Networks-on-Chip: Conceitos, Arquiteturas e Tendências

Henrique Cota de Freitas[1], Philippe Olivier Alexandre Navaux[2]

[1]Instituto de Informática
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Av. Dom José Gaspar, 500 – 30.535-901 – Belo Horizonte – MG – Brasil

[2]Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

cota@pucminas.br, navaux@inf.ufrgs.br

Resumo. *Como alternativa para a demanda crescente por desempenho, arquiteturas de processadores many-core têm sido apontadas como soluções e representam a próxima geração de processadores. Neste contexto, interconexões em chip globais possuem problemas de escalabilidade. Para solucionar este problema a literatura científica tem apontado as Networks-on-Chip (NoCs) para suportar as comunicações em chip e o alto número de núcleos de processamento. Devido a importância do tema e da necessidade de mais pesquisas sobre NoCs, o objetivo deste minicurso é apresentar os principais conceitos e características arquiteturais, além de tendências relativas ao projeto e desenvolvimento de Networks-on-Chip.*

1. Introdução

Técnicas para aumentar o desempenho de paralelismo no nível de instrução, através de arquiteturas superescalares, atingiram limites difíceis para que desempenhos sejam superados (KEYES, 2008). A principal alternativa utilizada tem se baseado em elevadas freqüências de operação e, conseqüentemente, alto consumo de potência (KEYES, 2001) (HO, 2001). O surgimento dos processadores *multi-core* (OLUKOTUN, 1996) (OLUKOTUN, 2005) tem relação direta com problemas de desempenho encontrados pela exploração de altas freqüências em arquiteturas de processadores *single-core* superescalares. No entanto, para novas gerações de processadores *many-core* (elevada quantidade de núcleos de processamento), novos problemas surgem (ASANOVIC, 2006). Um dos grandes problemas está relacionado às redes-em-*chip*, que precisam ter escalabilidade, flexibilidade, alta vazão e desempenho.

As redes-em-*chip*, conhecidas por NoCs (*Networks-on-Chip*) (BENINI, 2002) (BENINI, 2005) (BJERREGAARD, 2006), têm sido apontadas como alternativas para problemas de escalabilidade encontrados em interconexões tradicionais como barramentos e chaves *crossbars*. Para uma quantidade pequena de núcleos, e.g., processadores *dual-core* e *quad-core*, soluções globais baseadas em barramentos e chaves *crossbars* são adequadas e atingem alto desempenho. No entanto, à medida que a quantidade de núcleos aumenta, e.g., 64 núcleos, uma solução global resulta em alta

resistência do fio e problemas de roteamento do mesmo. Isto reduz a escalabilidade e, portanto, a possibilidade de aumentar a quantidade de núcleos. Com foco neste problema, as NoCs surgem com a característica de redução no comprimento do fio pela inserção de pequenos roteadores, entre si interconectados, responsáveis por prover acesso dos núcleos aos periféricos e pela comunicação coletiva entre núcleos. A arquitetura das NoCs normalmente é baseada em topologias regulares, e.g., *mesh* e *torus*, um roteador por núcleo, links de comunicação, e adaptadores de rede. Alternativas são baseadas em arquiteturas programáveis, reconfiguráveis, clusterizadas, topologias irregulares, inserção de novos componentes de rede, que buscam o aumento do desempenho no processamento de aplicações.

Além da arquitetura do processador e da própria NoC, outro fator que influencia no desempenho é a carga de trabalho (FREITAS, 2009c) (FREITAS, 2009d). Por este motivo, o projeto de uma NoC tem por obrigação o estudo das cargas de trabalho que serão executadas. A carga de trabalho depende do contexto, podendo ser de propósito geral ou específica, mas é fator fundamental para definição da arquitetura de suporte. Em processadores *many-core* de propósito geral, é esperada a execução de diversos programas paralelos que possam explorar a arquitetura nativamente paralela (vários núcleos). Estes programas paralelos geram uma quantidade elevada de comunicação coletiva (DUATO, 2002) através de passagem de mensagem que, conseqüentemente, devem ser suportadas pelas NoCs para envio e recebimento de pacotes.

Considerando a importância do tema NoC para as próximas gerações de processadores, o objetivo deste minicurso é apresentar os principais conceitos relativos às *Networks-on-Chips*. O texto foca em diferentes abordagens arquiteturais e possíveis tendências e/ou influências relacionadas ao projeto de processadores *many-core*.

Partes deste minicurso foram elaboradas a partir de uma síntese da fundamentação teórica da tese intitulada "Arquitetura de NoC Programável Baseada em Múltiplos Clusters de Cores para Suporte a Padrões de Comunicação Coletiva" (FREITAS, 2009d) defendida em junho de 2009 no PPGC da UFRGS pelo primeiro autor e orientada pelo segundo autor deste minicurso. Com base nesta fundamentação teórica, este minicurso apresenta uma discussão sobre impacto e tendências relativas às NoCs na nova geração de processadores *many-core*. Portanto, este minicurso está dividido em seções, conforme as seguintes descrições:

- Seção 2 – Processadores Multi/Many-Core: Esta seção apresenta uma evolução das arquiteturas de processadores *multi-core* até *many-core*, onde há uma grande demanda por projetos de *Networks-on-Chip*.
- Seção 3 – Arquiteturas de Networks-on-Chip: são apresentados os principais conceitos e arquiteturas relacionadas às NoCs. Características tais como, topologias, protocolos e componentes principais são discutidos.
- Seção 4 – Influência das Cargas de Trabalho no Projeto de NoCs: No projeto de NoCs, a influência das cargas de trabalho é um fator que deve ser observado, e neste caso, é necessário entender o contexto de aplicação/uso das NoCs, para avaliar o impacto das cargas de trabalho durante o projeto.

- Seção 5 – Propostas de NoCs Adaptáveis às Cargas de Trabalho: Já que a carga de trabalho é importante, alguns dos trabalhos de pesquisas focam em NoCs adaptáveis que possam aumentar o desempenho de processamento das aplicações. Nesta seção, algumas NoCs adaptáveis são analisadas.
- Seção 6 – Método de Projeto e Avaliação de NoCs: Nesta seção é dada ênfase em como usar modelos analíticos, de simulação e de experimentação/medição, e como aplicá-los durante o projeto e avaliação das NoCs.
- Seção 7- Influência das NoCs no Projeto de um Processador Many-Core: Esta seção apresenta uma visão geral da influência das NoCs no projeto de um processador *many-core*, possíveis problemas, novas demandas de pesquisa e desempenho, como por exemplo, memórias *cache* compartilhadas e NUCA (*Non-Uniform Cache Architecture*).
- Seção 8 – Pesquisas e Conferências Relacionadas às NoCs: O objetivo desta seção é apresentar temas para pesquisas, e algumas conferências, simpósios ou workshops importantes relacionados ao tema NoC.

2. Processadores Multi/Many-Core

A rápida evolução da área de arquitetura de computadores tem mostrado que é necessária cada vez mais uma maior interação entre áreas relacionadas (ASANOVIC, 2006). Projetos de processadores *multi-core*, principalmente *many-core* (BORKAR, 2007), são exemplos da necessidade de um estudo mais amplo capaz de abranger áreas até certo ponto distintas, tais como: redes de comunicação de dados, caracterização de cargas de trabalho paralelas e computação reconfigurável. Nesta seção é dada ênfase em conceitos básicos de arquiteturas de processadores *multi/many-core* que demandam *networks-on-chip*.

Apesar de processadores com múltiplos núcleos existirem desde a década passada, em sistemas dedicados, e.g., processadores de rede (INTEL, 2001) (COMER, 2003), o surgimento de processadores de propósito geral tem alguns anos. Os fatores que mais motivaram o surgimento destes processadores são os limites da Lei de Moore (KEYES, 2008) e o alto consumo de potência. A base do desempenho dos processadores se valia pelo paralelismo no nível de instrução, com *pipeline* superescalar, no máximo duas *threads* simultâneas (SMT: *Simultaneous Multihreading*) e a constante elevação da freqüência de operação. Porém, o consumo de potência já atingia limites impraticáveis, o paralelismo no nível de instrução já havia encontrado limites de desempenho e múltiplas *threads* simultâneas em um mesmo *pipeline* muitas vezes degradava o desempenho. Sendo assim, era necessária uma nova abordagem para a constante evolução no aumento do desempenho, atendendo limites de consumo de potência. Os processadores com múltiplos núcleos, já usados em outras áreas, se tornaram alternativas para computadores de propósito geral.

Existem processadores *multi-core* de propósito geral que suportam múltiplas *threads* (UNGERER, 2002) (UNGERER, 2003) (FREITAS, 2006) em cada núcleo, através de *Interleaved Multithreading* - IMT (KONGETIRA, 2005) ou SMT (KALLA, 2004). O motivo para escolha de uma técnica ou outra se deve muito a característica das cargas de trabalho. A Figura 1 ilustra um experimento realizado em 1996 (OLUKOTUN, 1996) que mostra as vantagens de usar uma arquitetura *multi-core* em relação a uma arquitetura superescalar. O experimento mostra que a área ocupada pelas duas soluções é a mesma, e as principais características de cada arquitetura são as seguintes:

- Ambas são baseadas em arquiteturas do processador MIPS R10000.
- A arquitetura (a) é um superescalar de seis vias de execução.
- A arquitetura (b) é *multi-core*, possui quatro núcleos, sendo que cada um possui *pipeline* superescalar de duas vias.

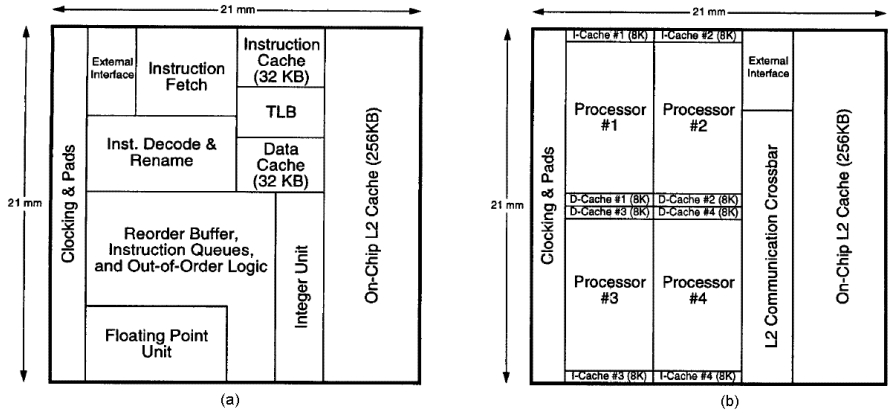

Figura 1: Comparação entre arquiteturas. (a) superescalar, (b) multi-core (OLUKOTUN, 1996)

Em resumo, os resultados mostram que uma arquitetura superescalar possui desempenho melhor se a carga de trabalho possuir alto paralelismo no nível de instrução. Por outro lado, a arquitetura *multi-core* possui melhor desempenho se a carga de trabalho tiver um alto paralelismo no nível de *thread*. Este experimento foi o início para o surgimento do processador Niagara (Ultrasparc T1) (KONGETIRA, 2005), que possui oito núcleos escalares com suporte IMT cada um. Este processador possui alta vazão de *threads* e baixa vazão de instruções. Uma arquitetura como esta não é adequada para um computador pessoal, mas adequada para um sistema de aplicações *web* e banco de dados, onde a carga de trabalho possui alto paralelismo de *threads*.

