Simulador de NoCs modelado como um SMA

Gustavo L. Lima[1], Nelson F. Traversi[1], Odorico M. Mendizabal[1], Cristina Meinhardt[2], Diana F. Adamatti[1], Eduardo W. Brião[3]

[1]Centro de Ciências Computacionais – C3, Programa de Pós Graduação em Computação -PPGComp, Universidade Federal do Rio Grande (FURG)

[2]Universidade Federal de Santa Catarina (UFSC)

[3]Instituto Federal do Rio Grande do Sul - Campus Rio Grande (IFRS)

{gustavolameirao, 20kfurg, dianaada}@gmail.com

odoricomendizabal@furg.br

cristina.meinhardt@ufsc.br

eduardo.briao@riogrande.ifrs.edu.br

Abstract. *This work presents the development of a NoC simulator modeled as a multiagent system. NoC improves the transmission of messages in SoCs. This simulator aims to provide the user with a high degree of environment abstraction to simulate a NoC, allowing to define the number of messages, packets and transmission speed among routers, thus facilitating the modeling of new systems. In addition, the simulator calculates the routers load, number of messages that passed through each router and the number of lost messages, allowing to observe real-time information. As a case study, the XY routing protocol was modeled and analyzed.*

Resumo. *Este trabalho apresenta o desenvolvimento de um simulador de NoCs modelado como um sistema multiagente. Nocs melhoram a transmissão de mensagens em SoCs. Este simulador visa proporcionar ao usuário um ambiente com alto grau de abstração para simulação de uma NoC, permitindo definir o número de mensagens, pacotes e velocidade de transmissão entre os roteadores, assim facilitando a modelagem de novos sistemas. Além disso, o simulador calcula a carga nos roteadores, o número de mensagens que passaram por cada roteador e o número de mensagens perdidas, permitindo observar parâmetros da simulação em tempo real. Como estudo de caso, foi modelado e analisado o protocolo de roteamento XY.*

1. Introdução

O constante avanço nas tecnologias de fabricação permite que sejam desenvolvidos circuitos integrados cada vez menores e mais eficientes. Além disso, estes circuitos também se tornam cada vez mais complexos, através da integração de diversos componentes (como processador, memórias) dentro de um único chip. Esta evolução levou a criação de sistemas completos dentro de um chip, sendo estes denominados SoC (*System-on-a-chip*). Um SoC tem inúmeras aplicações, principalmente em sistemas

embarcados e *smartphones*. As vantagens, quando comparadas aos sistemas antigos compostos por componentes individuais, são relacionadas principalmente com a melhora no desempenho, redução nos atrasos de comunicação e nos custos de fabricação, uma vez que tudo estará em um chip. Devido à complexidade de comunicação criada nos SoCs, foi proposta a adoção de redes internas nos chip, ou NoCs (*Network-on-Chip*) para realizar a comunicação entre os componentes do sistema (HEMANI, 2000).

O desempenho e confiabilidade das NoCs são afetados por diversas características, por exemplo, topologia, algoritmo de roteamento e controle de fluxo. Para melhorar a compreensão sobre o efeito das configurações associadas às características, é interessante testar o comportamento das NoCs em simuladores. Os simuladores de NoC na literatura podem ser classificados por vários fatores, como os parâmetros de entrada, tipo de resultados obtidos a partir de simulação, tipo de licença, interface de usuário, entre outros. Uma pesquisa abrangente sobre simuladores de NoC é apresentada em (BEN, 2013) (ALALAKI, 2017). Em (BEN, 2013), os simuladores são classificados de acordo com os parâmetros disponíveis e as métricas de saída. Em (ALALAKI, 2017), uma comparação entre é mostrada, de acordo com outras características como framework utilizado, topologias suportadas, suporte para GUI, ferramentas de benchmark, data de lançamento e operação síncrona / assíncrona.

O objetivo deste estudo é desenvolver um simulador de NoCs modelado na forma de um SMA, com foco nos algoritmos de roteamento. Algoritmos de roteamento tem a função de determinar qual caminho os pacotes irão seguir para que a troca de mensagens entre os nodos de destino e origem se comuniquem. O simulador tem a capacidade de simular e avaliar cenários com alto grau de abstração, independência do *hardware*, configuração versátil e obtenção de indicadores de uso de componentes como resultados de saída. Além disso, o simulador está preparado para que os usuários e desenvolvedores possam implementar novos algoritmos de roteamento e os avaliar. Como caso de uso, o algoritmo de roteamento XY foi escolhido. A escolha deste algoritmo se deve a sua simplicidade e vasta presença na literatura sobre NoCs. Além disso, características importantes do algoritmo puderam ser constatadas com a simulação.

