UNIVERSIDADE DO VALE DO ITAJAÍ
CENTRO DE CIÊNCIAS TECNOLÓGICAS DA TERRA E DO MAR
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ANÁLISE E IMPLEMENTAÇÃO DA CIFRA EM BLOCOS SIMON PARA TRATAR CONFIDENCIALIDADE NA REDE-EM-CHIP SOCIN

por

Antonio Frederico Mellies Neto

Itajaí (SC), novembro de 2017

UNIVERSIDADE DO VALE DO ITAJAÍ
CENTRO DE CIÊNCIAS TECNOLÓGICAS DA TERRA E DO MAR
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ANÁLISE E IMPLEMENTAÇÃO DA CIFRA EM BLOCOS SIMON PARA TRATAR CONFIDENCIALIDADE NA REDE-EM-CHIP SOCIN

Áreas de Redes de Computadores e Segurança de Informações

por

Antonio Frederico Mellies Neto

Relatório apresentado à Banca Examinadora do Trabalho Técnico-científico de Conclusão de Curso do curso de Ciência da Computação para análise e aprovação.
Orientador(a): Fabricio Bortoluzzi, Mestre
Coorientador(a): Cesar Albenes Zeferino, Doutor

Itajaí (SC), novembro de 2017

Dedico esse trabalho aos meus pais Silvio Mellies e Rosemeri Aparecida de Mello meus eternos professores.

AGRADECIMENTOS

Aos meus pais Silvio Mellies e Rosemeri Aparecida de Mello, por todo amor e apoio durante minha caminhada.

A todo os meus colegas de classe, em especial, Pablo Fagundes Wachsmann e Yan Alexander Venera, por todas as risadas compartilhadas.

Aos professores do curso, por todo o conhecimento transmitido, em especial ao professor Eduardo Alves da Silva pela dedicação e o auxílio para comigo.

Ao meu orientador Fabricio Bortoluzzi, por toda a dedicação e comprometimento com este trabalho.

Ao meu co-orientador Cesar Albenes Zeferino, pela confiança em mim empregada.

Não é preciso ter olhos abertos para ver o sol, nem é preciso ter ouvidos afiados para ouvir o trovão. Para ser vitorioso você precisa ver o que não está visível.
Sun Tzu

RESUMO

NETO, Antonio Frederico Mellies. Análise e implementação da cifra em blocos SIMON para tratar confidencialidade na rede-em-chip SoCIN. Itajaí, 2017. 61 f. Trabalho Técnico-científico de Conclusão de Curso (Graduação em Ciência da Computação) – Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí, Itajaí, 2017.

Com o aumento no número de núcleos na composição dos processadores, possibilitado pela diminuição dos canais de comunicação entre os transistores, se fez necessário o desenvolvimento de uma solução mais eficiente para prover elevadas taxas de comunicação entre os componentes. A solução natural para sanar esse problema encontra-se nas redes-em-chip. Essas funcionam analogamente as redes tradicionais de computadores paralelos. Nesse contexto, observa-se nas redes-em-chip vulnerabilidades similares às encontradas nas redes tradicionais. Considerando a existência dessas vulnerabilidades, esse trabalho apresenta uma entre as diversas tratativas possíveis para prover confidencialidade. Para alcançar o objetivo proposto, foi realizada uma implementação do algoritmo de criptografia em bloco SIMON nas comunicações da rede-em-chip SoCIN. Para escolha do algoritmo, foram consideradas algumas características, sendo, a principal delas, estar contido na classe de algoritmos de criptografia leve. Essa característica é considerada de maior relevância tendo em vista o ambiente limitado onde o trabalho se desenvolve. Dessa forma, tornando as mensagens que trafegam na rede "incompreensíveis" para qualquer componente que não esteja envolvido na comunicação. Como resultado foi observado um aumento na latência da rede e a necessidade de mais ciclos de *clock* para conclusão de cada troca de mensagens, sendo esses impactos justificados pela segurança provida pelo SIMON.

Palavras-chave: Rede-em-chip. Criptografia. Confidencialidade.

ABSTRACT

With the increase in the number of cores in the processors composition, made possible by the reduction of the communication channels between the transistors, it was necessary to develop a more efficient solution to provide high communication rates between the components. The natural solution to address this problem lies in networking on-chip. These work similarly to traditional parallel-computer networks. In this context, it is observed in the networks-in-chip vulnerabilities similar to those found in traditional networks. Considering the existence of these vulnerabilities, this work will present one of several possible deals to provide confidentiality. In order to reach the proposed goal, an implementation of the SIMON block encryption algorithm will be performed in the SoCIN network-on-chip communications. To choose the algorithm, some characteristics were considered, being the main one to be contained in the class of algorithms of light cryptography. This feature is considered of greater relevance in view of the limited environment where the work develops. In this way, making the messages that travel in the network "incomprehensible" for any component that is not involved in the communication. At the end, a comparison of impact on the performance of the network-on-chip will be realized, considering the states of the network with and without the SIMON deployment. As a result, an increase in network latency and the need for more clock cycles to complete each message exchange were observed. justified by the security provided by SIMON.

Keywords: Networks-in-chip. Encryption. Confidentiality.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AES	Advanced Encryption Standar
ASIC	Application Specific Integrated Circuit
CPU	Central Processing Unit
DES	Data Encryption Standard
DiSP	Dos Probe
ENIAC	Eletronic Numerical Integrator and Computer
Flit	Flow control unit
FPGA	Field Programmable Gate Array
IAP	Illegal Access Probe
IP	Intellectual Property blocks
IR	Interface de rede
MPSoCs	Multiprocessor System-on-Chip
NI	Network Interface
NoC	Networks-on-Chip
NSA	National Security Agency
NSM	Network Security Manager
NUMA	Non-Uniform Memory Access).
OCNs	On-Chip Networks
Phit	Physical unit
RFID	Radio-Frequency Identification
RTL	Register Transfer Level
SoC	System-on-Chip
SoCIN	System-on-Chip Interconnection Network
TTC	Trabalho Técnico-científico de Conclusão de Curso
UMA	Uniform Memory Access
UNIVALI	Universidade do Vale do Itajaí
VHDL	VHSIC Hardware Description Language
VHSIC	Very-High-Speed Integrated Circuit

LISTA DE SÍMBOLOS

&	AND
$\oplus$	XOR
b	Bloco de texto
k	Chave
L	Left
m	Subchaves de rodada
M	Texto claro (mensagem)
n	Número de blocos
R	Right
S	Descolamento circular de bits
T	Rodadas
ns	Nanossegundos

SUMÁRIO

1 INTRODUÇÃO

O crescente consumo de energia e a limitação de desempenho das arquiteturas de processadores de um único núcleo levaram ao advento dos processadores *multicore*, isto é, de múltiplos núcleos de execução de programas de computador (JERGER; PEH, 2009).

De outro lado, a diminuição das dimensões e da distância entre transistores para poucos nanômetros, tornou-se viável a integração de sistemas computacionais completos sintetizados em uma única pastilha de silício, sistemas esses denominados sistemas integrados ou SoCs (Systems-on-Chip). Esses baseiam-se no reuso dos blocos que os compõe, nos quais são previamente projetados e verificados, e comumente conhecidos como blocos de propriedade intelectual - IP (*Intellectual Property blocks*) ou núcleos (*cores*) (MARTIN; CHANG, 2003). A necessidade por desempenho crescente causa a criação de processadores com números cada vez mais elevados de núcleos, permitindo que um único chip possua milhares desses interligados. Essa integração unificada no chip, denominada MPSoC (Multiprocessor System-on-Chip), permite, por conta de sua arquitetura, a adição de componentes que visam melhorar o desempenho, podendo esses componentes serem: núcleos de processadores, memória embarcada e processadores dedicados a trabalhar com vídeo, áudio, comunicação, entre outros (BARON; 2013).

Os núcleos de um sistema integrado são interconectados por meio de uma estrutura de canais denominada de arquitetura de comunicação (ZEFERINO, 2003). Essa comunicação pode ser realizada de duas formas distintas: ponto a ponto dedicados, que requerem um canal exclusivo ao custo de não suportarem reusabilidade e que oferecem desempenho superior; e multiponto compartilhado, ou barramento, que permite reaproveitamento no sistema.

A abordagem de barramento é a solução tipicamente utilizada para interconectar os núcleos de um SoC (System-on-chip) com poucos núcleos por conta de sua característica reutilizável. Torna-se inviável a utilização de comunicação por barramento em sistemas que possuem perspectivas de escala por acréscimo de núcleos. Problemas como aumento no consumo de energia devido à sobrecarga capacitiva dos canais e aumento no tempo de propagação, que já se fazem presente nas comunicações entre núcleos, se tornam recorrentes com o aumento de núcleos do processador (ZEFERINO, 2003).

Nesse contexto, se fez necessário o desenvolvimento de uma solução que consiga sanar os problemas que surgem com o aumento no número de núcleos nos SoCs. Em virtude da relação direta entre comunicação entre tarefas e desempenho da aplicação, é imprescindível que

a infraestrutura de comunicação empregada suporte elevadas taxas de comunicação e alto grau de paralelismo. A solução natural para tal problema está nas redes de interconexão intra-chip (NoCs – Networks-on-Chip), as quais além de suprirem as necessidades relativas à largura de banda e paralelismo, proporcionam a escalabilidade necessária para o crescimento dos sistemas (CARARA, 2011).

A tecnologia de redes-em-chip (NoC) é geralmente associada a projetos de sistemas-em-chip (SoC) robustos, que necessitam de alto desempenho e que combinam seus diversos núcleos para realizar tratativas em questões que envolvem gráficos, áudio e vídeo. Nesses projetos, a NoC fornece caminhos de alta largura de banda para o tráfego de dados, garantindo alto desempenho e evitando o congestionamento. Conforme as redes tradicionais, as NoCs convertem os dados em inúmeros pacotes que são priorizados arbitrariamente pelo roteador e transmitidos através da rede até seu destino. Essas distribuições de dados são projetadas conforme a necessidade do SoC, podendo ser configuradas como um anel, árvore, formato multi camadas ou combinações entre elas (GWENNAP, 2015).

Seja inserido em um smartphone, em uma televisão, em um videogame ou mesmo em eletrodomésticos, atualmente, os SoCs estão presentes no cotidiano da maioria das pessoas. (METZGER, 2014). Sendo assim, dados são criados, processados e transmitidos o tempo todo, considerando que a grande maioria desses dispositivos possui algum tipo de conectividade com outros dispositivos externos, uma preocupação que se faz presente é no aspecto de segurança.

Um sistema seguro se caracteriza por ser capaz de manter as propriedades de segurança, propriedades essas descritas por Landwehr (2001) como: confidencialidade, integridade, disponibilidade, autenticação e não-repúdio. A aplicação de métodos de segurança em um sistema tem como objetivo evitar ataques, pelo qual é definido como uma tentativa de destruir, expor, alterar, incapacitar, roubar ou ganhar acesso não autorizado ou fazer uso não autorizado de um recurso (ISO/IEC 27000:2009, 2009).

Do mesmo modo, que se emprega segurança em redes tradicionais para proteger as informações trafegadas entre os pontos dessa rede, faz-se necessário proteger o tráfego entre os núcleos de uma rede-em-chip. Nesta linha de pesquisa, destacam-se os seguintes trabalhos:

a) Baron (2013), que avaliou mecanismos para proteger a rede-em-chip SoCIN contra ataques de negação de serviços.

b) Metzger (2014), que realizou uma modelagem experimental de diversas formas de ataque as redes-em-chip com ênfase na SoCIN.

c) Sopran (2016), que avaliou o desempenho de diversos algoritmos de segurança quanto a sua aplicabilidade em hardware e software, apontando o algoritmo SIMON como viável para o primeiro caso.

d) Silva (2015), que implementou um mecanismo para garantir a propriedade de segurança confidencialidade na rede-em-chip SoCIN, por meio do algoritmo de criptografia AES.

Propõe-se que, como linha específica de pesquisa a ser tratada neste trabalho, a implementação da cifra em blocos (algoritmo) SIMON na rede SoCIN visando avaliar sua viabilidade, aferindo corretude de funcionamento, degradação de desempenho causada e outras características que pareçam pertinentes durante o progresso das atividades.

SIMON é, dentre diversos mecanismos de segurança existentes, uma família de cifras de blocos publicamente disponibilizada pela Agência Nacional de Segurança (NSA – National Security Agency dos Estados Unidos da América), em 2013, e que tem dupla ênfase: é relativamente leve e projetada para desempenhar bem quando implementada diretamente em hardware (BEAULIEU *et. al.*, 2013).

Para o desenvolvimento, será utilizada a NoC parametrizável de topologia em malha 2-D denominada SoCIN. Essa plataforma é estabelecida como “de referência” e é utilizada em diversos estudos na área de NoCs no Laboratório de Sistemas Embarcados e Distribuídos e, recentemente, com contribuições do Laboratório de Redes de Computadores, ambos da Universidade do Vale do Itajaí (UNIVALI).

1.1 PROBLEMATIZAÇÃO

Redes de computadores tradicionais são susceptíveis a falhas de segurança sob forma de ataques às informações nela trafegadas. Redes-em-chip, que visam permitir a mesma escalabilidade em número de nós comunicantes das redes tradicionais, podem herdar, naturalmente, muitas dessas vulnerabilidades.

Um ataque à NoC pode afetar qualquer uma das propriedades esperadas de segurança: pode tornar-se inoperante, se inundada com tráfego propositalmente malicioso, revelar e alterar informações que transitam entre núcleos para um elemento posicionado no “meio do caminho” (*man-the-middle*) e forjar conteúdo, entre muitas outras.

Para cada tipo de ameaça, existe um conjunto de mecanismos de segurança cuja função é tratá-la, amenizá-la ou eliminá-la. Este trabalho, particularmente, concentra-se em avaliar a cifragem do SIMON como mecanismo para garantir confidencialidade na comunicação entre

dois núcleos. Nesse sentido, entende-se como confidencial o aspecto da informação que, apesar de detectável, é incompreensível para os elementos posicionados “entre” os comunicantes, de tal modo que mesmo capturado, esse tráfego não faça sentido para o atacante.

A rede SoCIN servirá como ambiente para operacionalização dos experimentos e permitirá discussões em cima das questões que permanecem em aberto: desempenho oferecido, sobrecarga causada por criptografia, viabilidade da SoC mediante alterações de incorporação desses recursos, entre muitas outras.

1.2 FORMULAÇÃO DO PROBLEMA

Considerando as possíveis ameaças à segurança das NoCs, é de grande importância garantir a normalidade funcional da rede em casos de ataque. Sendo esses ataques em parte visando capturar informações, a implantação de criptografia entre as comunicações se apresenta como solução natural para tratar a vulnerabilidade. Contudo, algoritmos criptográficos tradicionais possuem estruturas robustas, com grande necessidade de recursos para geri-los. Assim o desenvolvimento e/ou implantação de algoritmos com menores necessidades de recurso são importantes, principalmente quando essas implantações são voltadas a hardware.

A adoção de algoritmos de criptografia nas comunicações intrachip das redes-em-chip, se torna cada vez mais comum. Assim, redes que não os possuem ficam expostas a ataques e, consequentemente, há possibilidades de falhas e/ou indisponibilidade de uso.

Analisando a arquitetura da rede-em-chip SoCIN percebe-se que não há nenhuma estrutura em sua naturalidade responsável por prover segurança entre as comunicações que envolvam os núcleos. Dessa forma, a implementação de uma criptografia nas trocas de mensagens da rede SoCIN é relevante para garantir a confidencialidade das informações.

Uma vez que o SIMON, abordado na Seção 2.4.2, *parece ser* o algoritmo criptográfico mais adequado para este trabalho, a pergunta de pesquisa se formula da seguinte forma:

Qual impacto de desempenho ao se prover confidencialidade na rede-em-chip SoCIN através da cifra em bloco SIMON?

1.2.1 Solução Proposta

A solução é, entre diversas possíveis, de avaliar uma possível implementação do algoritmo de criptografia SIMON na comunicação entre núcleos da SoCIN. Essa escolha fundamenta-se essencialmente pelos resultados obtidos por Sopran (2016) que aponta a cifra de

blocos como substancialmente adequada em se tratando de implementação diretamente em hardware.

A abordagem será explorada por simulação funcional de simulador de arquitetura, onde emprega-se a descrição mais atualizada e adequada da SoCIN.

1.3 OBJETIVOS

Este Trabalho Técnico-Científico de Conclusão de Curso é constituído de um objetivo geral e cinco objetivos específicos.

1.3.1 Objetivo Geral

Avaliar a cifra em blocos SIMON enquanto mecanismo criptográfico para prover confidencialidade na rede-em-chip SoCIN através da abordagem de simulação.

1.3.2 Objetivos Específicos

1. Fundamentar os conceitos relacionados ao tema de pesquisa deste trabalho.
2. Qualificar os riscos de confidencialidade no tráfego intrachip da NoC SoCIN.
3. Desenvolver ou portar uma implementação da cifra de blocos SIMON na rede-em-chip SoCIN.
4. Avaliar a implementação do algoritmo na rede SoCIN sob os enfoques de eficácia e impacto ao desempenho.
5. Documentar este volume e produzir um artigo científico relatando os resultados obtidos e submetê-lo a um evento ou publicação específica da área.

1.4 METODOLOGIA

Considera-se esse trabalho uma pesquisa científica com foco na área de segurança. Barros e Lehfeld (2000), definem que a pesquisa científica é o produto de uma investigação, com o objetivo de resolver problemas e solucionar dúvidas mediante a utilização de procedimentos científicos.

O presente trabalho também se classifica por sua natureza aplicada de pesquisa, sendo que um dos objetivos é implementar de forma efetiva uma solução para o problema de pesquisa. Essa natureza e descrita por Gil (2002) por ter como objetivo a produção de conhecimentos que tenham aplicação prática e dirigidos à solução de problemas reais específicos, envolvendo verdades e interesses locais.

Ao fim desse trabalho foram avaliados os resultados obtidos no experimento com a finalidade de verificar a eficácia da solução proposta. A validação por experimentação em ambiente de simulação enquadra essa série de experimentos – a implementação do algoritmo criptográfico em si – como do tipo quantitativa. Já no todo, isto é, a implementação e subsequente série de experimentos desencadeados durante o desenvolvimento, torna este trabalho uma pesquisa mais de caráter livre e exploratório (NEVES; DOMINGUES, 2007).

1.5 ESTRUTURA DO TRABALHO

Este documento está estruturado em quatro capítulos. O Capítulo 1, Introdução, apresentou uma visão geral do trabalho. No Capítulo 2, Fundamentação Teórica, é apresentada uma revisão bibliográfica sobre: sistema-em-chip, rede-em-chip e segurança da informação, assim como uma análise a respeito dos motivos relacionados a escolhas dos artifícios para realização do presente trabalho. Nesse capítulo, também é feita uma descrição de trabalhos relacionados sobre segurança em redes-em-chip e implantações do algoritmo SIMON. O Capítulo 3 apresenta como o projeto será desenvolvido. O capítulo também discute a metodologia a ser utilizada no desenvolvimento, e os possíveis problemas a serem encontrados durante o desenvolvimento. Concluindo, no Capítulo 4, apresentam-se as Conclusões, onde são abordados os resultados esperados ao final da segunda etapa do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo encontra-se a fundamentação necessária para sustentar o desenvolvimento, adiante, no Capítulo 3.

2.1 SISTEMA-EM-CHIP

Um sistema-em-chip (do inglês *System-on-Chip* - SoC) é um circuito integrado que implementa a maioria ou todas as funções de um sistema computacional completo (JERRAYA; WOLF, 2005).

O desenvolvimento de soluções com o uso de circuitos digitais parar sanar problemas do cotidiano é o maior responsável pela criação e o avanço das tecnologias que envolvem um sistema integrado (VAHID, 2007).

Inicialmente, os circuitos digitais eram projetados com equipamentos denominados relés, esses eram responsáveis por realizar o chaveamento do fluxo de energia, assim tornando os circuitos que o utilizavam mais autônomos. Por conta da lentidão do relé causada por motivos físicos, viu-se necessário o desenvolvimento de outra maneira de automatizar os circuitos de forma mais eficiente. Assim, os projetos de circuitos passaram a incorporar válvulas termiônicas para substituir a função do relé. Essas, por sua vez, geravam resultados mais rápidos em comparação à solução anterior. Projetos de grande importância foram realizados com o uso de válvulas termiônicas. À exemplo desses, pode se destacar o computador ENIAC (*Eletronic Numerical Integrator And Computer*), máquina essa, considerada o primeiro computador de propósito geral. Porém, o uso de válvulas termiônicas provoca um aumento significativo no consumo de energia e de calor, e sofriam de falhas frequentes. A invenção do transistor tornou possível o advento de computadores menores e mais confiáveis, comparado aos que utilizavam o mecanismo de válvula (VAHID, 2007).

Os transistores são qualificados como discretos. Essa descrição refere-se à sua característica de estado sólido, diferenciando-se assim, de seus antecessores que possuíam vácuo ou partes móveis no seu interior. Embora todos esses avanços citados anteriormente tenham contribuídos significativamente para a evolução do uso de circuitos digitais, nenhum deles impactou tanto a área quanto a invenção dos circuitos integrados (CIs). O uso de CIs revolucionou a computação. Com ele foi possível sintetizar milhares de transistores em apenas uma placa de silício (*chip*). Com o passar do tempo, os *chips* de silício vêm recebendo cada vez mais transistores em suas composições, tornando assim cada geração dos *chips* capacitada à

realização de mais tarefas simultâneas e, com maior eficiência, em cada uma delas (VAHID, 2007).

Com o aumento no número de transistores em um único chip, novos componentes foram adicionados. Assim, se fez possível o desenvolvimento de sistemas com múltiplos núcleos (*multi-core*). Esses SoCs com múltiplos núcleos são denominados MPSoCs (do inglês *Multiprocessor System-on-Chip*). Tipicamente a grande maioria dos SoCs são MPSoCs, pois é muito difícil projetar SoCs complexos que possuam uma única CPU (*Central Processing Unit*) (JERRAYA; WOLF, 2005).

Esse crescimento no número de transistores na composição dos *chips* foi estimado por Gordon Moore (MOORE, 1965), onde descreve que a quantidade de transistores que compõe o *chip* dobraria a cada 18 meses, aproximadamente. Essa estimativa ficou conhecida como Lei de Moore, e é utilizada como referência desde então (JERRAYA; WOLF, 2005). Embora a estimativa tenha se mostrado verdadeira por um longo tempo, nos últimos anos ela tem sido refutada por conta das limitações físicas que envolvem o funcionamento do *chip* (HUANG, 2015).

