Avaliação de Desempenho de Rede-em-Chip Modelada em SystemC

Cesar Albenes Zeferino[1], Jaison Valmor Bruch[1], Thiago Felski Pereira[1]
Márcio Eduardo Kreutz[2], Altamiro Amadeu Susin[3]

[1] Universidade do Vale do Itajaí – CTTMar – MCA – GSED
Rodovia SC 407, Km 04 – Bairro Sertão do Maruim – 88122-000 – São José – SC

[2] Universidade de Santa Cruz do Sul – Departamento de Informática – GPSEM
Av. Independência, 2293 – Bairro Universitário – 96815-900 – Santa Cruz do Sul –RS

[3] Universidade Federal do Rio Grande do Sul – II – PPGC
Av. Bento Gonçalves, 9500 – 91501-970, Porto Alegre – RS

{jaison,felski,zeferino}@univali.br;
kreutz@unisc.br; altamiro.susin@ufrgs.br

Abstract. *Future technologies for integrated circuits will drive systems-on-a-chip (SoCs) construction comprised of several dozens of cores. Communication architectures based on busses architecture will not face communications performance constraints needed to these SoCs efficiency. As a solution, Networks-on-Chip (NoCs) has been considered as the best solution to meet dedicated applications requirements. Since the design space of NoCs is wide and thus, requires a specific infrastructure for exploration, this work presents the modeling of a parameterized NoC architecture by using SystemC, at the RT level, together with a set of CAD tools concerning to evaluate the performance of NoCs architectures under different configurations and traffic patterns. Also communication modeling for traffic of real applications based on the characterization of their traffic is presented. Results show the configuration of network that allows obtaining the best performance metrics.*

Resumo. *As próximas tecnologias de fabricação de circuitos integrados permitirão a integração de dezenas de núcleos em um mesmo chip. As arquiteturas de comunicação baseadas no barramento não atenderão às demandas de desempenho desses sistemas. Como solução, é consenso na literatura que as Networks-on-Chip (NoCs) oferecerão a melhor solução em desempenho em comunicação. Como o espaço de projeto de NoCs é bastante amplo e requer infra-estrutura específica para a sua exploração, este trabalho, apresenta a modelagem de uma arquitetura de NoC parametrizável utilizando o SystemC no nível RT e um conjunto de ferramentas de apoio que permitem avaliar o desempenho da rede sob diferentes configurações do seu espaço de projeto e condições de tráfego, possibilitando, também, realizar a modelagem do perfil de comunicação de aplicações reais a partir da caracterização de seu tráfego. Os resultados obtidos evidenciam a configuração de rede que apresenta os melhores índices de desempenho.*

1. Introdução

O advento das tecnologias de processo submicrônico tem permitido o projeto de sistemas completos em um único chip, os quais são denominados Sistemas Integrados ou SoCs (Systems-on-Chip). As metodologias de projeto de sistemas integrados são baseadas no reuso de blocos de circuito em silício pré-projetados e pré-verificados, contendo, usualmente, pelo menos 5.000 portas lógicas [Gupta e Zorian 1997]. Esses blocos reutilizáveis, que recebem o nome de núcleos (ou *cores*), podem ser aplicados na construção de diferentes sistemas. A capacidade de reuso dos mesmos é vital para minimizar o tempo de projeto e de verificação de novos produtos e garantir o aumento da competitividade e dos rendimentos de seus fabricantes.

Os núcleos de um sistema integrado são interligados por meio de uma arquitetura de comunicação, sendo que, atualmente, devido a sua reusabilidade, a arquitetura preferencial é o barramento. Contudo, embora o barramento seja adequado a sistemas com poucas dezenas de núcleos, ele não atenderá aos requisitos dos futuros SoCs, os quais integrarão de várias dezenas a centenas de núcleos. Essa inadequação deve-se ao fato de que sua freqüência de operação e sua largura de banda se degradam com o aumento do número de núcleos a ele conectados, e também porque um barramento suporta apenas uma única comunicação a cada instante, entre outros fatores [Guerrier e Greiner, 2000].

Dadas as limitações dos barramentos, diversos pesquisadores têm defendido o uso de arquiteturas de comunicação denominadas Redes-em-Chip ou NoCs (Networks-on-Chip) [Jantsch e Tenhunen, 2003], que são redes chaveadas integradas em chip baseadas nas arquiteturas de redes de interconexão para computadores paralelos.

Entre as vantagens das NoCs sobre os barramentos destacam-se: (i) a largura de banda das NoCs cresce com o aumento do número de núcleos, enquanto que a do barramento é fixa; e (ii) as NoCs oferecem paralelismo em comunicação e o barramento não. Além disso, as NoCs possuem capacidade de reuso equivalente à do barramento e são próprias para sistemas GALS (Global Assynchronous, Local Synchronous) em que o chip é divido em regiões síncronas que se comunicam de maneira assíncrona [Hermani et al. 1999].

No entanto, apesar de suas vantagens, as NoCs também apresentam limitações e desvantagens, sendo que a principal delas consiste na latência de comunicação. Como os núcleos são separados por um ou mais roteadores, cada comunicação é sujeita aos atrasos inerentes dos roteadores e à competição por canais da rede. Outros problemas que se destacam são o seu sobrecusto de silício e de energia.

Dado esse cenário, desde o final dos anos 90, diversos projetos de pesquisa têm sido desenvolvidos no sentido de abordar um ou mais problemas potencias das NoCs. O Projeto SoCIN (System-on-Chip Interconnection Network), iniciado na Universidade Federal do Rio Grande do Sul – UFRGS e hoje realizado em pareceria entre pesquisadores da UFRGS, da Universidade do Vale do Itajaí – UNIVALI e da Universidade de Santa Cruz do Sul – UNISC, objetiva a exploração de arquiteturas de NoCs com baixo custo de silício, visando à realização de sistemas embarcados escaláveis com alta demanda de comunicação. Nesse projeto, foi desenvolvida uma arquitetura de roteador parametrizável denominado ParIS (Parameterized

Interconnection Switch). O modelo VHDL do roteador ParIS permite a simulação e síntese de redes em malha 2-D com diferentes configurações de mecanismos de roteamento, controle de fluxo, arbitragem e memorização (*buffers* de entrada e de saída com profundidade parametrizável).

Essas alternativas de configuração tendem a apresentar índices de desempenho diferenciados em função das características do tráfego apresentado à rede. Para avaliar o desempenho de cada configuração de rede sob diferentes padrões de tráfego, é necessário realizar a simulação de sistemas reais ou sintéticos, em que núcleos ou geradores de tráfego sejam conectados à rede e injetem pacotes a taxas predeterminadas e que o tráfego da rede seja analisado a partir de dados coletados por unidades de medição.

Embora o modelo VHDL do roteador ParIS permita tanto a simulação quanto a síntese, a simulação de redes descritas em VHDL pode apresentar elevado custo computacional no que diz respeito ao tempo de simulação, conforme o tamanho do sistema. Além disso, a construção de geradores e analisadores de tráfego voltados à execução nos mesmos ambientes de simulação e de síntese da rede é limitada ao subconjunto do VHDL suportado pelo ambiente de desenvolvimento utilizado. Isso, por sua vez, restringe a generalidade desses modelos.

Essas limitações levam à necessidade do uso de outras alternativas de modelagem que permitam caracterizar melhor o desempenho de cada configuração da rede em um tempo computacional menor, a fim de facilitar a exploração do espaço de projeto.Visando resolver esse problema, e ainda mantendo a precisão em nível de ciclos e assegurando a possibilidade de realizar a síntese da rede, foi realizada a modelagem do roteador ParIS e da rede SoCIN em SystemC no nível RT (Register Transfer). O modelo implementado permite avaliar o desempenho da rede sob diferentes configurações de parâmetros e condições de tráfego.

