UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

MESTRADO ACADÊMICO EM ENGENHARIA DE TELEINFORMÁTICA

RONALDO TADEU PONTES MILFONT

REDUÇÃO DE LATÊNCIA EM REDES INTRACHIP TOLERANTES A FALHA ATRAVÉS DO USO DE MÚLTIPLOS CAMINHOS

FORTALEZA

2017

RONALDO TADEU PONTES MILFONT

REDUÇÃO DE LATÊNCIA EM REDES INTRACHIP TOLERANTES A FALHA ATRAVÉS DO USO DE MÚLTIPLOS CAMINHOS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia de Teleinformática do Programa de Pós-Graduação em Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia de Teleinformática. Área de Concentração: Sinais e Sistemas

Orientador: Prof. Dr. Paulo César Cortez

Co-Orientador: Prof. Dr. Jarbas Aryel Nunes da Silveira

FORTALEZA

2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M581r Milfont, Ronaldo Tadeu Pontes.
Redução de latência em redes intrachip tolerantes a falha através do uso de múltiplos caminhos / Ronaldo Tadeu Pontes Milfont. – 2017.
69 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2017.
Orientação: Prof. Dr. Paulo César Cortez.
Coorientação: Prof. Dr. Jarbas Aryel Nunes da Silveira.

1. Redes intrachip. 2. Tolerância a falhas. 3. Confiabilidade. I. Título.

CDD 621.38

RONALDO TADEU PONTES MILFONT

REDUÇÃO DE LATÊNCIA EM REDES INTRACHIP TOLERANTES A FALHA ATRAVÉS DO USO DE MÚLTIPLOS CAMINHOS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia de Teleinformática do Programa de Pós-Graduação em Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia de Teleinformática. Área de Concentração: Sinais e Sistemas

Aprovada em: 02 de Setembro de 2017

BANCA EXAMINADORA

Prof. Dr. Paulo César Cortez (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Jarbas Aryel Nunes da Silveira (Co-Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Fabian Luis Vargas
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)

Prof. Dr. Giovanni Cordeiro Barroso
Universidade Federal do Ceará (UFC)

Prof. Dr. Danielo Gonçalves Gomes
Universidade Federal do Ceará (UFC)

A Deus. Aos meus avós, Cirineu *(in memoriam)* e Maria de Jesus. A minha mãe, Ana Rosa. Ao meu irmão, Rodrigo. A minha esposa, Isabelle. A minha filha, Lia.

“Se algum de vocês tem falta de sabedoria, peça-a a Deus, que a todos dá livremente, de boa vontade; e lhe será concedida. Peça-a, porém, com fé, sem duvidar, pois aquele que duvida é semelhante à onda do mar, levada e agitada pelo vento.”

(Tiago 1:5,6)

RESUMO

As tecnologias de circuitos integrados estão atingindo escalas nanométricas e com isto aumentando a probabilidade de falhas permanentes, transientes e intermitentes. Como resultado, a demanda por estratégias de tolerância a falhas é o tema principal de diversas pesquisas visando projetos de Sistemas Intrachip. Em particular, os mecanismos de retransmissão consistem de uma das soluções mais utilizadas nas Redes Intrachip bidimensionais. Todavia estes mecanismos introduzem atrasos na latência dos pacotes. Este trabalho propõe o uso de múltiplos caminhos, mínimos e não mínimos, como forma de reduzir o atraso adicionado causado pelo impacto das retransmissões em sistemas críticos, isto é, onde a latência é um problema crítico. A técnica contempla utilizar diferentes conjuntos de caminhos para criar as tabelas de roteamento. Duas métricas são propostas para classificar os caminhos existentes para um par de comunicação considerando probabilidade de falha das conexões de comunicação e a quantidade de novas conexões adicionadas ao fazer uso de um novo caminho. Os resultados experimentais mostram que o uso de caminhos múltiplos, mínimos ou não, permite diminuir o impacto causado por retransmissões em 25% e 20% da latência média de pacotes para tecnologias CMOS de 22 *nm* e 65 *nm*, respectivamente. Além disso, a técnica proposta pode contribuir para uma maior adaptabilidade a falhas nas ligações e pode ser mais bem investigada em trabalhos futuros em circunstâncias de tráfego intenso e para topologias de Redes Intrachip tridimensionais.

Palavras-chave: Redes intrachip. Confiabilidade. Tolerância a falhas

ABSTRACT

Integrated circuit technologies are reaching nanometer scales and thereby increasing the likelihood of permanent, transient, and intermittent failures. As a result, the demand for fault tolerance strategies is the main subject of many types of research targeting Systems-on-Chip designs. In particular, retransmission mechanisms are one of the most used solutions in bidimensional Networks-on-Chip. However, these mechanisms introduce in the packet latency. This work proposes the use of multiple paths (minimal or not) as a way to reduce the extra delay caused by the impact of retransmissions in critical systems (i.e, where latency is a critical problem). The technique encompasses using different sets of paths to create the routing tables. Two metrics are proposed to classify the paths for a communication pair considering the probability of failure of the communication links and the number of new communication links added when making use of a new path. The experimental results show that the use of multiple paths can reduce the impact caused by retransmissions in 25% and 20% of the average packet latency for 22 *nm* and 65 *nm* CMOS technologies, respectively. Moreover, the proposed technique can contribute to greater adaptability to faults on links and could be better investigated in future work under circumstances of heavy traffic and for tridimensional Networks-on-Chip topologies.

Keywords: Network-on-Chip. Adaptability. Fault-tolerance

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BIST	*Built-In Self-test*
CMOS	*Complementary Metal Oxide Semiconductor*
CPU	*Central Processing Unit*
DVFS	*Dynamic Voltage and Frequency Scaling*
ECCs	*Error Correction Codes*
FinFETs	*Fin Field-Effect Transistors*
FTDR	*Fault-Tolerant Deflection Routing*
GPU	*Graphics Processing Unit*
ITRS	*International Roadmap for Semiconductors*
NBTI	*Negative Bias Temperature Instability*
NI	*Network Interface*
NoCs	*Networks-on-Chip*
OSR	*Overlapped Static Reconfiguration*
PE	*Processing Element*
RBR	*Region-based Routing*
RTL	*Register-Transfer Level*
SBR	*Segment-based Routing*
SEC	*Single Error Correcting*
SoCs	*Systems-on-Chip*
VHDL	*Very High Speed Integrated Circuits (VHSIC) Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuits*

SUMÁRIO

1 INTRODUÇÃO

O avanço tecnológico nas últimas décadas propiciou a construção de chips com bilhões de transistores. Esses chips, com alta integração, são comumente referenciados como *Systems-on-Chip* (SoCs) e tipicamente são constituídos de vários componentes complexos como processadores programáveis de propósito geral, controladores, blocos de hardware dedicados, memórias, interfaces de entrada/saída e uma arquitetura de comunicação que é responsável por interconectar esses componentes (PASRICHA; DUTT, 2010).

Para um bom funcionamento do SoC, a estrutura de comunicação precisa ser robusta, escalável e estruturada. Soluções convencionais baseadas em interconexão de blocos lógicos em um único chip através de barramentos ou ponto-a-ponto, não são eficientes para os SoCs devido à grande quantidade de componentes que podem estar presentes (MARCULESCU *et al.*, 2009).

A infraestrutura do tipo barramento pode ser classificada como sendo a mais tradicional e simples, em que o meio de transmissão dos pacotes é único e compartilhado por todas as unidades. Nessa, apenas um *core* poderá enviar pacotes por vez, que será recebido por todos os outros e ignorados por aqueles aos quais o pacote não se destina. Essa estrutura limita o paralelismo, no entanto dá suporte intrínseco a *broadcast*, o que pode ser bastante desejável em algumas ocasiões. Quanto à potência, ela dever ser suficiente para manter o nível lógico de uma ponta a outra do barramento e portanto também não é considerada escalável (DALLY; TOWLES, 2004). Na arquitetura ponto-a-ponto existem ligações dedicadas que interligam diretamente os dispositivos entre si, o que melhora bastante o fluxo de comunicação. No entanto, a escalabilidade de tal arquitetura é bastante limitada: um SoC com N dispositivos possui no pior caso $N(N-1)$ conexões e a adição de um único *core* aumenta de N o número de conexões (DALLY; TOWLES, 2004).

Neste contexto, surgem as redes intrachip, em inglês, *Networks-on-Chip* (NoCs), que têm sido largamente empregada como solução de comunicação para SoCs devido ao seu alto grau de paralelismo (uma vez que todos as conexões da rede podem ser utilizadas de forma simultânea, trafegando diferentes pacotes), escalabilidade (adição de um novo elemento consiste apenas da adição de conexões aos roteadores vizinhos) e reusabilidade (aplicações de propósito geral) (CHARIF *et al.*, 2016).

No entanto, a confiabilidade dos circuitos integrados está reduzindo com o surgimento das novas tecnologias de transistores *Complementary Metal Oxide Semiconductor* (CMOS) e *Fin Field-Effect Transistors* (FinFETs), uma vez que estas atingiram escalas nanomé-

tricas, aumentando, assim, a possibilidade de ocorrer diferentes tipos de falhas (HOEFFLINGER, 2011). Falhas em conexões de uma NoC podem comprometer o funcionamento de todo o sistema. Além disso, estudos mostram que, no futuro, 10% dos SoCs apresentará falhas (VITKOVSKIY *et al.*, 2013). Desta forma, a demanda por mecanismos de tolerância a falhas nos últimos anos vem sendo alvo de diversas pesquisas como forma de aumentar a resiliência das arquiteturas atuais.

As falhas em circuitos integrados podem ser classificadas em três tipos: permanente, transiente e intermitente (CONSTANTINESCU, 2003). Falhas do tipo permanente podem ocorrer devido a defeitos de fabricação, danos por desgaste, tais como o efeito *Negative Bias Temperature Instability* (NBTI) e ruptura dielétrica, dentre outros (PASRICHA; ZOU, 2011). Por outro lado, falhas transientes estão relacionadas a interferências eletromagnéticas, indutivas ou capacitivas e partículas radioativas. Este tipo de falha ocorre em intervalos de tempo bastante curtos e são extremamente difíceis de prever (BAHREBAR; STROOBANDT, 2014). Por fim, falhas do tipo intermitente são aquelas que ocorrem em uma frequência irregular e normalmente estão associadas a variações das condições de operação, tais como variação dos parâmetros do processo de fabricação, flutuações de tensão e mudanças na temperatura (YU; AMPADU, 2011).

Para contornar falhas do tipo permanente, existem diferentes técnicas de roteamento tolerantes a falha e estas são foco de estudos da literatura, como pode ser visto em (PARK *et al.*, 2006; YU *et al.*, 2011; FENG *et al.*, 2013; EBRAHIMI *et al.*, 2013). No tocante às falhas transientes e intermitentes, as estratégias de tolerância a falhas mais empregadas consistem em detectar e corrigir erros (BERTOZZI *et al.*, 2002; MURALI *et al.*, 2005), e em retransmitir a informação (BERTOZZI *et al.*, 2005; CHANG *et al.*, 2011). Todavia, essas estratégias geralmente degradam o desempenho da NoC, além de aumentar a área e o consumo de energia do sistema. Diversos trabalhos exploram o impacto destes mecanismos sob diferentes ópticas: eficiência energética, resiliência ao erro e desempenho da rede (MURALI *et al.*, 2005; BERTOZZI *et al.*, 2005; LEHTONEN *et al.*, 2007). Em particular, mecanismos de retransmissão são bastante eficientes, mas introduzem atraso na latência dos pacotes, uma vez que é necessário identificar erros e retransmitir dados. Especialmente para sistemas críticos, em que há um alto volume de comunicação e são desejáveis pequenos ou nenhum tipo de atraso de comunicação para garantir o funcionamento adequado. Assim, o aumento da quantidade de retransmissões pode comprometer todo o sistema (BERTOZZI *et al.*, 2002).

A topologia da rede também pode influenciar no mecanismo de tolerância a falhas,

uma vez que, perante a falha de pelo menos um componente, a regularidade da estrutura de comunicação pode ser parcialmente/completamente comprometida (MEJIA *et al.*, 2009). Logo, é desejável que a estratégia de tolerância a falhas faça uso da redundância intrínseca de caminhos das NoCs permitindo, assim, uma melhor adaptabilidade diante de falhas (PASRICHA; ZOU, 2011). Desta forma, os mecanismos de tolerância a falhas podem assegurar um determinado requisito de projeto, como, por exemplo, minimizar a latência dos pacotes para um par de comunicação crítico. Além disso, o mecanismo pode predefinir vários caminhos que são utilizados de forma dinâmica de acordo com o cenário da NoC, que pode mudar, em virtude da dinamicidade da rede devido ao tráfego ou a falhas em conexões.

Neste trabalho é proposta uma abordagem inédita de tolerância a falhas transientes e intermitentes nas conexões de comunicação entre roteadores para NoCs tolerantes a falha. Para isto, é proposta a utilização de múltiplos caminhos mínimos e não mínimos como forma de reduzir o impacto causado por mecanismos de tolerância a falhas baseados em retransmissão para sistemas críticos que são baseados em retransmissão.

A estratégia de tolerância a falhas baseada em retransmissão de pacotes foi escolhida como mecanismo-alvo devido ao seu melhor custo-benefício diante de outras alternativas (BERTOZZI *et al.*, 2002; PARK *et al.*, 2006). Neste trabalho é proposto o uso da redundância intrínseca de caminhos entre os pares de comunicação de uma NoC para ampliar a possibilidade de escolha de caminhos para envio de pacotes críticos. No entanto, é necessário caracterizar todos os caminhos para um par de comunicação crítico levando em consideração a dinamicidade de falhas de cada conexão envolvida no caminho de comunicação e não apenas seu o número de conexões e roteadores existentes no caminho (i.e, o número de saltos). Consideram-se ainda como importantes pontos de investigação: (i) a inserção de um novo caminho contribui para uma melhor distribuição do tráfego; (ii) a utilização de mais de um caminho para um mesmo par de comunicação pode diminuir os riscos de saturação da comunicação, reduzindo as chances de retransmissão. Neste âmbito, neste trabalho são propostas duas novas métricas: a primeira, capaz de estimar a latência de um caminho baseada na dinamicidade de falhas de todas as conexões envolvidas e a segunda capaz de identificar como um caminho contribui na distribuição do tráfego para um par de comunicação.

1.1 Objetivos

O objetivo geral neste trabalho é explorar uma nova abordagem de utilização de caminhos mínimos e não mínimos como forma de reduzir o número de retransmissões causado por mecanismos de retransmissão em NoCs tolerantes a falha e, por sua vez, reduzir a latência entre pares de comunicação críticos.

De forma geral, a estratégia mais usual para tolerância a falhas consiste em redundância de dados temporal e/ou espacial. Em particular, as soluções de redundância temporal com base em retransmissão de dados necessitam de um tempo maior para execução, impondo maior atraso para transmissão de pacotes; sendo que o aumento da latência pode comprometer pares de comunicação críticos, dados os requisitos necessários de desempenho. Neste contexto, este trabalho explora a utilização de caminhos mínimos e não mínimos como forma de reduzir o número de retransmissões e por sua vez reduzir a latência entre pares de comunicação críticos.

Dentre os objetivos específicos enumera-se:

1. Análise do desempenho de mecanismos de tolerância a falhas descritos na literatura;
2. Buscar uma heurística que estime a latência de um caminho dado o número de saltos e o percentual de falhas;
3. Buscar uma métrica capaz de identificar como um caminho de comunicação contribui com novos canais para diminuir a concentração de tráfego para um determinado par de comunicação.
4. Implementar ferramentas para simulação da NoC: geração de cenários de falhas, geração de tráfego sintético, geração de tabelas de roteamento e extração de resultados das simulações;

1.2 Estrutura do trabalho

O restante desta dissertação está organizado da seguinte forma: no Capítulo 2 é apresentado uma revisão literária de arquiteturas e mecanismos tolerantes a falhas, ressaltando suas principais contribuições. No Capítulo 3 são apresentados conceitos e generalidades importantes acerca das redes intrachip. Este capítulo pode ser deixado como leitura opcional, caso o leitor já tenha um conhecimento básico do tema. A abordagem por múltiplos caminhos e a metodologia empregada para validação são descritas no Capítulo 4. Adicionalmente, este capítulo detalha o

modelo de falhas utilizado para geração dos cenários de falhas, descrição e obtenção das métricas propostas para caracterizar os múltiplos caminhos entre um determinado par de comunicação, bem como as ferramentas desenvolvidas para atender a demanda experimental. No Capítulo 5 são apresentados os resultados experimentais e a análise do impacto da comunicação da técnica proposta. Finalmente, no Capítulo 6 são elencadas as conclusões deste trabalho, apresentando as principais contribuições e possíveis trabalhos futuros.

1.3 Publicações

Este trabalho gerou duas publicações em conferências internacionais:

- **Analysis of Routing Algorithms Generation for Irregular NoC Topologies**, apresentado no "18th IEEE Latin American Test Symposium (LATS), 2017 in Bogotá - Colômbia, 2017" - Autores: Ronaldo Milfont, Paulo Cortez, Alan Cadore, João Ferreira, Jarbas Silveira, Rafael Mota e César Marcon.
- **Latency Reduction of Fault-Tolerant NoCs by Employing Multiple Paths**, apresentado no "30th Symposium on Integrated Circuits and Systems Design (SBCCI), in Fortaleza - Brazil, 2017" - Autores: Ronaldo Milfont, João Ferreira, Daniel Tavares, Rafael Mota, Paulo Cortez, Jarbas Silveira e César Marcon.

2 TRABALHOS RELACIONADOS

Neste capítulo é apresentado um sumário da revisão literária de algumas arquiteturas propostas para tolerância a falhas em redes intrachip, fundamentando assim a pesquisa desta dissertação. Para melhor organização e entendimento do leitor, este capítulo está dividido nos seguintes temas: (i) algoritmos de roteamento tolerantes a falha; (ii) códigos detectores e corretores de erro aplicáveis em NoCs; (iii) mecanismos de retransmissão; (iv) reconfiguração da rede; e (v) múltiplos caminhos.

Segundo o *International Roadmap for Semiconductors* (ITRS), até 2020 teremos processos de fabricação de 5 *nm* (HOEFFLINGER, 2011). Ainda sob a perspectiva do ITRS, milhares de componentes estarão integrados em um único SoC nos próximos anos. Neste contexto, as redes intrachip têm desempenhado um papel importante para sobrepor as limitações advindas do uso de arquiteturas convencionais de comunicação (e.g, barramentos) que não são escaláveis, considerando a grande quantidade de blocos lógicos dos circuitos modernos e futuros.

No entanto, a diminuição do tamanho, o aumento da complexidade e alta densidade de integração dos transistores proporciona maior vulnerabilidade nas redes intrachip. A influência da variabilidade dos processos de fabricação, condições de operação e envelhecimento acelerado dos componentes podem ocasionar diversos tipos de falhas em NoCs, que, por sua vez, precisam ser robustas para atender demandas de diversas classes de aplicações.