A Figura 1 exibe as expectativas de perpetuação da Lei de Moore. Nela, Courtland (2016) demonstra a fatia de mercado (em bilhões de dólares) e a tecnologia de fabricação (distância entre transistores em nanômetros) ao longo dos anos para o período compreendido entre 2016 e 2025.

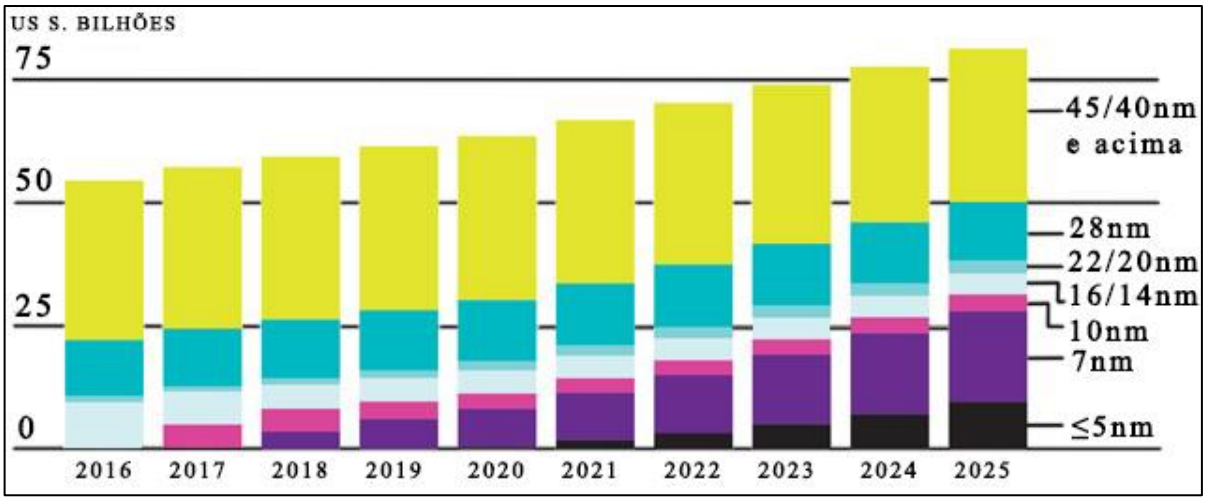

Figura 1. Projeção dos processadores
Fonte: Adaptado de Courtland, (2016).

O que se percebe é que atualmente, isto é, nos anos 2016 e 2017, há presença massiva das tecnologias de 45 nanômetros, com fatia de mercado superior a 50%, seguida das

tecnologias menores, de até 20 nanômetros e, em menor quantidade, as mais recentes arquiteturas que empregam distância entre transistores de apenas 14 nanômetros.

A parte mais relacionada a previsão em si, relata a introdução das arquiteturas de 10 e 7 nanômetros durante os anos de 2018 a 2025 e o aumento da concentração de mercado, progressivamente nessas escalas menores até o fim desse período, onde deve alcançar valores próximos a 50% da futura distribuição do mercado. Também se percebe a permanência de fabricação das tecnologias menos densas, bem como aumento acumulado de valores de venda em todas as tecnologias, devendo deixar o atual patamar concentrado de 50 bilhões de dólares/ano para alcançar a cifra de 75 bilhões/ano.

Para corroborar, Pressman (2017) apresentou a previsão de lançamento de processadores da fabricante Intel, que em nova geração, atingirá 10 nanômetros de distância entre transistores seguida, nos próximos anos, da subsequente entrada das arquiteturas mais avançadas de apenas 7 nanômetros.

2.1.1 Composição Sistema-em-Chip

Segundo Keutzer *et. al.* (2000), um SoC é formado por duas estruturas, (i) computação, responsável por armazenar e processar as informações; (ii) comunicação, responsável por estabelecer a comunicação entre os componentes existentes.

2.1.1.1 Estrutura de Memória

Em projetos de MPSoCs, fatores como tamanho de código, tempo de execução e consumo de energia são impactantes diretos no desempenho final. Considerando que sua utilização é principalmente empregada à sistemas de tempo real - pois fornecem simultaneidade computacional necessária para tratar eventos simultâneos no mundo real como por exemplo tratamentos de vídeo ou comunicação - sua arquitetura deve ser projetada de forma que otimize o uso de qualquer recurso necessário. Dos fatores que afetam significantemente o desempenho, destacam-se a arquitetura de processamento e a arquitetura de memória (JERRAYA; WOLF, 2005).

Os MPSoCs têm estruturas que se assemelham a computadores de processamento paralelo, os quais Hwang e Xu (1998, p. 237) classificam, de acordo com as estruturas de memória, em (*i*) memória compartilhada, com um único espaço de endereçamento e (*ii*) memória não-compartilhada, com múltiplos espaços de endereçamento (METZGER, 2014).

A estrutura de memória compartilhada ainda pode ser classificada em outras duas subcategorias, sendo; (*i*) memória centralizada e (*ii*) memória distribuída. A memória centralizada, também conhecida como sistema de acesso uniforme à memória (do inglês *Uniform Memory Access* - UMA), que se caracteriza por estar à mesma distância de todos os processadores, tornando assim o tempo de acesso de cada processador à memória, uniforme.

Segundo Larowe *et. al.* (1991), o uso de memória compartilhada de acesso uniforme em estruturas multiprocessadoras de barramentos é viável desde que o projeto seja de pequena escala.

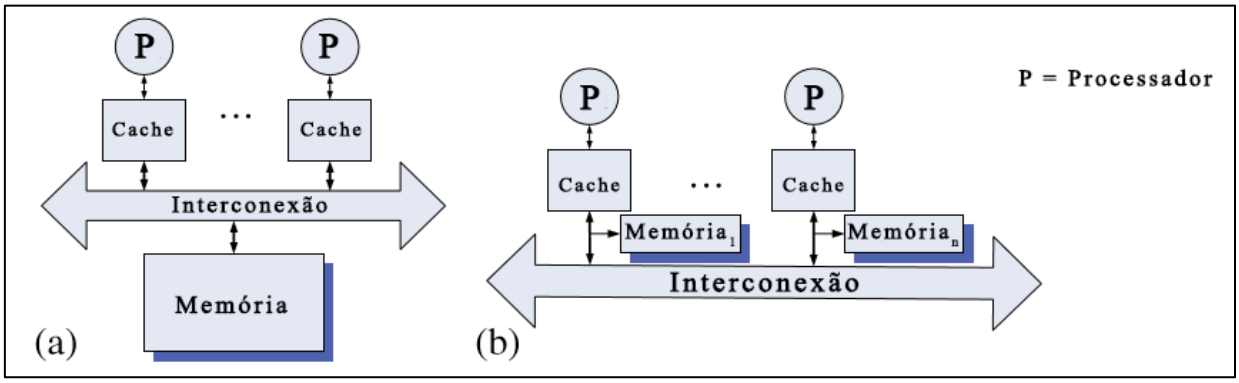

Figura 2. Arquitetura de memória. (a) arquitetura UMA; (b) arquitetura NUMA
Fonte: Adaptado de Zhanga *et. al.* (2014).

Em contrapartida às memorias compartilhadas UMA, existem memórias de acesso não uniforme, denominadas NUMA (do inglês *Non-Uniform Memory Access*). Esse sistema dedica uma parte da memória para cada processador, criando assim uma memória local para cada *core* (núcleo) do sistema. O conjunto delas forma a memória principal do sistema. Por sua vez, o uso de memórias compartilhadas com arquiteturas NUMA torna os acessos à memória realizados pelos núcleos assíncronos. Assim, os tempos de acessos, se tornam não uniformes (ZHANGA *et. al.*, 2014).

2.1.1.2 Arquiteturas de Comunicação

Em um MPSoC cujo sua arquitetura seja composta com uma grande quantidade de núcleos de processamento, é relevante considerar a técnica de comunicação empregada entre eles.

A arquitetura de comunicação é responsável por interconectar e gerir os múltiplos fluxos de informações, que trafegam entre os diversos componentes de um MPSoC, podendo ela, caso de mau empregada ou falha, ocasionar problemas como gargalo ou perdas de

informação por erro de rota em envio entre origem e destino, tornando o sistema inoperante. Entre os fatores que elevam a importância da arquitetura de comunicação na estrutura do MPSoC, pode se destacar a existência de múltiplas comunicações entre os diversos núcleos e componentes de forma simultânea (METZGER, 2014).

Existe, essencialmente, duas abordagens distintas para tratar a comunicação entre núcleos de um MPSoC. Barramento e/ou rede em chip.

Barramento

O barramento ou multiponto compartilhado é o exemplo mais simples de uma topologia de arquitetura de comunicação compartilhada e, é comumente encontrado em muitas aplicações de SoC. Seu uso comum se dá pelo fato de ter uma topologia simples, de fácil implementação e do baixo custo de área. Em relação as suas desvantagens destacam-se: sua falta de escalabilidade; consumo alto de energia e largura de banda limitada (SHANTHI; AMUTHA, 2011).

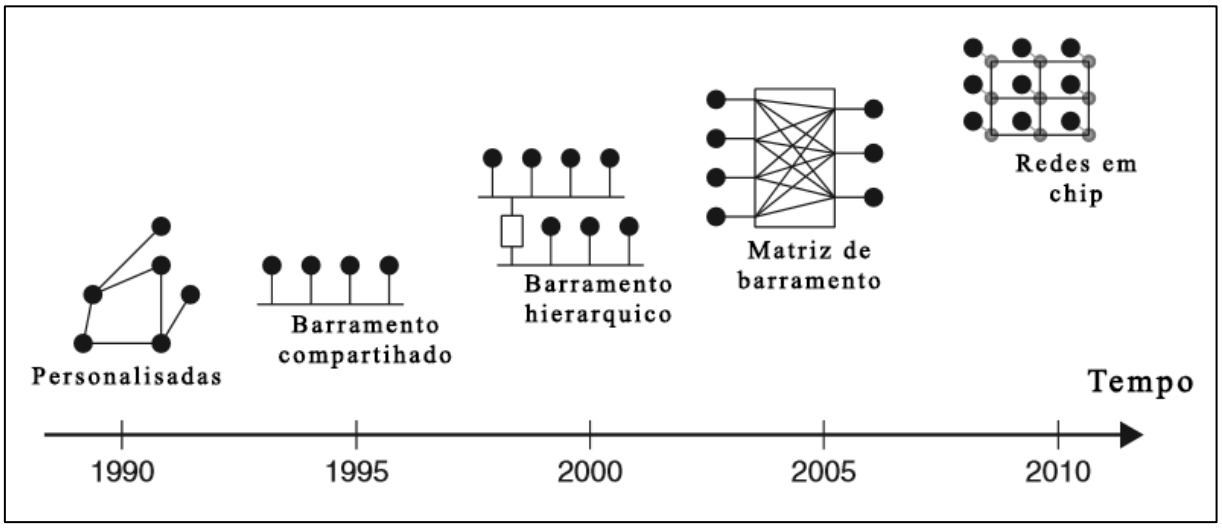

Figura 3. Evolução da arquitetura de comunicação intrachip
Fonte: Adaptado de Pasricha e Dutt (2010).

A utilização de barramento em um MPSoC limita o paralelismo das comunicações. Isso acontece por conta de seu funcionamento que é capaz de realizar apenas uma comunicação por vez. Com a sua utilização se torna possível o reuso dos núcleos, porém, os mesmos sofrem concorrência entre si para realizar suas comunicações com os componentes do sistema. Essas são tratadas por um controlador central, que escalona a utilização do barramento através de regras de arbitragem (ZEFERINO, 2003).

O sistema de arbitragem do barramento funciona de maneira simples; a unidade de controle de barramento, ou “árbitro”, é consultado antes de cada comunicação que necessita ser realizada, essa arquitetura é exemplificada na Figura 4. As comunicações são formadas por um componente “mestre”, responsável por iniciar a transferência, e um componente “escravo”, que será o destinatário da comunicação. A unidade de controle, em sistemas que possuem múltiplos mestres e recebe requisições concorrentes, podem utilizar-se de critérios pré-definidos ou dinâmicos para determinar qual comunicação será realizada. Após a escolha, o “árbitro” comunica o componente mestre escolhido e, assim, o mesmo realiza a transição (ZEFERINO, 2003).

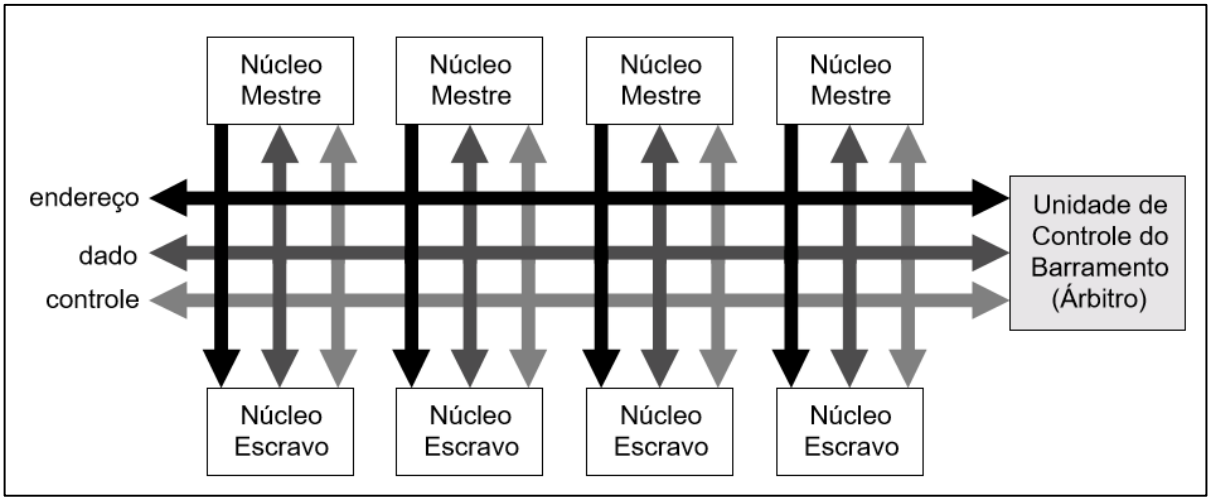

Figura 4. Arquitetura genérica de um SoC baseada em barramento
Fonte: Zeferino (2003).

Barramentos apresentam escalabilidade limitada, e o seu uso pode comprometer o desempenho de SOCs compostos por quantidades significativas de módulos com fluxo de dados intensivo. Em tais sistemas, o uso de redes intrachip, (do inglês Network-on-Chip ou NoC) pode ser mais adequado (JOHANN FILHO; PONTES; LEITHARDT, 2007).

Rede intrachip

Como alternativa aos problemas encontrados com o uso de barramento apresentados anteriormente, foi proposto uma nova arquitetura de comunicação denominada de rede intrachip ou rede-em-chip. Essa arquitetura será amplamente abordada no capitulo 2.2 do presente trabalho.

2.2 REDE-EM-CHIP

As redes-em-chip visam prover comunicação intrachip. Para solucionar problemas encontrados no uso de barramento, surgem as redes-em-chip. Essas redes chaveadas, quando aplicadas à comunicação intrachip, recebem múltiplas denominações na literatura como micronetworks, On-Chip Networks (OCNs) e Networks-on-Chip (NoC). Contudo, todas elas se referem a mesma base arquitetural, análogas a redes tradicionais de computadores paralelos (ZEFERINO, 2003).

Ao longo dos últimos anos, a ideia de usar NoCs como estrutura viável de comunicação para futuros sistemas multiprocessador em chips (MPSoCs), vem ganhando força. As NoCs são uma tentativa de reduzir os conceitos de redes de grande escala e aplicá-los ao domínio incorporado aos SoCs. Ao contrário das tradicionais arquiteturas de comunicação intrachip baseadas em barramento, os NoCs usam pacotes para encaminhar dados da fonte para o componente de destino, através de uma rede que consiste em switches (roteadores) e links de interconexão (PASRICHA; DUTT, 2010).

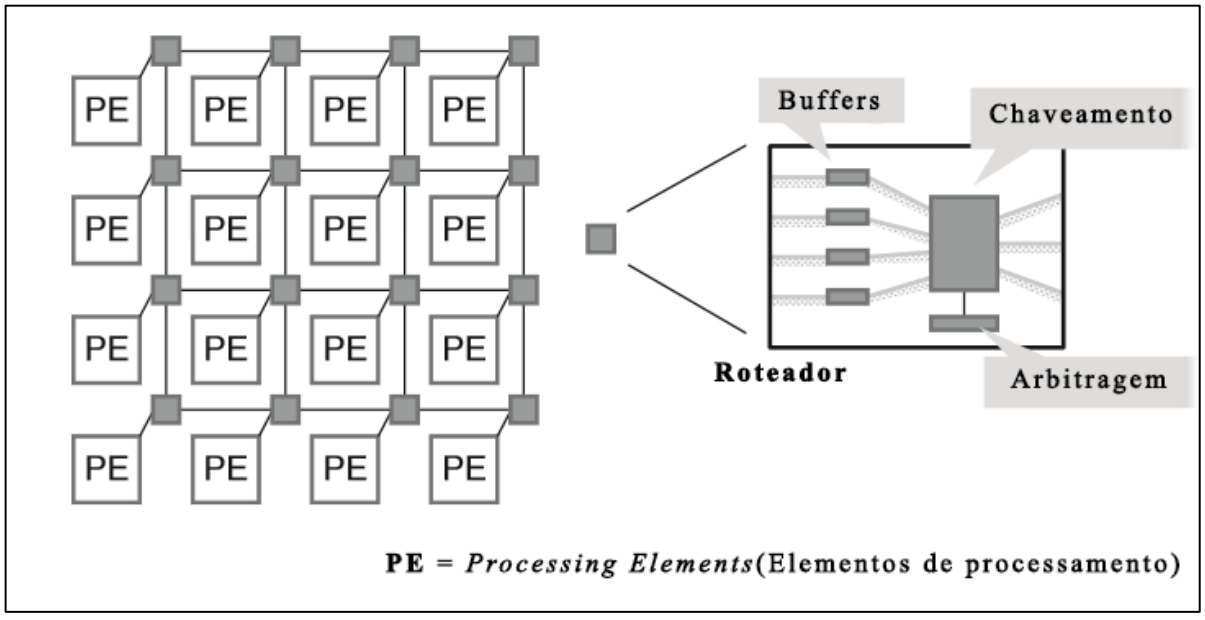

Figura 5. Exemplo de arquitetura de rede em chip
Fonte: Adaptado de Pasricha e Dutt (2010).

Um projeto de NoC pode ser especificado pela sua arquitetura e pelos seus mecanismos de comunicação. A topologia define a estrutura da rede, enquanto que a forma pela qual ocorre a transferência das mensagens é definida pelos mecanismos de comunicação. Os mecanismos de comunicação incluem controle de fluxo, chaveamento, memorização, roteamento e arbitragem (ZEFERINO, 2003).

2.2.1 Topologias

A topologia de um NoC especifica a organização física da rede de interconexão que conecta seus diversos componentes. As topologias aplicadas em NoCs podem ser classificadas em três tipos gerais, sendo eles redes diretas, redes indiretas e redes irregulares.

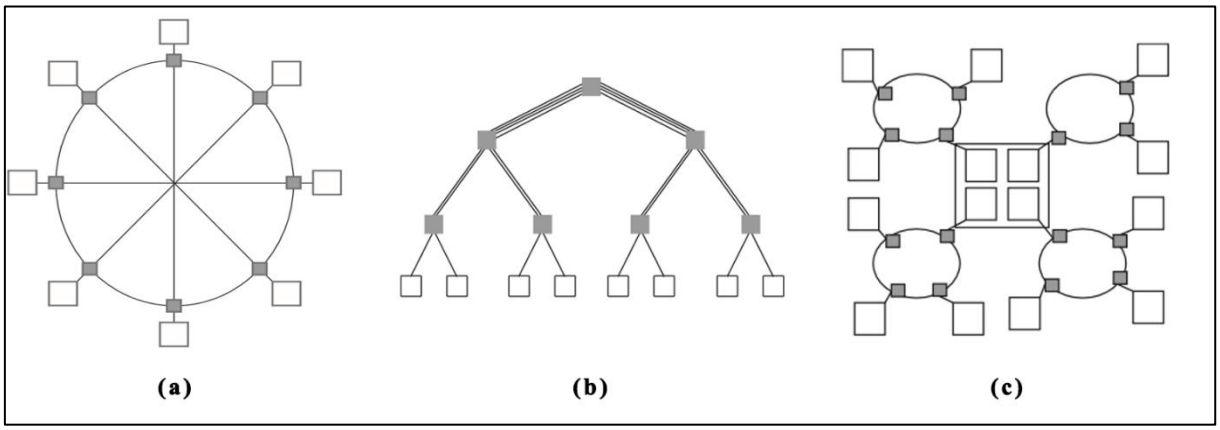

Figura 6. Exemplos de topologia. (a) Direta - Octagon; (b) Indireta - Fat tree; (c) Irregular – Baseada em Cluster, hibrida (malha + anel)
Fonte: Pasricha e Dutt (2010).

As topologias que se enquadram do tipo redes diretas, se caracterizam por cada um dos nós se conectar a outro por meio de links ponto-a-ponto com nós vizinhos. Quando a comunicação tem o destino além dos pontos vizinhos, a mensagem é trocada entre os roteadores até chegar ao ponto final. Assim cada componente do sistema possui um roteador exclusivo e responsável por realizar as comunicações entre os demais componentes do sistema.

Nas topologias de redes indiretas, os roteadores possuem um conjunto de portas bidirecionais para ligar com outros roteadores e/ou com os componentes do sistema. Somente alguns roteadores possuem conexão para os componentes e apenas estes últimos podem servir de fonte ou destinatário de uma mensagem. As topologias de rede irregulares são normalmente uma combinação de topologias de rede compartilhada direta e indireta. O objetivo dessas topologias é aumentar a largura de banda disponível em comparação com os barramentos compartilhados tradicionais e reduzir a distância entre os nós. As topologias irregulares são normalmente personalizadas para uma aplicação (PASRICHA; DUTT, 2010) (ZEFERINO, 2003).

A exemplo de uso de topologias em algumas arquiteturas de redes-em-chip pode-se destacar como do tipo direta a topologia denominada octagon, apresentada na Figura 6(a), essa é utilizada na NoC octagon. A topologia da arquitetura octagon e composta por 8 nós e 12 links bidirecionais. O uso de topologia indireta é visto na NoC SPIN, essa utiliza a topologia *fat-tree*

(“Arvore gorda”) que pode ser vista na Figura 6(b). Sua implementação é feita com dois caminhos de dados e links de 32 bits unidirecionais (PASRICHA; DUTT, 2010).

