Capítulo
6

Estudo de Networks on Chip (NoCs) em FPGAs

Maurício Acconcia Dias, Marcílio F. de Oliveira Neto
Centro Universitário da Fundação Hermínio Ometto
Araras, Brasil

Resumo

Processadores com mais de um núcleo de processamento, os multicores, são atualmente a base para o desenvolvimento de Systems on Chip (SoCs) tanto para aplicações de propósito geral quanto de propósito específico. A evolução da tecnologia de fabricação, aliada a necessidade de explorar o desempenho dos processadores em outros aspectos além da frequência de clock, gerou o cenário atual de desenvolvimento de unidades de processamento. Assim como existem as redes de computadores cujas características são analisadas em macro escala existem redes internas nos chips (chamadas de Networks on Chip, NoCs), desenvolvidas para realizar a comunicação entre todos os componentes de hardware de um determinado SoC de forma eficiente em diversos aspectos como latência, consumo de energia e área de chip ocupada. Considerando estas estruturas o objetivo principal deste trabalho é apresentar os conceitos básicos relacionados às NoCs juntamente com uma análise de qual o papel desempenhado e as modificações propostas pelos Field Programmable Gate Arrays (FPGAs) para auxiliar o desenvolvimento desta importante subárea da computação de alto desempenho.

6.1. Introdução

Com a evolução tecnológica dos materiais que compõe os processadores - transistores, bem como a evolução dos demais circuitos, arquiteturas que utilizam processadores *multicore* são a base da computação atual e são o foco também da computação de alto desempenho. Ao se colocar mais de um processador em um mesmo chip cria-se um problema de comunicação entre os núcleos que não deve ser solucionado com as ferramentas de barramentos atuais por não atingirem de forma satisfatória os requisitos de sistema como *throughput*, consumo de energia, confiabilidade. Então qual seria a solução para este problema?

O conceito de comunicação em rede surgiu no passado com objetivo de conectar de uma forma mais otimizada sistemas que compunham redes *peer-to-peer*. A tarefa prin-

cipal dos projetistas de redes é solucionar múltiplas instâncias de um problema complexo de otimização que possui diversas soluções possíveis. No caso das NoCs este problema retorna em um contexto diferente, porém não menos complexo. Redes on-chip estão prevalecendo no domínio dos servidores de alto desempenho integrados à sistemas embarcados on-chip (SoC - System on-chip) (PEH; JERGER, 2009). Ao se projetar uma NoC novos desafios para problemas antigos como roteamento, qualidade de serviço, fluxo, controle de congestionamento e confiabilidade são encontrados e precisam ser solucionados. Além disso novos desafios são incorporados ao problema como consumo de energia e área de chip ocupada que contribuem ainda mais para o aumento da complexidade das NoCs.

Os multiprocessadores SoC permitem soluções com alto poder de processamento em aplicações embarcadas e utilizam uma implementação de memória distribuída compartilhada (DSM - Distribuited Shared Memory) em hardware. Sistemas SoC são compostos por diversos núcleos de processamento, memória e coprocessadores de propósitos específicos, e baseiam-se em uma arquitetura de memória NUMA (Non-Uniform Memory Access), permitindo acesso variado à memória através da rede de alto desempenho (MONCHIERO et al., 2007). Na arquitetura de rede dos SoCs, barramentos compartilhados dominam as estruturas de interconexões em sistemas simples, sendo esses barramentos bidirecionais permitindo o tráfego de informações pelas vias. Tal estrutura compartilhada facilita a adição de novos dispositivos, além de permitir portabilidade de periféricos entre os sistemas (ABDALLAH, 2017).

Conforme o número de núcleos conectados ao barramento da estrutura da rede aumenta, há um aumento proporcional na demanda de largura de banda a fim de facilitar a utilização de todos os núcleos (PEH; JERGER, 2009) e no clock geral do sistema, ocasionando atrasos e gerando gargalos importante no desempenho da aplicação. Além disso, essa abordagem tradicional de comunicação falha quando requisitos de escalabilidade e consumo de energia são levados em consideração para buscar uma evolução de futuros SoCs. Assim, novos esquemas de interconexão estão sendo propostos, denominados *Networks on Chip - NoC* (ABDALLAH, 2017), a fim de mitigar o gargalo na comunicação de sistemas on-chip (TSAI et al., 2012). Este texto está organizado com a análise de camadas proposta por Benini e De Micheli (2006): a seção 1.2 trata da camada física seguida pela camada de enlace na seção 1.3, redes e transporte na seção 1.4, aplicação e implementação na seção 1.5 e, por fim, uma análise da implementação em FPGAs na seção 1.6. O texto encerra-se com a conclusão e as referências bibliográficas.

6.2. A camada física

Gordon Moore foi responsável por definir um dos principais guias responsáveis por orientar a o ramo da arquitetura de computadores, conhecido como Lei de Moore. Moore previu que o número de transistores em um chip dobraria a cada 12 meses, embora esse fator tenha sido alterado para cada 18 meses, a Lei de Moore foi por muito tempo um dos principais fatores para promover o avanço na área de circuitos integrados. Estudo atuais sobre a miniaturização dos transistores e o efeito do tunelamento quântico demonstram que não se pode diminuir o dispositivo a partir de um certo tamanho sem prejudicar seu funcionamento. Sendo assim a lei de Moore que previa que o número de transistores seria dobrado não é mais verdade, e sim que os mesmos transistores e outros componen-

tes serão utilizados de uma maneira mais inteligente formando hardware mais eficiente (TRACK; FORBES; STRAWN, 2017). Assim, diferentes arquiteturas que exploram diversos modelos de interconexões da rede utilizando inúmeros transistores, que visam o baixo consumo energético e a máxima frequência são oferecidas a fim de de uma melhor eficiência do sistema on-chip (ABDALLAH, 2017).

Em sistemas on-chip (SoC), cujo conceito arquitetural tem sido desenvolvido nas últimas décadas, cada processador - ou processadores, juntamente com memórias e seus conjuntos de periféricos estão interconectados através de um barramento implementado em um único chip. Entretanto, com o aumento do número de transistores, o design está a cada dia mais complexo, o que demanda novas técnicas para resolver o gargalo de comunicação e que possibilite que um chip resolva aplicações complexas de forma a aproveitar todos os benefícios fornecidos pelas tecnologias atuais (CHEN et al., 2012).

