NoCGen – UMA FERRAMENTA PARA GERAÇÃO DE REDES INTRA-CHIP BASEADA NA INFRA-ESTRUTURA HERMES

Fernando G. Moraes, Luciano C. Ost, Aline V. Mello, José C. Palma, Ney L. Calazans

Pontifícia Universidade Católica do Rio Grande do Sul (FACIN-PUCRS)
Av. Ipiranga, 6681 - Prédio 30 / BLOCO 4 - 90619-900 - Porto Alegre – RS – BRASIL
{moraes, ost, alinev, jpalma, calazans}@inf.pucrs.br

SUMMARY

The technology evolution allows increasing the number of IP cores at the same integrated circuit. This increasing drives the research of new intra-chip interconnection infrastructure, which must allow the communication requirements for future SoCs: reusability, scalability, and parallelism. A network on chip draws on concepts inherited from distributed systems and computer networks subject areas to interconnect IP cores in a structured and scalable way. The reusability of intra-chip interconnection infrastructure and IP cores are seen as a needed feature to enable future SoC designs. However, increasing the reusability design, the external interfaces of the IP cores and the intra-chip interconnection infrastructure must be standards. In this context, the present work focus the implementation of a tool which allows the generation of parameterizable NoCs based on infrastructure called HERMES. Besides the NoCs generation, the implemented tool called NoCGen, has the following characteristics: (i) testbench generation; (ii) traffic generation; (iii) traffic analyze and (iv) generation of network interface based on OCP protocol.

NoCGen – UMA FERRAMENTA PARA GERAÇÃO DE REDES INTRA-CHIP BASEADA NA INFRA-ESTRUTURA HERMES

Fernando G. Moraes, Luciano C. Ost, Aline V. Mello, José C. Palma, Ney L. Calazans

Pontifícia Universidade Católica do Rio Grande do Sul (FACIN-PUCRS)
Av. Ipiranga, 6681 - Prédio 30 / BLOCO 4 - 90619-900 - Porto Alegre – RS – BRASIL
{moraes, ost, alinev, jpalma, calazans}@inf.pucrs.br

ABSTRACT

The increasing complexity of integrated circuits drives the research of new intra-chip interconnection architectures. A network on chip draws on concepts inherited from distributed systems and computer networks subject areas to interconnect IP cores in a structured and scalable way. The present works describes a tool witch allows the generation of parameterizable NoCs based on infrastructure called HERMES. Besides the NoCs generation, the implemented tool called NoCGen, has the following characteristics: (i) testbench generation; (ii) traffic generation; and (iii) traffic analyze.

1. INTRODUÇÃO

O avanço tecnológico na construção de sistemas digitais complexos é tal que permite implementar em um único circuito integrado mais de 50 milhões de transistores (para microprocessadores). O ritmo desses avanços da tecnologia de fabricação tem se mantido exponencial nas últimas décadas, como atesta a Lei de Moore [SCH97]. A sigla SoC, do inglês *System on a Chip* [BER01], designa um sistema computacional completo implementado em um único circuito integrado. SoCs normalmente contêm um ou mais processadores de propósito geral, lógica digital (programável ou não), circuitos analógicos, além de bancos de memória dinâmica e estática. SoCs fornecem como vantagem maior desempenho, menor consumo de potência, menor volume e peso comparado com o projeto baseado em múltiplos circuitos integrados em uma placa de circuito impresso.

Projetar um processador já é uma tarefa complexa. Então, qual a técnica que se deve utilizar para projetar um SoC, contendo múltiplos processadores, módulos de hardware dedicado e software embarcado? Esta é uma questão ainda sem resposta plenamente satisfatória. Considera-se que parte da resposta está no *reuso* de módulos complexos de hardware, denominados núcleos de propriedade intelectual (IP cores), *núcleos IP* ou apenas *núcleos*. Um *núcleo* é um módulo digital ou analógico, podendo ser descrito em diferentes níveis de abstração [PAL02]. Núcleos são pré-projetados, pré-verificados e prototipados em hardware pelo menos uma vez. Segundo a Associação das Indústrias de Semicondutores [SIA99], estima-se que em 2012 90% da área de circuitos integrados VLSI seja preenchida por núcleos. O principal objetivo de seu uso é a redução do *time-to-market* de produtos. Apesar das vantagens inerentes à utilização de núcleos, identificam-se quatro grandes problemas que devem ser resolvidos para que se possa construir facilmente um SoC [BER01]: (i) como integrar núcleos entre si; (*ii*) quais linguagens para descrição de sistemas usar; (*iii*) como proteger a propriedade intelectual do autor e do usuário do núcleo; (*iv*) como testar projetos baseados em núcleos.