Pode ser visto na Figura 1b, que para quatro núcleos uma chave *crossbar* é suficiente para prover a comunicação interna. A mesma alternativa foi adotada para o processador Niagara, que possui oito núcleos. No entanto, para a nova geração de processadores *many-core*, uma única chave *crossbar* possui limites físicos que impedem o aumento da quantidade de núcleos interconectados a ela (CIDON, 2009). A Figura 2 mostra que no projeto Terascale da Intel (INTEL, 2006) (INTEL, 2007), está em desenvolvimento um processador *many-core* com oitenta núcleos e uma *Network-on-Chip*. Em realce, cada núcleo possui um roteador e, portanto, a comunicação interna é por troca de mensagens.

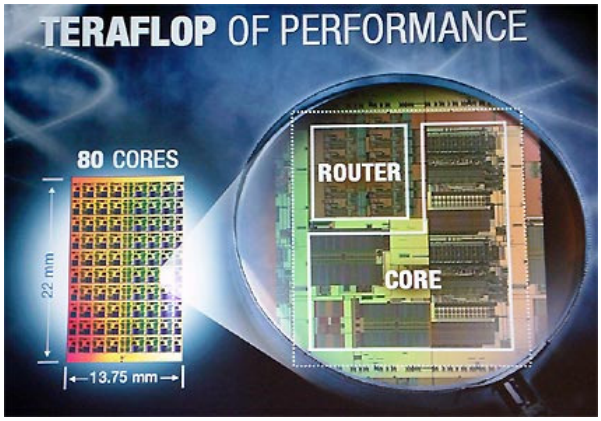

Figura 2: Arquiteturas Many-Core. Teraflops de desempenho (INTEL, 2006)

As principais vantagens dos processadores *many-core*, além do aumento do desempenho, podem ser ilustrados pela Figura 3, tais como:

- Ativação somente dos núcleos necessários.
- Redistribuição de carga de trabalho em função de altas temperaturas atingidas pelos núcleos.
- Núcleos reservas para substituição de núcleos com problemas.
- Expansão da funcionalidade com diversos dispositivos dentro do *chip*.

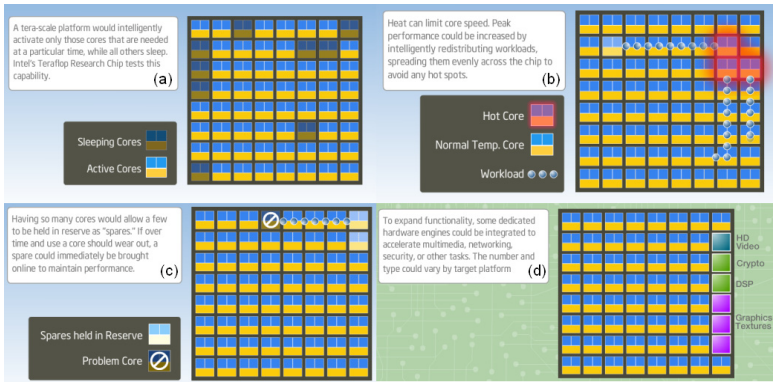

Figura 3: Justificativas do projeto Terascale. (a) utilização inteligente de núcleos, (b) redistribuição de carga de trabalho, (c) núcleos reservas, (d) integração de dispositivos dedicados (INTEL, 2007)

Um detalhe que pode ser percebido pela Figura 3 é a necessidade de uma rede de comunicação eficiente e que possibilite as vantagens apresentadas. Portanto, a próxima seção deste minicurso apresenta as principais características das *Networks-on-Chip*.

3. Arquiteturas de Networks-on-Chip

De acordo com os limites de escalabilidade impostos pelos fios, as tradicionais soluções de interconexão largamente utilizadas em arquiteturas *multi-core*, tais como barramento e chave *crossbar*, são impraticáveis para arquiteturas *many-core*. A solução que vem sendo estudada e proposta através de várias pesquisas é a *Network-on-Chip* (BENINI, 2002) (BENINI, 2005) (BJERREGAARD, 2006) (OGRAS, 2007b). A Figura 4 ilustra a evolução focada no problema relativo aos fios até uma arquitetura inicial de NoC, como forma de sumarizar o obstáculo no uso de fios globais para interconexão de grande número de núcleos.

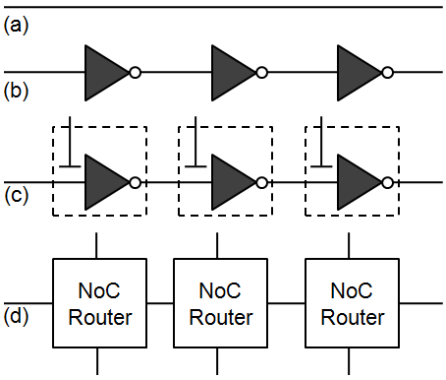

Figura 4: Evolução da interconexão global para NoC. (a) fio longo dominado pela resistência, (b) adição de repetidores ou buffers, (c) repetidores se tornam latches, e (d) latches evoluem para roteadores de NoC (CIDON, 2009)

Portanto, nesta seção é dada uma ênfase às NoCs apresentando as principais abordagens relativas às arquiteturas da rede ou roteador, topologias e protocolos, além de uma análise da viabilidade em comparação com as alternativas de interconexões tradicionais.

3.1. Arquiteturas de NoCs

Uma *Network-on-Chip* é composta por três elementos básicos: roteador, interface e *links* de comunicação. O roteador é o elemento principal responsável pela interconexão da rede, pela definição de rotas, pelo controle de fluxo, qualidade de serviço e, portanto, pela garantia de entrega do pacote de dados. Por se tratar de uma rede de comunicação composta por roteadores, o mecanismo de entrega de dados é através de passagem de mensagem ou pacotes de rede. Interligando os roteadores existem os *links* de comunicação. Estes *links* são os fios responsáveis pela existência do caminho a ser percorrido pelos pacotes. A forma como os roteadores estão interconectados pelos *links* dá origem à topologia da rede. A Subseção 3.2 apresenta detalhes e comparações entre topologias propostas para NoCs. O último elemento que compõe uma NoC é a interface de rede. Esta interface também é chamada de adaptador ou *wrapper*, sendo necessária para garantir a correta comunicação entre a rede (roteadores da NoC) e os núcleos ou periféricos que estão interconectados. Esta interface garante que haja uma correta comunicação entre protocolos diferentes (NoC, núcleo, memória, etc.).

A Figura 5 ilustra um exemplo de NoC baseada na topologia *mesh*. Neste caso, a NoC é uma *mesh* 3x3 que interconecta três núcleos de processamento através de três

interfaces de rede. Estas interfaces estão interconectadas, cada uma, a um roteador diferente da NoC e, portanto, cada roteador é específico para um núcleo de processamento. Os demais roteadores da NoC nesta figura não estão interconectados a elementos externos, mas poderiam ser memórias, ou qualquer *hardware* dedicado.

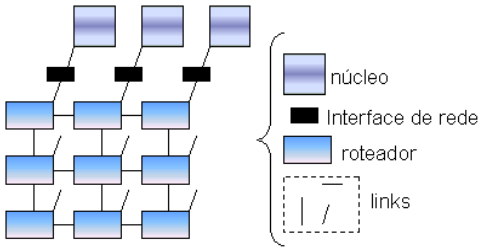

Figura 5: Arquitetura básica de uma NoC com topologia mesh (FREITAS, 2009a)

Um *chip* composto por uma NoC, núcleos de processamento, memórias e outros *hardwares* dedicados representam um sistema de computação. Portanto, para este caso existe um termo muito comum conhecido como Sistema-em-*Chip* ou *System-on-Chip* (SoC) (WOLF, 2004). Os projetos de arquiteturas de processadores *many-core* caminham para uma heterogeneidade de componentes em *chip* que os aproximam desta classificação.

A Figura 6 ilustra uma arquitetura típica de roteador para uma NoC *mesh* apresentada na Figura 5. Estes roteadores trabalham com roteamento XY e, portanto, possuem quatro saídas para norte, sul, leste e oeste. Além disso, é necessária uma saída para o núcleo de processamento através da interface de rede. Para a interconexão entre as saídas, o roteador se baseia em uma arquitetura simples de chave *crossbar*, neste exemplo, 5x5, normalmente sem nenhum adicional de complexidade, como, por exemplo, suporte a *broadcast*. Para definição das conexões que devem ser realizadas, é necessário um Árbitro. A função principal deste mecanismo de arbitragem está na solução de conflitos na utilização da chave *crossbar* e, por conseqüência, na liberação de pacotes dos *buffers* de entrada. Neste exemplo de roteador, os *buffers* estão presentes apenas nas entradas. É importante ressaltar, que a arquitetura deste roteador ilustra um exemplo de circuito dedicado e não programável.

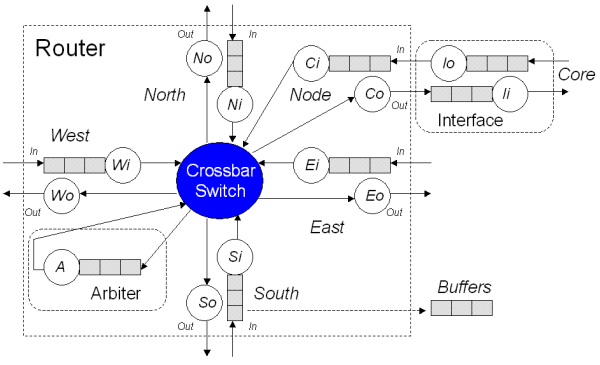

Figura 6: Arquitetura básica de um roteador de NoC para topologia mesh (FREITAS, 2009a)

O impacto da utilização de *buffers* pode estar relacionado ao tamanho, quanto maior a profundidade do *buffer*, maior o tamanho da NoC, ou pelos seguintes motivos relacionados à chave *crossbar* (AHMADI, 1989):

- *Buffers* de entrada: As técnicas de arbitragem são relativamente simples, possuem uma melhor relação de área e potência, além de proporcionar um melhor desempenho para a chave *crossbar*.
- *Buffers* de saída: Em função de N entradas conectadas a cada um dos *buffers* de saída, a chave *crossbar* precisa ser N vezes mais rápida. A adoção de *buffers* de saída não é a mais adequada para alto desempenho. No entanto, existem vantagens em se tratando da eliminação do bloqueio de pacotes que não receberam permissão de envio, porque o primeiro pacote da fila ainda não teve liberação de uma determinada saída. Este problema é conhecido como *head of the line blocking* e pode acontecer nas soluções com *buffers* de entrada.
- *Buffers* de *crosspoint*: Cada ponto de conexão da chave *crossbar* possui um *buffer*. É utilizada a técnica de roteamento chamada de *self-routing*. Neste caso, em cada *crosspoint* seria necessário, além do *buffer*, um decodificador para decisão de envio ou não do pacote. Esta solução aumenta o tamanho e a potência consumida da chave *crossbar*.

3.2. Topologias de NoCs

A definição de uma topologia de NoC, tal como de um roteador, está relacionada à carga de trabalho que será executada. A Tabela 1 ilustra as principais NoCs e suas respectivas topologias. Nesta subseção é feita uma análise das principais características destas topologias.

