Interfaces de Rede para Comunicação com Periféricos em MPSoCs

Tadeu Marchese, Fernando Gehm Moraes
School of Technology - PUCRS – Av. Ipiranga 6681, 90619-900, Porto Alegre, Brazil
tadeu.marchese@edu.pucrs.br, fernando.moraes@pucrs.br

***Resumo*—O aumento da complexidade em projetos de circuitos devido a quantidade de módulos de hardware pré-validados (IPs - Intellectual Properties) integrados em um mesmo chip, motivou o desenvolvimento de modelos de comunicação escaláveis como redes intra-chip (NoCs - Network-on-Chip). Módulos IPs são utilizados como forma de reduzir o custo de engenharia e tempo de projeto, sendo sua reutilização facilitada pela adoção de interfaces e protocolos padronizados. A possibilidade de executar aplicações de origens e perfis diferentes paralelamente em sistemas multiprocessados (MPSoC - Multiprocessor-System-on-Chip) torna-os suscetíveis a ataques de aplicações maliciosas, que podem se aproveitar do compartilhamento de recursos provido pela plataforma para extrair informações sensíveis ou até mesmo impedir o funcionamento. O objetivo do presente artigo é propor uma interface de rede para um MPSoC a fim de prover controle de acesso aos periféricos e ainda abstrair o protocolo interno do MPSoC, permitindo a integração de IPs de terceiros.**

***Index Terms*—Interface de rede; Rede intra-chip; Mecanismos de comunicação de entrada e saída.**

I. Introdução

À medida que mais aceleradores de hardware, neste artigo denominados IPs e periféricos, são integrados em um mesmo chip e tem seus recursos compartilhados entre diferentes processadores, aumenta-se a complexidade na comunicação, justificando a adoção de NoCs. Estes sistemas possuem grande poder de processamento e são capazes de executar diversas aplicações de diferentes características paralelamente. Exemplos de aplicações que demonstram a aplicabilidade de arquiteturas multiprocessadas incluem: (*i*) processamento digital de sinais: MPEG4 (H264), vídeo 4K e 8K, renderização de imagem em tempo real, realidade aumentada; (*ii*) reconhecimento de padrões: biometria, reconhecimento facial e de voz, sequenciamento de DNA; (*iii*) sistemas embarcados; (*iv*) *Internet of Things* (IoT).

A possibilidade de executar aplicações de origens e perfis diferentes paralelamente, em uma mesma plataforma, também torna-a suscetível a ataques. Aplicações maliciosas podem se aproveitar do compartilhamento de recursos provido pela plataforma para extrair informações sensíveis, impedir o funcionamento de outra aplicação e impactar o funcionamento geral do sistema. Assim, faz-se necessário um mecanismo seguro de comunicação com periféricos, restringindo também o acesso somente para entidades autorizadas.

Embora muitos sistemas embarcados possuam requisitos de segurança, a sua grande maioria também possui uma limitação de recursos de hardware e por isto necessitam de mecanismos de segurança de baixo custo. Um dos desafios de engenharia no projeto destes dispositivos é o balanceamento entre requisitos de segurança, custo e desempenho. Os algoritmos de criptografia leve (LWC - *Lightweight Cryptography*), surgem para suprir esta necessidade trazendo um nível de segurança aceitável com baixo impacto [1].

Este artigo tem por *objetivo* propor uma interface de rede (NI - *network interface*) que atua no controle da comunicação entre os elementos de processamento (PEs) e periféricos conectados ao MPSoC [2], como apresentado na Figura 1, sendo responsável por: (*i*) avaliar as requisições de leitura e escrita realizadas, podendo recusá-las ou aceitá-las; (*ii*) empacotar as informações, isto é traduzir de sinais específicos do periférico para forma de pacotes a ser transmitida na NoC; (*iii*) proteger os dados das mensagens através de criptografia leve evitando a exposição de informações sensíveis.