A interconexão por barramento é simples, sob o ponto de vista de implementação, apresentando entretanto diversas desvantagens [BEN02]: (*i*) apenas uma troca de dados pode ser realizada por vez; (*ii*) existe necessidade de desenvolver mecanismos inteligentes de arbitragem do meio físico para evitar desperdício de largura de banda; (*iii*) a escalabilidade é limitada; (*iv*) o uso de linhas globais em um circuito integrado com tecnologia submicrônica impõe sérias restrições. Estas desvantagens podem ser parcialmente contornadas através do uso de, por exemplo, hierarquia de barramentos, onde o problema continua existindo, sendo apenas minimizado.

Uma arquitetura de interconnexão que pode solucionar os problemas relacionados ao uso de barramento, simples ou hierárquicos, são as redes intra-chip [BEN02][BER01], um conceito denominado *network on chip* – NoC. NoCs herdam das redes de computadores e de sistemas distribuídos as características das camadas de protocolos e o conceito de ligação de nodos à rede. Nas NoCs, os núcleos do sistema são interligados por meio de uma rede composta por chaves e canais ponto–a-ponto. A comunicação entre os núcleos ocorre pela troca de mensagens transferidas por meio de chaves e canais intermediários até atingir o seu destino [BEN02].

Neste cenário, o presente trabalho tem como objetivo apresentar a ferramenta NoCGen. A ferramenta NoCGen possibilita a geração automática de NoCs parametrizáveis baseada na infra-estrutura HERMES [MOR03].

Este artigo está organizado da seguinte forma. A Seção 2 descreve sucintamente a rede HERMES. A seção 3 aborda a interconexão de núcleos a rede HERMES. A seção 4 apresenta a principal contribuição deste artigo, ou seja, uma ferramenta para geração de redes intra-chip, baseadas na infra-estrutura HERMES. A Seção 5 apresenta conclusões e direções para trabalhos futuros.

2. REDE HERMES

A rede HERMES [MOR03] é uma rede direta, com topologia mesh (*mesh*), que implementa três níveis hierárquicos do modelo de referência OSI: físico, enlace e rede. O nível físico da rede HERMES implementa a interface de comunicação entre as chaves, como apresentado na Figura 1. O nível de enlace adota o protocolo *handshake* para o envio e recebimento de dados de forma confiável. Para compor o nível de rede é asdotado o modo de chaveamento por pacote [HWA93][MOH98]. O pacote na rede HERMES é formado por 1 *flit* com endereçoo destino, 1 *flit* com o tamanho do payload e de 1 a 255 *flits* de payload.

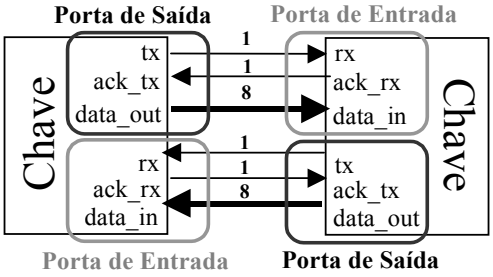

Figura 1 - Interface entre chaves.

A rede HERMES utiliza roteamento distribuído, determinístico e com caminho mínimo entre os nodos origem e destino. A política adotada para repasse de dados nas chaves intermediárias de uma comunicação é a *wormhole*. O modo de chaveamento *wormhole* possibilita a multiplexação de canais lógicos o que aumenta o desempenho do chaveamento [RIJ01]. No entanto, a rede HERMES utiliza até o presente momento um único canal lógico para cada canal físico, objetivando reduzir a complexidade da chave.

2.1 Chave HERMES

A chave HERMES possui uma lógica de controle de chaveamento e 5 portas bidirecionais: East, West, North, South E Local. Cada porta possui uma fila de tamanho parametrizável para o armazenamento temporário de *flits*. A porta Local estabelece a comunicação entre a chave e seu nodo. As demais portas ligam as chaves às suas chaves vizinhas. A lógica de controle de chaveamento engloba arbitragem e lógica de chaveamento.