Experimentos de simulação e avaliação de desempenho são suportados por um conjunto de ferramentas para automatizar a geração e a execução de *benchmarks* para a rede SoCIN. Esses *benchmarks* são baseados em um modelo de gerador de tráfego que possibilita a representação de diferentes padrões de tráfego, configurando-se a distribuição espacial das comunicações realizadas, bem como parâmetros referentes à carga de comunicação oferecida à rede. Modelos para medir o tráfego e ferramentas de análise de dados coletados permitem obter métricas de desempenho como, por exemplo, a latência média da rede e o tráfego aceito para uma dada carga oferecida, entre outras.

O artigo é organizado como segue. Na Seção 2, são apresentados conceitos sobre SystemC e trabalhos relacionados. A arquitetura da rede SoCIN e a organização do roteador ParIS são brevemente descritos na Seção 3. Na Seção 4, são descritos aspectos referentes à modelagem e ao fluxo de simulação da rede SoCIN usando o SystemC, e são apresentados os modelos de componentes e as ferramentas desenvolvidas para suportar a avaliação de desempenho da rede. Na Seção 5, são mostrados resultados de experimentos realizados buscando avaliar o impacto da variação da configuração rede sob diferentes padrões de tráfego. Finalizando, a Seção 6 apresenta conclusões e comentários adicionais referentes a pesquisas em andamento e a trabalhos futuros.

2. Conceitos e Trabalhos Relacionados

2.1 O SystemC

O SystemC [OSCI 2005] é uma linguagem de modelagem padrão que visa habilitar o projeto no nível de sistema e permitir o intercâmbio de núcleos em múltiplos níveis de abstração para sistemas contendo componentes de hardware e de software. Segundo Swan (2001), ele é completamente baseado em C++ e o seu padrão controlado por um grupo de trabalho composto por mais de trinta companhias de automação do projeto eletrônico (o OSCI – Open SystemC Initiative). Na versão 2.0, o SystemC foi melhorado de modo a permitir a modelagem em nível de sistema, incluindo sistemas que poderiam ser modelados em software, em hardware ou na combinação de ambos. Essa melhoria visa oferecer uma linguagem de projeto útil em uma larga faixa de modelos de computação, níveis de abstração e metodologias de projeto de sistema.

O SystemC também permite a modelagem no nível RT com precisão em nível de ciclos de relógio. Modelos de simulação nesse nível de abstração podem ser facilmente convertidos para modelos sintetizáveis com a inclusão de diretivas de compilação específicas para o compilador disponível. Um exemplo é o Co-Centric SystemC Compiler RTL da Synopsys (2003).

2.2 Avaliação de Desempenho de Redes-em-Chip

A avaliação de desempenho de Redes-em-Chip é um campo que começou a ser explorado recentemente na literatura. Nos primeiros trabalhos, os experimentos de avaliação de desempenho eram feitos utilizando simulações com base em modelos VHDL [Moraes et al., 2003]. Posteriormente, passou-se a utilizar o nível de sistema, com o uso de simuladores para Redes-em-Chip dedicados [Bolotin et al., 2004] ou de simuladores de sistema não dedicados[Andriahantenaina et al., 2003]. Por exemplo, Kreutz et al. (2005) utilizam um simulador escrito em C++ que descreve a arquitetura dos componentes internos de roteadores de NoCs no nível de transação – cada componente é descrito como uma classe C++ cujos métodos representam as suas operações. A precisão ocorre em nível de ciclos e a cada período do relógio todos os componentes da rede são executados.

O SystemC começou a ser adotado recentemente e está se tornando o padrão de fato nesse tipo de investigação. Alguns argumentos favoráveis ao uso do SystemC são apresentados por Grant Martin (2003). Entre eles destaca-se o fato de que os sistemas estão se tornando cada vez mais complexos, sendo que a modelagem em níveis de abstração mais altos que o RT (nível de sistema, por exemplo) nas etapas iniciais do desenvolvimento, facilita a exploração do espaço de projeto, ao mesmo tempo que reduz o tempo para verificação do funcionamento do sistema. Além disso, é necessário que o fluxo de projeto contemple não apenas a modelagem de hardware, mas, também, a de software, pois, nos sistemas computacionais embarcados, o software exerce um papel importante no custo e no tempo de desenvolvimento do sistema. O SystemC permite a modelagem tanto de hardware como de software, atendendo a esses requisitos.

É justamente nessa direção que a comunidade de pesquisadores em Redes-em-Chip está mirando de modo a adequar seus trabalhos ao padrão já estabelecido. Por exemplo, Chan e Parameswaran (2004) apresentam uma Rede-em-Chip modelada em

VHDL e com desempenho avaliado através de uma simulação híbrida SystemC/VHDL, na qual os geradores de tráfego são modelados em SystemC (no nível de sistema) e o roteador é modelado em VHDL (no RTL). Essa abordagem facilita o desenvolvimento do gerador de tráfego, pois permite o uso de estruturas da linguagem C++ que são mais adequadas para a implementação desse tipo de componente, o qual deve injetar pacotes na rede e colher os dados estatísticos necessários para a avaliação do seu desempenho. Contudo, essa abordagem híbrida ainda apresenta custo computacional razoável no que tange ao tempo para a execução da simulação.

Também é possível realizar uma modelagem híbrida totalmente em SystemC, com custo computacional de simulação intermediário. Utilizando-se um subconjunto de primitivas do SystemC, é possível modelar alguns componentes de um sistema no RTL e outros em níveis mais abstratos. Além disso, diversos fabricantes de ferramentas de automação do projeto eletrônica (EDA – Electronic Design Automation), como a Synopsys (2003) e a Forte Design Systems (2007) já oferecem ferramentas que traduzem um modelo SystemC para uma linguagem de descrição de hardware sintetizável, como o Verilog e o VHDL.

3. Arquitetura da Rede SoCIN

3.1 Características da rede SoCIN

A rede SoCIN é uma NoC que utiliza uma topologia em malha (*mesh*) 2-D. Cada roteador possui uma porta de comunicação, denominada Local ou L (Fig. 1), à qual pode ser conectado um núcleo ou um subsistema. Além dessa, possui também de duas a quatro portas de comunicação para conexão com roteadores vizinhos, conforme a posição do roteador na rede. Essas portas são chamadas de North (superior), East (à direita), South, (inferior) e West (à esquerda), ou, simplesmente, N, E, S e W. Os roteadores são endereçados por um sistemas de coordenadas XY (Coluna, Linha). Quando um núcleo (ou nodo) conectado a um roteador deseja enviar um pacote a um núcleo conectado a outro roteador, ele deve incluir as coordenadas do destinatário (e as suas) no cabeçalho do pacote para que os roteadores da rede possam definir o caminho a ser tomado (e para que o destinatário saiba quem enviou o pacote).

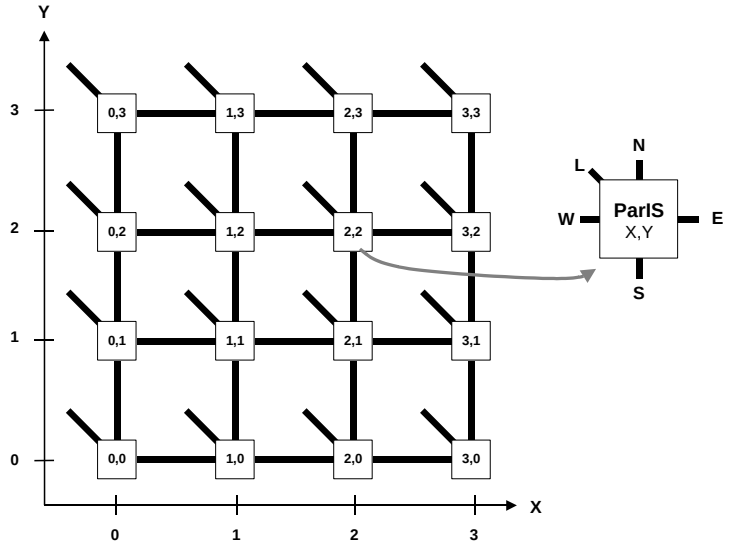

Figura 1. Topologia da rede SoCIN

3.2 Organização do roteador ParIS

O roteador ParIS possui arquitetura canônica parametrizável. Internamente, a arquitetura é composta por módulos que podem ser estendidos para suportar diferentes alternativas de mecanismos de comunicação. Atualmente, são disponibilizadas as seguintes alternativas:

- Chaveamento: wormhole;
- Controle de fluxo: handshake e baseado em créditos;
- Memorização: *buffer* FIFO (na entrada e, opcionalmente, na saída);
- Roteamento: XY e WF (West-First); e
- Arbitragem: estática e dinâmica (randômica e circular).

