XIRU: Interface de Rede Extensível para Integração de Núcleos a uma Rede-em-Chip

Douglas Rossi de Melo[1]

Michelle Silva Wangham[1]

Cesar Albenes Zeferino[1]

Resumo: Para a integração dos núcleos de um sistema integrado a uma NoC (*Network-on-Chip*), é necessário o uso de interfaces de comunicação que realizem a adaptação do protocolo dos núcleos ao da rede e que ofereçam os serviços de comunicação necessários aos núcleos. Este artigo descreve a arquitetura de uma interface de rede extensível para a integração de núcleos à NoC SoCIN (*System-on-Chip Interconnection Network*). A interface proposta utiliza uma arquitetura estruturada em três camadas que realizam a adaptação do protocolo, o empacotamento/desempacotamento de dados e o envio/recepção de pacotes, entre outros serviços. O artigo descreve a arquitetura da interface de rede e apresenta os resultados de sua validação e síntese em silício.

Abstract: For the integration of the cores of a SoC (System-on-Chip) to a NoC (Network-on-Chip), it is necessary the using of communication interfaces that perform the adaptation of the core protocol to the network protocol and provide the necessary communication services to the cores. This paper presents a research done in order to provide an extensible network interface for the integration of cores to SoCIN (System-on-Chip Interconnection Network) NoC. The proposed interface was designed using a layered architecture that performs protocol adaptation, data packetization/depacketization and sending/receiving of packets, among others services. The paper describes the architecture of the network interface and presents results of its validation and synthesis in silicon.

[1] Laboratório de Sistemas Embarcados e Distribuídos, UNIVALI
Rua Uruguai, 458 – Centro – CEP: 88302-202 – Itajaí, SC, Brasil
{drm, wangham, zeferino @univali.br}

1 Introdução

A evolução dos processos de fabricação de circuitos integrados têm permitido o aumento do nível de integração de componentes em silício viabilizado a criação de sistemas computacionais complexos em um único chip. Tais sistemas são conhecidos como sistemas integrados ou *Systems-on-Chip* (SoCs). Para atender a requisitos quanto ao tempo de projeto, esses sistemas são implementados com o uso de blocos de hardware pré-projetados e pré-verificados, os quais são denominados núcleos ou *cores* [1].

A interconexão dos núcleos de um sistema integrado é geralmente realizada via barramentos compartilhados, com o uso de estruturas centralizadas ou hierárquicas. No entanto, essa abordagem não atende aos requisitos de SoCs que integram de várias dezenas a centenas de núcleos em um único chip, pois o desempenho do barramento é limitado e não escala com tamanho do sistema. As Redes-em-Chip ou *Networks-on-Chip* (NoCs) [2] foram propostas como alternativa arquitetural para solucionar esse problema, pois são reutilizáveis como o barramento e oferecem desempenho escalável e paralelismo em comunicação.

Uma NoC é constituída de um conjunto de roteadores interconectados de forma estruturada por meio de canais ponto-a-ponto [3]. Cada roteador possui um conjunto de portas que são utilizadas para conectar-se com seus roteadores vizinhos e com o núcleo de processamento. As NoCs utilizam o modelo de comunicação de troca de mensagens e a comunicação entre os núcleos ocorre por meio do envio e recebimento de pacotes. Um exemplo de NoC é a SoCIN (*SoC Interconnection Network*) [3], uma Rede-em-Chip de baixo custo e com desempenho escalável baseada em um roteador parametrizável (denominado ParIS – *Parameterizable Interconnect Switch*) que permite a síntese de redes com diferentes características de custo e desempenho.

Além dos roteadores e dos enlaces, uma NoC requer interfaces de rede (ou NIs – *Network Interfaces*) para adaptar o seu protocolo aos protocolos utilizados pelos núcleos e também para prover serviços de comunicação necessários ao sistema. Em geral, as interfaces oferecem serviços relacionados às camadas inferiores do modelo de referência OSI (*Open Systems Interconnection*), porém, serviços de camadas superiores também são requisitados.

Este artigo descreve uma arquitetura de interface de rede para uso em sistemas baseados na rede SoCIN. A arquitetura proposta visa viabilizar a integração de núcleos compatíveis com diferentes protocolos de comunicação intrachip à NoC. A interface de rede utiliza uma arquitetura em camadas que facilita a construção de interfaces de adaptação e a inclusão de novos serviços de comunicação.

As seções a seguir apresentam alguns conceitos base sobre o contexto deste trabalho (Seção 2), uma análise dos trabalhos relacionados (Seção 3), a arquitetura de rede proposta (Seção 4), os resultados experimentais relativos à implementação da interface de rede (Seção 5) e uma discussão sobre a estrutura e a capacidade de adição de novos serviços à interface (Seção 6). Por fim, na Seção 7, são apresentadas as conclusões do trabalho.

2 Redes-em-Chip

2.1 Arquitetura de uma Rede-em-Chip

Conforme ilustrado na Figura 1, uma NoC é constituída de interfaces de rede, roteadores e enlaces. A interface de rede realiza a comunicação entre o núcleo e a rede de forma transparente, adaptando sinais de sincronismo e de dados [4]. O roteador tem como função encaminhar mensagens transferidas pela rede e é composto de um conjunto de *buffers* FIFO (*First-In, First-Out*), multiplexadores e controladores que implementam os mecanismos de comunicação necessários à transferência de mensagens pela rede [5]. Também possui portas de entrada e saída, para comunicação com outros roteadores e com um núcleo conectado à rede. Os enlaces são geralmente formados por dois canais ponto-a-ponto unidirecionais em oposição, síncronos ou assíncronos, sendo que cada canal é constituído de fios responsáveis pelo transporte de mensagens, enquadramento e regulação de tráfego [5].

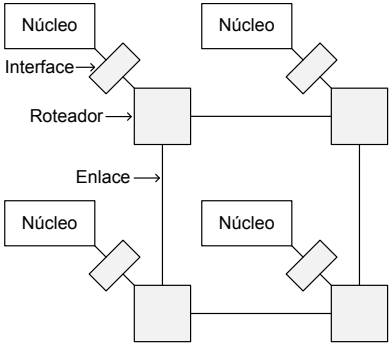


Figura 1. Componentes de uma Rede-em-Chip

2.2 Arquitetura de Interface de Rede

Um modelo genérico de arquitetura de interface de rede, conforme proposto em [6], é ilustrado na Figura 2. A visão estrutural desse bloco de hardware inclui as unidades *front-end* e *back-end*. A primeira unidade implementa a camada de Sessão do modelo de referência OSI, enquanto a segunda implementa as camadas inferiores (Transporte, Rede, Enlace e Física).

A unidade *front-end* implementa um protocolo ponto-a-ponto padronizado que permite o reuso de núcleos por diversas plataformas. Uma prática comum no desenvolvimento deste tipo de unidade é a compatibilidade com protocolos de comunicação existentes, tais como AMBA (*Advanced Microcontroller Bus Architecture*) e OCP (*Open Core Protocol)*. O modelo de comunicação é baseado em transações do tipo mestre e escravo. Mestres iniciam as transações por meio da emissão de requisições que, posteriormente, podem ser separadas em comandos e dados, sendo que um ou mais escravos recebem e executam cada requisição. Uma transação pode envolver uma resposta proveniente do escravo em direção ao mestre, para retornar dados ou apenas confirmar o atendimento da requisição. Dessa forma, a unidade

front-end pode ser vista como a implementação da camada de Sessão do modelo de referência OSI.