Tabela 1: Exemplos de topologias de NoCs (FREITAS, 2009d)

Network-on-Chip	Topologia
SPIN (ANDRIAHANTENAINA, 2003)	Árvore gorda.
ASoC (LIANG, 2000)	*Mesh* 2D.
Dally (DALLY, 2001)	*Torus* 2D.
Nostrum (MILLBERG, 2004)	*Mesh* 2D.
Sgroi (SGROI, 2001)	*Mesh* 2D.
Octagon (KARIM, 2002)	Anel Cordal.
Marescaux (MARESCAUX, 2003)	*Torus* 2D.
AEtheral (RIJPKEMA, 2003)	*Mesh* 2D.
Eclipse (FORSELL, 2002)	*Mesh* 2D hierárquica.
Proteo (SIGÜENZA-TORTOSA, 2002)	Anel bi-direcional.
Hermes (MORAES, 2004)	*Mesh* 2D.
SoCIN (ZEFERINO, 2003)	*Mesh/Torus* 2D.
SoCBUS (WIKLUND, 2003)	*Mesh* 2D.
QNoC (BOLOTIN, 2004)	*Mesh* 2D.
T-SoC (GRECU, 2004)	Árvore gorda.
Bouhraoua (BOUHRAOUA, 2006)	Árvore gorda.
BENoC (MANEVICH, 2009)	Barramento global integrado com *mesh* 2D.
BiNoC (LAN, 2009)	*Mesh* 2D com canais bidirecionais.
CHNoC (LENG, 2005)	*Clusters* com topologia irregular.
GigaNoC (NIEMANN, 2005)	*Clusters* com topologia regular *mesh* 2D.
IPNoSys (FERNANDES, 2008)	Programável. Roteadores ativos. *Mesh*, *torus* e *butterfly 2D*.

PNoC (HILTON, 2006)	Reconfigurável. Baseado em FPGA.
Bartic (BARTIC, 2005)	Reconfigurável. Baseado em FPGA.
CoNoChi (PIONTECK, 2008)	Reconfigurável. Baseado em FPGA.
Wireless NoC (WANG, 2007)	Sem fio irregular.
RECONNECT (JOSEPH, 2008)	*Honeycomb mesh*. Topologia fixa usada para suportar o conceito *Polymorphic ASIC*.
ReNoC (STENSGAARD, 2008)	Reconfigurável. Interface entre os roteadores e links para alteração/ativação de topologias. Usa o conceito *Polymorphic ASIC*.
Polymorphic NoC (MERCALDI-KIM, 2008)	Reconfigurável. Conjunto de Chaves *crossbar* interconectadas para reconfiguração de topologias.
MCNoC (FREITAS, 2008b)	Reconfigurável e Programável. Segundo nível de reconfiguração baseado em chaves *crossbar* para *clusters* de núcleos.

Através da Tabela 1 é possível citar três tipos de topologias: fixas, sem fio e reconfiguráveis. As topologias fixas são alternativas clássicas de adoção de uma determinada forma de interconexão que privilegie um comportamento específico de uma determinada carga de trabalho (BERTOZZI, 2005) (HO, 2006). Topologias sem fio são alternativas recentes para eliminar as limitações do fio no projeto de NoCs através de uma tecnologia chamada de *Radio-on-Chip* (CHANG, 2001). Por fim, as topologias reconfiguráveis utilizam plataformas programáveis para que sejam realizadas adaptações na forma de interligações em função de mudanças no padrão de comunicação das cargas de trabalho. Espera-se através da reconfiguração um aumento na flexibilidade de topologias da NoC focando no ganho de desempenho em relação a uma solução de topologia fixa.

As topologias mais encontradas na literatura são baseadas em *Mesh* e *Torus*. A principal característica está relacionada à capacidade de suportar aplicações cujos problemas podem ser particionados (e.g., operações com matrizes e processamento de imagens). As topologias *Honeycomb mesh* ou *mesh* hexagonal são consideradas da mesma família e possuem desempenho similar. A vantagem de uma topologia *honeycomb* em relação à *mesh* está na diminuição do custo em função de uma menor área ou quantidade de *links*, além de facilitar o mapeamento de aplicações.

A topologia Anel possui o menor custo entre as apresentadas, mas em contrapartida possui um diâmetro que cresce de forma linear em função do número de nós. Uma alternativa que mantêm um baixo custo é a topologia Anel Cordal que possui caminhos alternativos, aumentado o grau do nó e diminuindo o diâmetro. Esta alternativa é mais tolerante a falhas do que a versão original (Anel) e, portanto, mais confiável.

A topologia Árvore Binária é uma solução interessante para aplicações baseadas em algoritmos de divisão e conquista. O diâmetro cresce de forma linear em relação à altura, e a confiabilidade é relativamente baixa, já que a perda de um nó pode separar a topologia em duas partes. Outro problema está relacionado ao nó raiz que é um gargalo entre as subárvores da esquerda e direita. A solução para este problema está na Árvore Gorda. Esta solução é mais tolerante a falhas, pois possui um maior número de ligações entre os nós próximos da raiz. Outra solução é aumentar a largura de banda destas ligações, reduzindo as contenções de comunicação do efeito concentrador / gargalo da raiz da árvore.

Encontrar uma topologia, que atenda os melhores requisitos de desempenho, escalabilidade, confiabilidade e custo não é tão trivial (KREUTZ, 2005). No projeto Terascale da Intel (Figura 7) duas topologias estão em teste: anéis interconectados e *mesh*. Devido ao grande número de diferentes domínios de aplicação ou padrões de comunicação, algumas propostas apontam para o uso de conceitos de reconfiguração (COMPTON, 2002) (MARTINS, 2003) (TODMAN, 2005) (SO, 2006) para aumentar a adaptabilidade da rede (BJERREGAARD, 2006) (OGRAS, 2007a).

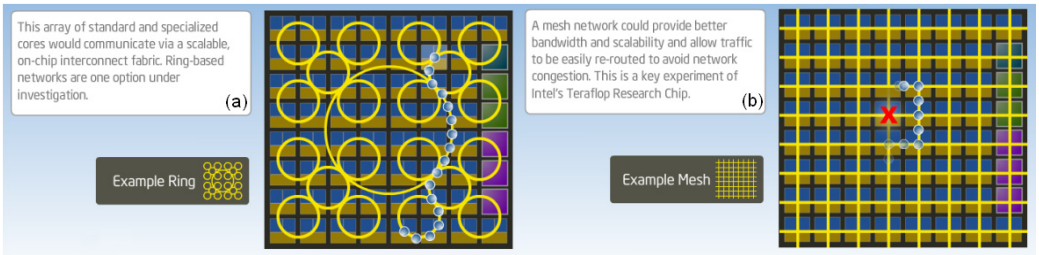

Figura 7: Topologias do projeto Terascale. (a) anéis interconectados, (b) mesh (INTEL, 2007)

3.3. Tipos de Protocolos

Políticas e estratégias de transporte de dados em uma NoC são de responsabilidade dos protocolos. Os protocolos descrevem as principais características de funcionamento da rede. Neste sentido, os protocolos são capazes de garantir a entrega dos dados, a confiabilidade da rede, a melhor rota, além do melhor desempenho, entre outras características. Os principais aspectos dos protocolos para NoCs (BJERREGAARD, 2006) são apresentados a seguir:

- Chaveamento por circuito: Existe uma rota dedicada para o caminho do dado. Esta rota (circuito) permanece reservada até que a transmissão do dado acabe.
- Chaveamento por pacote: Pacotes trafegam pela rede e o caminho é definido através de roteamento por salto. Não há reserva de caminho e, portanto, vários pacotes podem compartilhar o mesmo meio.
- Orientada a conexão: Existe uma conexão lógica entre origem e destino. Portanto, a transmissão só inicia após a confirmação de um estado de pronto entre o transmissor e receptor.
- Não orientada a conexão: Não existe uma conexão lógica. A comunicação é estabelecida sem o acordo entre origem e destino.
- Roteamento determinístico: A definição da rota é feita pela origem e destino. Um exemplo utilizado pelas topologias *mesh* e *torus* é o roteamento XY. Primeiro são percorridas as linhas e depois as colunas até o destino.
- Roteamento adaptativo: A definição da rota é feita a cada salto de roteamento. Neste caso, um roteador intermediário pode alterar o caminho que o pacote seguirá.

- Roteamento mínimo ou não mínimo: O roteamento é mínimo se sempre é feita a escolha do menor caminho entre origem e destino.
- Atraso versus perda: No caso de um protocolo baseado em atraso, o pior caso é um pacote atrasado. No caso de um protocolo baseado em perda, é possível excluir um pacote da rede, portanto, uma retransmissão seria o pior caso.
- Controle central ou distribuído: Pelo controle central, o roteamento é feito de forma global. No caso do distribuído, cada roteador define a melhor rota para o pacote.

Existem três técnicas de encaminhamento de pacotes conforme descrição a seguir:

- *Store-and-Forward*: Todo o pacote é armazenado em *buffers* do roteador para que o cabeçalho seja analisado. Em seguida o pacote é encaminhado.
- *Wormhole*: Enquanto o pacote é recebido seu cabeçalho é analisado. Definida a rota, todo o pacote é encaminhado para o nó seguinte sem a necessidade de armazenamento temporário em *buffers*.
- *Virtual cut-through*: Segue o mesmo mecanismo do *wormhole*, mas o nó antes de encaminhar espera uma confirmação do próximo nó destino para envio do pacote.

Os mecanismos para controle de fluxo são responsáveis por garantir o funcionamento da rede. Entre os principais benefícios estão:

- Garantir que pacotes não sejam descartados.
- Evitar retransmissões de pacotes.
- Diminuir o congestionamento da rede ou contenções nos roteadores.
- Otimizar o uso de recursos da rede.
    - *Buffers* menores.
    - Menor número de transmissão de pacotes.
    - Menor uso dos *buffers* e roteadores.
    - Menor consumo de potência e energia.
    - Melhor desempenho da rede.

Um dos conceitos básicos do controle de fluxo é ajustar a taxa de saída de dados de um transmissor para um receptor. Três abordagens clássicas são descritas a seguir:

- *Handshake*: também chamado de “aperto de mão”, consiste em um acordo entre transmissor e receptor através de linhas adicionais de controle para liberação do envio de pacotes. Transmissor envia sinal de solicitação e receptor envia sinal de *buffer* livre.

- Créditos: o receptor envia a quantidade de espaço livre no *buffer* de entrada para que o transmissor saiba quantos créditos estão disponíveis para envio de pacotes.
- Canais virtuais: técnica para eliminar o problema *head of the line blocking* já descrito na Subseção 3.1, em que o primeiro pacote bloqueia os demais da fila. Através de canais virtuais é possível criar diversas saídas do *buffer* de entrada, como se este tivesse várias pequenas filas.

Apesar do projeto de protocolos visar a entrega dos dados, existem problemas que podem ocorrer e impedir que pacotes cheguem ao destino. Estes problemas são conhecidos como *deadlock*, *livelock* e *starvation*, e devem fazer parte das preocupações de um projetista de NoC.

- *Deadlock*: O *deadlock* é a representação de uma dependência cíclica. Neste caso, um pacote não consegue progredir e fica restrito a um subconjunto de estados ou roteadores.
- *Livelock*: O *livelock* é a representação de uma contínua retransmissão do pacote sem atingir o nó destino. Comum em protocolos de roteamento.
- *Starvation*: O *starvation* é a representação da não alocação de um recurso devido a postergação indefinida de acesso ao mesmo. Comum em protocolos de arbitragem.

3.4. Análise da Viabilidade das NoCs

A Tabela 2 apresenta as vantagens e desvantagens da adoção de uma NoC em relação às soluções tradicionais (barramento e chave *crossbar*) mais utilizados atualmente nos processadores *multi-core*.

Apesar das vantagens das soluções tradicionais no que diz respeito à simplicidade, compatibilidade e latência, os limites físicos impostos pelo fio, questões relacionadas à escalabilidade e largura de banda apontam para a NoC como a melhor alternativa para futuras gerações de processadores *many-core*.

