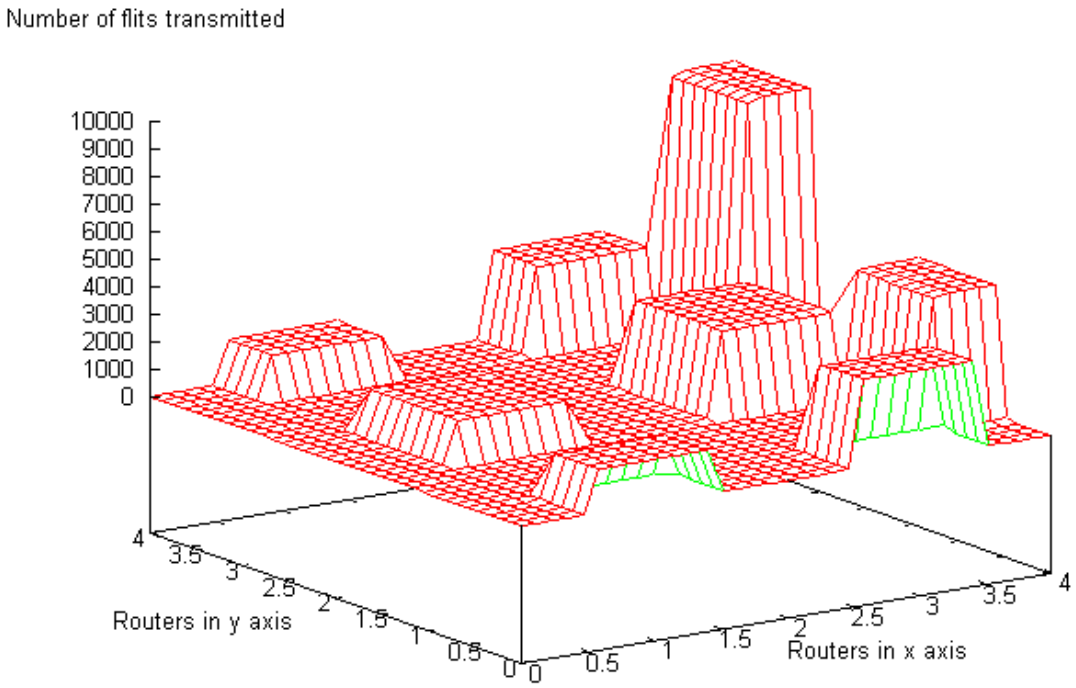

Figura 46 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede HermesFT no cenário 1.

5.4.2 Cenário 2: Roteador Não-Alvo com Falhas Transitórias

REDE HERMES

Na simulação realizada, a falha simulada foi a aplicação de um pulso de reset no roteador 00, no meio da transmissão de um pacote, mais especificamente no pacote de número 17. Na Figura 47 é apresentado o arquivo com os *flits* de saída da porta *EAST* do roteador, comprovando a falha.

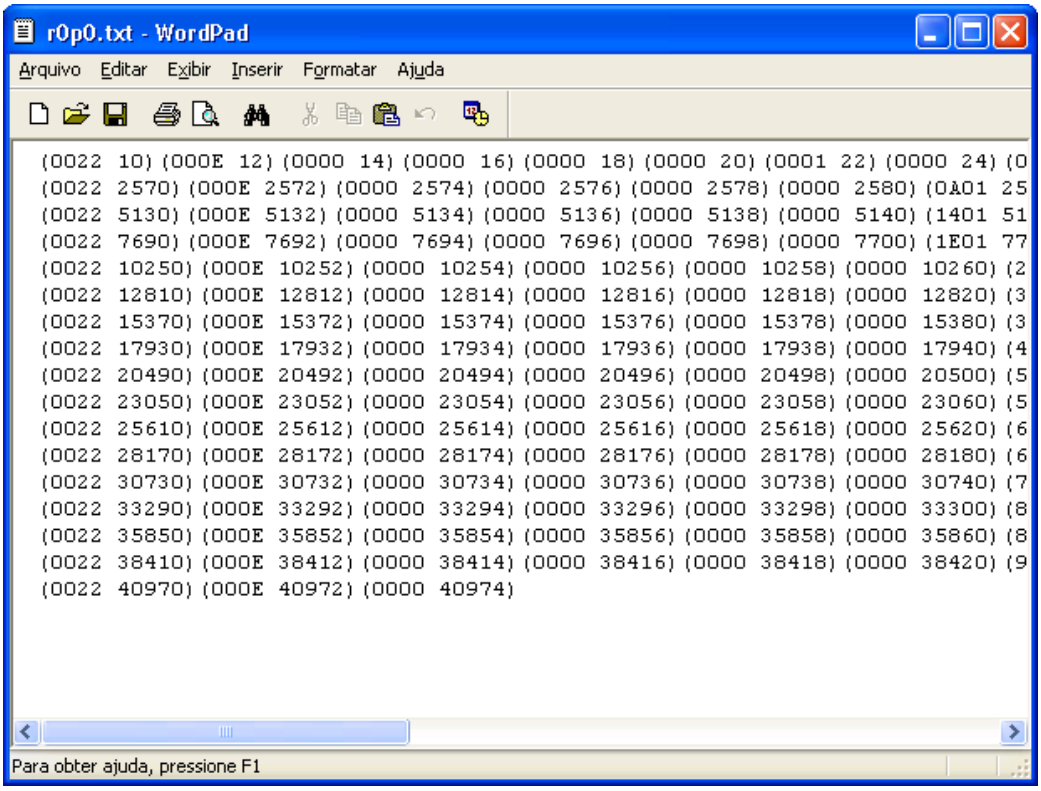

Figura 47 - Arquivo com os *flits* de saída da porta *EAST* do roteador 00.

Para ilustrar o efeito, na rede, de tal falha é apresentado, na Figura 48, também o arquivo com os *flits* de saída da porta *EAST* do roteador 10.

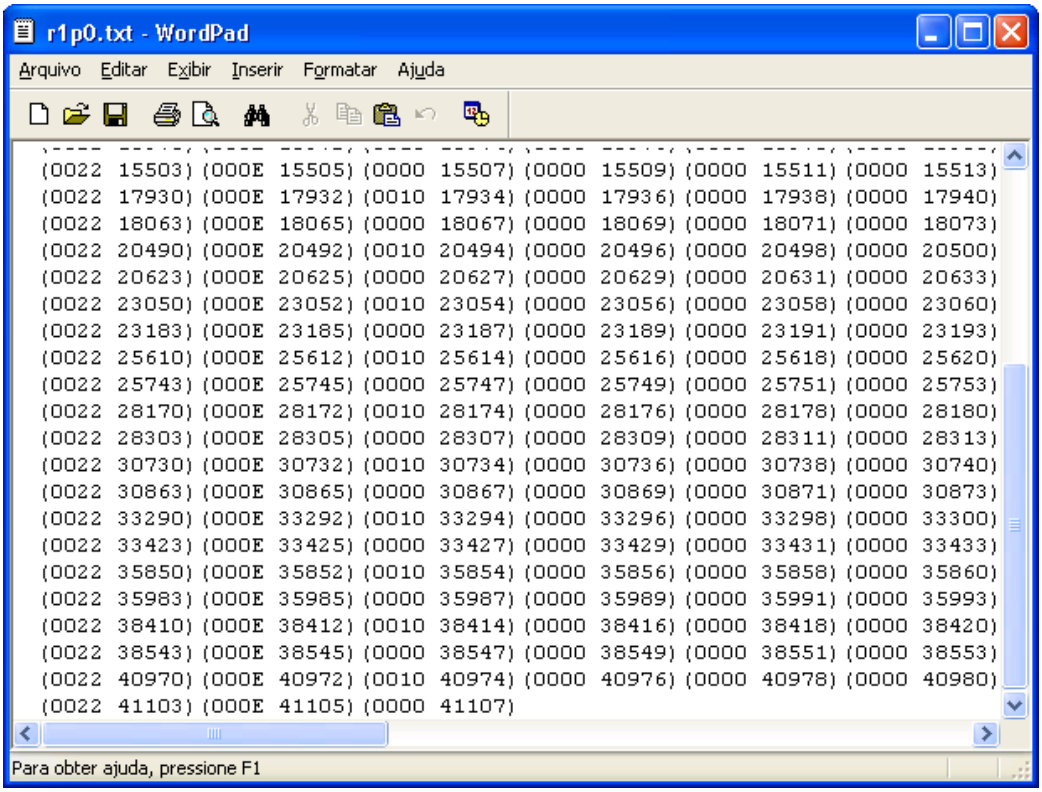

Figura 48 - Arquivo com os *flits* de saída da porta *EAST* do roteador 10.

A Figura 49 e a Figura 50 têm a representação textual e gráfica, respectivamente, da contabilização dos pacotes que cada roteador enviou por cada porta.