A arbitragem determina qual o pacote deve ser chaveado primeiro, quando mais de um chega à chave em um mesmo instante de tempo. A arbitragem dinâmica rotativa foi utilizada para permitir o chaveamento do pacote da porta de entrada com maior prioridade. A prioridade de cada porta depende da última que obteve permissão de chaveamento.

A lógica de chaveamento adota um algoritmo XY, que faz a comparação do endereço da chave atual com a chave destino do pacote. Por este motivo, as chaves da rede possuem endereços únicos formados pelas coordenadas XY da rede, onde X é a posição horizontal e Y a posição vertical. A Figura 2 ilustra os módulos principais que compõem a chave (árbitro, lógica de chaveamento e porta de entrada/saída, representada por duas filas), e o protocolo adotado para inicial e concluir uma conexão será apresentado a seguir.

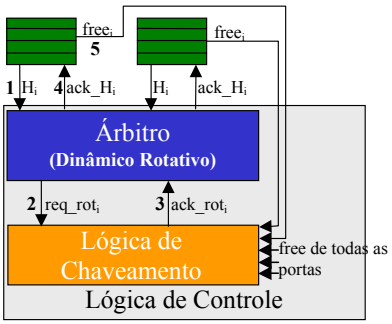

Figura 2 – Protocolo de estabelecimento e conclusão de conexão.

Cada uma das filas da chave (E, W, N, S e L), ao receber um novo pacote requisita chaveamento ao árbitro (índice 1). O árbitro seleciona a requisição de maior prioridade, quando existem requisições simultâneas, e encaminha o pedido de chaveamento à lógica de chaveamento (índice 2). A lógica de chaveamento verifica se é possível atender à solicitação. Sendo possível, a conexão é estabelecida e o árbitro é informado (índice 3). Por sua vez, o árbitro informa a fila (índice 4) que começa a enviar os flits armazenados. Quando todos os flits do pacote foram enviados, a conexão é concluída pela sinalização, por parte da fila, através do sinal free (índice 5).

3. INTERCONEXÕES DE NÚCLEOS A REDE HERMES

Um aspecto importante na abordagem de redes intra-chip consiste em como integrar núcleos à rede, garantindo a comunicação entre estes através do meio de comunicação. A menos que o núcleo atenda ao protocolo de comunicação da rede, torna-se necessário criar um invólucro que permita integrá-lo à mesma. Um invólucro deve possibilitar a integração física (interface - largura de sinais, sinais de entrada e saída) e os serviços de comunicação (segmentação e remontagem dos pacotes) entre o núcleo e a rede. No contexto de redes intra-chip denomina-se esse invólucro de interface de rede (IR).

Segundo Kumar [KUM03], é possível dividir o projeto interno da IR em duas partes: (i) a parte específica à rede (independente do núcleo), responsável pela temporização, bufferização e aspectos de sincronização durante a transmissão/recepção de dados; (ii) parte específica ao núcleo, responsável pela montagem/desmontagem dos pacotes.

Dentro desse contexto, visando aumentar a reusabilidade da HERMES em projetos distintos, padronizou-se as interfaces de rede com o protocolo OCP. Denomina-se HERMES-OCP a rede intra-chip HERMES com interface OCP. Objetiva-se com essa abordagem simplificar ao máximo as transações (núcleo-chave), tornando a HERMES o mais transparente possível para os núcleos que a empregam como meio de comunicação. Dependendo do núcleo, a porta local pode possuir uma interface de rede OCP do tipo mestre, escravo ou mestre-escravo. Uma interface do tipo mestre é aquela que tem a capacidade de começar uma transação através de uma requisição para um determinado escravo. Já uma interface do tipo escravo deve apenas responder a requisições enviando uma resposta ao núcleo solicitante.

Deve-se ressaltar que núcleos que desejem comunicar-se via a HERMES-OCP devem atender ao protocolo OCP adotado pela rede. Não basta que os núcleos possuam interfaces OCP compatíveis. Deve-se definir sobre o protocolo OCP o conjunto de sinais utilizados, e qual o protocolo de comunicação adotado pela rede. Em outras palavras, o nível físico da HERMES adota OCP, o nível de enlace define um formato de pacote, composto por uma seqüência de transações OCP e o nível de rede é uma comunicação ponto-a-ponto, que pode ser diferente para cada par de núcleos. Neste contexto foram desenvolvidos para a rede HERMES três tipos de IRs OCP: (*i*) IR mestre (IR-M); (*ii*) IR escravo (IR-E); e (*iii*) IR mestre-escravo (IR-ME).