Tabela 2: Análise comparativa para uso de NoCs (adaptado de BJERREGAARD, 2006) (FREITAS, 2009d)

Tipo de Interconexão		Prós (+) e Contras (-)	
Barramento	Fio	O aumento do fio aumenta a resistência degradando o desempenho.	-
Chave *Crossbar*		O aumento do fio aumenta a resistência degradando o desempenho.	-
Network-on-Chip		Os fios são ponto-a-ponto entre roteadores e o desempenho não degrada em função do aumento de nós.	+
Barramento	Árbitro	O árbitro é um gargalo à medida que o número de nós aumenta.	-
Chave *Crossbar*		O árbitro pode ser centralizado ou descentralizado e não é o fator principal para degradação do desempenho em função do aumento dos nós.	+-
Network-on-Chip		As decisões de roteamento são distribuídas e não representam um gargalo.	+

Tipo de Interconexão		Prós (+) e Contras (-)	
Barramento	Largura de banda	A largura de banda é limitada e compartilhada por todos os nós.	-
Chave *Crossbar*		Cada interconexão é independente e a largura de banda de comunicação por conexão não é afetada pelas demais.	+
Network-on-Chip		A largura de banda não é afetada pelo aumento da rede.	+
Barramento	Latência	Latência é afetada pelo fio.	+
Chave *Crossbar*		Latência é afetada pelo fio.	+
Network-on-Chip		Latência é afetada pelas contenções em roteadores	-
Barramento	Compatibilidade	Em sua maioria são compatíveis com qualquer IP (*Intelectual Property*) incluindo os softwares.	+
Chave *Crossbar*		Em sua maioria são compatíveis com qualquer IP (*Intelectual Property*) incluindo os softwares.	+
Network-on-Chip		São necessários adaptadores (*wrappers*) entre os IPs, e os softwares precisam de sincronização em sistemas *multi-core*.	-
Barramento	Complexidade	Conceitos simples e bem compreendidos.	+
Chave *Crossbar*		Conceitos simples e bem compreendidos.	+
Network-on-Chip		Projetistas precisam de uma reeducação em função dos novos conceitos.	-

3.5. NoCs Baseadas em Clusters

As arquiteturas mais comuns de NoCs são baseadas em um roteador por núcleo. No entanto, existem propostas de NoCs baseadas em *clusters* de núcleos e/ou memórias. Nesta subseção, três arquiteturas de NoCs são ressaltadas como forma de ilustrar as diferenças em relação às arquiteturas tradicionais.

A CHNoC (*Cluster-based Hierarchical NoC*) (LENG, 2005) é uma proposta de NoC baseada em uma topologia irregular e *clusters* de núcleos de processamento e memórias. A arquitetura é composta por roteadores de dois tipos: *core* e *edge*. *Core Routers* são os roteadores intermediários responsáveis pela interconexão da rede. *Edge Routers* são responsáveis pela interconexão de um *cluster* ao restante da rede. Cada *cluster* é composto por um sistema de memória, núcleos de processamento, um barramento de interconexão e uma interface de rede para conexão com um *Edge Router*. Os roteadores são circuitos dedicados com *buffers* de saída. A argumentação por este tipo de *buffer* é feita com base nos seguintes pontos: melhor aproveitamento da largura de banda e redução do efeito de latência. O protocolo utilizado é *wormhole* como uma estratégia para otimizar o uso dos *buffers*. A técnica de roteamento é baseada em uma abordagem distribuída e parcialmente adaptativa. O experimento de validação da CHNoC foi através de descrição em SystemC e mapeamento de uma aplicação *MPEG-4*

decoder. Por se tratar de uma topologia dedicada e irregular, projetada para mapear um domínio específico de aplicação, constatou-se um bom desempenho da proposta.

A proposta GigaNoC (NIEMANN, 2005) é baseada em uma arquitetura de NoC com topologia regular para o contexto de múltiplos Processadores de Rede. A arquitetura é composta pelos seguintes componentes: *Switch Box* (SB - equivalente ao roteador), barramento do *cluster*, *link* de comunicação entre SBs. Cada *cluster* é composto por múltiplos núcleos de processamento, memórias ou dispositivos dedicados. Os componentes do *cluster* são interconectados por um barramento que também está conectado ao *Switch Box*. As topologias utilizadas para ilustrar a proposta GigaNoC são: *mesh*, *torus* e *butterfly*. Cada *Switch Box* é composto por duas partes principais: i) Portas de entrada / saída e chave *crossbar*. As portas de entrada são compostas por *buffers*. ii) Estrutura de controle entre os núcleos e a rede. A proposta foi verificada através de descrição em VHDL (*VHSIC Hardware Description Language*) obtendo bons resultados de área ocupada e freqüência em FPGA (*Field Programmable Gate Array*).

O grande problema da aglomeração de núcleos em uma topologia fixa está na capacidade desta topologia em mapear os diversos domínios de aplicação (BJERREGAARD, 2006). Portanto, para um caso específico esta aglomeração pode resultar em alto desempenho e em outros casos um baixo desempenho. Como conseqüência desta baixa flexibilidade, algumas pesquisas exploram os conceitos de computação reconfigurável, com foco principal em FPGA, para aumentar a capacidade da NoC de se adaptar a novos padrões de comunicação. Uma alternativa é aumentar a flexibilidade do roteador para que este possa processar também partes dos programas em execução.

Seguindo esta linha de aumento de flexibilidade, o projeto da MCNoC (*Multi-Cluster NoC*) (FREITAS, 2008b) apresenta uma arquitetura baseada em *clusters* de núcleos de processamento, porém com roteadores programáveis. Através da programação de cada roteador da MCNoC, topologias são reconfiguradas em função de um determinado tipo de carga de trabalho. A MCNoC é apresentada novamente na Seção 5.1, ressaltando a característica programável para suporte às cargas de trabalho descritas na Seção 4.

4. Influência das Cargas de Trabalho no Projeto de NoCs

O projeto de uma determinada arquitetura deve ter como critérios dois itens principais: desempenho e custo. Espera-se da arquitetura um alto desempenho, e um baixo custo. No entanto, esta relação é subjetiva, principalmente quando se tem um referencial. Portanto, isso leva a algumas conclusões:

- Desempenho: Quanto menor o tempo de processamento/roteamento de pacotes pela NoC, melhor é o desempenho.
- Custo: Quanto menor o número de componentes para implementar a NoC para resolver o problema de roteamento de pacotes, melhor é o custo. No custo também estão envolvidos os consumos de potência e energia. Quanto menores os consumos de potência e energia, melhor é o custo da NoC.
- Referencial: Existe uma arquitetura de NoC como referência para o projeto e comparação de uma nova NoC. Se o desempenho da nova NoC for muito alto, pode-se esperar um custo superior ao referencial.
- Relação desempenho / custo: Esta relação é importante para verificar o quanto se ganha no aumento de desempenho em função do custo. Mesmo com um custo maior do que a arquitetura de referência, se o desempenho for muito alto, quanto maior a relação desempenho/custo para o novo projeto da NoC, melhor é esse projeto da NoC.

É importante ressaltar que um alto desempenho não se justifica sozinho. Um alto consumo de energia pode representar uma alta conta de energia elétrica, e neste caso, o alto desempenho, ou apenas um desempenho superior ao referencial, pode não valer a pena.

Quando se projeta a arquitetura de uma NoC, estes dois critérios devem ser avaliados e levados em conta. Porém, a avaliação depende de uma NoC já projetada, e antes de projetar, duas perguntas devem ser feitas:

- Qual o contexto de uso da NoC?
- Qual ou quais as cargas de trabalho que serão submetidas à NoC?

O contexto pode estar associado a um processador de propósito geral (GPP: *General-Purpose Processor*) e, portanto, cargas de trabalho de propósito geral. Ou seja, cargas diferentes ou categorias diferentes de programas podem ser executadas em um processador GPP.

Por outro lado, o contexto pode ser um processador dedicado para um determinado sistema embarcado. Cada tipo de sistema embarcado possui um conjunto específico de cargas de trabalho, que podem possuir comportamentos pré-determinados.

O contexto de propósito geral ou dedicado representa, em outras palavras, a necessidade de observação e caracterização das cargas de trabalho. O comportamento das cargas de trabalho demanda um determinado tipo de arquitetura do processador e da própria NoC que suporta a comunicação entre os núcleos. A adequação de uma determinada arquitetura de NoC às cargas de trabalho representa a priori um alto desempenho, mas pode acarretar em alto custo.

Considerando que uma NoC para processadores dedicados encaminha pacotes entre origens e destinos pré-definidos pelo próprio comportamento da carga, ou encaminha pacotes com maior freqüência entre determinados núcleos, pode-se esperar que a NoC tenha uma arquitetura dedicada, ou seja, com topologia fixa. Isto não inviabiliza projetos de NoCs com arquiteturas reconfiguráveis, ou seja, que podem adaptar-se a um novo padrão de comunicação das cargas de trabalho.

Esta mudança no padrão de comunicação das cargas de trabalho é mais freqüente em processadores de propósito geral. Nestes processadores a variedade de cargas, além da alternância de comportamento da própria carga, é muito maior, e nesses casos há uma demanda por NoCs adaptáveis. A adaptação da topologia ou arquitetura de uma NoC em relação aos padrões de comunicação das cargas de trabalho podem aumentar o desempenho dessa NoC em relação a uma dada arquitetura referencial de NoC. No entanto, há que se ressaltar qual é o custo de adaptação em relação ao desempenho.

Um detalhe que é muito importante está relacionado à nova e futura geração de processadores. Os processadores *multi-core* podem ser suportados por soluções simples de interconexões, tais como, barramentos e chaves *crossbar*. No entanto, para processadores *many-core*, a NoC é necessária ou essencial para prover a comunicação entre todos os núcleos. Nos dois casos, existem múltiplos núcleos, no *multi-core* baixa quantidade, e no *many-core* uma quantidade elevada. Porém, ambos são nativamente paralelos, ou seja, vários núcleos para suportar várias cargas simultâneas. Partindo dessa análise, explorar alto desempenho em processadores *multi/many-core* passa necessariamente por um programa paralelo.

Programas paralelos possuem comportamentos baseados em comunicação coletiva (DUATO, 2002). Podem ser desenvolvidos através de modelos por compartilhamento de memória ou passagem de mensagem. Mas o fato dos processos ou *threads* em execução trabalharem de forma coletiva implica em comunicação entre origens e destinos variados, que não possuem uma pré-determinação quando várias cargas estão em execução e dependem de uma correta alocação ou mapeamento em núcleos, que também pode variar. Processadores GPP suportam este tipo de carga de trabalho. Portanto, o projeto da NoC não pode supor que haja comportamento específico e que assim, possa haver uma topologia pré-determinada, uma vez que as técnicas de programação paralela podem variar. Conceitos de computação reconfigurável aplicados ao projeto da arquitetura da NoC podem significar o diferencial de desempenho e custo.

5. Propostas de NoCs Adaptáveis às Cargas de Trabalho

Apesar de cada arquitetura de NoC ser definida em projeto para atender a um determinado tipo de aplicação, para contextos de propósito geral, a flexibilidade é uma característica necessária para o aumento do desempenho. Portanto, nesta seção são apresentadas abordagens de NoCs adaptáveis que objetivam melhorar o processamento de cargas de propósito geral através de mudanças em *software*, arquitetura, ou *hardware*.

5.1. NoCs com Roteadores Programáveis

Aumentar o desempenho, através do processamento de instruções da aplicação nos roteadores da NoC (FERNANDES, 2008), é a alternativa apresentada pela IPNoSys (*Integrated Processing NoC System*). De acordo com Nguyen (2007) uma arquitetura *dataflow* pode ser explorada através do uso de NoCs. Instruções dentro dos laços de repetições são executadas pelos roteadores da NoC conforme o grafo de fluxo de dados gerado durante a compilação da aplicação.