Dentre os simuladores encontrados, concentramos nossa pesquisa em simuladores que oferecem a possibilidade de escolher e analisar algoritmos de roteamento. Em seguida, levamos em consideração a configuração dos parâmetros e as métricas de saída disponíveis. Por exemplo, comparamos os recursos dos simuladores para descrever o tamanho do NoC, o tamanho dos buffers, a geração de tráfego, a licença de uso e a interface do usuário. Três simuladores principais foram identificados: Noxim (CATANIA, 2015), DARSIM (LIS, 2010) e NS-2 (NS-2, 2007). Tanto o Noxim quanto o DARSIM apresentam parâmetros de entrada configuráveis similares aos escolhidos no nosso simulador, como tamanho de *buffers*, da rede e dos pacotes. As principais diferenças são relacionadas a geração de tráfego, onde o nosso é capaz de simular tráfego aleatório ou manual, e na existência de uma interface gráfica, que facilita a utilização da criação do cenário, além de ser possível observar comportamentos de saída durante a execução. Já o NS-2 embora seja usado para NoCs, não é um simulador específico de NoC. Ele é um simulador de redes de computadores, onde usuários podem criar abstrações para adaptar o funcionamento para simular uma NoC.

A Seção 2 deste trabalho apresenta os principais conceitos sobre NoCs. A Seção 3 mostra detalhes sobre a implementação do simulador (como os parâmetros de entrada e indicadores). A Seção 4 apresenta os resultados obtidos em simulações. Por fim, a Seção 5 apresenta as conclusões do trabalho.

2. Network-On-Chip

A complexidade criada pela integração de vários sistemas em um chip faz com que seja repensada a forma com que estas partes se comunicam. Algumas arquiteturas de comunicação, como barramentos ou fios dedicados, quando usadas nos SoCs, oferecem baixa escalabilidade. Os barramentos apresentam limitações de transmissão, pois apenas uma palavra pode ser transmitida por ciclo de clock. Conexões com fios dedicados também não são uma boa opção, devido a baixa reusabilidade e flexibilidade (BRIAO, 2008).

Como alternativa para estas limitações, surgem outras propostas de como realizar esta comunicação, as NoCs (Network-On-Chip) (HEMANI, 2000). De maneira geral, a comunicação é feita através da criação de uma grande rede, composta por nodos de processamento conectados a nodos de chaveamento (através de uma interface de rede), e os nodos de chaveamento interligados por *links* físicos. Os nodos de processamento, denominados núcleos ou *Intellectual Property cores*, têm a função de executar as ações que o sistema deve realizar. Já os nodos de chaveamento, os roteadores, são responsáveis por realizar a transferência de mensagens entre dois cores que desejam se comunicar, através de ligações físicas, ou seja, através dos *links*.

A forma com que os roteadores são dispostos depende da topologia adotada. Um exemplo de organização é apresentado na Figura 1 (TOPÎRCEANU, 2013). Na Figura, é apresentada uma NoC 3x3 destacando seus principais componentes, os Núcleos de Processamento, as *Interfaces* de rede, os roteadores e os *links* físicos.

Trocas de mensagens entre dois pontos da NoC podem ocorrer por diversos caminhos. Este fato faz com que seja necessário adotar estratégias de roteamento, de maneira similar ao que acontece em uma rede de computadores, para determinar o caminho por onde a troca de mensagens ocorrerá. O uso de roteamento faz com que a comunicação nas NoCs seja mais eficiente quando comparado com os barramentos convencionais, podendo ser observado qual o menor caminho entre os pontos, ou qual caminho está menos congestionado. Além disso, as decisões de roteamento dos pacotes que trafegam pela NoC podem ser distribuídas, auxiliando assim na escalabilidade do sistema.