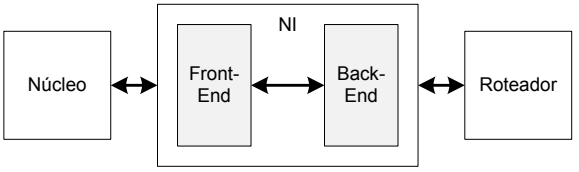


Figura 2. Estrutura básica de uma interface de rede

Serviços de comunicação de alto nível disponibilizados aos núcleos pela camada de Sessão devem ser implementados na unidade *back-end*. Essa unidade utiliza serviços da camada de Transporte para prover a transferência confiável de dados. A unidade também realiza o empacotamento de dados e as funções da camada de Rede relacionadas ao roteamento. Já a camada de Enlace assegura a confiabilidade da comunicação por meio de estratégias de detecção e de recuperação de erros, com o controle de fluxo realizado por meio da sinalização da chegada de dados provenientes do núcleo. Por fim, a camada Física deve ser projetada para atender os desafios do domínio físico, tais como cruzamento de domínios de relógio, enlaces com alta frequência de operação, esquemas de comunicação tolerantes a ruído, etc.

Em outra visão, [6] agrupa os serviços tipicamente oferecidos pelas interfaces de rede em quatro categorias: adaptação, relógio, rede e funcionais, as quais combinam serviços de diferentes camadas do modelo OSI. Os serviços de adaptação ajustam o protocolo de comunicação do componente ao protocolo de comunicação da rede. Os serviços de relógio dizem respeito ao sincronismo de relógio, adaptações de fase e ao gerenciamento de mecanismos para garantia de latência. Os serviços de rede incluem o ordenamento de transações, o provimento de transações confiáveis e a regulagem do fluxo de pacotes ao longo da rede. Os serviços funcionais adicionam novas funcionalidades ao sistema, como, por exemplo, coerência de cache, níveis de segurança (das informações ou do desempenho do sistema) e redução do consumo de energia (ou seja, da potência dissipada).

Um serviço obrigatório nas interfaces de rede diz respeito à memorização temporária (*buffering*) dos pacotes a serem enviados para a rede e dos pacotes recebidos da mesma. Esse serviço é necessário para assegurar que os pacotes não sejam descartados quando não puderem ser encaminhados imediatamente ao seu destinatário.

3 Trabalhos Relacionados

Diversos grupos de pesquisa têm realizado estudos sobre interfaces de rede para NoCs. Nesta seção, são descritos brevemente alguns desses estudos com o intuito de identificar a arquitetura dessas interfaces e os serviços de comunicação oferecidos. Esta análise serviu de referência para a especificação da interface de rede proposta.

Em [7], foi apresentada uma interface de rede para NoCs constituída de três unidades: Interface Genérica, Empacotador e Desempacotador. A Interface Genérica desacopla o núcleo da rede, abstraindo o conteúdo proveniente do núcleo e disponibilizando apenas informações de dado e de controle às demais unidades. O Empacotador é responsável por construir um pacote com os dados recebidos da Interface Genérica e injetá-lo na NoC. O Desempacotador recebe pacotes da rede, desempacota-os e entrega seus dados à Interface Genérica.

Em [8], foi apresentada uma alternativa de interface de rede com compartilhamento total de recursos, visando um melhor aproveitamento da lógica empregada em sistemas baseados em NoCs. O módulo responsável por agrupar o fluxo de dados proveniente dos núcleos mestres é denominado Agrupador de Tráfego, enquanto a divisão e a distribuição de dados para os núcleos escravos são desempenhadas pelo Divisor de Tráfego. A interface foi desenvolvida visando a integração de núcleos compatíveis com o padrão OCP.

Em [9], foram desenvolvidas interfaces de rede distintas para núcleos mestre e escravo compatíveis com o padrão AMBA AXI (*Advanced eXtensible Interface*). Cada interface é subdividida em dois blocos que determinam o caminho das transações: direto ou reverso. O bloco Caminho Direto é responsável por transmitir as transações AXI provenientes de um núcleo para o roteador, enquanto o Caminho Reverso recebe os pacotes de um roteador e os converte em transações AXI para serem apresentadas ao núcleo. O bloco Caminho Direto é constituído de uma fila AXI, uma unidade empacotadora e uma unidade de reordenação compartilhada com o bloco Caminho Reverso, o qual ainda inclui uma fila de pacotes e uma unidade desempacotadora.

Em [10], foi apresentada uma interface de rede compatível com o protocolo de comunicação AMBA AHB (*Advanced High-performance Bus*). A arquitetura da interface é dividida em módulos de injeção e extração, tanto para a conexão com a NoC como também para com o núcleo. A unidade encarregada da comunicação com a NoC é denominada *Kernel* e é responsável pelo controle de fluxo fim-a-fim e pelo desempacotamento de pacotes. A unidade ligada ao núcleo é denominada *Shell* e é responsável pelo controle de fluxo nos barramentos de comunicação externos e pelo empacotamento de pacotes.

Em [11], foi proposta uma interface de rede específica para uso em SoCs com características de tráfego do tipo *dataflow* com necessidade de sincronismo entre diferentes fluxos de comunicação, requisito típico de decodificadores H.264 usados no SBTVD (Sistema Brasileiro de Televisão Digital). A interface de rede disponibiliza funcionalidades como empacotamento, desempacotamento, sincronismo e adaptação do protocolo de comunicação entre o núcleo e o roteador. A principal contribuição do trabalho está no bloco responsável pelo sincronismo.

Em [12], foi apresentada uma interface de rede modular que busca atender requisitos de qualidade de serviço (QoS – *Quality-of-Service*), incluindo baixa latência e flutuação mínima de tráfego. Esses requisitos são atendidos por meio do uso de *buffers* separados para armazenamento do cabeçalho e da carga útil dos pacotes, permitindo assim a recepção de um novo pacote antes que o pacote atual seja completamente recebido.

Em [13], foi apresentada uma interface de rede de baixo custo para emprego em NoCs baseadas em comunicação TDM (*Time-Division Multiplexing*). A arquitetura proposta não

utiliza *buffers* FIFO e nem controle de fluxo baseado em crédito, o que, segundo os autores, respondem por 50 a 85% dos recursos de área consumidos pelas interfaces de rede existentes. Tal solução foi possível movendo-se os controladores de DMA (*Direct Memory Access*) dos núcleos de processamento para a interface de rede e sincronizando esses controladores com o escalonamento TDM da interface.

Em [14], é descrita uma arquitetura de interface de rede que visa prover o suporte em hardware para um conjunto avançado de funcionalidades de rede, incluindo: gerenciamento de erros, gerenciamento de potência, ordenamento, segurança, gerenciamento de QoS, programabilidade, interoperabilidade e remapeamento.

Com base na análise dos trabalhos relacionados, conforme resumido na Tabela 1, observa-se que todas as interfaces implementam os serviços básicos de interfaceamento, empacotamento, desempacotamento, controle de fluxo e de memorização (exceto em [13], que utiliza interfaces *bufferless*). Poucas propostas oferecem serviços de relógio [11][12] ou serviços funcionais [14]. O presente trabalho apresenta uma interface de rede que implementa todos os serviços básicos encontrados nas demais interfaces de rede e serviços funcionais implementados apenas em [14].

Tabela 1. Serviços implementados nas interfaces de rede

Serviço	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
Interfaceamento	×	×	×	×	×	×	×	×
Empacotamento/Desempacotamento	×	×	×	×	×	×	×	×
Memorização	×	×	×	×	×	×		×
Controle de fluxo	×	×	×	×	×	×	×	×
Serviços da classe Relógio					×	×		
Serviços da classe Funcional								×

Cabe destacar que, em geral, nos trabalhos analisados, as interfaces de rede foram implementadas com o uso de linguagens de descrição de hardware e sintetizadas para tecnologias de lógica programável ou de ASIC – *Application Specific Integrated Circuit*. As métricas de avaliação comumente utilizadas nesses trabalhos foram área de silício ocupada e potência dissipada. Alguns trabalhos também obtiveram métricas de desempenho, tais como latência da comunicação, vazão e frequência máxima de operação.