Baseado em Nguyen (2007), a IPNoSys é uma proposta de NoC com roteadores ativos, ou seja, roteadores que processam um pacote de instruções da aplicação. As principais características da IPNoSys são: topologia *mesh* ou *torus* 2D, roteamento XY, técnicas de encaminhamento *virtual cut-through* combinado com *wormhole*, dois canais virtuais, controle de fluxo baseado em crédito, arbitragem distribuída e memorização na entrada. Os roteadores são compostos pelos seguintes blocos: uma unidade lógica e aritmética e uma unidade de sincronização. Os núcleos de processamento foram substituídos por núcleos de acesso à memória. Portanto, a proposta IPNoSys é uma arquitetura totalmente *dataflow*, e executa todas as instruções na rede e não somente as pertencentes aos laços de repetições. O modelo de computação da IPNoSys é baseado em passagem de pacotes entre roteadores, caracterizando um *pipeline* e explorando o paralelismo das transmissões. A IPNoSys foi descrita em SystemC no nível de precisão de ciclos e para sua simulação foi necessário o projeto de uma linguagem de descrição de pacotes. A dependência de dados foi realizada através do grafo de fluxo de dados utilizado como linguagem intermediária para a descrição de pacotes. De acordo com os resultados, a IPNoSys possui um bom desempenho executando aplicações que tratam situações de *deadlock* e executam a transformada discreta do cosseno. A IPNoSys é uma proposta com topologias fixas e a pesquisa ainda não aponta o estudo dos padrões de comunicação para possível adaptação da arquitetura.

A MCNoC (FREITAS, 2008b) é uma arquitetura baseada em múltiplos *clusters* de núcleos de processamento. Cada *cluster* é gerenciado por um roteador programável (FREITAS, 2008a). Este roteador possui um Processador de Rede em *Chip*, chamado de NPoC (*Network Processor on Chip*) utilizado para adicionar flexibilidade no processamento / roteamento de pacotes através de programas de rede. Através do roteador programável é possível explorar características de reconfiguração aplicadas a uma chave *crossbar* com topologias reconfiguráveis (FREITAS, 2009b). A programação do NPoC é utilizada para identificar e implementar topologias em conformidade com os padrões de comunicação das cargas de trabalho. A reconfiguração baseada na chave *crossbar* não precisa de um dispositivo de *hardware* programável tal

como o FPGA (*Field Programmable Gate Array*), uma vez que a própria chave *crossbar* é utilizada como a plataforma para a reconfiguração de topologias, chamada de segundo nível de reconfiguração. O segundo nível de reconfiguração associa a reconfiguração à chave *crossbar*, podendo, nesse caso, haver um primeiro nível baseado em FPGAs ou PASICs (*Polymorphic Application Specific Integrated Circuits*). As Subseções 5.2 e 5.3 apresentam propostas de NoCs baseadas nestes tipos de reconfiguração.

5.2. NoCs Reconfiguráveis Baseadas em FPGA

Em (BARTIC, 2005) são apresentadas vantagens nas reconfigurações de topologias em função de um determinado domínio de aplicação. Uma das principais características do projeto é a arquitetura do roteador capaz de interconectar mais de um componente ou núcleo de processamento. Esta liberdade é conseguida através da nova implementação do roteador na plataforma FPGA. A técnica de encaminhamento de pacotes é baseada em *virtual cut-through* e o algoritmo de roteamento é determinístico com uma capacidade limitada de adaptabilidade. Cada roteador possui uma chave *crossbar* que utiliza técnica de arbitragem por saída para resolver problemas de conflitos entre as entradas. As portas de saída possuem *buffers* para alocação dos pacotes. A escolha foi feita em função do problema de *head of the line blocking*. O controle de fluxo é baseado na técnica *handshake* com linhas de requisição e confirmação. A arquitetura foi descrita em VHDL, e para cada protótipo foi implementada uma topologia de rede diferente, conforme relação a seguir: hipercubo, *mesh*, *torus*, árvore e *crossbar*. Os resultados e conclusões apontam para a escolha da topologia em função dos seguintes parâmetros: área, largura de banda, latência, coeficiente de saturação da rede e limitações físicas.

Em (HUR, 2007) é feito um estudo da adaptação da chave *crossbar* em função de domínios de aplicação. Ressalta-se neste trabalho que a chave *crossbar* é um dos blocos principais utilizados em roteadores e NoCs de diversos tipos, por este motivo, a importância do estudo. A técnica se baseia na implementação de chaves *crossbar* parciais. Uma chave *crossbar* completa possui *n* multiplexadores por porta, enquanto que na chave *crossbar* parcial o número de multiplexadores é variável dependendo do grafo de topologia da aplicação. A implementação dessa proposta é baseada no protocolo de controle de fluxo *handshake* e na técnica de arbitragem *round-robin*. O protótipo foi implementado em FPGA, e avaliado através de alguns grafos de aplicações. O objetivo principal era avaliar a redução de área e o desempenho da chave *crossbar* parcial em relação à completa. Em todos os resultados houve redução de área e melhor desempenho.

PNoC (*Programmable NoC*) (HILTON, 2006) é um projeto que explora a programação do FPGA para reconfiguração das interconexões da NoC. A topologia de rede proposta consiste de uma série de subredes que contém roteadores, núcleos de processamento, entre outros elementos. PNoC é baseada em chaveamento por circuito e controle de fluxo baseado em *handshake*. Os roteadores trabalham com tabelas de roteamento, *buffers* nas portas de entrada, e um árbitro central. Cada roteador é parametrizado em número de portas e largura dos dados e endereços das linhas de conexão de cada porta. O protótipo foi implementado em FPGA e os resultados principais, baseados em área ocupada e freqüência de operação, apontam a PNoC mais eficiente em comparação realizada com barramentos.

A reconfiguração parcial e dinâmica é explorada através da proposta de rede CoNoChi (PIONTECK, 2008). A CoNoChi (*Configurable Network-on-Chip*) suporta mudanças de topologias de rede em tempo de execução adicionando ou removendo elementos de rede sem parar seu funcionamento. A arquitetura da CoNoChi consiste em fatias de reconfiguração independentes que possuem *switches*, *links* ou unidades de processamento. O roteamento é realizado através de tabelas de rotas, que são distribuídas por uma unidade central para cada fatia, e o encaminhamento de pacotes está baseado no algoritmo *virtual cut-through*. A pilha de protocolos é baseada em endereços físicos e lógicos. Os endereços físicos são usados pelos elementos de rede para roteamento, os endereços lógicos pelas aplicações. O árbitro de cada *switch* funciona de forma adaptativa em função de parâmetros de cabeçalho, levando em consideração questões como qualidade de serviço, que também é suportado pela CoNoChi. Como a reconfiguração dinâmica e parcial não é trivial, o FPGA foi utilizado para verificar a viabilidade de implementação. A avaliação da proposta foi feita com o auxílio de uma simulação em SystemC.

A remoção de *links* de comunicação da NoC é a proposta feita por Wang (2008). Neste trabalho o objetivo é reduzir o custo de uma NoC através de uma análise prévia do tráfego de rede. Se em determinado *link* houver um tráfego baixo ou inexistente, é feita a remoção do *link*. A proposta também se baseia na redução do custo associado às políticas para tratamento a *deadlocks*, uma vez que com um número menor de *links* de comunicação, este tratamento fica mais fácil. A NoC proposta consiste de uma topologia *mesh*, roteamento *wormhole*, roteadores com portas de entrada *bufferizadas*, chave *crossbar* 5x5 e módulo de arbitragem. À medida que os *links* de comunicação da NoC são removidos, a topologia irregular se adapta ao tráfico da rede. Os principais resultados do trabalho se baseiam na redução de área e, principalmente, no aumento do desempenho através da NoC modelada em um simulador escrito em C++.

Em um caminho contrário ao proposto em (WANG, 2008), em (KIM, 2005) é proposto o aumento da largura de banda dos barramentos da chave *crossbar* para atender a necessidade de maior vazão em função do aumento do tráfego entre dois nós comunicantes. Este aumento da largura de banda se faz através de *links* adicionais para prover a comunicação. No mesmo caminho de (HUR, 2007) o estudo é focado na chave *crossbar* usada para o projeto de NoCs. Outras propostas de chaves *crossbar* reconfiguráveis (EGGERS, 1996) (FEWER, 2003) não abordam o contexto NoCs.

5.3. NoCs Reconfiguráveis Baseadas em ASICS Polimórficos

Metodologias para projeto de topologias de NoCs em função de domínios de aplicações são apresentadas em (BERTOZZI, 2005) e (HO, 2006). Estes trabalhos não propõem a reconfiguração de topologias, mas uma metodologia de avaliação da carga de trabalho para projeto específico de topologia em ASIC (*Application Specific Integrated Circuit*). No entanto, é possível encontrar na literatura propostas de NoC polimórficas baseadas em plataformas ASIC programáveis.

Em (MERCALDI-KIM, 2008) é apresentado a *Polymorphic On-Chip Network*. A principal motivação do trabalho está na adaptação da NoC, sem o uso de FPGAs, em função de padrões de tráfego. A NoC consiste de uma coleção de blocos de rede configuráveis, tais como *buffers* e *crossbar*. Os parâmetros adaptáveis são: largura de

banda do *link*, *buffers* e topologia da rede. As principais características da NoC avaliada são: topologias *butterfly*, *fat tree*, *flattened butter*, *mesh* e *ring*, algoritmo de roteamento determinístico e algoritmo de encaminhando *store-and-forward* e *wormhole*. A reconfiguração é baseada na repetição de padrões de blocos chamados de Regiões e *Crossbars*. Cada conexão dentro da *crossbar* é estaticamente configurável e cada região consiste de múltiplas fatias da rede compostas pelos seguintes elementos: *buffers* de entrada, roteadores, conexões *crossbar* e árbitros. A reconfiguração é baseada na ativação de conexões entre Regiões e *Crossbars*. A política de reconfiguração não está definida, mas propõe-se que seja feita através do sistema operacional ou alguma aplicação em tempo de execução. Os resultados foram validados através de um simulador de rede (WANG, 2003) executando aplicações de Pareto. A área é reduzida em função da adaptação da topologia da rede sem perder em desempenho.

Seguindo a mesma idéia de adaptação da topologia, em Stensgaard (2008) é apresentado o projeto da ReNoC. A proposta se baseia na inserção de uma nova camada entre os roteadores e os *links*. Esta nova camada consiste de interfaces responsáveis por ativar os *links* entre os roteadores. Desta forma, é possível alterar a topologia de uma *mesh* para uma nova topologia que segue o padrão de comunicação identificado na carga de trabalho. A adaptação está baseada em uma configuração de inicialização resultante de um grafo de topologias das aplicações. Os resultados da ReNoC focam em aplicações baseadas em *Video Object Decoder* para verificar área, e consumo de potência e energia.

Ao contrário das propostas anteriores, em (JOSEPH, 2008) não é feita uma adaptação da topologia através de interconexões, mas um mapeamento de nós de computação através do grafo da aplicação. A NoC chamada de RECONNECT é um ASIC composto de fatias de *arrays* regulares baseado na topologia *honeycomb* (STOJMENOVIC, 1997). Cada aplicação é fragmentada em subestruturas chamadas de *HyperOps*. Um *HyperOp* é um sub-grafo do fluxo de dados da aplicação em execução. Em tempo de execução, os elementos de computação, que constituem as fatias de *array*, são agregados para mapear as operações do *HyperOp*. Esta agregação e mapeamento é a configuração proposta para que a NoC se adapte ao domínio de aplicação. A RECONNECT consiste da seguinte estrutura: interface de rede entre roteador e elementos de computação, roteadores com 4 saídas para a rede (norte, sul, leste, oeste), algoritmo de roteamento XY, chave *crossbar* de roteador com arbitragem de saída (algoritmo *round-robin*), *buffers* nas portas de entrada e mecanismo de controle de fluxo por canais virtuais. Os resultados verificaram uma redução da área e potência em relação a topologia *mesh* e o aumento do desempenho através da execução de *kernels* de aplicações baseadas em multimídia e processamento de sinais digitais.