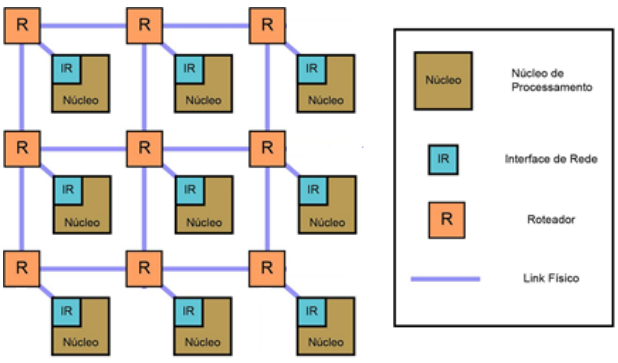

Figura 1: Organização de uma NoC

A Interface de Rede é responsável por fazer a ligação entre os processadores e a rede, tornando assim a comunicação mais transparente para o processo. A comunicação é dividida em duas partes: *back-end* e *front-end*. O *back-end* é responsável por fazer a comunicação com os núcleos, tanto do processo de envio, quanto no recebimento das mensagens. Esta estrutura permite a reutilização de multiprocessadores, bastando apenas mudar detalhes nesta camada. O *front-end* tem por objetivo transmitir os dados utilizados em serviços de alto nível da camada de transporte, como transferência de dados confiáveis e dados sobre qualidade de serviço (QoS) (NUNES, 2015).

A comunicação entre componentes da NoC é realizada pelo envio e recebimento de mensagens. Porém, como as mensagens que trafegam são grandes e de tamanhos variados, estas são divididas em pacotes menores, com tamanho menor ou igual a largura de banda do Enlace (NUNES, 2015). Cada pacote é dividido em três partes:

1. *Header* (cabeçalho) - contém as informações para rotear a mensagem pela rede.

2. *Payload* (carga útil) – é o corpo da mensagem.

3. *Trailer* (terminador) - comando que indica o término da mensagem.

Os roteadores são componentes cruciais dentro das NoCs. São responsáveis por transmitir os dados que trafegam por essa rede. A maioria dos roteadores de alto desempenho implementam uma rede de interconexão do tipo *crossbar* para permitir que múltiplas conexões entre portas de entrada e saída aconteçam simultaneamente. Como os roteadores recebem uma grande quantidade de mensagens, devem prover filas eficientes onde os pacotes serão armazenados até serem processados e enviados aos seus destinatários. As mensagens podem ser guardadas nos buffers de entrada, nos buffers de saída, em ambos ou até mesmo dentro do roteador (HENNESSY, 2008).

De forma geral, o roteamento é implementado utilizando uma máquina de estados finita ou uma tabela de encaminhamento, armazenada dentro da unidade de controle de roteamento no roteador. Na primeira opção, os dados de roteamento presentes no cabeçalho da mensagem são processados pela máquina de estados, e esta determina para qual saída a mensagem deverá ir. Já na segunda opção, as informações de roteamento presentes no cabeçalho da mensagem são utilizadas como um endereço de acesso a tabela de roteamento, carregada na inicialização do roteador, que contém as saídas a serem utilizada. A unidade de controle de roteamento também é responsável pela arbitragem, pois os pacotes podem chegar de maneira simultânea as portas de entrada, podendo dois pacotes requerer a mesma saída. Cabe ao roteador determinar quem terá prioridade ao acesso na porta requerida. A arbitragem pode ser centralizada ou distribuída. Na abordagem centralizada, todo o processo de distribuição dos pacotes às portas de saída é feito por um agente centralizador. Já na abordagem descentralizada, cada porta de entrada possui um módulo de roteamento, juntamente com um módulo de arbitragem ligado à sua respectiva porta de saída.

3. Desenvolvimento de um Simulador de NoC no Netlogo

O simulador de NoCs foi implementado no ambiente de criação de ferramentas multiagente chamado NetLogo (WILENSKY, 1999). Cada nodo (roteador) da NoC é representado por um agente. A definição do agente inteligente mostra similaridades. O roteador é uma entidade real, que opera em um ambiente, a rede NoC. Ele se comunica

com outras entidades: os demais roteadores. Tem o objetivo de realizar a comunicação de forma eficiente, possui recursos próprios tais como os *buffers* de entrada/saída e processamento. Além disso, é capaz de perceber mudanças no ambiente, por exemplo detectar congestionamento com seus vizinhos. E possui apenas uma representação parcial do ambiente completo sendo capaz de realizar serviços como enviar mensagens para outros roteadores. O mapeamento com as principais semelhanças entre agentes e roteadores é apresentado na Tabela 1.