Diante do exposto acima, as redes intrachip tolerantes a falhas têm sido alvo de diversas pesquisas para lidar com falhas em diferentes níveis de implementação. Adicionalmente, é desejável que os sistemas tolerantes a falhas tenham alta confiabilidade sem introduzir uma diminuição excessiva no desempenho do sistema (BORKAR, 2005). Diversas estratégias de software e hardware têm sido propostas nos últimos anos, com destaque para os algoritmos de roteamento tolerantes a falha, códigos detectores e corretores de erros, e mecanismos de retransmissão e reconfiguração. Alguns trabalhos utilizam essas estratégias para permitir robustez ainda maior perante situações de falhas.

Algoritmos de roteamento tolerantes a falhas devem permitir rotas livres de *deadlocks*, sendo estes algoritmos classificados como estáticos ou dinâmicos.

Na abordagem estática todas as possíveis falhas a serem toleradas devem ser conhecidas em tempo de projeto. Assim, durante a operação do sistema, quando uma é falha descoberta é detectada, o sistema é paralisado, os pacotes são descartados e um novo roteamento

(pré-computado) é empregado. Na abordagem dinâmica, não é necessário interromper a rede; a computação do roteamento é realizada em tempo de execução. Além disso, para evitar que alguns elementos de roteamento (e.g., tabelas de roteamento) estejam em um estado inconsistente durante a reconfiguração da rede, é necessária a implementação de um mecanismo de roteamento distribuído de alta complexidade. As estratégias que usam modelos de falha estáticos como vistas em (GOMEZ *et al.*, 2004; HO; STOCKMEYER, 2004) são mais fáceis de implementar e consomem menor área, no entanto, a quantidade de falhas suportada é limitada (STRANO *et al.*, 2012). Por outro lado, trabalhos que implementam estratégias com base em modelos de falha dinâmica, como em (FICK *et al.*, 2009; STRANO *et al.*, 2012; TRIVIÑO *et al.*, 2012), permitem maior adaptabilidade perante mudanças na topologia da rede, contudo, aumentam a complexidade do sistema. Neste trabalho foca-se nesta última abordagem (i.e.,estratégias tolerantes a falhas dinâmicas).

Alguns trabalhos combinam os algoritmos de roteamento com mecanismos de hardware para permitir tolerância a mais de um tipo de falha. Em (ZHANG *et al.*, 2010) é apresentado um mecanismo para tolerar falhas transientes a partir do uso de detectores de erros a nível de *flit* (ver Seção **??**). Por outro lado, para falhas do tipo permanentes, é sugerido o uso de um algoritmo de roteamento dinâmico capaz de recalcular caminhos em situações de falhas das conexões. (KIA; ABABEI, 2011) propõem um algoritmo capaz de não só tolerar falhas, mas também de evitar congestionamentos na rede. Para isso, os autores dividem a rede em regiões em que cada uma possui um monitor capaz de identificar conexões falhas e as condições de tráfego atual da rede; os monitores são então interconectados entre si para que sejam capazes de ter conhecimento de toda a rede. Desta forma, cada monitor é responsável por computar o caminho mais rápido de acordo com as condições atuais da rede. No entanto, este algoritmo não é livre de *deadlock* na rede, além de requerer uma maior área devido aos monitores. Os resultados experimentais foram comparados aos algoritmos XY e DyXY (LI *et al.*, 2006), e apresentaram uma eficiência superior a ambos.

Outra alternativa consiste na redundância de informação nos dados transferidos na rede, através da utilização de códigos corretores de erros(em inglês, *Error Correction Codes* (ECCs)). Um dos códigos mais conhecidos é o do tipo *Single Error Correcting* (SEC), como por exemplo, o código de Hamming (HAMMING, 1950). Além disso, múltiplos SECs podem ser utilizados para corrigir erros em canais adjacentes, combinando técnicas de intercalação, como apresentado em (YU; AMPADU, 2008; YU; AMPADU, 2011). Todavia, de forma geral, os

ECCs mais simples são limitados na quantidade de falhas que conseguem detectar e/ou corrigir, já os mais robustos possuem implementação complexa, com sobrecarga significativa no consumo de energia e área.

Neste âmbito, alguns trabalhos (GANGULY *et al.*, 2009; ZAMZAM *et al.*, 2011; YU *et al.*, 2011; WANG *et al.*, 2016) buscam otimizar a eficiência energética através da combinação de diferentes tipos de ECCs. No entanto, utilizar diferentes ECCs resulta em diferentes complexidades dos circuitos necessários para implementação, que, por sua vez, adicionam cargas capacitivas e atividade extra de chaveamento, aumentando a dissipação de potência.

Uma terceira abordagem para tolerar falhas consiste no uso de mecanismos para detecção de erros e retransmissão de dados. Desta forma, quando um erro é detectado, um sinal de retransmissão é enviado à fonte solicitando retransmissão. (KIM *et al.*, 2005) propõem cinco diferentes mecanismos de retransmissão para lidar com erros nas conexões das NoCs. Para cada mecanismo, são avaliados os custos de área de *buffers* e impacto nos atrasos dos pacotes. (PARK *et al.*, 2006) apresentam um novo mecanismo de retransmissão para lidar com falhas transientes nas conexões de comunicação a partir da modificação do tamanho do *buffer* de retransmissão. Resultados experimentais mostraram que o uso do mecanismo de retransmissão proposto gera sobrecarga menor quando comparado com outras técnicas utilizadas.

Em (BERTOZZI *et al.*, 2002), os autores avaliam o impacto de usar mecanismos de detecção e correção de erros comparado com o uso de mecanismos que ao detectarem erros requerem a retransmissão de dados, tendo como cenários arquiteturas do tipo NoC com falhas transientes nas conexões de comunicação. Os resultados experimentais obtidos mostraram que os códigos detectores de erro em conjunto com a retransmissão de dados é mais eficiente energeticamente, uma vez que o uso de componentes adicionais para correção do erro acarreta em maior custo de área e potência.

Alguns autores como (LEHTONEN *et al.*, 2007; YU *et al.*, 2011) utilizam combinações de códigos para detecção e correção de erros, mecanismos de retransmissão e conexões de reserva para prover proteção completa contra falhas permanentes, transientes e intermitentes. Contudo, estas soluções implicam em implementações de software e hardware bastante complexas e de alto custo.

Outra estratégia para tolerância a falhas bastante utilizada consiste na reconfiguração da rede, podendo ser implementada em software, hardware, ou em ambos, dependendo dos requisitos das aplicações e do custo envolvido. A reconfiguração da rede também pode ser

classificada em estática ou dinâmica. Sendo estático o mecanismo de reconfiguração, a ocorrência de uma falha implica a interrupção da rede, descarte de pacotes e recarga das novas tabelas de roteamento, para então, reinicializar completamente a rede de comunicação. Em contrapartida, reconfigurações do tipo dinâmica não requerem paralisação da rede e nem descarte de pacotes, sendo as tabelas atualizadas em tempo de execução.

(STRANO *et al.*, 2012) propõem o *Overlapped Static Reconfiguration* (OSR), uma técnica antiga usada em redes externas ao chip, mas agora adaptadas às restrições de NoCs. Esta abordagem permite que novos pacotes sejam injetados na rede enquanto pacotes antigos são roteados ou descartados. Os autores usam marcas sinalizadoras para separar pacotes antigos dos novos; e durante a propagação da marca sinalizadora a NoC permanece com ambas as configurações, a anterior e a nova. Isto possibilita uma rápida reconfiguração para que a NoC funcione apropriadamente, mas implica uma implementação de hardware complexa.

(FICK *et al.*, 2009) descrevem a arquitetura da Vicis, uma NoC tolerante a falhas que preserva a funcionalidade do sistema com base na redundância inerente encontrada em muitas redes. A NoC Vicis emprega reconfiguração de roteamento no nível de roteador e de rede. Cada roteador tem um circuito de *Built-In Self-test* (BIST) para diagnosticar falhas e configurar o hardware. A reconfiguração em dois níveis permite confinar algumas falhas e a ação correspondente no roteador, simplificando o processo de reconfiguração e permitindo que o desempenho da rede degrade suavemente enquanto a quantidade de falhas da rede aumenta.

(FENG *et al.*, 2013) propõem uma solução tolerante a falhas para redes intrachip sem buffers, em que o diagnóstico de falhas permanentes e transientes é realizado em tempo real, sendo capaz de corrigir erros simples e detectar erros duplos. Sua solução provê a reconfiguração das tabelas de roteamento durante a transmissão de pacotes por algoritmo do tipo *Fault-Tolerant Deflection Routing* (FTDR) que tolera falhas permanentes livres de *deadlock* e *livelock*, sem perdas de pacotes durante o processo de reconfiguração.

(SILVEIRA *et al.*, 2015) implementam um mecanismo inteligente de decisão capaz de definir limiares para determinar quando é necessário reconfigurar toda a rede ou utilizar mecanismos de retransmissão. Os resultados experimentais evidenciaram que muitas vezes é possível tolerar conexões parcialmente falhas devido ao custo de reconfiguração de toda a rede, pois, muitas vezes, é necessário reiniciar toda a NoC comprometendo os pacotes já em tráfego, o que pode inviabilizar seu uso em aplicações de tempo real.

De forma geral, nenhum destes trabalhos descreve as relações de custo-benefício

envolvidas. Estratégias baseadas em redundância de hardware implicam em uma maior área e consumo de energia. Por outro lado, abordagens de software podem consumir um tempo de computação excessivo, gerando atrasos que comprometem sistemas críticos ou que demandam alta performance.

A diversidade de caminhos inerente à topologia das redes intrachip, isto é, os diferentes caminhos mínimos e não mínimos entre um determinado par de comunicação podem permitir maior grau de adaptabilidade da rede perante falhas (RADETZKI *et al.*, 2013). A partir de uma maior diversidade de caminhos as falhas podem ser contornadas a partir da seleção de caminhos livres ou com menores quantidades de falhas. Todavia, trabalhos da literatura atual utilizam essa característica fora do contexto de tolerância a falhas.

Em (YIN *et al.*, 2012), o roteamento dos pacotes para uma NoC heterogênea é otimizado a partir da utilização de múltiplos caminhos. A NoC utilizada é apresentada como solução de comunicação para uma arquitetura envolvendo uma *Central Processing Unit* (CPU) e uma *Graphics Processing Unit* (GPU). Os autores, motivados pelo fato de que o tráfego de dados em processadores gráficos pode tolerar certo atraso, utilizam caminhos não mínimos como forma de melhorar a eficiência energética da rede, além de melhor distribuir o tráfego, evitando congestionamentos. Adicionalmente, utilizam a técnica de escalação dinâmica de tensão e frequência (em inglês, *Dynamic Voltage and Frequency Scaling* (DVFS)) para reduzir a dissipação de energia da rede. A principal ideia por trás do DVFS dimensionar dinamicamente a tensão de alimentação dos roteadores e conexões para fornecer velocidade de circuito suficiente para processar a carga de tráfego.(KUMAR *et al.*, 2014) propõem um novo algoritmo de roteamento não mínimo para evitar o congestionamento da rede, com base em restrição de caminhos. Os resultados mostraram que o uso de múltiplos caminhos aumentou o desempenho da rede pela distribuição do tráfego para áreas não congestionadas.

Caso o algoritmo de roteamento tome em consideração a diversidade de caminhos das redes intrachip, este pode selecionar caminhos não mínimos com menor probabilidade de falhar, evitando retransmissões. Adicionalmente, a inserção de conexões de comunicação que ainda não foram utilizadas por um par de comunicação aumenta a capacidade de tráfego aceito para um determinado par e, consequentemente, atrasa seu ponto de saturação. Desta forma, os atrasos acrescentados pelos mecanismos de tolerância a falha seriam reduzidos, contribuindo para um melhor desempenho da rede, especialmente para sistemas críticos.

Neste contexto, o trabalho aqui proposto tem como principal contribuição a utili-

zação de múltiplos caminhos como forma de reduzir o atraso acrescentado por mecanismos de retransmissão em sistemas críticos tolerantes a falha, que para nosso melhor conhecimento é inédito. No entanto, é necessário classificar os diversos caminhos, uma vez que caminhos com mais saltos podem resultar em maiores latências ou até mesmo maior perda de pacotes devido a falhas transientes e intermitentes. Neste âmbito, neste trabalho são apresentadas duas contribuições para caracterizar diversos caminhos entre um par de comunicação: (i) uma métrica analítica capaz de estimar a latência de um caminho baseada na dinamicidade das falhas de todas as conexões envolvidas; e (ii) uma métrica analítica capaz de identificar como um caminho contribui para distribuir o tráfego para um par de comunicação específico. No trabalho aqui proposto caracteriza um conjunto de caminhos, e os utiliza como solução para reduzir o impacto advindo de retransmissões, aumentando o desempenho da rede.

Na Tabela 1 são sintetizados os principais trabalhos e estratégias relacionados com esta dissertação. Destacam-se as abordagens de cada grupo de trabalhos relacionadas nos temas principais aqui descritos: (i) algoritmos de roteamento tolerantes a falha; (ii) códigos detectores e corretores de erro; (iii) mecanismos de retransmissão; (iv) reconfiguração da rede; e (v) múltiplos caminhos.

Tabela 1 – Tabela-resumo dos principais trabalhos relacionados.

ASSUNTO	REFERÊNCIA	ESTRATÉGIA	OBSERVAÇÕES
Algoritmos de roteamento tolerantes a falha	(GOMEZ et al., 2004; HO; STOCKMEYER, 2004)	Modelos de falha estáticos	Quantidade de falhas suportada é limitada
	(FICK et al., 2009; STRANO et al., 2012; TRIVIÑO et al., 2012)	Modelos de falha dinâmicos	Maior complexidade
	(ZHANG et al., 2010)	Uso combinado com mecanismos de hardware	Suporta falhas transientes e permanentes
	(KIA; ABABEI, 2011)	Uso de monitores para evitar congestionamento	É passível de deadlocks
Códigos detectores e corretores de erro	(BERTOZZI et al., 2002)	Modelagem de impacto de ECCs e mecanismos de retransmissão	Concluem que retransmissão possui melhor eficiência energética
	(YU; AMPADU, 2008; YU; AMPADU, 2011)	Utilizam múltiplos SECs para tolerar erros em canais adjacentes	Maior consumo de área e potência
	(GANGULY et al., 2009; ZAMZAM et al., 2011; YU et al., 2011; WANG et al., 2016)	Combinam diferentes ECCs visando eficiência energética	Complexidade dos circuitos necessários para implementação
Mecanismos de retransmissão	(KIM et al., 2005)	Comparação de diferentes mecanismos de retransmissão	Avalia custos de área e impacto nos atrasos dos pacotes
	(PARK et al., 2006)	Variação do tamanho do buffer	Menor sobrecarga quando comparado a técnicas usuais
	(LEHTONEN et al., 2007; YU et al., 2011)	Combinam ECCs, mecanismos de retransmissão e links de reserva	• Proteção completa contra falhas permanentes, transientes e intermitentes • Implementações complexas de software e hardware
Reconfiguração da rede	(FICK et al., 2009; FENG et al., 2013)	Reconfiguração por FTDR e FTDR-H	Redundância
	(STRANO et al., 2012)	Pacotes antigos e novos trafegam simultaneamente na rede	• OSR • Custo de hardware elevado
	(SILVEIRA et al., 2015)	Mecanismo inteligente para avaliar reconfiguração x retransmissão	Necessário definir limiares
Múltiplos caminhos	(YIN et al., 2012)	Melhoria de eficiência energética	Evita congestionamentos
	(KUMAR et al., 2014)	Roteamento não-mínimo	• Restringe caminhos • Melhor distribuição do tráfego
	Este trabalho	**Redução do atraso imposto por mecanismos de retransmissão**	**• Métricas para classificar caminhos considerando probabilidade de falha das conexão • Melhor distribuição do tráfego • Baixa sobrecarga no desempenho da rede**

Fonte: Próprio autor

3 REDES INTRACHIP

O avanço tecnológico no processo de fabricação de transistores CMOS e FinFets, têm permitido uma alta integração de componentes em um único chip. Adicionalmente, também há um aumento do poder de processamento dada a maior quantidade de componentes fortemente acoplados. Diante deste contexto, as infraestruturas de comunicação clássicas se tornaram restritivas para SoCs com muitos componentes. Como uma solução mais adequada para os problemas encontrados, surgiram as redes intrachip (NoCs). As NoCs possuem um elevado grau de paralelismo, alta escalabilidade e reusabilidade da arquitetura de comunicação. Desta forma, neste capítulo são apresentadas as principais características e conceitos das redes intrachip que estão relacionadas com este trabalho.

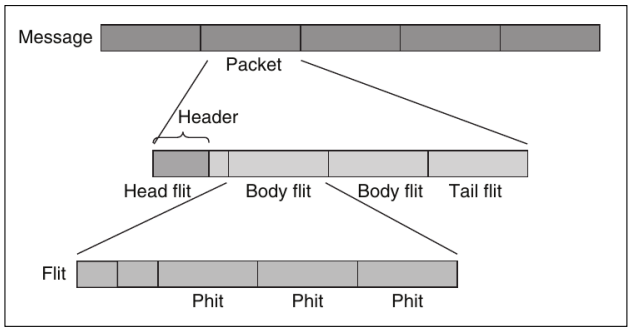
3.1 Conceitos básicos

A inspiração para o desenvolvimento das redes intrachip se iniciou baseada nas redes de computadores e sistemas distribuídos. Adicionalmente, as redes intrachip também fazem uso de diversos conceitos destas áreas, tais como: arquitetura em camadas, diferentes topologias possíveis, algoritmos de roteamento, mecanismos de chaveamento, controle de fluxo e congestão, balanceamento de carga, arbitragem, dentre outros. Para a modelagem de uma NoC normalmente são considerados três elementos principais: os elementos de processamentos (em inglês, *Processing Element* (PE)), os roteadores (em inglês, *switches*) e as conexões (em inglês, *links*). É importante ressaltar que na literatura o par PE-roteador é referenciado como nodo ou nó da rede.

Assim como nas redes de computadores e sistemas distribuídos, a comunicação entre os elementos de processamento ocorre através de trocas de mensagens. Para atender as restrições da arquitetura em camadas, as mensagens são então divididas unidades menores denominadas pacotes. Estes pacotes são constituídos de um cabeçalho (em inglês, *header*), corpo útil de dados (em inglês, *payload*) e cauda (em inglês, *tail*). De forma geral, o cabeçalho especifica dados voltados para o algoritmo de roteamento, o corpo útil de dados representa os dados úteis aos elementos de processamento e a cauda contém dados de redundância destinados a garantir a coerência do pacote que está sendo enviado. Novamente, visando atender restrições da arquitetura em camadas, os pacotes são divididos em *flits* (do inglês *flow control units*) e estes por sua vez são divididos em *phits* (do inglês *physical units*). Na Figura 1 são apresentadas estas

unidades, bem como a relação entre elas.