Duas das propostas apresentam o termo Polimorfismo em ASICs aplicado a NoCs quase simultaneamente, Mercaldi-Kim (Estados Unidos) em junho de 2008 e Joseph (Índia) em julho de 2008. No entanto, em termos de adaptação de topologias, apenas o primeiro trabalho satisfaz a característica esperada de mudança da forma ou das interconexões realizadas capazes de mapear um padrão de comunicação. Em Stensgaard (2008), apesar do termo polimorfismo não aparecer, a característica da NoC se enquadra nos conceitos PASIC.

Por fim, uma característica comum às NoCs PASIC é o fato da flexibilidade existir, mas não ser tão alta. Afinal, em uma plataforma ASIC não é possível adicionar

elementos ao projeto implementado, mas alterar e remover conexões seguindo restrições estabelecidas pelo projeto inicial implementado no ASIC. Por este motivo, as propostas que utilizam FPGA são mais flexíveis, podendo usar em novas configurações, componentes ainda não utilizados do dispositivo. Além disso, o fato do FPGA possuir inúmeros blocos lógicos, a realocação das unidades construtivas da NoC após uma reconfiguração possui um grau de liberdade maior, devido a alta flexibilidade do FPGA. A grande vantagem do PASIC é justamente o desempenho e o menor tempo de reconfiguração.

6. Método de Projeto e Avaliação de NoCs

No projeto de uma NoC, é necessário definir algumas etapas ou fases que são importantes para o conhecimento do contexto de aplicação. Para tal, as primeiras etapas são as seguintes:

- Analisar o contexto de aplicação do processador *many-core*.
- Analisar quantos núcleos possui o processador *many-core*.
- Analisar e caracterizar as cargas de trabalho para este processador *many-core*.

Essas etapas dizem respeito às seguintes afirmações:

- O projeto de uma arquitetura de NoC deve se basear no conhecimento do sistema (contexto + processador) que demanda o projeto.
- O projeto de uma arquitetura de NoC deve se basear no conhecimento da carga de trabalho que demanda o processamento / roteamento de pacotes da NoC.

Um projeto de NoC que não atende as três primeiras etapas corre um alto risco de ter como resultado uma arquitetura de NoC que não possui bom desempenho por não ser representativa ao contexto de aplicação e às cargas de trabalho em execução.

Em complemento às primeiras etapas, é necessário definir corretamente quais os protocolos e topologias, que podem influenciar diretamente em decisões, por exemplo, de tamanho de *buffers* de portas de entrada, e conseqüentemente, tamanho e consumo de potência da própria NoC, ou seja, o custo. Portanto, as etapas seguintes são:

- Analisar requisitos de projeto para definição de protocolos e topologias.
- Analisar o impacto (custo) das definições na arquitetura em projeto.

Uma vez que o sistema e as cargas de trabalho são conhecidas, além de requisitos de protocolos e topologias, são necessárias etapas para avaliação do projeto arquitetural da NoC. Para avaliação de desempenho e custo é possível adotar três tipos de modelos (JAIN, 1991): analítico, simulação e medição (experimental).

O modelo analítico é baseado em fórmulas matemáticas capazes de expressar características da arquitetura da NoC e fornecer resultados ou estimativas para implementação.

O modelo de simulação é baseado em algoritmos e fórmulas matemáticas. O simulador desenvolvido é um *software* capaz de aumentar o controle e variedades de testes sobre a NoC em avaliação, propiciando um maior número de estimativas em relação ao modelo analítico.

Através do modelo de medição, ou resultados são reais e não estimativas. Porém, a medição só pode ser utilizada se existe um protótipo da NoC, ou seja, a NoC tem que ser real. A grande vantagem está em não fornecer resultados baseados em estimativas, como os modelos anteriores, porém, o controle em um sistema real é mais complexo, estando a NoC sujeita a condições não esperadas de teste.

Cada modelo possui vantagens ou desvantagens, que podem estar associados ao tempo para desenvolver um simulador ou protótipo, ou nas estimativas fornecidas pela simulação ou modelo analítico. No entanto, é de se esperar que durante a fase de projeto e avaliação de uma NoC, a mesma não seja prototipada antes de uma avaliação em modelos analíticos ou de simulação, que podem ser indicativos para correções antes de uma versão real existir, que nesse caso, possui um custo de reprojeto mais alto. Portanto, o uso em conjunto dos três modelos durante projeto de uma NoC é uma alternativa interessante, conforme ilustração da Figura 8.

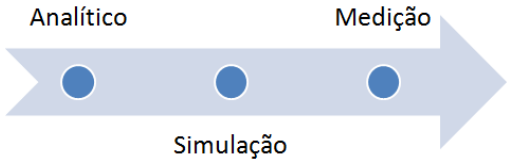

Figura 8: Fluxo de projeto baseado em modelos de avaliação

Etapas de avaliação de projeto, que estão associadas aos modelos, são as seguintes:

- Definir o momento e o propósito de uso da cada modelo.
- Definir as métricas de avaliação para a NoC.
- Escolher as ferramentas (*softwares* ou *hardwares*) que serão utilizados para avaliação da NoC.
- Definir pontos de instrumentação ou coleta para modelos de simulação e medição.
- Avaliar resultados de desempenho e custo.

Para simulação de NoCs, é possível usar ambientes de descrição de sistema e *hardware*, baseados em linguagens, tais como, SystemC (GHOSH, 2001) (RIGO, 2004), VHDL ou Verilog (ORDONEZ, 2003). Ambientes de simulação e síntese de *hardware* (MENTOR, 2008) (XILINX, 2008) possibilitam a verificação do projeto e avaliação de desempenho e custo. O uso do FPGA é uma boa alternativa para verificação de funcionamento de protótipo, além de ser útil para a utilização do modelo de medição.

O modelo analítico é considerado o mais rápido e simples para ser utilizado. No entanto, pode limitar a avaliação, uma vez que seu uso pode restringir o número de variáveis pela dificuldade de controle do modelo para combinações de testes. A seguir são apresentadas duas fórmulas que são úteis para avaliações preliminares.

Para cálculo de tempo de transmissão, duas métricas principais da rede são: latência (L) e largura de banda (B). O tempo de transmissão depende também do tamanho do pacote (P) que deve ser transmitido e do número de pacotes (n). Portanto, o tempo final de transmissão (T) pode ser expresso da seguinte forma (Equação 1):

$$T = \left(L + \frac{P}{B} \right) * n \qquad \text{Equação 1}$$

Os resultados de consumo de potência podem ser obtidos através de simulações baseada em descrições do *hardware* em VHDL, conforme já foi descrito. Através dos

resultados de consumo de potência (Cp), e do tempo de transmissão (Tt), é possível expressar o consumo de energia (Ce) de uma NoC através da Equação 2:

$$Ce = Cp * Tt \qquad \text{Equação 2}$$

O projeto de uma NoC depende do projeto e propósito do processador *many-core*. No entanto, outros elementos presentes em um processador *many-core* são influenciados pelo uso de uma NoC, conforme é descrito na Seção 7.

7. Influência das NoCs no Projeto de um Processador Many-Core

O custo de uma comunicação está associado ao número de roteadores na rota de envio de pacotes entre origem e destino. Quanto maior este número, maior a possibilidade de congestionamentos ou atrasos na comunicação. Em um processador *many-core*, isto é um problema grave. Imagine 1000 núcleos de processamento, e uma tentativa de comunicação do núcleo 1 com o núcleo 500. São centenas de roteadores envolvidos nas transmissões dos pacotes. Considerando a demanda por alto desempenho, é de se imaginar que esta comunicação não ajuda em reduzir o tempo final de processamento.

Os roteadores de uma NoC podem ser considerados gargalos nas transmissões de pacotes. Os roteadores são pontos de concentração e precisam ter alta vazão para evitar tempos de espera em fila pelos pacotes, sem contar possíveis perdas, por falta de espaço para alocações dos mesmos. Portanto, o mapeamento de processos em um processador *many-core* deve levar em consideração a influência da NoC. A NoC pode influenciar no projeto de um processador *many-core* com conseqüências em outras partes relacionadas ao projeto, conforme é ilustrado pela Figura 9.

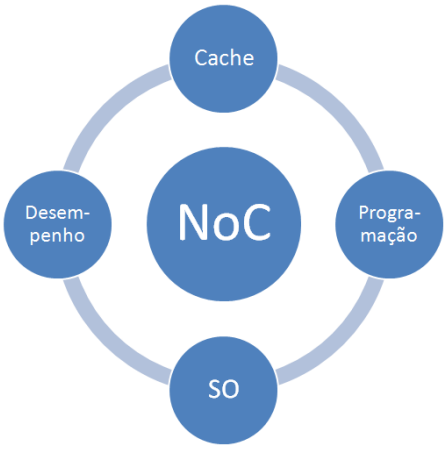

Figura 9: Influência da NoC no projeto de um processador many-core

Sistemas Operacionais (SO), além de suportarem uma quantidade elevada de núcleos, precisam estar preparados para mapear e escalonar processos comunicantes de forma que pacotes gerados para comunicação coletiva não tenham que trafegar por muitos roteadores. Reduzir o número de roteadores para comunicação coletiva, através de um mapeamento otimizado, pode aumentar o desempenho de processamento de aplicações paralelas.

Outra influência da NoC está associada ao compartilhamento de memórias *cache* (GIRÃO, 2007) (ALVES, 2009). Apesar das comunicações entre núcleos, todos os núcleos enxergam uma mesma memória principal. Conseqüentemente, existem níveis de memórias *cache*, e cada grupo de núcleos enxerga apenas uma memória *cache*, ou um conjunto de memórias *cache*, ou todas as memórias *cache*. Esta escolha depende do projeto do processador *many-core*. No entanto, acessar memórias *cache*, passa necessariamente por roteadores, e roteadores são gargalos, podendo reduzir o desempenho de processamento das aplicações. Avaliar quando compartilhar memória, ou não, é um problema influenciado, inclusive, pela NoC.

Como conseqüência das NoCs e da dificuldade de aumentar tamanho e número de portas das memórias *cache*, devido ao custo e aumento da latência de leitura e escrita, uma das alternativas é a memória NUCA (*Non-Uniform Cache Architecture*) (FREITAS, 2009a). A memória NUCA funciona como um conjunto de memórias *cache* pequenas, interconectadas aos roteadores e que podem ser acessadas por vários núcleos do processador, porém, com tempos de acessos não uniformes, ou seja, diferentes. A NUCA é uma boa alternativa para solucionar o problema relacionado ao custo e à alta latência de acesso às memórias, em função do aumento de tamanho das mesmas. Porém, com o uso das NoCs também há o adicional de atrasos impostos pelos roteadores.

As NoCs também podem influenciar o modo de programar. A camada de arquitetura costuma ser abstraída para facilitar a programação. No entanto, se há um conjunto de roteadores, há troca de pacotes, nesse caso, uma programação paralela (DUATO, 2002) baseada em passagem de mensagem pode ser uma boa alternativa. Se o programador utilizar passagem de mensagem para programar um processador *many-core*, é provável que a camada de arquitetura não fique tão abstrata, sendo necessário, para o programador, conhecer um pouco da arquitetura e topologia, para que haja um bom mapeamento de comunicação em *chip*.

Em resumo, o uso das NoCs tem como conseqüência o impacto nas decisões de projeto do processador *many-core*, no projeto de Sistemas Operacionais, na forma de programar, mas todas as decisões possuem algo em comum, o desempenho. Portanto, a arquitetura de uma NoC pode influenciar de forma direta no desempenho do processador *many-core*.