A respeito das topologias irregulares, suas arquiteturas são personalizadas para cada aplicação. A Figura 6(c) mostra um exemplo de uma topologia híbrida baseada em cluster que combina uma malha e uma topologia em anel. A exemplos de uso, Xpipes e Æthereal são dois tipos de arquiteturas de NoC que permitem o uso topologias irregulares (PASRICHA; DUTT, 2010).

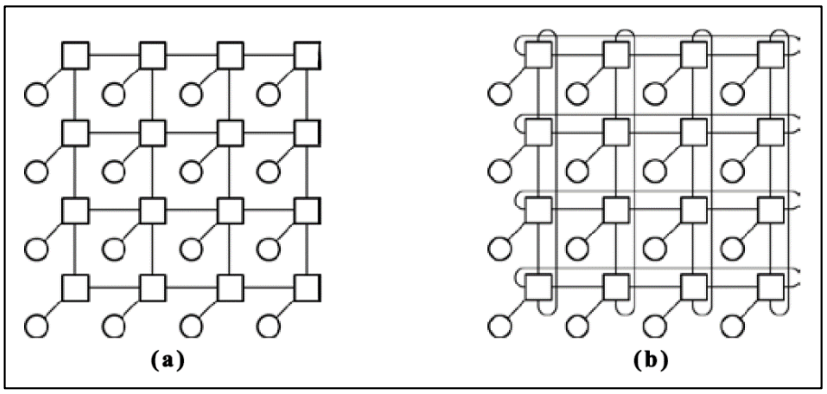

Figura 7. Topologias da SoCIN. (a) grade; (b) torus
Fonte: Adaptado de Reddy et. al (2014).

Em relação as topologias que podem ser utilizadas pela rede-em-chip SoCIN destacam-se duas do tipo 2-D direta: grade Figura 7(a) e torus Figura 7(b). A primeira é a mais comum entre todas as topologias de interconexão, onde cada roteador além daqueles nas bordas, está ligado a quatro roteadores adjacentes e um recurso de computação (núcleo), por meio de canais de comunicação. Esta topologia permite a incorporação de um grande número de componentes em uma estrutura de forma regular. A segunda topologia suportada pela rede-em-chip SoCIN é denominada de torus. Essa, por sua vez, tem uma semelhança com a topologia de grade com a diferença de que os roteadores de borda estão conectados aos roteadores das bordas opostas. Essa ligação entre bordas opostas define de forma padronizada que todos os roteadores possuem cinco portas, sendo elas quatro para ligar os roteadores adjacentes e uma para comunicação com um componente do sistema (comunicação local). Em sistemas que possuem distancias elevadas entre suas bordas podem sofrer atrasos em suas conexões (ZEFERINO; SUSIN, 2003; REDDY *et. al.*, 2014).

2.2.2 Roteamento

Segundo Duato *et. al.* (2003), o roteamento é responsável por estabelecer o caminho seguido por cada mensagem ou pacote de sua origem ao seu destino. De forma geral o algoritmo de roteamento escolhido para rede impacta consideravelmente no desempenho geral da rede. Algoritmos de roteamento possuem propriedades especificas, onde o comprimento das mesmas garante a funcionalidade e a assertividade do roteamento geral do sistema, são elas;

- Conectividade: Capacidade de encaminhar pacotes de qualquer nó de origem para qualquer nó de destino;
- Adaptabilidade: Capacidade de encaminhar pacotes através de caminhos alternativos na presença de contenção ou componentes defeituosos;
- Liberdade *deadlock* e *livelock*: Capacidade de garantir que os pacotes não bloquearão ou vagar pela rede para sempre; e
- Tolerância ao erro: Capacidade de encaminhar pacotes na presença de componentes defeituosos.

Os algoritmos de roteamento podem ser implementados de diferentes maneiras podendo se classificar em dois tipos; determinístico ou adaptativo. Algoritmos de roteamento determinístico sempre fornecem o mesmo caminho entre um dado par fonte-destino. Algoritmos de roteamento adaptativo usam informações sobre o tráfego de rede e/ou status do canal para evitar regiões congestionadas ou com defeito da rede.

2.2.3 Chaveamento

Em uma rede de interconexão, os dados são transferidos entre os componentes da rede através dos canais físicos que interligam esses componentes. O chaveamento define a forma pela qual esses dados são transferidos de um canal de entrada de um componente para um dos seus canais de saída.

Existem duas técnicas para realização do chaveamento, são elas: chaveamento por circuito e chaveamento por pacote. O chaveamento por circuito se caracteriza por reservar um caminho físico exclusivo entre o componente fonte e destinatário até o término da transferência da mensagem, este utiliza duas etapas para realizar o envio da mensagem, a primeira consiste em reservar o caminho físico ao destino final e notificar o componente fonte que o circuito estabelecido está pronto para transportar a mensagem. A segunda etapa é a transferência da mensagem em si e a liberação do caminho reservado. Diferentemente do método anterior o chaveamento por pacote não necessita reservar um caminho completo até o destino da

mensagem, para iniciar a transferência. Isso se dá pelo fato de que, nesse tipo de chaveamento, a mensagem é dividida em pacotes de tamanho fixo. Assim é necessário alocar somente os recursos para o envio do "pacote da vez". Dessa forma os roteadores armazenam o pacote em seu *buffer* e assim que possível encaminha para o próximo roteador ou componente, caso ele seja o destinatário da mensagem. (PASRICHA; DUTT, 2010) (ZEFERINO, 2003).

2.2.4 Controle de Fluxo

Pasricha e Dutt (2010) definem o controle de fluxo como o responsável por alocar os recursos necessários, para que os pacotes atravessem a rede e a mensagem seja transmitida até seu destinatário, podendo esse recurso ser capacidade dos *buffers*, largura de banda dos canais ou ainda uma combinação de ambos.

O controle de fluxo é fortemente acoplado com algoritmos de gerenciamento de *buffer* que determinam como os *buffers* de mensagem são solicitados e liberados e, como resultado, determinam como as mensagens são manipuladas quando bloqueadas na rede (DUATO *et. al.*, 2003).

Para Duato *et. al.* (2003), o controle de fluxo é um protocolo de sincronização para transmitir e receber uma unidade de informação. Assim a unidade de controle de fluxo refere-se à parte da mensagem cuja transferência deve ser sincronizada. Um exemplo prático de uso do protocolo de controle de fluxo se dá em uma transmissão de mensagens entre os componentes do sistema. Esse é responsável pela solicitação e confirmação dos recursos necessários para trafego da mensagem no receptor. Essa sinalização é usada para garantir a disponibilidade de espaço no buffer receptor. A eficiência de implementação gerencia a troca real destes sinais e, consequentemente, a utilização controlada dos recursos do sistema evitando assim o sobrecusto, permitindo um melhor compartilhamento dos *buffers* (DUATO *et. al.*, 2003).

Essas transferências são atômicas, no sentido de que é necessário fornecer buffer suficiente para que um pacote seja transferido na sua totalidade ou a transmissão seja atrasada, até que um espaço de buffer suficiente fique disponível. Assim, técnicas de controle de fluxo são classificadas por conta de sua granularidade, podendo operar com pacotes ou *flits* (do inglês *flow control units*). Um *flit* é a unidade básica utilizada pela maioria das técnicas de controle de fluxo, mas não contém nenhuma informação sobre roteamento, sendo necessário que siga o mesmo caminho já determinado (DALLY; TOWLES, 2004) (DUATO *et. al.*, 2003).

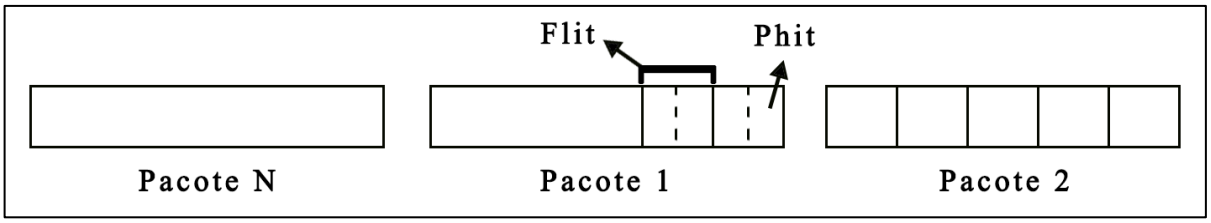

Figura 8. Unidades de controle de fluxo utilizadas na mensagem
Fonte: Adaptado de Duato *et. al.* (2003).

Os pacotes, por sua vez, podem ser divididos em unidades de controle de fluxo de mensagens ou *flits*. Devido às restrições de largura de canal, vários ciclos de canal físico podem ser usados para transferir um único *flit*. Um *phit* (do inglês *physical units*) é a unidade de informação que pode ser transferida através de um canal físico em um único passo ou ciclo. *Flits* representam unidades lógicas de informação, ao contrário de *phits*, que correspondem a quantidades físicas, ou seja, o número de bits que podem ser transferidos em paralelo em um único ciclo.

2.2.5 Arbitragem

Enquanto o roteamento determina qual canal de saída deve ser usado por um pacote que chega a um canal de entrada, a arbitragem define qual canal de entrada (ou *buffer* de entrada) poderá se utilizar de um determinado canal de saída (ou *buffer* de saída) (ZEFERINO, 2003).

Pode-se definir a arbitragem como um mecanismo que contém um conjunto de critérios responsáveis por priorizar a saída dos pacotes do roteador, sendo capaz de resolver conflitos provocados por inúmeros pacotes competindo pelos canais de saída. Outra característica que a arbitragem possui é de evitar que pacotes fiquem em espera indefinida no roteador para seguir até o destino, fenômeno esse conhecido como *startvation*.

Em contraste com o barramento que possui uma arbitragem centralizada, a arbitragem em uma NoC é distribuída. Isso se dá pelo fato de ser realizada em cada roteador, e ser baseada apenas em informações locais. A arbitragem dos recursos de comunicação (links, buffers) é realizada incrementalmente à medida que a solicitação ou resposta avança (DE MICHELI, 2006).

2.2.6 Arquiteturas de redes-em-chip

Nesta seção serão descritas algumas arquiteturas de redes-em-chip, suas definições, aplicações e características. Entre essas apresentadas, destaca-se a rede-em-chip SoCIN (2.2.6.4), rede a qual esse trabalho utilizará como base para seu desenvolvimento.

2.2.6.1 Æthereal

A rede-em-chip Æthereal conhecida inicialmente como rede sobre silício, teve seus conceitos e definições criadas entre os anos 2000 e 2001 pela Philips Research, para sistemas-em-chip destinados a equipamentos elétricos de consumo, em particular a TV digital (DTV) e as set-top boxes (STB). Esses sistemas incluem aplicações como descodificação e melhoria de áudio e vídeo que possuem requisitos reais e alto desempenho computacional a baixo custo (alta largura de banda (Gb/s) a área (mm^2)). Além disso, ao contrário da computação de propósito geral, os consumidores não toleram equipamentos com falhas ou que apresentem mau funcionamento, por esse motivo os sistemas-em-chip composto com essa rede-em-chip é mais robusto e menos suscetíveis a "quebras" (GOOSSENS; HANSSON, 2010).

2.2.6.2 HERMES

Desenvolvido na Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), o sistema HERMES é uma infraestrutura para gerar redes-em-chip do tipo direta com uma topologia em malha 2-D bidirecional. Diz-se infraestrutura por não ser se tratar de uma única rede e sim um mecanismo gerador de redes parametrizáveis. Existe um conjunto de parâmetros definíveis pelo usuário, tais como: (i) o controle de fluxo; (ii) a dimensão da rede; (iii) a largura do canal de comunicação; (iv) a profundidade dos buffers; e (v) o algoritmo de roteamento. O uso da topologia malha bidirecional é justificado por facilitar as tarefas de posicionamento e de roteamento em circuitos integrados e/ou FPGAs. Nessa topologia, cada roteador tem um número diferente de portas, dependendo de sua posição no que diz respeito aos limites da rede.

O chaveamento por pacote *wormhole* implica a divisão do pacote em *flits*. Sendo que os dois primeiros *flits* dizem respeito as informações de cabeçalho, tendo em sua composição o endereço do roteador destino e o número de *flits* do corpo do pacote. Outra característica do sistema HERMES é possuir em cada porta de entrada um *buffer* para diminuir a perda de desempenho com o bloqueio de *flits*. Essa perda de desempenho ocorre quando um *flit* é bloqueado em um roteador. Assim que bloqueado, os *flits* seguintes também permanecem bloqueados no roteador local e em outros roteadores da rede ou mesmo no núcleo. Com a

inserção de *buffers* nas portas de entradas, os *flits* novos no roteador entram em uma fila, e permanecem ali até o desbloqueio do primeiro *flit* dessa fila. O uso dessa estratégia faz com que as demais portas de entrada não fiquem bloqueadas, assim diminuindo as chances de bloqueio da rede (MELLO, 2006).

2.2.6.3 Xpipes

A rede Xpipes foi proposta por Bertozzi *et. al.* (2004) na Universidade de Bologna, para ser utilizada em MPSoCs. A Xpipes tem como objetivo fornecer alta performance e comunicação confiável para SoCs (BERTOZZI, 2004).

Ela possui roteamento *wormhole* e faz uso do algoritmo de roteamento estático denominado *street sign routing*. Este algoritmo de roteamento permite uma implementação simples do roteador, porque nenhuma decisão dinâmica tem que ser tomada no mesmo. A rede intra-chip Xpipes possui alto grau de parametrização, essa inclui o tamanho do flit, o espaço de endereçamento dos núcleos, o número máximo de roteadores entre dois núcleos, o número máximo de bits para controle de fluxo ponto-a-ponto, a profundidade do buffer, o número de canais virtuais por canal físico, entre outros (MELLO, 2006).

Para criação de projetos com rede-em-chip Xpipes é necessária uma ferramenta para instanciar os blocos construtivos da rede (roteadores, canais e interfaces de rede). Esta ferramenta é denominada XpipesCompiler, seu funcionamento pode ser dividido em três etapas. A primeira delas é responsável por definir os parâmetros da rede para aplicação que a mesma será utilizada, a segunda é a instancia do software em si e a terceira a geração da descrição em SystemC, pronta para ser compilada e simulada (MELLO, 2006).

2.2.6.4 SoCIN

A SoCIN é uma NoC escalável baseada em uma arquitetura de roteador parametrizável para ser usada na síntese de redes de baixo custo. Sua arquitetura é baseada no chaveamento *wormhole* e usa originalmente algoritmo de roteamento determinístico XY baseado na origem (SUSIN e ZEFERINO, 2003).

A NoC SoCIN pode ser construída em cima de dois tipos de topologias malha (Figura 7(a)) e torus (Figura 7(b)). Em relação a escolha entre o uso das topologias, cabe uma avaliação do cenário de aplicação da rede. Ao usar a topologia do tipo malha o custo é mais baixo em relação ao uso da torus. Entretanto, ao usar a segunda opção, há uma redução de latência entre as mensagens.

SoCIN usa *wormhole* como abordagem de comutação de pacotes. As mensagens são enviadas por meio de pacotes, que são compostos por *flits*. Um *flit* é a menor unidade sobre a qual o controle de fluxo é executado. Na SoCIN os núcleos podem ser classificados em dois tipos quando se trata de comunicação; núcleo iniciador e núcleo alvo. O núcleo iniciador na comunicação é o responsável por realizar a requisição ao núcleo alvo, o alvo por sua vez tem a função de responder à requisição ao iniciador. O modelo de comunicação que formata a rede SoCIN é baseado em troca de mensagens. Assim, todas as requisições e respostas pertencentes a rede são realizadas por mensagens. Salienta-se também que um núcleo pode ser simultaneamente um iniciador e um alvo. (SUSIN e ZEFERINO, 2003).

A rede SoCIN em seu projeto inicial possuía um roteador parametrizável conhecido como RASoC (do inglês Router Architeture for SoC). Por conta das limitações que envolviam as parametrizações foi desenvolvido uma nova versão da SoCIN, Essa nova versão da SoCIN foi denominada SoCIN*fp* (SoCIN *fully parameterizable*, ou seja, SoCIN totalmente parametrizável). Essa ganhou novas possibilidades de parametrização nas técnicas de arbitragem e controle de fluxo. Por conta disso, um novo roteador foi implementado, esse foi denominado ParRIS (do inglês Parameterizable Interconnect Switch). Essa abordagem permite a implementação de diferentes algoritmos de roteamento, desde roteamentos determinísticos a totalmente adaptativos (ZEFERINO; SANTO; SUSIN, 2004; BARON, 2013).

O funcionamento do roteador PaRIS permite que cada roteador da rede possua até cinco conexões ao máximo, sendo que uma delas essencialmente, seja conectada no núcleo local e as outras quatro conectadas nos roteadores vizinhos. O roteador ParIS ainda suporta controle de fluxo baseado em *handshake* ou em créditos, memorização na entrada e/ou na saída, roteamento XY (os pacotes devem se deslocar primeiro no eixo X e depois no eixo Y) ou WF (caso o destino esteja à oeste da origem, o deslocamento dos pacotes acontece primeiro na direção oeste e depois nas demais direções) e arbitragem estática ou dinâmica (ZEFERINO; SANTO; SUSIN, 2004; METZGER, 2014).

2.3 ESCOLHA DA REDE-EM-CHIP

A escolha da rede-em-chip que será utilizada no trabalho foi feita através da análise de dois aspectos fundamentais para que se consiga desenvolver o mesmo. Sendo esses aspectos: (i) disponibilidade do projeto da rede-em-chip, para realizar as implementações e testes e (ii) existência de vulnerabilidades na rede-em-chip em relação a propriedade de confidencialidade.

Em relação a plataforma de operacionalização dos experimentos desse trabalho, foi obtida através do Laboratório de Sistemas Embarcados da Universidade do Vale do Itajaí (UNIVALI) uma versão da rede-em-chip SoCIN, juntamente com o RedScarf (um ambiente de avaliação de desempenho de NoC que possui acurácia em nível de ciclos). Sendo ambas descrita na linguagem SystemC.

O aspecto de vulnerabilidades da rede SoCIN foi abordado por Baron (2013). Em seu trabalho foram elencadas as vulnerabilidades presentes na rede, porém com enfoque na propriedade de segurança de disponibilidade. Entretanto, as evidências de vulnerabilidade na rede SoCIN em relação a propriedade de confidencialidade se apresenta quando se observa a sua arquitetura. Em suas possíveis configurações (já que sua arquitetura é parametrizável), a rede SoCIN não apresenta nenhuma opção de criptografia na geração de dados ou na troca de mensagens entre os seus componentes. Sendo a criptografia a solução natural para problemas de confidencialidade, conclui-se que a rede em questão é vulnerável nesse aspecto.

Embora Silva (2015) tenha abordado o tema de confidencialidade na SoCIN, sua solução foi desenvolvida em cima do algoritmo de criptografia AES (do inglês *Advanced Encryption Standar*). Esse, por sua vez não se enquadra na relação apresentada por Sopran (2016) como um algoritmo de criptografia leve. Assim, a rede SoCIN ainda carece de uma solução criptográfica que forneça confidencialidade nas suas comunicações e utilize a menor quantidade de recursos possível.

2.4 SEGURANÇA DE INFORMAÇÕES

Segundo Tanenbaum (2011), o crescente desenvolvimento de técnicas voltadas a tratar vulnerabilidades da área de segurança da informação se dá por conta do aumento significativo e constante de informações disponíveis nas redes de computadores, informações essas geradas por milhões de usuários nos mais diversos serviços computacionais.

Landwehr (2001), um sistema seguro se caracteriza por manter suas propriedades de segurança livres de vulnerabilidades. Essas propriedades são:

- Confidencialidade: assegurar que as informações não sejam divulgadas sem a devida autorização;
- Integridade: assegurar que as informações no computador não sejam modificadas sem a devida autorização;
- Disponibilidade: assegurando que as informações sejam acessíveis aos usuários legítimos quando necessário.

- Autenticação: assegurar que cada elemento é quem afirma ser; e
- Não- repúdio: assegura que o autor da ação não negue o feito.

Em particular, a criptografia, figura como mecanismo comumente empregado para preservar confidencialidade e integridade.

A criptografia é entendida como conjunto de métodos e técnicas para cifrar ou codificar informações legíveis através de um algoritmo convertendo um texto original em um texto ilegível, sendo possível através do processo inverso recuperar as informações originais (SIMON, 1999).

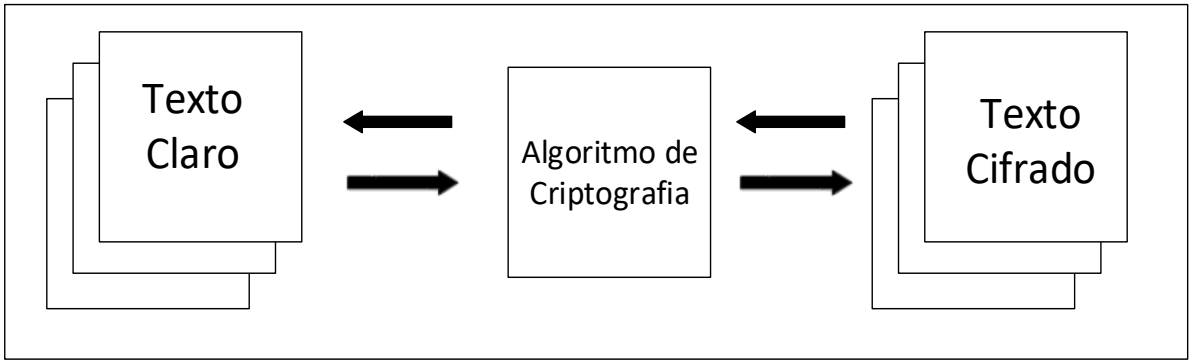

Figura 9. Esquema geral para cifragem de um texto
Fonte: Adaptado de Moreno (2005).

O uso da criptografia foi uma das primeiras medidas de segurança adotadas para assegurar que as informações não sejam reveladas a pessoas não autorizadas. Desde seus primeiros registros de utilização até o atual momento, a criptografia vem incorporando complexos algoritmos matemáticos em meio a esses avanços. Assim, surgiram algoritmos de criptografia robustos constituídos por sequências de procedimentos capazes de cifrar e decifrar grande quantidade de dados (MORENO; PEREIRA; CHIARAMONTE, 2005).

2.4.1 Mecanismos de cifra em blocos

A maioria dos algoritmos de criptografias usados atualmente baseiam-se em estruturas conhecidas como cifra de blocos. Uma cifra de bloco se caracteriza por tratar um bloco de texto claro como um todo, e usá-lo para produzir texto cifrado com o mesmo tamanho. Normalmente é utilizado um tamanho de bloco de 64 ou 128 bits (STALLINGS, 2008). A Figura 10 representa a segmentação em blocos de tamanho *b*, em relação ao texto claro e completo *M*.

$$\mathcal{M} = \{\overbrace{0\ldots00}^{b}, \overbrace{0\ldots01}^{b}, \overbrace{0\ldots10}^{b}, \overbrace{0\ldots11}^{b}, \ldots\ldots, \overbrace{1\ldots1}^{b}\}.$$

Figura 10. Exemplo de conjunto e separação de bloco
Fonte: Knudsen (2011).