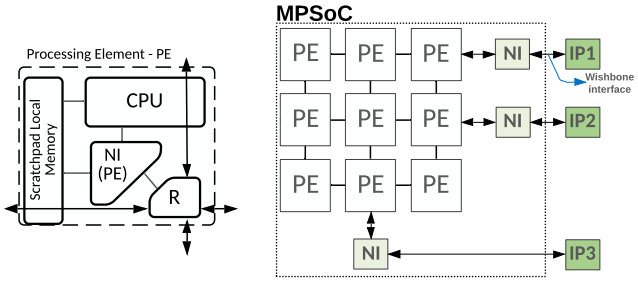

Figura 1. Visão geral do MPSoC. A NI proposta se posiciona entre os roteadores (R) e periféricos.

As *contribuições* deste artigo incluem: (*i*) integração de periféricos de terceiros implementando o suporte a protocolos padronizados; (*ii*) restrição da utilização dos periféricos somente à aplicações autorizadas através da definição de políticas de acesso; (*iii*) garantir a autenticidade e integridade das mensagens trocadas internamente no MPSoC entre as tarefas e periféricos através do uso de mecanismos de autenticação e assinatura.

II. Estado-da-arte

Sepúlveda et al. [3] propõe uma NI segura (SNI – Security NI) para MPSoCs baseados em NoC, cujo objetivo é estabelecer uma comunicação segura entre zonas-seguras e IPs. A SNI é responsável pela autenticação, controle de acesso e confidencialidade, além de empacotar e desempacotar os pacotes transmitidos. A autenticação e o controle de acesso são implementados através de uma tabela de segurança e a confidencialidade é realizada através da criptografia de dados. A tabela de segurança contém a informação de quais PEs são autorizados a se comunicar com uma zona segura. Cada vez que um pacote chega na SNI, as informações de roteamento do pacote, como origem e destino, são extraídas para verificação

junto ao conteúdo armazenado na tabela de segurança. Caso o pacote seja transmitido de um PE não autorizado a transação é negada. O *overhead* de área adicionado pela SNI, em relação à implementação não segura, é de aproximadamente 9,2%, 9,8% de energia e 14,8% de latência para uma zona segura de tamanho 8, podendo chegar a 27,8% de energia e 40% de latência para zonas seguras de tamanho 64.

Melo et al. [4] propões a implementação de uma NI para a NoC SOCIN. A NI proposta está dividida em três partes: específica, genérica e rede. O principal objetivo desta divisão é modularizar a implementação de protocolo, de forma a tornar a NI extensível para que seja possível suportar outros protocolos além do Avalon (protocolo utilizado pela Altera). A camada genérica é responsável pelo empacotamento e desempacotamento das mensagens, podendo atuar como mestre ou escravo.

Holsmark et al. [5] realizam o desenvolvimento e prototipação de uma NI sem requisitos de segurança para NoCs. A NI (BNI - *Bus-to-Network*) é responsável pela comunicação entre a rede e IPs (protocolo OPB). A prototipação foi realizada em uma plataforma Virtex II com FPGA e foram utilizados apenas 46% dos recursos de área disponíveis, sendo que a BNI ocupou apenas 1% do FPGA empregado pelo Autor. O desempenho foi medido em latência de um IP para outro passando por duas BNIs, resultando em 380 ns a uma frequência de 100 MHz.

Singh et al. [6] propõe uma NI para conectar aceleradores de hardware a NoCs. Para isto, foi desenvolvido um *wrapper* genérico para um IP de compressão de imagens a fim permitir a comunicação do IP e da NI através de um protocolo padrão (UART). A implementação foi realizada em FPGA e os resultados apontam um overhead de área mínimo no MPSoC, cerca de 48 slices e 57 flip-flops para o *wrapper* do IP JPEG, 227 slices e 228 flip-flops para a NI. A arquitetura da NI possui três funções: (1) abstrair o protocolo de comunicação com a rede, permitindo desenvolver esta parte de forma independente e tornando a interface genérica; (2) tornar possível a comunicação com qualquer IP que implemente um protocolo específico através do uso do *wrapper* que permite a configuração de tamanho do bloco de dado; (3) empacotamento e desempacotamento das mensagens para envio e recebimento através da NoC.

A maioria dos trabalhos relacionados a NI, como apresentado em [7], tem por objetivo a integração de IPs de terceiros aos MPSoCs [4]–[6]. Existem poucas publicações sobre comunicação segura entre aplicações e IPs de terceiros [3]. Este trabalho visa além do desenvolvimento de uma NI com interface padrão, Wishbone [8], prover a camada de software responsável pela comunicação com os periféricos, prover controle de acesso e proteger os dados através de criptografia leve [9].