Na implementação do simulador, a topologia inicial escolhida para a distribuição dos agentes foi uma malha 2D, formando uma grade. Os agentes foram organizados nesta grade distanciados uns dos outros por uma unidade de medida. Em seguida, criou-se os *links* entre cada um dos agentes dentro de uma distância 1. Dessa forma, as partes que compõem a grade foram conectadas apropriadamente. A malha formada é quadrada, e tem dimensão NxN, onde N é um valor escolhido pelo usuário. Para permitir a criação de diferentes cenários de simulação, este simulador disponibiliza ao usuário uma série de características configuráveis, sendo estas:

- **Número de roteadores:** Com base no valor N escolhido, será gerada uma NoC contendo *N* linhas e *N* colunas.
- **Tamanho do *buffer:*** Define em número de pacotes o tamanho do *buffer* de cada roteador.
- **Quantidade de pacotes por mensagem:** as mensagens são divididas em pacotes, e cada pacote possui um tamanho padrão. Esta opção define quantos pacotes são necessários para compor uma mensagem.
- **Pacotes por ciclo:** define quantos pacotes cada roteador é capaz de enviar em cada ciclo. Por exemplo, se for escolhido 3, a cada ciclo os roteadores irão enviar no máximo 3 pacotes, mesmo que tenham mais ou menos mensagens armazenadas em seus buffers.
- **Enviar mensagens aleatoriamente:** quando selecionado, este parâmetro realiza simulações onde remetente e destinatário de mensagens são escolhidos aleatoriamente.
- **Raio de mensagens:** determina o valor do raio onde cada roteador de origem irá escolher um roteador destino para enviar mensagens.
- **Completamente aleatório:** quando selecionado, o raio de escolha de pares de roteadores, remetente e destinatário, é totalmente aleatório, podendo escolher qualquer roteador da rede.
- **Número de ciclos para envio de novas mensagens**: define a frequência com que novas mensagens serão enviadas.
- **Mensagens a serem enviadas:** define um total de mensagens que devem ser enviadas em uma simulação.
- **Dados para envio de mensagem manual:** em um cenário onde determinadas mensagens definidas precisam ser enviadas, é possível escolher o agente origem através do campo Agente-Origem e as coordenadas X e Y do destinatário através dos campos Mensagem-X/Mensagem-Y. Após escolher os valores, basta disparar a mensagem através do botão mensagem-individual.

Tabela 1. Características de agentes e roteadores

Agente	Roteador
Entidade Real/Virtual	Entidade Real
Ambiente	Rede NoC
Comunicação com outros agentes	Ligação entre os roteadores
Objetivos/Satisfações	Comunicação eficiente
Recursos Próprios	Buffers de entrada, processamento, dentre outros
Percebe o ambiente	Detecta congestionamento entre vizinhos
Representação parcial	Somente se relaciona com vizinhos conectados
Competência/Prestação de serviços	Envio da mensagem ao destino desejado

Finalizando a configuração dos parâmetros de entrada, a NoC é montada através do botão *setup*, e a simulação é executada através do botão *go*. A simulação é medida por ciclos. Cada vez que o *go* é executado, contabiliza-se 1 ciclo de execução. A função *go* é repetida indefinidamente até atingir uma condição de parada. Nessa função, cada roteador é processado sequencialmente. No processamento do roteador, este verifica se tem mensagens armazenadas em seu *buffer*, e transmite até x mensagens para o próximo roteador de acordo com o algoritmo de roteamento. x é o número máximo de mensagens que um roteador consegue encaminhar por ciclo. Quando um roteador recebe um pacote no *buffer*, primeiramente é verificado se existe espaço disponível para o armazenamento no *buffer*. Caso não exista, o pacote é descartado e é contabilizado como perdido. Se existir espaço, o pacote é colocado no final da fila de ciclo *buffer* para ser processado. A fila do *buffer* segue o modelo FIFO (*First in, First Out*). Ao fim, é calculado quanto tempo levou desde o início da simulação e o recebimento do última pacote, para fins de medição de desempenho do simulador.

Além destas informações, os roteadores armazenam outros dados como: Quantidade de ciclos que ficou em cada cor, onde a cor está relacionada a taxa de ocupação dos *buffers* do roteador; Quantidade total de pacotes processados pelo mesmo; Quantidade de pacotes perdidos pelo mesmo; e Taxa de ocupação geral, sendo determinada como a média da porcentagem de ocupação de seus buffers ao longo da execução como um todo.

