Ferramenta Interativa para Elaboração de Sistemas Integrados Baseados em Múltiplos Núcleos

Thiago Haas Rausch
Universidade do Vale do Itajaí, Brasil
thiagorausch@edu.univali.br

Wesley Grignani
Université de Montpellier, França
wesley.grignani@umontpellier.fr

Luigi Dilillo
Université de Montpellier, França
luigi.dilillo@umontpellier.fr

Douglas Rossi de Melo
Universidade do Vale do Itajaí, Brasil
drm@univali.br

Abstract

Currently, most computers adopt the System-on-Chip (SoC) approach, integrating various components into a single chip. Traditionally, the interconnection of these components is achieved through buses, which are essential for effective communication. However, as system complexity and performance demands increase, Networks-on-Chip (NoCs) have emerged as a more efficient alternative, offering greater scalability and bandwidth. The integration of SoC components with buses or NoCs remains a challenge, requiring specific adaptations that make the process labor-intensive and prone to errors. This work aims to simplify this integration by developing a graphical interface tool that automates the integration of AXI bus-based components into the XINA NoC. The tool, developed in Python with a Qt interface, was validated using traffic generators and monitors and integrated into the AMD/Xilinx Vivado development environment.

Palavras-chave

Sistemas Integrados, Redes-em-Chip, Núcleos, Integração.

1 Introdução

Atualmente, o desenvolvimento de computadores frequentemente adota a abordagem de Sistema Integrado, ou System-on-Chip (SoC), que consiste na integração de diversos blocos funcionais em um único substrato de silício, proporcionando uma significativa redução de custos de produção [1]. Os SoCs incorporam numerosos blocos de Propriedade Intelectual, Intellectual Property (IP), reutilizáveis, cuja comunicação é tradicionalmente realizada por barramentos. Contudo, com o aumento da complexidade dos sistemas e a demanda por maior desempenho, as redes-em-chip (NoCs) emergem como alternativas mais eficientes, oferecendo maior escalabilidade e largura de banda [2].

A integração de componentes em NoCs apresenta desafios significativos, exigindo adaptações específicas que tornam o processo trabalhoso e suscetível a erros. Historicamente, a abordagem convencional para conectar esses blocos tem sido o uso de barramentos, conforme apontado por [1]. Embora ferramentas como a apresentada por [3] tenham simplificado a conexão de componentes ao barramento AMBA AXI, elas se limitam a essa forma de interconexão, sem oferecer alternativas para a integração com NoCs. Essa limitação destaca a necessidade de pesquisas e desenvolvimentos adicionais para abordar a interconexão de componentes em SoCs que utilizam NoCs.

Para atender a essa necessidade, este trabalho propõe o desenvolvimento de uma ferramenta com interface gráfica que automatiza a integração de componentes encapsulados no barramento AMBA AXI na NoC XINA. A ferramenta, desenvolvida em Python com interface Qt, foi validada por meio de geradores e medidores de tráfego e integrada ao Vivado, permitindo a síntese do projeto de forma eficiente e robusta. A comparação entre a abordagem manual e a ferramenta com interface gráfica mostrou que a ferramenta simplifica significativamente o processo de interconexão, reduzindo o tempo de desenvolvimento e minimizando a ocorrência de erros.

Os resultados demonstram que a ferramenta não introduz custos lógicos adicionais em relação à abordagem manual, enquanto oferece uma integração mais ágil e intuitiva. Ao automatizar a conexão de núcleos encapsulados no barramento AMBA AXI à NoC XINA, a ferramenta aproveita as vantagens das redes-em-chip para sistemas embarcados em ambientes desafiadores.

O artigo está organizado da seguinte forma: A Seção 1 apresentou a introdução, a Seção 2 apresenta a fundamentação teórica, a Seção 3 discute os trabalhos relacionados, a Seção 4 aborda o desenvolvimento da ferramenta proposta, a Seção 5 apresenta os resultados obtidos e a Seção 6 apresenta a conclusão deste trabalho.

2 Fundamentação Teórica

Esta seção apresenta os conceitos fundamentais relacionados a sistemas integrados, sistemas de interconexão, barramentos e redes-em-chip, estabelecendo a base teórica para a pesquisa desenvolvida.

Os sistemas integrados, predominantemente implementados por meio de System-on-Chip (SoC), integram processadores, caches, memória e dispositivos de entrada/saída em um único substrato de silício, proporcionando vantagens econômicas e de eficiência. As implementações mais comuns utilizam abordagens semi-custom com bibliotecas de células padrão ou FPGAs, sendo estes últimos adequados para produções de média escala devido ao alto custo de fabricação de hardware codificado permanentemente [1].

O design de SoCs é facilitado pelo uso de blocos de propriedade intelectual (IP) e ferramentas de descrição de hardware como Verilog e VHDL, que aumentam a eficiência do projeto ao abstrair a complexidade. Além disso, os Multiprocessor Systems-on-Chip (MPSoCs) incorporam múltiplos processadores independentes, melhorando o desempenho através da execução paralela de tarefas, diferenciando-se dos SoCs convencionais [1].