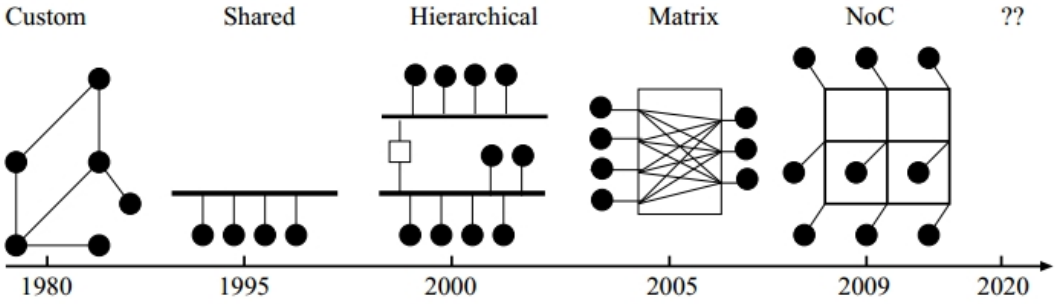

Figura 6.1. Linha do tempo da evolução da interconexão de sistemas on-chip. (ABDALLAH, 2017)

A interconexão é o principal elemento de sistemas multiprocessadores on-chip, uma vez que buscam fornecer baixa latência na camada de comunicação, a fim de minimizar o overhead gerado pelo sincronismo de processos, contrapondo a abordagem baseada em barramentos, o qual possui limitações físicas, garantindo alta largura de banda e paralelismo na comunicação devido à latência dos fios (Monchiero et al., 2006). Para tanto, sistemas on-chip precisam ser confeccionados cuidadosamente, pois podem consumir altos níveis de energia (PEH; JERGER, 2009). A Figura 6.1 exibe uma linha do tempo acerca das interconexões de sistemas on-chip, evidenciando que as arquiteturas de interconexão de redes que utilizam NoCs são as mais utilizadas nos últimos anos.

A estrutura básica de uma NoC é apresentada na Figura 6.2. Esta estrutura pode se alterar de acordo com a topologia escolhida, porém os componentes básicos são os mesmos. O *Switch* irá fazer a conexão entre os *Data Links* para transportar a informação dos Módulos de Processamento. As estruturas da camada física permitem que a camada de enlace possa executar suas operações.

6.3. Camada de enlace

No caso das NoCs a camada de enlace é responsável por garantir uma transferência confiável entre meios físicos de transmissão inerentemente não confiáveis (*data links*). Outra tarefa importante desta camada é regular o acesso a recursos compartilhados. Neste contexto existem dois problemas a serem tratados: a parte interna que precisa ser controlada

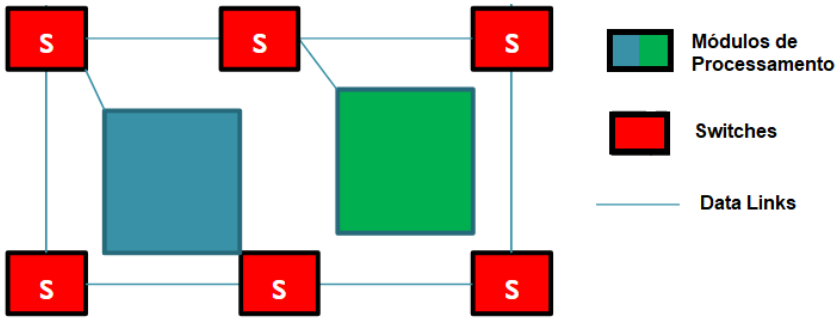

Figura 6.2. Estrutura básica de uma NoC genérica com foco em seus principais componentes.

da melhor forma possível com um escalonador, e a parte de interferências internas e externas (eletromagnéticas, ruídos térmicos, partículas alpha).

A disputa por recursos nas NoCs é praticamente inevitável. A existência de circuitos chamados árbitros permite que o pacote seja enviado e colida com outros para decidir de quem é o barramento. Como consequência a única maneira de se conseguir eliminar o árbitro é utilizar roteamento sem contenção que, caso seja utilizado, necessita que os pacotes sejam escalonados de forma a nunca utilizar o mesmo link ao mesmo tempo.

Os erros causados por interferências são normalmente resolvidos considerando a redundância de informações pela rede juntamente com técnicas de conferência de bits que são mais fáceis de serem implementadas em hardware.

6.4. Camadas de transporte e de rede

Apesar de todas as camadas desempenharem papeis importantes, as camadas de transporte e rede podem impactar significativamente o resultado de um projeto de NoC mesmo que as outras camadas estejam obtendo bom desempenho. O procedimento nesta etapa envolve o princípio de *switching* a ser utilizado (circuitos, pacotes), em seguida a topologia da rede deve ser escolhida e a conexão física realizada corretamente, por fim o esquema de endereçamento e roteamento devem ser definidos.

Neste momento o dispositivo onde a NoC será construída deve ser escolhido. Neste momento é importante ter em mente as vantagens e desvantagens de cada tipo de hardware sendo que o *chip* propriamente dito (ASIC) e os FPGAs são escolhas interessantes. A seção 1.5 irá tratar em mais detalhes as características de cada um dos dispositivos.

A questão da qualidade de serviço (QoS) de uma NoC é definida de diversas maneiras, porém três características são base para a definição: *data rate* que representa a taxa em que dados e instruções são processadas, a latência da rede, e a variação da latência (conhecida como *jitter*). Para garantir os requisitos de QoS é necessário definir com cuidado as partes da NoC, começando por sua topologia.

O principal foco quando se trata de arquiteturas multiprocessadas que utilizam plataformas SoC é a topologia das interconexões (PANDE et al., 2005) e quando o foco é a

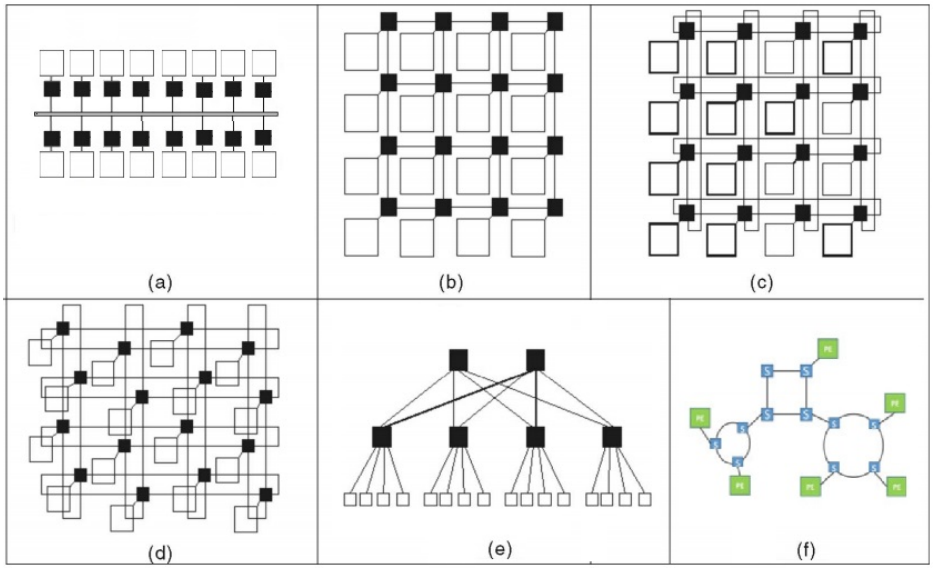

Figura 6.3. Topologias para NoCs. (a) Crossbar, (b) Mash, (c) Torus, (d) Folded Torus, (e) Tree, (f) Irregular(ABDALLAH, 2017)

perspectiva de comunicação, existem várias topologias que tangem os NoCs (AGARWAL; SHANKAR, 2009). O assunto sobre topologias de rede foi largamente explorado no contexto de computação paralela e distribuída. Neste trabalho serão discutidas as questões relevantes para NoCs. A diferença principal é que normalmente as questões de topologia em NoCs são concentradas em modelos 2D considerando o estado atual da fabricação de *chips*.