Figura 1 – Unidades de dados em redes intrachip



Fonte: (PASRICHA; DUTT, 2010)

É importante destacar que existem alguns problemas também relativos as redes e sistemas distribuídos que devem ser evitados nas redes intrachip, pois podem comprometer seu funcionamento. São eles: *deadlocks*, *livelocks* e *starvation*. *Deadlocks* são definidos como uma dependência cíclica de recursos entre os elementos de comunicação, impedindo que estes sejam disponibilizados para transmissão dos pacotes. Já os *livelocks* são definidos como situações em que o pacote jamais consegue atingir seu destino, pois fica realizando caminhos cíclicos na rede que não incluem o roteador destino final. Por fim, o fenômeno de *starvation* consiste de uma espera indefinida por uma quantidade de recursos para transmissão de um pacote, o que acarreta em uma protelação do envio do pacote de forma indefinida. Estes problemas são resolvidos/relativos a algoritmos de roteamento e mecanismos de arbitragem, sendo alvo de estudo da literatura como pode ser visto em (SILBERSCHATZ *et al.*, 1998; AGARWAL *et al.*, 2009; FICK *et al.*, 2009; STRANO *et al.*, 2012; TRIVIÑO *et al.*, 2012).

Na próximas seções são discutidos conceitos importantes relacionados a implementação das redes intrachip, tais como: topologias, chaveamento, memorização, controle de fluxo, arbitragem e roteamento.

3.2 Topologia

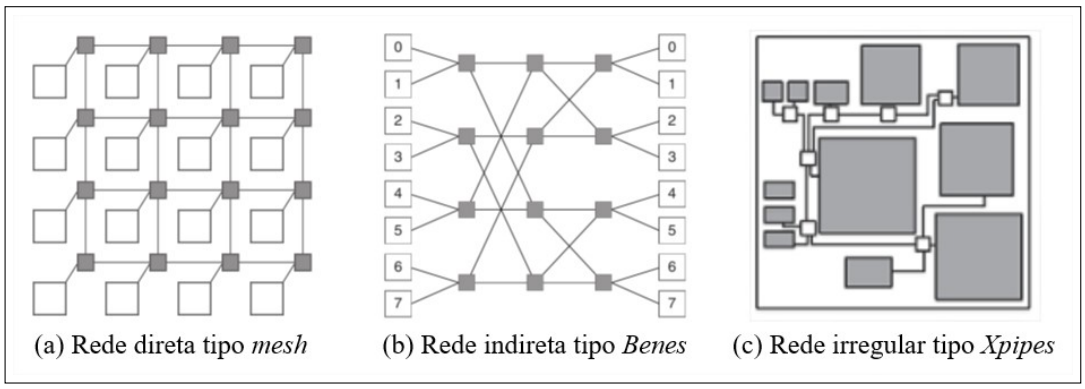
A topologia das redes intrachip diz respeito à forma como os nodos estão conectados entre si. De forma geral, a modelagem utilizada é baseada em grafos, em que os nodos são considerados como os vértices e as conexões os arcos. A definição da topologia a ser utilizada caracterizará o comportamento de largura de banda e latência da rede. Adicionalmente, há um impacto na escolha do algoritmo de roteamento e do controle de fluxo.

De acordo com (DUATO *et al.*, 2003) e considerando as restrições de arquitetura das NoCs, podemos classificar as redes em três grupos de acordo com sua topologia: redes diretas, indiretas e irregulares.

As redes diretas caracterizam-se por realizarem uma associação entre os roteadores e os elementos de processamento. Desta forma, os nós atuam tanto como fonte ou destino de pacotes. Já nas redes indiretas, não há uma associação direta entre os elementos de processamento e os roteadores. Assim, podem existir roteadores intermediários entre os elementos de processamento, de tal sorte que hajam roteadores que não são conectados a nenhum elemento de processamento, invalidando assim o conceito de nodo. Por fim, as redes irregulares são implementadas a partir de uma combinação das características das redes diretas e indiretas de forma a atender a vasta diversidade de componentes nos SoCs. Para tal, é necessário que no projeto da topologia sejam considerados diversos parâmetros, por exemplo o número de componentes e restrições de compatibilidade/conectividade.

Na Figura 2 temos exemplificados três tipos de topologia de rede. Maiores detalhamentos e caracterizações de cada um destes tipos pode ser encontrado em (DUATO *et al.*, 2003; GOOSSENS *et al.*, 2005; PEH; JERGER, 2009; PASRICHA; ZOU, 2011).

Figura 2 – Exemplos de tipos de topologias de rede



Fonte: (BERTOZZI; BENINI, 2004; PASRICHA; DUTT, 2010)

3.3 Mecanismos de chaveamento

A estratégia empregada para que o pacote seja transmitido da fonte até seu destino é denominada mecanismo de chaveamento. Assim como nas redes de computadores, as duas principais estratégias de chaveamento utilizadas nas redes intrachip são: chaveamento por circuito e chaveamento por pacote.

3.3.1 Chaveamento por circuito

No chaveamento por circuito há uma alocação prévia das conexões necessárias para transmissão do(s) pacote(s) da fonte até o destino previamente. Inicialmente, antes do envio dos pacotes, ocorre o estabelecimento de uma conexão para reserva dos recursos e, em seguida, são enviados os pacotes. A seguir, apresentamos algumas vantagens e desvantagens desta estratégia.

Vantagens:

- Não é necessário que hajam mecanismos de memorização nos roteadores (*buffers*), o que resulta em uma menor área ocupada pelo roteador;
- Realizada a reserva de recursos, a latência do pacote sempre será a mesma, dado que o pacote percorrerá as mesmas conexões. Adicionalmente, não há competição pelo uso das conexões.

Desvantagens

- Atraso inicial durante a reserva das conexões, o que pode ser prejudicial para sistemas com requisitos de baixas demandas de latência;
- Com o aumento da rede, há um aumento da alocação de recursos podendo assim comprometer toda a rede;
- As conexões permanecem ocupadas enquanto houver comunicação entre o par de comunicação que realizou a reserva de recursos, gerando assim uma baixa eficiência de comunicação.

3.3.2 Chaveamento por pacote

Nesta estratégia a transferência dos pacotes entre fonte e destino é realizada de forma dinâmica e definida em tempo de execução do roteamento. Como não há uma reserva de recursos para envio dos pacotes, normalmente se faz necessário que os elementos de roteamento empreguem mecanismos de memorização para armazenarem temporariamente os pacotes a serem transferidos. A seguir, apresentamos algumas vantagens e desvantagens desta estratégia.

Vantagens:

- Como não há reserva de recursos, a rede como um todo pode ser melhor utilizada, melhorando assim sua eficiência de comunicação;
- Maior velocidade de transferência de pacotes.

Desvantagens

- Atrasos variáveis devido aos diversos caminhos que podem ser tomados em tempo de roteamento;
- Entrega de pacotes foram de ordem, sendo necessário ordenação no destino final.

De forma geral, pode-se ainda dividir o chaveamento por pacotes em três diferentes técnicas: *Store and forward*, *Vitual Cut Through* e *Wormhole*.

No chaveamento do tipo *Store and forward* o roteador não pode repassar o pacote até que o tenha recebido de forma completa. Desta forma, é necessário que os roteadores tenham *buffers* capazes de armazenar todo o pacote. Em consequência, com uma maior demanda de tamanho de *buffer* há um aumento significativo na área ocupada pelo elemento de roteamento. Adicionalmente, temos uma maior latência, dado que cada roteador deverá esperar receber o pacote completo e só então repassá-lo (PEH; JERGER, 2009).

Na estratégia do tipo *Virtual Cut Through* o pacote é transferido de um roteador ao próximo desde que haja garantias de espaço para armazenar o pacote inteiro no próximo roteador. Desta forma há uma redução da latência quando comparado ao modo *Store and forward* mas os requisitos de memorização são praticamente os mesmos (KERMANI; KLEINROCK, 1979).

Já a estratégia do tipo *Wormhole* consiste de uma variação da estratégia *Virtual Cut Through*, em que não é necessário que haja espaço suficiente para o pacote inteiro no próximo roteador e sim para apenas um *flit*. Desta forma, os pacotes são enviados sequencialmente na rede, permitindo um melhor uso da diversidade de caminhos (DALLY; SEITZ, 1986).

3.4 Mecanismos de memorização

Dependendo da estratégia de chaveamento utilizada, pode ser necessário que os roteadores armazenem de forma temporária os pacotes antes de repassá-los na rede. Neste preceito, é definido o mecanismo de memorização, ou seja, a utilização de *buffers*. Diversas estratégias podem ser utilizadas, em que cada uma delas é diretamente relacionada com o controle de fluxo dos pacotes. Portanto, a escolha do mecanismo de memorização afeta diretamente a vazão de dados da rede.

Neste âmbito, diferentes mecanismos são propostos na literatura com destaque para: armazenamento na entrada, compartilhado e na saída. No abordagem utilizando armazenamento na entrada temos memorização nas entradas dos roteadores. É bastante simples, pois sua implementação realiza a leitura dos dados dos pacotes na mesma ordem na qual foram escritos. Já para no armazenamento compartilhado temos um comportamento similar, com alteração

apenas da memória a ser utilizada, pois há uma distribuição dos dados do pacote entre as portas de entrada. Por fim, no armazenamento de saída, o processo de bufferização é realizado nas portas de saída do roteador. Cada porta de saída possuirá uma quantidade *N* de filas, sendo *N* a quantidade de portas de entrada do roteador. Um estudo mais detalhado e comparativo destas abordagens podem ser melhor analisado em (KUMAR *et al.*, 2004).

3.5 Controle de fluxo

O controle de fluxo é responsável pela alocação de recursos de memorização e das conexões da rede, controlando, assim, a permissão de envio de dados entre os roteadores. Desta forma, o método de controle de fluxo utilizado é capaz de determinar a vazão da rede, bem como o uso eficiente das conexões ociosas e dos mecanismos de memorização disponíveis na rede.

Existem basicamente três estratégias para controle de fluxo: *Handshake*, *On-off* e baseado em créditos (PEH; JERGER, 2009). A técnica de *Handshake* apresenta ineficiência no uso dos mecanismos de memorização se tornando de difícil utilização nas redes intrachip (DALLY; TOWLES, 2004). Na abordagem *On-Off* temos uma sinalização binária de permissão, em que um sinal é enviado do roteador destino para o roteador fonte indicando se há ou não permissão de envio de dados. Para tal, são estabelecidos níveis de ocupação dos *buffers* e, assim, o sinal de permissão (sinal do tipo *on*) só é enviado quando o nível de ocupação do *buffer* estiver adequado. Caso contrário, o sinal de proibição de envio é enviado (sinal do tipo *off*) . Adicionalmente, esta abordagem permite reduzir o número de sinalizações utilizadas, no entanto pode requerer *buffers* com tamanhos maiores, o que afetará a área ocupada (PEH; JERGER, 2009). Por fim, na abordagem baseada em créditos a transmissão de dados do roteador fonte ao roteador destino só é permitida se houver espaço disponível no *buffer* do roteador destino. Para identificar quando há ou não espaço disponível, utiliza-se uma variável denominada crédito, esta por sua vez representa o espaço livre no *buffer* do roteador destino (PEH; JERGER, 2009). A desvantagem desta abordagem é que o número de créditos é diretamente relacionado ao número de *flits* e não de pacotes, o que eleva consideravelmente a quantidade de créditos necessária.

3.6 Arbitragem

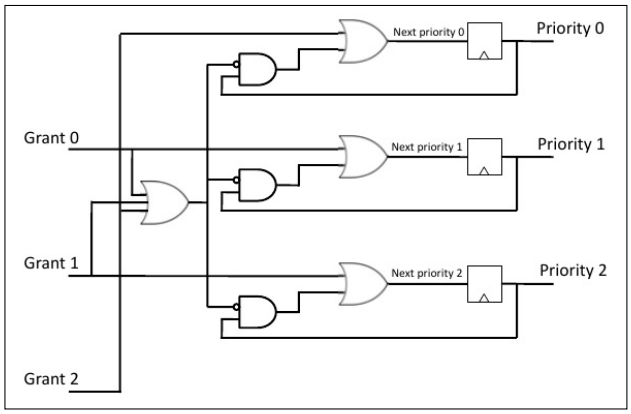
Os algoritmos de arbitragem são utilizados como forma de resolver os possíveis conflitos na alocação de recursos pois são capazes de gerenciar o acesso aos recursos compartilhados

no roteador. Desta forma, quando ocorre uma solicitação do mesmo recurso de roteamento advinda de diferentes portas de entrada para a mesma porta de saída, cabe ao árbitro tomar a decisão (prioridade) para quem terá acesso ao recurso primeiro.

A importância da escolha adequada do mecanismo de arbitragem é que este é capaz de mitigar situações de *starvation*. Diversos mecanismos são propostos na literatura, tais como: prioridade estática, prioridade dinâmica, *First Come First Served*, *Round Robin*, dentre outros. Detalhes de implementação, vantagens, desvantagens e comparativos podem ser encontrados em (SILBERSCHATZ *et al.*, 1998; AGARWAL *et al.*, 2009). Para este trabalho, convém falarmos brevemente do mecanismo do tipo *Round Robin* (Figura 3).

Nesta estratégia temos um árbitro circular de tal forma que a quando uma solicitação é atendida ela terá a menor prioridade na próxima rodada do árbitro. Desta forma, é intuitivo perceber que temos uma arbitragem justa, em que todas as solicitações são atendidas de forma igualitária (PEH; JERGER, 2009).

Figura 3 – Circuito que exemplifica o árbitro do tipo *Round-Robin*.



Fonte: (PEH; JERGER, 2009)

3.7 Roteamento

O algoritmo de roteamento define o caminho que os pacotes vão seguir ao longo da rede. As características ideais de um algoritmo de roteamento pode ser feita baseada em diferentes critérios, tais como:

- Ausência de *livelocks* e *deadlocks*;
- Tolerância a falhas, ou seja, o algoritmo deve ser capaz de garantir a entrega dos pacotes perante qualquer tipo falhas na rede;
- Minimizar congestionamentos através da distribuição do tráfego por toda a rede;

- Adaptatividade, ou seja, o algoritmo de roteamento deve ser capaz de lidar com situações de falhas e congestionamento na rede;

Além disso, em (DUATO *et al.*, 2003) é proposto uma classificação dos algoritmos sob diversos parâmetros:

- Decisão do roteamento: centralizada, quando todos os caminhos do pacote são estabelecidos por um único nodo; fonte quando o todo o caminho é estabelecido no roteador fonte; ou distribuída quando os caminhos do pacote são estabelecidos pelos roteadores ao longo da rede. Na decisão centralizada temos como vantagem a garantia de entrega dos pacotes de forma sequencial. No entanto, não há considerações quando a dinamicidade do tráfego na rede. Para decisão realizada na fonte temos uma menor latência visto que o caminho é predefinido, contudo o cabeçalho do pacote tem tamanho sobrecarregado pelos dados referentes ao caminho completo. Já na decisão distribuída as decisões são feitas pelos roteadores ao longo do caminho do pacote, em que o o cabeçalho do pacote contém apenas o endereço do próximo roteador.
- Realização do roteamento: estática, quando é implementada durante a compilação e dinâmica, quando é implementada em tempo de execução. Na abordagem estática não é possível atualizar os caminhos em tempo de execução, apresentando assim uma dificuldade de eficiência quando a rede apresenta falhas, sobretudo do tipo permanente.
- Adaptatividade: determinístico, quando o caminho entre fonte e destino percorrido é sempre o mesmo e adaptativo, quando o caminho percorrido é dinâmico, ou seja, determinado por diversos fatores da rede, tais como: falhas ou congestionamentos.
- Comprimento dos caminhos: utilização apenas de caminhos mínimos e/ou não mínimos. A utilização de caminhos mínimos é empregada no intuito de diminuir latência, no entanto há um uso ineficiente da rede, visto que o tráfego ficará concentrado apenas nos caminhos mínimos. Quando da utilização de caminhos não mínimos temos maiores possibilidades de roteamento, o que contribui para dispersar o tráfego. Todavia, é necessário avaliar a viabilidade do uso destes caminhos, uma vez que há maior consumo de energia.
- Disponibilidade de caminhos: parcial (quantidade de caminhos limitados) ou completo (todos os possíveis caminhos podem ser utilizados). De forma similar ao item anterior, a restrição de caminhos impede um uso eficiente da arquitetura de comunicação, dado que muitas conexões podem permanecer ociosas enquanto outras altamente congestionadas.
- Implementação do algoritmo: baseado em tabelas (armazenadas na memórias do rotea-

dores) ou baseado em máquinas de estado (algoritmos predefinidos implementados em software/hardware). O roteamento baseado em tabelas tem uma excelente vantagem no quesito reconfiguração, dado que as tabelas podem ser atualizadas em tempo de execução para atender a dinamicidade da rede (falhas e/ou congestionamentos). No entanto, o armazenamento das tabelas no roteadores pode ser proibitivo a depender do tamanho da rede, visto que implica diretamente em um maior consumo de área. Para a abordagem baseada em máquinas de estado temos como grande vantagem a simplicidade de implementação e agilidade no roteamento. Todavia, nesta técnica não há possibilidades de reconfiguração em tempo de execução, logo falhas e congestionamentos certamente afetarão a eficiência da rede.

Maiores detalhes e características sobre diversos algoritmos de roteamento podem ser encontrados em (DALLY; TOWLES, 2004; PEH; JERGER, 2009). Adicionalmente, no Capítulo 2 é realizada um estudo do estado da arte para algoritmos de roteamento tolerantes a falha, foco deste trabalho.

4 METODOLOGIA

Neste capítulo é apresentada a técnica proposta para redução de latência, as métricas propostas para classificar diferentes possíveis caminhos entre um par de comunicação: (i) estimativa da latência de um caminho; e (ii) como um caminho contribui com novos links para distribuir o tráfego. Adicionalmente, também são descritos: configuração experimental empregada para validação da técnica proposta, a NoC tolerante a falhas utilizada, o algoritmo de roteamento e o mecanismo de implementação empregados, as ferramentas desenvolvidas para geração e análise dos arquivos de tráfego, bem como é realizada a geração dos cenários de falha.

4.1 Abordagem baseada em Múltiplos Caminhos

A estratégia geral adotada neste trabalho consiste na utilização de múltiplos caminhos como forma de reduzir latência em NoCs tolerantes a falha que utilizam mecanismos de retransmissão. Em especial, neste trabalho foca-se em sistemas críticos de comunicação, tais como: sistemas de aplicações biomédicas, automotivos, militares, dentre outros do mesmo gênero de criticidade; para estes sistemas foi considerado um e apenas um par crítico. Consideramos ainda que este par crítico de comunicação como sendo tendo um alto volume de comunicação e é essencial para o bom funcionamento do sistema como um todo. Além disso, é importante ressaltar que não foram utilizados mais pares críticos, porque tornar outros pares críticos não irá aumentar o volume de comunicação da rede, apenas dividi-lo entre os vários caminhos. Adicionalmente, a cada novo par tornado crítico, há uma contribuição para a diminuição do desvio padrão do peso médio das conexões da rede. Desta forma, com muitos pares críticos, teremos um peso médio das conexões distribuído uniformemente, não sendo vantajoso utilizar mais de um caminho. Não obstante, dada a quantidade de caminhos entre um determinado par, utilizar uma quantidade pequena de pares críticos torna difícil a obtenção dos dados. Teríamos combinações entre os diferentes caminhos para os diferentes pares críticos. Neste trabalho temos como foco validar a ideia da abordagem utilizando múltiplos caminhos. Desta forma, a combinação/inclusão de novos pares críticos pode ser alvo de um novo trabalho, a partir do pré-processamento de caminhos em tempo de execução.