III. Interface de Rede

Esta Seção apresenta o projeto da NI, modelada em VHDL. Como forma organizar as funções da NI, a mesma foi dividida em blocos, conforme apresentado na Figura 2. Há blocos de controle, um bloco para armazenamento de requisições (CAM), e cripto cores (Simon e SipHash). A escolha destes módulos deu-se pela pequena área de silício necessária para implementá-los.

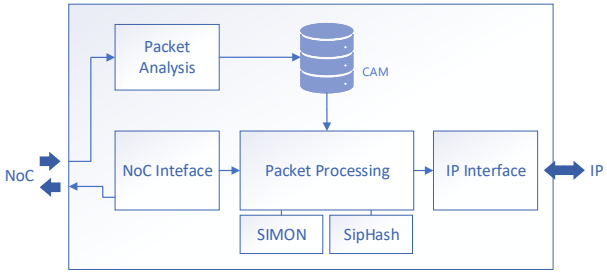

Figura 2. Arquitetura da interface de rede.

O bloco Simon [9] é responsável pela criptografia dos dados que transitam entre os PEs e os periféricos através da NoC. A criptografia dos dados é opcional, e configurável através de um bit no cabeçalho do pacote. As chaves criptográficas são criadas pelos PEs e distribuídas às NIs no momento de inicialização do sistema, podendo ser atualizadas posteriormente através de uma rede de controle dedicada. Observar que o periférico conectado à NI não tem acesso à chave criptográfica, evitando assim a exposição da mesma.

O modulo SipHash [10] garante a integridade e autenticidade dos cabeçalhos dos pacotes através de uma MAC (*Message Authentication Code*).

A. API de Comunicação

O modelo de comunicação é do tipo mestre-escravo, sendo as transações sempre iniciadas pelos PEs. Este modelo impede que periféricos iniciem a comunicação com o MPSoC, ou injetem pacotes na rede, fazendo com que eles se comportem de maneira reativa, respondendo a requisições de leitura e escrita. A seguinte API (*Application Programming Interface*) define os tipos de requisições que são suportados pela NI. O formato dos pacotes é apresentado na Figura 3:

1) Request – Requisição de reserva de periférico;
2) Request ACK – Reserva de periférico aceita;
3) Request NACK – Reserva de periférico recusada;
4) Read request – Requisição de leitura com o endereço alvo e tamanho;
5) Read response – Resposta de leitura com o status e os dados em sequência;
6) Write request – Requisição de escrita com o endereço inicial e dados em sequência;
7) Write response – Resposta de escrita com o status que informa se ela ocorreu ou não.

Services \ Flits	1	2	3	4	5	6	7	8	9
Request	Target	Size	Service	Source ID	Task ID				
Request ACK	Target	Size	Service	Peripheral ID	Task ID				
Request NACK	Target	Size	Service	Peripheral ID	Task ID				
Read Request	Target	Size	Service	Source ID	Task ID	Read Address	Size	hash	
Read Response	Target	Size	Service	Peripheral ID	Task ID	Status	Size	hash	Data
Write Request	Target	Size	Service	Source ID	Task ID	Write Address	Size	hash	Data
Write Response	Target	Size	Service	Peripheral ID	Task ID	Status		hash	

Figura 3. Estrutura dos pacotes relacionados à API de comunicação.

B. Máquinas de estado para o controle da NI

O tratamento de pacotes oriundos da NoC se inicia pelo módulo *Packet Analysis*. Este módulo é responsável por armazenar as requisições em uma memória (CAM – *Content-Addressable Memory*). Esta memória possui dupla porta para

que possam ocorrer operações de escrita e leitura simultaneamente pelas máquinas de estados (FSM). Desta forma as requisições podem ser ordenadas de acordo com um algoritmo de seleção, sendo atualmente por ordem de requisição de forma similar a uma fila do tipo *first-in-first-out*.

