PROJETO E VALIDAÇÃO DE CANAIS VIRTUAIS EM REDE DE COMUNICAÇÃO INTRACHIP[1]

Marcelo Felix CARLOS[2]
Discente de Graduação em Engenharia Eletrônica
IFSP/Câmpus São Paulo

Ricardo PIRES[3]
Docente do Departamento de Elétrica
IFSP/Câmpus São Paulo

Martha Johanna Sepúlveda FLÓREZ[4]
Pesquisadora Sênior
Technische Universität München, Alemanha

RESUMO
Este trabalho tem como finalidade apresentar o projeto e a validação dos chamados canais virtuais dentro de roteadores de redes intrachip, além de mostrar uma nova proposta de decisão de inserção de pacotes nesses canais, baseada nos tempos de permanência recente dos pacotes. Para a realização do projeto, foi utilizada a biblioteca SystemC, para a implementação dos módulos e para a simulação do sistema. Foram projetados três tipos de módulos, para servirem como componentes dos canais virtuais. Inicialmente, eles foram desenvolvidos e testados isoladamente. Após a verificação de seu funcionamento, eles foram inseridos numa rede e testados novamente, comparando-se os resultados de tráfego usando-se canais virtuais com os obtidos utilizando *buffers* simples. Foram obtidos resultados satisfatórios em termos de tempo para os pacotes chegarem aos seus destinos, o que permitiu constatar a eficiência do uso dos canais virtuais na implementação proposta.
Palavras-chave: Redes intrachip. Canais virtuais. SystemC. Circuitos integrados.

Introdução

Com o constante avanço tecnológico na Microeletrônica, nasce a possibilidade de se criarem sistemas eletrônicos cada vez menores e mais complexos, dando origem a circuitos integrados (CIs) que chegam a ter bilhões de transistores. Desta possibilidade, nascem os chips de aplicação específica do tipo *SoCs* (*System-on-chip*).

Segundo Gupta (1997), o projeto de um *SoC* é baseado no reúso de módulos IP (*intellectual property*). Gupta (1997) define estes módulos como sendo peças de hardware pré-

[1] Orientador Prof. Dr. Ricardo Pires e Co-orientadora Profa. Dra. Martha Johanna Sepúlveda Flórez.
[2] Endereço eletrônico: marcelo.carlos1997@gmail.com
[3] Endereço eletrônico: ricardo_pires@ifsp.edu.br
[4] Endereço eletrônico: johanna.sepulveda@tum.de

projetadas e pré- verificadas, que podem ser usadas como blocos para maiores e mais complexas aplicações em um CI. De acordo com Mitić (2006), *SoCs* são circuitos integrados que podem incluir um ou mais componentes programáveis, assim como processadores de propósito geral, processadores de sinais digitais, ou outros módulos IP, além de controladores *Analog front-end*, memória *on-chip*, dispositivos I/O e outras aplicações de circuitos específicos. Em geral, segundo Carara (2007), *SoCs* são sistemas computacionais completos implementados em um único CI. Para F. Moraes,

> Normalmente, a arquitetura de interconexão, utilizada nos *SoC* para interconectar os módulos IP, é baseada em fios dedicados e barramentos compartilhados, que, em sistemas com poucos módulos, faz-se muito eficiente. Entretanto, conforme o número de módulos aumenta e a complexidade do sistema cresce, essa eficiência tende a cair (MORAES, 2004).

O uso de barramentos e de fios dedicados em sistemas complexos tende a ocupar muita área em relação à dos módulos IP e requerer consumo de energia de ordem de grandeza próxima à deles, além de ter um desempenho limitado pelo fato de que um barramento não permite paralelismo temporal na comunicação. Para solucionar estes problemas, foi proposto o conceito de *NoC* (*Network-on-chip)*.

Em uma *NoC*, os módulos do sistema são interligados por meio de uma rede composta por roteadores e canais ponto-a-ponto. "A comunicação entre os núcleos ocorre pela troca de mensagens transferidas por meio de roteadores e canais intermediários até atingir o seu destino" (BENINI, 2002). O emissor da mensagem divide-a em partes e encapsula cada uma numa estrutura chamada pacote, o qual contém as informações necessárias ao seu encaminhamento ao longo da rede. Um roteador é um módulo que possui vários terminais de entrada, nos quais são recebidos pacotes, e vários terminais de saída, pelos quais os pacotes recebidos são encaminhados a outros roteadores que estejam no caminho a ser seguido pelos pacotes até os seus destinos (ASLAM; KUMAR; HOLSMARK, 2013). Um pacote pode, ainda, ser subdividido em fragmentos chamados *flits*. Essa subdivisão permite que partes sucessivas possam estar, em um dado momento, em diferentes roteadores pelo seu caminho, eliminando a necessidade de um roteador ter de armazenar um pacote completo, antes de encaminhá-lo ao roteador seguinte. O fluxo chamado *wormhole* é aquele em que cada pacote é subdividido em *flits* e em que todos os flits de um pacote seguem o mesmo caminho que o seu primeiro *flit*, sem que *flits* de outros se interponham.