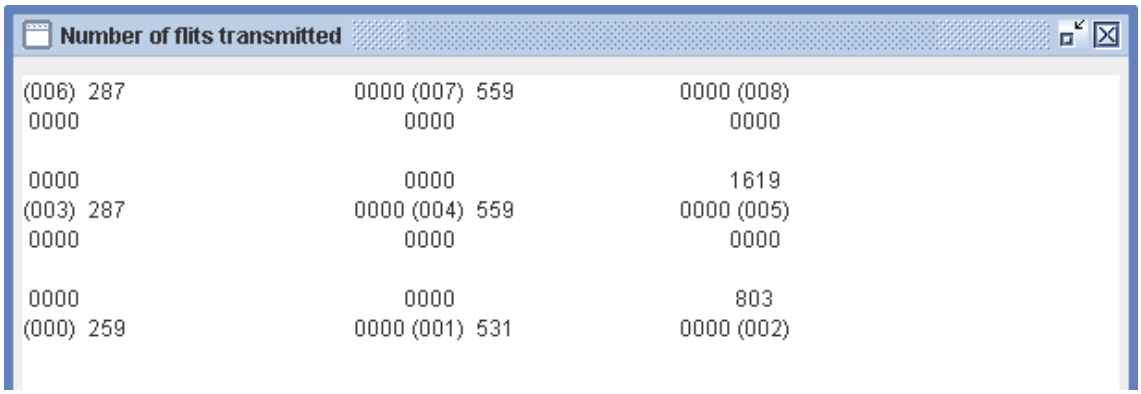

Figura 49 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede Hermes no cenário 2.

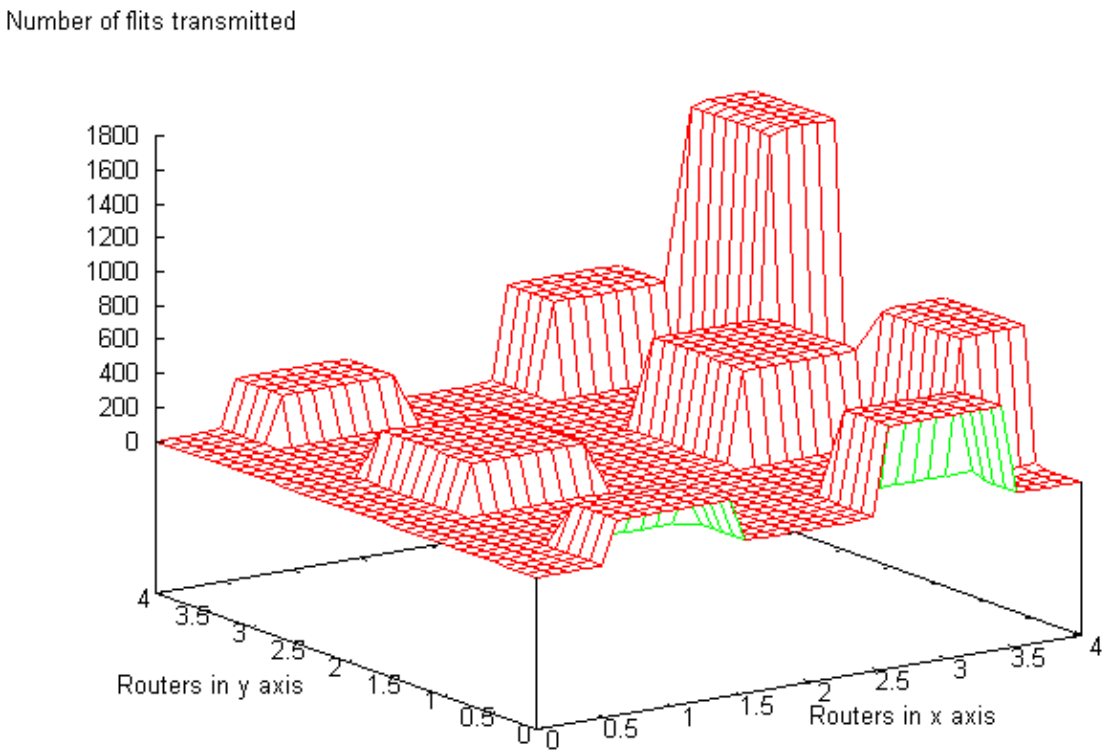

Figura 50 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede Hermes no cenário 2.

REDE HERMESFT

Na simulação realizada, a falha simulada foi a aplicação de um pulso de reset no roteador 00, no meio da transmissão de um pacote, mais especificamente no pacote de número

17. Na Figura 51 é apresentado o arquivo com os *flits* de saída da porta *EAST* do roteador, comprovando a falha.

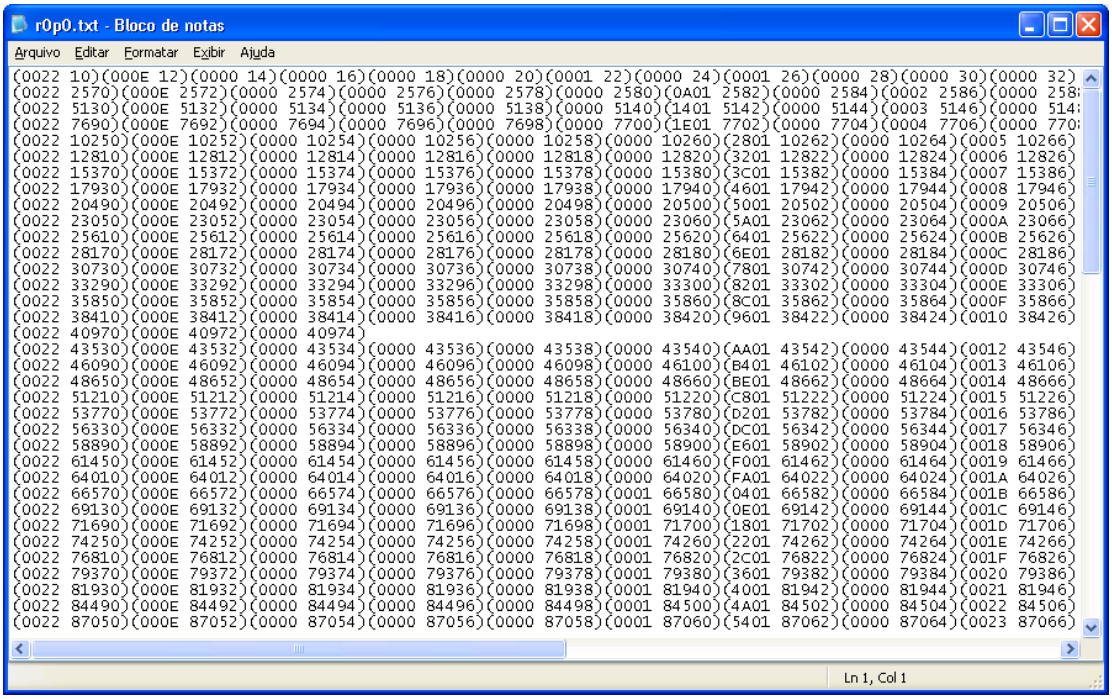

Figura 51 - Arquivo com os *flits* de saída da porta *EAST* do roteador 00.

Para ilustrar o efeito, na rede, de tal falha é apresentado, na Figura 52, também o arquivo com os *flits* de saída da porta *EAST* do roteador 10.

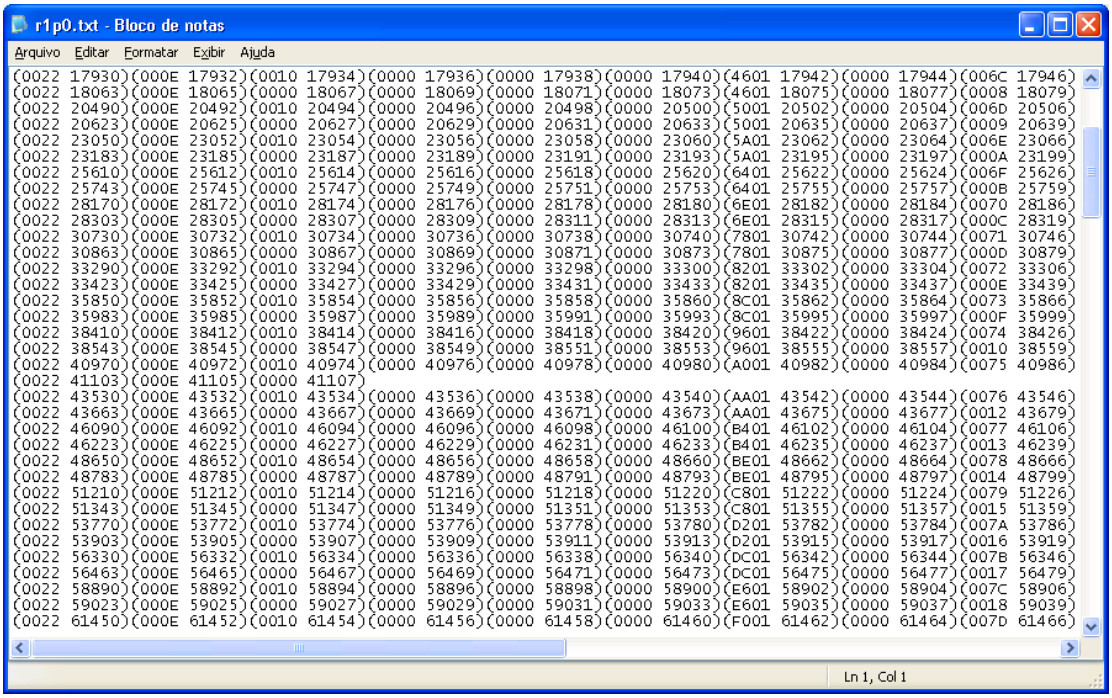

Figura 52 - Arquivo com os *flits* de saída da porta *EAST* do roteador 10.