As topologias podem incluir redes de diversos tipos como apresentados na Figura 6.3. Tantas variedades buscam explorar diferentes topologias e implementações para plataformas SoCs que usufruem de NoCs (AGARWAL; SHANKAR, 2009). Os objetivos de escolha de uma determinada topologia envolvem requisitos de área de chip, desempenho e consumo de energia (BENINI; MICHELI, 2006).

O uso de malhas em 2D faz com que as NoCs sejam mais eficientes em termos de latência, consumo energético e facilidade na implementação (AGARWAL; SHANKAR, 2009). A topologia Crossbar (Figura 6.3(a)) apresenta todos os nós da rede interconectados, porém a escalabilidade é baixa. O Mesh (Figura 6.3(b)) é popular entre os projetos de NoCs (60% dos casos de topologias 2-D (PEH; JERGER, 2009)), a área cresce linearmente em relação ao tamanho do Mesh e o tipo de conexão acaba por concentrar um tráfego muito alto no centro. No caso do Torus (Figura 6.3(c)) a área ocupada e a dissipação de energia cresce linearmente com o número de núcleos e o desempenho é inversamente proporcional ao tamanho da estrutura pela demora na transmissão de linhas muito longas. Quando longos barramentos em linha existem, problemas de resistência e capacitância não podem ser ignorados.

Este problema pode ser resolvido com a topologia da Figura 6.3(d) que possui *bypasses* para otimizar a comunicação sendo o tipo mais utilizado em FPGAs. A topologia em árvore (Figura 6.3(e)) pode apresentar bons resultados em redes pequenas, porém

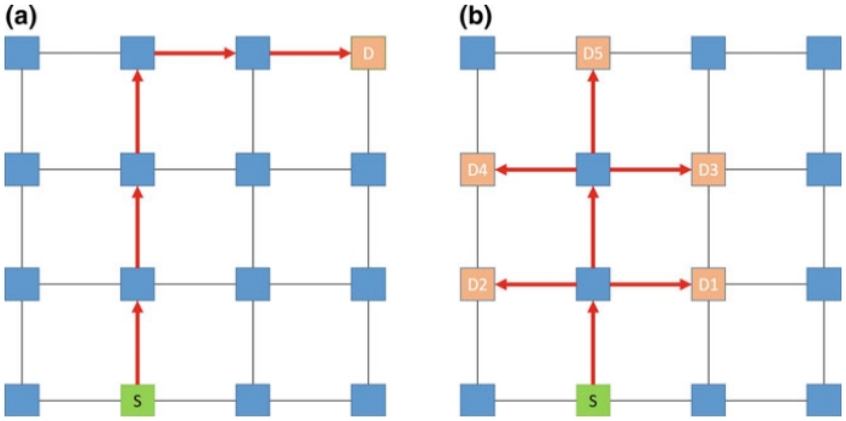

Figura 6.4. Categorização dos algoritmos de roteamento. a) *unicast*; b) *multicast* (ABDALLAH, 2017)

tende a ser pior que as duas anteriores em redes normais devido ao tempo de acesso por começar sempre na raiz e ter que percorrer a árvore. Topologias mistas podem ser empregadas no uso de aplicações específicas, uma vez que a o desenvolvimento pode exigir restrições rigorosas, e.g.: área e energia. Assim, as topologias regulares podem não ser a abordagem adequada, uma vez que topologias irregulares podem oferecer melhor flexibilidade (ABDALLAH, 2017). Um exemplo de topologia irregular pode ser visto na Figura 6.3(f).

As plataformas que utilizam NoC são baseadas em redes comutadas por pacotes, gerando novas e eficientes formas de roteamento. Segundo (AGARWAL; SHANKAR, 2009), os roteadores implementam várias funcionalidades, desde uma comutação simples até o roteamento inteligente, ou seja, suponha que um roteador baseado em topologia de malha possua quatro entradas e quatro saídas para outros roteadores e outras entradas e saídas para interfaces de rede (NI - Network Interface), dado esse cenário, o roteamento deve ser projetado baseado no uso do hardware. Para redes com comutação de circuitos, roteadores podem ser projetados utilizando filas (buffering), e para redes comutadas por pacotes, há a necessidade de se dispor de uma certa quantidade de buffers para suportar a transferência de dados intermitentes.

Os algoritmos de roteamento estão intrinsecamente ligados às topologias de rede. Tais algoritmos podem ser classificados, segundo (ABDALLAH, 2017), de acordo com vários critérios, sendo:

- **Número de destinações:** Conforme o número de nós de destino, os quais recebem os pacotes destinados a eles, os algoritmos de roteamento podem ser classificados em *unicast* e *multicast*. O roteamento *unicast* envia pacotes de um único nó de origem para um único nó destino. Já os *multicast* enviam pacotes de um único nó de origem para vários nós de destino. Além disso, o *multicast* pode ser dividido em roteamento baseado em árvore e baseado em caminho. A Figura 6.4 ilustra as duas categorias.
- **Decisão de localidade:** De acordo com as decisões de localidade, os algoritmos

de roteamento (sejam *multicast* ou *unicast*, podem ser classificados em roteamento de origem ou roteamento distribuído. No roteamento distribuído, haverá um cabeçalho - roteamento *unicast* - contendo o endereço do nó de destino. Com isso, as informações são calculadas *in place* toda vez que o cabeçalho entra em um nó do comutador. Já no roteamento de origem, os caminhos são calculados no nó de origem.

- **Adaptativo:** Em todas as implementações de roteamento, o algoritmo pode ser determinístico ou adaptativo. No roteamento determinístico, os caminhos gerados entre fonte e destino são calculados estaticamente e sempre serão parecidos. Já o algoritmo adaptativo pode desviar de regiões não favoráveis ou defeituosas na rede. Além disso, podem ser divididos em totalmente adaptativos ou parcialmente adaptativos.

- **Minimalidade:** Algoritmos que podem ser classificados em minimais ou não-minimais. Em linhas gerais, um algoritmo minimal não permite rotas afastadas do nó destino, garantindo sempre o caminho mínimo. O não-minimal permite desvios, acarretando em rotas distantes do nó de destino, que podem existir de formas aleatórias ou seguindo certas regras.

Além disso, em Agarwal e Shankar (2009) pode ser visto uma classificação dos algoritmos de roteamento baseados no número de nós de destino, sendo *unicast* e *multicast*. Tal classificação, além dos já citados acima, levam em consideração o número de caminhos gerados, o quão progressivo é o algoritmo e a forma de implementação, que podem utilizar tabelas de consulta ou máquinas de estados. As implementações que utilizam tabela de consulta são as mais populares, uma vez que são implementados em *software*, onde armazenam todos os nós em uma tabela. Já as implementações baseadas em máquinas de estados podem ser implementadas tanto em *software* quanto em *hardware*.