Neste trabalho são avaliados e comparados caminhos de comunicação para o mesmo par fonte-destino objetivando construir a tabela de roteamento de um nodo. Todos os possíveis caminhos entre um par de comunicação são pesquisados e ordenados de forma crescente de

acordo com sua *latência estimada* (descrita na Seção 4.2.1). O primeiro conjunto é composto de apenas um único caminho e o enésimo conjunto é composto dos *n* primeiros caminhos.

É esperado que caminhos não mínimos tenham maiores latências, entretanto podem existir caminhos mínimos com maiores probabilidades de falhas. Estes, por sua vez, podem gerar retransmissões desnecessárias, aumentando a latência. Adicionalmente, inserir novos caminhos também resulta na inserção de novas conexões que ainda não foram utilizadas pelos caminhos já acrescentados. Desta forma, a maior diversidade de links também contribui para eficiência da técnica, pois melhora a distribuição do tráfego para o par em questão. É desejável também que a latência média de toda a rede não sofra impactos negativos severos, de forma que não invalide a solução proposta.

4.2 Métricas propostas

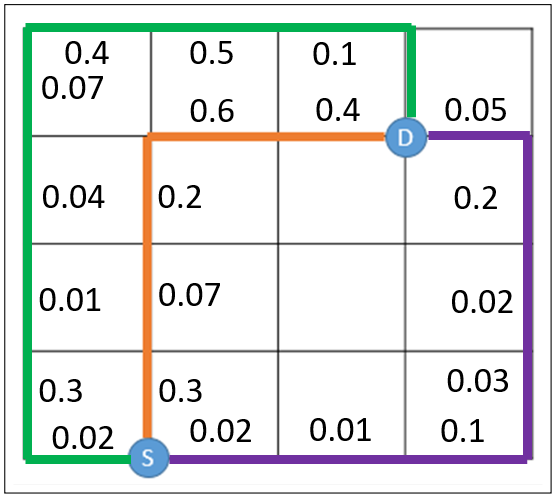
4.2.1 Latência Estimada

Para estimar a latência real de um caminho, neste trabalho é proposta uma métrica denominada *latência estimada*. Esta métrica toma como base o número de saltos do caminho e a probabilidade de falha dinâmica de cada conexão envolvida no caminho para estimar a latência real do mesmo. Assim, é possível ordenar os caminhos de forma crescente de acordo com sua *latência estimada*. Logo, caminhos estimados como mais rápidos são adicionados/utilizados primeiro. Dado que a *latência estimada* de um caminho depende não apenas da quantidade de saltos e da probabilidade de falha de cada conexão, caminhos com muitos saltos ou com conexões que tenham grandes probabilidades de falhas tendem a ser mais lentos; portanto, estes caminhos devem ser utilizados posteriormente.

Na Figura 4 é apresentado a justificativa para a obtenção da *latência estimada*, ilustra-se uma NoC 5x5 de topologia do tipo malha, ressaltando a fonte (S) e o destino (D) e três caminhos para o mesmo par comunicante. O caminho em laranja (A), com 5 saltos (*hops*), representa um caminho mínimo. Já o caminho de cor roxa (B) representa um caminho alternativo e não mínimo, possuindo 7 saltos. Por fim, o caminho verde (C) consiste de outro caminho alternativo, também não mínimo, porém com mais saltos, 9. Para cada caminho são apresentadas as probabilidades de falha de cada conexão envolvida. É possível observar que o caminho não mínimo B apresenta menores probabilidades de falha que o caminho mínimo A. Consequentemente, o caminho A mesmo sendo mínimo, pode apresentar uma latência maior.

Entretanto, há casos em que um caminho tem menor probabilidade de falha que outro, mas por ter muito saltos, tende a apresentar maior latência. Este caso é representado na comparação entre os caminhos A e C.

Figura 4 – Três possíveis caminhos entre um par fonte-destino de comunicação justificando a necessidade da métrica *latência estimada*.

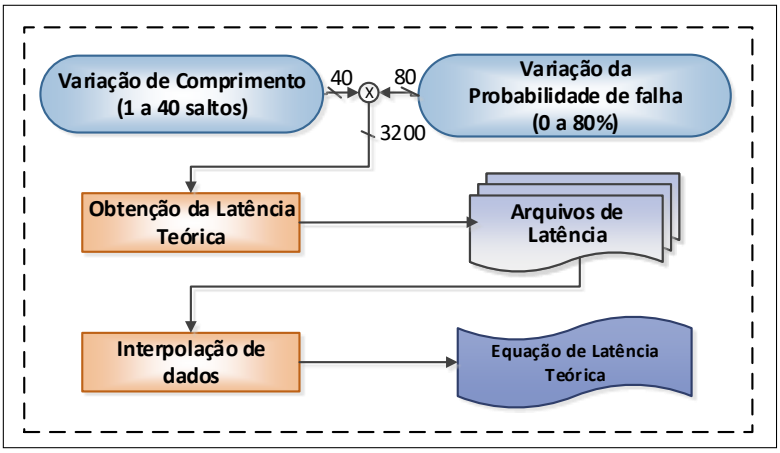


Fonte: Próprio autor

Para estimar a latência de um caminho foi implementado um experimento para modelar como percentuais de falha e quantidade de saltos de um caminho impactam a latência da rede. Utilizando interpolação polinomial sobre os dados de saída deste experimento, permitiu-se definir a métrica *latência estimada* de um caminho com N saltos e probabilidade de falha P para um par comunicante (x, y) da rede.

As simulações consistiram de múltiplas execuções com variação do caminho, de 1 a 40 saltos, e para cada variação desta, uma variação de 0 a 80% do percentual de falhas do caminho. Estes parâmetros são baseados na maior quantidade possível de saltos e no maior percentual de falha encontrados nos cenários de falha gerados na Seção 4.3. A arquitetura alvo tolerante a falhas para o experimento foi a NoC Phoenix (vide Seção 4.4). Utilizamos uma taxa de injeção de 30% para contornar possíveis problemas de saturação da rede. O ponto de saturação é definido como a taxa de injeção a partir da qual a rede não é mais capaz de responder de forma equivalente. A partir desse ponto, o tráfego aceito médio não aumenta de forma significativa e a latência média passa a crescer acentuadamente, prejudicando os resultados. Na Figura 5 é apresentado o diagrama de fluxo de cada iteração da simulação.

Figura 5 – Diagrama ilustrativo do experimento realizado para obtenção da equação da *latência estimada*.



Fonte: Próprio autor

Após obter os valores de latência para cada interação, realizamos uma interpolação polinomial para encontrar a equação que representa a latência de um caminho a partir do seu comprimento e probabilidade de falha. A probabilidade de falha de um caminho P é encontrada a partir da formulação matemática da união de N eventos independentes, N é o número de saltos do caminho e seu valor é dado pela probabilidade de falha de cada respectiva conexão (vide equação 4.1).

$$P(qualquer\ evento) = 1 - P(\bar{A}).P(\bar{B})... = 1 - [(1 - P(A)).(1 - P(B))...] \tag{4.1}$$

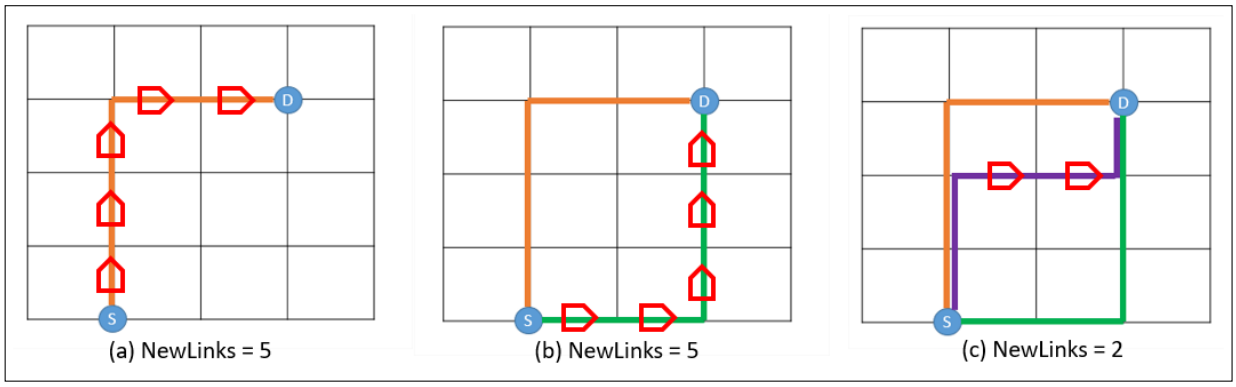
4.2.2 NewLinks

Para identificar como um caminho contribui no aumento da diversidade de conexões disponíveis para o par fonte-destino, neste trabalho é proposta uma métrica denominada *NewLinks*. Esta métrica representa o número de conexões (em inglês, *links*) que ainda não foram utilizadas por caminhos previamente acrescentados. A inserção de novos caminhos com altos valores de *NewLinks* contribui para distribuir o tráfego para o par de comunicação em questão, reduzindo, assim, o impacto gerado por retransmissões e, consequentemente, reduzindo a latência. Embora caminhos com muitos saltos possam contribuir positivamente para maiores valores de *NewLinks*, o número de saltos pode aumentar a latência. Assim, é necessário avaliar até quando é viável aumentar a diversidade de conexões.

Na Figura 6 apresenta três possíveis caminhos para um mesmo par fonte (S) - destino

(D), em uma NoC 5x5 de topologia do tipo malha. Na Figura 2(a) é representada a seleção de um caminho contendo cinco novas conexões. Já na Figura 2(b) apresenta a inserção e seleção de um novo caminho para o mesmo par de comunicação. É possível observar que este novo caminho não reutiliza nenhuma conexão previamente utilizada pelo caminho da Figura 2(a), representando, então, um valor de *NewLinks* = 5 para o caminho da Figura 2(b). Entretanto, na Figura 2(c) apresenta um caso para um outro caminho alternativo que reutiliza algumas conexões previamente utilizadas pelos caminhos já inseridos, desta forma este caminho apresenta apenas *NewLinks* = 2.

Figura 6 – Exemplos de três possíveis caminhos entre um par fonte-destino de comunicação ilustrando a métrica *NewLinks*.

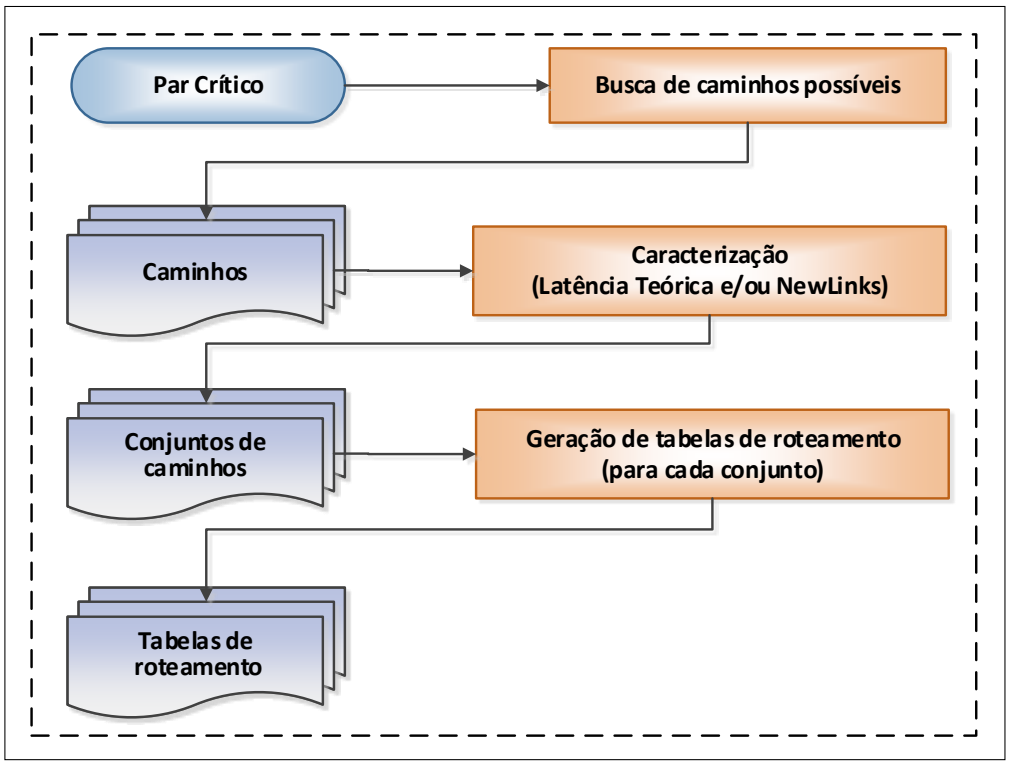


Fonte: Próprio autor

Para calcular as novas conexões ainda não utilizadas a partir da adição de um novo caminho é realizada uma comparação das conexões presentes no caminho a ser acrescentado e das conexões que já foram inseridas pelos caminhos já adicionados para o par crítico. Desta forma, a ordem em que os caminhos são adicionados influencia diretamente no seu respectivo valor para a métrica *NewLinks*.

Com base nestas duas métricas é possível caracterizar todos os caminhos entre um par fonte-destino. Na Figura 7 é apresentado o fluxo que descreve o funcionamento da técnica proposta a partir da utilização de caminhos mínimos e não mínimos. Para avaliar a eficiência da técnica proposta, foi realizada a implementação de uma NoC com arquitetura tolerante a falhas que utiliza mecanismos de retransmissão.

Figura 7 – Diagrama representativo da técnica proposta.



Fonte: Próprio autor

4.3 Geração dos cenários de falha

A variabilidade do processo de fabricação de circuitos VLSI aumenta com a redução das dimensões dos dispositivos devido a fenômenos como a imprecisão nos processos de deposição dos materiais dopantes e campo de exposição de processos litográficos não uniforme. Esta variabilidade pode causar desvios na especificação nominal do circuito ou mesmo impedir o seu funcionamento parcial ou total (SHINTANI *et al.*, 2014), sendo uma fonte de falhas dinâmicas e estáticas. Portanto, e sem perda de generalidade, é utilizado o modelo de variabilidade proposto por Hargreaves et al. (HARGREAVES *et al.*, 2008) para gerar cenários de falha sintéticos. Este modelo considera o efeito da variabilidade nos atrasos de conexões, roteador a roteador, empregando dois parâmetros de variação: a variabilidade do atraso σ e a variabilidade na força de correlação espacial λ.

No sentido de explorar cenários com processos de fabricação CMOS de 65 *nm* e 22 *nm*, a variabilidade de atraso nas conexões foi definida como 5% ($\sigma = 0.05$) e 18% ($\sigma = 0.18$), respectivamente, como previsto pelo ITRS (HOEFFLINGER, 2011). Adicionalmente, os experimentos foram produzidos com $\lambda = 0.4$ e $\lambda = 1.2$, representando a força de alta e baixa

variabilidade da correlação espacial, nesta mesma ordem. Estes valores são representativos da correlação típica induzida por processos de fabricação para as referidas tecnologias de fabricação(HARGREAVES *et al.*, 2008). Estes dois valores de variabilidade para cada parâmetro permitem combinar 4 tipos de distribuição (tipo 1, 2, 3 e 4). Na Tabela 2 é mostrado que os experimentos contaram com 100 amostras para cada tipo de distribuição, gerando assim um total de 400 topologias de NoCs. O tamanho de NoC utilizado foi 5x5, uma vez que NoCs de tamanhos maiores requerem muitos caminhos entre um par de comunicação aumentando muito o tempo de computação e simulação para obtenção de resultados.

Tabela 2 – Cenários de distribuição de falhas gerados a partir do modelo de variabilidade para processos de fabricação CMOS.

Distribuição (tipo)	Variabilidade (σ)	Correlação espacial (λ)	Processo de Fabricação
1	0.05 (fraca)	1.2 (fraca)	65 *nm*
2	0.18 (forte)	1.2 (fraca)	65 *nm*
3	0.05 (fraca)	0.4 (forte)	22 *nm*
4	0.18 (forte)	0.4 (forte)	22 *nm*

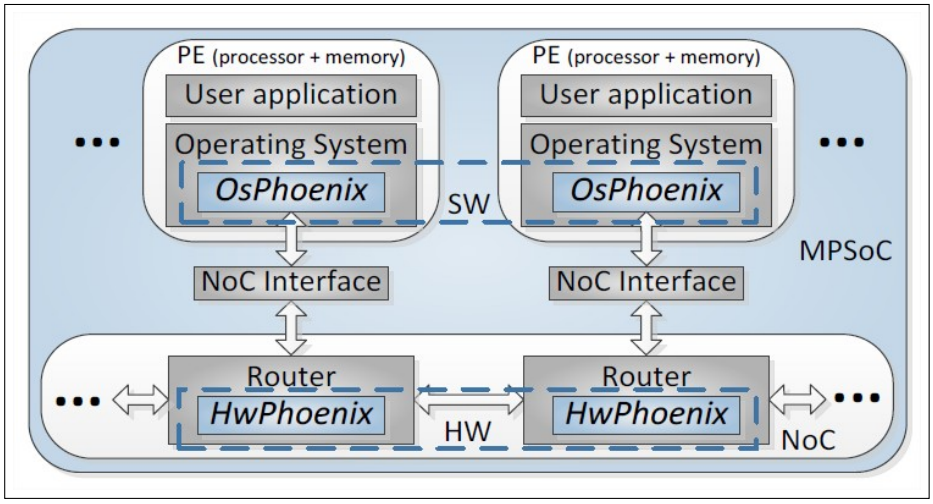
Fonte: Próprio autor

4.4 NoC empregada

As simulações foram feitas com a NoC Phoenix (MARCON *et al.*, 2013) implementada na linguagem de descrição de hardware *VHSIC Hardware Description Language* (VHDL), no ambiente de simulação ModelSim. A NoC Phoenix dispõe de conexões tolerantes a falhas com um mecanismo implementado tanto em hardware quanto em software.

Na Figura 8 é mostrada a arquitetura básica da Phoenix, compreendendo uma parte em hardware (HwPhoenix), presente em cada roteador da NoC e uma parte em software (OsPhoenix), presente em cada sistema operacional de cada elemento de processamento (em inglês, PE) composto de um par processador-memória. Adicionalmente, cada PE é conectado à porta local do roteador através de uma interface de rede (em inglês, *Network Interface* (NI)).

Figura 8 – Arquitetura da Phoenix.