O roteador, ilustrado na Fig. 2, é organizado em módulos de entrada (ex. Lin) e de saída (ex. Lout) e por um módulo central (CPM) que representa a estrutura de conexões relacionada com o tipo de roteamento implementado (ex. XY ou West-First).

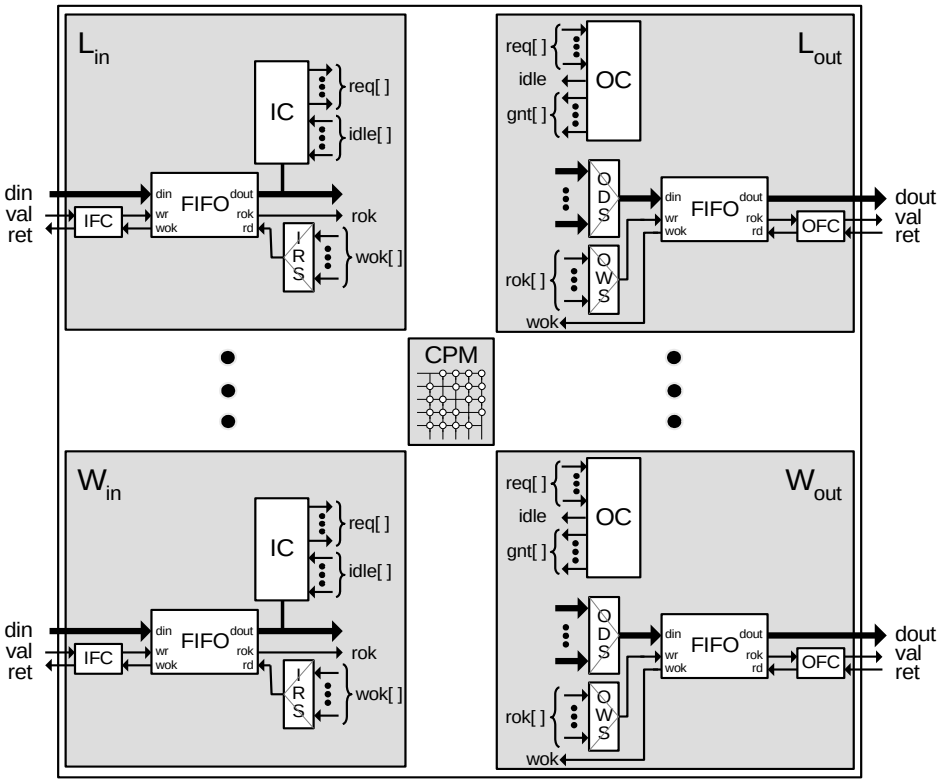

Figura 2. Organização do roteador ParIS

O módulo de entrada é constituído por blocos de controle de fluxo (IFC – Input Flow Controller), armazenamento (*buffer* FIFO – First-In, First-Out), roteamento (IC – Input Controller) e de chaveamento (IRS – Input Read Switch). O módulo de saída é composto por um árbitro (OFC – Output Flow Controller), chaveadores (ODS – Output Data Switch e OWS – Output Write Switch), armazenamento de saída (opcional) e controle de fluxo de saída (OFC – Output Flow Controller).

Quando um pacote é enviado a um canal de entrada *i* do roteador, o bloco IFC utiliza um protocolo de dois fios (*val* e *ret*) para regular a admissão dos *flits* (*flow control units*) presentes no canal de dados (*din*) de acordo com o espaço disponível no *buffer* de entrada. Quando o cabeçalho do pacote atinge a saída do *buffer* de entrada, ele é roteado pelo bloco IC que seleciona o canal de saída *j* a ser utilizado para encaminhá-lo em direção ao seu destino. O bloco IC envia uma requisição (*req*) ao bloco OC do canal de saída *j*, o qual realiza a arbitragem de múltiplas requisições, selecionando uma delas e comandando os chaveadores (IRS, ODS e OWS) para estabelecer a conexão entre o canal de entrada selecionado e o *buffer* de saída. Os *flits* recebidos por esse *buffer* são encaminhados para fora do roteador, sendo que a emissão dos *flits* é controlada pelo bloco OFC. Esse *buffer* é um bloco opcional do circuito.

Mais detalhes sobre o roteador ParIS e a sua implementação em VHDL são fornecidos por Zeferino, Santo e Susin (2004).

4. Implementação da Rede SoCIN em SystemC

4.1 Modelagem da rede SoCIN no SystemC

A implementação da rede SoCIN em SystemC segue uma abordagem híbrida, modelando-se o roteador ParIS em SystemC RTL e os módulos de geração (TG) e de medição (TM) de tráfego no nível de sistema, o que permitiu uma maior flexibilidade para utilização de primitivas nativas da linguagem C.

Na implementação de um sistema baseado no roteador ParIS, no gerador de tráfego (TG) e no medidor de tráfego (TM), um par TG-TM é conectado à porta Local de um roteador, conforme é ilustrado na Fig. 3. O TG é responsável pela geração de tráfego enviando pacotes a outros TGs e recebendo pacotes enviados por outros TGs. O TM monitora a chegada de pacotes na porta Local e registra dados que permitem calcular a latência de cada pacote, bem como outras informações relevantes. O modelo do TM é genérico o suficiente para que possa ser utilizado para medir dados referentes a pacotes no momento em que são injetados na rede ou que passam por enlaces entre roteadores.

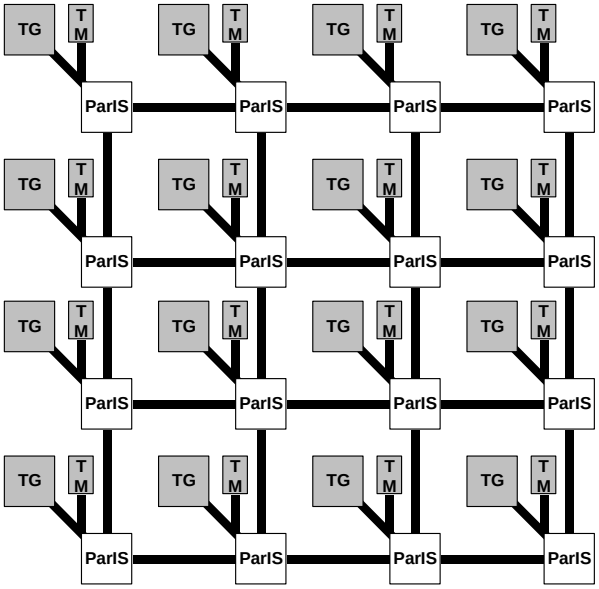

Figura 3. Plataforma para avaliação de desempenho da rede SoCIN

4.2 Modelos SystemC

O modelo SystemC do roteador ParIS foi totalmente implementado no nível RT e reproduz a arquitetura modelada originalmente em VHDL (ilustrada na Fig. 2). Esse modelo pode ainda ser estendido pela inclusão de diretivas de compilação que permitam a realização da síntese utilizando compiladores como o da Synopsys, conforme já mencionado na Seção 2.1.