Os sistemas de interconexão são essenciais para a comunicação entre os diversos blocos de IP em um SoC, facilitando a troca de

comandos e respostas entre iniciadores e alvos, geralmente processadores e periféricos [1, 4]. Tradicionalmente, barramentos são utilizados para essa finalidade, mas alternativas como redes-em-chip (NoC) estão ganhando destaque devido à crescente complexidade dos SoCs e à necessidade de soluções mais eficientes. Ferramentas geradoras de interconexão padronizam o encapsulamento de IPs e a geração de interconexões, garantindo a integração eficiente dos componentes e a padronização necessária para a produção [1].

Os barramentos são a forma mais comum de interconexão em SoCs devido à sua simplicidade e capacidade de automação através do encapsulamento de IPs [1]. Eles suportam diferentes modos de transmissão, como simplex, half-duplex e full-duplex, além de conexões de streaming, tornando-os versáteis para diversas aplicações [1]. O AMBA (Advanced Microcontroller Bus Architecture) é o barramento mais amplamente adotado tanto na academia quanto na indústria, oferecendo interfaces flexíveis como AXI, AXI-Lite e AXI-Stream, que facilitam a comunicação eficiente entre componentes de hardware em SoCs [5–7].

Redes-em-chip (NoC) emergem como uma alternativa eficiente aos barramentos tradicionais, especialmente em SoCs com grande número de núcleos, onde a capacitância parasita dos barramentos pode limitar o desempenho do sistema [2, 8]. As NoCs interligam núcleos por meio de roteadores e canais ponto-a-ponto, sendo caracterizadas por atributos como topologia, controle de fluxo, roteamento, arbitragem, memorização e chaveamento [8]. Exemplos de NoCs incluem a SoCIN, uma rede altamente parametrizável utilizada em diversos estudos para qualidade de serviço e redução de consumo energético [9–13], e a XINA, uma NoC específica para aplicações espaciais que incorpora tolerância a falhas e controle parametrizável, atendendo às exigências de ambientes hostis [14–17].

3 Trabalhos Relacionados

Esta seção apresenta ferramentas para a geração de SoCs que operam de forma similar ao objetivo deste projeto, destacando suas principais características e contribuições.

O GRLIB [18] é uma biblioteca de propriedade intelectual (IP) desenvolvida para facilitar o projeto de SoCs, oferecendo uma ampla variedade de IPs reutilizáveis como processadores, controladores de periféricos e blocos de comunicação. A abordagem modular e configurável do GRLIB agiliza o desenvolvimento, permitindo a integração de componentes pré-validados e adaptáveis conforme necessário. Dentro do GRLIB, o processador LEON, desenvolvido pela ESA, e o NOEL-V, um processador RISC-V de código aberto, exemplificam a flexibilidade da biblioteca em atender a diferentes requisitos de desempenho e aplicação [18].

O Rocket Chip Generator [19] é uma ferramenta de geração de processadores para SoCs integrada aos projetos do Chips Alliance, baseada na arquitetura RISC-V de código aberto. Oferece uma abordagem modular e configurável, permitindo aos projetistas personalizar processadores selecionando módulos como caches, unidades de execução e periféricos. Após a configuração, a ferramenta gera automaticamente o design completo do processador em VHDL ou Verilog, facilitando a síntese e implementação em dispositivos FPGA ou ASIC [19].

O PULPissimo [20], uma versão atualizada do PULPINO, é um projeto de SoC open source desenvolvido pela Universidade de Zurique dentro do grupo PULP (Parallel Ultra-Low-Power). Baseado na arquitetura RISC-V, o PULPissimo foca em soluções de baixo consumo de energia para sistemas embarcados, destacando-se por sua alta modularidade que permite adaptações significativas para diferentes dispositivos [20].

HeMPS (Hermes Multiprocessor Systems) [21] é um MPSoC baseado em redes-em-chip (NoC) com topologia 2D-mesh, interconectando Processing Elements (PEs) compostos por processadores RISC, interfaces de rede de memória direta (DMNI) e memórias privadas. As aplicações são modeladas como conjuntos de tarefas comunicantes, utilizando passagem de mensagens para comunicação. HeMPS utiliza SystemC ou VHDL para descrição de hardware e C para software, oferecendo ferramentas para depuração de mapeamento, agendamento de aplicativos e tráfego na NoC [21].

A Plataforma de Integração de Componentes para Sistemas Embarcados em Aplicações Espaciais [3] permite a configuração e prototipação de SoCs para aplicações espaciais, oferecendo opções de personalização para processadores com técnicas de tolerância a falhas e para o barramento AMBA AXI4-Lite. A integração de aceleradores externos e a geração automática do arquivo VHDL do sistema são facilitadas por scripts Python e ferramentas como o AMD Vivado. A interface gráfica da plataforma permite configurar processador HARV [22], barramentos, softwares/aplicações e periféricos, além de suportar a incorporação de novos IPs.

Este trabalho distingue-se ao utilizar a Rede-em-Chip XINA para integrar núcleos no barramento AMBA AXI, incorporando técnicas de tolerância a falhas e permitindo uma configuração mais robusta. A integração via interface de rede desenvolvida para a XINA facilita a conexão de geradores e medidores de tráfego, alinhando-se às necessidades específicas de sistemas embarcados em ambientes desafiadores.

4 Desenvolvimento

A ferramenta desenvolvida possibilita a integração de componentes previamente descritos. Nesse contexto, a NoC XINA e sua interface de rede já estabelecida foram utilizadas, bem como geradores e medidores de tráfego encapsulados em AMBA AXI para validação, garantindo que a solução atenda às necessidades de interconexão e comunicação entre os componentes.