Fonte: (MARCON *et al.*, 2013)

A NoC Phoenix tem topologia malha 2D com m linhas e n colunas, consistindo de $m \times n$ roteadores com conexões bidirecionais posicionadas entre roteadores e entre roteador e PE. A NoC emprega tabelas de roteamento para decisões de roteamento distribuído e o OsPhoenix executa algoritmos de roteamento (segmentação da rede e cálculo de regiões virtuais, descritos nas Seções 4.5 e 4.6) para preencher as tabelas de roteamento de acordo com a posição de cada PE e das conexões com falha. A NoC Phoenix implementa chaveamento *wormhole*, o qual divide os pacotes, demandando pequenos *buffers* para armazenamento de dados. Adicionalmente, a NoC Phoenix usa controle de fluxo do tipo *on-off*, para reduzir o número de fios de sinalização. A Phoenix usa dois tipos de pacotes: (i) pacote de dados, que transporta as mensagens entre PEs; e (ii) pacote de controle, que controla os mecanismos de tolerância a falhas. O OsPhoenix comunica-se com o HwPhoenix através de pacotes de controle bidirecionais via porta local de cada PE. Maiores características e detalhes podem ser encontrados em (MARCON *et al.*, 2013). Para este trabalho, é relevante destacar que a NoC Phoenix se recupera de falhas dinâmicas a partir de mecanismos de retransmissão.

Para realização de todos os experimentos descritos neste trabalho, foram utilizadas as seguintes configurações: a topologia utilizada foi do tipo malha regular de dimensão 5x5. O roteador empregado possui *buffers* de entrada com capacidade para 16 *flits* e o chaveamento é do tipo *wormhole*. O controle de fluxo é *on-off*, a arbitragem do tipo *round-robin* e o roteamento implementado por tabelas distribuídas. Na Tabela 3 são resumidas as configurações escolhidas para as simulações.

Tabela 3 – Configurações da NoC Phoenix utilizadas neste trabalho.

Topologia	mesh
Buffer	16 bits (entrada)
Chaveamento	wormhole
Controle de Fluxo	on-off
Árbitro	Round-robin
Roteamento	Distribuído, baseado em tabelas

Fonte: Próprio autor

4.5 Algoritmo de roteamento

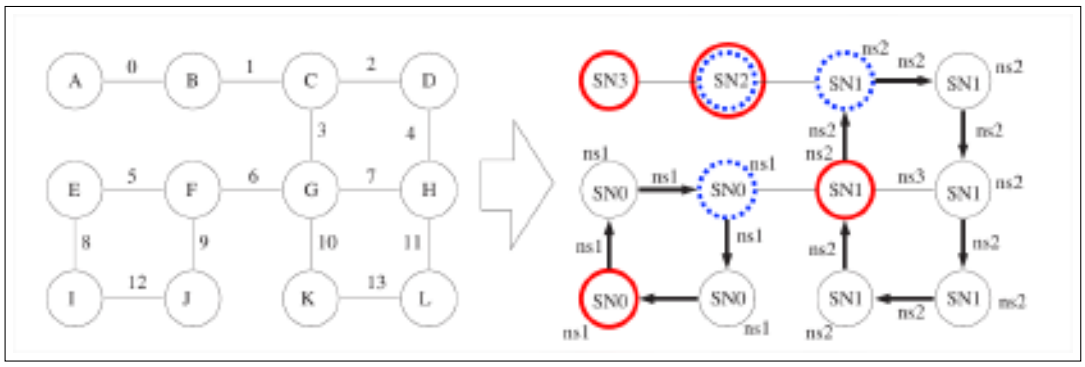
O roteamento baseado em regiões (em inglês *Segment-based Routing* (SBR)) é um método, independente de topologia, de geração de algoritmos de roteamento livres de *deadlock* baseado na inserção de restrições de roteamento.

Ele divide a topologia em sub-redes, e essas em segmentos, e a cada segmento atribui uma restrição de roteamento bidirecional (MEJIA *et al.*, 2006). A independência entre os segmentos garante maior liberdade na atribuição das restrições de roteamento em relação a outros métodos como os aplicados em (ALFARO *et al.*, 2000; SANCHO *et al.*, 2000).

As sub-redes são os subgrafos da topologia que se separam desta por apenas uma conexão, chamada *conexão ponte*. Essas sub-redes podem ser tratadas separadamente, já que não há possibilidade de ciclos entre elas desde que as restrições não sejam adicionadas a *conexão ponte*. Pelo mesmo motivo não há necessidade de atribuição de restrições às *conexões ponte*.

Cada sub-rede possui no máximo um segmento do tipo *início*, que é um conjunto de conexões e nodos (ou vértices e arestas) que forma um ciclo. Os segmentos restantes (do tipo regular ou unitário) têm as extremidades ligadas a outros segmentos. Além disso, não há cruzamento entre segmentos (MEJIA *et al.*, 2008). A Figura 9 ilustra como uma NoC pode ser segmentada.

Figura 9 – Exemplo de segmentação de uma rede.



Fonte: (MEJIA *et al.*, 2006)

4.6 Mecanismo de implementação - Construção das Tabelas de Roteamento

O roteamento baseado em regiões (em inglês *Region-based Routing* (RBR)) implementa um algoritmo de roteamento que usa tabelas distribuídas para diminuir o número de entradas nas tabelas através da redução da redundância, que é tipicamente presente nessas estruturas (FLICH *et al.*, 2007). A ideia subjacente ao seu funcionamento é agrupar em regiões os destinos que possuem características semelhantes quanto às opções de roteamento.

De posse de todos os caminhos de roteamento a serem usados na rede, o algoritmo inicia gerando tuplas compostas pela porta de entrada *Pin*, porta de saída *Pout* e o destino *dst*, para cada roteador do caminho de roteamento. A essas tuplas, (*Pin*; *dst*; *Pout*) representam as opções de roteamento.

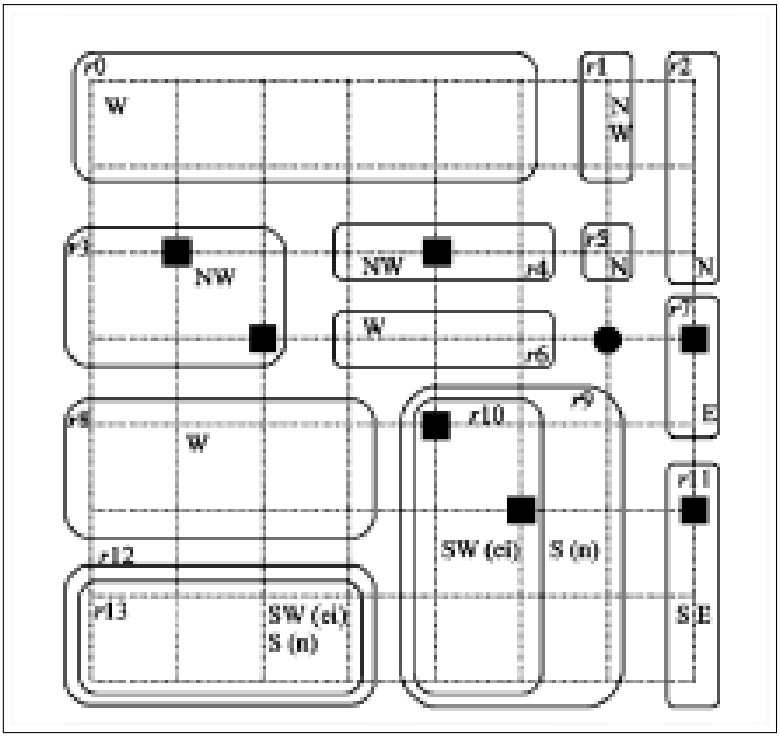
Em uma segunda etapa, essas opções de roteamento são combinadas: tuplas em um mesmo roteador, que possuem o mesmo destino podem ser combinadas por porta de entrada ou de saída. Por exemplo, se duas tuplas possuem o mesmo conjunto de portas de entrada elas podem ser combinadas (por portas de saída) em uma tupla em que o conjunto de portas de saída é a união dos conjuntos de portas de saída das duas.

Numa terceira fase, as tuplas resultantes da fase anterior são agrupadas em regiões de roteamento. Para cada roteador, são agrupadas opções de roteamento que possuem as mesmas portas de saída e o mesmo conjunto de portas de entrada.

Finalmente o algoritmo tenta combinar as regiões para limitar ao máximo a quantidade de posições nas tabelas de roteamento. Isso é feito restringindo o roteamento: é reduzido o número de portas de saída que pode ser usado em um dado roteador para alcançar um dado conjunto de destinos.

A Figura 10 exemplifica as regiões geradas pelo algoritmo para um roteador específico (marcado com ■). As restrições de roteamento foram definidas no processo de segmentação do SBR.

Figura 10 – Exemplo de regiões do RBR.



Fonte: (FLICH *et al.*, 2007)

As implementações das ferramentas descritas em 4.5 e 4.6, foram realizadas em Java. Onde a segmentação da rede e, por sua vez, as tabelas de roteamento é obtida a partir da leitura dos cenários de falhas gerados (vide Seção 4.3). Mais informações e o download do repositório desta ferramenta podem ser encontrados em <https://github.com/LaNoC-UFC/rsbr>.

4.7 Geração dos arquivos de tráfego

Para gerar os arquivos de tráfego sintético (como arquivos de texto) que atuam como entrada para as redes intrachip foi desenvolvida uma ferramenta implementada em Java. Esta ferramenta foi nomeada como *TrafficGenerator* e é baseada na ferramenta descrita em <https://corfu.pucrs.br/redmine/projects/atlas/wiki/Traffic_Mbps>. Adicionalmente, esta ferramenta pode ser utilizada para redes intrachip do tipo malha de qualquer dimensão.

É possível configurar a ferramenta para taxas de injeção, tamanho de flit e pacotes desejados. Além disso, são suportados os seguintes tipos de tráfegos: uniforme, *bit reversal*,

butterfly, *perfect shuffle*, *matrix transpose*, *complement* e *hotspot*.

Mais informações e o download do repositório desta ferramenta podem ser encontrados em <https://github.com/LaNoC-UFC/traffic-gen>.

4.8 Análise dos arquivos de simulação

Para avaliar e analisar os arquivos de simulação gerados após a simulação de uma rede intrachip e extrair diversas informações relevantes para um par comunicante ou da rede como um todo, foi desenvolvida uma ferramenta em Java. Esta ferramenta foi nomeada de *TrafficEvaluator* e é baseada na ferramenta proposta em <https://corfu.pucrs.br/redmine/projects/atlas/wiki/Performance_Evaluation>.

A ferramenta gera os seguintes arquivos de saída: gráficos do tipo *Chaos Normal Form* (CNF) da latência e do tráfego aceito em função da carga oferecida, distribuição espacial da latência e do tráfego aceito para cada par de roteadores e um arquivo de texto contendo informações como latência mínima, média, máxima, desvio-padrão e do tráfego total aceito.

Mais informações e o download do repositório desta ferramenta podem ser encontrados em <https://github.com/LaNoC-UFC/traffic-eval>. É importante ressaltar que todas as ferramentas desenvolvidas nas Seções 4.5, 4.6, 4.7, 4.8 são *open-source* e estão disponíveis para uso, manutenção e atualizações por parte da comunidade.

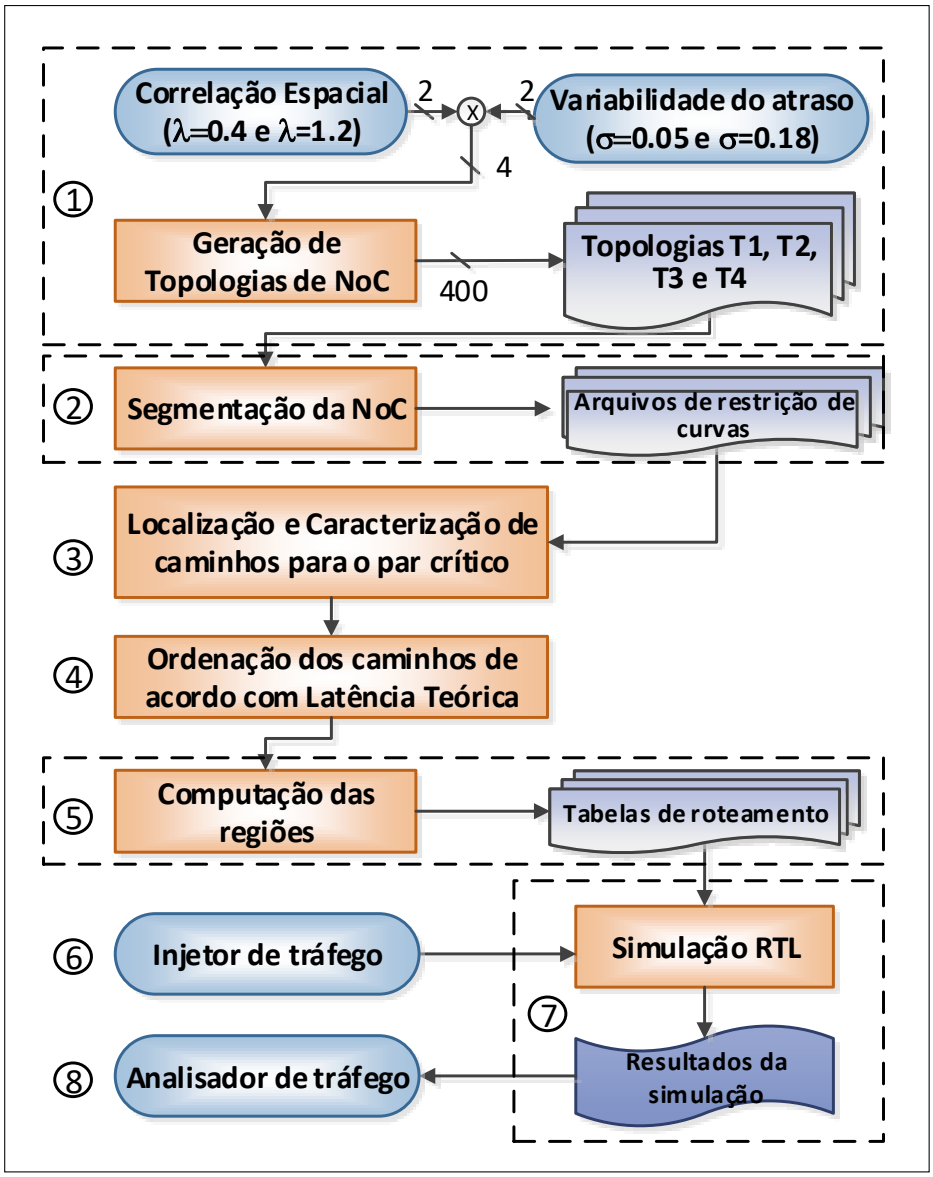
4.9 Configuração dos experimentos

Para validação da proposta desta dissertação, são utilizados dois experimentos: (i) o primeiro ordena de forma crescente os caminhos considerando a métrica *latência estimada* para cada caminho, a métrica *NewLinks* é obtida apenas para análise comparativa posterior; e (ii) o segundo, também ordena os caminhos de forma crescente considerando a métrica *latência estimada* para caminho, no entanto, os caminhos que tem a métrica *NewLinks* iguais a zero são eliminados e, portando, não utilizados. É importante destacar que toda a computação dos experimentos é realizada de forma *offline*, sendo as tabelas de roteamento carregadas nas simulações.

Na Figura 11 são apresentadas as configurações dos experimentos empregados incluindo a geração dos cenários de falhas, implementação do algoritmo de roteamento e mecanismo de implementação, geração das tabelas de roteamento e extração dos dados de

simulação para análise.

Figura 11 – Diagrama de blocos do experimento implementado para validação da técnica proposta neste trabalho.



Fonte: Próprio autor

A seguir é explicada cada uma das etapas na Figura 11, observando apenas a diferença na etapa (3), que distingue os experimentos.

4.9.1 Experimento 1

O Experimento 1 inicia a partir da geração de 100 cenários de falha (1) para cada uma das quatro distribuições apresentadas na Tabela 2, este cenários são obtidos através im-

plementação do modelo de falhas proposto por (HARGREAVES *et al.*, 2008) utilizando uma ferramenta de modelagem matemática (vide Seção 4.3). Em sequência, os 400 cenários gerados passam, um a um, pelos seguintes processos (etapa(2)): a partir do algoritmo de roteamento SBR (MEJIA *et al.*, 2006)(vide Seção 4.5) cada topologia de cada cenário de falha é dividida em sub-redes e então em segmentos. Posteriormente, são postas restrições de caminhos na rede com o intuito de evitar situações de *deadlock*. Na etapa (3), um par de comunicação crítico é escolhido aleatoriamente e então todos os possíveis caminhos (mínimos e não mínimos) são encontrados e, para cada caminho, é encontrado sua *latência estimada* (vide Seção 4.2.1) . Na etapa (4), os caminhos são ordenados de forma crescente de acordo com sua respectiva *latência estimada*. Na etapa (5), para C possíveis caminhos encontrados, são geradas C tabelas de roteamento a partir do mecanismo de implementação RBR (FLICH *et al.*, 2007) (vide Seção 4.6). Desta forma, a enésima tabela contém os primeiros C caminhos encontrados na etapa (3). É importante perceber que estas tabelas de roteamento representam diferentes conjuntos de caminhos, em que o árbitro de cada roteador terá maiores opções de escolha para envio dos pacotes a partir do momento em que o número de caminhos aumenta.

A partir de uma ferramenta desenvolvida pelo autor (vide Seção 4.7), na etapa (6) são gerados os arquivos de tráfego seguindo um padrão uniforme, onde todos os nós (com exceção do par crítico) se comunicam igualmente com todos os outros nós da rede, utilizando 1000 pacotes com 32 *flits* cada e uma taxa de injeção de 5%. Com o objetivo de avaliar a eficiência da técnica proposta utilizou-se uma taxa de injeção de 30% de comunicação para o par crítico. A etapa (7) realiza a simulação *Register-Transfer Level* (RTL) (utilizando o software ModelSim) para cada tabela de roteamento utilizando a NoC Phoenix. Por fim, na etapa (8), novamente utilizando uma ferramenta desenvolvida pelo autor (vide Seção 4.8), os arquivos de saída das simulações são analisados, obtendo assim os resultados que são discutidos no Capítulo 5.

4.9.2 Experimento 2

O segundo experimento utiliza a mesma configuração experimental do primeiro experimento, com exceção da etapa (3). Anteriormente eram computados todos os possíveis caminhos para o par crítico eleito aleatoriamente, porém, para este experimento, após encontrados todos os caminhos, são utilizados apenas os caminhos que possuem $NewLinks > 0$ (vide Seção 4.2.2). Todo o fluxo restante é mantido. Desta forma, o algoritmo proposto retira caminhos que não agregam novas conexões de comunicação, apenas tendem a saturar as conexões já

utilizadas anteriormente. Consequentemente, o algoritmo monta conjuntos de caminhos que sempre contribuem com novas conexões de comunicação, favorecendo melhor distribuição do tráfego. Os arquivos de saída das simulações e os resultados são analisados e discutidos no Capítulo 5.