O modelo de gerador de tráfego foi implementado no nível de sistema no SystemC. O modelo implementa o método genérico para geração de tráfego proposto por Tedesco (2005). Com base nesse método, o gerador de tráfego pode enviar pacotes com tamanho fixo ou variável e com intervalo entre pacotes também fixo ou variável. Esses valores são calculados em tempo de compilação ou em tempo de simulação e podem ser configurados de modo a submeter a rede a uma determinada carga de trabalho. A taxa de geração pode ser constante ou regida por funções de probabilidade como a Normal, a Exponencial ou a Pareto On/Off. O perfil da comunicação também pode ser caracterizado especificando-se a distribuição espacial de comunicação entre geradores de tráfego. Atualmente, os seguintes padrões são suportados:

a) Padrões com um único destinatário por remetente (emissor) – detalhes disponíveis em [Duato, Yalamanchili e Ni, 1997]:
   - *bit-reversal*;
   - *perfect-shuffle*;
   - *butterfly*;
   - matriz transposta; e
   - complemento
b) Padrões com múltiplos destinatários por remetente:
   - uniforme: todos os nodos do sistema têm a mesma probabilidade de serem destinatários. Cada remetente envia o mesmo número de pacotes a todos os demais nodos;
   - não-uniforme: a probabilidade de um nodo enviar pacotes a outro nodo diminui com a distância entre eles. Nodos vizinhos trocam uma quantidade de pacotes máxima, enquanto que nodos não vizinhos trocam um número de pacotes que é inversamente proporcional ao número de roteadores entre eles.
   - local: todos os nodos vizinhos tem a mesma probabilidade de serem destinatários, recebendo o mesmo número de pacotes. Porém, nodos não adjacentes possuem probabilidade nula (nenhum pacote lhes é remetido).

O TG se baseia em arquivo de configuração (*traffic.cfg*) que descreve o perfil de tráfego para geração de pacotes por todas as instâncias de TGs (nodos) em um dado sistema. Esse arquivo é gerado automaticamente por uma ferramenta de apoio (denominada *gtr*) que recebe os parâmetros do tráfego e gera o arquivo correspondente aos parâmetros fornecidos. A Fig. 4 mostra o conteúdo do arquivo de configuração de

tráfego para um sistema com 16 TGs conectados a uma rede 4x4, utilizando o padrão de tráfego espacial do tipo complemento (cada TG envia pacotes ao TG com endereço complementar ao seu). Por exemplo, no arquivo ilustrado, o *tg_0_0* (00,00 em binário) deve enviar 1000 pacotes com 49 *flits* (mais o cabeçalho) ao *tg_3_3* (11,11 em binário), a uma taxa de injeção constante com uma largura de banda requerida de 20%. Para pacotes com 50 *flits*, devem ser inseridos 800 ciclos de intervalo entre o fim de um pacote e o início do pacote seguinte.

```
tg_0_0
1
0 3 3 49 1000 0.200000 800 0 0 0 0.00 0.00

tg_0_1
1
0 3 2 49 1000 0.200000 800 0 0 0 0.00 0.00

                      (...)

tg_3_3
1
0 0 0 49 1000 0.200000 800 0 0 0 0.00 0.00


// Parameters (from left to right)
 0  type
 1  x_dest
 2  y_dest
 3  payload_length
 4  pck_2send
 5  required_bw
 6  idle_cycles
 7  iat
 8  burst_size
 9  last_payload_length
10  alfa_on (for Pareto-based generation)
11  alfa_off(for Pareto-based generation)
```

Figura 4. Exemplo de arquivo de configuração de tráfego
(o trecho central foi omitido por ser desnecessário ao entendimento)

O modelo do medidor de tráfego também foi implementado no nível de sistema no SystemC. Durante a execução de uma simulação, esse modelo gera um arquivo de saída em que são registrados os dados referentes a cada pacote recebido pelo medidor. Conforme ilustrado na Fig. 5 (a seguir), as informações incluem o número do pacote recebido, incluindo: as coordenadas XY dos TGs destinatário e fonte do pacote, os ciclos de relógio em que o pacote foi criado e injetado na rede (que podem ser diferentes no caso de contenção), os ciclos de relógio em que o cabeçalho (*header*) e o terminador (*trailer*) foram recebidos pelo destinatário, o tamanho do pacote e a largura de banda requerida. As informações referentes às coordenadas e ao momento de criação e injeção dos pacotes são incluídas no seu cabeçalho e em parte do seu corpo no momento do envio e lidas pelo medidor de tráfego no momento da sua recepção.

```
FILE: ext_0_0_out
    Packet Xd Yd Xs Ys    Packet    Packet    Header   Trailer Packet  Req.
    Number              Creation Injection  at cycle  at cycle Length Band.
#
         1  0  0  3  3         3         3       433       629     50  0.20
         2  0  0  3  3      1003      1003      1433      1629     50  0.20
         3  0  0  3  3      2003      2003      2433      2629     50  0.20
         4  0  0  3  3      3003      3003      3433      3629     50  0.20
         5  0  0  3  3      4003      4003      4433      4629     50  0.20
         6  0  0  3  3      5003      5003      5433      5629     50  0.20

                                    (...)
```

Figura 5. Segmento inicial do arquivo de saída gerado pelo medidor de tráfego conectado ao roteador de coordenadas 0,0.

Os arquivos gerados por cada medidor de tráfego são analisados por uma ferramenta, denominada *atr*, que lê todos os arquivos, determina a latência acumulada em função da latência de cada pacote e calcula a latência média e o tráfego aceito pela rede (número de *flits* por nodo por ciclo). Ele também permite obter outras informações, como o número de pacotes cuja latência situa-se em uma determinada faixa (ex. de 10% a 20% acima da latência mínima a vazio) ou que violaram requisitos de tempo de entrega, entre outras. A modelagem no nível de sistema oferece bastante flexibilidade para inclusão de diferentes tipos de análise.

A Fig. 6 ilustra um dos formatos de relatório resumido apresentado pela ferramenta *atr*. Nela, observa-se que foram gastos 80004 ciclos de relógio para entregar 16.000 pacotes com uma latência média de 30 ciclos por pacote (neste caso, pacotes com tamanho igual a 50, incluindo o cabeçalho). O tráfego aceito pela rede foi de 0,10 flits/nodo/ciclo.

```
Simulation time = 80004 cycles
16000 packets were delivered
Average latency = 30.00 cycles/packet
Accepted traffic= 0.10 flits/node/cycle
```

Figura 6. Relatório de saída da ferramenta *atr*

4.3. Fluxo para Avaliação de Desempenho

Para automatizar a geração de modelos de redes e de sistemas de qualquer tamanho, bem como o processo de geração de padrões de tráfego e de análise de resultados da simulação, foi implementado um conjunto de ferramentas de apoio que são utilizadas conforme o fluxo mostrado na Fig. 7. A ferramenta *gnoc* gera redes compatíveis com a arquitetura SoCIN (*socin.h*), enquanto que a ferramenta *gsys* gera a descrição de um sistema (*main.cpp*) com arquitetura baseada na plataforma ilustrada na Fig. 3. Essa descrição é compilada pelo *gcc* que gera um simulador do sistema (*system.x*). Na sua execução, esse simulador lê o arquivo de configuração de tráfego gerado pela ferramenta *gtr* e produz arquivos de *log* (*ext_x_y.out*) gerados pelos medidores de tráfego. Esses arquivos são então analisados pela ferramenta *atr* que determina os índices de desempenho da rede.