O módulo *Packet Analysis* possui a FSM apresentada na Figura 4, sendo responsável por receber e armazenar as requisições de acesso aos periféricos. O estado *Analysing* verifica se CAM está cheia ou não, decidindo assim se a nova requisição será armazenada ou descartada. Caso a requisição seja descartada, o PE solicitante recebe uma mensagem de NACK e pode tentar o acesso novamente no futuro. No estado *Buffering* as requisições são armazenadas na CAM com as seguintes informações: (i) *Task ID* (identificação da tarefa para a requisição); (ii) PE de origem da requisição (solicitante); (iii) periférico de destino (recurso).

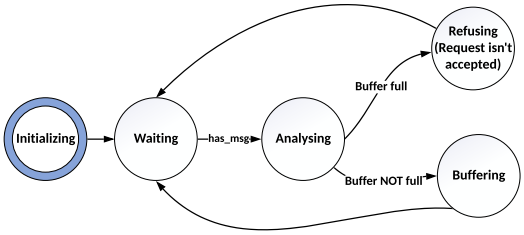

Figura 4. Análise inicial e armazenamento das requisições.

O módulo *Packet Processing* é responsável por processar as requisições armazenadas na CAM. Este módulo possui a FSM apresentada na Figura 5. No estado *Accepting* uma mensagem de ACK é enviada ao PE solicitante, que a partir deste momento é liberado para enviar a sua requisição de leitura ou escrita para o periférico. Os dados recebidos pela NI são analisados no estado *Analysing* para verificar a autenticidade da mensagem transmitida pelo PE, assim como também verificar sua integridade através da verificação da assinatura do cabeçalho (MAC).

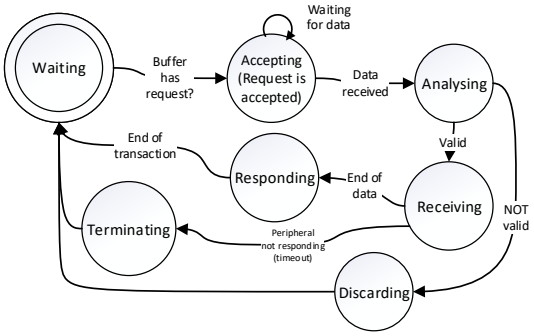

Figura 5. Processamento das requisições de leitura e escrita.

Uma vez comprovada a autenticidade e integridade da requisição, os dados são recebidos no estado *Receiving*, e a comunicação com o periférico é estabelecida. No término da operação uma mensagem de resposta é enviada ao PE solicitante. No caso de operações de escrita essa resposta pode ser de sucesso ou falha, e no caso de operações de leitura esta mensagem pode conter os dados lidos ou então algum código de erro.

O encaminhamento dos dados ao periférico é feito pelo módulo *IP Interface*. Este módulo possui a FSM apresentada na Figura 6, responsável pela interface com o periférico segundo o protocolo Wishbone para operações de leitura e escrita. Os sinais de controle de fluxo da NoC (RX/TX) são sincronizados com os sinais de controle do periférico Wishbone eliminando assim a necessidade de bufferização dos dados.

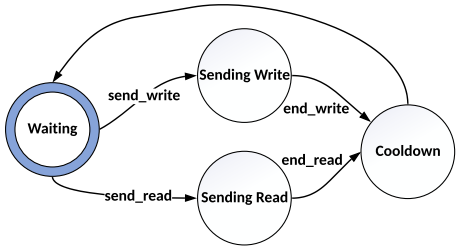

Figura 6. Envio/recepção de dados para periférico.

Esta FSM passa de um estado ocioso para um estado ativo quando *send_write* ou *send_read* é ativado. Estes sinais indicam que o barramento de dados possui um dado válido para ser enviado para o periférico, ou recebido do periférico. Ao escrever no periférico, caso a transmissão de dados da NoC seja mais rápida do que o periférico é capaz de processar, será realizada uma contenção dos dados através do sinal de crédito, e enquanto o dado se mantém válido no barramento à disposição do periférico a FSM permanece no estado de *sending_write*. Ao ler, caso a transmissão de dados do periférico seja mais rápida do que a NoC é capaz de processar será realizada a contenção de dados mantendo o mesmo endereço de leitura para o periférico porém interrompendo a transmissão à NoC. Ao final dos ciclos de leitura ou escrita o estado "cooldown" é atribuído à FSM. Este ciclo encerra a operação desativando os sinais *cyc_o* e *stb_o* do protocolo Wishbone.