Simultaneamente, a ferramenta integra múltiplas tecnologias em um único fluxo de projeto, oferecendo uma interface gráfica desenvolvida em PyQt6 sobre Python 3.10, scripts bash no ambiente Linux (Ubuntu 24.04 LTS) e suporte à síntese na ferramenta AMD Vivado 2024.1 (FPGA Zynq-7000). Por meio de TCL, a solução automatiza a geração de um SoC completo a partir de componentes catalogados, empregando o barramento AMBA AXI5-Full, a NoC XINA e suas interfaces de rede. Dessa forma, o projeto é gerado automaticamente em VHDL, com visualização das conexões, reduzindo o esforço manual e automatizando etapas críticas do fluxo.

A Figura 1 ilustra uma visão comparativa da ferramenta proposta, destacando as etapas que devem ser realizadas manualmente em vermelho, as etapas comuns em amarelo e as etapas automatizadas com o uso da ferramenta proposta em verde. A ferramenta permite a configuração de um SoC completo utilizando componentes previamente registrados por meio de interfaces de configuração.

Após a definição das configurações, a ferramenta gera automaticamente um arquivo topo em linguagem de descrição de hardware, integrando os componentes selecionados. Além disso, fornece uma representação visual das conexões no barramento, incluindo as interfaces de rede conectadas aos roteadores da NoC.

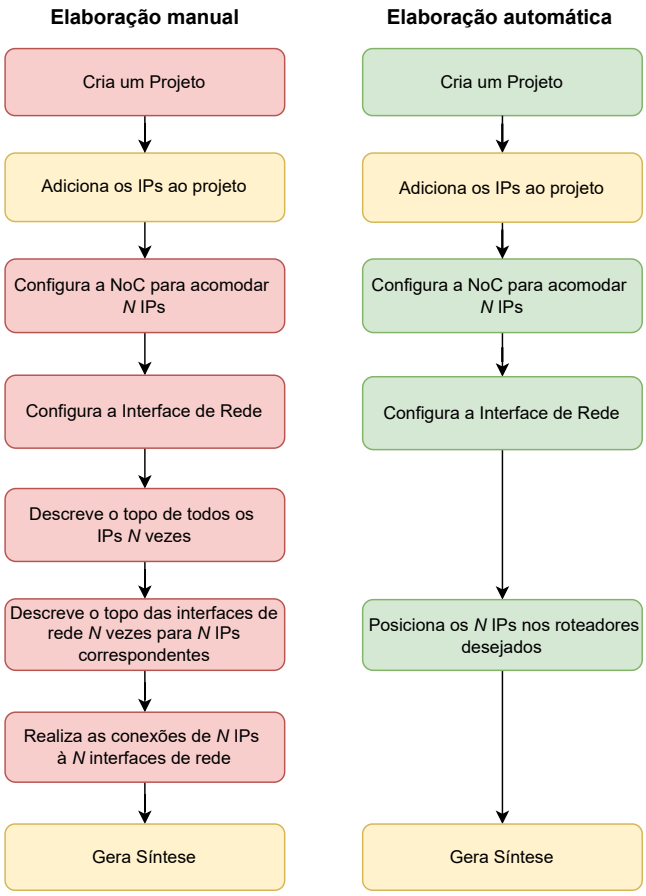

Figura 1: Fluxo de elaboração de um SoC

4.1 Elaboração manual de um SoC

A elaboração manual teve como objetivo demonstrar o processo manual de criação de um SoC utilizando NoC, destacando a complexidade e o tempo necessário para executar cada etapa de forma individual. Esse método enfatizou o quão laboriosa é a atividade, especialmente ao configurar e integrar componentes sem o suporte de automações, evidenciando a importância de uma abordagem automatizada para aumentar a eficiência e reduzir a probabilidade de erros.

A Figura 2 ilustra como pode ser feita a elaboração manual de um SoC. Em verde estão as conexões dos IPs, em marrom os sinais das interfaces de rede e em azul as conexões da XINA. Neste exemplo, é apresentado uma NoC 2×2 utilizando dois IPs gerentes e dois IPs subordinados, interconectados por 4 roteadores XINA por suas respectivas interfaces de rede.

4.2 Elaboração automática de um SoC

A Figura 3 apresenta a visão geral de um SoC gerado utilizando a ferramenta. Em verde estão as conexões dos IPs, em marrom as interfaces de rede, em azul as conexões da XINA e em laranja o bloco de interconexão. Esta abordagem proporciona maior abstração no processo de criação do SoC, reduzindo o número de etapas necessárias para conectar seus componentes. Inicialmente, o usuário descreve os IPs gerentes e subordinados em um template reutilizável e os salva como arquivos de texto. Em seguida, declara-se qual IP será conectado em cada posição. A ferramenta analisa esses arquivos de configuração e gera automaticamente o módulo topo, com todas as conexões realizadas.

4.3 Implementação da interface gráfica

A topologia do SoC em desenvolvimento é sempre exibida à direita da aba de configuração que está aberta e é atualizada dinamicamente conforme o usuário ajusta as configurações do SoC. Assim, qualquer alteração no tamanho do SoC ou nas conexões dos IPs provoca uma atualização automática da representação da topologia. Essa representação é gerada como um grafo interativo utilizando a biblioteca Matplotlib do Python, onde cada quadrado representa um roteador, conectado ou não a um IP. Todos os roteadores são interativos, permitindo que o usuário clique em um ou mais roteadores para editar os IPs aos quais estão conectados, seja individualmente ou em lote.