8. Pesquisas e Conferências Relacionadas às NoCs

Pesquisas relacionadas às NoCs têm aumentado significativamente nos últimos anos. Os principais grupos de pesquisa podem ser identificados neste minicurso a partir da apresentação de cada NoC, seja na Tabela 1, ou nas Seções 3 e 5. No entanto, um bom início de estudo sobre NoCs pode ter como base os principais veículos de publicação de artigos científicos. Nesse caso, a principal conferência sobre *Networks-on-Chip* é a NOCS (*International Symposium on Networks-on-Chip*). Além desse simpósio, também se pode ressaltar o NoCArc (*International Workshop on Networks-on-Chip*), que acontece juntamente com o MICRO (*International Symposium on Microarchitecture*). A seguir é apresentada uma lista de algumas das principais conferências relacionadas ao tema:

- *IEEE International Symposium on Networks-on-Chip (NOCS)*
- *ACM International Workshop on Networks-on-Chip (NoCArc)*
- *SBC/IEEE International Symposium on Computer Architecture and High Performance Processing (SBAC-PAD)*
- *SBC/IEEE Symposium on Integrated Circuits and Systems Design (SBCCI)*
- *IEEE International Symposium on Computer Architecture (ISCA)*
- *IEEE International Symposium on Microarchitecture (MICRO)*
- *IEEE International Conference on Field Programmable Logic and Applications (FPL)*
- *IEEE Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP)*
- *IEEE International Symposium on High Performance and Computer Architecture (HPCA)*
- *IEEE International Parallel & Distributed Processing Symposium (IPDPS)*
- *IEEE International Symposium on Circuits and Systems (ISCAS)*
- *SBC Simpósio em Sistemas Computacionais (WSCAD-SSC)*

Provavelmente o WSCAD-SSC é a única conferência brasileira e de língua portuguesa, que aborda o tema devido ao conteúdo de tópicos voltados para arquitetura de computadores.

Em *Networks-on-Chip* é possível elencar vários tópicos para projetos de pesquisa, tais como:

- Projeto de protocolos;
- Roteamento de pacotes;
- Controle de fluxo, detecção e correção de erros;
- Qualidade de serviço;

- Consumo de potência e energia;
- Níveis e tipos de reconfiguração;
- Desenvolvimento de modelos analíticos e simuladores;
- Experimentação e pontos de instrumentação;
- Caracterização de cargas de trabalho;
- Avaliação de desempenho.

Consumo de potência e energia ganha certo destaque devido ao que foi discutido na Seção Introdução. Com a necessidade de manter o constante crescimento de desempenho em processadores, sem o aumento de consumo de potência, surgem os *multi-core*. Essa afirmação deve ser entendida como requisito básico para o projeto de uma NoC de alto desempenho, ou seja, a NoC também precisa consumir pouca potência e energia. O alto consumo da NoC pode prejudicar a eficiência do processador *many-core*.

O conhecimento da carga de trabalho também merece destaque. Conhecer a carga significa saber qual o tipo de arquitetura que melhor pode processar, ou no caso das NoCs, rotear pacotes entre núcleos de processamento. Este tema foi discutido nas Seções 4, 5 e 6 deste minicurso. Não necessariamente cargas de trabalho demandam arquiteturas de NoCs reconfiguráveis, mas projetos que abordam conceitos de reconfiguração em níveis ou usando tipos diferentes procuram encontrar resultados que mostram a eficiência da adaptação à carga de trabalho como vantagem em relação às arquiteturas de NoCs estáticas.

Avaliar desempenho de NoCs pode parecer algo relativamente fácil uma vez que existem vários modelos, métodos e técnicas, que são aplicados em outras áreas. No entanto, ainda é um pouco cedo para dizer que cargas de trabalho para NoCs já são amplamente conhecidas, uma vez que os processadores *many-core* ainda não estão disponíveis e poucos são os projetos de pesquisa que agregam todas as variáveis envolvidas no processador, além da própria NoC. Isso leva à necessidade de bons simuladores e modelos analíticos que possam direcionar os projetos e apresentar resultados confiáveis para futuras experimentações.

Apesar de redes-em-*chip* serem diferentes de redes de computadores, uma vez que latências, largura de banda, componentes, e limitações do ambiente são diferentes, problemas relacionados às redes de computadores também aparecem. Existe uma grande demanda por pesquisas em temas como, por exemplo, roteamento, detecção de erros, e qualidade de serviço.

9. Conclusões

Com o surgimento dos processadores *multi-core* e da proximidade da nova geração dos processadores *many-core*, a questão do desempenho não está envolvida apenas com o núcleo de processamento, seja ele escalar ou superescalar, mas diz respeito também à interconexão que suporta toda a comunicação *intra-chip*. Para esse suporte, a nova abordagem ou alternativa é a *Network-on-Chip*.

Apesar da quantidade relativamente alta de propostas de arquiteturas de NoCs apresentadas pelo estado da arte, existe uma necessidade de validação de arquiteturas em função do contexto de processamento, seja ele de propósito específico ou geral. Aumentar o desempenho das aplicações ou cargas de trabalho em processadores *many-core* depende de escolhas corretas no que diz respeito às arquiteturas dos núcleos, mas sobretudo no que diz respeito à arquitetura da NoC, se é adaptável ou não. Essas escolhas, necessariamente dependem do conhecimento das cargas de trabalho que serão executadas no processador *many-core* / NoC.

A viabilidade das NoCs não está associada apenas a uma grande idéia ou alternativa de suporte, mas da necessidade de solucionar problemas que impedem o crescimento de desempenho dos processadores. A decisão por adotar a NoC como arquitetura de interconexão para processadores *many-core* está baseada em dois aspectos, descritos a seguir, e também estão ilustrados na Figura 10.

- Ao fato de que os processadores precisam ser baseados em múltiplos núcleos, porque depender apenas de aumento de freqüência em um processador *single-core* significa alto consumo de potência e energia, além de soluções caras para manter o processador em temperatura normal de trabalho.
- Relacionado aos limites físicos dos fios, quando estes precisam ser utilizados como interconexão global.

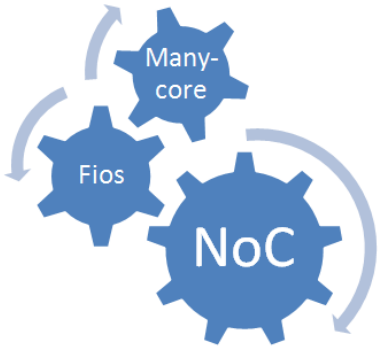

Figura 10: Relação de dependência

Se barramentos ou chaves *crossbars* possuem vantagens como o fato de serem mais simples, estas vantagens desaparecem quando os aspectos levantados são apresentados. Usar pequenos barramentos ou *crossbars* como parte da arquitetura da NoC pode ser uma solução. O uso da NoC pode mudar a forma com se vê o processador, afinal, a NoC possui como característica a troca de pacotes. Portanto, no *chip* de processador há uma rede de comunicação. Como programar? Talvez por

passagem de mensagem. No entanto, todos os núcleos enxergam uma mesma memória. Talvez por memória compartilhada. Talvez pelos dois modelos de programação. O certo é que existem muitas perguntas, e muitas respostas que geram outras perguntas. Ou seja, não faltam problemas relacionados à NoC.

A tese (FREITAS, 2009d) que gerou este minicurso possui mais informações e reforça a importância da caracterização da carga de trabalho para o projeto de arquiteturas de *Networks-on-Chip*. Resultados mostram como o conhecimento do padrão de comunicação pode ajudar no projeto de uma NoC com alto desempenho.

Referências

AHMADI, H., DENZEL, W. E., A Survey of Modern High-Performance Switching Technique, **IEEE Journal on Selected Areas in Communications**, v.7, n.7, p.1091-1103, September 1989.

ALVES, M. A. Z., FREITAS, H. C., NAVAUX, P. O. A., Investigation of Shared L2 Cache on Many-Core Processors, **Workshop on Many Cores, WMC 2009**, in conjunction with the LNCS ARCS 2009, Delft, Netherlands, pp.21-30, 2009.

ANDRIAHANTENAINA, A., et al., SPIN: a scalable, packet switched, on-chip micro-network, **Design Automation and Test in Europe Conference and Exhibition (DATE)**, p.70–73, March 2003.

ASANOVIC, K., et al., The Landscape of Parallel Computing Research: A View from Berkeley, Technical Report No. UCB/EECS-2006-183, **Electrical Engineering and Computer Sciences, University of California at Berkeley**, December 18, 2006.

BARTIC, T. A., et al., Topology adaptive network-on-chip design and implementation, **IEE Proc. Comput. Digit. Tech.**, v.152, n.4, p.467-472, July 2005.

BENINI, L., MICHELI, G. D., Networks on chips: a new SoC paradigm, **IEEE Computer**, v.1, p.70-78, January 2002.

BENINI, L., MICHELI, G. D., Network-on-chip architectures and design methods, **IEE Proceedings Computers & Digital Techniques**, v.152, n.2, p.261-272, 2005.

BERTOZZI, D., et al., NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip, **IEEE Transactions on Parallel and Distributed Systems**, v.16, n.2, p.113-129, February 2005.

BJERREGAARD, T. and MAHADEVAN, S., A Survey of Research and Practices of Network-on-Chip, **ACM Computing Surveys**, v.38, n.1, p.1-51, March 2006.

BOLOTIN, E., et al., QNoC: QoS architecture and design process for network on chip, **The Journal of Systems Architecture, Special Issue on Networks on Chip**, v.50, n.2, p.105–128, 2004.

BORKAR, S., Thousand core chips: a technology perspective, **ACM Proceedings of the 44th Annual Design Automation Conference**, pp.746-749, New York, NY, USA, 2007.

BOUHRAOUA, A., ELRABAA, M. E., A High-Throughput Network-on-Chip Architecture for Systems-on-Chip Interconnect, **IEEE International Symposium on Systems-on-Chip**, p.1-4, 2006.

CHANG, M.-C. F., et al., RF/wireless interconnect for inter- and intra-chip communications, **Proc. of The IEEE**, v.89, n.4, p.456-466, April 2001.

CIDON, I., KOLODNY, A., GINOSAR, R., The Elements of NoC, Tutorial (Slides), **ACM/IEEE International Symposium on Networks-on-Chip (NOCS)**, San Diego, USA, 2009.

COMER, D. E., Network Systems Design Using Network Processors, [S.I.], **Prentice Hall**, 2003.

COMPTON, K., HAUCK, S., Reconfigurable Computing: A Survey of Systems and Software, **ACM Computing Surveys**, v. 34, n.2, p.171-210, June 2002.

DALLY, W., TOWLES, B., Route packets, not wires: on-chip interconnection networks, **38th Design Automation Conference (DAC)**, p.684–689, June 2001.

DUATO, J., YALAMANCHILI, S., NI, L., Interconnection Networks, [S.I.], **Morgan Kaufmann**, 2002.

EGGERS, H., et al., Fast Reconfigurable Crossbar Switching in FPGAs, **6th International Workshop on Field-Programmable Logic and Applications (FPL)**, LNCS 1142, p.297-306, 1996.

FERNANDES, S., et al., IPNoSys: uma nova arquitetura paralela baseada em redes em chip, **Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD)**, p.53-60, 2008.

FEWER, C., et al., A High I/O Reconfigurable Crossbar Switch, **Annual IEEE Symposium on Field-Programmable Custom Computing Machines**, p.3-10, 2003.

FORSELL, M., A scalable high-performance computing solution for networks on chips, **IEEE Micro**, v.22, n.5, p.46–55, 2002.

FREITAS, H. C., NAVAUX, P. O. A., Chip Multithreading: Conceitos, Arquiteturas e Tendências. (Desenvolvimento de material didático ou instrucional), **TI 1253**, **PPGC/UFRGS**, 2006.