4 Arquitetura Extensível de Interface de Rede para a NoC SoCIN

Com o objetivo de permitir a interconexão de núcleos à rede SoCIN, foi proposta uma arquitetura de interface de rede provendo os serviços básicos de adaptação, empacotamento, desempacotamento e memorização, além de alguns serviços funcionais. Para facilitar a inclusão de novos serviços, a interface de rede foi organizada em camadas que estruturam os diferentes serviços implementados. Em virtude dessa característica, a interface foi denominada *eXtensible Interface for Routing Unit* – XIRU. Esta seção apresenta as características da SoCIN e descreve a interface de rede proposta.

4.1 Características da SoCIN

A SoCIN [3] utiliza topologia em malha 2D e técnica de chaveamento de pacotes do tipo *wormhole*, possuindo as seguintes características adicionais: (*i*) largura de canal parametrizável; (*ii*) controle de fluxo *handshake* ou baseado em créditos; (*iii*) roteamento XY ou baseado no *Turn Model*; (*iv*) arbitragem *round-robin*, randômica ou fixa; e (*v*) *buffers* FIFO parametrizáveis nos canais de entrada e de saída.

Com o propósito de prover QoS na comunicação, em [15] foram acrescentados mecanismos baseados em técnicas de chaveamento de circuito, canais virtuais e envelhecimento. Na técnica de chaveamento de circuitos, pacotes de controle são utilizados para alocar e liberar circuitos, os quais são utilizados para transferir pacotes com latência mínima. Os canais virtuais foram adotados para distinguir os fluxos de comunicação em quatro classes de tráfego e priorizar as classes com requisitos temporais mais restritos. A técnica de envelhecimento visou priorizar pacotes que estivessem trafegando há mais tempo na rede.

4.2 Serviços Previstos para a Interface de Rede

Na especificação da arquitetura de interface de rede XIRU, foi considerada as implementações imediata e futura de um conjunto de serviços de comunicação, incluindo serviços das classes adaptação, rede e funcional.

Na classe serviços de adaptação, foram previstos os serviços interfaceamento, empacotamento e desempacotamento. Com relação ao serviço interfaceamento, a estrutura da interface de rede foi definida de forma a facilitar a adaptação a diferentes protocolos intrachip, conforme será descrito a seguir.

Na classe serviços de rede, foram previstos os serviços de memorização, controle de integridade, controle de fluxo e diferenciação de fluxos. O primeiro refere-se ao armazenamento temporário dos pacotes. O segundo diz respeito à detecção e sinalização de erro em pacotes transferidos pela rede, tipicamente decorrentes de faltas transitórias (*e.g. crosstalk*). O serviço de controle de fluxo diz respeito à regulagem do tráfego transferido de modo a evitar perdas de pacotes e trata-se de um serviço nativo da rede SoCIN. O serviço de diferenciação de fluxos baseia-se na abordagem adotada em [15] e consiste na identificação dos fluxos segundo quatro classes de tráfego: RT0 e RT1 (para fluxos tempo real) e nRT0 e nRT1 (para fluxos não tempo real) de modo a permitir que os roteadores da rede possam adotar prioridades diferentes para cada classe. Adicionalmente, os pacotes são rotulados como requisições e respostas, o que viabiliza a adoção de canais virtuais distintos para esses tipos de pacotes.

Na classe serviços funcionais, foram previstos os serviços de baixa potência (do inglês, *low-power*) e de segurança. Para o primeiro, optou-se pela codificação de barramentos *bus-invert* [16] com o objetivo de reduzir o chaveamento de bits nos canais de dados e nos *buffers* da rede e, com isso, reduzir a potência dinâmica dissipada. Com relação aos serviços segurança, foi prevista a adoção de soluções para evitar ataques de negação de serviço,

incluindo controle de admissão e de banda. Essas soluções já foram implementadas sob a forma de *wrapper* de hardware [17] e serão futuramente integradas à interface de rede.

Os parâmetros da interface de rede, que determinam a síntese e o tipo de serviços suportados na versão atual, são apresentados na Tabela 2.

Tabela 2. Parâmetros da interface de rede XIRU

Parâmetro	**Valores / Alternativas disponíveis**
Parâmetros do protocolo	Larguras das vias de endereço e de dado
Natureza do núcleo	Mestre ou Escravo
Endereço de rede do nodo	Coordenadas XYZ
Arquitetura dos *buffers* FIFO	Circular ou Deslocamento
Profundidade dos *buffers* FIFOs	>= 2
Técnica de controle de fluxo	Baseada em créditos ou *handshake*
Técnica de redução de potência	*Bus-invert* ou nenhuma
Técnica de controle de integridade	Verificação de paridade ou nenhuma
Tabela de roteamento	Parâmetros de mapeamento de endereços e de classificação dos fluxos

4.3 Arquitetura da Interface de Rede

A interface de rede foi estruturada em três camadas: Específica, Genérica e Rede, as quais são ilustradas na Figura 3. A camada Específica faz parte do *front-end* da interface. Essa camada é responsável pela comunicação com o núcleo e a adaptação do seu protocolo para um protocolo genérico da camada inferior. A camada Genérica integra o *back-end* e realiza o empacotamento e o desempacotamento de dados. A camada Rede, que também integra o *back-end*, lida com o armazenamento temporário (memorização) dos pacotes e o envio e recebimento desses pacotes, realizando a comunicação direta com a NoC.

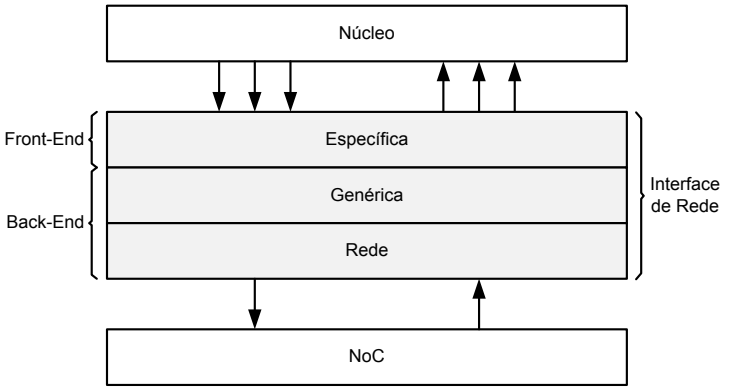


Figura 3. Arquitetura em camadas da interface de rede XIRU

4.4 Protocolo de comunicação

Para prover os serviços previstos para a interface de rede, foram feitas modificações no pacote original da rede SoCIN [3], incluindo as adaptações já feitas em [15] quando da implementação do suporte a serviços de QoS no roteador. A Figura 4 apresenta o formato de pacote proposto, o qual é constituído de dois *flits* de cabeçalho, *flits* de carga útil e um *flit* terminador do pacote.