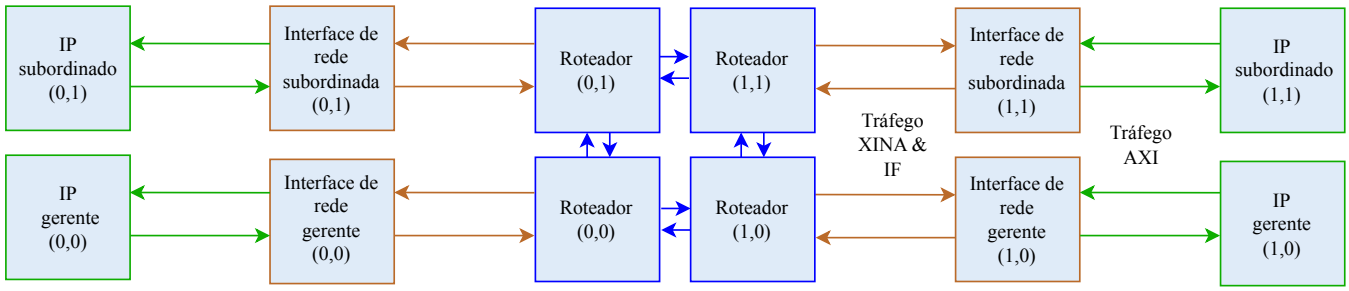

Figura 2: Visão geral de um SoC 2×2 elaborado manualmente

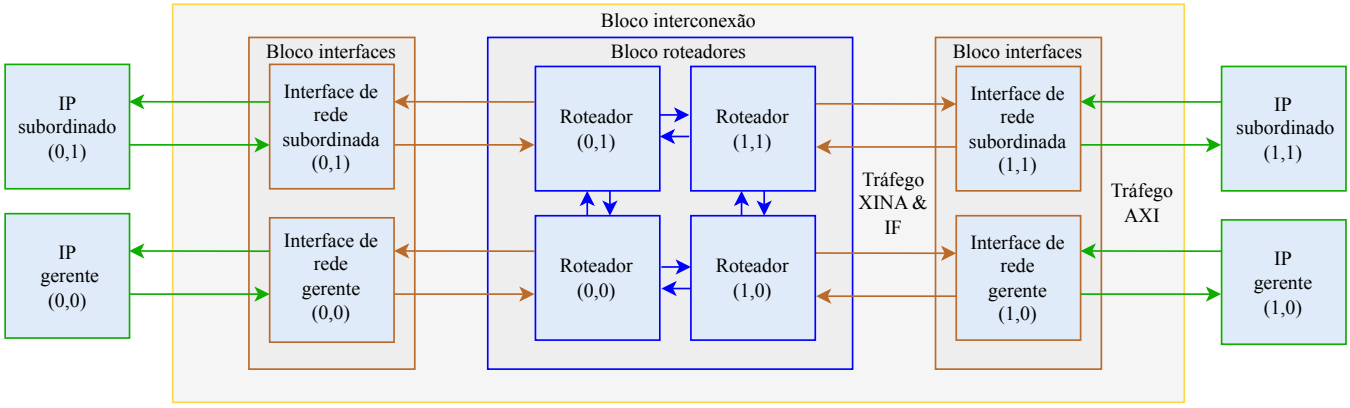

Figura 3: Visão geral de um SoC 2×2 elaborado automaticamente

A primeira parte da configuração é a configuração da **XINA**, a Figura 4 demonstra a interface na aba da XINA. Nela é possível configurar os parâmetros de tolerância a falhas, largura dos buffers, tamanho da NoC e largura de dados. Caso configure uma NoC maior ou menor, o grafo representativo do SoC se ajusta para comportar os roteadores.