Na prática o modelo de NoC segue o padrão semelhante a um grid. Os módulos de interfaces de rede transformam os pacotes de dados gerados dos clientes - processadores - em sinais de controle de fluxo de comprimento fixo, denominados *flits*. Os *flits* associados ao pacote de dados consiste em fornecer um cabeçalho, uma saída e um identificador. Esse conjunto será roteado em direto ao nó destino, passando de um roteador ao outro. Neste modelo de NoC, cada roteador possui cinco portas de entradas e cinco portas de saídas, que são conectadas aos vizinhos, formando o grid. Assim, a função do roteador é rotear cada *flit* de uma porta de entrada à uma porta de saída apropriada. A fim de performar essa tarefa, os roteadores são preparados com buffers em cada porta, *switches* para redirecionar o fluxo e lógica de controle para garantir a corretude da execução (TSAI et al., 2012), conforme mostrado na Figura 6.5.

Protocolos podem requisitar diversas classes de mensagens, consequentemente, diversas classes de coerência devem existir a fim de retornar com a ação correta. Esse tráfego, quando não consistente, podem surpreender o algoritmo de roteamento e acarretar em *deadlocks*. Portanto, garantir que a rede esteja acessível em nível de protocolo é fundamental para que não ocorra impedimentos (*deadlocks*).Peh e Jerger (2009) ilustram essa situação na Figura 6.6, na qual a rede é exposta a inúmeras solicitações que não podem ser consumidas até a interface de rede iniciar a resposta, caracterizando uma

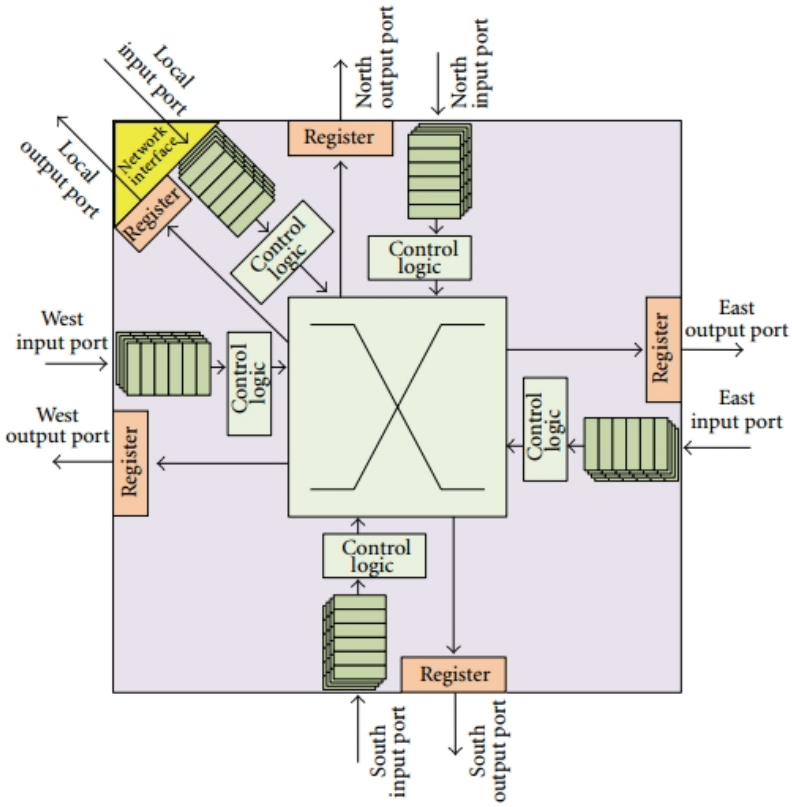

Figura 6.5. Arquitetura típica de roteador em NoC. (TSAI et al., 2012)

dependência cíclica. Assim, se duas unidades funcionais gerarem solicitações ao ponto de sobrecarregar a rede, essas unidades ficaram ociosas aguardando respostas. Se tais respostas utilizarem os mesmos recursos da rede que as solicitações, não haverá avanço no processo, caracterizando um *deadlock*. Portanto, a implementação de SoCs demandam grupos específicos de protocolos baseados diretamente na aplicação que o sistema comportará (TSAI et al., 2012).

6.5. Camada de aplicação e implementação

Sistemas que demandam poder computacional elevado se beneficiam de dois principais tipos de sistemas: multiprocessadores de chip de memória compartilhada e SoCs de multiprocessadores (MPSoCs). Com esses sistemas, a programação paralela se tornou amplamente utilizável e importante em inúmeras aplicações, tal crescimento se deve ao uso de um espaço de endereçamento global compartilhado, facilitando a implementação de aplicações que dependem de programação paralela. Assim, com o modelo de memória compartilhada, a comunicação ocorre de forma implícita através de instruções de *load* e *store* (PEH; JERGER, 2009). A arquitetura de um multiprocessador de chip de memória compartilhada pode ser visto na Figura 6.7

O modelo de memória compartilhada oferece uma forma intuitiva de realizar o compartilhamento. O espaço global de memória compartilhada é baseado na aplicação

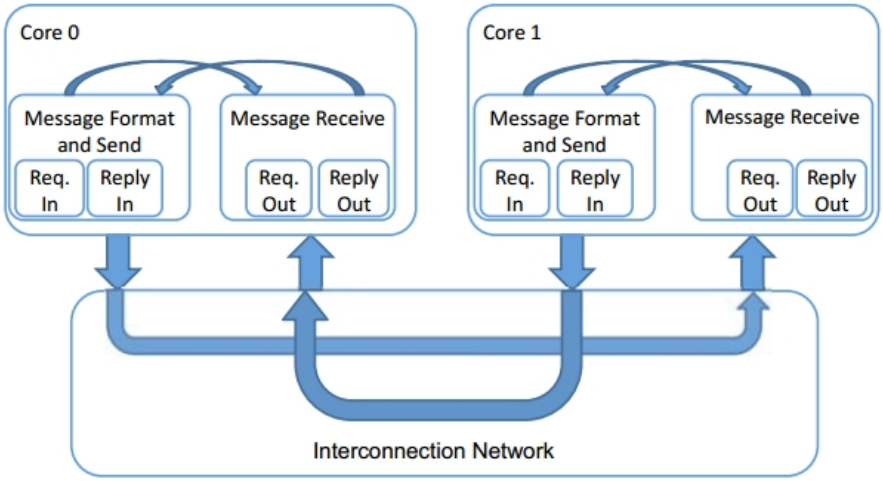

Figura 6.6. *Deadlock* a nível de protocolo. (PEH; JERGER, 2009)