5 RESULTADOS

Neste capítulo são apresentados e discutidos os resultados experimentais deste trabalho, em especial: (i) análise da abordagem de roteamento considerando múltiplos caminhos para diferentes distribuições de falhas, (ii) uso das métricas *latência téorica* e *NewLinks*, (iii) análise do impacto do roteamento considerando múltiplos caminhos no desempenho da rede.

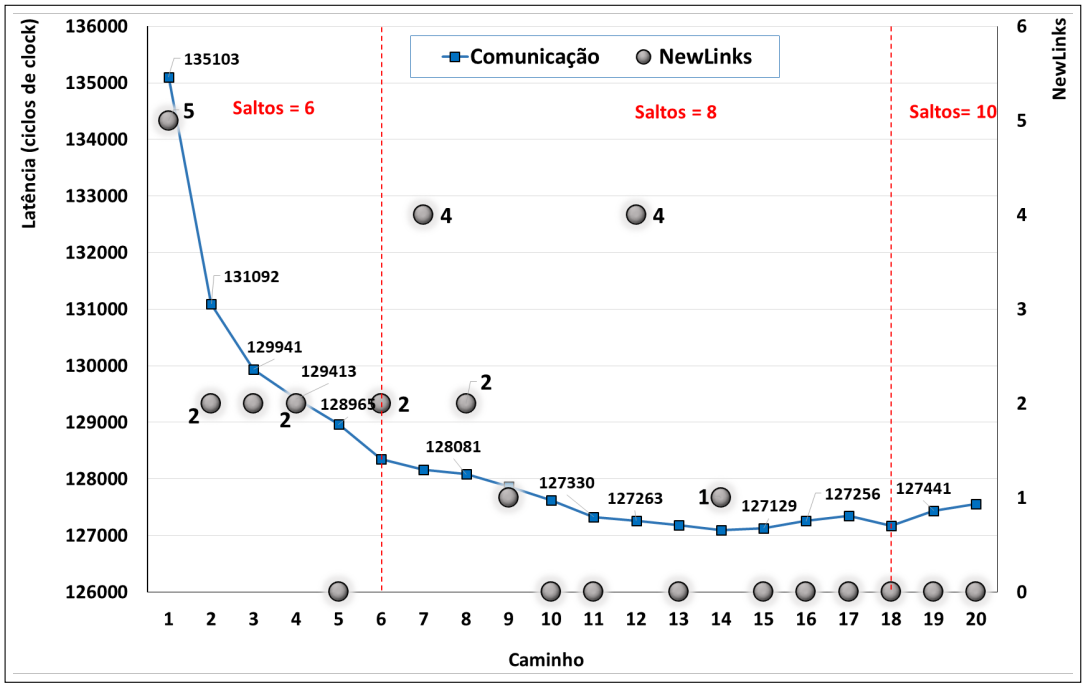
5.1 Resultados do Experimento 1

O objetivo deste experimento consiste na avaliação da latência média para um par de comunicação crítico escolhido aleatoriamente, esta escolha foi restringida a pares com distância topológica mínima de 5 saltos, de forma a evitar pares muito próximos e com poucas opções de caminhos. Em seguida, todos os possíveis caminhos de comunicação para o par crítico são avaliados, sendo estes previamente ordenados de acordo com a *latência estimada* e inseridos um a um de forma cumulativa.

Todos os gráficos subsequentes apresentam duas informações: latência média em função da quantidade de caminhos adicionados (eixo esquerdo, medido em ciclos de relógio) e o valor da métrica NewLinks para o último caminho inserido (eixo direito, medido em quantidade de conexões de comunicação). Além disso, são mostrados os respectivos comprimentos de cada caminho medido em saltos, em que um salto corresponde a uma conexão entre um determinado par de comunicação (categorizações em vermelho no gráfico).

Na Figura 12 são apresentados os resultados para um caso representativo da distribuição tipo 1, que representa uma tecnologia de fabricação de 65 *nm* com baixa variabilidade do atraso ($\sigma = 0.05$) e baixa força de correlação espacial ($\lambda = 1.2$). Nesta distribuição, os atrasos são menores e as falhas amplamente distribuídas pela topologia. É possível observar que, à medida que novos caminhos são inseridos, a latência média diminui de forma independente do comprimento do caminho, visto que o experimento se iniciou com caminhos de comprimento de 6 saltos e utilizou caminhos com comprimentos de até de 10 saltos. Todavia, esta diminuição é pequena uma vez que a disposição de falhas nesta distribuição é menos agressiva que as demais distribuições, resultando em poucas retransmissões. Além disto, a latência média apresentada na curva azul tende a decrescer à medida que são inseridos caminhos com $NewLinks > 0$ e estabiliza no momento em que são adicionados caminhos que não contribuem com novas conexões de comunicação ($NewLinks = 0$).

Figura 12 – Resultados experimentais para a distribuição tipo 1 (65 *nm* e baixa variabilidade do atraso).

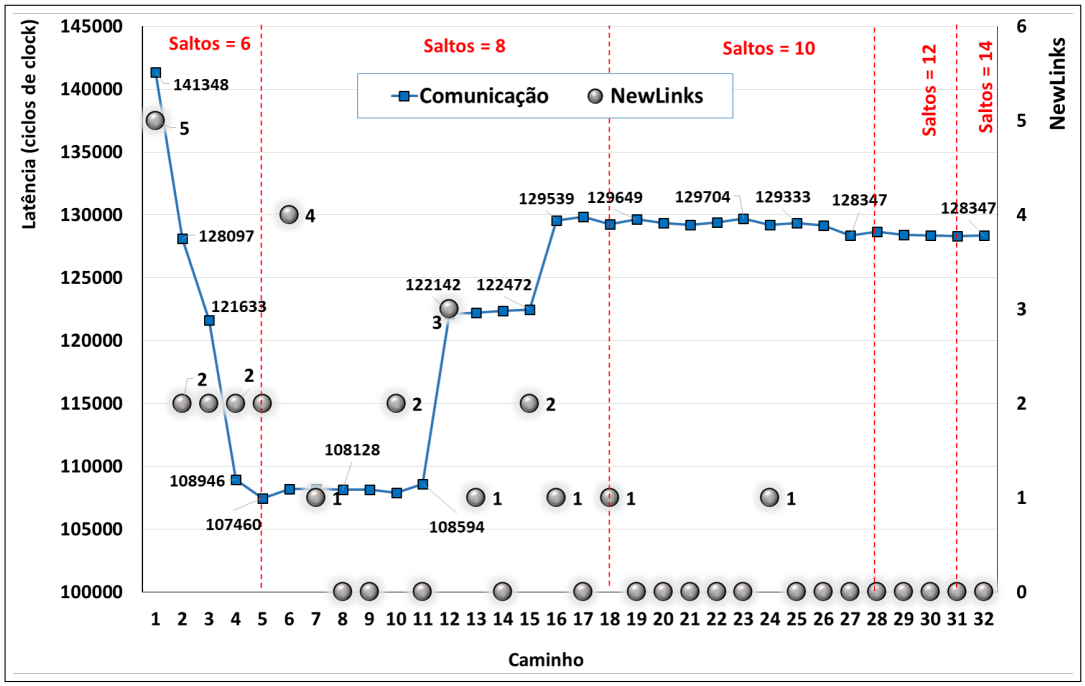


Fonte: Próprio autor

Na Figura 13 são mostrados os resultados experimentais para um caso representativo para a tecnologia de fabricação de 22 *nm*, com baixa variabilidade do atraso (σ = 0.05) e com alta força de correlação (λ = 0.4) (distribuição tipo 3). O experimento mostra que a técnica proposta tem alta eficiência e resulta em latências menores quando comparada com a solução que utiliza apenas um caminho mínimo. Dado que esta distribuição apresenta uma maior agressividade de falhas, a inserção de novos caminhos contribui para a diminuição de retransmissões e, por sua vez, menores latências. Adicionalmente, percebemos que mesmo utilizando caminhos com comprimento de mais de duas vezes maior que o comprimento do caminho mínimo. Contudo, é possível identificar que a latência volta a crescer a partir de 11 caminhos, porque os novos caminhos inseridos não contribuem para o aumento da diversidade de novas conexões, comprometendo o uso das conexões já presentes. Todavia, este aumento não chega a ser maior que a latência para o caminho mínimo.

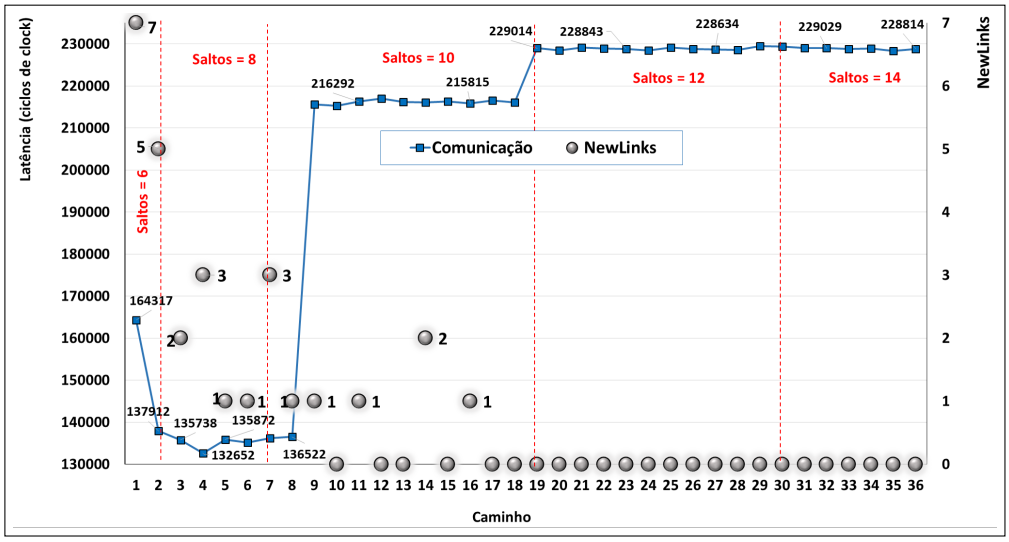
Os resultados experimentais da técnica proposta para as distribuições tipo 2 (65 *nm* e alta variabilidade do atraso) e tipo 4 (22 *nm* e alta variabilidade do atraso) são bastante similares. Nas figuras 14 e 15 são ilustrados estes resultados, tomando em consideração um caso representativo para cada distribuição.

Figura 13 – Resultados experimentais para a distribuição tipo 3 (22 nm e baixa variabilidade do atraso).



Fonte: Próprio autor

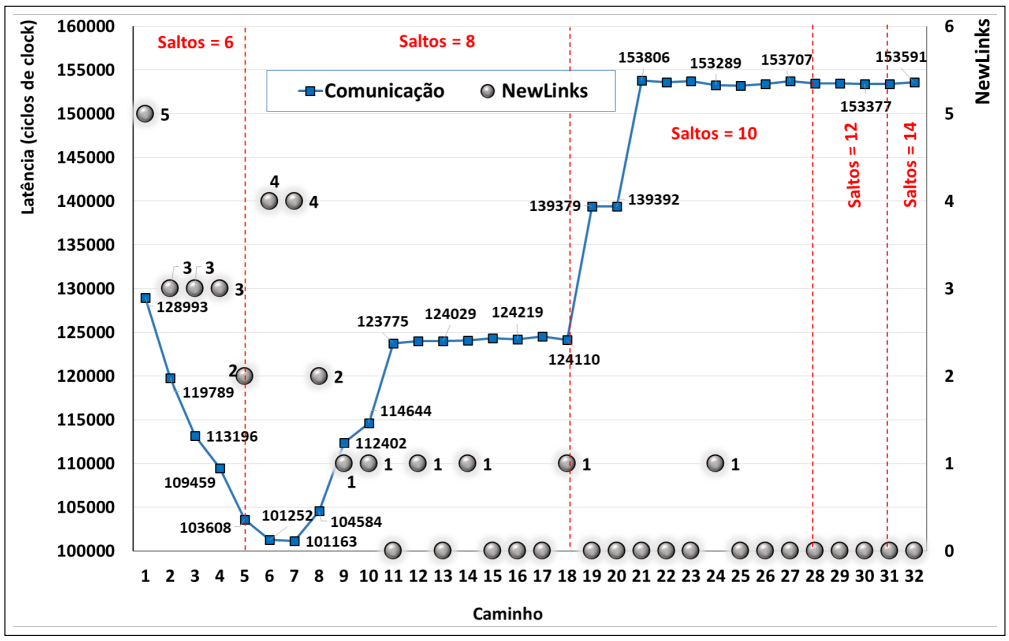
Figura 14 – Resultados experimentais para a distribuição tipo 2 (65 *nm* e alta variabilidade do atraso).



Fonte: Próprio autor

99

Figura 15 – Resultados experimentais para a distribuição tipo 4 (22 *nm* e alta variabilidade do atraso).



Fonte: Próprio autor

À medida que novos caminhos são inseridos, a latência média tende a decrescer, porém volta a crescer novamente quando os caminhos adicionados possuem um comprimento superior a 60% do comprimento do caminho mínimo. Isto acontece devido à alta variabilidade do atraso para ambas as distribuições, o que acarreta em falhas mais próximas e atrasos maiores, aliadas a caminhos muito longos. Além disso, quando há uma força da correlação espacial maior, os atrasos são ainda maiores e mais próximos, configurando uma distribuição mais agressiva de falhas (distribuição tipo 4, $\sigma = 0.18$ e $\lambda = 0.4$). Também se percebe na curva azul que a inserção de novos caminhos com *NewLinks* = 0 acarreta em um impacto negativo, resultando em um aumento da latência média, pois estes não contribuem com novas conexões, apenas sobrecarregam as conexões já em uso.

Os resultados relativos ao Experimento 1 demonstram que a abordagem baseada em múltiplos caminhos é capaz de reduzir a latência média para um determinado par de comunicação crítico. Isto é evidenciado ao inserirmos novos caminhos (mínimos ou não) com *NewLinks* > 0, visto que esta inserção contribui para uma maior distribuição do tráfego para o par crítico e, por consequência, resulta em menores latências.

Não obstante, a adição de alguns caminhos muito longos com baixos valores de *NewLinks*, não contribui para menores latências. Na realidade, esses caminhos acabam competindo

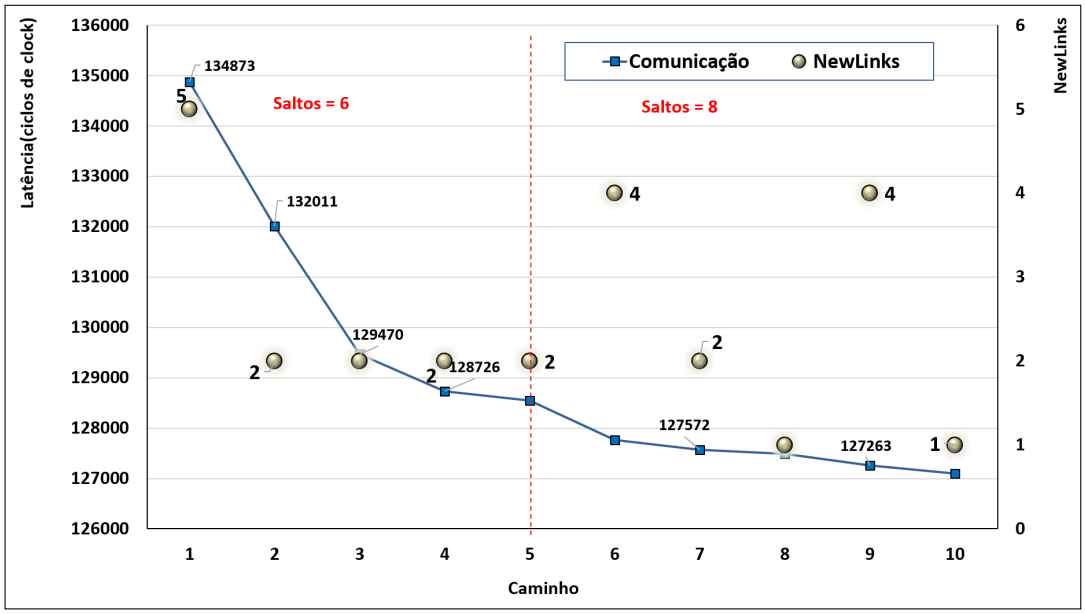
pelo uso de conexões já utilizadas por outros caminhos, reduzindo, assim, a eficiência da técnica proposta. Diante disto, é proposto o Experimento 2 (vide Seção 4.9.2), onde são considerados apenas caminhos com valores positivos e não nulos de *NewLinks* (e.g., caminhos com *NewLinks* = 0 são removidos).

5.2 Resultados do Experimento 2

O objetivo deste experimento é bastante semelhante ao Experimento 1 (vide Seção 4.9.1); todavia, apenas os caminhos com *NewLinks* > 0 são considerados. Para realizar este procedimento, após serem encontrados todos os possíveis caminhos entre o par crítico e serem ordenados de acordo com sua latência estimada, são removidos os caminhos que não contribuem com nenhuma nova conexão de comunicação. Desta forma, é esperado que o impacto negativo da técnica proposta seja minimizado.

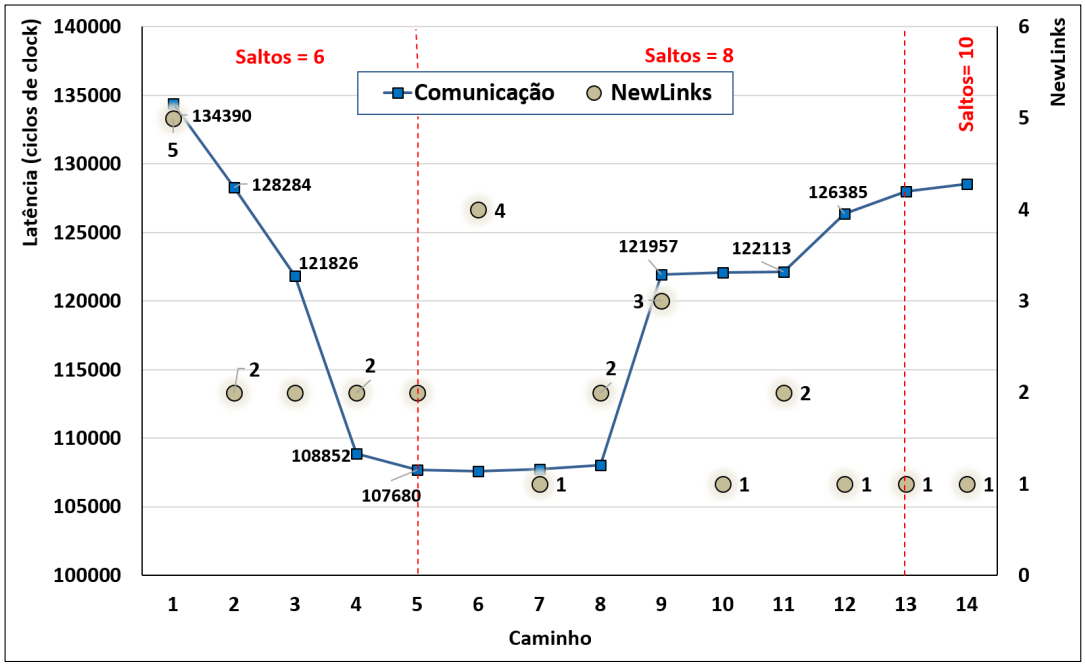
Todos os gráficos a seguir (Figuras 16 a 19) apresentam duas informações: latência média em função da quantidade de caminhos adicionados (eixo esquerdo, medido em ciclos de relógio) e o valor da métrica *NewLinks* para o último caminho inserido (eixo direito, medido em quantidade de conexões de comunicação). Além disso, são mostrados os respectivos comprimentos de cada caminho medido em saltos, em que um salto corresponde a uma conexão entre um determinado par de comunicação (categorizações em vermelho no gráfico).