A Figura 53 e a Figura 54 têm a representação textual e gráfica, respectivamente, da contabilização dos pacotes que cada roteador enviou por cada porta.

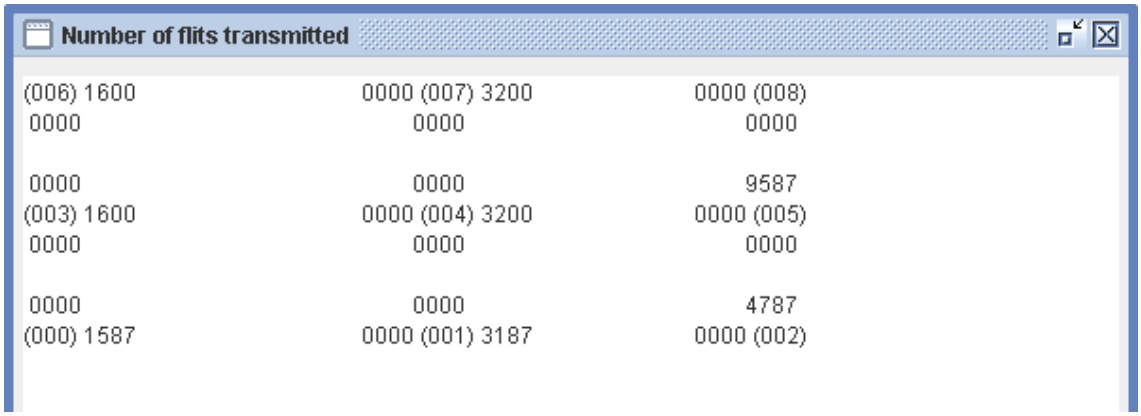

Figura 53 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede HermesFT no cenário 2.

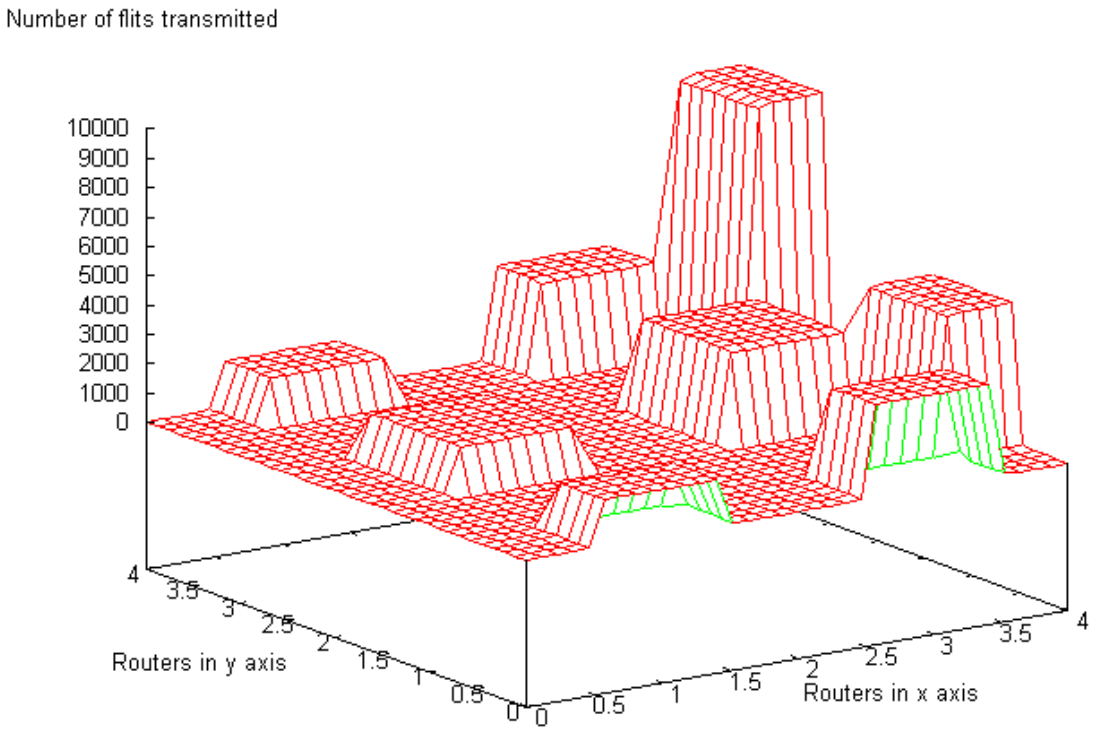

Figura 54 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede HermesFT no cenário 2.

5.4.3 Cenário 3: Roteador Não-Alvo com Falha Permanente

REDE HERMES

Na simulação realizada, a falha foi simulada no roteador 20: os sinais de recepção dos pedidos de roteamento enviados ao árbitro foram grampeados em nível lógico baixo, já no início da simulação, antes mesmo do primeiro pedido. A Figura 55 e a Figura 56 têm a representação textual e gráfica, respectivamente, da contabilização dos pacotes que cada roteador enviou por cada porta.

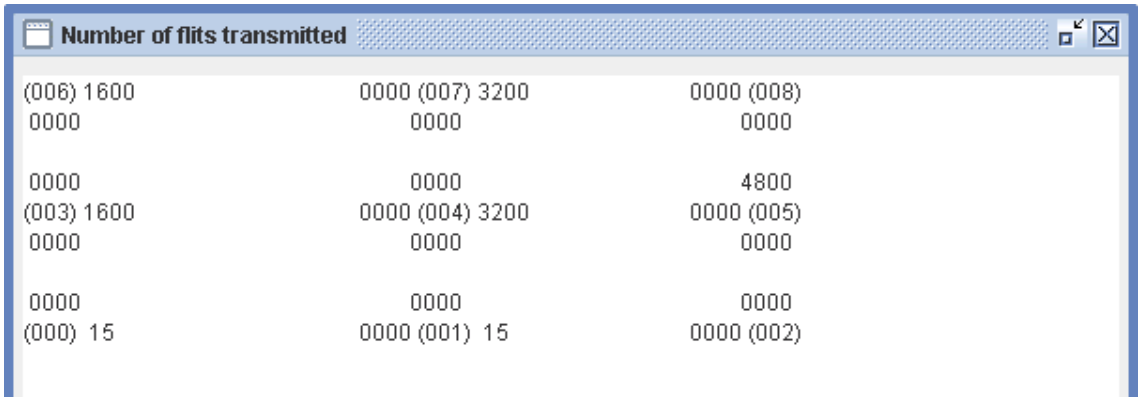

Figura 55 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede Hermes no cenário 3.

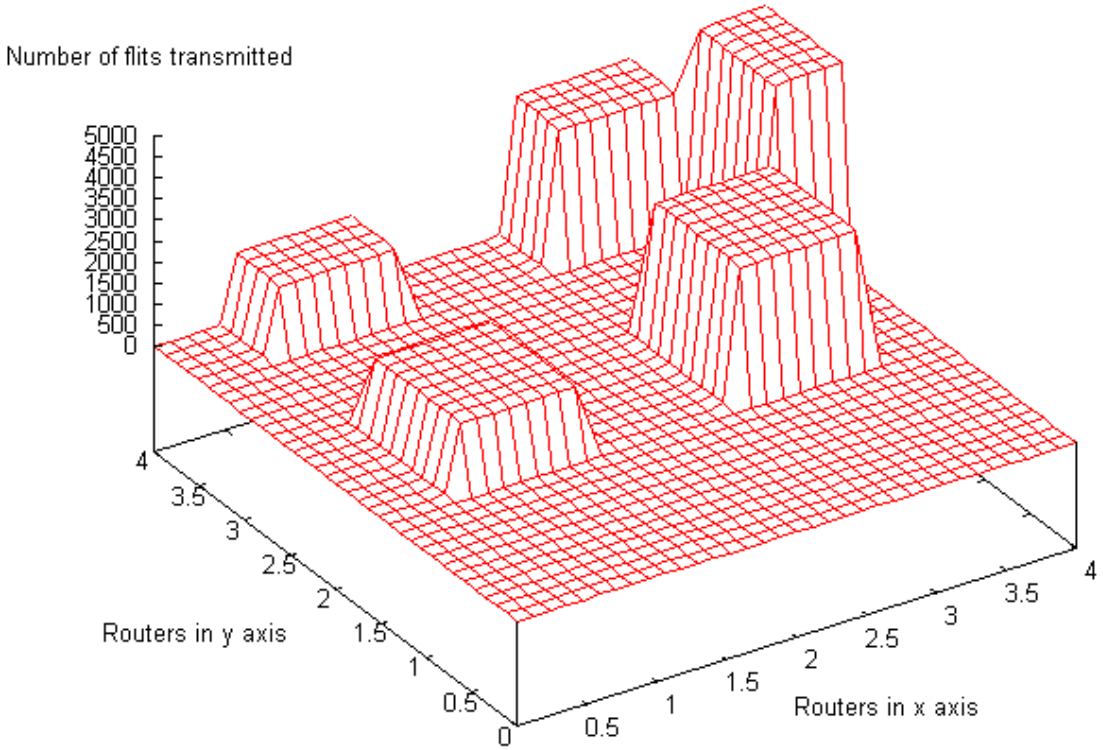

Figura 56 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede Hermes no cenário 3.