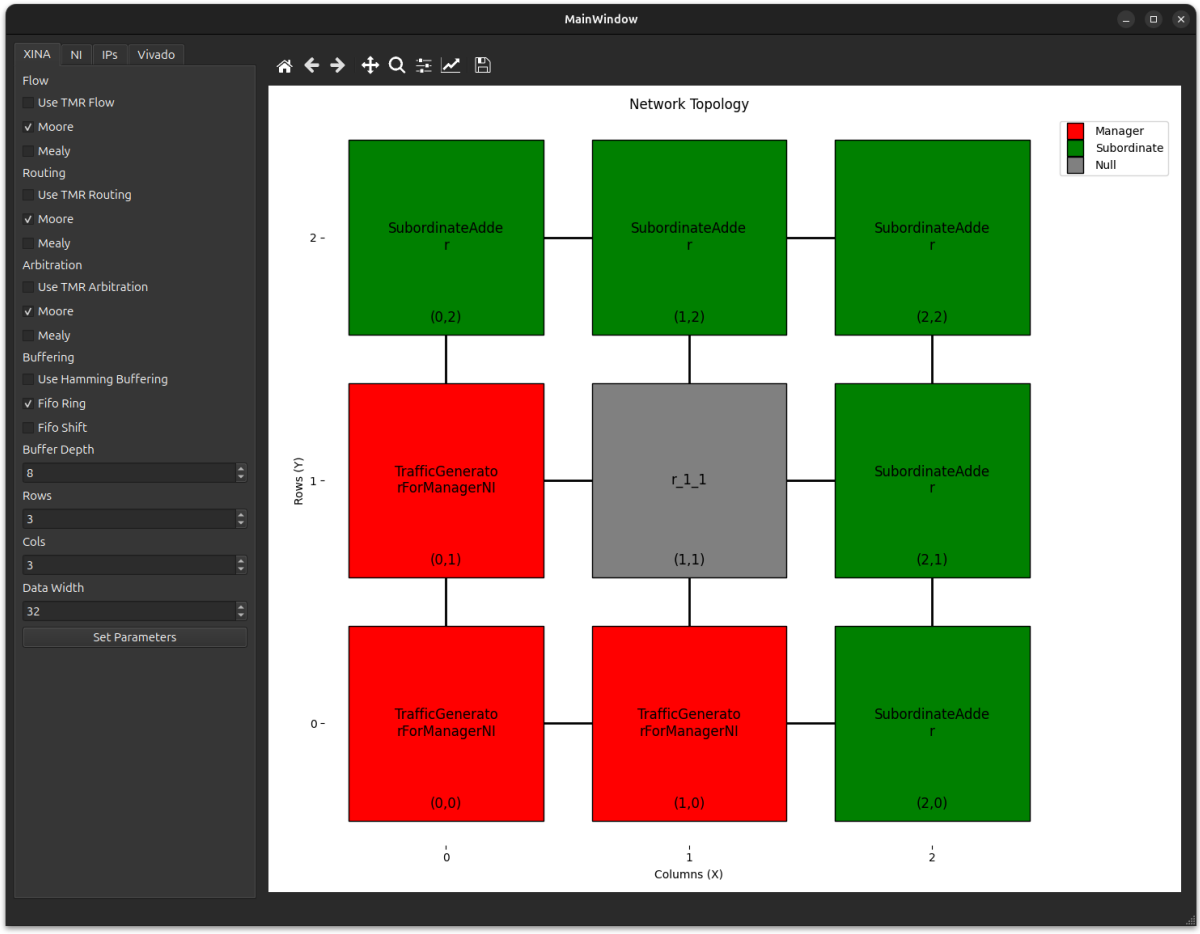

Figura 4: Aba de configuração XINA

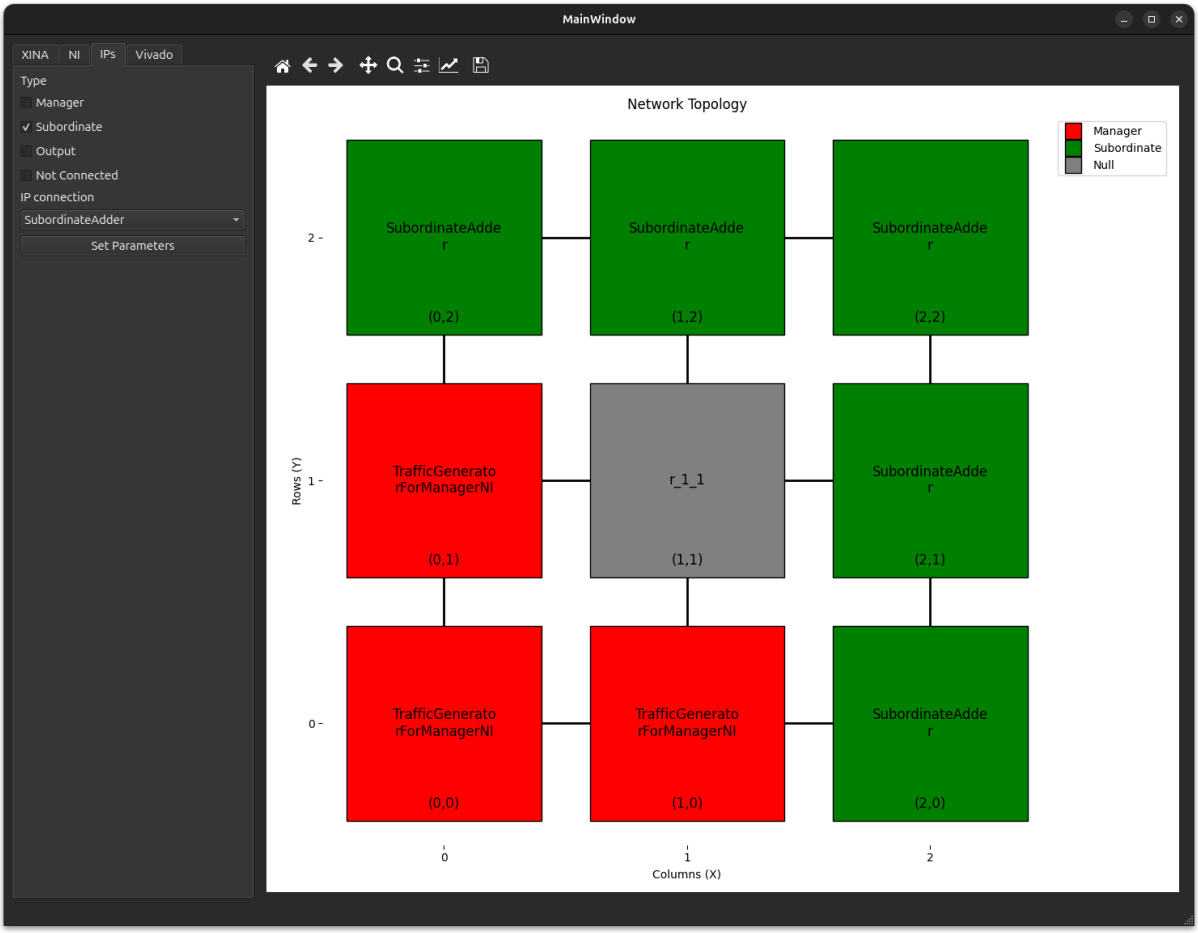

Figura 5: Aba de configuração dos IPs

A segunda etapa da configuração refere-se à configuração da **NI**. Nessa seção, o usuário pode ajustar parâmetros como a tolerância a falhas, a largura dos buffers e os parâmetros de largura dos sinais AXI. Em relação à largura de dados, à largura do ID AXI e à largura do endereçamento AXI, esses valores devem ser mantidos em 32 bits, 5 bits e 32 bits, respectivamente, devido às limitações na parametrização da própria interface de rede.

A terceira etapa da configuração, ilustrada na Figura 5, envolve a adição de **IP**s ao SoC. Nesta etapa, é possível configurar três tipos distintos de conexão: não conectado, conectado a um IP gerente ou conectado a um IP subordinado. Para configurar qualquer uma dessas opções, o usuário precisa apenas selecionar os roteadores desejados, escolher o tipo de conexão e adicionar o IP correspondente a partir da lista de disponíveis. Para a inclusão de novos IPs para utilização com a ferramenta, é necessário criar um template de conexão específico para o IP desejado, garantindo sua compatibilidade com a ferramenta.

A quarta etapa da configuração é opcional e trata da integração com o **Vivado**. Nessa etapa, o usuário pode especificar o nome do projeto e o dispositivo FPGA alvo, com duas opções disponíveis: "Gerar Projeto" e "Rodar Síntese". Ao selecionar "Gerar Projeto", o programa executa um script TCL que cria um projeto Vivado com o SoC configurado, copiando todos os arquivos VHDL para garantir portabilidade fora da ferramenta. A opção "Rodar Síntese" executa a síntese do arquivo de nível superior do projeto e verifica sua viabilidade. Como essa etapa também utiliza um script TCL, o progresso e os resultados da síntese são exibidos diretamente no terminal. Por fim, é possível abrir o projeto no Vivado com a opção "Abrir Projeto" que executa uma chamada para abrir o Vivado no projeto escolhido. Com o projeto aberto, é possível executar outras etapas como simulação e escrita no FPGA por exemplo.