Até o presente momento, as interfaces de rede propostas aqui suportam apenas transações de escrita e leitura. Núcleos que queiram escrever (enviar pacotes) ou ler dados (receber pacotes) para/de outros núcleos através da HERMES-OCP devem atender ao protocolo de comunicação da rede, tanto a interface de rede OCP e o protocolo de comunicação da rede HERMES são descritos em [OST03].

4. FERRAMENTA NoCGen

A rede HERMES é uma infra-estrutura para a geração de redes intra-chip. Diz-se infra-estrutura porque não existe uma única rede intra-chip. Existe um conjunto de módulos, como árbitro, filas, portas de entrada/saída, todos parametrizáveis em função das restrições do projeto, como largura do canal, profundidades do buffer, topologia de rede, entre outras.

A parametrização das chaves e a geração manual da rede a partir destas é um processo passível de erros, devido à grande quantidade de fios que ligam as chaves entre si e ao núcleo local, como ilustra a Figura 3. A automatização desse processo foi a principal motivação para o desenvolvimento da ferramenta NoCGen.

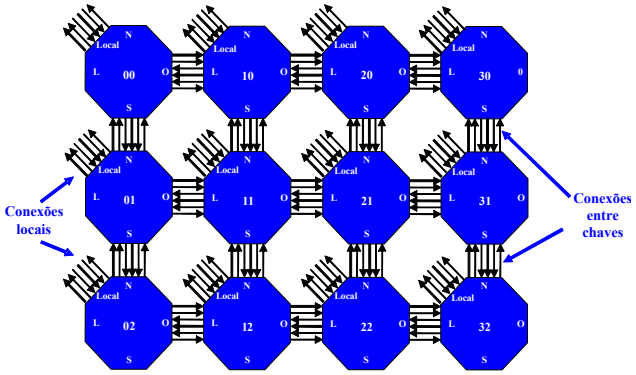

Figura 3 - Interconexões entre chaves em uma rede intra-chip 3 x 4.

Além de automatizar o processo de interconexão das chaves, a ferramenta permite a escolha do algoritmo de roteamento e a configuração de parâmetros como: (*i*) largura dos canais de dados; (*ii*) profundidade das filas de entrada das chaves e (*iii*) dimensão da rede. Pretende-se a partir da configuração desses parâmetros pré-definidos gerar redes intra-chip que respeitem os requisitos de uma aplicação especifica. Outra possibilidade é utilizar a ferramenta para gerar diversas configurações, possibilitando a análise de qual configuração tem melhor desempenho para uma dada aplicação.

Uma outra funcionalidade da ferramenta é a remoção de filas de entrada de canais não utilizados na chave, o que reduz a área da HERMES em hardware. Além disso, a ferramenta gera tráfego e testbenchs, permitindo com isso validar a rede gerada. A descrição da parametrização do código da HERMES e as funcionalidades suportadas pela ferramenta são descritas a seguir.

4.1 Adaptação dos arquivos VHDL

Para gerar uma NoC, a ferramenta NoCGen utiliza arquivos VHDL que descrevem a rede HERMES. A partir destes arquivos, a ferramenta gera a estrutura da NoC desejada em função dos parâmetros definidos pelo usuário. Para que os arquivos VHDL da rede HERMES pudessem ser utilizados pela ferramenta, foram inseridos *flags* que definem os pontos do código a serem alterados pela mesma. Dentre os arquivos alterados destaca-se o *Hermes_Package.vhd* (biblioteca específica da rede HERMES). Nesta biblioteca são inseridos os valores configurados pelo usuário, que são utilizados por todos os outros arquivos VHDL da NoC. A Figura 4 ilustra um trecho do código da biblioteca *Hermes_Package*. Neste trecho pode-se observar os *flags* *$flit_size$*, *n_rot*, *$buff_depth$* e *$pointer_size$*, que correspondem, respectivamente, à largura dos canais de dados, ao número de roteadores da rede, à profundidade das filas e ao tamanho do ponteiro necessário para percorrer todas as posições da fila.

```
constant TAM_FLIT :
         integer range 1 to 32 := $flit_size$;
constant NROT : integer := $n_rot$;
constant BUFFER_TAM : integer := $buff_depth$;
constant TAM_POINTER : integer:= $poiter_size$;
```

Figura 4 - Trecho do código VHDL da biblioteca Hermes_Package.