Lee (2007) afirma que as *NoCs* não apenas escalam muito bem em termos de área, como também em consumo de energia e em desempenho.

Nos roteadores das *NoCs*, podem ser usados os chamados *buffers*. Neste contexto, um *buffer* é um espaço de memória dedicado a armazenar, junto a uma entrada de um roteador, pacotes em uma fila, que esperam para ser transmitidos na rede. Um componente fundamental do roteador é o comutador, o qual recebe cada pacote que entra e o encaminha para a saída que estiver no melhor caminho para o seu destino.

Um dos problemas que as *NoCs* enfrentam são os impasses (ou *deadlocks*), principalmente quando é usado o método de chaveamento *wormhole*. Um impasse pode ser definido como uma dependência cíclica ao longo dos nós da rede, requisitando acesso a um conjunto de recursos. "Desta forma nenhum progresso pode ser conseguido, não importa a sequência em que os eventos acontecerem" (MORAES, 2004). "Uma vez que um pacote A é alocado a um *buffer bi*, nenhum outro pacote B pode usar o canal associado *ci* até que A deixe *bi*" (DALLY, 1992), conforme a Figura 1.

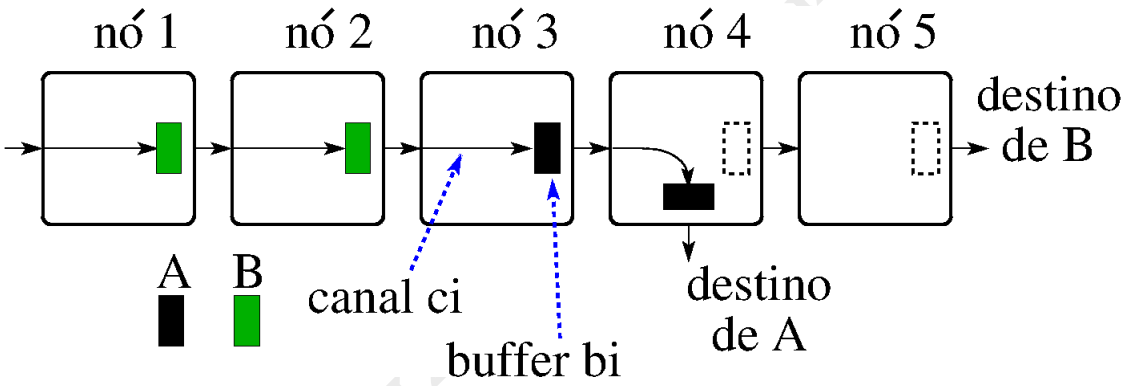

Figura 1 – O pacote B está bloqueado atrás do pacote A, o qual está fazendo curva à direita, enquanto há canais físicos ociosos no caminho de B, em frente. Cada nó da rede é um roteador
Fonte: Adaptado de DALLY (1992)

Visando a solucionar este tipo de problema, foram inventados os canais virtuais, os quais são caminhos paralelos dentro do roteador. "Adicionar canais virtuais em uma rede de interconexão é análogo a adicionar pistas em uma estrada" (DALLY, 1992), permitindo a passagem de pacotes bloqueados, em situações como a da Figura 1, e, por consequência, aumentando a taxa de transferência de pacotes na rede. A Figura 2 ilustra o uso de canais virtuais, como solução para o bloqueio ilustrado na Figura 1. Agora, o pacote B pode contornar o pacote A, por um canal paralelo, já que A e B devem prosseguir por caminhos diferentes.

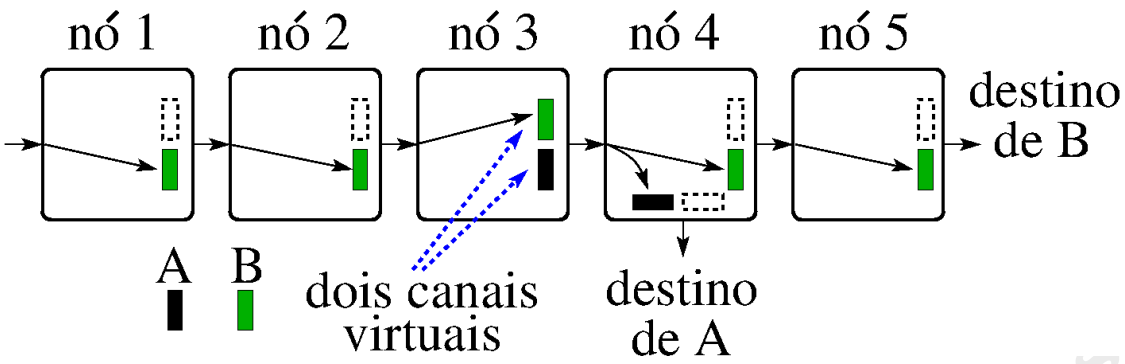

Figura 2 – Com o uso de canais virtuais, o pacote B pode seguir sem ser bloqueado por A
Fonte: Adaptado de DALLY (1992)