5 Resultados

Os resultados acompanham a mesma estrutura do desenvolvimento, ocorrendo em duas etapas interdependentes: inicialmente, são apresentados os resultados do SoC desenvolvido manualmente, seguido dos resultados da ferramenta com interface gráfica.

A ferramenta com interface gráfica, desenvolvida em Python e Qt, possui um tamanho total de 92,56 KB, excluindo as dependências. Durante a elaboração do SoC descrito na Seção 4.2, a ferramenta apresentou um consumo máximo de memória de 129,77 MB de RAM considerando o comando `/usr/bin/time -v bash`.

5.1 Síntese

Nesta seção, são apresentados os resultados da síntese do SoC realizados por meio de dois métodos distintos: a elaboração manual e a elaboração automática utilizando a ferramenta desenvolvida. Em ambos os casos, o SoC consiste em 2 IPs geradores de tráfego gerente, 2 IPs medidores de tráfego subordinado, suas respectivas interfaces de rede e 4 roteadores XINA em uma malha de tamanho 2×2. A comparação entre esses métodos permite avaliar a eficácia da ferramenta e verificar se o SoC gerado automaticamente é equivalente ao construído manualmente em termos de estrutura.

A Figura 6 apresenta o resultado de síntese em RTL no Vivado 2024.1 de um SoC construído manualmente seguindo o diagrama da Figura 2. As cores dos fios do diagrama RTL seguem a seguinte lógica: verde para as conexões dos IPs, marrom para os sinais das interfaces de rede, azul para as conexões da XINA e magenta para os sinais de clock e reset.

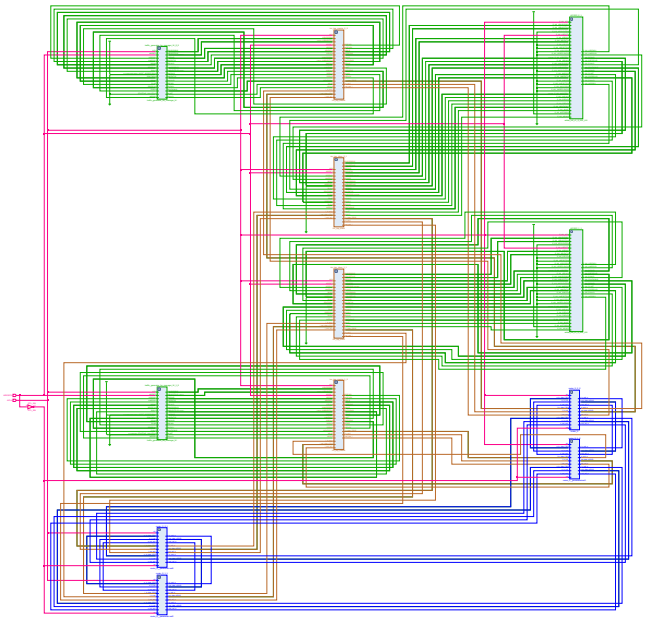

Figura 6: Diagrama RTL do SoC 2×2 elaborado manualmente

A Figura 7 apresenta o resultado de síntese em RTL do SoC construído com a ferramenta seguindo o diagrama da Figura 3. Esse SoC é equivalente ao resultado manual.