REDE HERMESFT

Na simulação realizada, a falha foi simulada no roteador 20: os sinais de recepção dos pedidos de roteamento enviados ao árbitro foram grampeados em nível lógico baixo, já no início da simulação, antes mesmo do primeiro pedido. A Figura 57 e a Figura 58 têm a representação textual e gráfica, respectivamente, da contabilização dos pacotes que cada roteador enviou por cada porta.

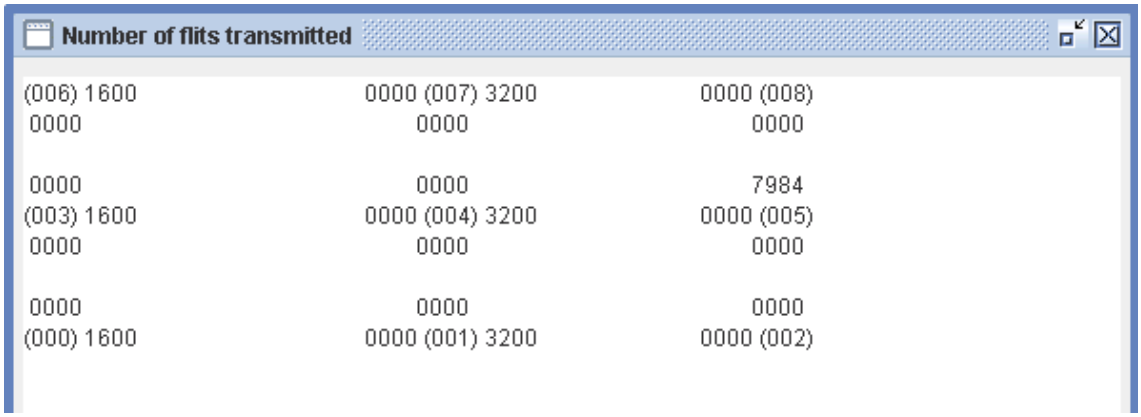

Figura 57 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede HermesFT no cenário 3.

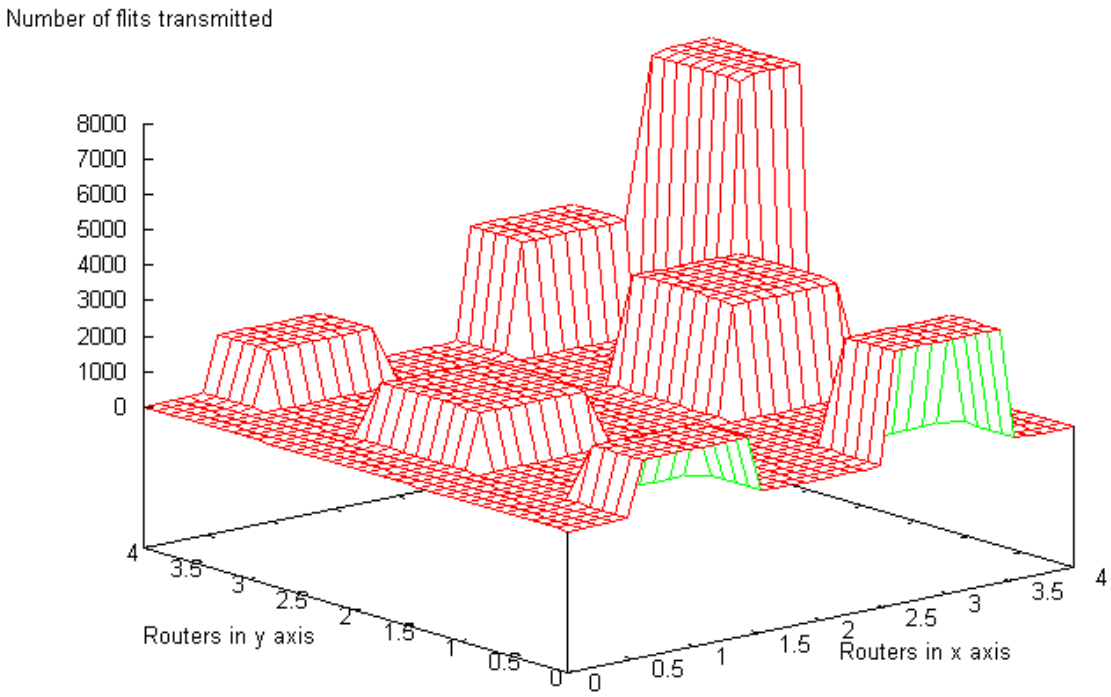

Figura 58 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede HermesFT no cenário 3.

5.4.4 Cenário 4: Roteador Alvo com Falha Permanente

REDE HERMES

Na simulação realizada, a falha foi simulada no roteador 22: os sinais de recepção dos pedidos de roteamento enviados ao árbitro foram grampeados em nível lógico baixo, já no início da simulação, antes mesmo do primeiro pedido. A Figura 59 e a Figura 60 têm a representação textual e gráfica, respectivamente, da contabilização dos pacotes que cada roteador enviou por cada porta.

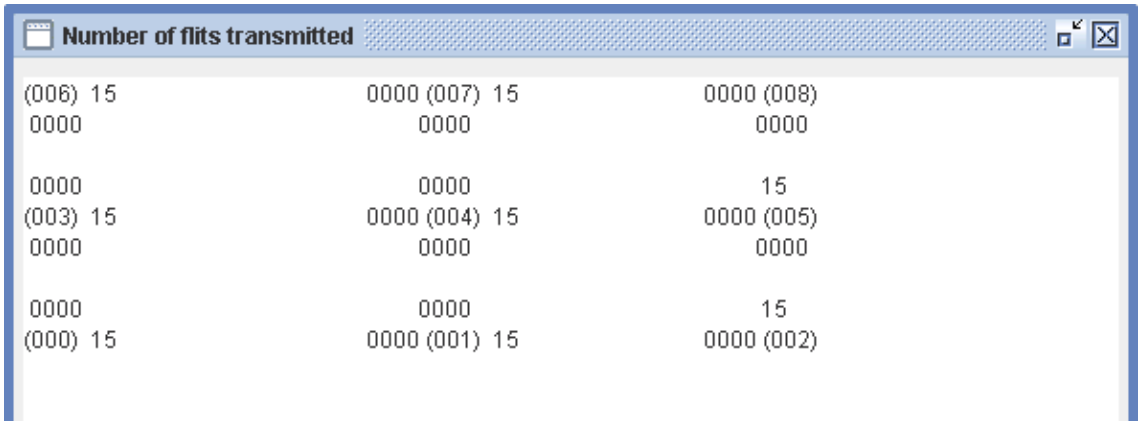

Figura 59 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede Hermes no cenário 4.

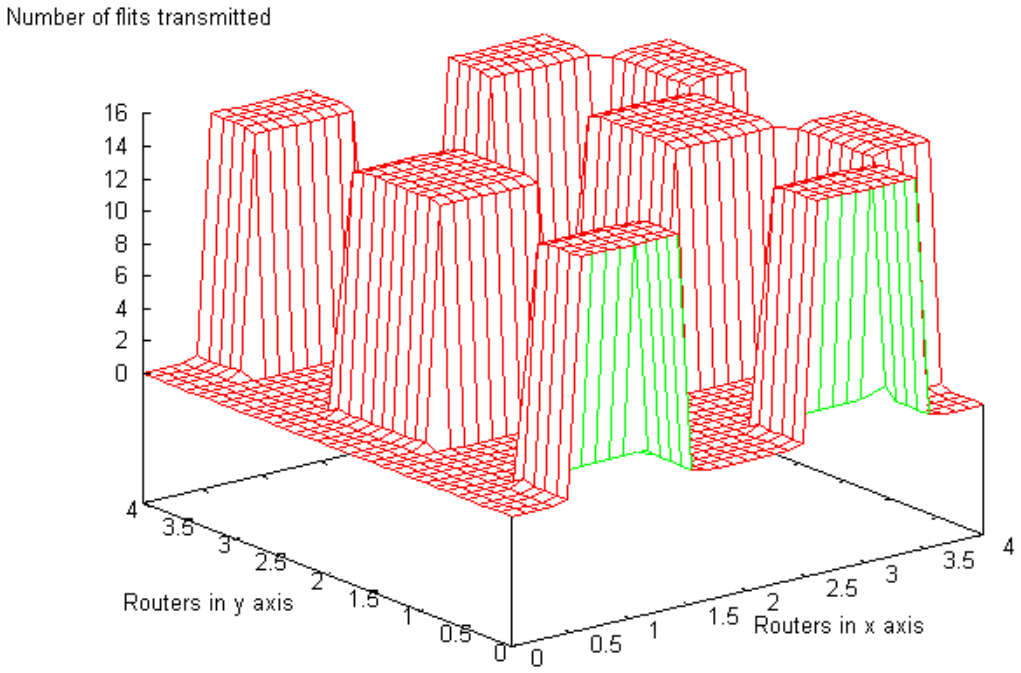

Figura 60 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede Hermes no cenário 4.