Revisão da literatura

Há muitos trabalhos sobre canais virtuais explorando como melhorar o desempenho e a taxa de transferência nas *NoCs*. Dally (1992) propõe canais virtuais nos quais o primeiro *flit*, ao chegar ao *buffer*, aloca para si um canal virtual, deixando os outros canais virtuais disponíveis para *flits* de outros pacotes que chegarem. Entretanto, neste tipo de alocação, conforme há um aumento de tráfego na rede, há uma tendência de haver a possibilidade de ocorrência de impasses. Já Nicopoulos (2006) propõe uma alocação dinâmica, combinando conceitos de *buffers* multifilas dinamicamente alocadas (*dynamically allocated multi-queue*, *DAMQ*) (JAMALI, 2009), conforme o tráfego da rede aumenta. Entretanto, esta alocação não escala muito bem em diferentes tipos de tráfego. Assim como Nicopoulos (2006), Lai (2008) propõe canais virtuais dinamicamente alocados, combinando os canais virtuais em quantidade e tamanho, e variando-os de acordo com o tráfego na rede. Xu (2010) propõe o uso de um pré-roteamento para alocar cada canal virtual. No entanto, este pré-roteamento insere, no sistema, uma redundância em relação à função do roteador, não sendo a melhor solução do ponto de vista de uso de área ocupada por hardware.

O presente trabalho tem como um dos objetivos apresentar um algoritmo, não encontrado na literatura, de escolha de em qual canal virtual colocar cada pacote, baseado no tempo de permanência dos pacotes anteriores em cada canal virtual, e verificar e analisar seu desempenho. Outro objetivo deste trabalho é analisar o efeito que os canais virtuais podem trazer em redes com chaveamento do tipo *wormhole*.

Metodologia

Para a realização deste trabalho, foi utilizada a SystemC, uma biblioteca de classes e de macros da linguagem C++ (PANDA, 2001; IEEE, 2012), que a estendem de forma a torná-la semelhante a linguagens de descrição de hardware, entrando, porém, na categoria de linguagem de modelagem em nível de sistema. A escolha da utilização da SystemC neste trabalho se deu devido ao fato de a rede intrachip que foi fornecida pelo grupo de pesquisa estar totalmente descrita com base nesta biblioteca. Ela permite a realização de simulações do comportamento de hardware digital, com precisão temporal de ciclos de relógio.

Definida a linguagem a ser utilizada no projeto, foi necessário o estudo dos conceitos de canais virtuais e trabalhos relacionados. Nesta etapa, percebeu-se, na literatura, a ausência de um algoritmo de decisão de colocação de pacotes em canais virtuais baseado no tempo de permanência de pacotes anteriores recentes em cada canal. Por este algoritmo, cada pacote que chega à entrada de um módulo com canais virtuais é inserido no canal que tiver o melhor fluxo recente de pacotes. Desta forma, um canal bloqueado há muito tempo por algum pacote prévio é evitado pelos pacotes seguintes.

Definiu-se que, em cada roteador da rede, formado por um módulo comutador e por um módulo contendo canais virtuais, este último seria composto por três tipos de submódulos: *buffers* (implementando os canais virtuais), um seletor de entrada e um multiplexador (mux) de saída, conforme a Figura 3. O módulo seletor desempenha o papel de interface com os módulos externos fornecedores dos pacotes. Ele seleciona o canal ao qual cada pacote entrando no roteador deve ser encaminhado. Para esta seleção, este módulo é o responsável por comparar, entre si, os tempos de permanência dos pacotes mais antigos em cada canal. O seletor escolhe o canal no qual esse tempo é o menor, o que indica que aquele está com melhores condições de fluxo, recentemente. A implementação dessa estratégia requer que cada *buffer* tenha associado a ele uma fila contendo os valores dos tempos de entrada de todos os pacotes presentes no canal. Essa fila caminha paralelamente aos pacotes. Um terminal de saída (em verde na Figura 3) do módulo que implementa a fila informa ao seletor, a cada instante, o tempo de permanência do pacote mais antigo no *buffer* (próximo pacote a sair) desde a sua chegada. Desta forma, mais um terminal de comunicação passa a integrar o *buffer*, além daqueles de entrada e de saída de pacotes (em vermelho na Figura 3) e daqueles de protocolo (em azul na mesma Figura).