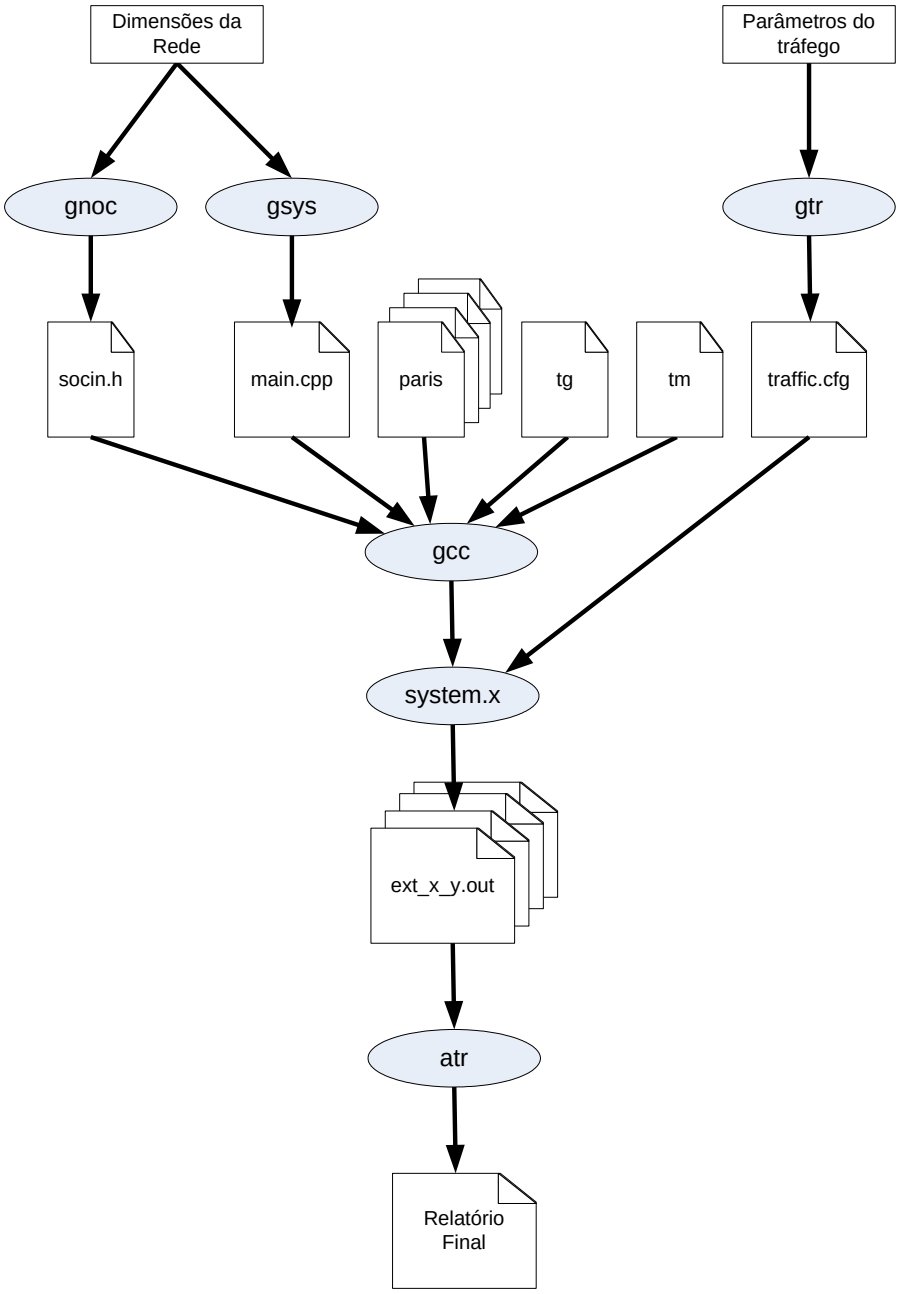

Figura 7. Fluxo para avaliação de desempenho

4.4 Validação

Os modelos e as ferramentas desenvolvidas foram validados exaustivamente ao longo do desenvolvimento e após a sua integração. Para cada componente do modelo do roteador ParIS, foi construído um *testbench* voltado à verificação funcional do modelo. A validação da plataforma completa foi feita com a simulação de sistemas de diferentes tamanhos, desde redes 2x2 até redes 8x8, com 64 geradores de tráfego enviando pacotes a todos os demais geradores da rede. Esse tipo de simulação oferece uma taxa de cobertura para validação bastante alta, pois explora a grande maioria dos caminhos e circuitos da rede.

5 Experimentação

5.1 Configurações

Nos diferentes experimentos realizados algumas variáveis da simulação foram mantidas constantes, incluindo:

- Tamanho da rede: 4x4;
- Largura do *flit*: 32 bits
- Arbitragem: round-robin;
- Técnica de controle de fluxo: Baseada em créditos;
- Tamanho do pacote: 16 *flits* (mais o cabeçalho)
- Número de pacotes enviados de um remetente para um destinatário: 1000 pacotes (na distribuição não-uniforme, esse número diminui com a distância entre remetente e destinatário); e
- Tipo de injeção de tráfego: constante.

A partir dessa configuração base, as demais variáveis foram alteradas para estabelecer diferentes configurações de experimentos, incluindo:

- Algoritmos de roteamento: XY e WF;
- Armazenamento (*buffers*): somente na entrada e na entrada e na saída;
- Profundidade dos *buffers*: 2, 4 e 8 posições;
- Distribuições espaciais: complemento, uniforme, não-uniforme e local;
- Carga oferecida (largura de banda requerida): 10 a 55% da largura de banda máxima do canal (com incrementos de 5%);

As distribuições espaciais *bit-reversal*, *perfect-shuffle*, *butterfly* e matriz transposta não são consideradas nesses experimentos, pois, em seus padrões, nem todos os nodos da rede injetam pacotes, levando a uma largura de banda efetiva menor do que a especificada pela taxa de injeção, dificultando a comparação dos resultados. Por exemplo, quando é especificada uma taxa de injeção de 40% da largura de banda da rede, no padrão *butterfly*, onde apenas metade dos nodos injeta pacotes, a carga efetiva é de 20%, enquanto que nos padrões em que todos os nodos injetam pacotes, a carga efetiva é de 40%.

5.2 Aspectos gerais dos experimentos realizados

A partir dos espaços de configurações acima, foi realizado um conjunto de experimentos de modo a avaliar o impacto da variação dos parâmetros do roteador e/ou dos padrões de tráfego no desempenho da rede.

A execução dos múltiplos experimentos foi suportada por uma ferramenta (denominada *gsim*) que automatiza a execução de uma seqüência de simulações em que são variadas apenas a distribuição espacial e a carga oferecida. Por exemplo, com uma única interação com o usuário, é possível realizar simulações para todas as oito distribuições espaciais disponíveis, variando-se a carga de trabalho de 10% a 60% com

incremento de 2%. Sem o uso dessa ferramenta, seriam necessárias 240 interações com o usuário para a definição de todas essas configurações.

Na análise dos pacotes transferidos na rede, são desconsiderados os primeiros (0 a 10% do total) e os últimos (90% a 100% do total) pacotes recebidos por cada nodo, pois esses podem ser beneficiados com uma menor contenção na rede.

No contexto dos resultados apresentados a seguir, utiliza-se o termo "ponto de saturação" para definir o momento a partir da qual a rede não é mais capaz de aceitar qualquer tráfego adicional. Conforme Duato, Yalamanchili e Ni (1997), a vazão da rede pode ser medida pelo seu tráfego aceito, expresso pelo número de *flits* por nodo por ciclo. Acima do ponto de saturação, mesmo incrementando a taxa de injeção da rede, com o aumento da carga de trabalho oferecida, ela não consegue suportar a largura de banda desejada e a latência média sofre um aumento excessivo em relação ao seu valor com a rede trabalhando sob uma carga mais leve, ou seja, a rede está saturada. Isso caracteriza um ponto de operação a partir do qual a configuração utilizada é inadequada para o tráfego aplicado.

5.3 Resultados

O primeiro experimento compara os diferentes tipos de distribuição de tráfego espacial, com carga oferecida variando de 10 a 55% da largura de banda disponível (medida em *flits*/ciclo/nodo), a uma taxa de injeção constante. É utilizada uma configuração de rede com roteamento XY, *buffers* de 4 *flits* apenas na entrada dos roteadores e controle de fluxo baseado em créditos. Com exceção do padrão não-uniforme, cada comunicação envolve a transferência de 1000 pacotes com 16 *flits* (mais o cabeçalho). No padrão não-uniforme, o número de pacotes transmitidos diminui com o aumento da distância entre os nodos. Por exemplo, nodos não vizinhos, separados por 3 roteadores, trocam 500 pacotes, enquanto que os separados por 4 roteadores, trocam 250 pacotes, e assim por diante. Os nodos mais distantes na rede 4x4, (0,0) e (3,3), trocam apenas 32 pacotes.