Para parametrizar a chave foram inseridos no arquivo *Chave.vhd* os flags: (*i*) *$chave$*, que deve ser substituído pelo nome que define o tipo de chave; (*ii*) *$filas$*, onde serão inseridas (mapeadas) as filas utilizadas de acordo com o tipo de chave; e (*iii*) *$zeros$*, onde são "aterrados" (recebem o valor '0') os sinais que deveriam ser conectados às filas removidas. Os sinais não utilizados devem ser aterrados para que não interfiram no funcionamento do controle de chaveamento.

No arquivo *Fila.vhd* não foram inseridos *flags*. Porém, a largura das palavras , a profundidade da fila e os ponteiros que as percorrem são baseados nas constantes geradas na biblioteca *Hermes_Package.vhd*.

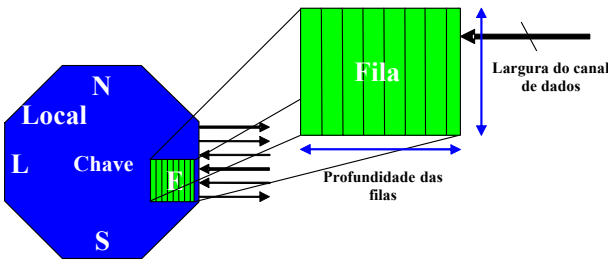

Figura 5 - Pontos parametrizáveis na chave e na filas.

A Figura 5 mostra os pontos onde ocorre a parametrização nas chaves, ou seja, as filas mapeadas, a profundidade (capacidade) das filas e a largura dos canais de dados (que também afeta a largura das filas).

4.2 Topologia e redução de área

A partir da configuração da Chave é possível escolher a topologia mais adequada para aplicação desejada. Até o presente momento a ferramenta NoCGen permite a geração de quatro topologias diferentes utilizadas em redes intra-chip: (*i*) mesh; (*ii*) torus; (*iii*) torus dobrado; e (*iv*) anel. As topologias (*ii*) (*iii*) utilizam todas as portas de conexão das chaves, o que não acontece nas topologias (*i*) e (*iv*) que apresentam canais abertos (desconectados). Porém, mesmo quando os canais estão abertos as filas de entrada permanecem presentes nas chaves, ocasionando desperdício de área em hardware. A Figura 6 ilustra um exemplo de rede intra-chip organizada sob a topologia mesh. Nesta topologia, as filas não utilizadas devem ser removidas da descrição VHDL, a fim de reduzir a área da rede intra-chip. É o caso da chave presente na última coluna da última linha, que possui dois canais sem conexão (sul e oeste).

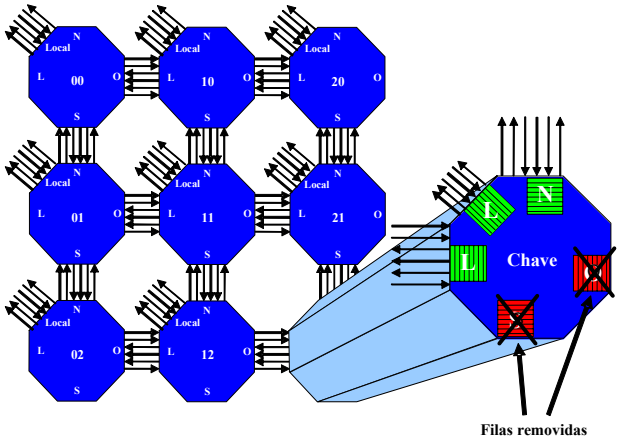

Figura 6 - Descrição dos nove tipos de chaves em uma topologia mesh e um exemplo de chave onde remoção de filas deve ocorrer.

O tipo de chave é dado pela posição da chave na rede. Por exemplo, a chave mostrada na Figura 6 é classificada como ChaveBR (*Bottom Right*), ou seja, pertence à última linha e à coluna bem à direita da rede intra-chip. Desta forma, são nove os tipos possíveis de chaves: *Top Left* (TL), *Top Center* (TC), *Top Right* (TR), *Center Left* (CL), *Center* (CC), *Center Right* (CR), *Bottom Left* (BL), *Bottom Center* (BC) e *Bottom Right* (BR).