REDE HERMESFT

Na simulação realizada, a falha foi simulada no roteador 22: os sinais de recepção dos pedidos de roteamento enviados ao árbitro foram grampeados em nível lógico baixo, já no início da simulação, antes mesmo do primeiro pedido.

Na Figura 61 e na Figura 62 são apresentados os arquivos com os *flits* que foram enviados à porta *GARBAGE* dos roteadores 12 e 21 – vizinhos do roteador em falha os quais receberão os pacotes do mesmo após o curto-circuito dos enlaces.

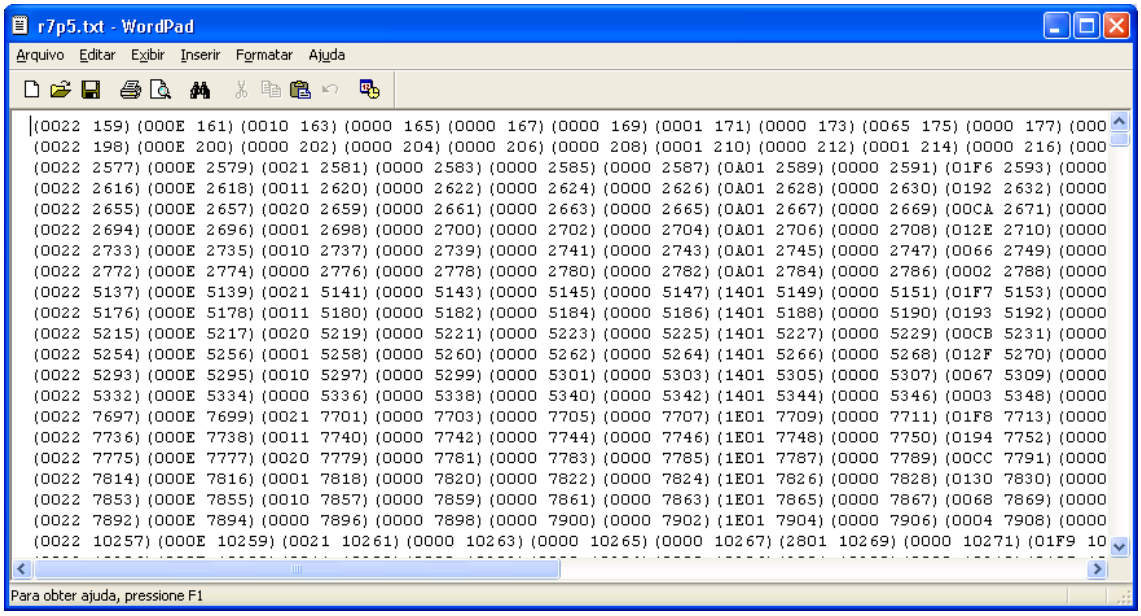

Figura 61 - Arquivo com os *flits* enviados a porta GARBAGE do roteador 12.

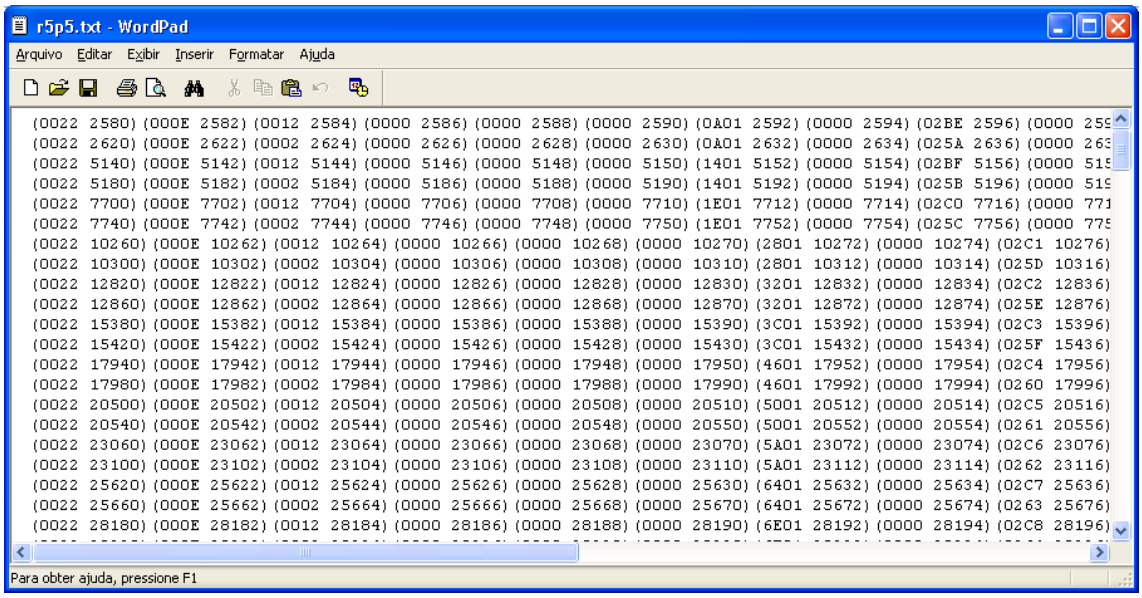

Figura 62 - Arquivo com os *flits* enviados a porta GARBAGE do roteador 21.

A Figura 63 e a Figura 64 têm a representação textual e gráfica, respectivamente, da contabilização dos pacotes que cada roteador enviou por cada porta (a porta *GARBAGE* não está incluída em nenhuma das duas representações).

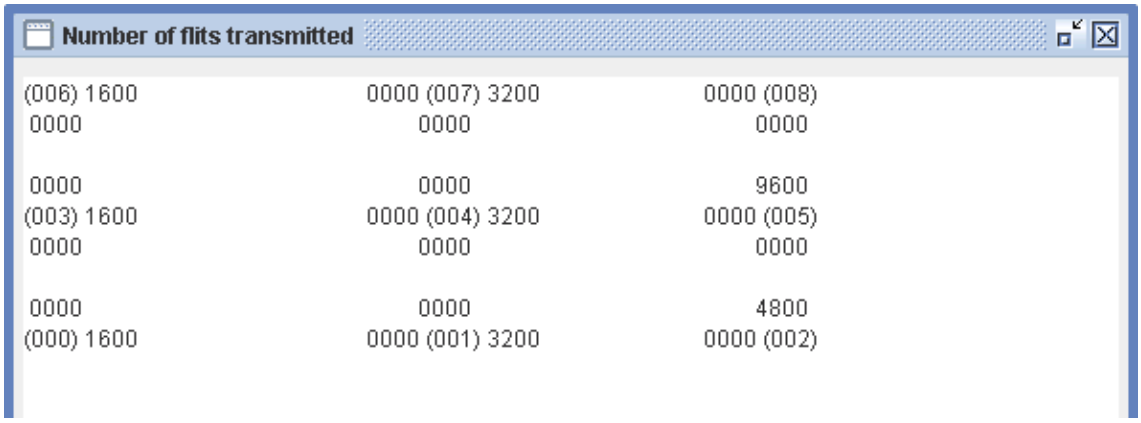

Figura 63 - Representação textual do número de *flits* enviados pelas portas de saída de todos os roteadores da rede HermesFT no cenário 4.