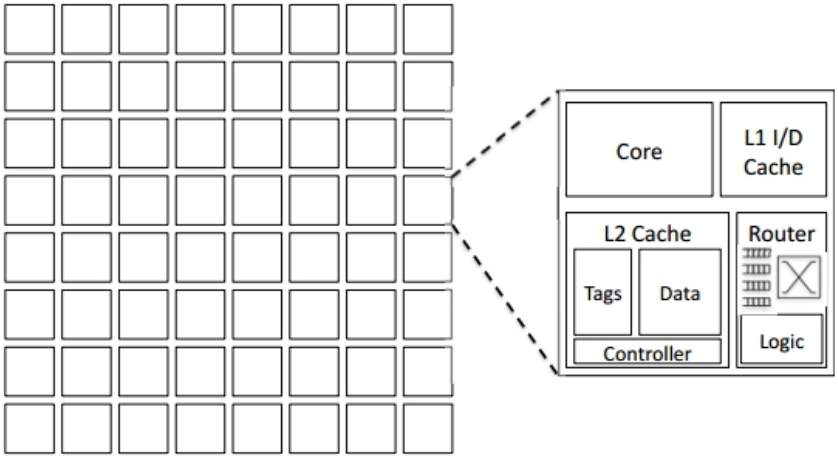

Figura 6.7. Arquitetura de chip de memória compartilhada. (PEH; JERGER, 2009)

que o sistema deve suportar, oferecendo acesso aos endereços de memória local e remota de maneira uniforme (Vasava; Rathod, 2015). Sendo que o acesso remoto ocorre através de uma rede interconectada que permite a troca de dados entre os processadores (RAUBER; RüNGER, 2010). Assim, sistemas que utilizam o espaço de memória compartilhado mantém várias cópias de dados através dos nós a fim de garantir a leitura desses dados, mesmo quando isso não é possível (Xu et al., 2017).

Para garantir a correta execução da aplicação, bem como garantir a consistência dos dados em memória, é necessário que protocolos de semântica e coerência de cache sejam aplicados. Sistemas que utilizam multiprocessadores utilizam um ou dois diferentes tipos de protocolos para garantir a coerência, cada um resultando em diferentes características de tráfego na rede interconectada (RAUBER; RüNGER, 2010). Protocolos de coerência de cache necessitam de diversos tipos de mensagens, como: *unicast*, *multicast*

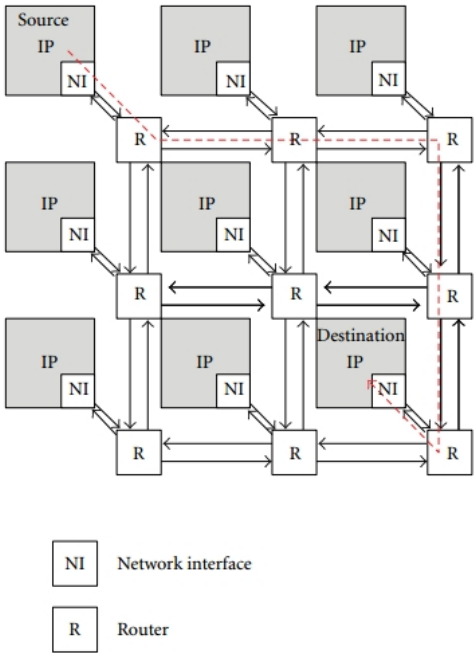

Figura 6.8. Arquitetura típica de NoC. (TSAI et al., 2012)

e *broadcast*. *Unicast* é um tráfico que possui uma única fonte e um único destino, e.g: nível L2 ao controlador de memória). *Multicast* possui um tráfego de uma única fonte para múltiplos destinos. Por fim, *Broadcast* envia uma mensagem de uma única fonte para toda rede interconectada (PEH; JERGER, 2009).

Não só os protocolos dependem intimamente da aplicação que um sistema SoC comportará, como também sua arquitetura e organização. Uma típica arquitetura de um SoC baseado em NoC consiste em múltiplos segmentos de fios e roteadores, como mostrado na Figura 6.8.

As características apresentadas sobre NoCs são aplicáveis independentemente dos dispositivos utilizados para sua implementação. Entretanto em cada caso é necessário considerar a melhor forma de aplicar a teoria para que o resultado na prática seja o melhor possível. Um dos dispositivos mais utilizados para o teste e implementação de NoCs são os FPGAs e, ao desenvolver redes para estes dispositivos, algumas práticas específicas são adotadas. A próxima seção apresenta os FPGAs e seu papel no desenvolvimento das NoCs.

6.6. NoCs em dispositivos reconfiguráveis

Os *Field Programmable Gate Arrays* (FPGAs) são ferramentas de desenvolvimento de hardware amplamente utilizadas para prototipação de sistemas permitindo sua descrição em diferentes níveis de abstração. O FPGA pode ser reconfigurado um número suficiente de vezes para justificar sua utilização como dispositivo de prototipação. Assim como a evolução do hardware que levou aos dispositivos *multicore*, os FPGAs também apresentam diferentes opções de plataformas e ferramentas de desenvolvimento com foco no

desenvolvimento de co-projetos de hardware/software.

Dentre as diversas configurações possíveis para o hardware de um FPGA a mais comum seria a organização hierárquica de blocos de células lógicas interconectadas por barramentos e, na camada mais externa, blocos responsáveis por operações de E/S de dados (BOBDA, 2008). A ideia básica do funcionamento de FPGA é a conversão do hardware descrito em funções lógicas básicas que podem ser implementadas por *Look-Up Tables*. Após esta conversão as funções podem ser armazenadas em elementos lógicos básicos que, em conjunto, formam os blocos de células lógicas programáveis do dispositivo. Os barramentos existentes promovem a conexão dos elementos lógicos que recebem informação dos bolocos de E/S (HAUCK; DEHON, 2008).

Porém, antes de analisar uma figura com a disposição desta arquitetura é conveniente a expansão deste conceito para arquiteturas modernas de FPGAs. A evolução do conceito de desenvolvimento de hardware para os co-projetos de hardware/software mostrou a necessidade de integração entre as soluções de hardware criadas e processadores que executavam os componentes de software (BERGER, 2001).

O movimento inicial nos FPGAs com relação a inclusão de processadores que poderiam ser utilizados juntamente com o hardware desenvolvido foram os *soft-processors* como os processadores Nios II da Intel®e MicroBlaze da Xilinx®. Estes processadores são totalmente descritos e configurados utilizando os componentes lógicos do FPGA e podem se comunicar com qualquer bloco desenvolvido utilizando a própria estrutura do FPGA. O baixo desempenho destas soluções para um grande número de aplicações ressaltou a questão de que comunicar os blocos com um processador é um requisito crítico do co-projeto e a solução foi a criação dos *Systems on a Chip* (SoCs).

Os SoCs apresentam um sistema completo (diversos componentes como processador, memória, lógica reconfigurável) implementado em um mesmo chip. Esta é uma solução que se tornou uma alternativa amplamente utilizada para diminuir os custos de comunicação de co-projetos de hardware/software (WOLF, 2016). No caso dos FPGAs foram desenvolvidos SoCs que contém toda a parte do desenvolvimento da lógica reconfigurável e a conexão com processadores (geralmente ARM®e RiscV®) feita totalmente dentro do chip. O tempo de comunicação neste caso diminui consideravelmente o que impacta diretamente na qualidade da solução final.