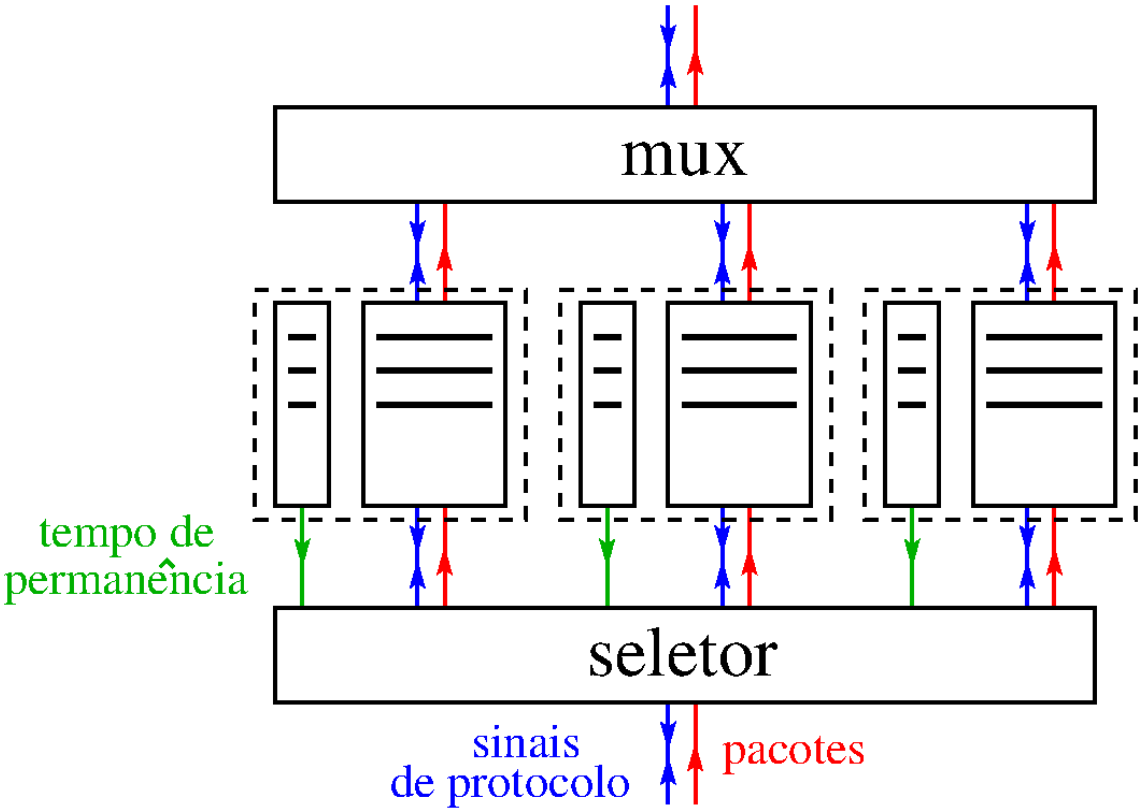

Figura 3 – Diagrama de blocos do módulo contendo canais virtuais, desenvolvido neste trabalho. Neste exemplo, há três canais virtuais (contidos nos retângulos em tracejado)
Fonte: Autores

Por fim, há o módulo multiplexador (mux), o qual, a cada instante, seleciona o canal virtual de onde será retirado um pacote a ser apresentado à saída. Neste projeto, o mux varre as suas entradas (ligadas às saídas dos canais), transmitindo os pacotes em espera em cada canal, um por vez, desde que os caminhos a serem seguidos adiante por eles estejam livres.

Os canais virtuais (*buffers*) são parametrizados pelo projetista, em termos de seu número e das suas capacidades, ou seja, de quantos pacotes cabem em cada canal.

Definidas as especificações do sistema, iniciou-se o desenvolvimento do código em SystemC, a começar pela descrição dos canais. Para avaliar o funcionamento dos módulos construídos durante esta etapa, criou-se um trecho mínimo de rede, constituindo-se de um gerador de pacotes, um *buffer* e um módulo de leitura.

Verificado o correto funcionamento, em separado, dos módulos desenvolvidos, construiu-se um módulo hierárquico, chamado VC (*virtual channel*), contendo, reunidos, todos os módulos indicados na Figura 3. Com isso, o desenvolvimento do módulo dos canais virtuais estava completo.

A seguir, iniciou-se a edição do código da rede em SystemC, com a finalidade de inserir nela os canais virtuais. Ela consistiu, conforme a Figura 4, em um sistema contendo 16 roteadores (indicados com R), em arranjo matricial 4 x 4, no qual as duas linhas acima contêm processadores (indicados com P) e as duas linhas abaixo contêm módulos de memória (indicados com M).