A Figura 8 mostra o bloco superior onde as conexões AXI entram no sistema. Essas conexões se conectam ao bloco de interfaces de rede, iniciando o processo de conversão do tráfego.

Na Figura 9, é detalhado o bloco de interfaces de rede. Nesse bloco, o tráfego AXI é interpretado e convertido em tráfego compatível com a Rede-em-Chip XINA, permitindo a comunicação eficiente entre os núcleos através da rede.

A Figura 10 apresenta o bloco dos roteadores XINA. Aqui, o tráfego convertido é roteado através da Rede-em-Chip, permitindo a comunicação entre os diferentes IPs do sistema. Após o roteamento, o tráfego retorna ao bloco de interfaces de rede para ser reconvertido em tráfego AXI e entregue aos IPs de destino.

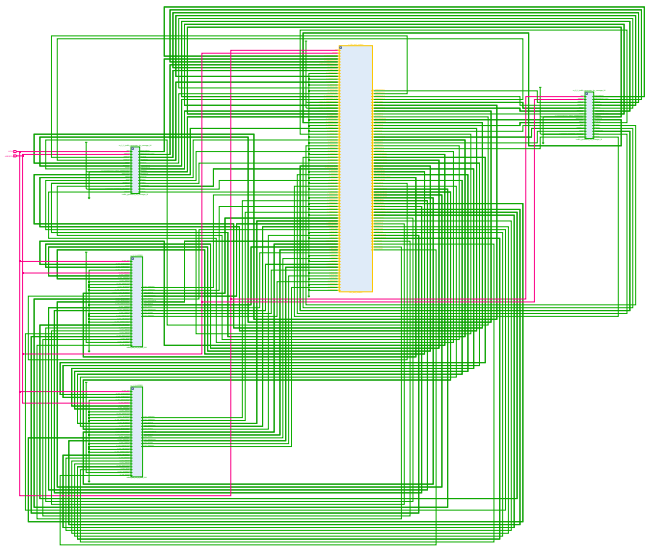

Figura 7: Diagrama RTL do SoC 2×2 elaborado automaticamente

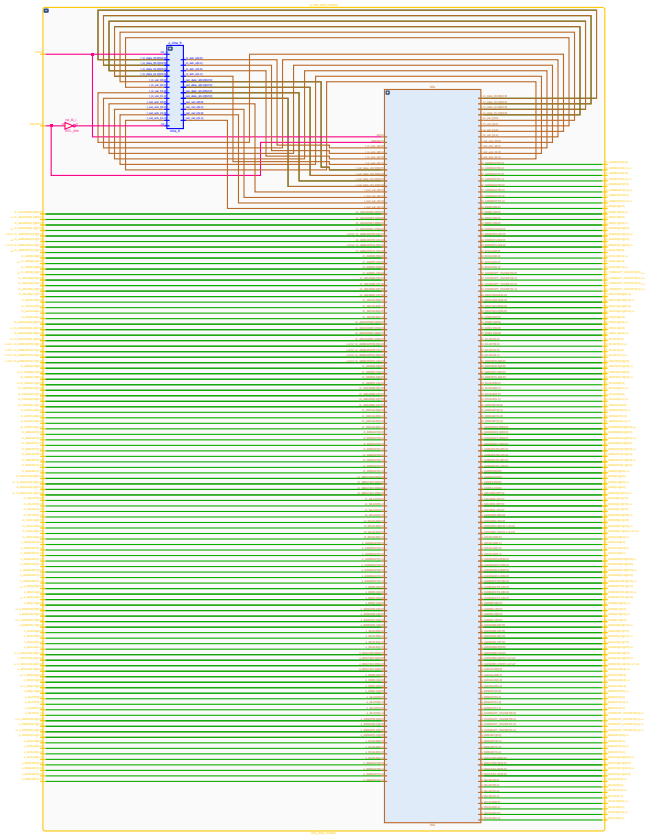

Figura 8: Diagrama RTL do bloco de interconexão

Enquanto na versão manual todos os componentes estão diretamente no nível superior, resultando em pouca abstração, na versão gerada pela ferramenta o RTL é simplificado ao agrupar toda a interconexão em blocos específicos. Isso aumenta a modularidade e facilita a compreensão do SoC.