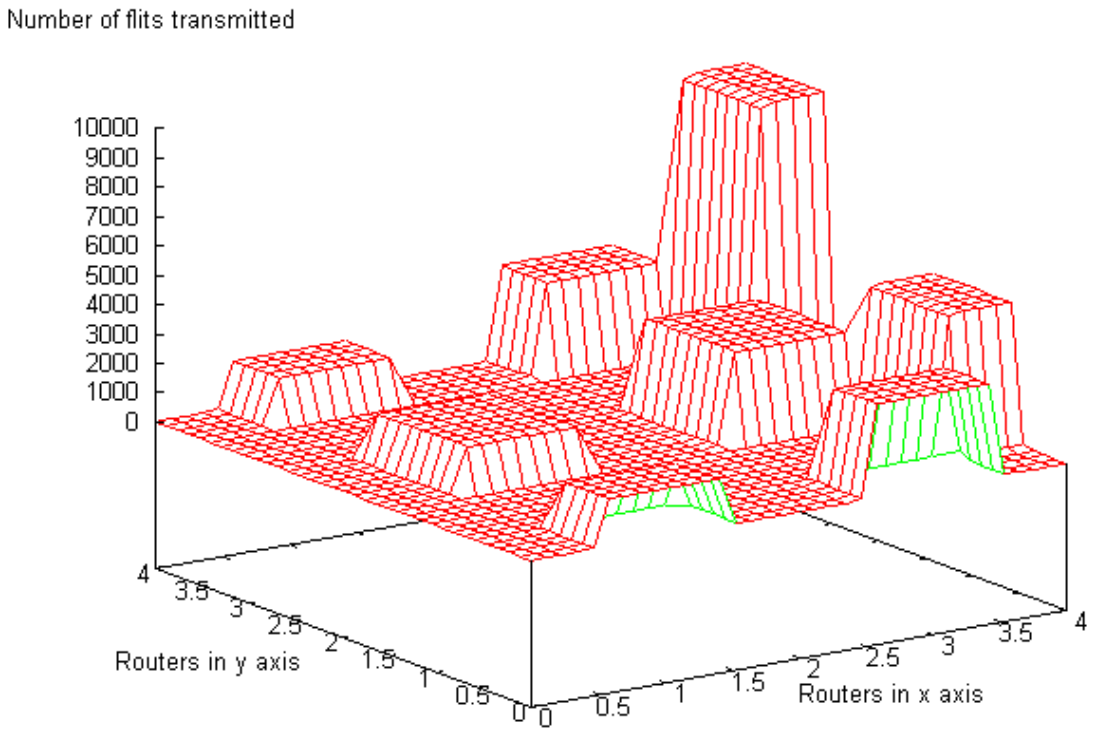

Figura 64 - Representação gráfica do número de *flits* que percorreram todos os enlaces da rede HermesFT no cenário 4.

5.5 Conclusões Sobre os Resultados

As conclusões que são expostas nessa seção foram obtidas através da análise dos gráficos, dos arquivos de saída e formas de onda dos sinais envolvidos. Como apenas os gráficos e os arquivos foram apresentados na seção anterior, quando for necessária informação adicional para o entendimento de alguma conclusão, será feito um destaque da porção importante da(s) forma(s) de onda. Ao final da apresentação das conclusões para cada cenário, tem-se uma tabela na forma de um resumo comparativo entre o desempenho das redes em cada cenário.

5.5.1 Cenário 1

No cenário sem falhas, nenhuma surpresa. Ambas as redes proveram o serviço desejado. O número de *flits* enviados pelas portas de saída aumenta à medida que se aproxima do roteador 22. Como exemplo para discussão, será feita a análise do fluxo dos roteadores 00, 10, 20 e 12.

Da porta *EAST* do roteador 00 saem 1600 *flits*, o que era o esperado, pois o núcleo 00 envia 100 pacotes, de 16 *flits* cada, para o núcleo 22.

Já da porta *EAST* do roteador 10 saem 3200 *flits*, o que também era o esperado, pois o núcleo 10 envia 100 pacotes, de 16 *flits* cada, para o núcleo 22 e além desses, o roteador 10 precisa ainda encaminhar os pacotes do roteador 00 para a mesma porta. Assim, chega-se a conta de 200 pacotes, de 16 *flits* cada, totalizando 3200 *flits*.

A mesma análise vale para os roteadores 20 e 12. O primeiro precisa ejetar pela sua porta *NORTH* os pacotes do núcleo conectado a sua porta local, e os pacotes vindos dos roteadores 00 e 10. Já o segundo, tem uma carga ainda maior, pois precisa ejetar pela sua porta *NORTH*, já ao roteador 22, os pacotes vindos dos roteadores 00, 10, 20, 01, 11 e do núcleo conectado a sua porta local.

5.5.2 Cenário 2

Nesse cenário já se nota uma grande diferença entre o desempenho da rede Hermes e o da HermesFT.

Na primeira, após o *reset* ser atuado no roteador 00, no meio da transmissão do pacote 17, nenhum outro pacote consegue mais ser enviado devido à contenção geral na rede. Tal fato pode ser explicado devido à falta de um controle de timeout da chegada de pacotes de um mesmo fluxo, assunto já discutido anteriormente.

O efeito final é que o header desse pacote chegou ao roteador destino, conseguiu ser roteado à porta *LOCAL* do mesmo e a conexão não foi desfeita por conta do restante dos *flits* não terem chegado. Assim, nenhum outro fluxo conseguiu ser roteado à porta *LOCAL* da chave 22, causando assim, um verdadeiro colapso em toda rede.

Ainda houve outra falha. Por não saber do reset do roteador, o núcleo 00 continuou enviando os *flits* seguintes do pacote quebrado. O efeito é que o *flit* recebido logo após o reset foi interpretado como um header. O *flit* em questão tinha o valor 0000, e assim, o roteamento foi realizado para a porta *LOCAL*. O resultado obtido está apresentado na Figura 65 com o arquivo cujo conteúdo possui os *flits* recebidos, como pacotes, pelo núcleo 00.

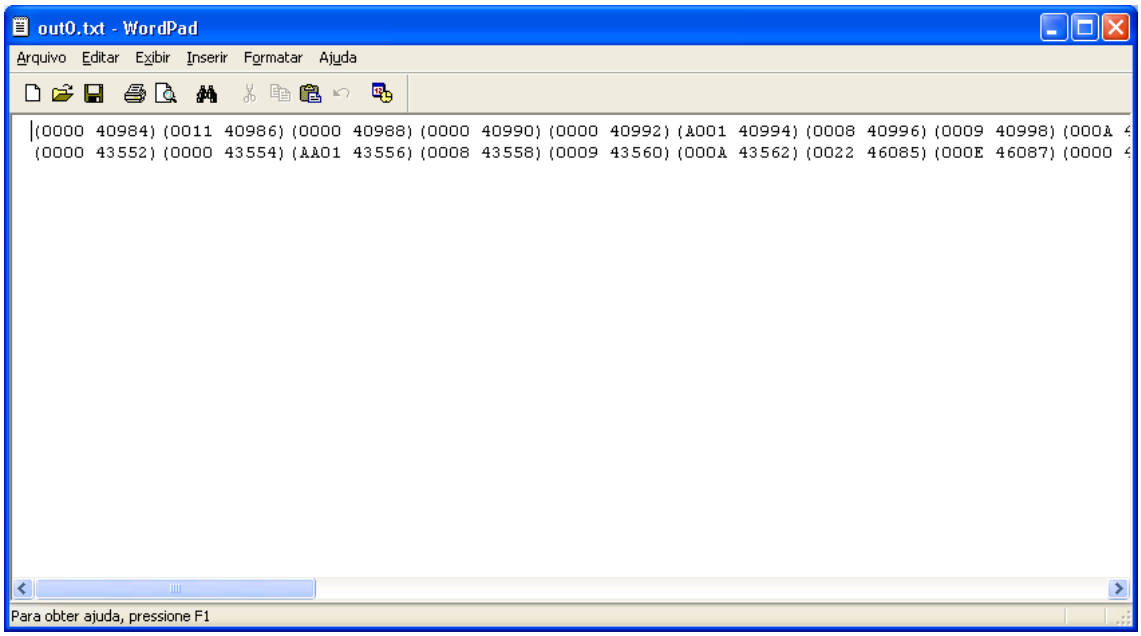

Figura 65 - Arquivo com os *flits* recebidos pelo roteador 00.

Já na NoC HermesFT, após o <u>*reset*</u> ser atuado no roteador 00, no meio da transmissão do pacote 17, não ocorrerá de o roteador confundir um *flit* de dado com um header, devido ao bit <u>bop</u>. Só na chegada do header do próximo pacote é feito o roteamento.

Já o problema de inanição enfrentado na porta *LOCAL* do roteador alvo, foi também sanado pelo controle de timeout de recebimento dos *flits* de um mesmo fluxo. Após o estouro do timeout para o recebimento do próximo *flit* (que nunca chegará) do fluxo 17 vindo do núcleo 00, a conexão entre a porta de entrada e a porta *LOCAL* do núcleo 22 é desfeita – dando assim a oportunidade dos fluxos de outros núcleos chegarem ao *IP* alvo.

Pela falha não ter afetado as regras temporais internas da chave, o mecanismo de detecção não detectou e assim, não atuou.

Na Tabela 3, é apresentado um resumo, com o número de pacotes enviados, por cada núcleo, antes e depois do instante em que a falha ocorreu.

Núcleo Remetente	Hermes		HermesFT	
	Número de Pacotes Enviados Antes da Falha	Número de Pacotes Enviados Depois da Falha	Número de Pacotes Enviados Antes da Falha	Número de Pacotes Enviados Depois da Falha
00	17	0	17	83
10	17	0	17	83
20	17	0	17	83
01	17	0	17	83
11	17	0	17	83
21	17	0	17	83
02	17	0	17	83
12	17	0	17	83

Tabela 3 - Comparativo da disponibilidade das redes antes e depois da falha no cenário 2.