A função *transmit* é responsável por determinar a forma com que os pacotes são roteados. Os algoritmos de roteamento nas NoCs têm o papel de determinar o caminho pelo qual uma mensagem deve trafegar para chegar ao seu destino. Existe uma série de fatores que diferenciam estes algoritmos e que são utilizadas para classificar os mesmos. Os algoritmos de roteamento mais conhecidos são o XY, NL, NF e WF (CARARA, 2004). Nesta primeira versão do simulador, a função *transmit* implementa o algoritmo XY (CARARA, 2004), devido a sua simplicidade e aparecer frequentemente na literatura. Entretanto, novos algoritmos de roteamento podem ser acrescentados sobrescrevendo a função *transmit*. Permitindo com o simulador a avaliação do algoritmos de roteamento e seu impacto na NoC simulada. O Algoritmo XY é um algoritmo determinístico, isto significa que para cada entrada determinada, o algoritmo

sempre produzirá a mesma saída. Neste algoritmo, os pacotes são roteados na direção X até atingir a coordenada X destino, em seguida são roteados na direção Y até atingir Y destino. Um exemplo de funcionamento é mostrado na Figura 2, onde cada quadrado representa um roteador, cada número identifica o roteador, e as setas indicam o caminho que a mensagem fez para ir do roteador 1 ao 10 (CARARA, 2004).

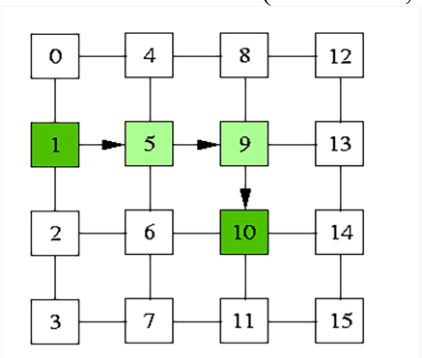

Figura 2: Estrutura do algoritmo XY

O simulador também disponibiliza uma série de dados da simulação, tanto durante quanto ao final da execução. Dentre os indicadores, é interessante destacar:

- **Coloração dos roteadores:** durante cada ciclo, os roteadores recebem a cor referente a sua taxa de ocupação.
- **Contabilização de pacotes:** É possível verificar a quantidade de pacotes enviados, recebidos (ao chegar no destino), perdidos (por não existir buffers para o armazenamento) e trafegando pela rede (que ainda não atingiram seu destino, mas estão em movimento).
- **Total do número de saltos:** contabiliza por quantos roteadores os pacotes passaram.
- **Número médio de saltos:** define-se uma média de quantos saltos os pacotes deram até chegar ao destino.
- **Tabela de ciclos:** tabela com a quantidade de ciclos que o roteador ficou em determinada cor.
- **Tabela de taxa de ocupação:** tabela com a taxa de ocupação de cada roteador ao longo da execução.
- **Tempo de execução**: desde o início da simulação até o recebimento da última mensagem.

A coloração dos registradores é realizada após processar as mensagens, mediante um cálculo realizado para determinar a porcentagem de ocupação do *buffer* do roteador. Com base nesta porcentagem, o simulador oferece uma saída visual da utilização do roteador, colorindo de acordo com a classificação da capacidade ocupada dos buffers apresentada na Tabela 2.

Tabela 2. Representação de cor conforme ocupação do buffer

Cor	Ocupação
Azul	0~1% da capacidade do buffer ocupada
Verde	1~30% da capacidade do buffer ocupada
Amarelo	30~60% da capacidade do buffer ocupada
Vermelho	60~99% da capacidade do buffer ocupada
Preto	100% da capacidade do buffer ocupada

4. Estudo de Caso: Algoritmo XY

Para averiguar o funcionamento da ferramenta desenvolvida, foram criadas simulações variando os principais atributos de entrada. Vale ressaltar que, para o estudo de caso, o algoritmo de roteamento utilizado na ferramenta foi o XY. Dois grupos de simulações foram propostos. No grupo 1, foram realizados testes variando o raio de escolha de roteadores para envios de mensagens, a quantidade de pacotes por mensagens, a quantidade de buffers e a quantidade de pacotes processados por ciclos em cada roteador. No grupo 2, foi criado um caso padrão (o mesmo do primeiro grupo) e foi variado o tamanho da NoC. A metodologia de cada experimento e os resultados são detalhados na sequência deste trabalho.