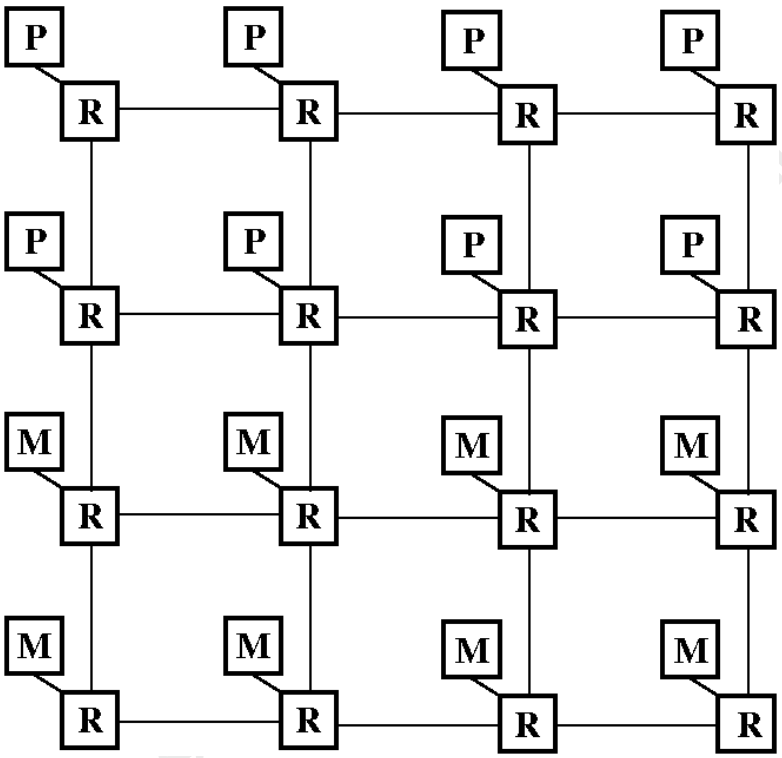

Figura 4 – Diagrama de blocos da rede *intrachip* usada nas simulações. Módulos de memória são indicados com M, processadores com P e roteadores com R
Fonte: Autores

Quanto mais intenso é o tráfego de pacotes numa região da rede, maior é a probabilidade de ocorrência de congestionamento nela. Por isso, para se obterem resultados que pudessem revelar bem a eficácia do uso de canais virtuais em relação ao uso de *buffers* simples nos roteadores, foram realizadas simulações com tráfego cada vez mais intenso e mais concentrado numa região. Com essa finalidade, o código da rede foi editado, de modo a fazer com que vários processadores se comunicassem com um único módulo de memória.

Tendo os canais virtuais inseridos na rede, a próxima etapa foi a definição dos parâmetros para realização das simulações. Foi decidido que os fatores a serem analisados seriam: o tempo médio para que os pacotes alcançassem seus destinos; o tempo máximo; e, por fim, o tempo mínimo. Esses fatores foram decididos para verificar a eficiência da rede em termos de

velocidade, comparando-se os resultados obtidos quando se usam canais virtuais com os resultados obtidos quando se usam *buffers* simples (ou seja, neste caso, sem canais paralelos nos roteadores). Os parâmetros dos testes consistiam em: tamanho do *buffer* (no caso de uso de *buffer* simples, em lugar de canais virtuais); número de canais e capacidade dos canais (no caso dos virtuais), taxa de injeção de pacotes pelos geradores (parâmetro próprio da rede, baseado em um algoritmo gerador de números aleatórios, podendo ser ajustado de 0 a 100% dos ciclos de relógio do sistema), e tempo de simulação. Em cada experimento, foram adotados valores iguais de parâmetros para todos os módulos de mesmo tipo.

Planejados os fatores e parâmetros a serem levados em conta nas simulações, desenvolveu-se um *script* (CANNON, 2015) para filtrar os resultados relevantes da simulação e, automaticamente, calcular os valores médio, máximo e mínimo dos tempos citados no parágrafo anterior. O *script* acionou o programa Octave (GNU OCTAVE, 2017) para a realização desses cálculos.

Terminada a criação do *script* para a análise dos resultados, iniciou-se a fase de simulações. Para cada conjunto de valores de parâmetros definidos, foram realizadas cinco simulações e foi calculada a média dos valores de cada fator avaliado. Esse procedimento foi necessário, a fim de se obterem resultados mais significativos estatisticamente do que se não houvesse as repetições.

Ainda nesta etapa, decidiu-se comparar o modelo de canais virtuais aqui proposto, usando algoritmo de escolha de canal baseado no tempo de permanência, com aquele proposto por Dally (1992), aqui chamado de "tradicional", a fim de se verificar se houve melhoria de um modelo para outro. Com este intuito, a partir da descrição dos canais virtuais desenvolvidos, obteve-se uma nova versão destes, seguindo o modelo tradicional. Para ele, ficou convencionado que os pacotes seriam alocados sequencialmente, ou seja, cada pacote, ao chegar, seria alocado a um canal diferente, sequencialmente (ignorando-se os tempos de permanência de pacotes anteriores nos canais), e, quando se chegasse ao último canal, o ciclo se repetiria.