Pode-se definir dois importantes parâmetros para a cifra em bloco:

- Tamanho do bloco (denotado por b); e
- Tamanho da chave (denotado por k).

Tendo uma determinada chave *k* e um bloco de *b* bits contendo o texto claro, para produzir um bloco de *b* bits com o texto criptografado, existem 2^b diferentes blocos de texto claro possíveis e, para a criptografia ser reversível, cada um produz um único bloco de texto cifrado (STALLINGS, 2008).

Tabela 1 - Mapeamentos reversíveis e irreversíveis para um bloco de b = 2

Mapeamento Reversível		**Mapeamento Irreversível**	
Texto Claro	Texto Cifrado	Texto Claro	Texto Cifrado
00	11	00	11
01	10	01	10
10	00	10	01
11	01	11	01

Fonte: Adaptado de Stallings, 2008.

Conforme apresentado na Tabela 1, no mapeamento irreversível, percebe-se que o texto cifrado “01” é a decifragem de dois blocos diferentes, impossibilitando a descriptografia, pois não é possível a decifragem quando são retornadas respostas iguais a entradas diferentes. Já no mapeamento reversível é perceptível que cada bloco tem 2^2 (2^b) transformações diferentes.

Para garantir uma cifra de bloco segura, não se espera que informações exploráveis sobre o processo de criptografia vazem. Essas fontes podem incluir dados sobre a escolha da chave, informações sobre a criptografia ou descriptografia de entradas ainda não vistas, ou informações sobre as permutações geradas usando chaves diferentes (KNUDSEN, 2011). Porém, existe um problema prático com a cifra em bloco ao utilizá-la com blocos pequenos como demonstrado na Tabela 1. Esse problema caracteriza-se por tornar a cifragem vulnerável a técnicas simples, como a análise do texto claro.

Dessa forma, inúmeros algoritmos de criptografia surgiram, cada um com seu propósito de aplicação específico. Sopran (2016) relacionou vinte métodos criptográficos, todos baseados em chaves simétricas e cifras em blocos, levando como parâmetro o algoritmo, ano de publicação, especificação, disponibilidade de implementação em software e hardware, e suas aplicações em sistemas embarcados.

Tabela 2 - Relação de algoritmos de criptografia.

Algoritmos	**Técnica**	**Ano**	**Especificação**	**Disponibilidade Implementação em Hardware**	**Disponibilidade Implementação em Software**	**Aplicação em Sistemas Embarcados**
DES	Simétrica	1976	Aberta	Sim	Sim	Videocipher II, um sistema de cifragem de um satélite de TV
IDEA	Simétrica	1991	Aberta	Sim	Sim	IDEACrypt Kernel (Ascom)
BLOWFISH	Simétrica	1994	Aberta	Sim	Sim	Uso não identificado
TWOFISH	Simétrica	1998	Aberta	Sim	Sim	Uso não identificado
RC6	Simétrica	1998	Aberta	Sim	Sim	Uso não identificado
SERPENT	Simétrica	1998	Aberta	Sim	Não	Uso não identificado
CAMELLIA	Simétrica	2000	Aberta	Sim	Sim	Uso não identificado (Aplicações previstas do domínio automóvel e móvel)
NOEKEON	Simétrica	2000	Aberta	Sim	Sim	Uso não identificado
UNICORN-A	Simétrica	2000	Aberta	Sim	Não encontrado	Uso não identificado
AES	Simétrica	2001	Aberta	Sim	Sim	Intel and AMD processors, IBM zSeries, SPARC S3, SPARC T4, SPARC T5
ARIA	Simétrica	2003	Aberta	Sim	Sim	Uso não identificado
SHA-2	Simétrica	2003	Proprietária	Sim	Não encontrado	DRM, Digital cards e discos de DVD Áudio e vídeo
IDEA NXT	Simétrica	2003	Aberta	Sim	Sim	Uso não identificado
SMS4	Simétrica	2006	Aberta	Sim	Sim	Utilizado no “the Chinese National Standard” para wireless LAN WAPI
CLEFIA	Simétrica	2007	Proprietária	Sim	Sim	Dispositivos áudio e vídeo e sistemas DRM
PRESENT	Simétrica lightweight	2007	Aberta	Sim	Sim	Criptografia leve, cogitado para etiquetas RFID, sensores, cartões inteligentes, etc..
THREEFISH	Simétrica	2008	Aberta	Sim	Sim	Uso não identificado
CHIASMUS	Simétrica	2013	Fechada	Não[5]	Não encontrado	Uso não identificado
SPECK	Simétrica lightweight	2013	Aberta	Sim	Sim	ASIC, processadores avançados (high-end), ARM Cortex-M3 microcontroladores
SIMON	Simétrica lightweight	2013	Aberta	Sim	Sim	

Fonte: Sopran (2016).

Dentre os algoritmos apresentados na Tabela 2, pode-se observar que três são classificados como “simétrico *lightweight*”, essa nomenclatura se dá a criptografias leves, ou algoritmos criptográficos adaptados à ambientes restritos como etiquetas RFID (do inglês

Radio-Frequency IDentification), sensores, cartões inteligentes sem contato e dispositivos portáteis em geral, sendo esses os mais recomendados para implementação em hardware, pois já são projetados para tal objetivo (SOPRAN, 2016).

A característica de alguns algoritmos serem considerados leves e mais aptos a serem implantados em hardwares, auxiliou Sopran (2016) a escolher o algoritmo que norteou seu trabalho. Outros critérios considerados para escolha do algoritmo foram sua atualidade considerando a data de publicação, disponibilidade de implementações em hardware e a garantia de corretude e eficiência.

Juels e Weis (2005), salientam também importância da implementação de algoritmos de criptografia leves quando se trata hardware. A exemplo, pode-se citar a implementação de algoritmos simétricos padrão como DES (do inglês *Data Encryption Standard*) e AES, em um sistema de etiquetas com a tecnologia RFID. Embora possível, a implementação implica em um esforço e um gasto inviável. Esse esforço, pode ser explicado relacionando o número de portas existentes e necessárias para a implantação do algoritmo de AES, em etiquetas de produtos. Enquanto uma etiqueta possui em média 2000 portas para uso, o AES implementado de forma leve necessita aproximadamente de 5000. Pode-se supor que a Lei de Moore acabará por permitir que algoritmos como o AES ou similares possam ser implementados de forma completa em dispositivos cada vez menores e simples. Contudo é de fundamental importância manter o máximo do recurso disponível no sistema para suas eventuais necessidades naturais.

Por dispor de características consideradas importantes como corretude e eficiência, e se enquadrar na categoria de algoritmo de criptografia leve, Sopran (2016) elegeu o algoritmo SIMON como a solução mais apta a ser implementada em hardware. Desta forma, optou-se por empregar o SIMON devido a essas características e para preservar e ampliar a linha de pesquisa nesse sentido.

2.4.2 SIMON

Os algoritmos criptográficos existentes foram, em sua maior parte, projetados para atender às necessidades da era da computação de mesa. Essa criptografia tende a não ser particularmente bem adaptada à era emergente da computação difusa, na qual muitos dispositivos altamente restritos de hardware e software precisarão se comunicar entre si (BEAULIEU *et. al.*, 2013).

Com base nesse contexto e visando atender essa demanda na área de segurança, em 2013 a agência de segurança nacional norte americana (NSA) disponibilizou dois novos algoritmos

de criptografia baseados em cifras em blocos. Esses foram denominados SIMON e SPECK. Ambos foram projetados de forma que seus algoritmos fossem leves, consequentemente fazendo com que se enquadrassem na categoria de algoritmos *lightweight*. Embora os algoritmos sejam considerados irmãos, seus propósitos são distintos, sendo o SIMON desenvolvido para ser implementado em hardware e o SPECK, por sua vez, para software. Este trabalho aborda particularmente o SIMON.

2.4.2.1 Algoritmo SIMON

Assim como a grande maioria dos algoritmos de criptografia em blocos da atualidade, o SIMON baseia-se em um modelo conhecido como cifra de bloco Feistel. Esse modelo utiliza essencialmente os seguintes passos para realizar a criptografia de uma mensagem de texto em claro;

- Dividir a entrada em duas partes iguais *Left* e *Right* (esquerda e direita);
- As duas partes sofrem processamento de *n* iterações;
- Cada iteração recebe *Left*-1 e *Right*-1 em suas entradas, elas são derivadas da iteração anterior, além disso, uma sub-chave K_i é derivada da chave inicial K;
- Através de um algoritmo gerador de chaves, sub-chaves distintas são geradas;
- Todas as iterações têm a mesma estrutura e são chamadas de round (rodadas);
- Para cada round, aplica-se uma função denominada *round function* (função de rodada), que envolve lógica XOR bit a bit.

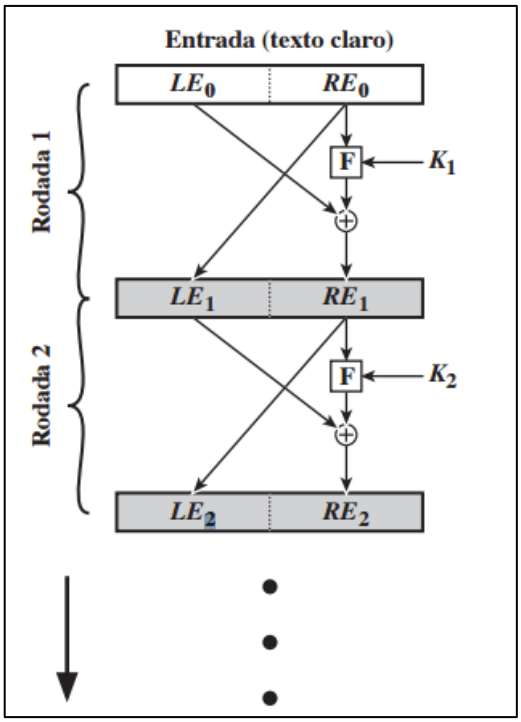

Figura 11. Modelo de cifra de bloco Feistel

Fonte: Stallings (2014).

O bloco de cifra do algoritmo SIMON é composto por uma palavra de *n* bits (portanto um bloco 2n-bits), este é denotado como Simon2*n*, onde *n* pode ser 16, 24, 32, 48 ou 64. Simon2*n* possui *m*-palavras chaves (*mn*-bit) que se refere a Simon2*n*/*nm*. Por exemplo, Simon64/128 refere-se à versão de Simon em blocos de texto simples de 64 bits, usando uma chave de 128 bits (BEAULIEU *et. al.*, 2013).

SIMON utiliza as operações XOR (⊕) bit a bit, AND (&) bit a bit e descolamento circular de bits (S_n) nas suas rodadas de encriptação e decriptação, onde L e R são as duas partes que compõem o texto plano (esquerda e direita respectivamente) e *k* é a chave da rodada.

A função de encriptação pode ser expressa como:

$$R(l, r, k) = ((S^1(l) \,\&\, S^8(l)) \oplus S^2(l) \oplus r \oplus k, l)$$

E a função de decriptação é expressa como:

$$R^{-1}(l, r, k) = (r, (S^1(r) \,\&\, S^8(r)) \oplus S^2(r) \oplus l \oplus k)$$

Para as operações de cifração e decifração são necessárias T rodadas, e consequentemente a geração das chaves para cada rodada a partir da expansão da chave inicial. Exceto a chave da rodada, todas as rodadas de Simon são exatamente as mesmas, e as operações são perfeitamente simétricas em relação ao mapa de deslocamento circular em n-bits palavras (COSTA *et. al.*, 2016).

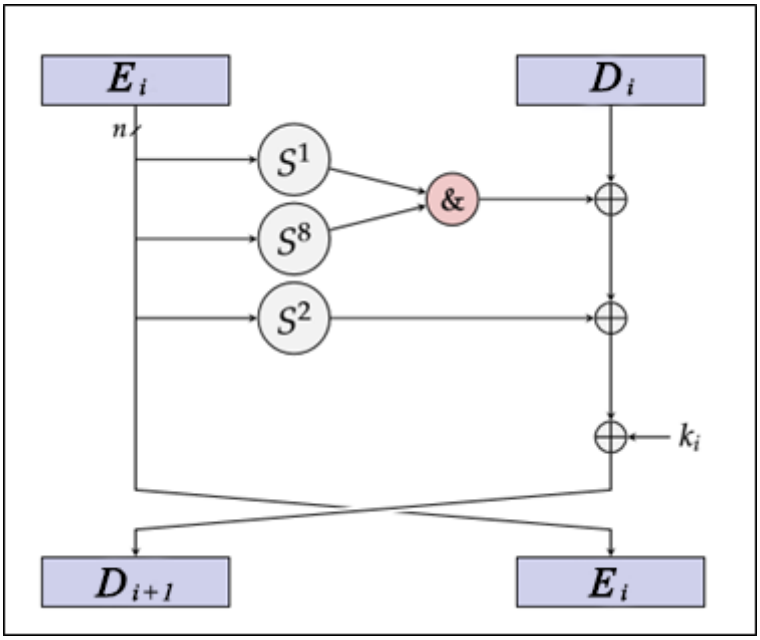

Figura 12. Função de rodada SIMON

Fonte: Beaulieu *et. al.* (2013).

Conforme apresentado na Figura 12, o algoritmo utiliza como base arquitetural de criptografia e descriptografia o mecanismo de Feistel com algumas modificações no interior da função de rodada. Entre essas alterações observa-se a inserção de três deslocamentos circulares representados como; S_1, S_2 e S_8. Esses utilizam como entrada o lado esquerdo da divisão em blocos no caso da primeira rodada. Nas demais, a entrada se refere a saída dos blocos direitos da rodada anterior. Durante a rodada ainda existem quatro operações além dos movimentos circulares, apresentadas no Quadro 1.

Quadro 1 - Operações de rodada do SIMON

Operação	Entrada 1	Entrada 2	Resultado
AND (&)	S_1	S_8	R1
XOR (⊕)	R1	D_i	R2
XOR (⊕)	S_2	R2	R3
XOR (⊕)	R3	Chave de Rodada (*k*)	D_{i+1}

Em relação ao número de rodadas (*T*) do algoritmo SIMON, sua escolha é baseada no tamanho do bloco de entrada (*n*). Os possíveis valores de *T* são apresentados no Quadro 2.

Quadro 2 - Parâmetros SIMON

Tamanho de bloco	Tamanho de chave	Subchaves de rodada (*m*)	Rodadas (*T*)
32 bits	64 bits	4	32
48 bits	72 bits	3	36
	96 bits	4	
64 bits	96 bits	3	42
	128 bits	4	44
96 bits	96 bits	2	52
	144 bits	3	54
128 bits	128 bits	2	68
	192 bits	3	69
	256 bits	4	72

Fonte: Adaptado de Beaulieu, 2013.

2.5 SIMULAÇÃO

Em relação a simulação do experimento, destacam-se duas ferramentas importantes; o ambiente de simulação RedScarf e a biblioteca SystemC, descritas a seguir.

2.5.1 RedScarf

O RedScarf, é um ambiente de simulação com ênfase na avaliação de desempenho de NoCs que permite a parametrização da rede, dos roteadores e dos modelos de tráfego via interface gráfica. Sua interface gráfica facilita o uso na configuração e visualização dos resultados de experimentos. O seu funcionamento é divido em dois blocos, *front-end e back-end*. O bloco *front-end* possui uma interface gráfica que permite o usuário definir o dimensionamento da rede, padrões de tráfego por nodo e configurar experimentos com diferentes alternativas arquiteturais. Outras responsabilidades do *front-end* é a geração dos modelos de arquitetura definida nos arquivos de configuração, além de invocar o *back-end* para realizar a simulação. Esse bloco também responsável por comunicar com *a back-end*, dessa forma, controla a simulação, captura os resultados e analisa o desempenho dos experimentos. Entre os recursos dispostos no *front-end* relacionados diretamente com o *back-end* destacam-se;

- Gerador de NoC: responsável por gerar os modelos da rede, as ligações dos roteadores, bem como, a parametrização do sistema e técnicas utilizadas;
- Gerador de SoC: responsável por gerar o controlador de parada da simulação e o topo da hierarquia do sistema;
- Gerador de tráfego: responsável por gerar o arquivo de configuração de trafego que fornece as informações dos fluxos que servem de estimulo para a simulação. Também realiza a validação das configurações de tráfego realizadas pelo usuário; e
- Analisador de resultados: responsável por ler os registros as simulações, coletar os dados e realizar os cálculos das métricas de desempenho.

O *back-end* possui modelos de hardware (em SystemC) dos componentes da arquitetura de comunicação e os de softwares dos terminais de instrumentação. Esses componentes variam entre modelos simples como implementação da porta lógica "AND" ("E" lógico) à modelos mais complexos, como geradores de fluxos, responsáveis por injeção de pacotes na rede. A construção da simulação no *back-end* e resultado das configurações e definições realizadas pelo usuário no *front-end*. Atualmente o RedScarf está limitado aos modelos arquiteturais e de geração de tráfego apresentados por Zeferino (2007), em que a rede utilizada é a SoCIN (SILVA, 2014).

Embora o simulador seja um software executável, o uso da biblioteca SystemC com implementação RTL (do inglês *Register Transfer Level*) faz com que o modelo reproduzido seja análogo ao hardware, provendo assim, acurácia adequada em relação a sua representação em hardware. Em relação a sua interface gráfica, foi desenvolvida em Qt, um *framework* C++ multiplataforma para desenvolvimento de aplicações com interface gráfica. Outra característica do RedScarf é ter suporte nas principais plataformas *desktop* (Windows, Linux e OS X) (SILVA, 2014).

2.5.2 SystemC

A segunda ferramenta que será utilizada para o desenvolvimento do projeto e a biblioteca de classes e macros SystemC desenvolvidas para linguagem C++. Essa biblioteca foi desenvolvida com o objetivo de auxiliar no desenvolvimento de projetos de componentes de hardware. Sua utilização permite realizar um alto nível de abstração no projeto de hardware. Dessa forma, possibilitando um aumento na produtividade e uma verificação precoce nos

possíveis erros através da criação modelos de componentes de hardware em sistemas de simulação (BLACK, 2004).

O SystemC implementa todas as estruturas necessárias para modelagem de hardware, propiciando construções que permitem aplicar conceitos de tempo, tipos de dados de hardware, hierarquia nas comunicações de estrutura e concorrência (BLACK, 2004).

A base do SystemC é um núcleo de simulação dirigido por eventos. Esse núcleo reage a eventos e realiza a respectiva troca das tarefas para execução. Essas tarefas são tratadas de forma genérica pelo simulador. Os demais elementos do SystemC são portas, módulos (para representar estruturas), interfaces (assinaturas que serão utilizadas por outros módulos ou canais) e canais que provém abstração na comunicação dos componentes da simulação. Junto ao núcleo ainda existem tipos de dados definidos na biblioteca SystemC para bits, vetores de bits, inteiros de precisão arbitraria e outros tipos. Além de alguns tipos comuns às linguagens de descrição de hardware, com possibilidades de utilizar tipos personalizados usando a sintaxe adequada no C++ (CANTANHEDE, 2007).

O seu funcionamento é baseado em blocos, conhecidos como módulos na linguagem. Esses incorporam as implementações dos processos (métodos). Os módulos se comunicam através de portas de entrada e saída, podendo conter variáveis que atuam como registradores do sistema e outros módulos em seu interior utilizando-os de foram hierárquica (CANTANHEDE, 2007).

Por fim, os módulos podem ser categorizados em dois tipos, método e linha de execução (*thread*), categorizando assim o tipo do processo em seu interior. Em relação ao tipo método, observa-se que ele não termina ao chegar no fim da execução. Essa característica ocorre por conta de sua execução ser estimulada por uma lista sensitiva de sinais, assim a cada alteração nos sinais ligados ao método, o mesmo excuta o procedimento programado. Já o processo baseado em *thread,* executa de forma sintonizada com o simulador, possibilitando assim utilizar recursos de tempo para atrasar sua execução (CANTANHEDE, 2007).

2.6 TRABALHOS RELACIONADOS

Esta seção é destinada a relacionar os trabalhos já realizados que possuem tema em comum com a presente dissertação. Para melhor clareza das informações apresentadas nessa seção, a mesma foi dividida em duas subseções.

2.6.1 Segurança em redes-em-chip

Gebotys e Gebotys (2003) foram os primeiros a presentar uma solução de segurança para redes-em-chip com o uso de criptografia de chave simétrica, com a proposta de evitar acesso não autorizado no conteúdo que trafega pela rede. O mecanismo consiste em proteger os núcleos com uma camada de rede, sendo que os núcleos possuintes dessa camada são conhecidos como *Secure Core*. Também foi implementado um núcleo centralizado para armazenar as chaves do sistema, o qual é denominado *Key-keeper*. O funcionamento do sistema consiste em, antes de cada comunicação entre núcleos (*Secure Core*) sua interface de rede solicitar ao *Key-keeper* a sua chave privada, a qual está armazenada de forma segura na memória. A comunicação entre a interface de rede e o núcleo *Key-keeper* é igualmente criptografada, contudo com uma chave armazenada no próprio núcleo. Após o armazenamento da chave a comunicação é feita através de criptografia de chave pública privada. A implementação dessa solução demonstrou ser necessário um baixo custo para sua viabilização. Outro aspecto apresentado na análise da solução foi o baixo impacto no desempenho que não superou os 11%.

Fiorin, Palermo e Silvano (2008) sugeriram como método de segurança o uso de monitoramento de trafego da rede, assim podendo identificar a extração de dados críticos ou a existência de geração de dados inúteis, a fim de desperdiçar largura de banda e causar uma maior latência nas comunicações intrachip, e/ou até a saturação da rede. A solução proposta foi realizar uma coleta de dados à nível de interface de rede e enviar para uma unidade central (NSM - Network Security Manager). Essa é responsável por neutralizar as ações do atacante. A composição da arquitetura agrega dois tipos de sondas responsáveis por detectar situações adversas diferentes. A primeira sonda é denominada de *Illegal Access Probe* (IAP), e é responsável por detecta tentativas não autorizadas de acesso a locais de memória, e a segunda é conhecida como Dos Probe (DiSP) e tem a tarefa de detectar um comportamento não natural de tráfego. A solução foi implementada por prototipação usando a tecnologia 0.13µm HCMOS9GPHS da STMicroelectronics. Após a implementação ouve uma comparação entre as versões com e sem o monitoramento com relação aos sobrecustos de área e de energia. Os resultados mostraram que o sobrecusto da implementação da sonda IAP na NI (do *inglês Network Interface*) não é significativo (0,02% em área) e o da sonda DoSP aumenta linearmente com a inclusão de configurações de entrada. O sobrecusto total é de 34,7% se comparado com a solução sem a monitoração de segurança.