4.1. Grupo 1

A partir de uma NoC 20x20, foram enviadas 1000 mensagens. O raio da escolha de roteadores foi variado para 3 (pequeno), 9 (médio) e 30 (grande), tal valor de raio determina a distancia maxima possivel que o roteador destino pode ser escolhido a partir do roteador de origem. Dentro de cada raio, também foram criados 4 casos de teste, variando mais características, totalizando 12 casos de teste. Os casos são:

- **Caso padrão:** 10 pacotes por mensagens, *buffers* de tamanho 100 em cada roteador e cada roteador processa 10 pacotes por ciclo.

- **Caso 1:** 50 pacotes por mensagens, *buffers* de tamanho 100 em cada roteador e cada roteador processa 10 pacotes por ciclo, simulando um ambiente com mensagens maiores.

- **Caso 2:** 10 pacotes por mensagens, *buffers* de tamanho 500 em cada roteador e cada roteador processa 10 pacotes por ciclo, representando um ambiente com roteadores com mais *buffers*.

- **Caso 3:** 10 pacotes por mensagens, *buffers* de tamanho 100 em cada roteador e cada roteador processa 100 pacotes por ciclo, simulando um ambiente onde os roteadores com maior capacidade de processamento.

- **Caso 4:** 10 pacotes por mensagens, *buffers* de tamanho 15 em cada roteador e cada roteador processa 10 pacotes por ciclo, representando um ambiente onde roteadores têm menos *buffers*, para simular perdas de pacotes.

Na Tabela 3 é possível observar os valores de tempo de simulação, quantidade de pacotes enviados e recebidos, encontrados nas simulações dos 5 casos.

Tabela 3. Resultados obtidos nos casos com raio 3, 9 e 30. Relação entre tempo de simulação (medido em segundos), número de pacotes enviados e número de pacotes perdidos.

Caso	Raio 3	Raio 9	Raio 30
Padrão	16,015s//10000//0	22,064s//10000//0	31,452s//10000//0
1	55,574s//50000/0	100,381s//49970//30	159,896s//49900//100
2	45,446s//10000//0	52,107s//10000//0	60,652s//10000//0
3	15,607s//10000//0	23,67s//10000//0	30,949s//10000//0
4	15,194s//9970//30	15,092s//9665//335	22,179s//8975//1025

Para ilustrar a quantidade de mensagens manipuladas por cada roteador, foi escolhido o caso 2. O tipo de gráfico escolhido foram os mapas de calor. Estes mapas são utilizados para mostrar a intensidade da ocorrência de alguma situação. Os mapas de calor deste caso nos raios 3, 9 e 30 são apresentados nas Figuras 3, 4 e 5, respectivamente. Nestes gráficos, os pontos onde ocorreram a maior manipulação de mensagens estão com a coloração mais acentuada. Vale ressaltar que houve uma normalização nos gráficos. O ponto máximo da escala de cada gráfico foi escolhido com base no maior valor apresentado na representação em questão.