Resultados obtidos

Nos experimentos, o tempo de simulação foi de 5000ns, com cada ciclo de relógio tendo a duração de 1ns. Foi escolhido esse número porque valores muito menores do que esse causam maior variabilidade nos resultados de uma simulação para outra, ao passo que tempos de simulação da ordem de grandeza usada levam a uma maior convergência dos resultados entre simulações com mesmos valores de parâmetros. A taxa de injeção de pacotes foi variada de 50 a

100%. Embora fosse necessário analisar o desempenho dos módulos sob um tráfego intenso, era necessário também observar o comportamento destes em taxas de injeção relativamente baixas, a fim de se ter uma maior variedade de condições para as comparações entre os desempenhos dos módulos. A capacidade de cada canal foi escolhida buscando-se usar valores relativamente baixos, pois, embora houvesse um grande tráfego na rede, com um valor de capacidade muito grande, os canais e o *buffer* não seriam devidamente testados, enquanto seus limites de capacidade não fossem exercitados. Ficou definido, então, que a capacidade de armazenamento à entrada de cada módulo de entrada (*buffer* simples ou CV) de cada roteador seria de quatro pacotes. Para um maior grau de confiança nas comparações, os canais virtuais também tiveram como tamanho limite o valor de quatro pacotes, dividido entre os canais. Caso o resultado dessa divisão não fosse inteiro, seria adotado o inteiro imediatamente inferior.

Foram testados os módulos: o *buffer* simples, com capacidade para 4 pacotes, os canais virtuais, numa versão com dois canais com capacidade de dois pacotes por canal e três canais com capacidade de um pacote por canal e, por fim, o modelo tradicional adaptado a rede com canais virtuais, que seguiram o mesmo esquema de teste dos canais desenvolvidos.

Os resultados obtidos para o tempo máximo, mínimo e médio para um pacote chegar ao seu destino são mostrados na Figura 5, na Figura 6 e na Figura 7, respectivamente. Nessas figuras, os canais virtuais aqui desenvolvidos foram chamados de VCN2 e VCN3, para as versões com dois e três canais, respectivamente, enquanto os canais tradicionais adaptados foram chamados de VC2 e VC3.

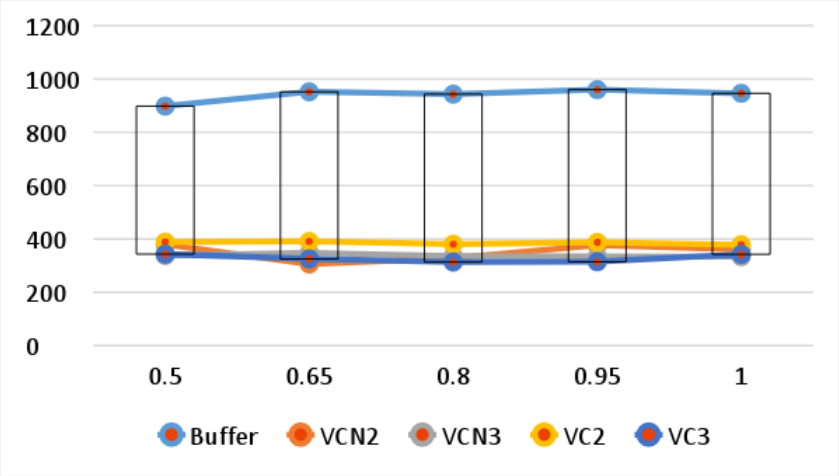

Figura 5 – Tempo máximo de um pacote na rede – tempo (ns) x taxa de injeção
Fonte: Autores

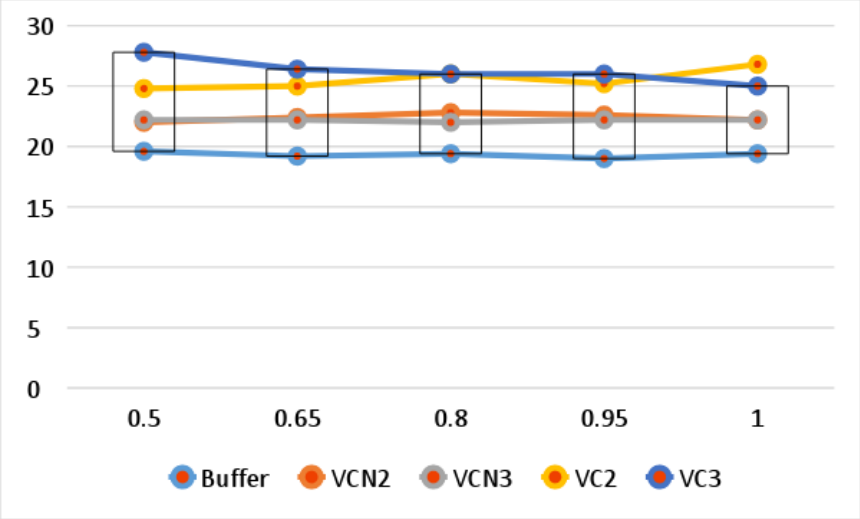

Figura 6 – Tempo mínimo de um pacote na rede – tempo (ns) x taxa de injeção
Fonte: Autores