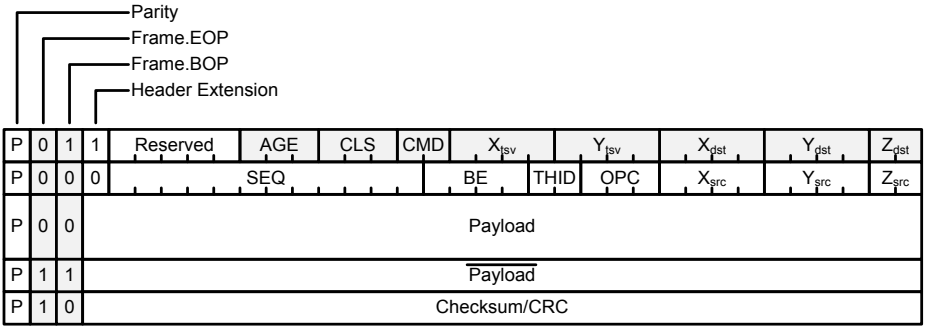


Figura 4. Formato do pacote proposto

No formato do pacote proposto, foi considerado o suporte a endereçamento em um espaço tridimensional para futura implementação da rede utilizando topologia em malha 3D. Para isso, no primeiro *flit* do cabeçalho, é incluído o endereço do destinatário baseado em coordenadas 3D (ou seja, X_{dst}, Y_{dst} e Z_{dst}), bem como as coordenadas X_{tsv} e Y_{tsv} que servem para identificar o TSV (*Through-Silicon Via*) a ser utilizado para rotear o pacote de um plano a outro em um chip 3D. Além disso, a estrutura de pacote proposta suporta a extensão de cabeçalho e os seguintes serviços:

- Chaveamento de circuitos baseado em pacotes (campo CMD);
- Diferenciação de fluxos e distinção entre requisições e respostas (campo CLS);
- Até 8 tipos de operação (campo OPC);
- Suporte à identificação de threads (campo THID);
- Sinalização de habilitação de bytes (campo BE);
- Suporte à identificação de ordem de pacotes (campo SEQ);
- Controle de integridade (bit de paridade e *flit Checksum*/CRC); e
- Codificação *bus-invert* (*Payload* invertido sinalizado nos bits de enquadramento).

A diferenciação de fluxos, realizada por meio do campo CLS, aloca canais virtuais distintos para requisições (REQ) e respostas (RSP), com 4 níveis de prioridade (RT0, RT1, nRT0, nRT1). As operações suportadas são codificadas de acordo com o padrão OCP [18].

4.5 Organização Interna da Interface de Rede

As interfaces disponibilizadas são das naturezas mestre e escravo e divididas em camadas (Específica, Genérica e Rede). A Figura 5 ilustra o diagrama de blocos geral da interface de rede mestre, identificando as unidades da interface (*front-end* e *back-end*) e as suas três camadas (Específica, Genérica e Rede).

A camada Específica é composta de uma lógica que realiza a adaptação entre a interface do núcleo e um barramento genérico derivado do processador MIPS [19]. Essa lógica é constituída de circuitos combinacionais que adaptam os sinais do protocolo do núcleo ao protocolo do barramento genérico. No caso de uma diferença entre a largura dos canais de dados e de endereço, essa camada é responsável por realizar a serialização e a deserialização necessárias (*e.g.* de 64 bits para 32 bits e vice-versa).

A camada Genérica é constituída dos blocos Empacotador e Desempacotador (ilustrados na Figura 6) e de suas respectivas unidades de controle, as quais são implementadas sob a forma de máquinas de estados finitos. O Empacotador é responsável por empacotar os dados, construindo o cabeçalho e serializando os *flits* do pacote. Ele possui uma tabela de roteamento que mapeia os endereços de um espaço de endereçamento de 32 bits em endereços de rede para a construção do primeiro *flit* do cabeçalho, o qual é encaminhado pela entrada 0 do multiplexador de saída, conforme ilustrado na Figura 6(a). O segundo *flit* do cabeçalho inclui a operação associada ao pacote (*e.g.* escrita, leitura), o endereço de rede de origem (para retorno da resposta) e outras informações (não representadas na figura). O Desempacotador realiza o processo inverso, extraindo o cabeçalho e deserializando os *flits* do pacote, conforme mostra a Figura 6(b).

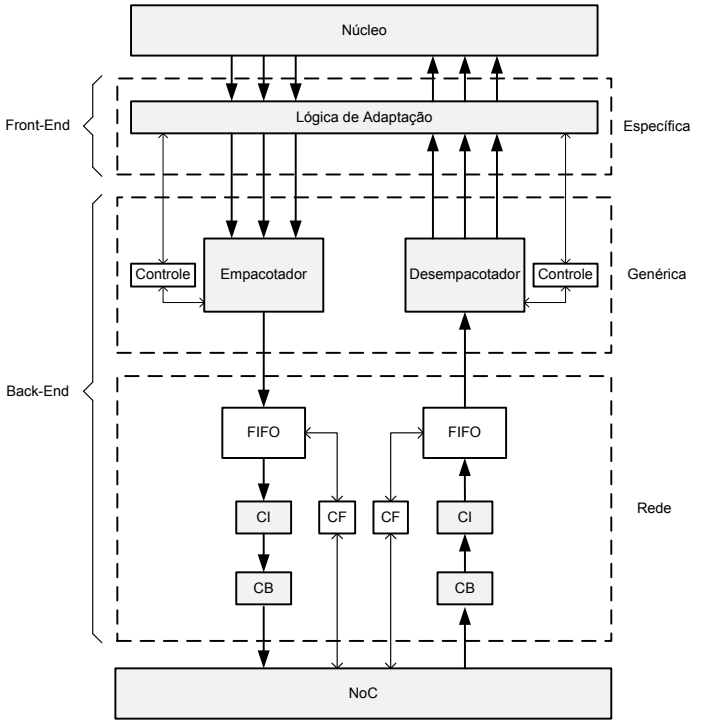
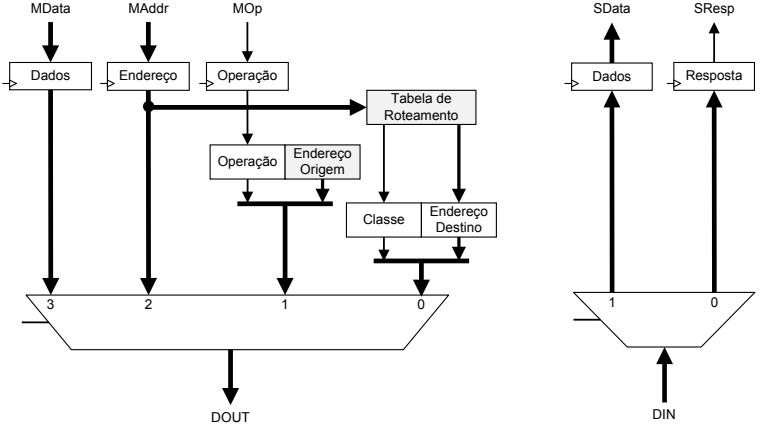


Figura 5. Diagrama de blocos da interface de rede XIRU



(a) (b)

Figura 6. Módulos da camada Genérica: (a) Empacotador; (b) Desempacotador

A camada Rede inclui os *buffers* FIFO de envio e de recepção que armazenam temporariamente os pacotes transferidos. Esta camada também inclui os blocos Controle de Integridade (CI), Codificação de Barramento (CB) e Controle de Fluxo (CF).

Os blocos CI constituem-se em unidades de cálculo de paridade baseado em paridade par. No emissor, o bloco CI determina se a quantidade de bits em 1 em cada *flit* a ser transmitido é ímpar e, neste caso, atribui 1 ao bit de paridade, do contrário atribui 0. No lado da recepção, o cálculo do número de bits em 1 é repetido e o resultado é comparado com o bit de paridade indicado pelo emissor. No caso de detecção de erro de paridade, o *flit* é descartado e o erro é sinalizado à camada superior por um código de erro.

Os blocos CB aplicam a técnica *bus-invert* para a codificação (no envio) e a decodificação (na recepção) dos *flits* do pacote a fim de reduzir a atividade de chaveamento na transferência da carga útil do pacote pela rede. No emissor, cada *flit* a ser enviado é comparado com o *flit* encaminhado anteriormente a fim de identificar quantos dos *N* bits do canal variaram entre os dois *flits* (conforme ilustrado na Figura 7). Se a variação for maior que *N*/2, os bits do *flit* a ser enviado são invertidos a fim de reduzir a atividade de chaveamento associada à transferência desses *flits*. Com isso, reduz-se também o consumo de energia para transmissão do pacote. Os *flits* invertidos são sinalizados fazendo-se a atribuição de início e fim de pacote iguais a 1, sendo que o bloco CB do receptor complementa esses *flits* no momento da recepção para restaurar o pacote original.