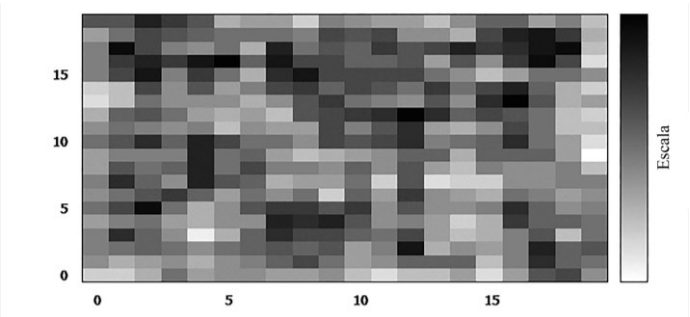

Figura 3: Estrutura do algoritmo XY

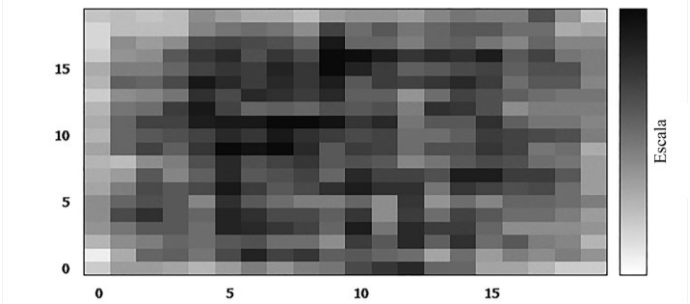

Figura 4: Estrutura do algoritmo XY

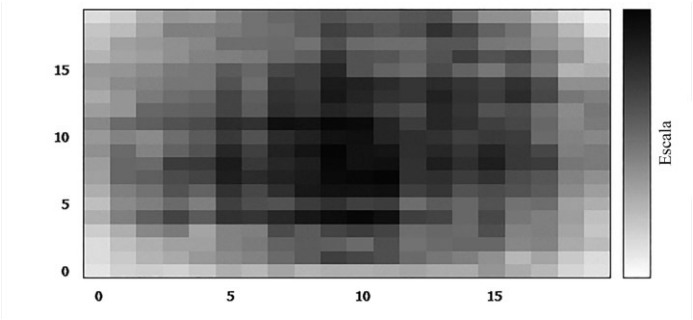

Figura 5: Estrutura do algoritmo XY

Ao comparar o comportamento do mapa de calor de quantas mensagens passaram pelos roteadores em um mesmo caso, variando o raio (exemplo: caso 1 nos raios 3, 9 e 30), é possível perceber que conforme o raio aumenta, acontece uma concentração de mensagens manipuladas no centro, como mostra a Figura 5. Este comportamento é esperado para esse algoritmo de roteamento, devido ao fato de que, ao utilizar um raio pequeno, os roteadores trocam mensagens apenas com roteadores vizinhos. Porém, ao aumentar o raio, existe troca de mensagens entre roteadores que podem estar muito distantes. Por esse motivo, as mensagens passam pelo meio, e há uma maior carga nestes roteadores centrais ilustrada na Figura 5. É importante ressaltar que, embora seja mostrada apenas a comparação do caso 2, este comportamento se repete em todos os casos.

Além desse comportamento, também foi possível observar que boa parte das variáveis não influenciam diretamente no desempenho do sistema, mas sim na perda de pacotes. Nas simulações aconteceram poucas perdas de pacote, pois foram ambientes balanceados e projetados para que isso ocorresse. O caso que teve maior variação de tempo, comparado aos demais, foi o caso 1, que utiliza um maior número de pacotes por mensagem. De maneira simplificada, mais pacotes por mensagem significa mensagens maiores. Esse caso fica mais pesado ainda conforme o raio aumenta devido ao fato de que podem ser feitas comunicações entre roteadores mais distantes, ou seja, vários pacotes precisam trafegar a rede inteira para chegar ao destino. Como são vários pacotes por mensagem, os roteadores intermediários terão seus buffers ocupados e irão precisar manipular vários pacotes que atravessam a rede por grandes distâncias.

4.2. Grupo 2

Na segunda bateria de testes, a partir do mesmo caso padrão anterior e com o mesmo número de 1000 mensagens, variou-se o tamanho da NoC e a forma de escolha dos roteadores para as trocas adotando o modo totalmente aleatório. Os tamanhos escolhidos para a NoC foram 3x3, 9x9, 20x20 e 50x50. A relação do tamanho da NoC com o tempo de execução é mostrada na Figura 6. Este grupo de testes foi simulado para testar a escalabilidade do sistema, conforme o tamanho da rede aumenta. Conforme observado no gráfico, o aumento é linear. Mesmo no maior dos casos, onde a NoC é 50x50, o tempo de simulação ficou em torno de 101,253 segundos.