Pode-se notar que a rede Hermes entra em colapso após a falha e não consegue mais enviar nenhum pacote. Já a rede HermesFT mantém o serviço de comunicação disponível, enviando todos os pacotes. Conclui-se assim que a tolerância da NoC HermesFT, a esse tipo de falha, é do tipo total, já que o sistema de comunicação, mesmo após a falha, opera sem degradação nenhuma de desempenho.

5.5.3 Cenário 3

Nesse cenário já se nota também uma grande diferença entre o desempenho da rede Hermes e o da HermesFT.

Na primeira, após o roteador 20 parar seu funcionamento nenhum outro pacote, que necessita passar pelo mesmo (como o das chaves 00 e 10), foi roteado com sucesso.

Já na NoC HermesFT, após o roteador 20 parar seu funcionamento, o circuito detector de falhas detecta o mau funcionamento e tenta resetar a chave três vezes para que a mesma volte ao seu funcionamento normal. Por se tratar de uma falha permanente, na falha seguinte os enlaces são curto-circuitados e os pacotes das chaves 00 e 10 são então roteados com sucesso até o núcleo 22.

A forma de onda apresentada na Figura 66 contém os sinais do circuito detector de falhas para demonstrar a evolução temporal da detecção e da correção. Além desses sinais, contém ainda os sinais das portas para explicar a perda de pacotes verificada nos gráficos.

Foram destacados cinco instantes representativos para o entendimento do comportamento do circuito.

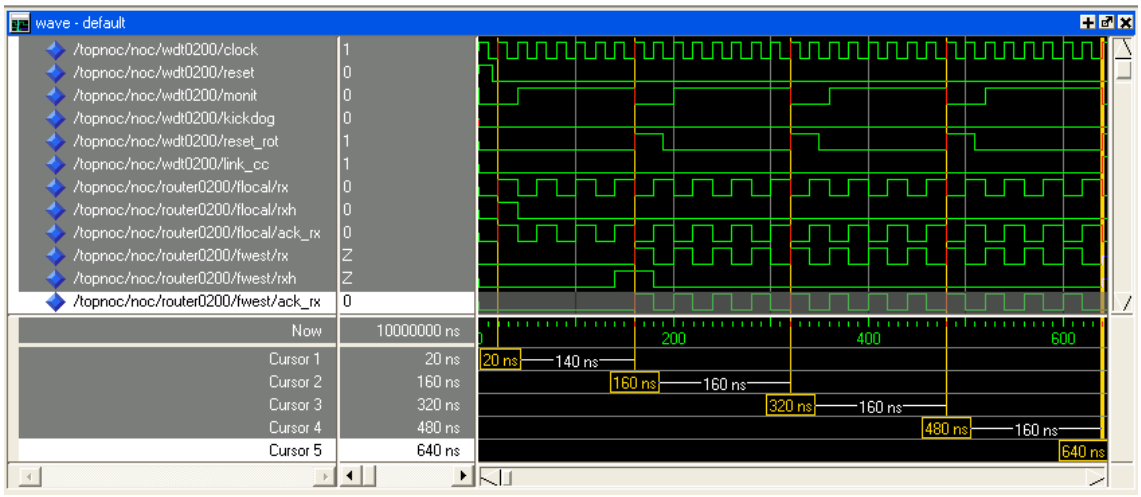

Figura 66 - Forma de onda com os sinais do circuito detector de falha.

a) Instante 1 (t = 20ns): Chega o header do primeiro pacote enviado para a porta *LOCAL* do roteador 20 pelo núcleo conectado a ele. A seguir, os outros *flits* vão sendo aceitos (como pode ser percebido pelas respostas no sinal ack_rx da porta).

b) Instante 2 (t=160ns): Chega o header do primeiro pacote enviado para a porta *WEST* do roteador 20 pelo seu vizinho, o roteador 10. A seguir, os outros *flits* vão sendo aceitos (como pode ser percebido pelas respostas no sinal ack_rx da porta). No mesmo instante o circuito detector de falhas atua o sinal de reset da chave 20. As filas então são esvaziadas, porém nem o núcleo 20 nem tampouco a chave 10 percebem que o roteador 20 foi resetado e continuam enviando os *flits* restantes do mesmo pacote.

c) Instante 3 (t=320ns): O circuito detector de falhas atua novamente o sinal de reset da chave. Novamente as filas são esvaziadas, porém nem o núcleo 20 nem tampouco a chave 10 percebem que o roteador 20 foi resetado novamente e continuam enviando os *flits* restantes do mesmo pacote.

d) Instante 4 (t=480ns): O circuito detector de falhas atua novamente o sinal de reset da chave. Novamente as filas são esvaziadas, porém nem o núcleo 20 nem tampouco a chave 10 percebem que o roteador 20 foi resetado novamente e continuam enviando os *flits* restantes do mesmo pacote.

e) Instante 5 (t=640ns): Acabadas as tentativas de reestabelecer o funcionamento normal da chave, o circuito detector de falhas então curco-circuita os enlaces do mesmo.

Dessa forma, até o instante 5, foram perdidos 2 pacotes. Adicionando-se a esses os 99 pacotes restantes (o primeiro foi perdido no instante 2) do núcleo 20 que não mais chegarão ao seu destino, tem-se o total de 101 pacotes perdidos. Em *flits*, tem-se 101*16 = 1616.

Para verificar essa quantia, basta analisar a quantidade de *flits* que saiu pela porta *NORTH* do roteador 21. Normalmente, esse valor deve ser de 9600 (1600 de cada um dos 6 núcleos que utilizam a chave para enviar seus pacotes ao roteador 22) como obtido no cenário 1. Subtraindo desse valor os *flits* perdidos, tem-se 9600 – 1616 = 7984. Exatamente o valor encontrado no gráfico.

Na Tabela 4, é apresentado um resumo, com o número de pacotes enviados, por cada núcleo, antes e depois do instante em que a falha ocorreu.

Núcleo Remetente	Hermes		HermesFT	
	Número de Pacotes Enviados Antes da Falha	Número de Pacotes Enviados Depois da Falha	Número de Pacotes Enviados Antes da Falha	Número de Pacotes Enviados Depois da Falha
00	1	0	1	99
10	1	0	1	99
20	1	0	1	0
01	1	99	1	99
11	1	99	1	99
21	1	99	1	99
02	1	99	1	99
12	1	99	1	99

Tabela 4 - Comparativo da disponibilidade das redes antes e depois da falha no cenário 3.

Pode-se notar que, apesar de apenas o núcleo 20 estar em falha permanente, os núcleos 00 e 10, na rede Hermes, não conseguem mais enviar nenhum pacote. Isto se explica pelo fato de esses núcleos dependerem do roteador 20 para que seus pacotes cheguem ao núcleo alvo. Já na rede HermesFT, esses núcleos conseguem ainda enviar seus pacotes, mesmo depois da falha, por conta do curto-circuito dos enlaces do roteador 20, explicado anteriormente. Conclui-se assim que a tolerância da NoC HermesFT, a esse tipo de falha, é do tipo degradação consentida, já que o sistema de comunicação, após a falha permanente, opera com um roteador a menos e assim, com desempenho inferior.

5.5.4 Cenário 4

Nesse cenário é, novamente, notada uma grande diferença entre as redes comparadas.

Na primeira, após a falha, toda a rede entra em colapso pela falta de um controle de timeout no recebimento dos *flits* de um mesmo pacote. A rede ficará nesse estado infinitamente e caso o núcleo 00 quisesse enviar um pacote ao núcleo 10, seu vizinho, não conseguiria por conta da contenção da porta por um pacote que jamais será entregue.

Já na NoC HermesFT, esse efeito não existe devido ao controle de timeout realizado. Quando é detectado o atraso anormal dos *flits* de um pacote, o mesmo é descartado e a porta

de saída, liberada. Assim, do ponto de vista dos núcleos, todos os pacotes foram enviados com sucesso, mas na realidade os mesmo foram descartados em troca da disponibilidade de toda a rede. Cabe assim, a aplicação confirmar o recebimento de cada pacote. A rede faz o melhor esforço, mas deve garantir a disponibilidade do serviço que provê.

Na Tabela 5, é apresentado um resumo, com o número de pacotes enviados, por cada núcleo, antes e depois do instante em que a falha ocorreu.

Núcleo Remetente	Hermes		HermesFT	
	Número de Pacotes Enviados Antes da Falha	Número de Pacotes Enviados Depois da Falha	Número de Pacotes Enviados Antes da Falha	Número de Pacotes Enviados Depois da Falha
00	1	0	1	99
10	1	0	1	99
20	1	0	1	99
01	1	0	1	99
11	1	0	1	99
21	1	0	1	99
02	1	0	1	99
12	1	0	1	99

Tabela 5 - Comparativo da disponibilidade das redes antes e depois da falha no cenário 4.

Assim, conclui-se que a tolerância da NoC HermesFT, a esse tipo de falha, é do tipo degradação consentida, já que o sistema de comunicação, após a falha permanente, opera com um roteador a menos e assim, com desempenho inferior.

6 Conclusões

Esta monografia realizou um estudo sobre a aplicação de técnicas de tolerância a falhas em redes-em-chip.