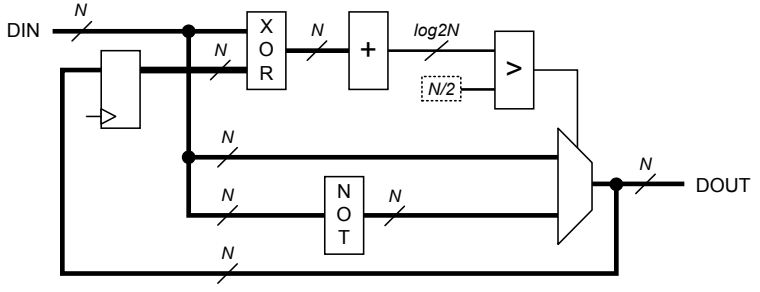


Figura 7. Bloco CB do emissor

Os blocos CF lidam, essencialmente, com o controle de fluxo em nível de *flit* e regulam a transferência de dados entre um emissor e um receptor interligados por um canal. A interface de rede suporta as mesmas técnicas adotadas na rede SoCIN, ou seja: baseada em créditos e *handshake*. No controle de fluxo baseado em créditos, o bloco CF do emissor possui um contador que monitora a capacidade do *buffer* FIFO do canal de entrada do receptor. A cada *flit* enviado, uma posição desse *buffer* é ocupada e um crédito é descontado do contador. Quando o contador atinge o valor 0, o que representa que o *buffer* do receptor não possui espaço disponível, o emissor suspende o envio de *flits* até que um crédito seja recebido do receptor, o que indica que uma posição do *buffer* foi liberada. No lado do receptor, o bloco CF é responsável por enviar um crédito ao emissor cada vez que um *flit* previamente recebido e armazenado no seu *buffer* de entrada é reencaminhado. Na segunda alternativa de técnica de

controle de fluxo (*handshake*), os blocos CF do emissor e do receptor utilizam dois sinais assíncronos para realizar a regulação do tráfego de *flits*, os quais são denominados validação e reconhecimento. O sinal de validação indica a disponibilidade de um novo dado para envio pelo emissor, enquanto o sinal de reconhecimento informa quando o receptor está apto a receber esse dado. Esse protocolo é implementado por duas máquinas de estados e a transferência do *flit* ocorre quando os dois sinais de controle de fluxo estão ativos em 1 ao mesmo tempo.

5 Implementação e Resultados

5.1 Implementação

A interface de rede XIRU foi descrita em VHDL utilizando-se o ambiente de desenvolvimento Quartus II versão 13.0 da Altera. A implementação seguiu uma abordagem *bottom-up*, com a descrição dos blocos construtivos de cada camada, seguida da integração desses blocos nos módulos correspondentes às três camadas da interface (Específica, Genérica e Rede). Posteriormente, esses módulos foram integrados para formar as unidades *front-end* e *back-end* que compõem a interface de rede.

O módulo da camada Específica implementa a lógica de interfaceamento entre o protocolo do barramento Avalon da Altera [20] e o barramento genérico. Como esses dois barramentos são bastante similares e utilizam canais de dado e de endereço com as mesmas larguras (32 bits), a lógica implementada limita-se à conversão entre sinais do barramento Avalon e comandos do barramento genérico.

A camada Genérica é responsável pelo empacotamento e desempacotamento das informações trocadas entre o núcleo e a NoC, podendo ser de duas naturezas distintas, mestre ou escravo, cada qual com seu próprio caminho de dados e unidade de controle. A camada Genérica da interface de rede escravo diferencia-se por não possuir uma tabela de roteamento, dado que apenas responde a requisições recebidas de algum mestre. Para envio da resposta, a camada possui um registrador que armazena o endereço de rede do requisitante, o qual é incluído como endereço destino no momento da construção do pacote de resposta pelo Empacotador.

Por fim, o módulo correspondente à camada Rede é composto de sub-módulos de envio e de recepção de pacotes. O sub-módulo de envio de pacotes realiza a memorização, a codificação *bus-invert*, o cálculo de paridade e a injeção de pacotes na NoC, enquanto o sub-módulo de recepção realiza a extração de pacotes da NoC, a decodificação, o cálculo e a verificação de paridade, seguidos da memorização em um *buffer* FIFO.

5.2 Verificação

Na Figura 8, é ilustrado o diagrama de sequência de uma transação composta de uma requisição seguida de uma resposta, como ocorre em operações de leitura de um mestre para um escravo. Inicialmente, o mestre envia uma requisição à interface de rede mestre que, após o recebimento, executa o empacotamento da mensagem e a posterior injeção na NoC. Na rede,

o pacote com a requisição é encaminhado até o roteador ao qual a interface de rede do nodo escravo está conectada. Essa interface realiza o desempacotamento e a entrega da requisição ao núcleo escravo. O núcleo escravo, por sua vez, realiza a tarefa solicitada (*e.g.* leitura de um dado) e responde à interface de rede escravo. Esta, então, empacota a resposta e a injeta na NoC para encaminhamento à interface de rede mestre. Finalizando, essa interface desempacota a resposta e a entrega ao núcleo que originou a requisição.

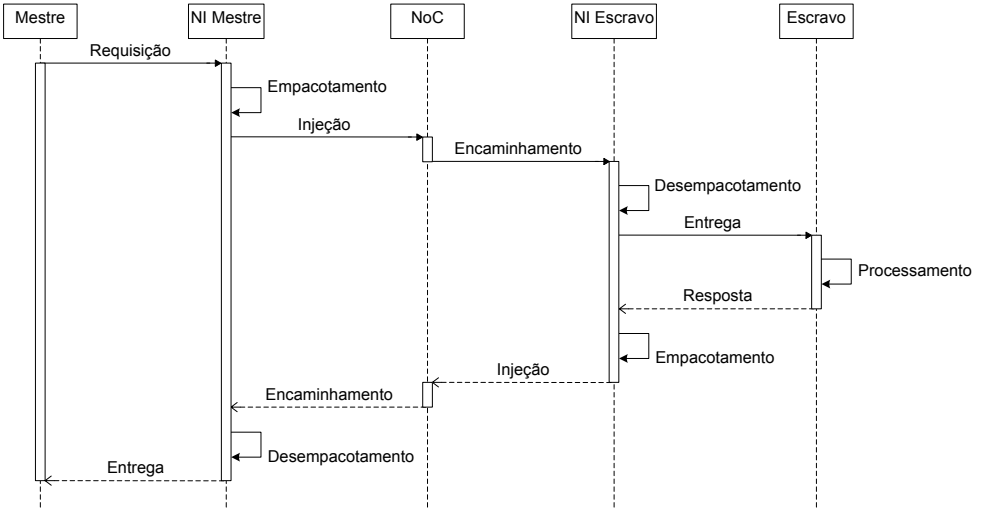


Figura 8. Diagrama de sequência de uma requisição mestre/escravo com resposta

Para a validação da interface descrita, foram simuladas operações de escrita e de leitura provenientes de um núcleo do tipo Avalon mestre. A Figura 9 apresenta o diagrama de formas de onda para uma operação de leitura, incluindo apenas os sinais associados a essa operação e um contador de ciclos decorridos. O diagrama de formas de onda segue o diagrama de sequência previamente ilustrado.