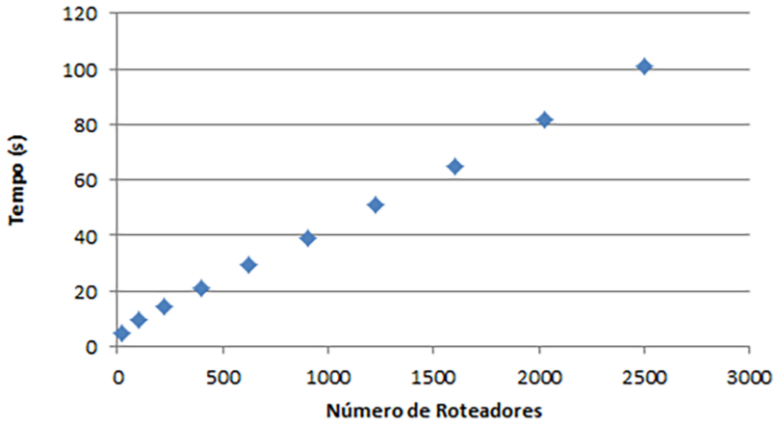

Figura 6: Estrutura do algoritmo XY

5. Conclusões

Este estudo mostrou as etapas da construção de um simulador de NoCs desenvolvido em um ambiente multiagente. Para demonstrar o potencial de aplicação do simulador, foram apresentados os resultados obtidos em simulações variando diversos parâmetros, como tamanho da rede, o raio de escolha de roteadores, tamanho das mensagens, quantidade de buffers, capacidade de processamento dos roteadores, dentre outros. Foi possível observar comportamentos nas simulações, como o da concentração de mensagens que passam pelo centro da NoC. Estes comportamentos são esperados por conta do algoritmo de roteamento utilizado, o XY.

Em síntese, conclui-se que o simulador possui uma alta versatilidade, onde o usuário pode criar uma NoC descrevendo vários parâmetros de maneira simples e obter indicadores. Dessa forma, a ferramenta pode auxiliar os desenvolvedores de NoCs a obter os parâmetros adequados para a sua aplicação, de maneira eficiente e simples. Acredita-se que estudos envolvendo as áreas de NoC e SMA são relevantes e que este trabalho possa contribuir neste sentido. Como trabalhos futuros, pretende-se implementar novos algoritmos de roteamento, para que seja possível comparar o desempenho destes e ver qual se adapta melhor a NoC desenvolvida e realimentar o simulador com parâmetros mais acurados.

Referências

ALALAKI, M.; AGYEMAN, M. A Study of Recent Contribution on Simulation Tools for Network-on-Chip (NoC), International Journal of Cumpter Systems, vol. 11 no.4, 2017.

ALI, M. et. al. Using the NS-2 Network Simulator for Evaluating Network on Chips (NoC), in International Conference on Emerging Technologies, 2006, pp. 506 - 512.

BRIAO, E. W. Métodos de Exploração de Espaço de Projeto em Tempo de Execução em Sistemas Embarcados de Tempo Real Soft Baseados em Redes-em-Chip. Tese de doutorado (Ciência da Computação). UFRGS. Porto Alegre. 2008.

BEN, A.; SAOUD, S. A Survey of Network-On-Chip Tools, International Journal of Advanced Computer Science and Applications, vol. 4 no. 9, 2013.

CARARA, E. A. Uma exploração arquitetural de redes intra-chip com topologia malha e modo de chaveamento Wormhole. Trabalho de Conclusão de Curso, 2004.

CARDOZO, E.; MAGALHÃES, M.F. Redes de computadores: modelo OSI, 2012. Disponível em: < ftp://ftp.dca.fee.unicamp.br/pub/docs/eleri/apostilas/osi.pdf >. Acesso em: ago. 2017.

CATANIA, V. et. al. Noxim: An Open, Extensible and Cycle-accurate Network on Chip Simulator, in Application-specific Systems, Architectures and Processors (ASAP), 2015 IEEE 26th International Conference on, Toronto, 2015.

HEMANI, A. et. al. Network on chip: An architecture for billion transistor era. In: IEEE NORCHIP CONFERENCE, 2000. Procceding... [S.1.:s.n.], 2000.

HENNESSY, J. L.; PATTERSON, D. A. Arquitetura de Computadores: Uma abordagem quantitativa. Elsevier, 4 edition, 2008.

HENNESSY, J. L.; PATTERSON, D. A. Organização e Projeto de Computadores – A Interface Hardware Software. 3 edition, 2005.

LIS, M. et. al. DARSIM: a parallel cycle-level NoC simulator, in IEEE Asian SolidState, Circuits Conference, Saint Malo, 2010.

NS-2 website. Available: nsnam.isi.edu/nsnam/index.php/Main_Page, 2007.

NUNES, C. A. K. Uma estratégia para redução de congestionamento em sistemas multiprocessadores baseados em NOC. 2015. Dissertação (Mestrado em Ciência da Computação). Centro de Informática, UFPE, Pernambuco.

TOPÎRCEANU, A. Networks On Chip, 2013. Disponível em: < https://sites.google.com/site/alexandrutopirceanu/research/networks-on-chips>. Acesso em: ago. 2017.

WILENSKY, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.