Inicialmente foi feito um estudo sobre tecnologias de comunicação em SoCs (do inglês, *System on Chip*), passando pelo antigo barramento e, mais atualmente, pelas redes-em-chip, ou NoCs (do inglês, *Network on Chip*). Foram apresentados os prós e contras de cada arquitetura, o que mostrou a importância atual do estudo e aperfeiçoamento de NoCs, motivando assim a realização desse trabalho.

No Capítulo 2 foram discutidas as principais características das redes-em-chip, já introduzindo conceitos importantes para o entendimento dos problemas que surgem nesse tipo de arquitetura de comunicação. Foram dados também alguns exemplos de redes-em-chip existentes, inclusive a que viria a ser o ponto de partida desse trabalho – a rede Hermes. Ainda nesse Capítulo foram apresentados os conceitos mais importantes para o entendimento das principais técnicas e mecanismos utilizados para prover tolerância a falhas em sistemas de um modo geral.

No Capítulo 3, a rede escolhida foi estudada com mais detalhes. Suas características foram apresentadas e suas principais fragilidades foram expostas e discutidas.

No Capítulo 4 foram apresentadas as modificações realizadas na rede-em-chip Hermes, discutindo como e em que casos cada melhoria ou técnica implementada agiria. Foi discutida também a melhoria agregada à essa arquitetura de comunicação, do ponto de vista dos conceitos de tolerância a falhas (como a confiabilidade e a disponibilidade do serviço provido).

No Capítulo 5 foi então apresentada a metodologia de testes utilizada para a obtenção dos resultados. Os resultados foram baseados na comparação entre os resultados obtidos para a rede original e a rede proposta. Ao final desse Capítulo foi feita uma análise detalhada dos resultados obtidos nos quatro cenários de testes propostos e simulados.

Os resultados obtidos foram satisfatórios, validando a hipótese inicial do trabalho de que as alterações propostas proveriam tolerância a falhas a NoC Hermes. Em todos os cenários com falha, transitória ou permanente, a NoC HermesFT conseguiu restabelecer o serviço de comunicação, apresentando assim maior robustez e disponibilidade, ao contrário da rede original. Assim, o esforço do trabalho foi justificado, tendo em vista a contribuição do mesmo para a evolução dos próximos, realizados no campo de tolerância a falhas em circuitos

de redes-em-chip. Foi mostrado assim, que através de idéias simples, como a verificação temporal de funcionamento do circuito, pode-se tornar o sistema mais disponível e confiável.

6.1 Perspectivas para Trabalhos Futuros

Nesta monografia foram realizadas melhorias e implementadas novas técnicas no projeto base da rede-em-chip Hermes. Assim, muitas das variações desse projeto foram deixadas de lado, já que a intenção era de justificar a utilização e validar a implementação das alterações. Variações como o número de canais virtuais em cada enlace, protocolos de controle de fluxo alternativos (como o baseado em créditos) e outros algoritmos de roteamento podem, e devem ser testados.

Ainda como alternativa, tem-se também a utilização de outras técnicas de detecção de falhas (do mesmo tipo, ou de outros tipos). Uma evolução interessante seria a utilização de uma técnica de codificação (como paridade) ou mesmo, apesar de mais custosa, a utilização do cálculo do CRC de cada pacote.

Outro ponto de melhoria é a criação de novos cenários de teste, de modo a prover uma validação mais rigorosa da implementação realizada. Além disso, uma análise da potência adicional requerida para a evolução sugerida também poderia ser feita por ser uma variável tão importante nos projetos atuais, principalmente em sistemas embarcados.

Finalmente, a implementação, em hardware, é também outra necessidade importante para avaliar o aumento físico do circuito, mesmo que através da simulação em uma FPGA.

Referências Bibliográficas

AVIZIENIS, LAPRIE e RANDELL, *Fundamental Concepts of Dependability*, Report 739 – University of Newcastle, 2001.

BENGTSSON, T., KUMAR, S., e PENG, Z., *Application Area Specific System Level Fault Models: A Case Study with a Simple NoC Switch*, In Proc. (electronic) International Design and Test Workshop (IDT), 2006.

BARTIC, T.; MIGNOLET, J.-Y.; NOLLET, V.; MARESCAUX, T.; VERKEST, D.; VERNALDE, S.; LAUWEREINS, R., *Topology adaptive network-on-chip design and implementation*. In: IEE Proceedings on Computer and Digital Tecniques, vol. 152-4, Jul. 2005, pp. 467-472.

BASTOS, E., *MERCURY: Uma rede Intra-chip com Topologia Toro 2D e roteamento adaptativo*. Dissertação de Mestrado, 2005, Porto Alegre, PPGCC/PUCRS, p. 148.

BEZERRA, J. C., *Verificação e prototipação de redes intrachip : o estudo de caso Hermes-TB*. Dissertação de Mestrado, 2009, Porto Alegre, PPGCC/PUCRS.

CARARA, E., *Uma Exploração Arquitetural de Redes Intrachip com Topologia Malha e Modo de Chaveamento Wormhole*. Dissertação de Mestrado, PPGCC-FACIN-PUCRS, Porto Alegre, Dec. 2007, p. 65.

CARDOZO, R. S, *Redes-em-Chip de Baixo Custo*. Dissertação de Mestrado, PPGCC-UFRGS, Porto Alegre, Set. 2005, p. 17.

DUATO, J. et al., *Interconnection Networks.* Los Alamitos, California: IEEE Computer Society Press, 1997, p. 515.

FRANTZ, A. P., CARRO, L., COTA, E. KASTENSMIDT, F. L., *Evaluating SEU and Crosstalk effects in Network-on-Chip Routers*. In: IEEE International On-Line Testing Symposium – IOLTS, 12., Lake of Como, 2006. Proccedings… Lake of Como: IOLTS, 2006. p. 191-192.

GORDON, M.; THIES, W.; AMARASINGHE, S. "Exploiting Coarse-Grained Task, Data, and Pipeline Parallelism in Stream Programs". In: International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). San Jose, EUA. Outubro, 2006. pp.151-162.

GUERRIER, P.; GREINER, A., *A generic architecture for on-chip packet switched interconnections*. In: DESIGN AUTOMATION AND TEST IN EUROPEAN CONFERENCE, 2000, Paris. Proceedings... Los Alamitos: IEEE Computer Society Press, 2000.

HEMANI A. et al., *Lowering Power consumption in clock by using globally asynchronous locally synchronous desing style*. In DESING AUTOMATION CONFERENCE, 36., New Orleans, 1999.Proceedings... Los Alamitos: IEEE Computer Society Press, 1999.

JALOTE, P. Fault *tolerance in distributed systems*. Prentice Hall, Englewood Cliffs, New Jersey, 1994.

KOIBUCH, M. et al., *A Lightweight Fault-Tolerant Mechanism for Network-on-Chip*. In: IEEE Internacional Symposium On Network-On-Chip – NOCS, 2008, Newcastle. Proccedings… Washington: IEEE Computer Society. 2008.

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

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

LUCAS, A. H. e MORAES, F. G., *Crosstalk fault tolerant NoC – design and evaluation*. In: IEEE Internacional Conference On Very Large Scale Integration – IFIP, 2009, Florianopolis. Proccedings... Florianópolis: VLSI-SoC. 2009.

MORAES, F. G. et al., *A Low Area Overhead Packet-switched Network on Chip: Architecture and Prototyping*. In: INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION, 2003, Darmstadt. Proceedings... [S.1.: s.n.], 2003.

MORAES, F.; CALAZANS, N.; MELLO, A.; MÖLLER, L.; OST, L., *Hermes: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip*. Integration the VLSI Journal, vol. 38-1, Oct. 2004, pp. 69-93.

NIKOLIC, K., Sadek, A., and Forshaw, M., *Architectures for Reliable Computing with Unreliable Nanodevices*, In Proc. IEEE NANO, 2001, 254.259.

NI, M.; MCKINLEY, P., *A Survey of Wormhole Routing Techniques in Direct Networks*. IEEE Computer, vol. 26-2, 1993, pp. 62-76.

PRADHAN, D. K., *Fault-Tolerant System Design*. Prentice Hall, New Jersey, 1996.

PULLINI, A. et al., *Fault tolerant overhead in Network-on-Chip flow control schemes*. In: Simposium On Integrated Circuits and Systems – SBCCI, 18., 2005, Florianópolis. Proceedings... New York: ACM Press, 2005. p. 224-229.

SILBERSCHATZ, A. e GALVIN, P., *Operating Systems Concepts*. New York, Addison Wesley, 5th ed., 1997, 888 p.

WEBER, T. S., *Tolerância a falhas: conceitos e exemplos*. 2001.

ZEFERINO, C. A., *Redes-em-chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho*. Tese de doutorado, Universidade Federal do Rio Grande do Sul - UFRGS/PPGC, Porto Alegre, RS, 2003a.

ZEFERINO, C. A.; SUSIN, A. A., *A SoCIN: a parametric and scalable network-on-chip*. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS, 16., 2003b, São Paulo. Proceedings... Los Alamitos:IEEE Computer Society , 2003b. p. 169-174.