A Fig 8 ilustra a latência média de cada distribuição e a sua relação com o aumento da carga oferecida. A distribuição com menor latência média é a local, pois cada nodo se comunica apenas com seus nodos vizinhos, com uma distância de 2 roteadores entre cada par de nodos comunicantes. Na distribuição uniforme, cada nodo envia pacotes a todos os demais nodos da rede, com uma distância média de 3,94 roteadores entre nodos comunicantes, para a rede considerada (4x4). Já na distribuição complemento, embora cada nodo envie pacotes para apenas um nodo, a distância média entre nodos comunicantes é de 5 roteadores. Por isso, a sua latência média é maior que a das demais. A distribuição não-uniforme tem distância e latência médias intermediárias entre as da distribuição local e as da distribuição uniforme, pois cada nodo se comunica com todos os demais da rede, mas com probabilidade maior para se comunicar com nodos mais próximos.

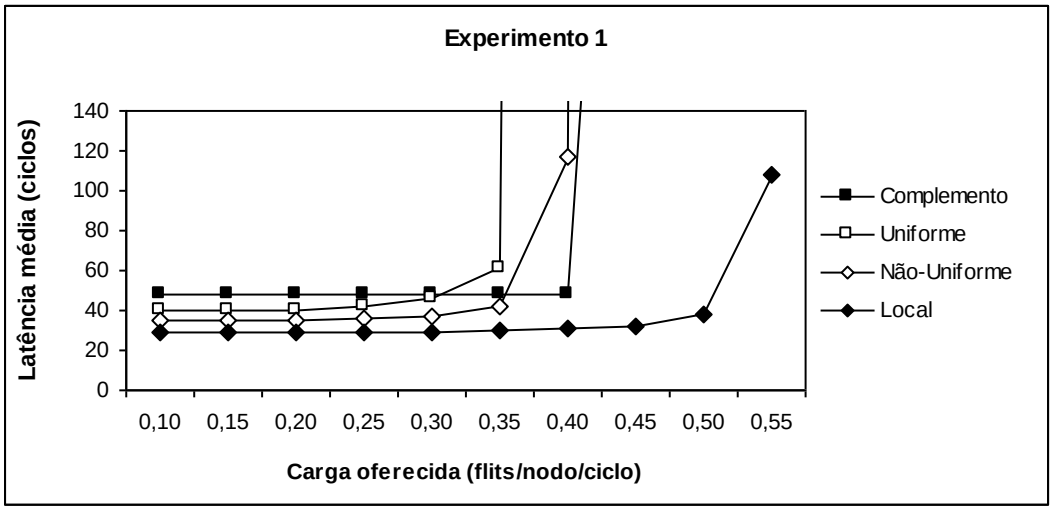

Figura 8. Comparação da latência média para diferentes padrões de distribuição espacial – configuração: roteamento XY, *buffers* de 4 *flits* nas entradas, controle de fluxo baseado em créditos

A Fig 9 ilustra as medições tomadas quanto ao tráfego aceito. Neste gráfico, observa-se que as distribuições com maior localidade são as que aceitam menos tráfego. Isso acontece porque nessas distribuições há uma probabilidade maior de ocorrer contenção no uso dos canais *Lout* dos roteadores pelos quais os pacotes saem da rede.

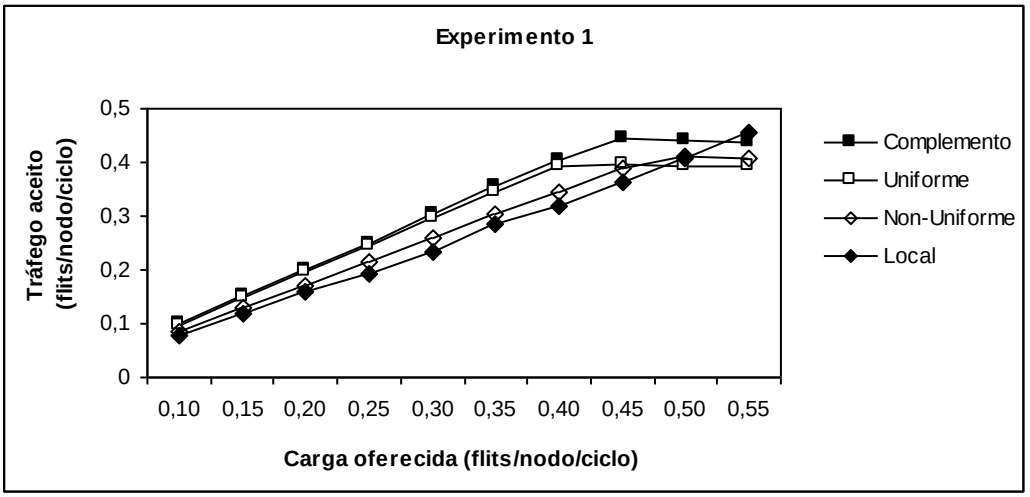

Figura 9. Comparação do tráfego aceito para diferentes padrões de distribuição espacial – configuração: roteamento XY, *buffers* de 4 *flits* nas entradas, controle de fluxo baseado em créditos

Analisando-se os dois gráficos acima, observa-se que as distribuições que possuem uma maior competição por enlaces entre roteadores (uniforme e não-uniforme) são as que saturam mais cedo, com uma menor carga oferecida. Nota-se, ainda, que a distribuição local é capaz de sustentar sua latência média para cargas maiores que as demais distribuições, começando a saturar com uma carga de 55%, quando a latência média ultrapassa 100 ciclos de relógio.

Considerando as aplicações típicas para sistemas embarcados, das quatro distribuições analisadas, as de maior localidade são que mais se aproximam do padrão

de comunicação de aplicações reais, onde os núcleos não se comunicam com a mesma probabilidade. Por isso, o mapeamento dos núcleos na rede deve ser feito de maneira a manter mais próximos aqueles que se comunicam com maior freqüência. No entanto, a distribuição local é muito restritiva, e não considera a comunicação entre nodos não adjacentes, o que de fato ocorre em aplicações reais. Por isso, a distribuição não-uniforme é usada nos experimentos a seguir que comparam o desempenho da rede utilizando diferentes configurações do roteador ParIS. Nesses experimentos, cada simulação envolve a transferência de 104.116 pacotes de 16 *flits* entre os 16 geradores de tráfego.

No segundo experimento, compara-se o desempenho da rede sob duas configurações de rotamento: XY e WF (West-First). Como pode ser observado nos gráficos a seguir, o uso do roteamento XY resulta em uma latência média levemente menor e em um maior tráfego aceito. Além disso, a rede consegue sustentar uma latência mais baixa para uma carga oferecida maior, saturando em 50% no XY contra 45% no WF. O pior desempenho do roteamento WF se deve ao fato de ele ser parcialmente adaptativo (o XY é determinístico) e permitir o uso de diferentes caminhos para pacotes enviados em direção ao leste. Com o aumento do tráfego, ele produz uma concentração de pacotes no meio da rede, aumentando a contenção.

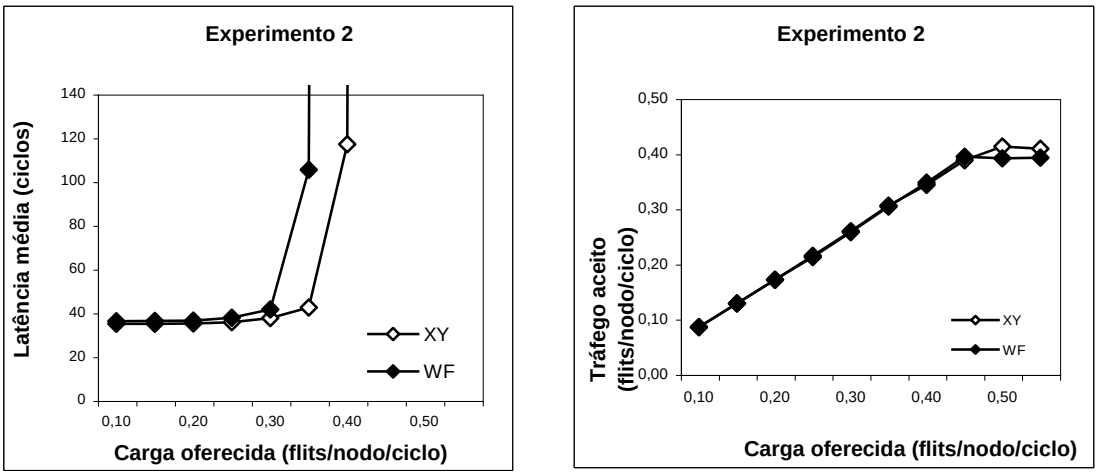

Figura 10. Avaliação do impacto do algoritmo de roteamento na latência média e no tráfego aceito pela rede – configuração: *buffers* de 4 *flits* nas entradas, controle de fluxo baseado em créditos, distribuição espacial não-uniforme