A remoção de filas não utilizadas veio a colaborar para redução de área das chaves que não possuem conexões em todas as portas. A Tabela 1 mostra o consumo de área de uma rede HERMES 3 x 3 (com 9 chaves), com largura do canal de dados de 8 bits, filas com profundidade de 8 palavras, prototipada na plataforma com o FPGA XC2V1000. Foram feitas duas implementações dessa NoC: (*i*) todas as cinco filas em todas chaves, inclusive as localizadas na periferia (não utilizadas), com um consumo de 57% do dispositivo; (*ii*) NoC 3 x 3 gerada pela ferramenta NoCGen e, portanto, sem as filas da periferia (filas removidas), obtendo-se um consumo de 42% do dispositivo. Com base nestes dados, fica claro que a redução de área obtida com a remoção das filas sem utilização é importante, dado que a fila é o componente que mais consome área na chave.

Tabela 1 - Consumo de área de duas implementação de uma HERMES 3 x 3: uma com todas a filas, inclusive as não utilizadas e outra apenas com a filas utilizadas.

Implementação	Custo (área)
Normal, com aterramento das filas não utilizadas.	2932 dos 5120 Slices do dispositivo XC2V1000 (57%)
NoC gerada pela ferramenta, com remoção de filas não utilizadas.	2191 dos 5120 Slices do dispositivo XC2V1000 (42%)

3.4. Interface e funcionalidades da NoCGen

A interface da ferramenta NoCGen é dividida em quatro blocos básicos: (*i*) área de visualização; (*ii*) barra de parametrização da rede; (*iii*) barra de menus; e (*iv*) log de processos. Esta interface é apresentada na Figura 7.

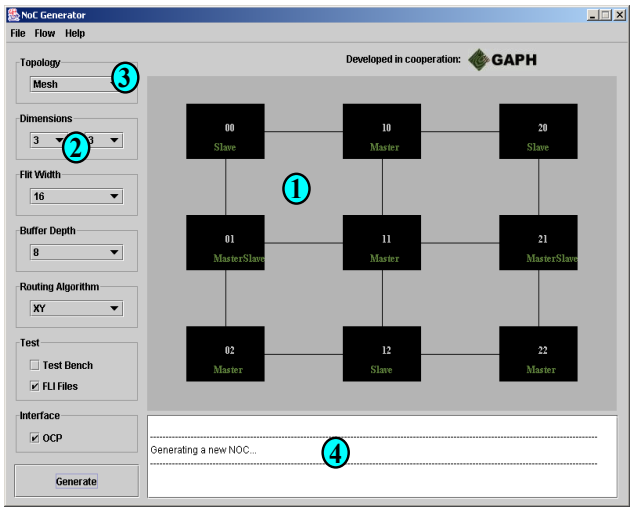

Figura 7 - Interface da ferramenta NoCGen.

A *área de visualização* (índice 1 na Figura 7) representa a rede de acordo com os valores configurados pelo usuário nos campos de parametrização da rede. Com isso, é possível visualizar a dimensão da rede, os endereços e as conexões das chaves. Além disso, é possível escolher o tipo de interface OCP de cada chave, clicando sobre a área da mesma (caso a opção Interface OCP esteja selecionada). São três os tipos se interfaces OCP como mencionado na seção : mestre OCP (IM-OCP), escravo OCP (IE-OCP) e mestre-escravo OCP (IME-OCP). Na Figura 7 a área de visualização representada uma rede mesh de dimensão 3x3, sendo essa composta por três chaves com IR-E (00, 20, 12), quatro chaves com IR-M (10, 11, 02, 22) e duas chaves com IR-ME (01, 21).

O usuário parametriza a rede a partir dos campos contidos na *barra de parametrização* (índice 2 na Figura 7). A partir desses campos é possível configurar: (*i*) a topologia; (*ii*) a dimensão da rede; (*iii*) a largura do canal de comunicação e (*iv*) profundidade das filas de entrada das chaves. No caso da topologia mesh o usuário pode escolher o algoritmo de roteamento para sua rede. Com isso, torna-se possível verificar o

desempenho do mesmo frente a sua aplicação. Dentre os algoritmos suportados pela ferramenta citam-se: (*i*) XY puro; (*ii*) *West-First Minimal*; (*iii*) *West-First Non-Minimal*; e (*iv*) *Negative-First Non-Minimal*. As particularidades inerentes a esses algoritmos de roteamento são descritos em [GLA94]. A implementação de algoritmos livre de *deadlock* para as topologias torus e torus dobrado está em andamento. Estes algoritmos devem ser inseridos na ferramenta ao término do processo de validação e implementação. Até o presente momento a ferramenta NoCGen só permite a geração de redes intra-chip baseadas na estrutura HERMES.