Baron (2013) propôs um mecanismo para proteger a rede-em-chip SoCIN contra ataques de DoS (ataques de negação de serviço). Essa solução foi implementada como componentes de hardware, na forma de *wrappers*, que realizam a filtragem dos fluxos de comunicação injetados na rede, descartando pacotes que comprometam a sua disponibilidade ou regulando a taxa de injeção de fluxos que pretendam consumir uma largura de banda maior do que a que foi prevista pelo projetista do SoC. O mecanismo foi implementado em VHDL (do inglês *VHSIC Hardware Description Language*) e SystemC. A versão descrita em VHDL foi realizada utilizando o ambiente de desenvolvimento Quartus II versão 12.0 da Altera e sintetizada para a tecnologia de dispositivo lógico programável FPGA (Field Programmable Gate Array). Posteriormente foi sintetizada para uma tecnologia ASIC (Application Specific Integrated Circuit), usando a ferramenta Design Compiler da Synopsys . Para auxiliar nos testes e simular a implementação, uma descrição em SystemC foi realizada utilizando a versão do conjunto de bibliotecas C++ SystemC 2.3.0 e o BrownPepper com o objetivo de validar e avaliar a operação dos wrappers quando integrados à rede SoCIN.

Silva (2015) implementou um mecanismo para assegurar as propriedades de segurança de confidencialidade e autenticidade em sistemas baseados na rede-em-chip SoCIN, através da introdução do algoritmo de criptografia simétrica AES nas comunicações da rede. Para realização do experimento uma plataforma MPSoC de referência baseada na rede SoCIN foi implementada em lógica programável para viabilizar experimentos de avaliação da eficácia dos mecanismos de segurança em um sistema sintetizado em FPGA. Em relação aos resultados obtidos, o experimento demostrou eficácia contra ataques e ocasionou um impacto aceitável no desempenho da rede, juntamente com um sobrecusto equivalente comparado com trabalhos similares.

Frey e Yu (2017) analisaram o aumento no número de ameaças provocadas através de modificações maliciosas em hardware. Essas modificações, são conhecidas como hardware trojans (HT) e causam preocupações em relação a segurança do processador. Sendo que a grande maioria dos esforços para deter essas ameaças não são voltadas a sistemas que utilizem redes-em-chip como arquitetura de comunicação, os autores propuseram um mecanismo de verificação de integridade a nível de flit entre as comunicações dos roteadores da rede. Em relação aos tipos de ataques, foram considerados; (i) modificação no tipo do flit, (ii) alteração do endereço de destino de pacotes para endereços não autorizados e (iii) sabotagem na integridade de um pacote. Para a escolha dos tipos de ataques foi considerado que o principal alvo de um HT na rede é a largura de banda. Ao final do experimento os resultados apresentaram melhora no número de pacotes recebidos em até 70,1% em relação aos outros métodos de

segurança contra HT que controlam o endereço de destino do pacote na NoC. A disponibilidade média de link apresentou 43,7% maior que a dos métodos existentes.

2.6.2 Implementações do SIMON

Wetzels e Bokslag (2015) implementaram o algoritmo SIMON64/128 na plataforma Xilinx Spartan-6 FPGA series com diferentes tipos de arquitetura em circuitos combinacionais. As arquiteturas utilizadas para implementação foram: *round function*, arquitetura iterativa, arquitetura de desdobramento de laço, arquitetura *inner-round pipelining*, arquitetura *outer-round pipelining* e arquitetura *mixed inner-outer-round pipelining.* Para se obter uma análise das arquiteturas em sua forma natural, nenhuma otimização foi realizada nas implementações. Outro aspecto não considerado nas implementações foi segurança, assim nenhum mecanismo para evitar falha ou ataques foi desenvolvido. Ao final, os autores destacam, que devido ao prazo de projeto, não foi possível realizar a validação correta das arquiteturas *inner-round pipelining* e *mixed pipelining*, considerando assim os resultados obtidos como parciais. Em relação aos resultados apresentados, a arquitetura que obteve um maior resultado no aspecto de vazão (Mbits/s) foi o desdobramento de laço.

Costa *et. al.* (2016) realizou uma pesquisa, implementação e a comparação de dois algoritmos de criptografia de cifras de blocos leves Simon e Speck em relação a área e desempenho quando projetados em FPGA. Foram adotadas diferentes arquiteturas para estes algoritmos, sendo que em uma primeira implementação é considerada a menor área (SSR), segunda possui maior frequência (SFR) e a terceira possui menor frequência (AFR). Em relação a implantação, ambos os algoritmos foram desenvolvidos em VHDL com o tamanho de entrada de 32 bits e uma chave de 64 bits (SIMON32/64 e SPECK32/64). Para as operações de cifragem e decifragem, os números de rodadas foram definidos com 32 rodadas para o algoritmo SIMON e 22 rodadas para o SPECK. Como resultado foi possível comparar os dados estatísticos dos algoritmos Simon e Speck. A implementação do algoritmo Simon utilizando SFR, tem uma relação de área/vazão de 1,95, o qual é 271,79% melhor do que Speck com 5,3. A arquitetura de Speck (SSR), tem relação de área/vazão de 4,24, o qual é 26% melhor que a versão Simon com 5,37. A arquitetura de Speck com AFR tem uma relação de área/vazão de 0,55, o qual é 365% melhor do que Simon com 2,01.

Sopran *et. al.* (2017) desenvolveu e analisou o custo e desempenho da técnica de criptografia SIMON. A técnica foi implementada em FPGA abordando diferentes plataformas. O mesmo algoritmo foi implementado em software, hardware e em uma abordagem hardware-

software. A abordagem de software foi implementada utilizando a linguagem C, e executada em uma versão sintetizada do processador LEON3 em FPGA. Os blocos em hardware foram descritos em VHDL e conectados ao barramento do processador. Para realizar a implementação particionada foi escolhido o procedimento de geração de subchaves para ser executado em software e o procedimento de rodada para ser executado em hardware, em decorrência da estrutura sequencial do algoritmo. Para avaliação de resultados, foram gerados relatórios de síntese realizados na ferramenta Quartus II e *debugging* SignalTap II. Em relação aos resultados obtidos, a implementação particionada manteve o equilíbrio entre custo, desempenho e consumo.

3 DESENVOLVIMENTO

Esse capítulo se divide, na visão geral do desenvolvimento e nas implementações realizadas, para que o objetivo geral do trabalho fosse alcançado. Primeiramente, é apresentada toda a parametrização necessária para o funcionamento correto da rede. Também e apresentado o diagrama de sequência referente a rede com a implementação do SIMON. A partir dessas definições, é demonstrada a implementação da criptografia na rede. Ainda, na implementação, são apresentadas todas as modificações necessárias na rede SoCIN, para que a mesma possa suportar o SIMON nas suas comunicações.

3.1 VISÃO GERAL

Para desenvolvimento do trabalho foi utilizado o ambiente de simulação RedScarf. Já o algoritmo SIMON foi adaptado de Sopran (2016) e revisado com base no artigo de publicação da cifra. Quanto a versão da rede-em-chip, foi utilizada a SoCINfp com o roteador ParIS em modelo dinâmico, possibilitando assim a parametrização no número de roteadores e portas via vetores, o que evita a geração de código de compilação no processo de simulação.

A topologia da rede que foi utilizada como base para o desenvolvimento é representada na Figura 13. Nela é ilustrada a topologia de grade 2D (2D-Mesh) 3x3, na qual cada CPU (núcleo) possui uma interface de rede (NI - *Network Interface*), que realiza a comunicação entre o núcleo e o roteador.

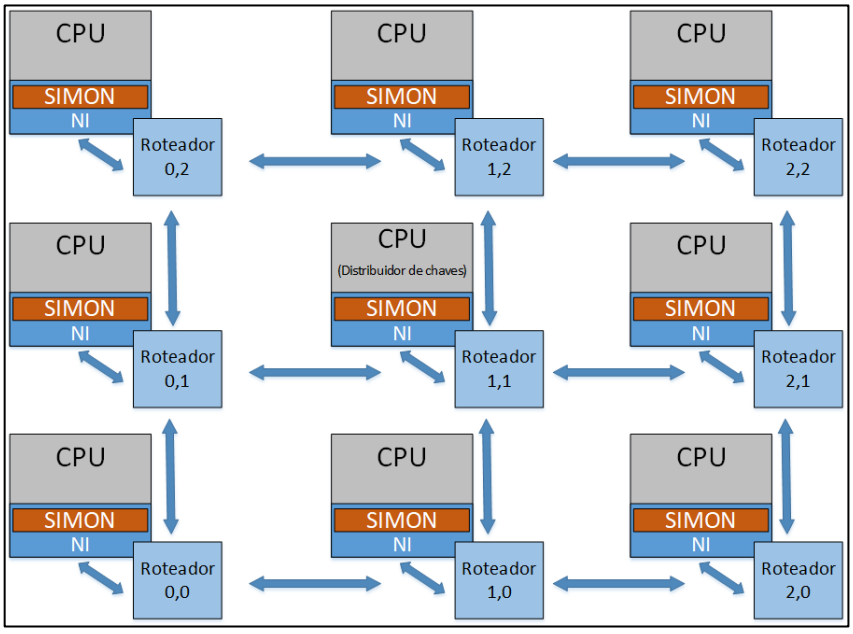

Figura 13. Topologia da rede para desenvolvimento

Em relação à implementação do algoritmo SIMON nas comunicações da rede, cada processador recebeu, em sua interface de rede, uma parte do algoritmo responsável por criptografar e descriptografar as mensagens, conforme ilustra a Figura 13. Com relação à geração de chaves, o roteador mais centralizado da rede, que está localizado na posição [1,1] na grade, foi conectado localmente com o distribuidor de chaves. Este realiza a entrega das chaves entre os núcleos envolvidos nas trocas de mensagens. Tal modelo é apresentado na Figura 13. Sua interface de rede também recebe a criptografia.

Com relação ao tamanho dos blocos para criptografia, foi implementado o SIMON32/64. Essa definição significa que o tamanho máximo dos blocos de entrada é de 32bits e sua criptografia emprega uma chave de 64bits. Dessa forma, a comunicação da rede também se limita a mensagens com tamanho fixo em 32bits.

A aferição da implementação do algoritmo SIMON, foi realizada por meio dos testes de parâmetros predefinidos na documentação oficial, inclusa no Anexo A.

A comunicação entre os núcleos segue o diagrama de sequência apresentado na Figura 14.

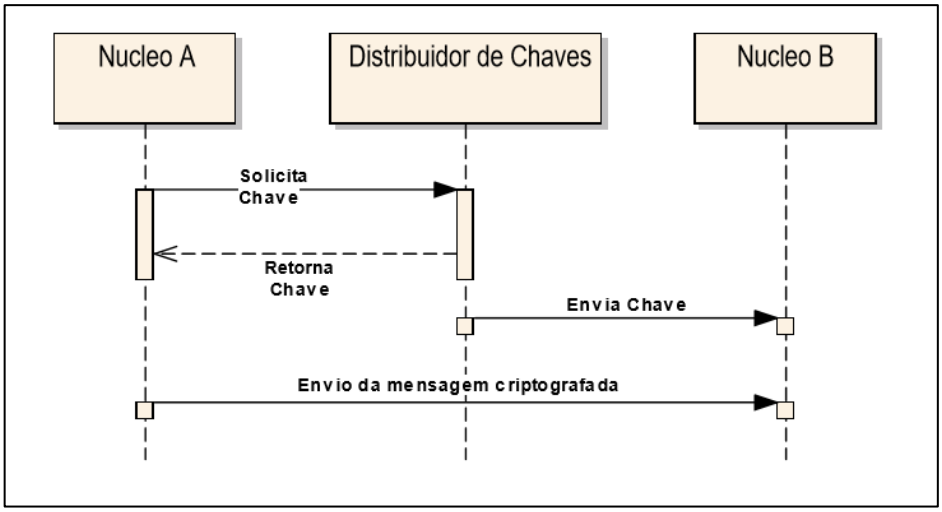

Figura 14. Diagrama de sequência da comunicação

O núcleo que deseja transmitir informações, neste exemplo, o Núcleo A, solicita uma chave ao Distribuidor de Chaves. O Distribuidor de Chaves então realiza duas ações: Retorna uma chave ao núcleo solicitante e, em seguida, transmite a mesma chave ao núcleo que deve receber o tráfego, neste exemplo, o Núcleo B. Então, o Núcleo A pode escrever uma mensagem criptografada que será decriptografável exclusivamente pelo Núcleo B.

Cabe considerar que as transmissões originadas e/ou destinadas para o Distribuidor de Chaves são criptografadas empregando-se a chave pré-compartilhada exclusiva de cada um dos

núcleos, o que impede quaisquer núcleos intermediários de compreendê-las mesmo que venham a transportá-las em seus roteadores.

3.2 IMPLEMENTAÇÃO

Dentre as alterações realizadas nos componentes da rede, destacam-se as modificações realizadas nos seguintes componentes; gerador de tráfego denominado *Flow Genereator* (FG), o *Terminal Instrumentation* (TI) responsável por conter a FG e realizar a sua ligação com a roteador e consequentemente, com a rede e na classe *Parameters* responsável por carregar e armazenar todas as definições iniciais da rede, esses estão no modulo responsável por realizar as simulações.

O FG, embora não realize nenhum processamento de dados, atua na rede sob o papel de um núcleo do processador. Assim, suas funcionalidades se resumem em gerar, enviar e receber dados. Em relação aos dados gerados, o FG realiza a criação dos pacotes, montados conforme a parametrização inicial da rede. Cada pacote possui necessariamente um *flit* de cabeçalho (*header*) e um *flit* de cauda que representa o fim do pacote (*trailer*). O pacote possui também uma quantidade de *flits* de corpo (*payload*), responsáveis por armazenar as mensagens do pacote. Sua quantidade é definida também na parametrização inicial.

As informações que compõem cada cabeçalho de pacote são; o tipo do *flit*, uma *flag* de informação, utilizada para casos de cabeçalhos estendidos e os endereços de origem e destino (Figura 15). Para identificar o tipo do *flit*, é necessário preencher os dois bits mais significativos, denominados BOP (do inglês *Begin-of-Packet*), que identifica o início do pacote, e EOP (do inglês *End-of-Packet*), que identifica o fim do pacote e, é preenchido somente no último *flit*. Em relação à informação de extensão do cabeçalho, ela é informada no terceiro bit mais significativo do primeiro *flit* do pacote, e serve para informar que o próximo *flit* ainda faz parte do cabeçalho. Por fim, as informações de origem e destino estão nos 16 bits menos significativos do *flit*. Essa estrutura de formação do *flit* de cabeçalho não necessitou ser alterada para que o SIMON fosse implementado na rede.

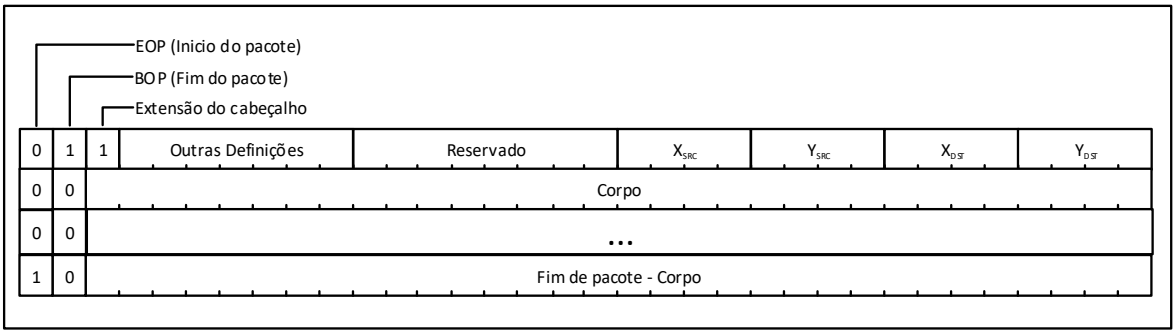

Figura 15. Estrutura dos flits do pacote.

Com a implementação do SIMON na rede, o FG passou a poder se comportar de duas formas distintas, sendo elas: gerador de fluxo e distribuidor de chaves. O distribuidor de chaves é responsável pela geração e o envio das chaves de criptografia para os elementos comunicantes da rede. Dentre os FGs que compõem a rede, apenas um pode se comportar como distribuidor de chaves. Essa definição é realizada nos parâmetros iniciais da rede.

O FG, quando se comporta como distribuidor de chaves, realiza a criação, o envio e o recebimento dos pacotes de forma que possa atender as solicitações dos componentes. Essas solicitações resumem-se em receber uma solicitação de chave, gerá-la e enviá-la aos elementos comunicantes.

Quanto ao recebimento da solicitação, o FG distribuidor verifica qual núcleo solicitou a chave, extraindo essa informação da origem do pacote solicitante. Posteriormente, por análise do corpo do pacote, o distribuidor verifica com qual núcleo o solicitante deseja se comunicar, essa informação permanece criptografada até chegar no distribuidor. Assim que for identificado o fim do pacote com o recebimento do último *flit*, o *trailer*, o FG realiza o envio de dois pacotes, um para o solicitante e outro para o destinatário, conforme representado na Figura 16. Esses pacotes possuem a mesma estrutura de um pacote comum, porém com o número de *flits* de corpo limitados em dois. Essa limitação é proveniente do tamanho dos *flits* em relação à chave. A chave possui 64 bits e a rede troca mensagens de tamanho fixo de 32 bits. Por isso, é necessário que a chave seja particionada e enviada em dois *flits* distintos.

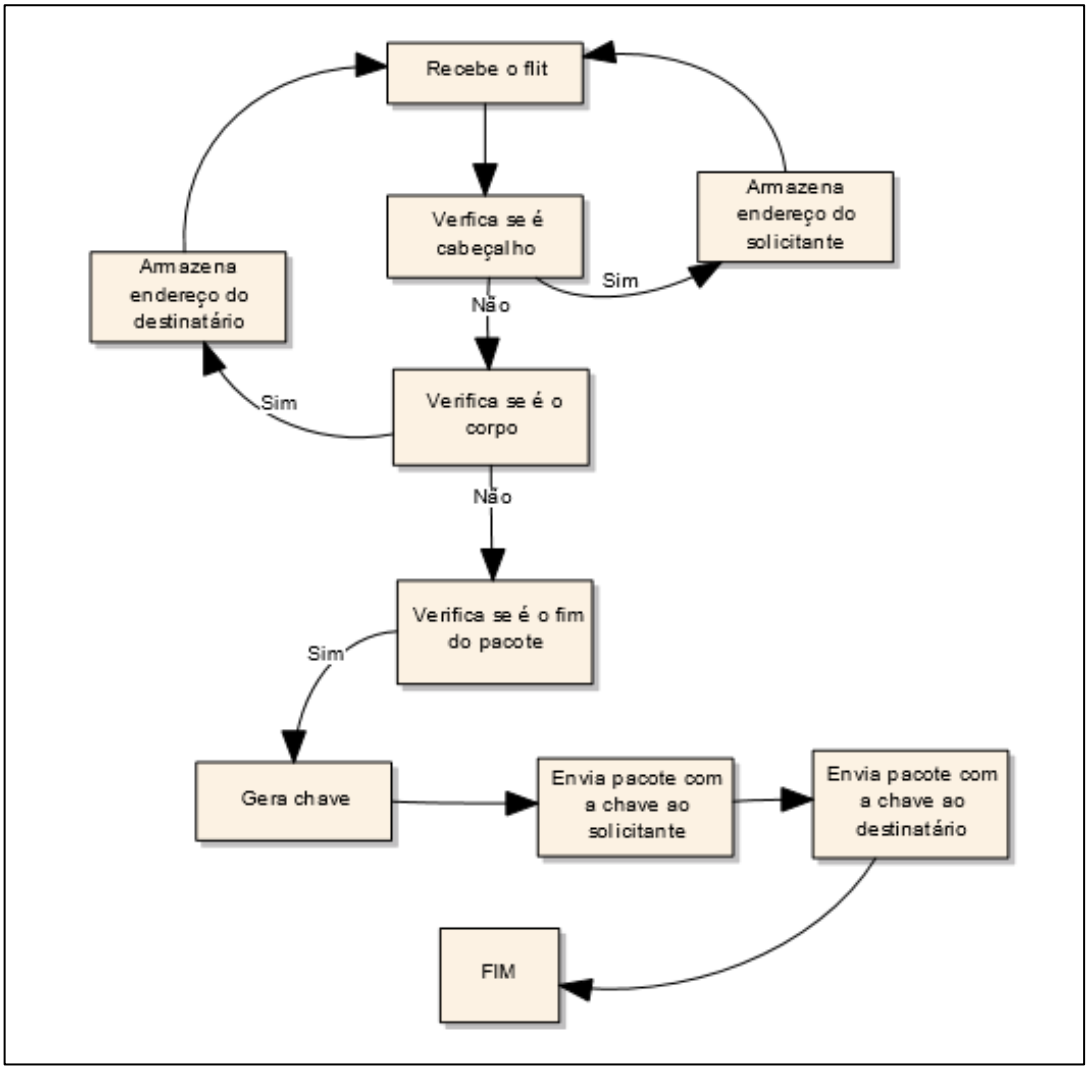

Figura 16. Sequência de recebimento de solicitação de chave.

Em relação ao envio das chaves, o distribuidor cria o pacote identificando no primeiro *flit* (*header*) além das informações essenciais já citadas, o endereço do seu parceiro de chave. A Figura 17 representa um exemplo de pacotes enviados pelo distribuidor para os elementos comunicantes. Na figura, o Núcleo A solicitou uma chave ao distribuidor para se comunicar com o Núcleo B e, dessa forma, o distribuidor enviou dois pacotes, uma para o Núcleo A e outro para o Núcleo B. O corpo dos pacotes é composto apenas pela chave de criptografia, porém nos seus respectivos cabeçalhos existe o endereço do núcleo parceiro. Essa informação é utilizada para que o componente do SIMON do solicitante e do destinatário saibam onde armazenar a chave para utilizá-la posteriormente na criptografia e descriptografia.

Outra informação enviada pelo distribuidor de chaves, que é incorporada no cabeçalho do pacote de envio de chave é o endereço do núcleo solicitante da chave. Essa informação é

necessária para que o núcleo solicitante, saiba ao receber uma chave do distribuidor, que já está apto a enviar a informação para o núcleo destino.