O envio de pacotes para a NoC é realizado pela FSM ilustrada na Figura 7. Esta é responsável pelo empacotamento da mensagem e envio para a NoC utilizando o protocolo baseado em créditos. A FSM passa do estado ocioso ("waiting") para um estado ativo quando algum sinal *accepting*, *refusing* ou *responding* é ativado. Esta FSM é responsável pelo empacotamento das respostas que devem ser enviadas da NI para a NoC:

- ACK – mensagem de aceite de utilização do periférico (recurso alocado);
- NACK – mensagem recusando a utilização do recurso;
- Write response – mensagem de sucesso ou falha na operação de escrita realizada
- Read response – mensagem contendo os dados lidos ou código de erro.

IV. Resultados

Esta Seção apresenta o funcionamento da NI através de formas de onda extraídas de simulações com a ferramenta ModelSim. Nos cenários apresentados a seguir (I e II), consideramos um MPSoC [2] de dimensão 3x3 (similar à Figura 1). A requisições são enviadas do PE (0,0) para o periférico conectado na porta leste do PE(2,2). O periférico é uma memória que comunica-se através do protocolo Wishbone.

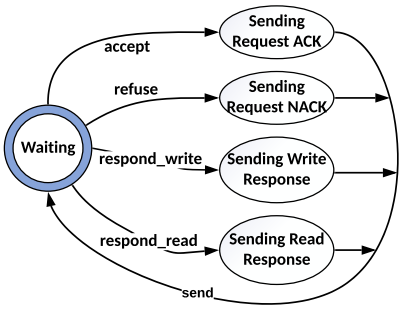

Figura 7. Envio de respostas para a NoC.

Observe que a NoC desconhece o protocolo do periférico, assim como o mesmo desconhece o protocolo interno da NoC. Cabe à NI a tarefa de comunicar as duas partes.

O cenário I consiste em enviar uma solicitação de acesso à memória externa para realização de duas operações em sequência: escrita (*TaskID A*) e leitura (*TaskID B*). A forma de onda da Figura 8 mostra o funcionamento interno do módulo *packet analysis* e as transições de sinais que acontecem ao receber os pacotes oriundos da NoC. No cabeçalho (*header*) é possível identificar o destino da mensagem assim como o seu tamanho. A seguir, ainda no barramento *From NoC* a NI recebe o tipo de operação, neste caso trata-se de uma requisição de acesso. Por fim, a identificação da operação (*TaskID*) é recebida e o conjunto de informações da requisição é armazenado na memória CAM.

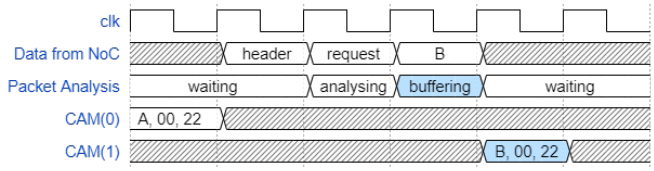

Figura 8. Tratamento da NI à requisições de acesso a periférico

Ainda considerando o cenário I, a Figura 9 mostra o funcionamento do módulo *packet processing* ao aceitar uma requisição de escrita. Esta, transmite inicialmente o seu *header*, seguido do tipo de operação, *TaskID*, endereço de início e assinatura para verificação de integridade. Como uma assinatura válida foi fornecida nesta simulação os dados serão propagados ao periférico. Podemos observar ainda que o módulo *IP interface* atua no controle dos sinais do protocolo Wishbone para comunicação com o periférico através de uma operação do tipo *classic cycle pipelined mode* [8].

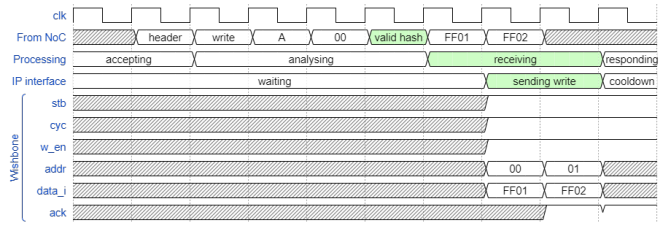

Figura 9. Dados transmitidos da NoC para o Periférico

O cenário II consiste no recebimento de uma requisição na NI que apresenta dois problemas. O primeiro é que a origem não estar autorizada a enviar requisições de leitura e escrita pois ela não recebeu uma mensagem de ACK aprovando, então sua requisição não está armazenada na CAM. O segundo é que o cabeçalho não passou na verificação de assinatura, isto indica que esta mensagem foi adulterada ou criada por uma entidade que não possui a chave simétrica que é utilizada no processo de assinatura SipHash [10]. Como podemos observar na Figura 10 a mensagem é descartada e não é transmitida para o periférico.