Figura 16 – Resultados experimentais para a distribuição tipo 1 (65 *nm* e baixa variabilidade do atraso) apenas para caminhos com NewLinks > 0.



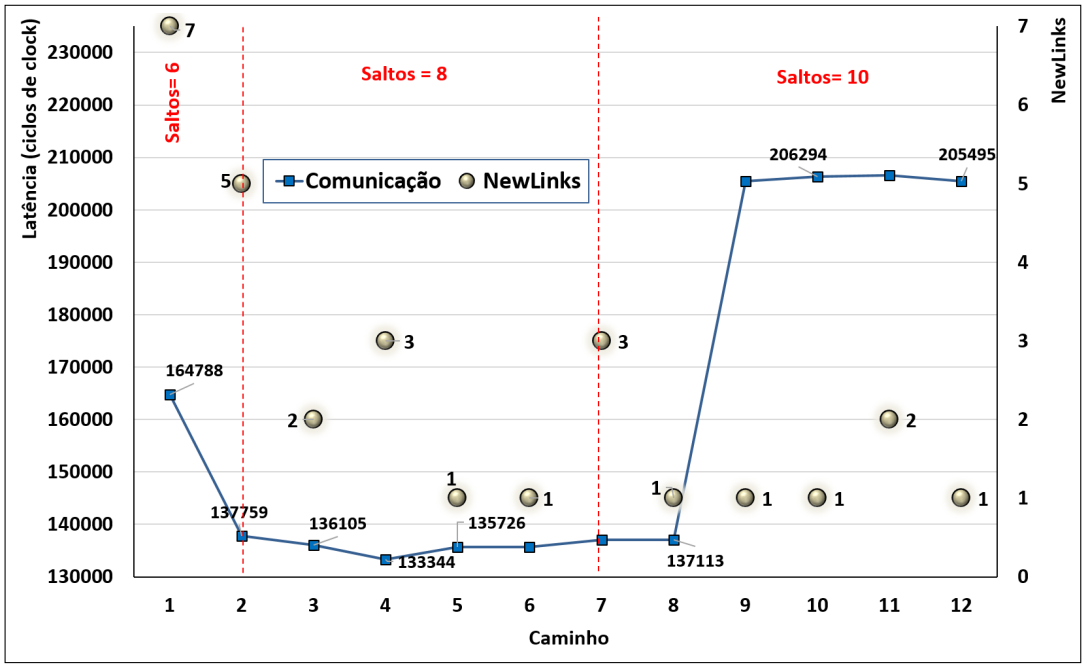
Fonte: Próprio autor

Figura 17 – Resultados experimentais para a distribuição tipo 3 (22 *nm* e baixa variabilidade do atraso) apenas para caminhos com NewLinks > 0.



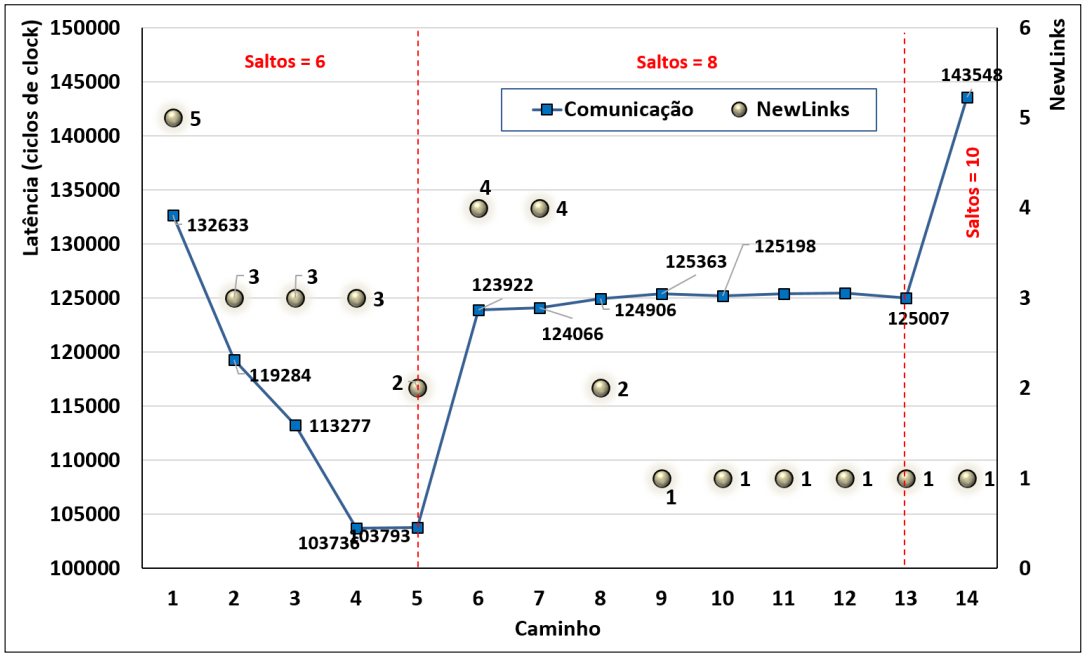
Fonte: Próprio autor

Figura 18 – Resultados experimentais para a distribuição tipo 2 (65 *nm* e alta variabilidade do atraso) apenas para caminhos com NewLinks > 0.



Fonte: Próprio autor

Figura 19 – Resultados experimentais para a distribuição tipo 4 (22 *nm* e alta variabilidade do atraso) apenas para caminhos com NewLinks > 0.



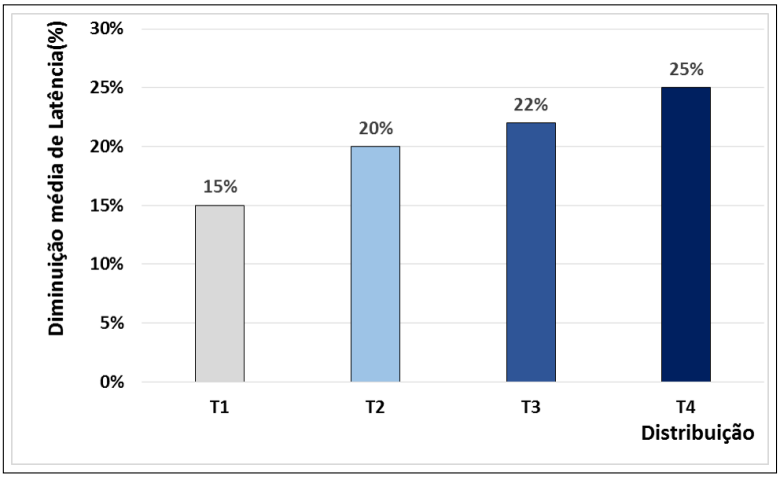
Fonte: Próprio autor

Os resultados relativos ao Experimento 2 demonstram que a remoção de caminhos

que não contribuem com novas conexões é capaz de melhorar ainda mais a proposta baseada em múltiplos caminhos, reduzindo assim a latência média para um determinado par de comunicação crítico. Todavia, as distribuições com alta variabilidade do atraso (tipo 2 e tipo 4) continuam apresentando momentos em que a inserção de caminhos muito longos (60% maiores que o caminho mínimo) gera aumento da latência média.

Na Figura 20 são apresentados a redução média de latência para cada distribuição utilizando apenas caminhos que contribuem com novas conexões. Os resultados experimentais mostram que a latência média tende a decrescer em maiores escalas à medida que as falhas se tornam mais presentes e mais agressivas (alta variabilidade do atraso, $\sigma = 0.18$). Fica evidente que a técnica proposta promove uma redução de latência, principalmente para processos de fabricação para tecnologias mais recentes (distribuições tipo 3 e 4, 22 *nm*).

Figura 20 – Redução média na latência média para cada distribuição.



Fonte: Próprio autor

No intuito de comparar os experimentos, nas figuras 21 e 22 são ilustrados os gráficos do tipo *boxplot*, para as distribuições utilizadas (tipo 1 a tipo 4) para os Experimentos 1 e 2.

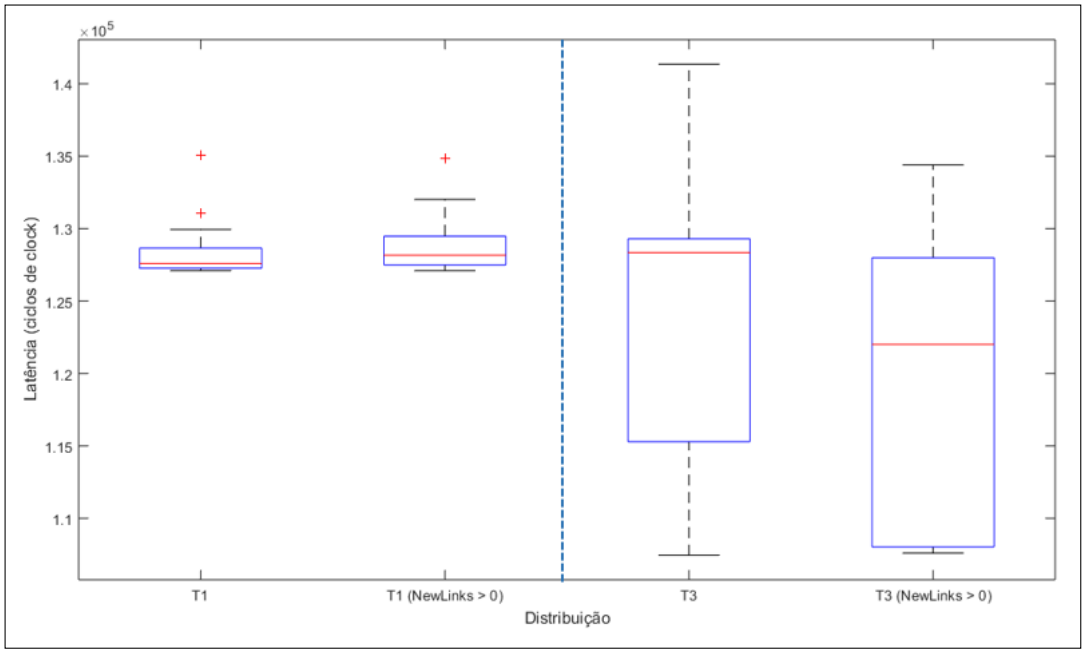
De forma geral, fica evidente a redução da latência das distribuições com *NewLinks* > 0, ressaltando a eficiência da técnica proposta. A exceção da distribuição tipo 1 (65 *nm* e baixa variabilidade do atraso), em que há uma baixa distribuição de falhas e pequenos atrasos.

Os *outliers* máximos correspondem aos valores para os primeiros caminhos mínimos utilizados, ou seja, antes da inclusão de novos caminhos, em que se espera uma maior latência. Já os *outliers* mínimos são referentes aos caminhos ótimos e representam os bons resultados da aplicação da técnica.

Na Figura 21, a distribuição tipo 3 (22 *nm* e baixa variabilidade do atraso) com *NewLinks* > 0 apresenta menor média de latência e grande parte dos dados se concentra abaixo da média de latência de T3 sem a utilização da técnica.

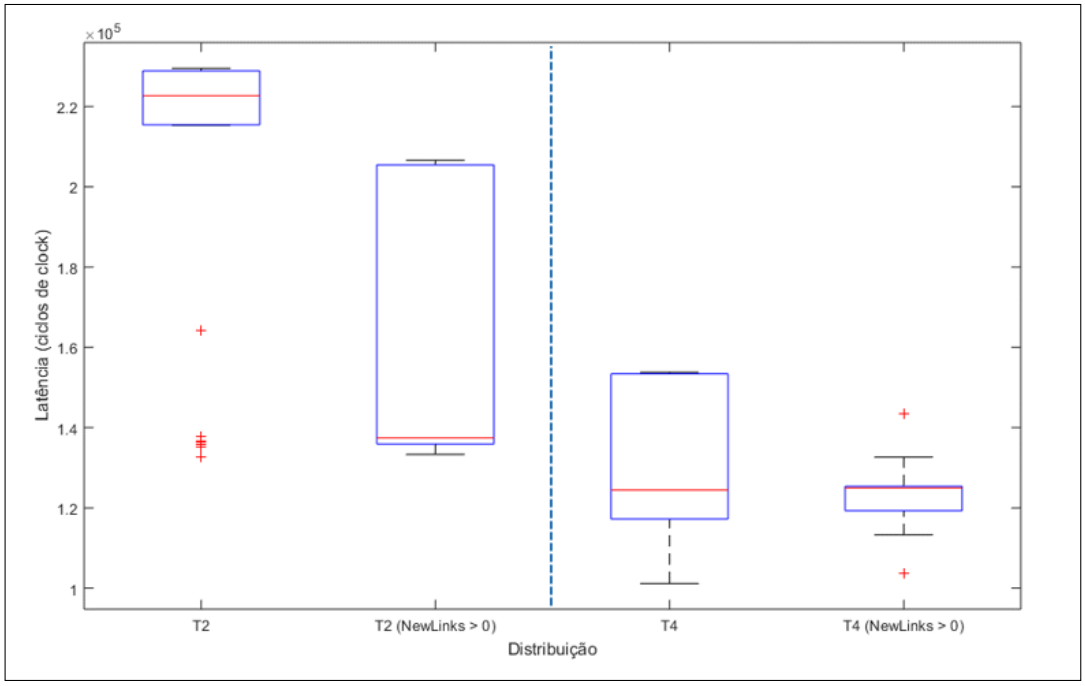
Na Figura 22, a distribuição tipo 2 (65 *nm* e alta variabilidade do atraso) com *NewLinks* > 0 apresenta resultados bastante expressivos; metade dos resultados encontrados está concentrado abaixo de $1,3 \times 10^5$ ciclos de relógio. Houve uma redução na média de latência de $8,5 \times 10^4$ ciclos de relógio. Já na distribuição tipo 4 (22 *nm* e alta variabilidade do atraso) com *NewLinks* > 0 há uma menor variação de resultados de latência, e uma ligeira redução da média de latência. Adicionalmente, metade dos resultados se encontra abaixo da média de latência para esta distribuição quando não é usada a técnica proposta.

Figura 21 – *Boxplot* para as distribuições T1 e T3 (ambas com menor variabilidade do atraso).



Fonte: Próprio autor

Figura 22 – *Boxplot* para as distribuições T2 e T4 (ambas com maior variabilidade do atraso).



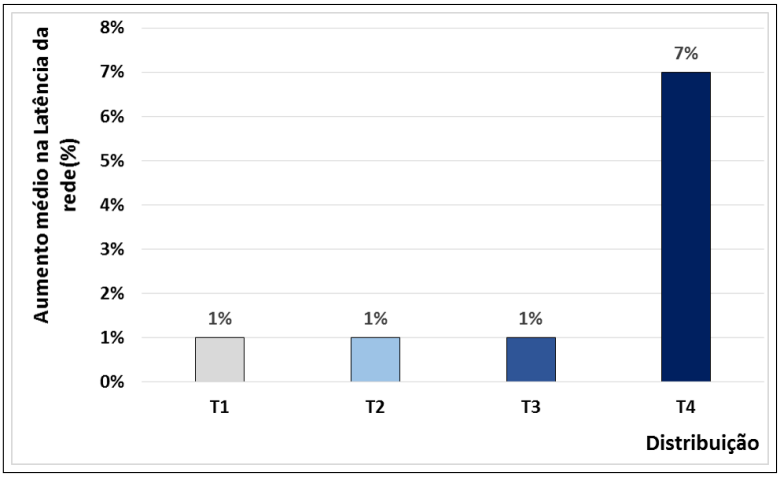
Fonte: Próprio autor

5.3 Impacto na rede

Os experimentos apresentados até o momento destacam apenas os resultados de latência média considerando o par crítico de comunicação; todavia, é interessante analisar o impacto da técnica na latência média considerando toda a rede. Para isto, os resultados foram avaliados considerando não somente o par crítico em questão, mas todos os nós comunicantes da rede.

Na Figura 23 é apresentado o impacto negativo médio (e.g., aumento de latência) da técnica proposta para cada distribuição. O pior impacto avaliado consistiu para a distribuição tipo 4, que representa uma distribuição mais agressiva de falhas com uma maior variabilidade do atraso e maior força de correlação espacial ($\lambda = 0.4$). No entanto, o impacto em cada distribuição pode ser negligenciado, de acordo com os benefícios advindos da técnica proposta.

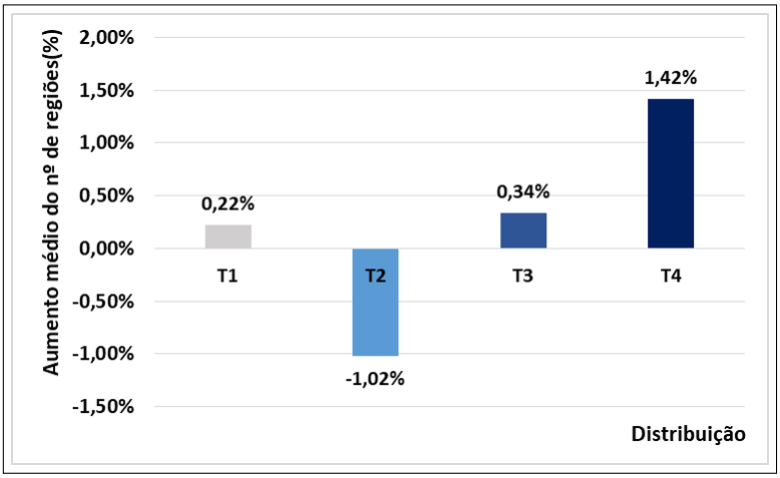
Figura 23 – Aumento da latência da rede com utilização da técnica proposta.



Fonte: Próprio autor

Dado que o roteamento é distribuído e implementado por meio de tabelas de roteamento, é necessário também avaliar o impacto da técnica no número de regiões necessárias pelo RBR (FLICH *et al.*, 2007), uma vez que o tamanho das tabelas reflete diretamente na área necessária. Na Figura 24 é mostrado o impacto médio no número de regiões para cada distribuição, destacando que abordagem. com base em múltiplos caminhos possui um baixo impacto, cerca de 1,5%, no pior dos casos. Também houve casos em que houve uma redução no número de regiões (distribuição tipo 2), isto é possível diante de uma maior quantidade de *merges* realizados a partir de uma maior diversidade de caminhos. Na figura 24 é ilustrado ainda que o pior caso ocorre para a distribuição que possui maior severidade de falhas e atrasos, afetando o algoritmo de formação das regiões.