A Figura 6.9 apresenta um modelo de arquitetura moderna de FPGAs simplificado indicando a conexão entre um *hard-core processor*, células de memória e a parte responsável pela lógica reconfigurável. Os fabricantes possuem seus próprios modelos de arquitetura, porém de forma básica o que pode ser encontrado é muito similar ao apresentado na Figura 6.9. Um dos FPGAs mais modernos disponíveis no mercado, o Stratix X da fabricante Intel®, possui um bloco com um ARM Cortex®quad-core que pode chegar a 1.5Ghz de clock e um bloco de memória on-chip de 256MB de RAM[1].

Considerando a situação apresentada é possível analisar como os FPGAs são utilizados em relação ao desenvolvimento, criação, verificação, teste e utilzação de NoCs. A implementação de NoCs utilizando FPGAs possui alguns componentes básicos (de Lima et al., 2016):

[1]<https://www.intel.com.br/content/www/br/pt/products/programmable/soc/stratix-10.html>

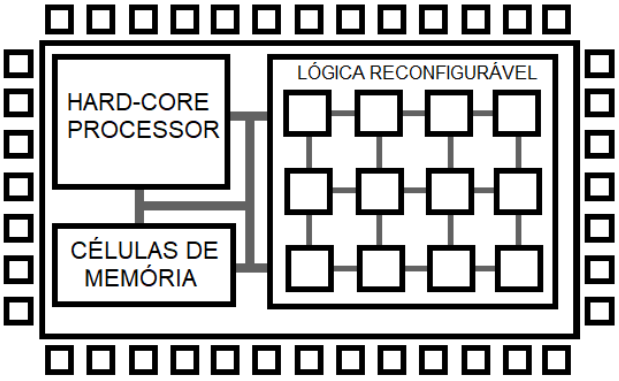

Figura 6.9. Arquitetura de SoCs com FPGAs.

- Host - parte do sistema responsável por receber os cenários de avaliação e enviar os resultados da avaliação.
- Cenários de Avaliação - são as configurações da NoC que está sendo implementada para avaliação. Deve considerar os parâmetros da NoC a serem avaliados e configurar o sistema.
- Resultados da Avaliação - a rede irá receber uma carga de trabalho, direcionar os dados e fazer o envio dos dados solicitados. A partir do comportamento da rede o host avalia todas as métricas desejadas. O relatório final com toda a avaliação deve ficar disponível no host para consulta.
- Geradores de Tráfego - esta parte é responsável por gerar o fluxo de dados que será tratado pela NoC em determinado cenário.
- Receptores de Tráfego - irão receber da NoC os dados após seu processamento e verificar todas as informações necessárias para o cálculo das métricas.
- NoC - a rede propriamente dita com os roteadores conectados e toda a estrutura necessária para seu funcionamento configurada pelos parâmetros descritos pelos Cenários de Avaliação.

A diferença entre os sistemas apresentados está normalmente em como estes componentes são desenvolvidos e onde estão implementados (de Lima et al., 2016). Uma das formas de implementar todos os componentes do sistema no FPGA, sendo que o problema é que devido ao número de elementos lógicos necessários para implementar todo o sistema a NoC que pode ser avaliada acaba por ter um tamanho reduzido. Nestes casos é possível implantar a NoC utilizando um conjunto de FPGAs permitindo que a estrutura avaliada seja maior e mais complexa.

Caso não seja possível implementar todo o sistema em hardware de uma forma funcional é possível utilizar o hardware do FPGA para acelerar o tempo da simulação da rede. Esta aceleração é possível pois implementa-se apenas o mínimo necessário para a

NoC executar o cenário, porém esta técnica não é indicada para testes de novas estruturas devido às simplificações efetuadas na implementação. Ainda assim o hardware do FPGA pode ser um limitante na execução. A saída neste caso é tentar implementar todos os roteadores da rede em um só, multiplexando informações. O tempo de simulação é maior, porém a limitação do hardware é solucionada.

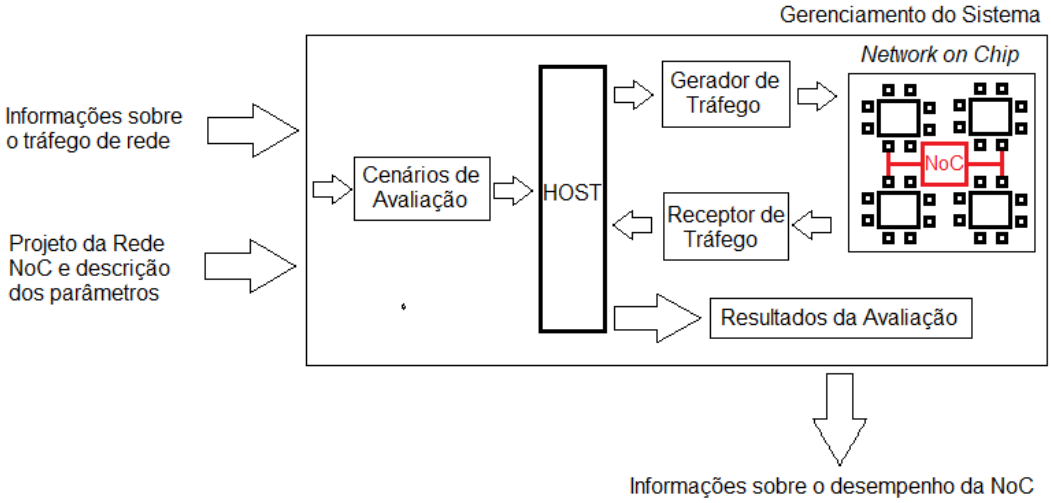

Figura 6.10. Modelo completo de implementação de NoCs utilizando hardware reconfigurável.

A camada de geração de tráfego na rede descreve uma distribuição espaço-temporal do tráfego de informação a ser trabalhado pela NoC. Duas abordagens são utilizadas neste caso: o tráfego sintético que tem por objetivo estressar a rede para descobrir seus problemas estruturais e de projeto de uma forma geral; e o tráfego orientado a aplicação queé importante pois irá utilizar o tráfego real das aplicações para as quais a NoC foi desenvolvida. A geração de tráfego de aplicativos reais para a NoC a ser testada pode seguir três técnicas basicamente (de Lima et al., 2016):

- *Trace-based* - inicialmente a aplicação é executada para se obter os rastros da comunicação realizada pela rede. Os resultados são utilizados para criar cenários baseados nos pontos mais críticos e de forma escalar. Este procedimento gera situações complexas de teste para a NoC. O tamanho reduzido da memória RAM nos FPGAs é o principal problema da técnica limitando os cenários de teste.
- *Stochastic* - análises estatísticas são feitas no comportamento da aplicação e então o tráfego é gerado para o teste da NoC de acordo com os resultados desta análise.
- *Application Cores* - este modelo gera o tráfego de acordo com a execução real dos núcleos das aplicações que irão utilizar a NoC projetada.