No terceiro experimento, avalia-se a influência da profundidade dos *buffers* de entrada no desempenho da rede. A Fig. 11 ilustra resultados para simulações com *buffers* de profundidade igual a 2, 4, 8 e 16 *flits*. Analisando-se os resultados, nota-se que *buffers* muito rasos limitam demasiadamente o tráfego aceito, pois, quando um pacote é bloqueado, ele mantém alocados todos os canais e *buffers* já reservados. Com *buffers* de dois *flits* nos roteadores e pacotes de 16 *flits* (mais o cabeçalho), em uma rede 4x4, um pacote pode reservar todos os roteadores do seu caminho enquanto estiver bloqueado, provocando um aumento da contenção na rede. Com *buffers* mais profundos, esse efeito é minimizado, reduzindo-se a latência e estendendo-se o ponto de saturação. Mas, pode-se perceber que *buffers* de profundidade igual a 8 e a 16 *flits* possuem latências médias similares para uma carga oferecida de até 25%.

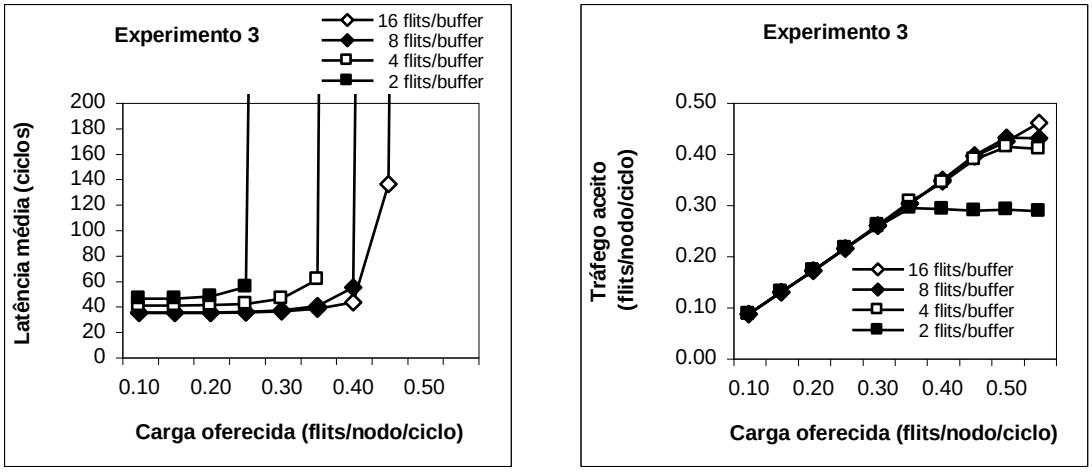

Figura 11. Avaliação do impacto da profundidade dos *buffers* de entrada na latência média e no tráfego aceito pela rede – configuração: roteamento XY, controle de fluxo baseado em créditos, distribuição espacial não-uniforme

No quarto experimento, avalia-se o efeito da inclusão de *buffers* (com capacidade de armazenar 4 *flits*) nos canais de saída dos roteadores. Dois efeitos podem ser notados a partir da análise dos resultados ilustrados na Fig. 12. Primeiro, há um aumento da latência média, o que se deve ao ciclo adicional para registro do cabeçalho nos *buffers* de saída dos roteadores (a latência mínima para o cabeçalho de um pacote atravessar um roteador aumenta de 3 para 4 ciclos, incluindo o roteamento, a arbitragem e o armazenamento nos *buffers* de entrada e de saída). O segundo efeito é o aumento do tráfego aceito. Isso ocorre porque os *buffers* de saída reduzem a contenção dentro dos roteadores. Por exemplo, considerando um pacote de 16 *flits* bloqueado na rede e que seu último *flit* esteja bloqueado em um roteador, sem o uso de *buffers* de saída, esse *flit* irá reservar uma conexão interna do roteador até que ele possa ser encaminhado para outro roteador. Com o *buffer* no canal de saída, o *flit* é absorvido por esse *buffer*, liberando a conexão interna e a saída do *buffer* do canal de entrada pelo qual ele chegou no roteador.

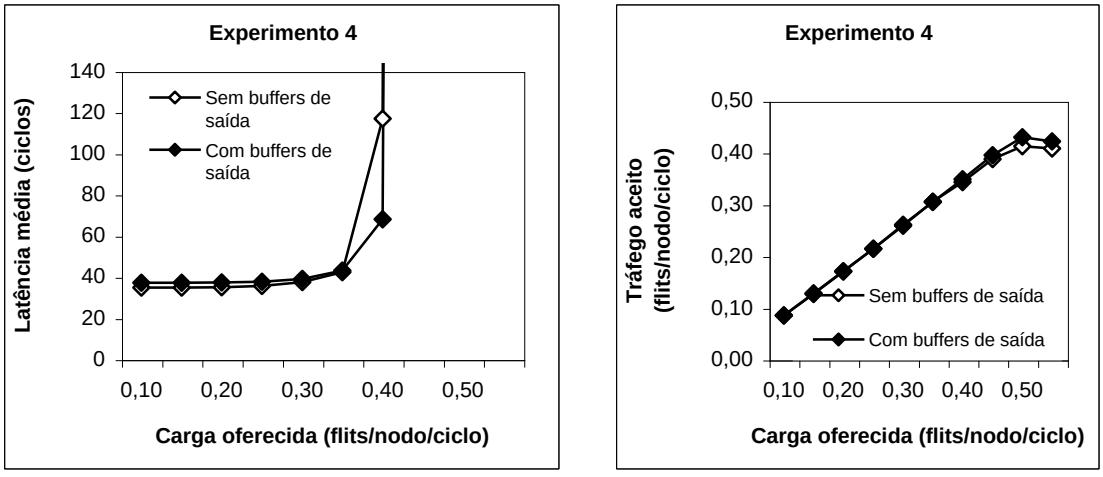

Figura 12. Avaliação do impacto do uso de *buffers* nos canais de saída na latência média e no tráfego aceito pela rede – configuração: *buffers* de 4 *flits*, roteamento XY, controle de fluxo baseado em créditos, distribuição espacial não-uniforme

A melhoria do desempenho proporcionada pela inclusão dos *buffers* de saída se deve, também, ao aumento da capacidade de armazenamento dos roteadores, não apenas pela posição dos *buffers*. Por exemplo, o quinto experimento mostra que se for dobrada a capacidade dos *buffers* na configuração que não adota memorização de saída (ou seja, 8 *flits*), essa passa a ter melhor desempenho do que com uso de *buffers* de 4 *flits* nas entradas e nas saídas dos roteadores.