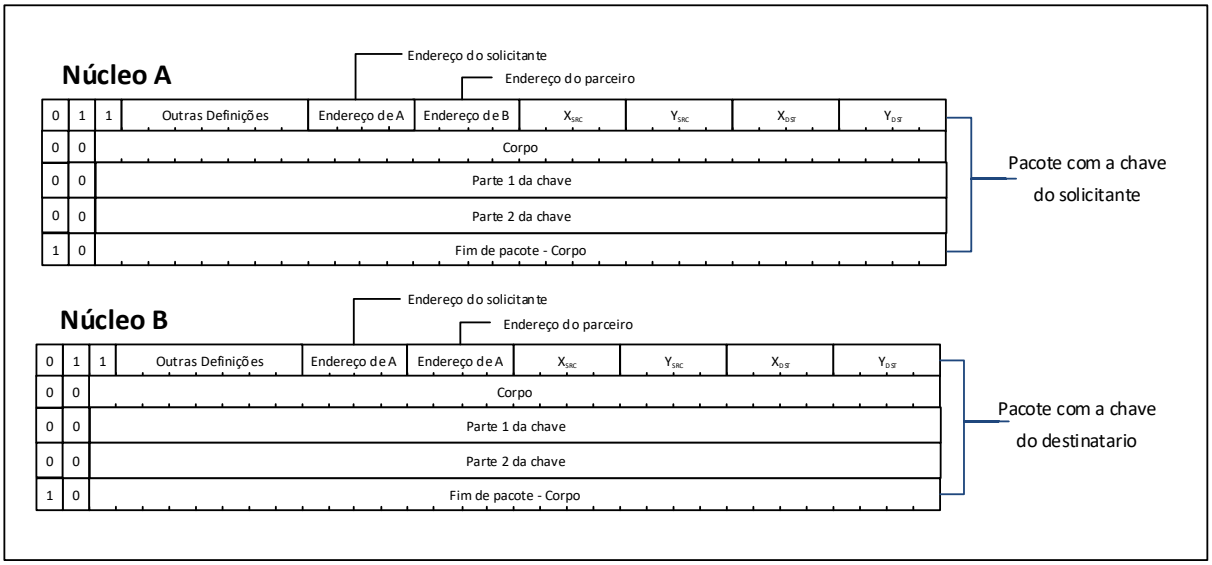

Figura 17. Exemplo de pacotes enviados pelo distribuidor.

Para realizar o envio de um pacote ao destinatário, quando o FG se comporta como gerador de fluxo, é necessário requisitar antes uma chave ao distribuidor. Essa requisição é feita através de um envio de um pacote de solicitação de chave para o distribuidor (Figura 18). Esse pacote é composto de um *flit de* cabeçalho, um *flit* de corpo no qual é informado o endereço do destino que se deseja comunicar e um *flit* de fim de pacote.

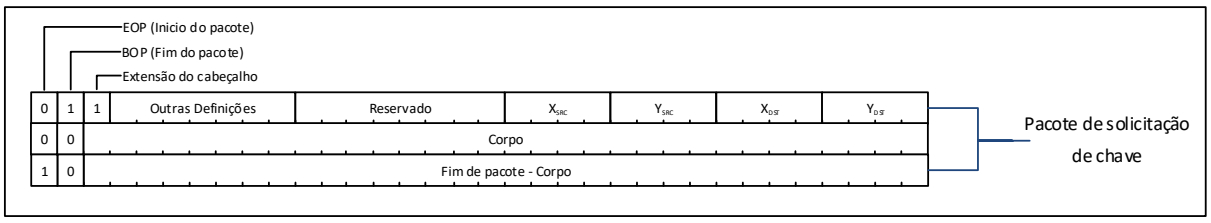

Figura 18. Representação de pacote de solicitação de chave.

Após a solicitação ser concluída, o pacote com a mensagem é escalonado em uma fila esperando a chegada da chave solicitada e, então, assim que recebida, o mesmo pode ser enviado para o destinatário.

Com relação à classe *Parameters*, dois novos parâmetros foram adicionados:

1. USE_SIMON: Identifica se as comunicações da simulação serão criptografadas ou não; e
2. DISTRIBUTOR_KEY_POS: Define qual a posição do distribuidor de chaves na rede.

Esses parâmetros foram incluídos com o intuito de tornar a simulação mais flexível e facilitar a obtenção de resultados comparando o desempenho da rede com e sem criptografia e com variações na posição do distribuidor. Vale ressaltar que nenhuma adaptação foi realizada, para que esses parâmetros pudessem ser alterados via interface da simulação.

Quanto à implantação da criptografia na rede, fez-se necessária a criação de um novo módulo especifico para seu funcionamento, em virtude de se tentar aproximar-se, tanto quanto o possível, de futuros cenários reais. Esse módulo é o responsável por realizar as operações de criptografia, descriptografia e armazenamento das chaves. Em relação ao posicionamento desse módulo, foi considerado que o mesmo atuasse no interior da interface de rede, fazendo um papel de filtro de entrada e saída. Por conta da rede SoCIN não possuir uma interface de rede nativa, a inclusão do módulo do SIMON está localizada após as FIFOs de entrada e saída do FG. Dessa forma, o fluxo de dados é controlado na entrada do SIMON.

A Figura 19 exibe o módulo do SIMON incorporado à rede. Ele possui duas entradas e duas saídas. Em suas funções de criptografia e decriptografia, o SIMON utiliza apenas recursos combinacionais. Assim, não necessita de um ciclo de *clock* para realizar as operações e não requer nenhum tipo de controle.

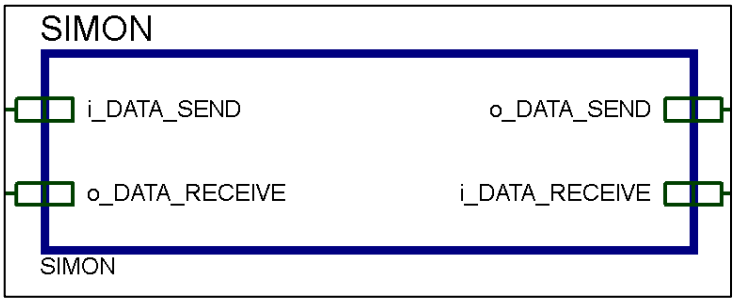

Figura 19. Módulo do SIMON

Cada núcleo possui apenas um módulo do SIMON. Essa configuração foi escolhida por conta do compartilhamento do vetor de chaves. Esse vetor é o responsável por armazenar as chaves de criptografia e sua estrutura pode ser vista na Tabela 3 - .

O preenchimento dos vetores de chave pode ser efetuado em dois momentos distintos: na criação do módulo ou durante a simulação. A distinção no momento do preenchimento está relacionada ao comportamento do núcleo. Esse, quando se comporta como distribuidor de chaves, tem suas chaves carregadas no momento de sua criação na simulação, sendo essas chaves pré-definidas via código. Em relação aos núcleos não distribuidores, apenas a chave na posição do núcleo do distribuidor é preenchida. Assim todos os núcleos possuem uma chave pré-compartilhada com o distribuidor. Dessa forma, os vetores dos núcleos não distribuidores são preenchidos conforme as solicitações são feitas ao distribuidor.

Tabela 3 - Estrutura do vetor de chaves

Posição do Vetor	Chaves
0	Chave parceiro 1
1	Chave parceiro 2
2	Chave parceiro 3
3	Chave Distribuidor
4	Chave parceiro 5
5	Chave parceiro 6
6	Chave parceiro 7
7	Chave parceiro 8
8	Chave parceiro 9

A Figura 20 exibe os módulos originais da SoCIN em azul e as adições propostas por este trabalho em verde, especificamente nas ligações internas do módulo *Terminal Instrumentation*. Esse módulo foi o local em que o projeto atual realizou todas as alterações e inclusões necessárias para que a rede pudesse criptografar e descriptografar as mensagens com o SIMON. Como pode ser observado na Figura 20 (b) em comparação à Figura 20 (a), além da implementação do módulo do SIMON, fez-se necessário acrescentar mais uma estrutura de fila (FIFO). Dessa forma, foi possível controlar a sequência de saída de dados na rede na ordem como foi gerada. Logo, o fluxo de saída e de entrada ficou da seguinte forma:

- Saída: FG → FIFO → SIMON → FIFO → Rede;
- Entrada: Rede → FIFO → SIMON → FG.

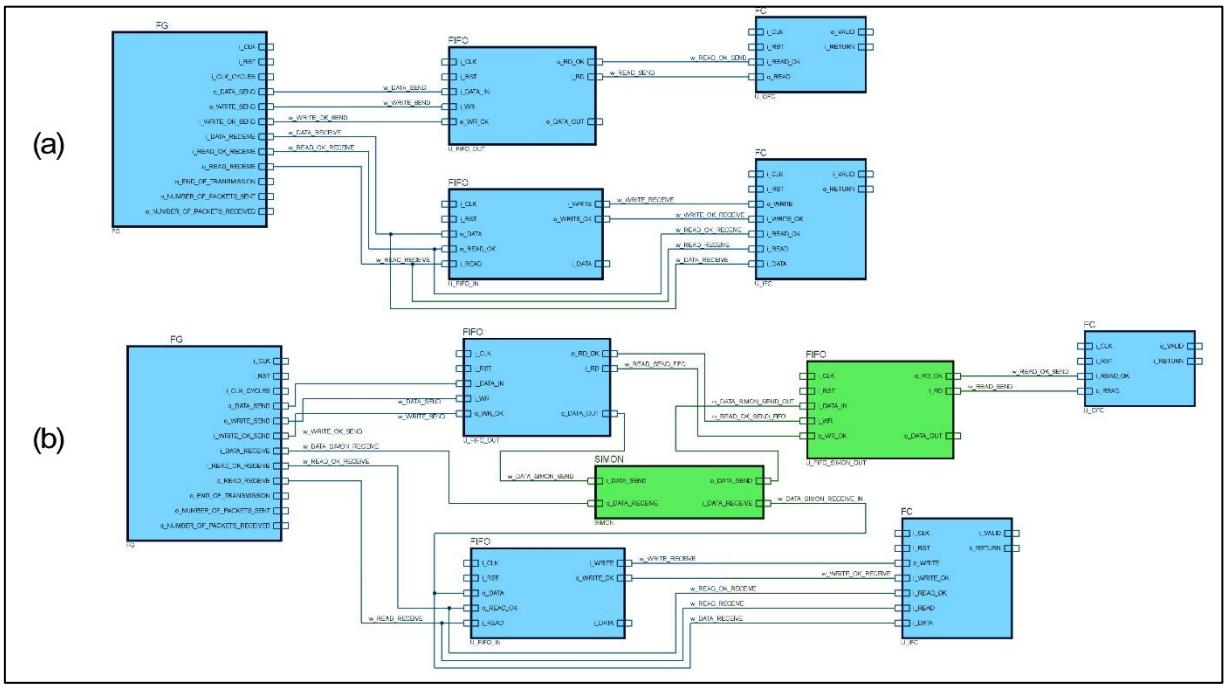

Figura 20. Modelo *Terminal Instrumentation.* (a) sem SIMON; (b) com SIMON.

3.3 CONSIDERAÇÕES

Este capítulo apresentou as alterações e parametrizações realizadas para portar o algoritmo SIMON para dentro da rede SoCIN. O próximo capítulo apresenta os resultados obtidos em comparação entre os modelos de rede com e sem a implementação do SIMON.

4 RESULTADOS

Este capítulo apresenta os resultados obtidos com a execução dos experimentos propostos. Inicialmente, são apresentadas as parametrizações definidas para realizar ambas as simulações, com e sem a presença do SIMON na rede. Foram realizadas comparações entre os resultados considerando latência de comunicação, número de pacotes enviados, topologia, algoritmo de roteamento e forma de onda.

No Quadro 3 - são apresentados os parâmetros que foram usados como referência para a realização dos testes. Além desses parâmetros já citados, outras definições serão ajustadas seguindo a proposta de testes iniciais, sendo elas:

- 9 elementos organizados em uma estrutura 3x3;
- Tamanho da mensagem de 32 bits;
- Largura de banda requerida de 320 Mbps; e
- Duas simulações para cada combinação de parâmetros: com e sem criptografia.

Quadro 3 - Parâmetros de referência para simulação

Testes	Topologia	Roteamento	Árbitro	Número de pacotes	Núcleo Distribuidor de chaves
Distribuidor centralizado	2D-Mesh	XY	Round-Robin	1	4
	2D-Mesh	XY	Round-Robin	3	4
	2D-Torus	DOR	Round-Robin	1	4
	2D-Torus	DOR	Round-Robin	3	4
Distribuidor na extremidade	2D-Mesh	XY	Round-Robin	1	6
	2D-Mesh	XY	Round-Robin	2	6
	2D-Torus	DOR	Round-Robin	1	6
	2D-Torus	DOR	Round-Robin	2	6
Distribuidor entre as extremidades	2D-Mesh	XY	Round-Robin	1	7
	2D-Mesh	XY	Round-Robin	3	7
	2D-Torus	DOR	Round-Robin	1	7
	2D-Torus	DOR	Round-Robin	3	7

Quanto aos testes, todos foram parametrizados com um intervalo de canais de 80MHz a 110MHz. Em relação ao método de parada da simulação, o mesmo foi configurado para a entrega de todos os pacotes.

Todos os experimentos realizados neste trabalho foram executados em um notebook com um processador Intel Core i5 operando a 2,20 GHz e com 4GB de memória RAM. O sistema operacional sobre o qual os testes foram executados foi o Windows 10 versão de 64 bits, com a instalação da biblioteca SystemC na versão 2.3.1a.

4.1 SIMULAÇÃO

Na elaboração de todas as simulações, foram realizadas comunicações entre apenas dois núcleos. Essa abordagem foi adotada para reduzir o escopo a seu cenário mínimo.

Para avaliar o impacto que a posição do distribuidor de chaves ocasiona no desempenho da rede, os testes foram realizados com o distribuidor em três posições distintas, apresentadas na Figura 21.

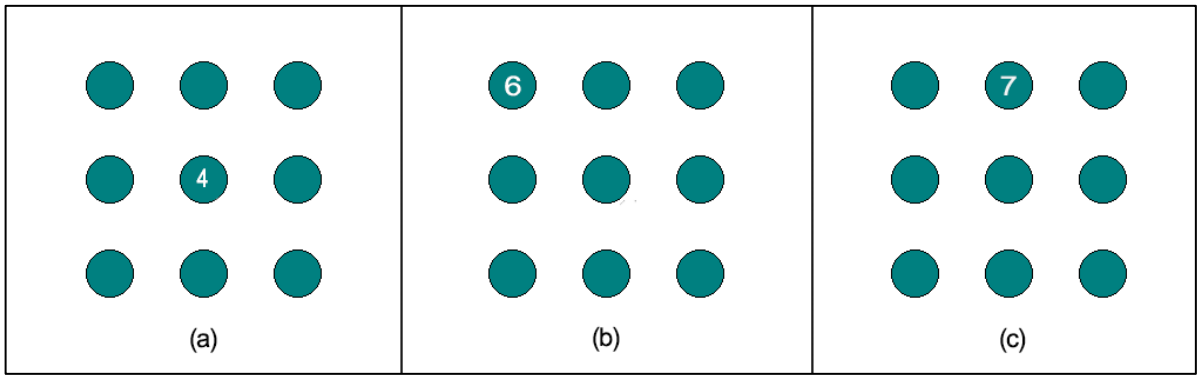

Figura 21. Posição do distribuidor nos testes: (a) Centralizado; (b) Extremidade; (c) Entre as extremidades.

A escolha dos núcleos envolvidos em cada teste foi variada de forma que seja testada em todas as distâncias possíveis. Assim, foram avaliadas quatro distâncias entre os núcleos envolvidos nas trocas de mensagens. Essas distâncias são representadas na Figura 22 e configuram-se em todas as distâncias possíveis com essa distribuição de núcleos. A relevância na avaliação entre as distâncias dos núcleos comunicantes se dá tendo em vista que, quanto mais longe um núcleo estiver do outro, mais saltos[1] serão feitos entre os roteadores da rede até que a mensagem chegue ao seu destino. Por exemplo, considerando a Figura 22(d), com mensagem partindo do núcleo 0 até o núcleo 8, com um roteamento XY, a mensagem saltará duas vezes para os roteadores da esquerda e mais duas vezes para cima, totalizando quatro saltos. Em comparação com Figura 22(b), é realizado o dobro de saltos, para a conclusão da comunicação entre os núcleos.

[1] Do inglês, *hop*, que, no escopo deste tema, designa o número de roteadores que distanciam dois comunicantes. O pacote “salta” por entre as interfaces de rede dos roteadores intermediários. (RFC1058).

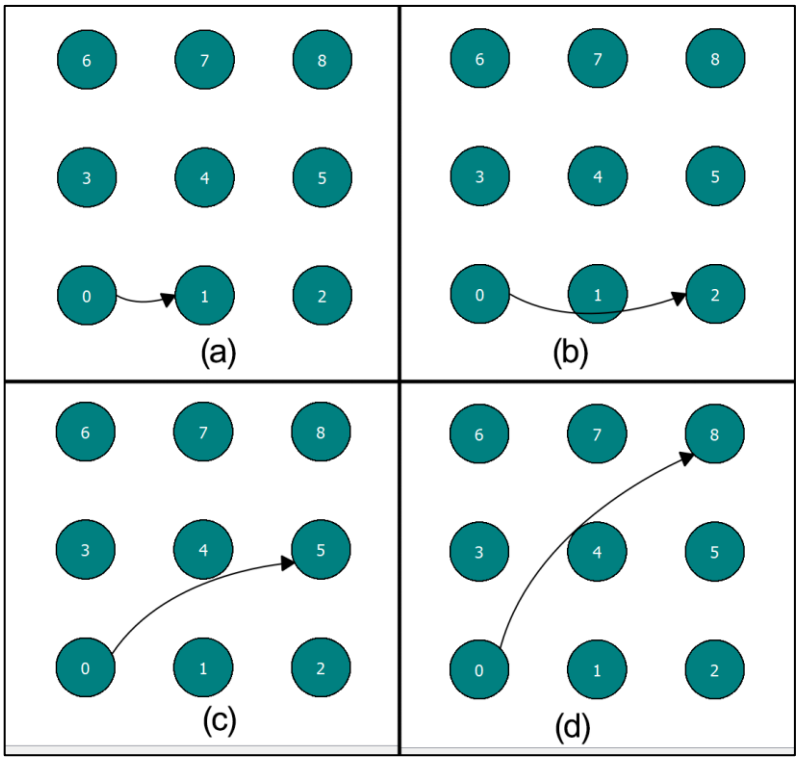

Figura 22. Distâncias de comunicação: (a) Um salto; (b) Dois saltos; (c) Três saltos ;(d) Quatro saltos.

4.1.1 Distribuidor centralizado

Nesta seção, são apresentados os resultados encontrados nos testes realizados com o distribuidor de chaves na posição mais centralizada da rede. Para realização dos testes, foram executadas duas simulações, sendo: uma com o envio de um pacote e outra com o envio de três pacotes.

Na Figura 23, são apresentados os resultados obtidos com o envio de um único pacote entre os núcleos comunicantes. Conforme ilustrado na Figura 23(a), que se refere ao envio de um pacote com um salto na rede, observa-se que a diferença de latência média medida em nanossegundos é maior, no caso da rede com criptografia. Esse aumento é aproximadamente de 134% com relação à mesma comunicação sem a presença do SIMON. Ressalta-se que, por conta da proximidade dos núcleos comunicantes e/ou pelo fato do distribuidor de chaves estar centralizado na rede, o tipo do roteamento não provocou alterações significativas nos resultados com a criptografia envolvida.

Com relação aos resultados no envio de três pacotes, que podem ser observados na Figura 24, não foram encontradas mudanças expressivas com relação aos envios de apenas um pacote. As diferenças aparentes estão no valor da latência média inicial das comunicações. Esse fato ocorre por conta do envio inicial simultâneo dos pacotes.

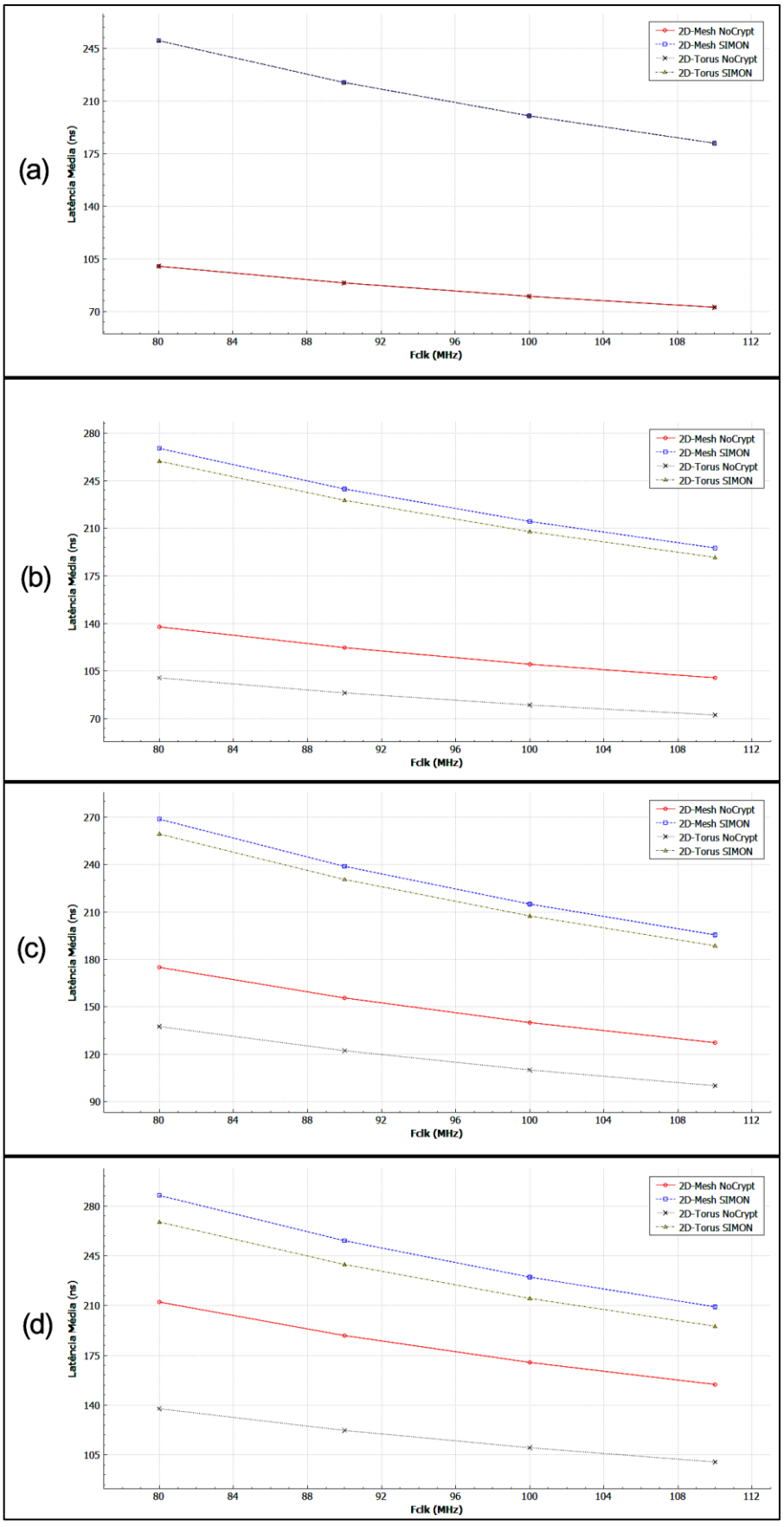

Figura 23. Latência x frequência com um pacote – distribuidor centralizado: (a) Um salto; (b) Dois saltos; (c) Três saltos; (d) Quatro saltos.