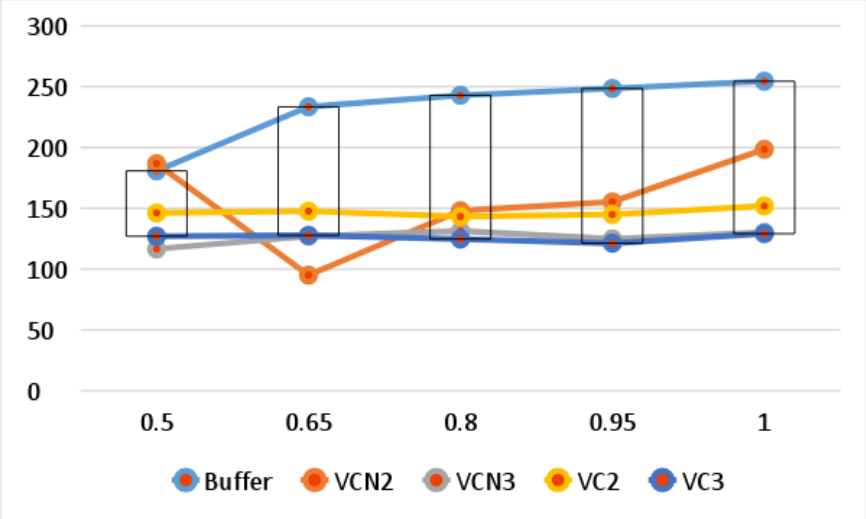

Figura 7 – Tempo médio de um pacote na rede – tempo (ns) x taxa de injeção
Fonte: Autores

Como pode ser observado nos gráficos, o tempo máximo de um pacote na rede utilizando canais virtuais é cerca de 60% menor do que em uma rede com *buffers* simples. Do tempo máximo, podem-se interpretar valores altos como devidos a ocorrência de *livelocks* ou *starvation*, que ocorrem quando um pacote vaga na rede sem fazer progresso em direção ao seu destino, ou fica por muito tempo preso nos *buffers*, respectivamente. Embora o mesmo não possa ser dito em relação ao tempo mínimo, a diferença de tempo mínimo entre a versão com canais virtuais e a versão com *buffer* simples pode ser explicada pelo fato de que os pacotes devem cruzar mais blocos lógicos (seletores e multiplexadores) em uma rede com canais virtuais do que com canais simples e, portanto, são necessários os correspondentes pulsos de

relógio para que o pacote alcance seu destino. No entanto, pode-se dizer que essa diferença é irrelevante ao se tratar da velocidade da rede, uma vez que, embora o tempo mínimo de entrega de pacote, em alguns casos, seja menor no caso do uso de *buffers* simples, não muda o fato de que este ainda apresenta um tempo médio muito maior em relação aos casos em que foram usados canais virtuais. Com relação ao tempo médio, é nítida a diferença entre os tempos médios de entrega entre o caso com *buffer* simples e os casos com canais virtuais. Pode-se dizer que os casos com canais virtuais apresentam um desempenho cerca de 50% melhor do que o obtido utilizando *buffer* simples. A partir disso, pode-se verificar que o uso de canais virtuais em redes intrachip melhora a sua eficiência e o seu desempenho.

Com relação à diferença de desempenho, em termos de tempo médio, entre as duas formas de implementação de canais virtuais, percebe-se que, na Figura 7, a implementação VC2 (tradicional) foi pior do que as versões com três canais virtuais (VC3 e VCN3), para todas as taxas de injeção de pacotes usadas. O melhor dentre todos os resultados de desempenho, em tempo médio, foi obtido pela versão VCN2, mas apenas para a taxa de injeção de 0,65. Para outras, ela foi inferior às demais implementações com canais virtuais, mas sempre superior à versão sem canais virtuais (*buffer* simples). Dentre as versões com três canais virtuais (VC3 e VCN3), houve variação com o valor da taxa de injeção quanto a qual delas teve o melhor desempenho.