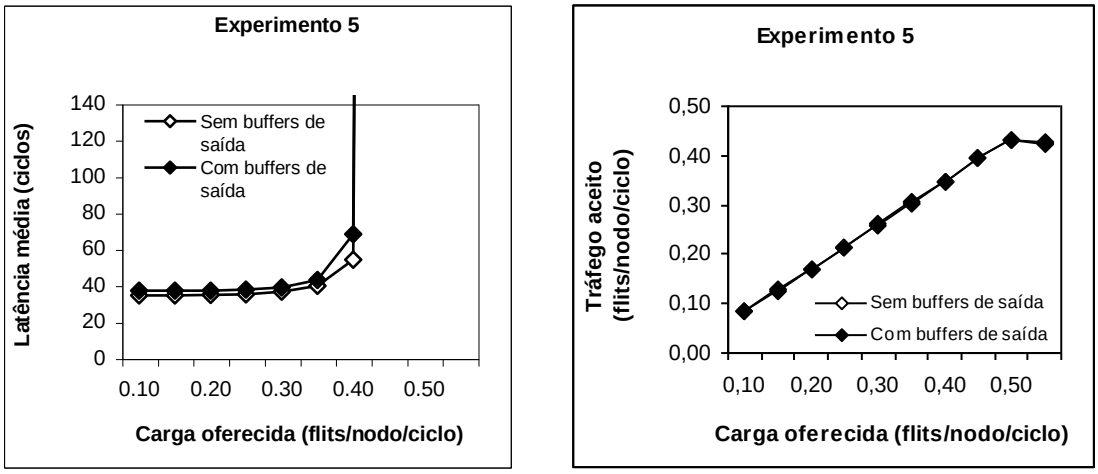

Figura 13. Avaliação do impacto do uso de *buffers* nos canais de saída na latência média e no tráfego aceito pela rede – configuração: *buffers* de entrada de 8 *flits* e *buffers* de entrada e de saída de 4 *flits*, roteamento XY, controle de fluxo baseado em créditos, distribuição espacial não-uniforme

Além dos experimentos mostrados acima, foram realizados outros que permitiram analisar o impacto da configuração do roteador no desempenho da rede sob diferentes padrões de tráfego. Esses experimentos também confirmaram a capacidade do ambiente desenvolvido (modelos SystemC + ferramentas de apoio ao projeto de experimentos) em auxiliar na análise das melhores alternativas de configuração de rede. No entanto, os resultados obtidos permitem extrair conclusões apenas para padrões de tráfego genéricos. O objetivo do projeto ao qual este trabalho está associado é construir um ambiente integrado para modelagem em SystemC de aplicações reais baseadas em arquiteturas de comunicação do tipo Rede-em-Chip, e posterior análise de tráfego para auxiliar na escolha tanto do mapeamento da aplicação na rede, quanto da configuração de roteador a ser adotada.

7 Conclusões

Neste artigo, foi apresentado a modelagem de uma Rede-em-Chip em SystemC visando a simulação da rede para avaliar o impacto de diferentes configurações de roteador no desempenho da rede. Foi adotadata uma abordagem de modelagem híbrida, em que os roteadores foram descritos no nível RT e os demais componentes de *benchmarking* no nível de abstração de sistema. Também foi descrito um conjunto de ferramentas de apoio desenvolvidas para automatizar os processos de geração de sistema e de geração e análise de tráfego.

Os resultados obtidos com os experimentos realizados permitiram mostrar que, para o padrão de tráfego considerado, o roteamento XY oferece melhor desempenho que

o roteamento WF. Também se pôde verificar que o uso de *buffers* muito rasos limita o desempenho das redes, mas que *buffers* muito profundos (16 *flits*), não trazem benefícios significativos, sendo que o uso *buffers* de 8 *flits* produziu os resultados mais satisfatórios. Além disso, pôde-se identificar que essa configuração de *buffer* também supera, em desempenho, a configuração com *buffers* de entrada e de saída com 4 *flits*.

Como já mencionado, este trabalho será complementado com a construção de um ambiente integrado para modelagem e avaliação de desempenho de aplicações reais, integrada a uma ferramenta já desenvolvida que realiza o mapeamento das tarefas da aplicação nos terminais de Redes-em-Chip. Para esse ambiente integrado, será construída uma interface visual e serão modeladas outras alternativas de configuração do roteador. Também será realizada a construção de modelos do roteador em SystemC em níveis mais abstratos, com potencial de oferecer um melhor tempo computacional nas primeiras etapas de exploração do espaço de projeto.

O grupo de pesquisadores associados a este trabalho também está investigando questões relativas ao provimento de QoS (Quality-of-Service) para NoCs para aplicações multimídia e de telecomunicações, uma vez que o roteador ParIS é do tipo melhor esforço (*best effort*) e não oferece qualquer garantia de QoS. O objetivo é desenvolver uma arquitetura de NoC de baixo custo que consiga possa prover níveis de serviço diferenciados e garantir a latência e/ou a vazão necessários para atender às restrições de projeto de aplicações dedicadas.

Agradecimentos

Este trabalho recebeu apoio do programa de bolsas Art. 170 do Governo do Estado de Santa Catarina (2006) e conta com fomento do Edital Universal 2006 do CNPq (2007-2008).

Referências

Andriahantenaina, A. et al. (2003), "SPIN: a Scalable, Packet Switched On-Chip Micro-Network", In: Design, Automation and Test on Europe - DATE, 2003, Munich. Proceedings... Los Alamitos: IEEE Computer Society. p. 70 73.

Bolotin, E. et al. (2004), "QnoC: QoS Architecture and Design Process for Network-on-Chip", Journal of Systems Architecture, v.5, n.2-3, p.105-128.

Chan, J., Parameswaran, S. (2003), "NoCGen - a Template Based Reuse Strategy for Networks-on-Chip", In: International Conference on VLSI DESIGN, 17, 2004, India. Proceedings... Los Alamitos: IEEE Computer Society. p. 70 73.

Duato, J., Yalamanchili, S., Ni, L. (1997), Interconnection Networks. Los Alamitos, IEEE CS Press.

Forte Design Systems (2007). "Cynthesizer Closes the ESL-to-Silicon Gap", disponivel em: <http://www.forteds.com/products/cynthesizer.asp>. Acesso em: 11 maio 2007.

Guerrier, P., Greiner, A. (2000), "A Generic Architecture for On-Chip Packet-Switched Interconnections", In: Design, Automation and Test on Europe - DATE, 2000, Paris. Proceedings... Los Alamitos: IEEE Computer Society Press. p. 250-256.

Gupta, R. K., Zorian, Y. (1997), "Introducing core-based system design", IEEE Design & Test of Computers, [S.l.], v. 14, n. 4, p. 15-25.

Hemani et al. (1999), “Lowering Power Consumption in Clock by Using Globally Asynchronous Locally Synchronous Design Style”, Design Automation Conference, ACM Press. p. 873-878.

Jantsch, A., Tenhunen, H. (2003), Networks on Chip. Boston: Kluwer Academic Publishers. 303p.

Kreutz, M. E. et al. (2005), “Design Space Exploration Comparing Homogeneous and Heterogeneous Network-on-Chip Architectures” In 18th Symposium on Integrated Circuits and Systems Design. Proceedings... New York: ACM.

Martin, G. (2003), “SystemC: from Language to Applications, from Tools to Methodologies”, In: Symposium on Integrated Circuits and Systems, 16, 2003, São Paulo. Proceedings... Los Alamitos: IEEE Computer Society. p. 3.

Moraes, F. et al. (2003), “A Low Area Overhead Packet-Switched Network-on-Chip: Architecture and Prototyping”, In: IFIP WG 10.5 Conference on Very Large Scale Integration of System-On-Chip, 2003, Darmstadt. Proceedings… Darmstadt: Technische Universität Darmstadt. p. 318-323.

OSCI (2002), SystemC version 2.0 user´s guide. 212p.

OSCI (2005), Draft Standard SystemC Language Reference Manual. 438 p.

Swan, S. (2001), An introduction to system level modeling in SystemC 2.0. [S.l.]: OSCI, May. 10p.

Synopsys (2003), CoCentric SystemC Compiler RTL: user and modeling guide. VU-2003.06. Synopsys.

Tedesco, L. P. (2005), Uma Proposta para Geração de Tráfego e Avaliação de Desempenho para NoCs, Dissertação (Mestrado em Ciência da Computação), PUCRS, Programa Pós-Graduação em Ciência da Computação.

Zeferino, C. A., Santo, F. G. M. E., Susin, A. A. (2004), “Paris: A Parameterizable Interconnect Switch for Networks-on-Chip”, In: 17th Symposium on Integrated Circuits and Systems, 2004, Porto de Galinhas. Proceedings. New York : ACM Press. p. 204-209.