Ainda na *barra de configuração* pode-se selecionar o tipo de *testbench* a ser gerado. Até o momento a ferramenta NoCGen suporta a geração de dois tipos de *testbench*: (*i*) descrito em VHDL que a partir da inclusão de arquivos externos gera estímulos (tráfego) a NoC gerada (pode ser utilizado tanto no *ModelSim* quanto no simulador *ActiveVDHL*); e (*ii*) implementado com *FLI*. *FLI* é uma interface de comunicação que permite descrever o comportamento de um módulo em VHDL utilizando código objeto gerado à partir de um programa escrito em C. O código objeto gerado pode ser executado durante uma simulação para validação de um módulo em *hardware* no simulador *ModelSim*® [MOD02]. Para facilitar o processo de simulação é gerado um *script* que é utilizado para compilação e simulação da NoC no simulador *ModelSim*. Este *script* informa ao simulador *ModelSim* a ordem de compilação dos arquivos e o tempo de simulação do sistema (NoC e *testbench* gerados). O *testbench*, nos dois formatos, consiste de leituras e escritas em arquivos, onde cada arquivo de entrada possui os pacotes que um núcleo deve enviar para a rede. O arquivo de saída apresenta os pacotes que chegaram ao núcleo ao qual o arquivo esta associado.

A *barra de menu* (índice 3 na Figura 7) possui três menus: *File*, *Flow* e *Help*. O menu *File* possui a opção de criar um novo projeto, onde se determina o nome e o diretório onde será criado o projeto da nova NoC. O menu *Flow* automatiza o processo de teste da NoC gerada. Este menu é composto por três itens: (*i*) Traffic Generation, (*ii*) Simulation e (*iii*) Traffic Analyser.

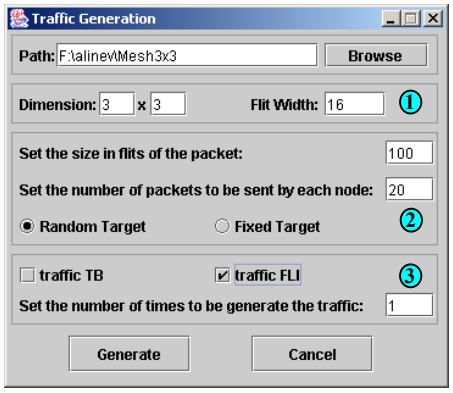

Figura 8 - Interface do Traffic Generation.

Traffic Generation gera os arquivos de entrada dos testes em dois formatos (TB e *FLI*) e possui a interface apresentada na Figura 8. Nessa interface é permitido escolher a dimensão da rede, a largura do *flit*, o número de pacotes a serem enviados por cada núcleo, o número de *flits* por pacote, o destino dos pacotes e número de vezes que será gerado o tráfego. O índice 1 da Figura 8 informa que o tráfego será gerado para 9 núcleos (dimensão 3x3); o índice 2 da Figura 8 informa que serão gerados 20 pacotes por núcleo, cada pacote será composto por 100 *flits* e o destino destes pacotes serão escolhidos aleatoriamente, o índice 3 da Figura 8 informa que esta configuração será gerada apenas 1 vez e somente no formato de arquivo *FLI*.

Após ser gerado o tráfego é possível testar a NoC através do item *Simulation*. Este item executa os scripts e ao final gera os arquivos de saídas.

Traffic Analyser é responsável por analisar os arquivos de saídas e apresentar as seguintes estatísticas: (*i*) número de pacotes recebidos por cada nodo da rede; (*ii*) número total de pacotes recebidos; (*iii*) tempo mínimo, médio e máximo de transmissão de um pacote; (*iv*) desvio padrão do tempo de transmissão de um pacote e (*v*) tempo total de transmissão de todos os pacotes. Estes resultados são apresentados em um arquivo chamado *analise.txt* criado no diretório onde a analise foi realizada. A analise de tráfego possibilita a comparação de diferentes configurações de NoC para um determinado fluxo de dados, como ilustrado na Tabela 2.

Tabela 2 - Resultados da analise de tráfego para duas configurações de uma mesh 5x5: (*i*) profundidade de buffer de 8 posições e (*ii*) profundidade de buffer de 16 posições. Nesta analise utilizou-se um tráfego onde cada núcleo conectado a rede transmite 20 pacot tes com 20 flits cada, totalizando 500 pacotes.