A Figura 6.10 representa todos os componentes do processo e como se comunicam. Toda esta estrutura é gerenciada por um sistema que está em uma camada acima do que foi descrito até o momento. No caso dos FPGAs os softwares dos fabricantes

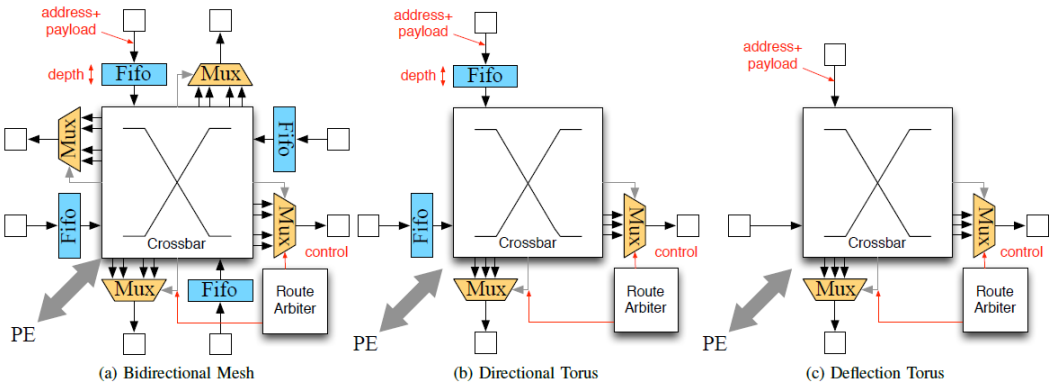

Figura 6.11. Imagens das possíveis arquiteturas de switches para NoCs em FPGAs (Kapre; Gray, 2015).

proveem uma plataforma que soluciona grande parte dos problemas encontrados já que implementam toda a interface de teste segundo padrão IEEE JTAG (IEEE, 2012).

O desenvolvimento da área nos últimos anos permitiu que a abordagem dos trabalhos recentes na área tivesse foco principalmente na otimização de cada um dos componentes apresentados no modelo da Figura 6.10 e no projeto específico para aplicações. A seguir são analisados alguns dos trabalhos recentes da área de NoCs em hardware reconfigurável e seus resultados.

O início da análise será um artigo de 2015 que apresentou uma NoC que seria a base de diversos trabalhos apresentados em conferências a Hoplite (Kapre; Gray, 2015). O trabalho apresenta inicialmente um gráfico comparativo que atesta na prática que a rede Hoplite que possui a topologia *deflection torus* e foi desenvolvida em nível RTL apresentava a melhor relação *Throughput x Utilização de LUTs (Area)* em comparação com a mesma topologia gerada automaticamente, com a Mesh e com a Torus.

A principal contribuição na arquitetura da rede foi a mudança da arquitetura dos *switches* da rede como mostra a Figura 6.11. Dentre as possibilidades, o switch proposto (letra (c) da Figura 6.11) elimina movimentos multidirecionais ao eliminar todas as estruturas FIFO e metade dos multiplexadores em comparação com a implementação mais completa (letra (a) da Figura 6.11). Quando os resultados do artigo foram apresentados a rede apresentava uma área de chip 3.5x menor do que a da melhor rede até o momento com o dobro de frequência máxima de clock e 2.5% de vantagem sobre o throughput da rede 2D bidirecional. Estes resultados expressivos fizeram com que esta rede fosse considerada o estado-da-arte em NoC para FPGAs.

Consumo de energia é uma das características mais importantes atualmente para os SoCs, devido a suas aplicações em sistemas críticos e dispositivos móveis. Uma comparação entre uma NoC implementada em um FPGA (chamado de *soft* pelos autores) e uma NoC implementada em um ASIC (chamado de *hard* pelos autores) é apresentada por Abdelfattah e Betz (2014). Os resultados demonstram que o consumo de energia de uma NoC em um ASIC é, em média, quase a metade do consumo total em comparação com a rede configurada em um FPGA. Porém o gráfico que mostra os resutlados apresenta

um consumo extremamente baixo FPGA até 3 nós principais da rede sugerindo que para redes pequenas os FPGAs são indicados. Além disso, existe a flexibilidade da estrutura que indiscutivelmente é maior quando se utiliza lógica reconfigurável (ABDELFATTAH; BETZ, 2014).

Algumas tecnologias presentes em dispositivos específicos podem auxiliar no desempenho das NoCs. O funcionamento das NoCs desenvolvidas para FPGAs é usualmente baseado em memória, sugerindo que caso a estrutura de memória possa ser otimizada o resultado da rede também será. A técnica de organização de memória RAM em cascata implementada pela fabricante Xilinx, que apresenta uma forma otimizada de conectar a memória melhorando o tempo de acesso, foi utilizada e os resultados apresentados melhoraram o *troughput* da rede em 40% e o consumo de energia em 10% no pior caso (KAPRE, 2017).

O foco de implantação de NoC em sistemas de tempo-real foi demonstrado inicialmente com a modificação da rede Hoplite proposta anteriormente por uma versão RT (WASLY; PELLIZZONI; KAPRE, 2017) onde foram modificados a função de roteamento que passou a priorizar deflecções e a taxa de vazão dos pacotes. Em seguida um algoritmo de roteamento baseado em prioridade foi implementado para otimizar o sistema o que ocasiou a mudança para uma topologia de anel direcional com *bypasses* (RIBOT; NELISSEN, 2019). Outra modificação para melhorar o desempenho da rede Hoplite foi o Hoplitebuf (GARG et al., 2019) que adicionou estruturas de *buffers* FIFO em cada roteador para diminuir a deflexão de pacotes e ordená-los no próprio roteador chegando a atingir uma taxa de até 50% de viabilidade com o aumento de até 20% na injeção de pacotes.

Ringnet (Siast; Łuczak; Domański, 2019), uma alternativa para o design da Hopnet, apresenta uma comunicação totalmente baseada em memória o que, segundo os autores, torna a implementação mais voltada para FPGAs. Outros recursos implementados foram o controle de injeção de pacotes exclusivamente pelos elementos lógicos e memória distribuída em pequenos *buffers*. Apesar dos resultados do artigo serem classificados como bons pelos autores nenhuma comparação com a Hopnet em nenhuma de suas versões foi apresentada diretamente.

O artigo mais recente encontrado sobre a questão é sobre a HopliteBuf (GARG et al., 2020) no que parece ser uma continuação do trabalho anterior (GARG et al., 2019). Neste artigo são apresentados com mais profundidade os conceitos incorporados à implementação inicial da NoC Hoplite e o resultado de ocupação de 30 a 40% menor de LUTs do FPGA utilizado no trabalho.

OS trabalhos relacionados analisados indicam que a direção principal do desenvolvimento de NoCs para FPGAs tem como base a implementação da NoC Hoplite e modificações que visam solucionar os problemas da estrutura para aplicações em sistemas críticos de tempo-real.

6.7. Conclusão

As NoCs são estruturas já consolidadas nos projetos de SoCs e são uma área de pesquisa ampla em constante desenvolvimento dentro do contexto da computação de alto desempe-

nho. O início da área que conhecemos hoje pode ser considerado o início dos anos 2000 com a publicação de materiais completos sobre o assunto (BENINI; MICHELI, 2006) e a primeira edição da principal conferência sobre o assunto o IEEE/ACM International Symposium on Networks on Chip (NOCS) em 2007.