A operação de leitura é iniciada no ciclo 2 com a ativação dos sinais `chipselect`, `begintransfer` e `read` pelo mestre do barramento Avalon em conjunto com a definição do endereço destino (`address = 0xAAAA`). Após, no ciclo 3, a interface de rede inicia a construção de um pacote de requisição de leitura constituído por dois *flits* de cabeçalho e um *flit* com o endereço destino. Nos ciclos 4 a 6, o pacote de requisição de leitura é injetado pelo canal de saída (`OUT`). Continuando, nos ciclos 9 a 11, o pacote de resposta da leitura, com dois *flits* de cabeçalho e um *flit* de dado, é recebido pela interface de rede pelo canal de entrada (`IN`). Concluindo, no ciclo 12, o dado recebido é entregue ao mestre pelo sinal `readdata~result`. Os sinais da operação de leitura (`read` e `waitrequest~result`) mantêm-se ativos até que o dado lido seja recebido e entregue ao mestre. A fim de conferir uma melhor legibilidade à figura, no diagrama, não é levada em consideração a latência da rede para encaminhamento dos pacotes de requisição e de resposta.

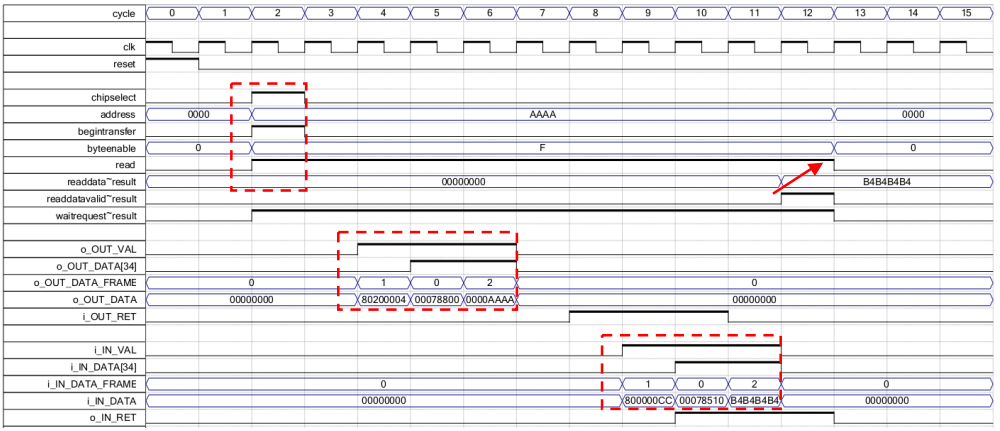


Figura 9. Diagrama de formas de onda para uma operação de leitura

A Figura 10 apresenta o diagrama de formas de onda de simulação para uma operação de escrita, incluindo apenas os sinais da interface de rede associados a essa operação. A operação de escrita é iniciada no ciclo 2 com a ativação dos sinais `chipselect`, `begintransfer` e `write` pelo mestre com a definição do endereço destino (`address = 0xAAAA`) e do dado a ser escrito (`writedata = 0xFFFFFFFF`). No ciclo 3, a interface de rede inicia a construção de um pacote de requisição de escrita constituído por dois *flits* de cabeçalho, um *flit* com o endereço destino e um *flit* com o dado. Nos ciclos 4 a 7, o pacote é injetado pelo canal de saída (`OUT`). A partir do início da operação de escrita (ciclo 2) e até o momento em que o pacote é montado e escrito no *buffer* de saída (ciclo 6), a interface de rede ativa o sinal `waitrequest~result` a fim de evitar que uma nova operação seja iniciada pelo mestre. Quando este sinal é desativado, o sinal de escrita é baixado pelo mestre, habilitando-o a iniciar uma nova operação.

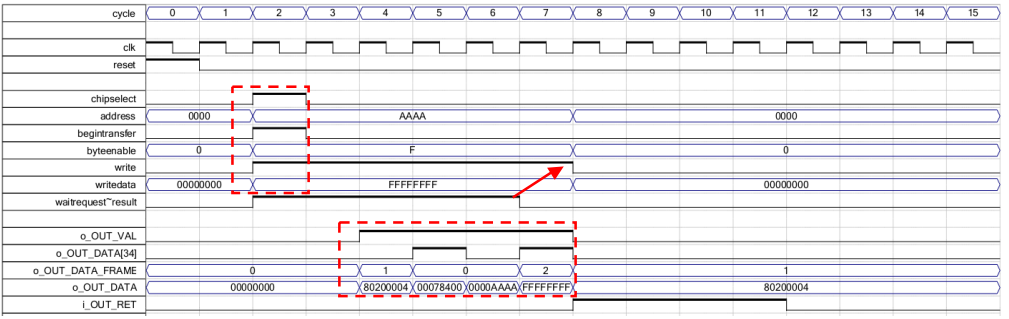


Figura 10. Diagrama de formas de onda para uma operação de escrita

5.3 Estimativa de desempenho

A interface de rede consome 2 ciclos entre o início de uma operação de escrita ou de leitura e a injeção do pacote de requisição na rede. Já no lado da recepção, a interface consome 1 ciclo entre o recebimento do terminador do pacote e a entrega do dado recebido ao núcleo. Já a latência mínima da rede para envio de um pacote com *N flits* entre dois nodos separados por *D* roteadores, é dada por (1), considerando a rede livre de contenção e com controle de fluxo baseado em créditos.

$$Latência_{\text{NoC}} = 3 \cdot D + N - 1 \tag{1}$$

Dessa forma, as latências mínimas para execução de operações de escrita e de leitura, entre dois nodos separados por *D* roteadores, considerando os tamanhos dos correspondentes pacotes (4 *flits* para escritas e 3 *flits* para leituras), são dadas por (2) e (3).

$$Latência_{\text{Escrita}} = 2 + (3 \cdot D + 4 - 1) + 1 = 3 \cdot D + 6 \tag{2}$$

$$Latência_{\text{Leitura}} = 2 \cdot [2 + (3 \cdot D + 3 - 1) + 1] = 6 \cdot D + 10 \tag{3}$$

Ao considerar dois nodos separados por apenas dois roteadores (distância mínima), a latência mínima de uma operação de escrita é igual a 12 ciclos. Já a latência mínima de uma operação de leitura é igual a 22 ciclos, sem considerar a latência de acesso ao dado no nodo escravo. Portanto, a contribuição máxima da interface de rede em uma operação de escrita é igual a 25% da latência total. Porém, essa contribuição decai com o aumento da distância entre os nodos, conforme ilustra a Figura 11. Já em uma operação de leitura, a contribuição da interface de rede é de 27,3%, também decaindo com o aumento da distância percorrida pelo pacote. Ressalta-se que esses valores percentuais são menores se for levado em conta o fato de que a latência do pacote tende a aumentar com a competição pelo uso dos recursos da rede com outros pacotes (contenção), reduzindo a contribuição das interfaces de rede à latência total do pacote.

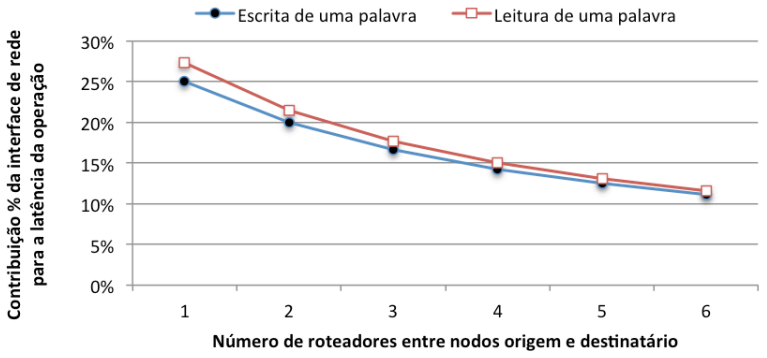


Figura 11. Impacto da interface de rede na latência de comunicação

5.4 Resultados de Síntese

A fim de verificar os custos da interface implementada, cada modelo foi testado e validado com a ferramenta Quartus II e sintetizado para uma tecnologia ASIC de 32nm, utilizando a ferramenta Design Compiler da Synopsys, versão F-2011.09, com a biblioteca de células *saed32/28*. A interface de rede foi configurada utilizando *buffers* FIFO do tipo circular com quatro posições de 32 bits (dimensões adotadas no roteador ParIS), controle de fluxo baseado em créditos, codificação *bus-invert* e controle de integridade baseado na verificação de paridade. Foram sintetizadas duas versões da interface de rede XIRU, uma suportando apenas os serviços básicos (XIRU *lite*) e outra suportando todos os serviços adicionais já implementados (XIRU *full*).