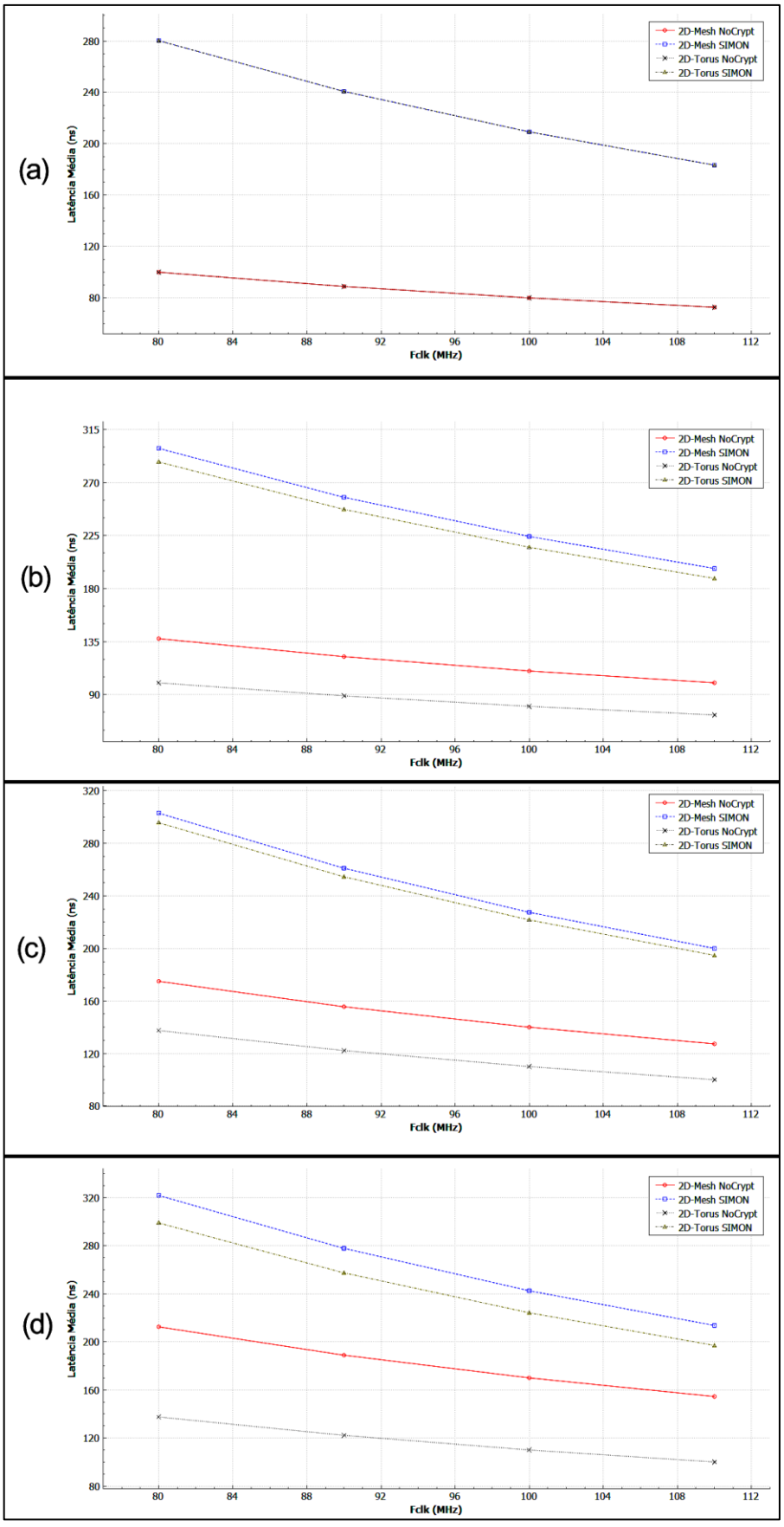

Figura 24. Latência x frequência com três pacotes – distribuidor centralizado: (a) Um salto; (b) Dois saltos; (c) Três saltos; (d) Quatro saltos.

4.1.2 Distribuidor na extremidade

Com o distribuidor posicionado em uma extremidade da rede, os resultados apresentaram uma variação na latência média inicial ao envio de um pacote. Essa variação é de aproximadamente 100 ns, quando enviado com apenas um salto para o destino e sem criptografia na mensagem e, 300 ns, quando enviado com dois saltos e com criptografa da mensagem. Ambos os resultados estão relacionados a topologia 2D-Mesh.

Comparando as trocas de mensagens criptografadas e considerando o roteamento, pode-se observar um aumento médio aproximado de 28%, quando a topologia está configurada como 2D-Mesh com relação a 2D-Torus.

Na Figura 25(d) e Figura 26(d), são apresentados os resultados obtidos com o envio de um e dois pacotes respectivamente, ambos efetuados com quatro saltos na rede. Esses apresentam dentre os resultados obtidos com o distribuidor na extremidade, as maiores diferenças entre as comunicações com e sem a criptografia. Em comparação, considerando as topologias, a rede com a topologia 2D-Mesh apresentou um aumento aproximado na latência media inicial de 42% quando as mensagens foram criptografadas, já na rede com topologia 2D-Torus esse aumento foi de aproximadamente 62%.

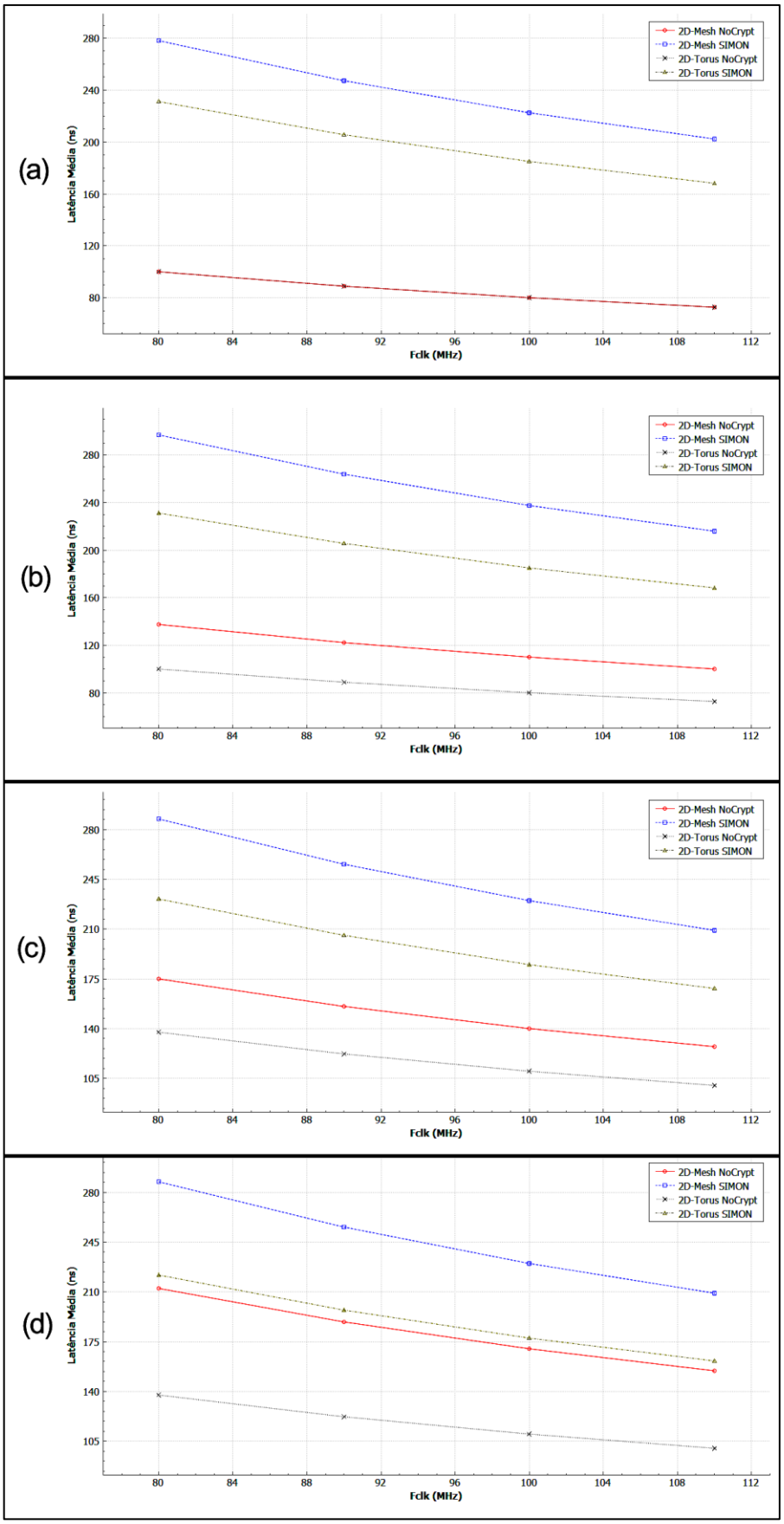

Figura 25. Latência x frequência com um pacote – distribuidor na extremidade da rede: (a) Um salto; (b) Dois saltos; (c) Três saltos; (d) Quatro saltos.

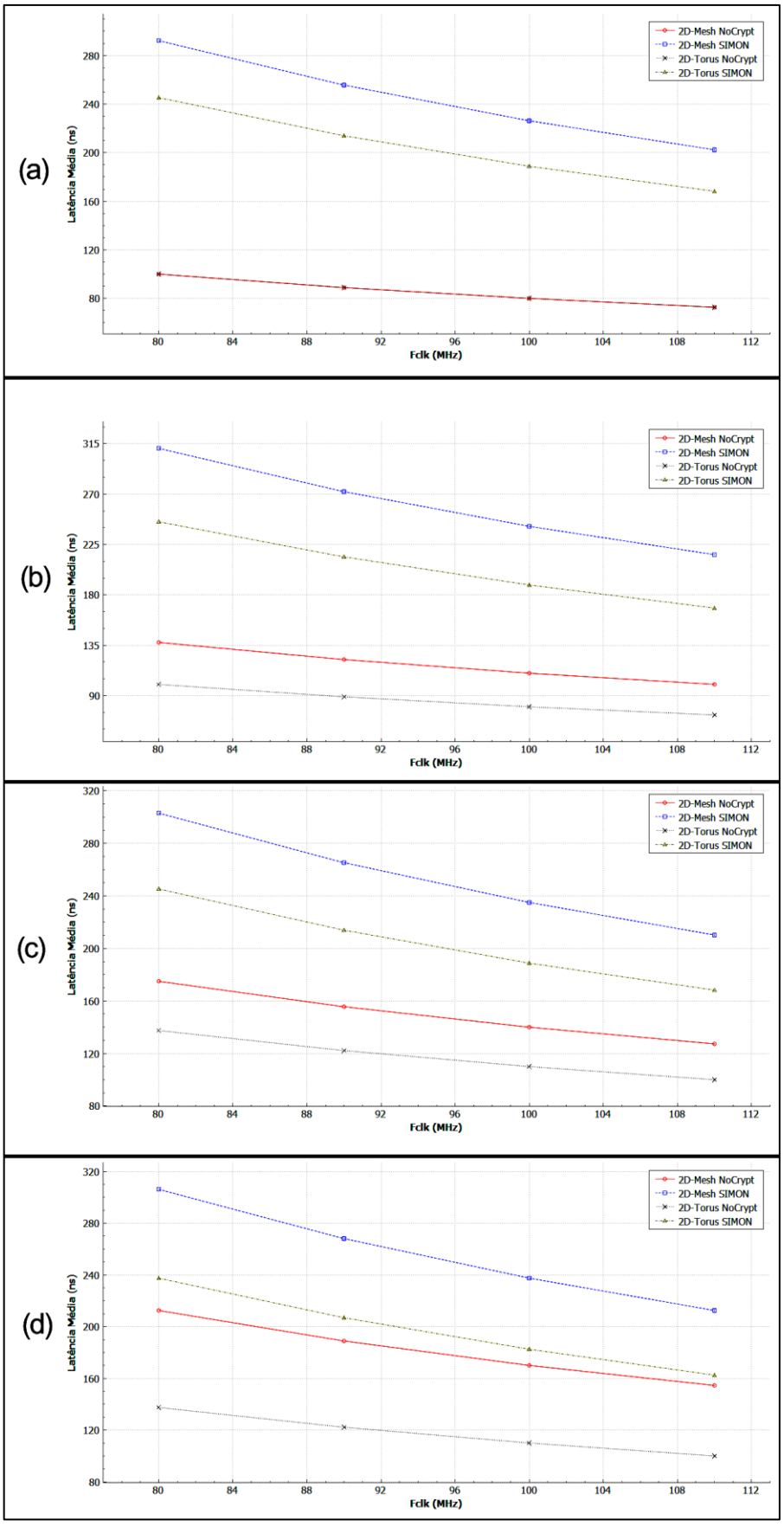

Figura 26. Latência x frequência com dois pacotes – distribuidor na extremidade da rede: (a) Um salto; (b) Dois saltos; (c) Três saltos; (d) Quatro saltos.

4.1.3 Distribuidor entre as extremidades

Na Figura 27 e na Figura 28, são apresentados os resultados obtidos com os testes executados com o posicionamento do distribuidor de chaves na posição entre as extremidades da rede. As possíveis posições para o distribuidor de chaves são: 1, 3, 5 e 7. Para a realização dos testes apresentados, a posição escolhida foi a 7. Essa escolha foi tomada por dois motivos: não está sendo utilizado para envio ou recebimento de pacotes em nenhum experimento, e por ser a possibilidade com maior distância das comunicações.

Os resultados apresentaram uma regularidade em relação à latência média entre as comunicações que realizam mais de um salto na rede. Assim, quando se compara a latência média inicial das comunicações criptografadas com apenas um salto em relação às que exigem mais saltos para serem concluídas. Pode-se perceber que houve um aumento aproximado de 12,5% com a topologia 2D-Torus e 9,7% com a topologia 2D-Mesh, isso considerando o envio de um único pacote. Quanto ao envio de três pacotes, apenas na topologia 2D-Mesh foram apresentados aumentos significativos na latência media inicial. Esse aumento foi aproximadamente de 9,3%.

Com relação aos modelos com e sem criptografia, constatou-se um aumento na latência média inicial de aproximadamente 115,8% quando a topologia utilizada foi a 2D-Mesh. Já com a topologia 2D-Torus, o aumento foi de aproximadamente 125,5%. Esses resultados foram obtidos quando enviados apenas um pacote. Quanto ao envio de três pacotes, os aumentos foram: 132,5% com a topologia 2D-Mesh, e, 157%, com a topologia 2D-Torus.

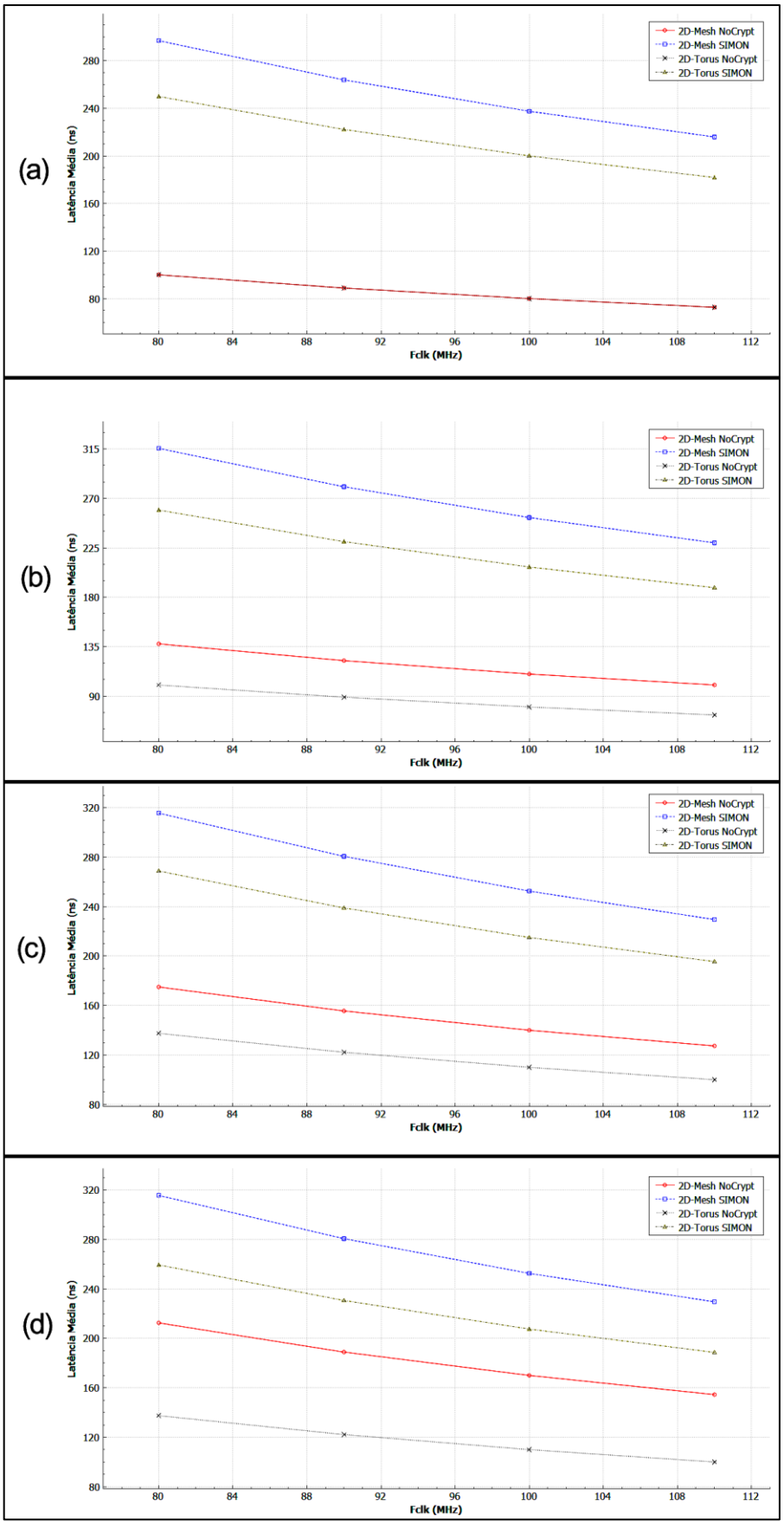

Figura 27. Latência x frequência com um pacote – distribuidor na entre as extremidades da rede: (a) Um salto; (b) Dois saltos; (c) Três saltos; (d) Quatro saltos.

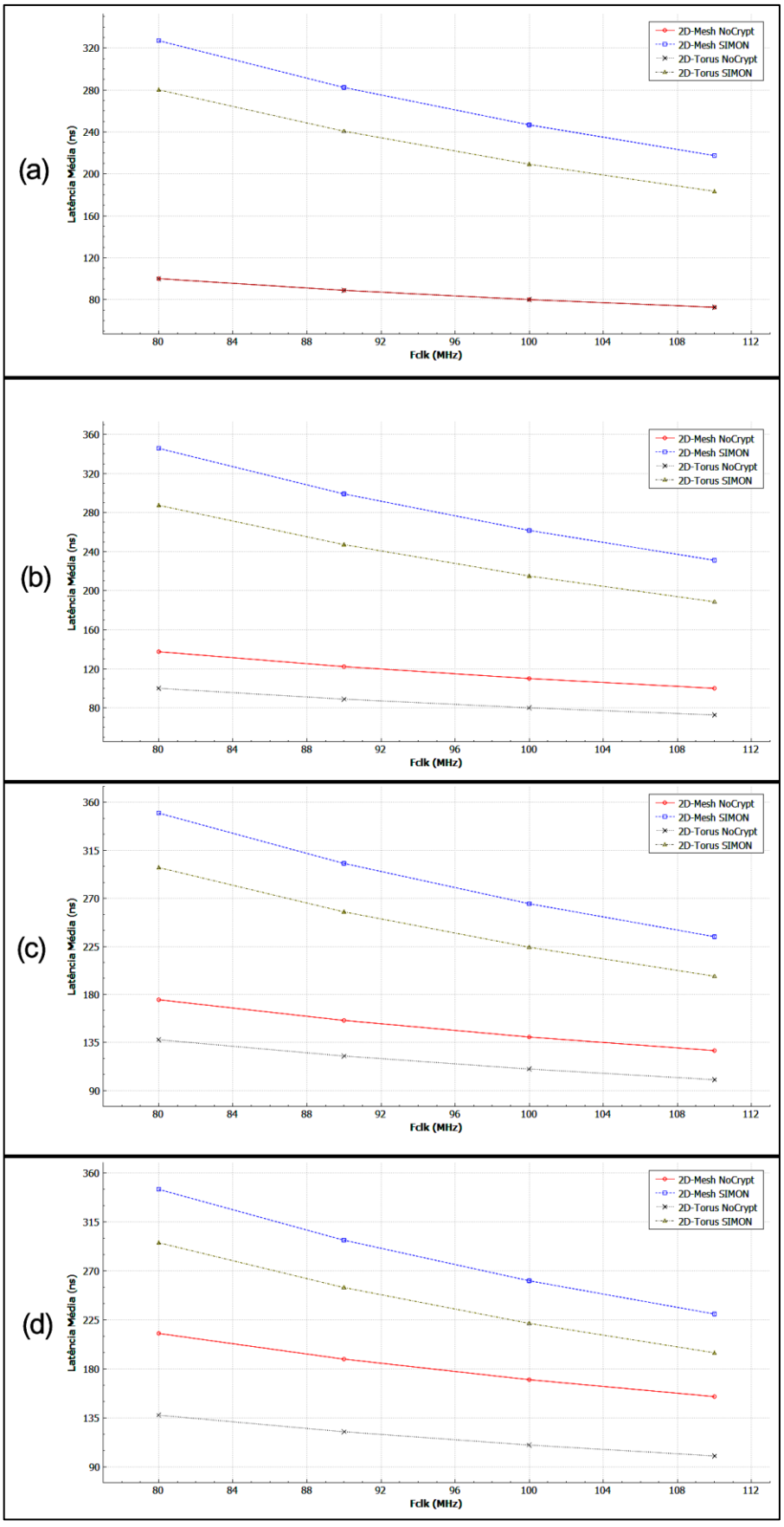

Figura 28. Latência x frequência com três pacotes – distribuidor na entre as extremidades da rede: (a) Um salto; (b) Dois saltos; (c) Três saltos; (d) Quatro saltos.