Diversas tecnologias de projeto, implantação, verificação e fabricação estão sendo utilizadas e testadas com objetivo de encontrar as melhores soluções possíveis para um problema extremamente complexo. Este trabalho apresentou os conceitos básicos relacionados às NoCs de uma maneira geral, em seguida uma visão do mesmo contexto e foco em computação reconfigurável e, por fim, uma análise superficial do estado-da-arte do desenvolvimento de NoCs em FPGAs.

Para um panorama mais amplo do estado-da-arte da área existem conferências especializadas no assunto como, por exemplo, o IEEE/ACM International Symposium on Networks on Chip (NOCS), o International Forum on MPSoC for Software-Defined Hardware (MPSoC), e também as conferências de hardware reconfigurável como International Conference on Field-Programmable Logic and Applications (FPL) e o ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA). Revistas (*journals*) com foco em desenvolvimento de hardware, circuitos e sistemas também apresentam artigos interessantes sobre o assunto.

O futuro da utilização de NoCs em computação reconfigurável caminha para a disponibilização de redes já implementadas nas arquiteturas dos chips dos FPGAs. As principais empresas do ramo já perceberam que o caminho são os chips *multicore* híbridos com *hard-processors* conectados a blocos de lógica programável e dispositivos de memória. Então nada mais natural que o próximo passo seja a criação em hardware das interconexões necessárias para a implantação de NoCs e que as ferramentas de design permitam apenas a configuração destas estruturas sem a necessidade do desenvolvimento das mesmas *from scratch*.

Referências

ABDALLAH, A. B. *Advanced Multicore Systems-On-Chip: Architecture, On-Chip Network, Design*. [S.l.: s.n.], 2017. ISBN 978-981-10-6091-5. páginas 2, 3, 5, 6

ABDELFATTAH, M. S.; BETZ, V. Energy-efficient embedded nocs on fpgas. In: . [S.l.: s.n.], 2014. páginas 15

AGARWAL, A.; SHANKAR, R. Survey of network on chip (noc) architectures and contributions. *Journal of Engineering, Computing and Architecture*, v. 3, 01 2009. páginas 5, 6

BENINI, L.; MICHELI, G. D. *Networks on Chips. Technology and Tools*. [S.l.]: Morgan Kaufmann, 2006. ISBN 978-0-12-370521-1. páginas 5, 16

BERGER, A. S. *Embedded Systems Design: An Introduction to Processes, Tools and Techniques*. 1st. ed. [S.l.]: CMP Books, 2001. páginas 11

BOBDA, C. *Introduction to Reconfigurable Computing: Architectures, Algorithms, and Applications*. 1st. ed. [S.l.]: Springer Netherlands, 2008. páginas 11

CHEN, S.-J. et al. Reconfigurable networks-on-chip. 01 2012. páginas 3

de Lima, O. A. et al. A survey of noc evaluation platforms on fpgas. In: *2016 International Conference on Field-Programmable Technology (FPT)*. [S.l.: s.n.], 2016. p. 221–224. páginas 11, 12, 13

GARG, T. et al. Hoplitebuf: Fpga nocs with provably stall-free fifos. In: *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. New York, NY, USA: Association for Computing Machinery, 2019. (FPGA '19), p. 222–231. páginas 15

GARG, T. et al. Hoplitebuf: Network calculus-based design of fpga nocs with provably stall-free fifos. *ACM Trans. Reconfigurable Technol. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 13, n. 2, 2020. páginas 15

HAUCK, S.; DEHON, A. *Reconfigurable computing: the theory and practice of FPGA-based computation*. 1st. ed. [S.l.]: Morgan Kaufmann, 2008. (Systems on Silicon). páginas 11

IEEE. *Std. 1149.1 - Standard Test Access Port and Boundary-Scan Architecture*. [S.l.]: Official IEEE Std. 1149.1 Standard Working Group, 2012. páginas 14

KAPRE, N. Implementing fpga overlay nocs using the xilinx ultrascale memory cascades. In: *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. [S.l.: s.n.], 2017. p. 40–47. páginas 15

Kapre, N.; Gray, J. Hoplite: Building austere overlay nocs for fpgas. In: *2015 25th International Conference on Field Programmable Logic and Applications (FPL)*. [S.l.: s.n.], 2015. p. 1–8. páginas 14

Monchiero, M. et al. Exploration of distributed shared memory architectures for noc-based multiprocessors. In: *2006 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*. [S.l.: s.n.], 2006. p. 144–151. ISSN null. páginas 3

MONCHIERO, M. et al. Exploration of distributed shared memory architectures for noc-based multiprocessors. *Journal of Systems Architecture*, v. 53, n. 10, p. 719 – 732, 2007. páginas 2

PANDE, P. P. et al. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *IEEE Trans. Comput.*, IEEE Computer Society, USA, v. 54, n. 8, p. 1025–1040, ago. 2005. ISSN 0018-9340. Disponível em: <https://doi.org/10.1109/TC.2005.134>. páginas 4

PEH, L.-S.; JERGER, N. E. *On-Chip Networks*. 1st. ed. [S.l.]: Morgan and Claypool Publishers, 2009. ISBN 1598295845. páginas 2, 3, 5, 8, 9, 10

RAUBER, T.; RüNGER, G. *Parallel Programming - for Multicore and Cluster Systems.* [S.l.: s.n.], 2010. ISBN 978-3-642-04817-3. páginas 9

RIBOT, Y.; NELISSEN, G. Design and implementation of an fpga-based noc for real time systems. In: *CISTER Research Centre Conference Paper*. [S.l.: s.n.], 2019. p. 1–3. páginas 15

Siast, J.; Łuczak, A.; Domański, M. Ringnet: A memory-oriented network-on-chip designed for fpga. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, v. 27, p. 1284–1297, 2019. páginas 15

TRACK, E.; FORBES, N.; STRAWN, G. The end of moore's law. *Computing in Science Engineering*, v. 19, n. 2, p. 4–6, 2017. páginas 3

TSAI, W.-C. et al. Networks on chips: Structure and design methodologies. *J. Electrical and Computer Engineering*, v. 2012, 01 2012. páginas 2, 7, 8, 10

Vasava, H. D.; Rathod, J. M. Software based distributed shared memory (dsm) model using shared variables between multiprocessors. In: *2015 International Conference on Communications and Signal Processing (ICCSP)*. [S.l.: s.n.], 2015. p. 1431–1435. páginas 9

WASLY, S.; PELLIZZONI, R.; KAPRE, N. Hoplitert: An efficient fpga noc for real-time applications. In: *2017 International Conference on Field Programmable Technology (ICFPT)*. [S.l.: s.n.], 2017. p. 64–71. páginas 15

WOLF, M. *Computers as Components: Principles of Embedded Computing System Design*. 5th. ed. [S.l.]: Morgan Kaufmann Publishers, 2016. páginas 11

Xu, P. et al. The research of distributed shared memory technology in power system. In: *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. [S.l.: s.n.], 2017. p. 1309–1313. ISSN null. páginas 9