A Tabela 3 apresenta os custos das duas versões da interface de rede mestre em termos de área de silício expressa em μm^2, do número de portas lógicas equivalentes e da potência dissipada. O número de portas de lógicas é obtido pela divisão da área total do circuito pela área da porta lógica *nand* com duas entradas (*nand2*) na tecnologia utilizada. Esses valores são comparados com os de uma instância do roteador ParIS com 5 portas de comunicação, canais de dados de 32 bits e *buffers* FIFO com capacidade de armazenar 4 *flits*. Como pode ser observado, a interface de rede possui custo aproximado entre 40,4% (versão *lite*) e de 60,6% (versão *full*) da área de silício roteador. Quanto à potência, o custo varia entre 42,5% (versão *lite*) e de 72,7% (versão *full*) da potência dissipada pelo roteador.

Tabela 3. Custos da interface de rede XIRU e do roteador ParIS

Núcleo	**Área (μm^2)**	**Portas lógicas**	**Potência (μW)**
Interface de Rede XIRU *lite*	4.726,57	3.098	136,8
Interface de Rede XIRU *full*	7.558,50	4.957	234,2
Roteador ParIS	11.694,94	7.670	322,2

Pela análise da Tabela 3, observa-se que os serviços adicionais incluídos na versão *full* resultam um sobrecusto importante em área (37.5%) e em potência (41,6%) com relação à versão *lite*.

A Tabela 4 apresenta os custos percentuais das diferentes camadas e serviços da interface de rede XIRU *full*. É possível observar que a camada Rede possui os serviços de maior custo, sendo que o serviço memorização consome a maior parte dos recursos combinacionais (51%) e sequenciais (62%), seguido do serviço codificação de barramento. Os custos do interfaceamento e do controle de fluxo são praticamente desprezíveis, enquanto o empacotamento, o desempacotamento e o controle de integridade consomem uma pequena área.

Tabela 4. Custos dos serviços implementados na interface de rede XIRU *full*

Camada	**Serviço**	**Área Combinacional**	**Área Sequencial**
Específica	Interfaceamento	0,76 %	0,27 %
Genérica	Empacotamento	8,04 %	15,85 %
	Desempacotamento	5,75 %	11,48 %
Rede	Memorização	51,36 %	62,02 %
	Controle de fluxo	1,50 %	1,09 %
	Codificação de barramento	26,63 %	9,29 %
	Controle de integridade	5,97 %	0,00 %

A Tabela 5 compara os resultados de síntese de diferentes configurações da interface de rede XIRU (*lite* e *full* com *buffers* FIFO com profundidades de 4 e de 8 *flits*) com outras interfaces de rede descritas na literatura. A fim de facilitar essa comparação, são apresentadas apenas interfaces de rede que foram sintetizadas para tecnologias do tipo ASIC e para as quais foram reportadas a tecnologia utilizada e a área ocupada. Como essas interfaces foram sintetizadas em diferentes nodos tecnológicos, o custo de silício é expresso pelo número de portas lógicas equivalentes, pois a área de um mesmo circuito varia de acordo com a tecnologia adotada. Cabe ressaltar que poucos trabalhos dentre os que foram discutidos na Seção 3 apresentaram resultados de síntese com detalhamento suficiente para esta análise e, por isso, apenas uma parte deles foi considerada. No quadro, além da tecnologia utilizada e do custo do circuito, são apresentados dados sobre a disponibilização de serviços adicionais, a largura do canal de dados e os *buffers* FIFO utilizados (quantidade e profundidade). Ressalta-se que todos os resultados referem-se à síntese de interfaces do tipo mestre.

Tabela 5. Comparativo do custos da interface de rede XIRU com outras interfaces de rede

Núcleo	Serviços adicionais	Largura do canal (bits)	*Buffers* FIFO (Qtd x *flits*)	Tecnologia (nm)	Custo (k*gates*)
XIRU *lite*	Não	32	2 x 4	32	3,1
XIRU *lite*	Não	32	2 x 8	32	5,2
XIRU *full*	Sim(1)	32	2 x 4	32	5,0
XIRU *full*	Sim(1)	32	2 x 8	32	7,2
[12]	Sim(2)	32	3 x 4	130	6,7
[13]	Sim(2)	32	Não usa	90	81,1
[14]	Não	32	4 x 8	65	8,2
[14]	Sim(3)	64	(2 x 64) + (2 x 128)	65	41,5

Serviços adicionais:
(1) Controle de integridade e codificação de barramento
(2) Qualidade de Serviço (QoS)
(3) Gerenciamento de erros, gerenciamento de potência, ordenamento, segurança, gerenciamento de QoS, programabilidade, interoperabilidade e remapeamento.

Comparando as diferentes configurações sintetizadas da interface de rede XIRU com as demais interfaces apresentadas na Tabela 5, observa-se que: (*i*) a versão *lite* com 4 *flits* por *buffer* possui o menor custo, dado que realiza somente os serviços básicos e possui apenas dois *buffers*; e (*ii*) as demais configurações possuem custos compatíveis (e até menores) com as outras interfaces de rede de complexidade similar. Duas interfaces se destacam pelo alto custo decorrente dos serviços implementados. Em [13], são aplicadas técnicas de QoS baseadas em TDM com o emprego de tabelas (de DMA e de reserva de fatias de tempo) que ocupam uma área equivalente a 59,5 *Kgates* (73% do custo total), sendo que os serviços básicos da interface consomem o equivalente a 21,6 *Kgates*, o que é maior que o custo da versão mais cara da interface de rede XIRU. A segunda configuração de [14] apresenta um maior custo pois utiliza canais mais largos (64 bits), *buffers* profundos (dois de 64 *flits* e dois de 128 *flits*) e implementa outros serviços funcionais não considerados na interface XIRU *full*.

6 Discussão

A interface de rede proposta utiliza uma estrutura em três camadas com os objetivos de separar os serviços básicos de empacotamento e desempacotamento (implementados na camada Genérica) dos serviços de interfaceamento (implementado na camada Específica) e funcionais (implementados na camada Rede) e facilitar a adição de novos serviços à interface.

Entende-se que a estrutura adotada permite a inclusão de um novo serviço funcional à interface modificando-se apenas a camada Rede. Por exemplo, em [15] foi implementado o suporte ao serviço de diferenciação de classes de tráfego no roteador da rede SoCIN, o qual utiliza canais virtuais para priorizar pacotes de tempo real. Para suportar esse serviço na interface de rede XIRU, devem ser implementados *buffers* separados para pacotes RT e nRT e um escalonador que priorize os pacotes do *buffer* RT. Já em [17] foram implementados *wrappers* de hardware que filtram pacotes de tráfegos maliciosos que tentam degradar a

disponibilidade da rede SoCIN. Esses *wrappers* podem ser facilmente integrados à interface XIRU, posicionados logo após o *buffer* de saída da camada Rede. Caso o pacote a ser injetado viole alguma regra de segurança, ele será descartado ou retido por um tempo determinado, conforme a regra em questão. A implementação desses serviços limita-se principalmente a modificações na camada Rede da interface XIRU.