FREITAS, H. C., SANTOS, T. G. S., NAVAUX, P. O. A., Design of Programmable NoC Router Architecture on FPGA for Multi-Cluster NoCs, **IET Electronics Letters**, v.44, n.16, p.969-971, July 2008a.

FREITAS, H. C., SANTOS, T. G. S., NAVAUX, P. O. A., NoC Architecture Design for Multi-Cluster Chips, **IEEE International Conference on Field Programmable Logic and Applications (FPL)**, p.53-58, 2008b.

FREITAS, H. C., ALVES, M. A. Z., NAVAUX, P. O. A., NoC e NUCA: Conceitos e Tendências para Arquiteturas de Processadores Many-Core, **Escola Regional de Alto Desempenho (ERAD)**, Cap.3, 2009a.

FREITAS, H. C., NAVAUX, P. O. A., On the Design of Reconfigurable Crossbar Switch for Adaptable On-Chip Topologies in Programmable NoC Routers, **ACM Great Lakes Symposium on VLSI (GLSVLSI)**, p.129-132, 2009b.

FREITAS, H. C., ALVES, M. A. Z., SCHNORR, L. M., NAVAUX, P. O. A., Performance Evaluation of NoC Architectures for Parallel Workloads, **ACM/IEEE International Symposium on Networks-on-Chip (NOCS)**, p.87, 2009c.

FREITAS, H. C., Arquitetura de NoC Programável Baseada em Múltiplos Clusters de Cores para Suporte a Padrões de Comunicação Coletiva, **Tese de Doutorado, PPGC, UFRGS**, 125p., 2009d.

GHOSH, A., et al., System modeling with SystemC, **International Conference on ASIC**, p.18-20, 2001.

GIRÃO, G., OLIVEIRA, B. C., SOARES, R., Silva, I. S., Cache coherency communication cost in a NoC-based MPSoC platform, **ACM Symposium on Integrated Circuits and Systems Design**, p. 288-293, 2007.

GRECU, C., et al., A scalable communication-centric SoC interconnect architecture, **IEEE International Symposium on Quality Electronic Design (ISQED)**, p.343-348, 2004.

HILTON, C. and NELSON, B., PNoC: A Flexible Circuit-Switched NoC for FPGA-based Systems, **IEE Proceedings of Computer & Digital Techniques**, v.153, n.3, p.181-188, May 2006.

HO, R., et al., The Future of Wires, **Proceedings of the IEEE**, v.89, n.4, April 2001.

HO, W. H., PINKSTON, T. M., A Design Methodology for Efficient Application-Specific On-Chip Interconnects, **IEEE Transactions on Parallel and Distributed Systems**, v.17, n.2, p.174-190, February 2006.

HUR, J. Y., STEFANOV, T., WONG, S., VASSILIADIS, S., Systematic Customization of On-Chip Crossbar Interconnects, **International Workshop on Applied Reconfigurable Computing**, p.61-72, 2007.

INTEL, IXP1200 Network Processor Family, Hardware Reference Manual, **Intel**, December, 2001.

INTEL, 80 FPU Cores on a Chip for Teraflop Performance, **Tom's Hardware**, Disponível em: <**http://www.tomshardware.com/2006/09/27/idf_fall_2006/page2.html**>. Acesso em: 20 nov. 2006.

INTEL, 80-Core Programmable Processor First to Deliver Teraflops Performance, **Intel Platform Research,** Disponível em: <**http://www.intel.com/research/platform/index.htm**>**.** Acesso em: 25 out. 2007.

JAIN, R., The Art of Computer Systems Performance Analysis, [S.I.], **John Wiley**, 1991.

JOSEPH, N., et al., RECONNECT: A NoC for Polymorphic ASICs Using a Low Overhead Single Cycle Router, **IEEE**, p.251-256, 2008.

KALLA, R., SINHAROY, B., TENDLER, J. M., IBM Power5 Chip: A Dual-Core Multithreaded Processor, **IEEE MICRO**, v.24, n.2, p.40-47, 2004.

KARIM, F., NGUYEN, A., DEY, S., An interconnect architecture for network systems on chips, **IEEE Micro,** v.22, n.5, p.36-45, 2002.

KEYES, R. W., Fundamental Limits of Silicon Technology, **Proceedings of the IEEE**, v.89, n.3, p.227-239, March 2001.

KEYES, R. W, Moore's Law Today, **IEEE Circuits and Systems Magazine**, v.8, n.2, p.53-54, 2008.

KIM, D., et al., A Reconfigurable Crossbar Switch with Adaptive Bandwidth Control for Networks-on-Chip, **IEEE International Symposium on Circuits and Systems (ISCAS)**, p. 2369 – 2372, 2005.

KONGETIRA, P., et al., Niagara: a 32-way multithreaded Sparc processor, **IEEE MICRO**, v.25, n.2, p.21-29, March-April 2005.

KREUTZ, M., MARCON, C., CARRO, L., CALAZANS, N., SUSIN, A. A., Energy and Latency Evaluation of NOC Topologies, **International Symposium on Circuits and Systems (ISCAS)**, p.5866-5869, 2005.

LAN, Y.-C., LO, S.-H., LIN, Y.-C., HU, Y.-H., CHEN, S.-J., BiNoC: A Bidirectional NoC Architecture with Dynamic Self-Reconfigurable Channel, **ACM/IEEE International Symposium on Networks-on-Chip (NOCS)**, p.266-275, 2009.

LENG , X., et al., Implementation and Simulation of a Cluster-based Hierarchical NoC Architecture for Multi-Processor SoC, **IEEE International Symposium on Communiations and Information Technology**, p.1163-1166, 2005.

LIANG, J., SWAMINATHAN, S., TESSIER, R., aSOC: A scalable, single-chip communications architecture, in: **IEEE International Conference on Parallel Architectures and Compilation Techniques**, p.37–46, October 2000.

MANEVICH, R., WALTER, I., CIDON, I., KOLODNY, A., Best of Both Worlds: A Bus-Enhanced NoC (BENoC), **ACM/IEEE International Symposium on Networks-on-Chip (NOCS)**, p.173-182, 2009

MARESCAUX, T., Networks on chip as hardware components of an OS for reconfigurable systems, **Field-Programmable Logic and Applications (FPL)**, p.595-605, September 2003.

MARTINS, C. A. P. S., et al., Computação Reconfigurável: Conceitos, Tendências e Aplicações, Cap.8, **Jornada de Atualização em Informática**, CSBC, 2003.

MENTOR GRAPHICS, ModelSim, **Mentor Graphics,** Disponível em: <**http://www.mentor.com/products/fv/digital_verification/modelsim_se/**>. Acesso em: 01 nov. 2008.

MERCALDI-KIM, M., Davis, J. D., Oskin, M., Austin, T., Polymorphic On-Chip Networks, **IEEE International Symposium on Computer Architecture (ISCA)**, p.101-112, 2008.

MILLBERG, M., The Nostrum backbone - a communication protocol stack for networks on chip, **Proceedings of the VLSI Design Conference**, p.693-696, January 2004.

MORAES, F., et al., HERMES: an infrastructure for low area overhead packet-switching networks on chip, **Integration the VLSI Journal**, p.69-93, 2004.

NGUYEN, H. T. L., Network-on-Chip Dataflow Architecture, **U. S. p. a. t. office, Ed. United States**, Hanoi Tran Le Nguyen (VN), p.9, 2007.

NIEMANN, J., PORRMANN, M., RÜCKERT, U., A Scalable Parallel SoC Architecture for Network Processors, **IEEE Annual Symposium on VLSI**, p.311-313, 2005.

OGRAS, U. Y. et al., Challenges and Promising Results in NoC Prototyping Using FPGAs, **IEEE Micro**, v.27, n.5, p.86-95, Sept.-Oct. 2007a.

OGRAS, U. Y. et al., On-Chip Communication Architecture Exploration: A Quantitative Evaluation of Point-to-Point, Bus, and Network-on-Chip, **ACM Transactions on Design Automation of Electronics Systems**, v.12, n.3, Article 23, p.1-20, August 2007b.

OLUKOTUN, K. et al., The Case for a Single-Chip Multiprocessor, **7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)**, p.2-11, 1996.

OLUKOTUN, K., HAMMOND, L., The Future of Microprocessors, **ACM Queue**, v.3, n.7, p.26-29, September 2005.

ORDONEZ, E. D. M., et al., Projeto, Desempenho e Aplicações de Sistemas Digitais em Circuitos Programáveis (FPGAs), [S.I.], **Bless Gráfica e Editora Ltda**, 2003.

PIONTECK, T., et al., Adaptive Communication Architectures for Runtime Reconfigurable Systems-on-Chip, **Parallel Processing Letters**, v.18, n.2, p.275-289, 2008.

RIGO, S., et al., ArchC: A SystemC-Based Architecture Description Language, **International Symposium on Computer Architecture and High Performance Processing (SBAC-PAD)**, p.66-73, October 2004.

RIJPKEMA, E., GOOSSENS, K., RADULESCU, A., Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip, **Design, Automation and Test in Europe (DATE)**, p.350–355, March 2003.

SGROI, M., Addressing the system-on-chip interconnect woes through communication-based design, **38th Design Automation Conference (DAC)**, p.667–672, June 2001.

SIGÜENZA-TORTOSA, D., NURMI, J., Proteo: a new approach to network-on-chip, **IASTED International Conference on Communication Systems and Networks (CSN)**, September 2002.

SO, H., Introduction to Reconfigurable Computing, **CS61c sp06 Lecture (5/5/06), Machine Structures**, University of California, Berkeley, 2006.

STENSGAARD, M. B., SPARS, J., ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology, **ACM/IEEE International Symposium on Networks-on-Chip**, p.55-64, 2008.

STOJMENOVIC, I., Honeycomb Networks: Topological Properties and Communication Algorithms, **IEEE Transactions on Parallel and Distributed Systems**, v.8, n.10, p.1036-1042, October 1997.

TODMAN, T. J., et al., Reconfigurable Computing: architectures and design methods, **IEE Proc.-Comput. Digit. Tech.**, v.152, n.2, p.193-207, March 2005.

UNGERER, T, et al., Multithreaded Processors, **The Computer Journal, British Computer Society**, v.45, n.3, p.320-348, 2002.

UNGERER, T., et al., A Survey of Processors with Explicit Multithreading, **ACM Computing Surveys**, v.35, n.1, p.29-63, March 2003.

WANG, H., PEH, L.-S., MALIK, S., Power-driven design of router microarchitectures in on-chip networks. **International Symposium on Microarchitecture**, p.105–115, 2003.

WANG, Y., ZHAO, D., Design and Implementation of Routing Scheme for Wireless Network-on-Chip, **IEEE International Symposium on Circuits and Systems (ISCAS)**, p.1357-1360, May 2007.

WANG, D., et al., A Link Removal Methodology for Networks-on-Chip on Reconfigurable Systems, **IEEE International Conference on Field Programmable Logic and Applications (FPL)**, p.269-274, 2008.

WIKLUND, D., LIU, D., SoCBUS: switched network on chip for hard real time systems, **International Parallel and Distributed Processing Symposium (IPDPS)**, April 2003.

WOLF, W., The Future of Multiprocessor System-on-Chip, **Proceedings of the DAC**, June, 2004.

XILINX, ISE Design Suite, **Xilinx,** Disponível em: <**http://www.xilinx.com/ise/logic_design_prod/**>. Acesso em: 10 nov. 2008.

ZEFERINO, C. A., SUSIN, A. A., SoCIN: A Parametric and Scalable Network-on-Chip, **IEEE Symposium on Integrated Circuits and Systems Design (SBCCI)**, p.169-174, 2003.