Conclusão

Verificou-se, pelas simulações, a eficácia do emprego de canais virtuais em relação ao uso de *buffers* simples nos roteadores de uma rede intrachip. Um módulo que implementa canais virtuais foi desenvolvido, descrito em SystemC e testado com sucesso.

Quanto ao algoritmo aqui proposto de escolha do canal virtual com base no tempo de permanência recente de pacotes nos canais, este se mostrou superior ao algoritmo tradicional para alguns valores de taxa de injeção de pacotes na rede e inferior para outros valores. Assim, sua adoção deve ser avaliada caso a caso, com base nas características de tráfego do sistema sendo projetado.

Referências

ASLAM, M.A.; KUMAR, S.; HOLSMARK, R. **An Efficient Router Architecture and Its FPGA Prototyping to Support Junction Based Routing in NoC Platforms, Proceedings of Euromicro Conference on Digital System Design**, Los Alamitos, CA, Estados Unidos, 2013. p. 297-300.

BENINI, Luca; DE MICHELI, Giovanni. Networks on chip: a new paradigm for systems on chip design. In: **Design, Automation and Test in Europe Conference and Exhibition**, 2002. Proceedings. IEEE, 2002. p. 418-419.

CANNON, J. **Shell Scripting. Create Space Independent Publishing Platform**, ISBN-10 151738043X, 2015.

CARARA, E.; MORAES, Fernando Gehm. **Uma Exploração Arquitetural de Redes Intrachip com Topologia Malha e Modo de Chaveamento Wormhole.** 2007. Tese de Doutorado. Dissertação de Mestrado, PPGCC-FACIN-PUCRS, Porto Alegre.

DALLY, William J. Virtual-channel flow control. **IEEE Transactions on Parallel and Distributed systems**, v. 3, n. 2, 1992. p. 194-205.

GNU OCTAVE. **Scientific Programming Language.** Disponível em <https://www.gnu.org/software/octave/>. Acesso em dez. 2017.

GUPTA, Rajesh K.; ZORIAN, Yervant**.** Introducing core-based system design. **IEEE Design & Test of Computers**, v. 14, n. 4, 1997. p. 15-25.

IEEE Standards Association. IEEE Standard for Standard SystemC Language Reference Manual. **IEEE Computer Society**, New York, EUA, 2012

JAMALI, Mohammad Ali Jabraeil; KHADEMZADEH, Ahmad. Improving the performance of interconnection networks using DAMQ buffer schemes. **IJCSNS**, v. 9, n. 7, 2009. p. 7.

LAI, Mingche *et al.* A dynamically-allocated virtual channel architecture with congestion awareness for on-chip routers. In: **Design Automation Conference**, 2008. DAC 2008. 45th ACM/IEEE. IEEE, 2008. p. 630-633.

LEE, Hyung Gyu *et al***.** On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, v. 12, n. 3, 2007. p.23.

MITIĆ, Milica; STOJČEV, Mile. A survey of three system-on-chip buses: AMBA, coreconnect and wishbone. In: **Proc. 41st Int. Conf. Inform. Commun**. Energy Syst. Technol.(ICEST). 2006. p. 282-285.

MORAES, Fernando *et al*. HERMES: an infrastructure for low area overhead packet-switching networks on chip. **INTEGRATION, the VLSI journal**, v. 38, n. 1, 2004. p. 69-93.

NICOPOULOS, Chrysostomos A. *et al*. ViChaR: A dynamic virtual channel regulator for network-on-chip routers. In: **Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture**. IEEE Computer Society, 2006. p. 333-346.

PANDA, P.R. SystemC: A Modeling Platform Supporting Multiple Design Abstractions. In: **Proceedings of the 14th International Symposium on Systems Synthesis**, Montreal, Canada, ISBN 1-58113-418-5, ACM, New York, EUA, 2001. p. 75-80.

XU, Yi *et al*. Simple virtual channel allocation for high throughput and high frequency on-chip routers. In: **High Performance Computer Architecture (HPCA)**, 2010 IEEE 16th International Symposium on. IEEE, 2010. p. 1-11.

DESIGN AND VALIDATION OF VIRTUAL CHANNELS ON A NETWORK ON CHIP

ABSTRACT
This paper aims to present the design and validation of the so-called virtual channels within NoCs routers, in addition to present a new proposal for a decision of placing packages in these channels, based on the recent length of staying time of the packages. The design has been performed using the SystemC library to implement the modules and to simulate the system. Three types of modules have been designed to serve as components of the virtual channels. Initially they have been designed and tested in isolation. After their operation had been verified, they were included in a network and tested again, with the traffic results being compared using virtual channels and using simple buffers. Satisfactory results have been obtained regarding the time it took to the packages arrive at their destinations. This made it possible the verification of the efficiency in using virtual channels in the proposed implementation.
Keywords: *Network on chips. Virtual channels. SystemC.*

Envio: dezembro/2017
Aceito para publicação: março/2018