	Buffer 8 posições				Buffer 16 posições			
	Traffic			Média	Traffic			Média
	1	2	3		1	2	3	
Média	201	195	190	193	255	264	254	258
Devio Padrão	121	109	109	113	149	154	154	152
Mínmo	51	51	51	51	51	51	51	51
Máximo	768	717	787	757	992	976	1211	1063
Tempo Total	3.035	2.816	3.114	2.988	2.800	2.728	2.873	2.800

O *log de processos* (índice 4 na Figura 7) serve para informar ao usuário das operações que estão sendo executadas pela ferramenta.

5. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho descreveu as principais funcionalidades da ferramenta NoCGen. Esta ferramenta automatizou a geração de redes intra-chip baseadas na infra-estrutura HERMES. A geração de redes intra-chip através da ferramenta traz os seguintes benefícios: (*i*) redução do tempo de projeto; (*ii*) unifica as versões dos códigos dos componentes, simplificando o controle de versões; (*iii*) facilidade para geração de diferentes redes intra-chip para a mesma aplicação, permitindo explorar o espaço de soluções para o problema; (*iv*) geração automática de tráfego e de *testbenchs*, produzindo arquivos de teste a rede; (*vi*) verificação do tráfego na rede, ou seja, se todos os pacotes enviados são recebidos no seus respectivos destinos; e (*vii*) geração do template VHDL de nível superior à rede, para conexão de núcleos à mesma.

Outras funcionalidades como algoritmos livre de *deadlock* para as topologias *Torus* e *Torus-dobrado*, decodificação de vídeo ou imagem para a geração de tráfego mais real às aplicações de SoCs e a geração de NoCs parametrizáveis em

nível TL System C são futuras contribuições para o presente trabalho.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[BEN02] Benini, L. De Micheli,G. "*Networks on chips: a new SoC paradigm*". Computer, Volume: 35(1), Jan. 2002, pp. 70-78.

[BER01] Bergamaschi, R. A.; Bhattacharya, S.; Wagner, R.; Fellenz, C.; Muhlada, M.; White, F.; Daveau, J. M.; Lee, W. R. "*Automating the design of SoCs using cores"*. IEEE Design & Test of Computers, v. 18(**5**), September-October 2001, pp. 32 - 45.

[DAL01] Dally, W.J.; Towles, B. *"Route packets, not wires: on-chip interconnection networks"*. In: Design Automation Conference, 2001, pp. 684-689.

[GLA94] Glass, C.; Ni, L. The Turn Model for Adaptive Routing. Journal of the Association for Computing Machinery, v. 41(5), Sep. 1994, pp. 874-902.

[HWA93] Hwang, K. "*Advanced Computer Architecture : Parallelism, Scalability, Programmability*". New York : McGraw-Hill, 1993, pp. 213-256.

[KUM03] Kumar, S. **On Packet Switched Network for Chip Communication.** In: Axel Jantsch and Hannu Tenhunen, editors, *Networks on Chip*, chapter 5, Kluwer Academic Publishers, 2003, pp. 85-106.

[MOH98] Mohapatra, P. "*Wormhole Routing Techniques for Directly Connected Multicomputer Systems*". ACM Computing Surveys, Volume: 30(3), Sep. 1998.

[MOR03] Moraes, F. G.; Mello, A.; Möller, L.; Ost, L.; Calazans, N. "A Low Area Overhead Packet-switched Network on Chip: Architecture and Prototyping". In: Very Large Scale Integration (VLSI)-SOC, December 2003.

[MOD02] ModelSim EE/PLUS User's Manual , chapter 11 – VHDL Foreign Language Interface and Verilog.

[OST03] Ost, L.C. "*Redes Intra-Chip Parametrizáveis com Interface Padrão para Síntese em Hardware "*. Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2003, 102 p.

[RIJ01] Rijpkema, E.; et al. "Router Architecture for Networks on Silicon". In: Workshop PROGRESS'2001, Netherlands, 2001.

[SCH97] Schaller, R. R. "*Moore´s Law: Past, Present and Future*". IEEE Spectrum, Volume: 34(6), pp. 52-59, Jun., 1997.

[WIN01] Wingard, D. *"MicroNetwork-based integration for SoCs"*. In: Design Automation Conference, 2001, pp. 673-677.