Figura 24 – Aumento médio do número de regiões para cada distribuição.



Fonte: Próprio autor

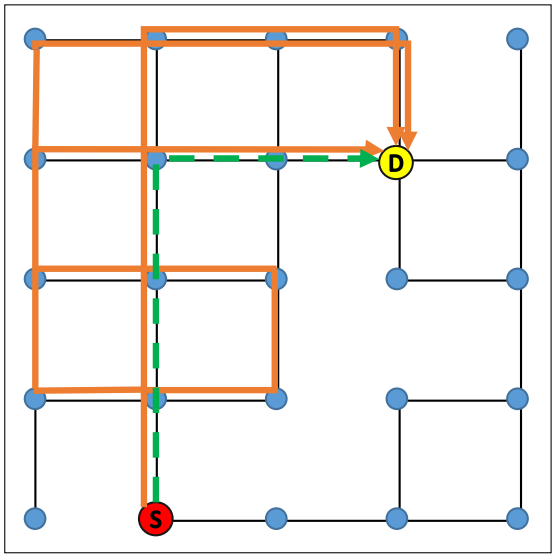
5.4 Discussão Final

Diante dos resultados experimentais apresentados, é possível averiguar que a técnica proposta baseada na utilização de caminhos mínimos e não mínimos possibilita uma redução na latência média, pois foram obtidas reduções significativas na latência média para o par crítico com baixos impactos na latência média da rede. Além disso, as métricas propostas demonstram ser eficazes para caracterizar e ordenar os caminhos para melhor distribuir o tráfego na rede, dado que se obtiveram reduções de latência de até 20% e 25% para tecnologias de fabricação de 65 *nm* e 22 *nm* respectivamente (vide Figura 20).

No entanto, as distribuições com alta variabilidade do atraso (tipo 2 e tipo 4) apresentaram casos em que a inclusão de caminhos com comprimento maior que 60% do comprimento do caminho mínimo não contribuem para diminuir a latência, indicando o momento em que é necessário parar de utilizar caminhos alternativos. Isto tem como causa o fato de que algumas topologias possuem poucos caminhos possíveis entre o par de comunicação e estes por sua vez contém muitos saltos.

Na Figura 25 é exemplificado o caso citado acima. Considerando o par de comunicação (S $\rightarrow$ D), em que a fonte é o nó em vermelho e o destino o nó em amarelo, em laranja estão identificados possíveis caminhos não mínimos e em verde tracejado está ilustrado um caminho mínimo. Embora os caminhos não mínimos possuam valores positivos de *NewLinks*, a latência aumentou devido ao seu comprimento ser muito maior que o caminho mínimo.

Figura 25 – Exemplo de roteamento para um caso da topologia de fabricação CMOS de 22 *nm*.



Fonte: Próprio autor

6 CONCLUSÕES

Com a tecnologia de fabricação dos transistores em escalas nanométricas, a probabilidade de diversos tipos de falhas aumenta cada vez mais. Desta forma, a demanda por estratégias de tolerância a falhas é um tema pertinente das pesquisas contemporâneas na área de microeletrônica. Em particular, nas redes intrachip, mecanismos de retransmissão têm sido amplamente utilizados devido à melhor relação custo-benefício quando comparados a outras soluções. Entretanto, a retransmissão de dados aumenta a latência de pacotes com erro, uma vez que é necessário identificar o erro, solicitar retransmissão e aguardar novamente a chegada do pacote. Este fato pode ser um problema para sistemas com requisitos temporais críticos. Desta forma, é necessário o desenvolvimento de alternativas capazes de minimizar a sobrecarga imposta, quase sempre, pelas estratégias de tolerância a falhas.

Neste trabalho foi proposta uma nova técnica capaz de reduzir a latência ocasionada pelos mecanismos de retransmissão a partir do uso de múltiplos caminhos (mínimos e não mínimos) para pares de comunicação críticos. Para isto, foram elencadas duas novas métricas: *latência estimada*, como forma de avaliar a tendência do comportamento da latência real de um caminho; *NewLinks* como forma de verificar como um caminho contribui para distribuir o tráfego para o par de comunicação em questão. De posse destas métricas, foi possível então caracterizar os diversos caminhos entre um par de comunicação, e, desta forma, utilizá-los dinamicamente como forma de aumentar a performance de comunicação para o par crítico (e.g., reduzir latência).

Os resultados experimentais mostram que a utilização da técnica proposta, para tecnologias de fabricação CMOS de 22 *nm* e 65 *nm*, é capaz de reduzir latência para pares críticos em até 25% e 20%, respectivamente. Adicionalmente, a técnica proposta teve um baixo impacto na rede como um todo, não comprometendo a latência média da rede e nem no número de regiões necessárias pelo mecanismo de implementação, o que por sua vez, implica em baixa sobrecarga de área.

Embora esta dissertação tenha alcançado resultados importantes a partir de uma sólida metodologia experimental, o tema central ainda pode ser explorado. Como principal trabalho futuro está a verificação da eficácia da técnica proposta sob circunstâncias de tráfego intenso sintético e real, uma vez que os experimentos aqui realizados consideraram somente um tráfego sintético do tipo uniforme. Além disso, destaca-se ainda a aplicação da solução para as emergentes NoCs 3D, em virtude ainda a maior diversidade de caminhos que pode ser encontrada. Também se considera importante avaliar a diminuição do consumo de energia a

partir da utilização da técnica proposta, uma vez que com uma menor taxa de retransmissão o consumo tende a ser menor. Finalmente, considera-se promissora a utilização da técnica proposta de forma dinâmica, ou seja, as tabelas sejam reconfiguradas de acordo com a ocorrência de falhas na rede.

REFERÊNCIAS

AGARWAL, A.; ISKANDER, C.; SHANKAR, R. Survey of network on chip (noc) architectures & contributions. **Journal of engineering, Computing and Architecture**, v. 3, n. 1, p. 21–27, 2009.

ALFARO, F.; BERMÚDEZ, A.; CASADO, R.; DUATO, J.; QUILES, F.; SÁNCHEZ, J. On the performance of up*/down* routing. **Communication, Architecture, and Applications Network-Based Parallel Computing**, Springer, p. 61–72, 2000.

BAHREBAR, P.; STROOBANDT, D. Adaptive and reconfigurable fault-tolerant routing method for 2d networks-on-chip. In: IEEE. **ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on**. [S.l.], 2014. p. 1–8.

BERTOZZI, D.; BENINI, L. Xpipes: A network-on-chip architecture for gigascale systems-on-chip. **IEEE circuits and systems magazine**, IEEE, v. 4, n. 2, p. 18–31, 2004.

BERTOZZI, D.; BENINI, L.; MICHELI, G. D. Low power error resilient encoding for on-chip data buses. In: IEEE COMPUTER SOCIETY. **Proceedings of the conference on Design, automation and test in Europe**. [S.l.], 2002. p. 102.

BERTOZZI, D.; BENINI, L.; MICHELI, G. D. Error control schemes for on-chip communication links: the energy-reliability tradeoff. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 24, n. 6, p. 818–831, 2005.

BORKAR, S. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. **IEEE Micro**, IEEE, v. 25, n. 6, p. 10–16, 2005.

CHANG, Y.-C.; CHIU, C.-T.; LIN, S.-Y.; LIU, C.-K. On the design and analysis of fault tolerant noc architecture using spare routers. In: IEEE PRESS. **Proceedings of the 16th Asia and South Pacific Design Automation Conference**. [S.l.], 2011. p. 431–436.

CHARIF, A.; ZERGAINOH, N.-E.; NICOLAIDIS, M. Addressing transient routing errors in fault-tolerant networks-on-chips. In: IEEE. **Test Symposium (ETS), 2016 21th IEEE European**. [S.l.], 2016. p. 1–6.

CONSTANTINESCU, C. Trends and challenges in vlsi circuit reliability. **IEEE Micro**, IEEE, v. 23, n. 4, p. 14–19, 2003.

DALLY, W. J.; SEITZ, C. L. The torus routing chip. **Distributed computing**, Springer, v. 1, n. 4, p. 187–196, 1986.

DALLY, W. J.; TOWLES, B. P. **Principles and practices of interconnection networks**. [S.l.]: Elsevier, 2004.

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

EBRAHIMI, M.; DANESHTALAB, M.; PLOSILA, J.; TENHUNEN, H. Minimal-path fault-tolerant approach using connection-retaining structure in networks-on-chip. In: IEEE. **Networks on Chip (NoCS), 2013 Seventh IEEE/ACM International Symposium on**. [S.l.], 2013. p. 1–8.

FENG, C.; LU, Z.; JANTSCH, A.; ZHANG, M.; XING, Z. Addressing transient and permanent faults in noc with efficient fault-tolerant deflection router. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, IEEE, v. 21, n. 6, p. 1053–1066, 2013.

FICK, D.; DEORIO, A.; HU, J.; BERTACCO, V.; BLAAUW, D.; SYLVESTER, D. Vicis: A reliable network for unreliable silicon. In: ACM. **Proceedings of the 46th Annual Design Automation Conference**. [S.l.], 2009. p. 812–817.

FLICH, J.; MEJIA, A.; LOPEZ, P.; DUATO, J. Region-based routing: An efficient routing mechanism to tackle unreliable hardware in network on chips. In: IEEE. **Networks-on-Chip, 2007. NOCS 2007. First International Symposium on**. [S.l.], 2007. p. 183–194.

GANGULY, A.; PANDE, P. P.; BELZER, B. Crosstalk-aware channel coding schemes for energy efficient and reliable noc interconnects. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, IEEE, v. 17, n. 11, p. 1626–1639, 2009.

GOMEZ, M. E.; DUATO, J.; FLICH, J.; LOPEZ, P.; ROBLES, A.; NORDBOTTEN, N. A.; LYSNE, O.; SKEIE, T. An efficient fault-tolerant routing methodology for meshes and tori. **IEEE Computer Architecture Letters**, IEEE, v. 3, n. 1, p. 3–3, 2004.

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

HAMMING, R. W. Error detecting and error correcting codes. **Bell Labs Technical Journal**, Wiley Online Library, v. 29, n. 2, p. 147–160, 1950.

HARGREAVES, B.; HULT, H.; REDA, S. Within-die process variations: How accurately can they be statistically modeled? In: IEEE. **Design Automation Conference, 2008. ASPDAC 2008. Asia and South Pacific**. [S.l.], 2008. p. 524–530.

HO, C.-T.; STOCKMEYER, L. A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers. **IEEE Transactions on Computers**, IEEE, v. 53, n. 4, p. 427–438, 2004.

HOEFFLINGER, B. Itrs: The international technology roadmap for semiconductors. In: **Chips 2020**. [S.l.]: Springer, 2011. p. 161–174.

KERMANI, P.; KLEINROCK, L. Virtual cut-through: A new computer communication switching technique. **Computer Networks (1976)**, Elsevier, v. 3, n. 4, p. 267–286, 1979.

KIA, H. S.; ABABEI, C. A new fault-tolerant and congestion-aware adaptive routing algorithm for regular networks-on-chip. In: IEEE. **Evolutionary Computation (CEC), 2011 IEEE Congress on**. [S.l.], 2011. p. 2465–2472.

KIM, J.; PARK, D.; NICOPOULOS, C.; VIJAYKRISHNAN, N.; DAS, C. R. Design and analysis of an noc architecture from performance, reliability and energy perspective. In: ACM. **Proceedings of the 2005 ACM symposium on Architecture for networking and communications systems**. [S.l.], 2005. p. 173–182.

KUMAR, A.; MANJUNATH, D.; KURI, J. **Communication networking: an analytical approach**. [S.l.]: Elsevier, 2004.

KUMAR, M.; LAXMI, V.; GAUR, M. S.; DANESHTALAB, M.; KO, S.-B.; ZWOLINSKI, M. *et al.* A novel non-minimal/minimal turn model for highly adaptive routing in 2d nocs. In: IEEE. **Networks-on-Chip (NoCS), 2014 Eighth IEEE/ACM International Symposium on**. [S.l.], 2014. p. 184–185.

LEHTONEN, T.; LILJEBERG, P.; PLOSILA, J. Online reconfigurable self-timed links for fault tolerant noc. **VLSI design**, Hindawi Publishing Corporation, v. 2007, 2007.

LI, M.; ZENG, Q.-A.; JONE, W.-B. Dyxy: a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In: ACM. **Proceedings of the 43rd Annual Design Automation Conference**. [S.l.], 2006. p. 849–852.

MARCON, C.; AMORY, A.; WEBBER, T.; VOLPATO, T.; POEHLS, L. B. Phoenix noc: A distributed fault tolerant architecture. In: IEEE. **Computer Design (ICCD), 2013 IEEE 31st International Conference on**. [S.l.], 2013. p. 7–12.

MARCULESCU, R.; OGRAS, U. Y.; PEH, L.-S.; JERGER, N. E.; HOSKOTE, Y. Outstanding research problems in noc design: system, microarchitecture, and circuit perspectives. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 28, n. 1, p. 3–21, 2009.

MEJIA, A.; FLICH, J.; DUATO, J. On the potentials of segment-based routing for nocs. In: IEEE. **Parallel Processing, 2008. ICPP'08. 37th International Conference on**. [S.l.], 2008. p. 594–603.

MEJIA, A.; FLICH, J.; DUATO, J.; REINEMO, S.-A.; SKEIE, T. Segment-based routing: An efficient fault-tolerant routing algorithm for meshes and tori. In: IEEE. **Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International**. [S.l.], 2006. p. 10–pp.

MEJIA, A.; PALESI, M.; FLICH, J.; KUMAR, S.; LÓPEZ, P.; HOLSMARK, R.; DUATO, J. Region-based routing: a mechanism to support efficient routing algorithms in nocs. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, IEEE, v. 17, n. 3, p. 356–369, 2009.

MURALI, S.; THEOCHARIDES, T.; VIJAYKRISHNAN, N.; IRWIN, M. J.; BENINI, L.; MICHELI, G. D. Analysis of error recovery schemes for networks on chips. **IEEE Design & Test of Computers**, IEEE, v. 22, n. 5, p. 434–442, 2005.

PARK, D.; NICOPOULOS, C.; KIM, J.; VIJAYKRISHNAN, N.; DAS, C. R. Exploring fault-tolerant network-on-chip architectures. In: IEEE. **Dependable Systems and Networks, 2006. DSN 2006. International Conference on**. [S.l.], 2006. p. 93–104.

PASRICHA, S.; DUTT, N. **On-chip communication architectures: system on chip interconnect**. [S.l.]: Morgan Kaufmann, 2010.

PASRICHA, S.; ZOU, Y. Ns-ftr: A fault tolerant routing scheme for networks on chip with permanent and runtime intermittent faults. **Proceedings of the 16th Asia and South Pacific Design Automation Conference**, p. 443–448, 2011.

PEH, L.-S.; JERGER, N. E. **On-chip networks**. [S.l.]: Morgan & Claypool Publishers, 2009.

RADETZKI, M.; FENG, C.; ZHAO, X.; JANTSCH, A. Methods for fault tolerance in networks-on-chip. **ACM Computing Surveys (CSUR)**, ACM, v. 46, n. 1, p. 8, 2013.

SANCHO, J. C.; ROBLES, A.; DUATO, J. A flexible routing scheme for networks of workstations. **Lecture notes in computer science**, Springer, p. 260–267, 2000.

SHINTANI, M.; UEZONO, T.; TAKAHASHI, T.; HATAYAMA, K.; AIKYO, T.; MASU, K.; SATO, T. A variability-aware adaptive test flow for test quality improvement. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 33, n. 7, p. 1056–1066, 2014.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G.; SILBERSCHATZ, A. **Operating system concepts**. [S.l.]: Addison-wesley Reading, 1998. v. 4.

SILVEIRA, J.; CORTEZ, P.; CADORE, A.; MOTA, R.; MARCON, C.; BRAHM, L.; FERNANDES, R. Smart reconfiguration approach for fault-tolerant noc based mpsocs. In: IEEE. **Integrated Circuits and Systems Design (SBCCI), 2015 28th Symposium on**. [S.l.], 2015. p. 1–6.

STRANO, A.; BERTOZZI, D.; TRIVINO, F.; SÁNCHEZ, J. L.; ALFARO, F. J.; FLICH, J. Osr-lite: Fast and deadlock-free noc reconfiguration framework. In: IEEE. **Embedded Computer Systems (SAMOS), 2012 International Conference on**. [S.l.], 2012. p. 86–95.

TRIVIÑO, F.; SÁNCHEZ, J. L.; ALFARO, F. J.; FLICH, J. Network-on-chip virtualization in chip-multiprocessor systems. **Journal of Systems Architecture**, Elsevier, v. 58, n. 3, p. 126–139, 2012.

VITKOVSKIY, A.; SOTERIOU, V.; NICOPOULOS, C. Dynamic fault-tolerant routing algorithm for networks-on-chip based on localised detouring paths. **IET Computers & Digital Techniques**, IET, v. 7, n. 2, p. 93–103, 2013.

WANG, J.; HUANG, L.; LI, Q.; LI, G.; JANTSCH, A. Optimizing the location of ecc protection in network-on-chip. In: ACM. **Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis**. [S.l.], 2016. p. 19.

YIN, J.; ZHOU, P.; HOLEY, A.; SAPATNEKAR, S. S.; ZHAI, A. Energy-efficient non-minimal path on-chip interconnection network for heterogeneous systems. In: ACM. **Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design**. [S.l.], 2012. p. 57–62.

YU, Q.; AMPADU, P. Adaptive error control for noc switch-to-switch links in a variable noise environment. In: IEEE. **IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems (DFTVS)**. [S.l.], 2008. p. 352–360.

YU, Q.; AMPADU, P. A dual-layer method for transient and permanent error co-management in noc links. **IEEE Transactions on Circuits and Systems II: Express Briefs**, IEEE, v. 58, n. 1, p. 36–40, 2011.

YU, Q.; ZHANG, M.; AMPADU, P. Exploiting inherent information redundancy to manage transient errors in noc routing arbitration. In: IEEE. **Networks on Chip (NoCS), 2011 Fifth IEEE/ACM International Symposium on**. [S.l.], 2011. p. 105–112.

ZAMZAM, D. M.; GHANY, M. A. A. E.; HOFMANN, K.; ISMAIL, M. Highly reliable and power efficient noc interconnects. In: IEEE. **NORCHIP**. [S.l.], 2011. p. 1–4.

ZHANG, Y.; WU, N.; YU, P. W.; GE, F.; ZHOU, F. Fault-tolerant schemes for noc with a network monitor. In: IEEE. **International Symposium on Communications and Information Technologies (ISCIT)**. [S.l.], 2010. p. 1083–1086.