Considerando o exposto, a Figura 12 ilustra os serviços da interface de rede, destacando: (*i*) os serviços básicos essenciais; (*ii*) os serviços funcionais implementados atualmente; e (*iii*) os serviços funcionais previstos para futuras versões.

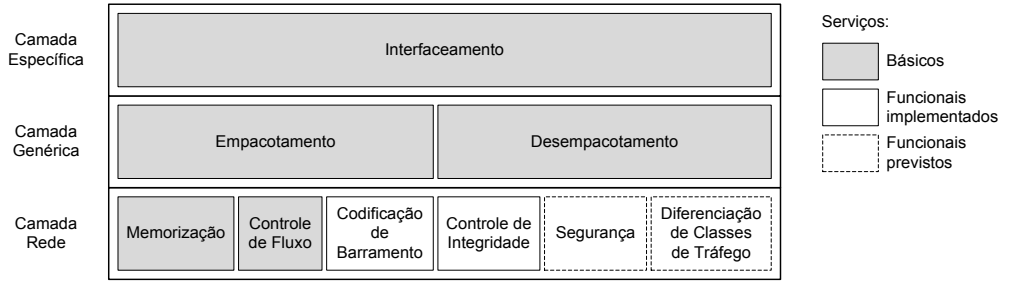


Figura 12. Serviços básicos e funcionais da interface de rede

7 Conclusões

Neste artigo, foi apresentada uma interface de rede extensível para a rede SoCIN, a qual foi modelada usando VHDL, sintetizada em tecnologia ASIC e caracterizada quanto ao seu impacto no desempenho e nos custos da rede. Foi possível identificar que a contribuição da interface na latência total da comunicação diminui com o aumento da distância percorrida pelo pacote. Observou-se também que a camada Rede é responsável por maior parte dos custos da interface, dado que implementa os serviços mais caros em termos de área (memorização e codificação de barramento).

Por se tratar de um modelo extensível e parametrizável, os custos da interface de rede podem variar de acordo com os serviços utilizados e a aplicação alvo. Uma configuração mínima que implementa apenas os serviços básicos essenciais (versão *lite*) apresenta um custo baixo, correspondente a 40% do custo do roteador da rede SoCIN. Já uma configuração mais completa (versão *full*) possui um sobrecusto maior em virtude dos serviços adicionados.

A principal contribuição deste trabalho reside na proposta de uma arquitetura de interface de rede organizada de forma a facilitar a adição de novos serviços de comunicação. O uso da estrutura em camadas permite que os serviços adicionados sejam implementados com modificações em apenas uma das camadas, com o mínimo ou nenhum impacto às demais camadas. Além dessa característica, a arquitetura proposta facilita a customização da interface de rede conforme os serviços efetivamente necessários à aplicação alvo. Com isso, é possível dimensionar a interface de modo a limitar o seu impacto no custo total da rede.

Como trabalhos futuros, pretende-se prover o suporte a outros barramentos de comunicação, adicionar uma técnica de controle de integridade em nível de pacote, implementar o suporte à diferenciação de classes de tráfego e prover técnicas de segurança para evitar ataques de negação de serviço.

8 Agradecimentos

Este trabalho contou com o apoio do CNPq e do INCT NAMITEC.

9 Referências

[1] R. K. Gupta and Y. Zorian, "Introducing Core-based System Design," *IEEE Design & Test of Computers,* vol. 14, no. 4, pp.15-25, Oct.-Dec. 1997.

[2] A. Jantsch and H. Tenhunen, *Networks on Chip*, Boston: Kluwer, 2003.

[3] C. A. Zeferino and A. A. Susin, "SoCIN: a Parametric and Scalable Network-on-Chip," *in 16th Symp. on Integrated Circuits and Systems Design* (SBCCI), São Paulo, Sept. 8-11, 2003, pp. 169-174.

[4] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Amsterdan: Morgan Kaufmann, 2004.

[5] C. A. Zeferino, *Redes-em-Chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho*, Tese de Doutorado, Programa de Pós-Graduação em Computação, UFRGS, 2003.

[6] D. Bertozzi, "Network Interface Architecture and Design Issues," in G. De Micheli and L. Benini, *Networks on Chips: Technology and Tools*, San Francisco: Morgan Kaufmann, 2006.

[7] S. Singh, S. Bhoj, D. Balasubramanian, T. Nagda, D. Bhatia and P. Balsara, "Generic Network Interfaces for Plug and Play NoC Based Architecture," *in 2nd Int. Workshop on Reconfigurable Computing: Architectures and Applications* (ARC), Delft, March 1-3, 2006, pp. 287-298.

[8] A. Ferrante, S. Medardoni and D. Bertozzi, "Network Interface Sharing Techniques for Area Optimized NoC Architectures," in *11th Euromicro Conf. on Digital System Design Architectures, Methods and Tools* (DSD), Parma, 3-5, Sept. 2008, pp. 10-17.

[9] M. Ebrahimi, M. Daneshtalab, N. P. Sreejesh, P. Liljeberg and H. Tenhunen, "Efficient Network Interface Architecture for Network-on-Chips," *in 27th IEEE NORCHIP Conf.,* Trondheim, Nov. 16-17, 2009, pp.1-4.

[10] B. Attia, W. Chouchene, A. Zitouni, A. Nourdin and R. Tourki, "Design and Implementation of Low Latency Network Interface for Network on Chip," *in 5th Int. Design and Test Workshop* (IDT), Abu Dhabi, Dec. 14-15, 2010, pp. 37-42.

[11] D. Matos, M. Costa, L. Carro and A. A. Susin, "Network Interface to Synchronize Multiple Packets on NoC-based Systems-on-Chip," *in 18th IEEE/IFIP Conf. VLSI System on Chip* (VLSI-SoC), Florianópolis, Sept. 27-29, 2010, pp. 31-36.

[12] B. Attia, A. Zitouni, W. Chouchene, K. Tourki and R. Tourki, "A Modular Network Interface Design and Synthesis Outlook," *in International Journal of Computer Science Issues* (IJCSI), v. 9, n. 3, 2012.

[13] J. Sparsø, E. Kasapaki, and M. Schoeberl. "An area-efficient network interface for a TDM-based Network-on-Chip," *in Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 1044-1047.

[14] S. Saponara, T. Bacchillone, E. Petri, L. Fanucci, R. Locatelli and M. Coppola. "Design of an NoC Interface Macrocell with Hardware Support of Advanced Networking Functionalities," *in IEEE Transactions on Computers*, vol.63, no.3, pp. 609-621, March 2014.

[15] M. D. Berejuck and C. A. Zeferino, "Adding Mechanisms for QoS to a Network-on-Chip," *in 22th Symp. on Integrated Circuits and Systems Design* (SBCCI), Natal, Aug. 31 – Sept. 3, 2009, pp. 153-158.

[16] M. R. Stan and W. P. Burleson, "Bus-invert Coding for Low-Power I/O," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 3, no. 1, pp. 49-58, March 1995.

[17] S. Baron, M. S. Wangham and C. A. Zeferino, "Security Mechanisms to Improve the Availability of a Network-on-Chip," *in IEEE Int. Conf. on Electronics, Circuits, and Systems* (ICECS), 2013, Abu Dhabi, Dec. 8-13, 2013.

[18] *Open Core Protocol Specification 3.0*. Redwood City: OCP-IP Association, 2009.

[19] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: the Hardware/Software Interface*, *5th ed.*, Amsterdan: Morgan Kaufmann, 2014.

[20] *Avalon Interface Specifications 13.0*. San Jose: Altera, 2014.