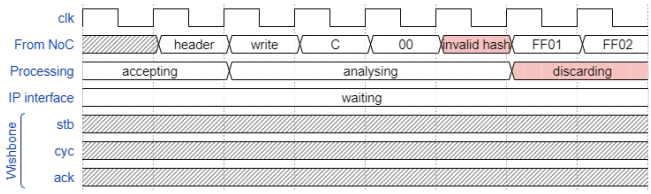

Figura 10. Requisição não autorizada é descartada pela NI.

V. Conclusão

A NI proposta neste artigo permite a integração de IPs de terceiros em uma plataforma MPSoC na forma de periféricos, sendo capaz de atuar no controle de acesso, estabelecer uma ordem justa para o atendimento das requisições e ainda verificar a integridade e autenticidade das mensagens por meio de um módulo de hardware dedicado (SipHash). Observa-se na simulação uma latência de 10 ciclos de *clock* devido as informações adicionadas no cabeçalho, API de requisições e análise da assinatura.

Trabalhos futuros incluem o suporte a operações via DMA (*Direct Memory Access*), que não foram abordadas até o presente momento, a avaliação da latência adicionada pela criptografia leve (apenas testes de corretude foram executados), assim como avaliação de área e potência da NI proposta.

Referências

[1] B. S. Oliveira, R. Reusch, H. M. Medina, and F. Moraes, "Evaluating the cost to cipher the noc communication," in *LASCAS*. IEEE, 2018, pp. 1–4. [Online]. Available: https://doi.org/10.1109/LASCAS.2018.8399914

[2] M. Ruaro, L. L. Caimi, V. Fochi, and F. Moraes, "Memphis: a framework for heterogeneous many-core socs generation and validation," *Design Automation for Embedded Systems*, vol. 23, pp. 103 – 122, 2019.

[3] M. J. Sepúlveda, D. Flórez, V. Immler, G. Gogniat, and G. Sigl, "Efficient security zones implementation through hierarchical group key management at noc-based mpsocs," *Microprocess. Microsystems*, vol. 50, pp. 164–174, 2017. [Online]. Available: https://doi.org/10.1016/j.micpro.2017.03.002

[4] D. R. de Melo, "Interface de Comunicação Extensível para a Rede-em-Chip SOCIN," Master's thesis, Universidade do Vale do Itajaí, 2012.

[5] R. Holsmark, A. Johansson, and S. Kumar, "On Connecting Cores to Packet Switched On-Chip Networks: A Case Study with MicroBlaze Processor Cores," in *DDECS*, 2004, pp. 1–6.

[6] S. P. Singh, S. Bhoj, D. Balasubramanian, T. Nagda, D. Bhatia, and P. Balsara, "Network interface for NoC based architectures," *International Journal of Electronics*, vol. 94, no. 5, pp. 531–547, 2007. [Online]. Available: https://doi.org/10.1080/00207210701306462

[7] B. Aghaei *et al.*, "Network adapter architectures in network on chip: comprehensive literature review," *Clust. Comput.*, vol. 23, no. 1, pp. 321–346, 2019. [Online]. Available: https://doi.org/10.1007/s10586-019-02924-2

[8] OpenCores.org, "WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores - rev. B4 ," 2010. [Online]. Available: https://cdn.opencores.org/downloads/wbspec_b4.pdf

[9] R. Beaulieu *et al.*, "The SIMON and SPECK Lightweight Block Ciphers," in *DAC*, 2015, pp. 175:1–175:6.

[10] J.-P. Aumasson and D. J. Bernstein, " SipHash: a fast short-input PRF," in *INDOCRYPT*, 2012, pp. 489–508.