4.1.4 Análise de forma de onda

Nesta seção, são apresentadas as formas de onda obtidas por meio de duas análises: uma comunicação sem criptografia, e outra, com criptografia. Ambas as comunicações possuem a mesma origem e o mesmo destino, sendo eles os nodos 0 e 1, respectivamente. Nesta análise, a posição do distribuidor estava centralizada na rede.

Conforme ilustra a Figura 29, o envio de uma mensagem sem criptografia apresenta apenas duas ações na rede: o envio da mensagem, que se inicia no ciclo de *clock* 13 e termina no ciclo 15, e, o recebimento da mensagem, que inicia no ciclo 19 e se conclui no ciclo 21.

Quando se analisam as comunicações com criptografia (Figura 30), observa-se que os núcleos envolvidos na troca de mensagens realizam mais ações em comparação à comunicação sem cifragem. Além do envio das mensagens, o núcleo que inicia a comunicação, a origem, solicita uma chave. Essa solicitação é recebida pelo distribuidor, que ao recebê-la, envia uma chave para origem e para o destinatário. O recebimento da solicitação e o envio das chaves pode ser visto na Figura 31. Após o recebimento da chave, o núcleo de origem está apto a enviar a mensagem. No exemplo apresentado, a operação inicia-se no ciclo de *clock* 13. O mesmo ocorre com a comunicação sem cifragem, porém, sua conclusão foi realizada no ciclo 49.

Logo, a operação cifrada necessitou de 23 ciclos de *clock* a mais que as operações não cifradas. Isso representa um aumento de 176% no número de ciclos necessários para concluir trocas de mensagens cifradas.

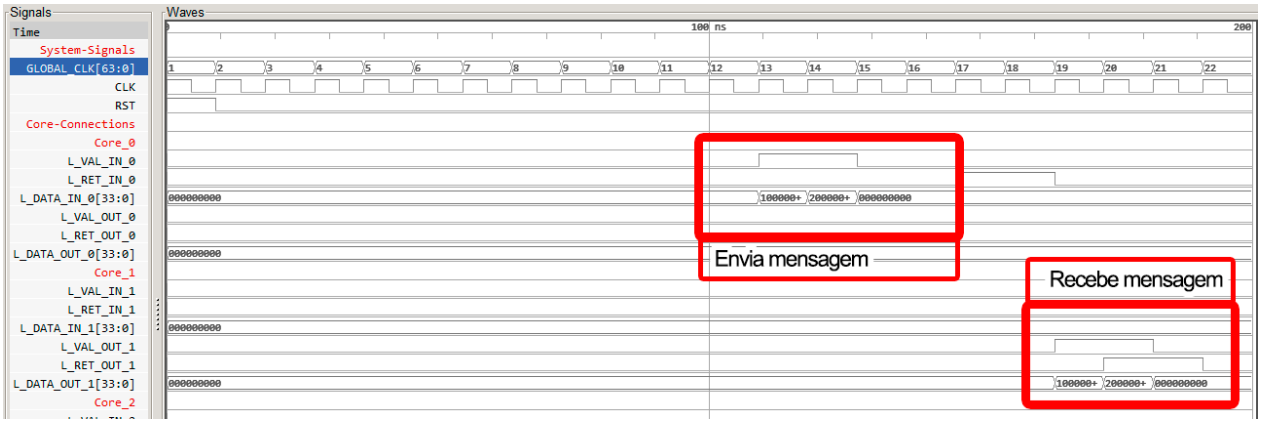

Figura 29. Forma de onda de envio de um pacote na rede sem criptografia.

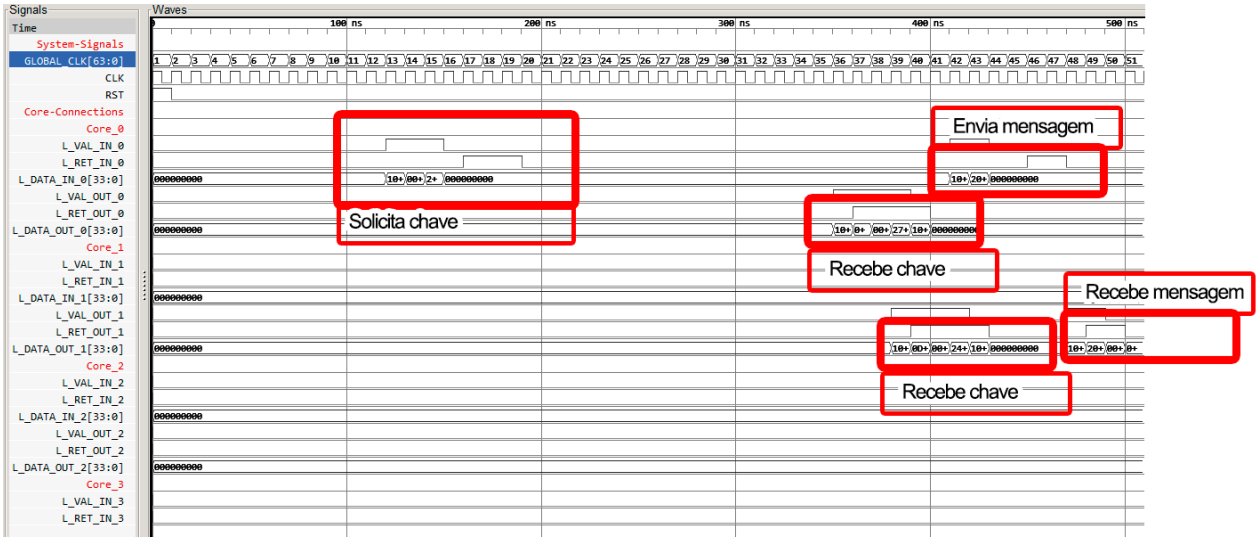

Figura 30. Forma de onda de envio de um pacote na rede com o SIMON.

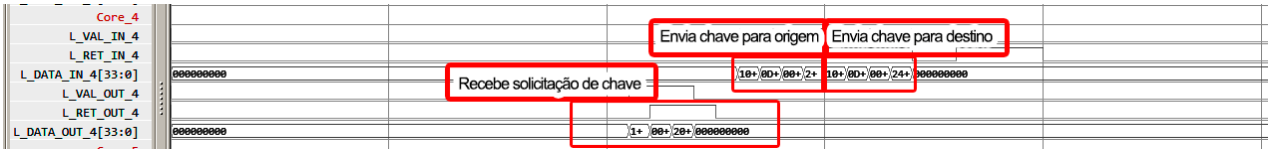

Figura 31. Forma de onda do distribuidor de chave.

4.2 DISCUSSÃO

Conforme observado nos resultados das simulações apresentadas na seção anterior, ao implementar a cifra em blocos SIMON como mecanismo criptográfico nas comunicações da rede SoCIN, houve um impacto considerável na latência média e na quantidade de ciclos de *clock* necessários para a conclusão de cada comunicação.

Embora a posição do distribuidor de chaves tenha sido definida inicialmente no trabalho como a mais centralizada da rede, viu-se necessário realizar a simulação com a posição do distribuidor nas três variações possíveis. Essa abordagem tornou possível a observação do impacto causado pela posição do distribuidor na rede.

Quanto aos resultados apresentados, em relação aos impactos na latência média das comunicações, foi considerada para todos os testes apenas a latência média inicial de cada simulação, sendo esse valor de 80MHz. As demais frequências foram adicionadas nos resultados como forma de demonstrar a variação sofrida na latência das comunicações, quando alterado o seu valor.

Em relação às duas topologias utilizadas nos testes, 2D-Mesh e 2D-Torus, ambas foram escolhidas por conseguirem suportar o modelo proposto, 3x3. Embora suas características já tenham sido apresentadas na parte de fundamentação do trabalho, as diferenças de desempenho quanto aos seus usos foram demonstradas somente no capítulo de resultados.

Quanto ao número de pacotes enviados na rede por cada comunicação concluída, observou-se que em uma comunicação na rede sem a criptografia, o número de total de pacotes é definido pela quantidade parametrizada inicialmente, isto é, o transporte de um pacote não cifrado requer apenas a geração de um único pacote. Já para se enviar esse mesmo pacote criptografado, é necessário, primeiramente, solicitar a chave (1 pacote), ambos os núcleos receberem as chaves (2 pacotes), e então efetivar o envio da mensagem cifrada (1 pacote), em um total de quatro pacotes cifrados para cada pacote sem criptografia no cenário de referência.

5 CONCLUSÃO

Em relação aos conceitos apresentados na fundamentação, destaca-se a necessidade de garantir a segurança nos mais diversos sistemas. Quando o alvo em questão é um processador, garantir sua confidencialidade se tornasse especialmente importante. Assim, esse trabalho apresentou uma proposta de utilizar um mecanismo de criptografia leve para garantir a segurança com o menor impacto possível no desempenho da rede SoCIN.

A escolha da rede SoCIN foi fundamentada pela ligação dela com o grupo de pesquisa em que esse trabalho se insere. Outra escolha importante para o desenvolvimento do trabalho é referente ao algoritmo criptográfico SIMON. Em outro estudo realizado por pesquisadores da Univali, ele foi considerado adequado para implementação em ambientes com recursos restritos.

A fundamentação evidenciou a necessidade de haver mecanismos para prover segurança nas redes-em-chip e consequentemente seu impacto no desempenho da rede ao implementa-los. Esses impactos causados pela inserção de mecanismos ou técnicas de segurança na arquitetura de uma rede-em-chip, foram comprovados através da análise dos resultados obtidos após a execução deste trabalho.

Com relação aos resultados, foram percebidos aumentos significativos na latência das comunicações. Quanto ao impacto na quantidade necessária de ciclos de *clock* para a conclusão das comunicações com criptografia, foi necessário 2.76 vezes ciclos a mais em relação as mesmas mensagens sem cifragem. Esse impacto no desempenho das comunicações da rede é justificado pelo segurança provida pela implementação do SIMON nas trocas de mensagem.

Como sugestão de trabalhos futuros, sugere-se a implementação de novas configurações do SIMON , a avaliação dos impactos do SIMON em diferentes topologias, a avaliação do impacto da implementação do SPECK32/64 em comparativo com o SIMON32/64, a implementação de outros mecanismos de segurança para tratar vulnerabilidades que a criptografia não resolva.

REFERÊNCIAS

BARON, Sidnei. **Segurança em Redes-Em-Chip:** Mecanismos para proteger a rede SoCIN contra ataques de Negação De Serviço. 2013. 117 f. Dissertação (Mestrado em Computação Aplicada), Universidade do Vale do Itajaí - UNIVALI, 2013.

BARROS, Aidil Jesus Paes de; LEHFELD, Neide Aparecida de Souza. **Fundamentos da metodologia:** um guia para iniciação científica. São Paulo: McGraw-Hill,1986.

BEAULIEU, R. *et. al.* **The SIMON and SPECK Families of Lightweight Block Ciphers**. Cryptol ogy ePrint Archive, Report 2013/404, 2013.

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

BLACK, David C.; DONOVAN, Jack. **SystemC**: From the Ground Up, Eklectic Ally. 2004.

CARARA, Everton Alceu. **Serviços de comunicação diferenciados em sistemas multiprocessados em chip baseados em redes intra-chip**. 2011. 107 f. Tese (Doutorado em Ciencia da Computação) - Faculdade De Informática, Universidade Católica do Rio Grande do Sul , 2011.

COSTA, Claudio Roberto *et. al.* **Análise de Metodologias de Implementação e Desempenho em FPGA dos Algoritmos Criptográficos Leves Simon e Speck**. 2016.

CANTANHEDE, Roberto Silva. **Suporte a simulação distribuída em SystemC**. 2007.

COURTLAND, Rachel**. Intel Finds Moore's Law's Next Step at 10 Nanometers**. 30 dezembro 2016. Disponível em: http://spectrum.ieee.org/semiconductors/devices/intel-finds-moores-laws-next-step-at-10-nanometers. Acesso em: 18 março 2017.

DALLY, W. J.; TOWLES, B**. Principles and practices of interconnection networks**. São Francisco: Morgan Kaufmann, 2004.

DE MICHELI, Giovanni; BENINI, Luca. **Networks on chips:** technology and tools. Academic Press, 2006.

DUATO, Jose; YALAMANCHILI, Sudhakar; NI, Lionel M**. Interconnection networks:** an engineering approach. Morgan Kaufmann, 2003.

FIORIN, Leandro; PALERMO, Gianluca; SILVANO, Cristina. **A security monitoring service for NoCs**. In: Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis. ACM, 2008. p. 197-202.

FREY, Jonathan; YU, Qiaoyan. **A hardened network-on-chip design using runtime hardware Trojan mitigation methods**. Integration, the VLSI Journal, v. 56, p. 15-31, 2017.

GEBOTYS, Catherine H.; GEBOTYS, Robert J**. A framework for security on NoC technologies**. In: VLSI, 2003. Proceedings. IEEE Computer Society Annual Symposium on. IEEE, 2003. p. 113-117.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2002.

GOOSSENS, Kees; HANSSON, Andreas. **The *Æ*thereal network on chip after ten years:** Goals, evolution, lessons, and future. In: Design Automation Conference (DAC), 2010 47th ACM/IEEE. IEEE, 2010. p. 306-311.

GWENNAP, Linley. **The Linley Group - Low-Power Design Using NoC Technology**. Maio 2015.

HUANG, Andrew Bunnie. **The Death of Moore's Law Will Spur Innovation**. 3 de março de 2015. Disponível em: http://spectrum.ieee.org/semiconductors/design/the-death-of-moores-law-will-spur-innovation. Acesso em: 18 março 2017.

HWANG, K.; XU, Z. **Scalable parallel computing: technology, architecture, programming**. Nova Iorque: McGraw-Hill, 1998.

ISO/IEC. ISO/IEC 27000:2009: information technology – security techniques – information security management systems – overview and vocabulary. Genebra, 2009.

JERGER, Natalie Enright; PEH, Li-Shiuan. **On-Chip Networks:** Synthesis Lectures on Computer Architecture. [S.l.]: Morgan & Claypool, 2009.

JERRAYA, A. A.; WOLF, W. **Multiprocessor Systems-on-Chips**. São Francisco: Morgan Kaufmann, 2005.

JOHANN FILHO, Sergio; PONTES, Julian; LEITHARDT, Valderi. **Multiprocessor System on a Chip,** 2007.

JUELS, Ari; WEIS, Stephen A. **Authenticating pervasive devices with human protocols**. In: Annual International Cryptology Conference. Springer Berlin Heidelberg, 2005. p. 293-308.

JÚNIOR, Sérgio Vargas; DA SILVA, Eduardo A.; ZEFERINO, Cesar A. **Produção de Material Instrucional para o Ambiente de Simulação RedScarf**. Anais do Computer on the Beach, p. 337-346, 2017.

KEUTZER, Kurt *et. al.* **System-level design:** orthogonalization of concerns and platform-based design. IEEE transactions on computer-aided design of integrated circuits and systems, v. 19, n. 12, p. 1523-1543, 2000.

KNUDSEN L.R; ROBSHAW M.J.B. **The block cipher companion**. Nova York: Springer, 2011

LANDWEHR, Carl E.. **Computer security. Internation Journal of Information Security**, Nova Iorque, v.1, n.1, p. 3-13, 2001.

LAROWE JR, Richard P.; WILKES, James T.; ELLIS, Carla S. **Exploiting operating system support for dynamic page placement on a NUMA shared memory multiprocessor**. In: ACM SIGPLAN Notices. ACM, 1991. p. 122-132.

MARTIN, Grant E.; CHANG, Henry. **Winning the SoC revolution: experiences in real design**. New York: Springer, 2003, p. 320.

MELLO, Aline V. **Qualidade de Serviço em Rede Intra-chip. Implementação e Avaliação sobre a Rede Hermes**, 2006. 129f. 2006. Tese de Doutorado. Master Dissertation, PUCRS, Porto Alegre, Brasil.

METZGER, Luiz Gustavo. **Análise experimental das vulnerabilidades de SoCs baseados em NOC**. 2013. Trabalho Técnico-científico de Conclusão de Curso de Ciência da Computação, Universidade do Vale do Itajaí - UNIVALI, 2014.

MORENO, Edward David; PEREIRA, Fábio Dacêncio; CHIARAMONTE, Rodolfo Barros. **Criptografia em software e hardware**. São Paulo: Novatec, 2005.

MOORE, G. **Cramming more components onto integrated circuits**. Electronics, S.l., v.38, n.8, abr. 1965.

NEVES, Eduardo Borba; DOMINGUES, Clayton Amaral. **Manual de metodologia da pesquisa científica / org**. - Rio de Janeiro: EB/CEP, 2007. 204p.

PASRICHA, Sudeep; DUTT, Nikil. **On-chip communication architectures: system on chip interconnect**. Morgan Kaufmann, 2010.

PRESSMAN, Aaron. **Here's How Intel Is Finally Getting Back on Track With Moore's Law**. 05 janeiro 2017. Disponível em: http://fortune.com/2017/01/05/intel-ces-2017-moore-law/. Acesso em: 06 maio 2017.

REDDY, Tetala Neel Kamal *et. al.* **Performance assessment of different Network-on-Chip topologies**. In: Devices, Circuits and Systems (ICDCS), 2014 2nd International Conference on. IEEE, 2014. p. 1-5.

SILVA, Eduardo A. **RedScarf**: ambiente para avaliação de desempenho de Rede-em-Chip. Itajaí, 2014. 87p. Trabalho Técnico-cientifico de Conclusão de Curso (Graduação em Ciência da Computação) - Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí, 2014.

SILVA, Marcos Roberto da. **Mecanismos para garantir confidencialidade e autenticidade na rede socin**. 2015. Dissertação (Mestrado em Computação Aplicada), Universidade do Vale do Itajaí - UNIVALI, 2015.

SIMON, S.. **The Code Book**, Anchor Books, EUA, 1999.

SHANTHI, D; AMUTHA, R. **Design of Efficient On-Chip Communication Architecture in MpSoC**. IEEE-International Conference on Recent Trends in Information Technology, 2011.

SOPRAN, Robson. **Análise comparativa do custo e do desempenho de um algoritmo de criptografia explorando o particionamento hardware/software**. 2016. Trabalho Técnico-científico de Conclusão de Curso de Engenharia de Computação, Universidade do Vale do Itajaí - UNIVALI, 2016.

SOPRAN, Robson; MELO, Douglas Rossi; ZEFERINO, Cesar. A; BEZERRA. Eduardo A. **Análise Comparativa do Custo e do Desempenho de um Algoritmo de Criptografia para Sistemas Embarcados Explorando o Particionamento Hardware/Software**. Anais do Computer on the Beach, p. 259-268, 2017.

STALLINGS, William. **Criptografia e segurança de redes: princípios e práticas**. Pearson Prentice Hall, 2014.

TANENBAUM, Andrew S.; DAVID, J. Wetherall. **Computer networks**, 5th ed. 2011.

VAHID, Frank. **Sistemas Digitais: projeto, otimização e HDLs.** Bookman Editora, 2007.

WETZELS, Jos; BOKSLAG, Wouter. Simple SIMON: **FPGA implementations of the SIMON 64/128 Block Cipher**. arXiv preprint arXiv:1507.06368, 2015.

ZHANGA, Yuang; LI, Li; LU, Zhonghai; JANTSCH, Axel; GAO, Minglun; PAN, Hongbing; HAN, Feng. **A survey of memory architecture for 3D chip multi-processors**. Microprocessors and Microsystems, 2014.

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

ZEFERINO, Cesar A.; SUSIN, Altamiro A.. **SoCIN: a parametric and scalable Network-on-Chip**. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS, 16., 2003, São Paulo. Proceedings... Los Alamitos: IEEE Computer Society Press, 2003. p. 169-174.

ZEFERINO, Cesar A; SANTO, Frederico G. M. E. e SUSIN, Altamiro A. **ParIS: A Parameterizable Interconnect Switch for Network-on-Chip**. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN, 2004, Pernambuco. Proceedings… New York:ACM, 2004, p. 204-209.

ANEXO A. PARAMETROS DE TESTES DO SIMON

Simon32/64

Chave: 1918 1110 0908 0100

Texto Claro: 6565 6877

Texto Criptografado: c69b e9bb

Simon48/72

Chave: 121110 0a0908 020100

Texto Claro: 612067 6e696c

Texto Criptografado: dae5ac 292cac

Simon48/96

Chave: 1a1918 121110 0a0908 020100

Texto Claro: 726963 20646e

Texto Criptografado: 6e06a5 acf156

Simon64/96

Chave: 13121110 0b0a0908 03020100

Texto Claro: 6f722067 6e696c63

Texto Criptografado: 5ca2e27f 111a8fc8

Simon64/128

Chave: 1b1a1918 13121110 0b0a0908 03020100

Texto Claro: 656b696c 20646e75

Texto Criptografado: 44c8fc20 b9dfa07a

Simon96/96

Chave: 0d0c0b0a0908 050403020100

Texto Claro: 2072616c6c69 702065687420

Texto Criptografado: 602807a462b4 69063d8ff082

Simon96/144

Chave: 151413121110 0d0c0b0a0908 050403020100
Texto Claro: 746168742074 73756420666f
Texto Criptografado: ecad1c6c451e 3f59c5db1ae9

Simon128/128
Chave: 0f0e0d0c0b0a0908 0706050403020100
Texto Claro: 6373656420737265 6c6c657661727420
Texto Criptografado: 49681b1e1e54fe3f 65aa832af84e0bbc

Simon128/192
Chave: 1716151413121110 0f0e0d0c0b0a0908 0706050403020100
Texto Claro: 206572656874206e 6568772065626972
Texto Criptografado: c4ac61effcdc0d4f 6c9c8d6e2597b85b

Simon128/256
Chave: 1f1e1d1c1b1a1918 1716151413121110 0f0e0d0c0b0a0908 0706050403020100 Texto Claro: 74206e69206d6f6f 6d69732061207369
Texto Criptografado: 8d2b5579afc8a3a0 3bf72a87efe7b868