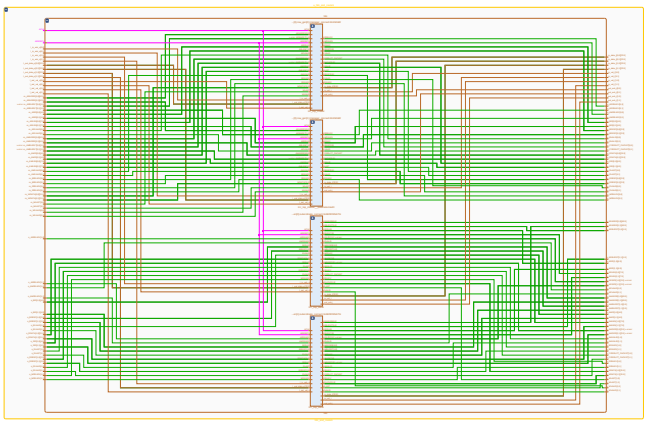

Figura 9: Diagrama RTL do bloco de interfaces de rede

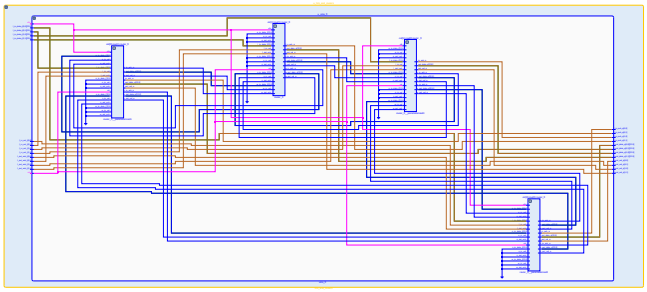

Figura 10: Diagrama RTL do bloco de roteadores

5.2 Custo e desempenho

A síntese dos circuitos comparou as abordagens manual e automatizada para a implementação no dispositivo FPGA Zynq-7020. A análise dos resultados mostrou que ambas as versões apresentaram exatamente a mesma utilização de recursos: 3.301 LUTs (Look-Up Tables), 987 FFs (Flip-Flops) e 842 LUTRAMs (Look-Up Tables RAMs). Além disso, a frequência máxima alcançada em ambas as abordagens foi de 184,36 MHz. Esses resultados indicam que a ferramenta automatizada gera um hardware equivalente ao da versão manual, com a diferença principal estando na organização hierárquica do projeto.

A validação por simulação do SoC foi realizada utilizando uma transação de escrita com retorno, na qual o gerente envia um dado ao subordinado, que realiza uma operação de soma e devolve o resultado. A simulação foi conduzida sobre o hardware construído com base no barramento AMBA AXI e na Rede-em-Chip XINA. Durante a simulação, os sinais monitorados confirmaram o funcionamento correto da comunicação, demonstrando que o dado enviado pelo gerente foi processado adequadamente pelo subordinado, e que o resultado foi retornado conforme esperado.

5.3 Usabilidade

A ferramenta desenvolvida simplifica a geração de SoCs, reduzindo o tempo de desenvolvimento e depuração. No processo manual, a declaração de sinais exige cerca de 29 linhas de código e a instanciação de IPs aproximadamente 108 linhas, totalizando cerca de 8 minutos de trabalho, mesmo para um desenvolvedor experiente. Em contraste, a solução automatizada elimina a necessidade de escrever código para essas etapas, requerendo cerca de 5 cliques por IP e reduzindo o tempo em aproximadamente 16×.

Além da redução do tempo, a ferramenta também minimiza a chance de erros, pois elimina a necessidade de lidar diretamente com a sintaxe do código, evitando equívocos manuais comuns. Além disso, a ferramenta permite a inclusão em lote de múltiplos IPs idênticos, resultando em menos cliques e maior eficiência no fluxo de projeto.

6 Conclusão

Foi desenvolvida uma ferramenta em Python que automatiza a criação e integração de Sistemas em Chip (SoCs), combinando núcleos do barramento AMBA AXI com a Rede-em-Chip XINA. A ferramenta simplifica significativamente o processo de interconexão, permitindo a geração e síntese de projetos na ferramenta Vivado. Os resultados obtidos demonstraram que o custo lógico da ferramenta é equivalente ao da criação manual, sem comprometer o desempenho ou aumentar o uso de recursos, evidenciando sua eficácia e viabilidade para o desenvolvimento de sistemas integrados complexos.

Além disso, foi realizada uma análise comparativa com trabalhos similares, aproveitando plataformas e ferramentas existentes como base para a solução proposta. Este estudo permitiu identificar e implementar melhorias significativas, culminando na criação de uma ferramenta interativa que facilita a integração de múltiplos componentes tecnológicos em SoCs.

Para trabalhos futuros, planeja-se aumentar a compatibilidade da ferramenta com outros núcleos desenvolvidos em pesquisas anteriores, como processadores e aceleradores. Também está prevista a expansão da parametrização da interface de rede para incluir suporte ao protocolo AMBA AXI-Stream, o que permitirá a integração de uma gama mais ampla de núcleos.

7 Agradecimentos

Este trabalho foi financiado, em parte, pela Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina – FAPESC (contratos 2023TR000880 e 2024TR001897), pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq (processos 313513/2021-0, 350208/2022-0, 408641/2023-1 e 350794/2023-5), pelo Projeto EU RADNEXT – Horizon 2020 (contrato 101008126) e pela École Doctorale I2S e o Projeto HARV na ação Accelerateur d'innovation da Université de Montpellier.

Referências

[1] D. Greaves. *Modern System-on-Chip Design on Arm.* ARM Education Media, 2021. ISBN 9781911531364. URL https://www.arm.com/resources/ebook/modern-soc.

[2] L Tedesco. Uma proposta para geração de tráfego e avaliação de desempenho para NoCs. Master's thesis, Pontificia Universidade Catolica Do Rio Grande Do Sul: Porto Alegre, 2005.

[3] Carlos Natham Domingos, Douglas Almeida Santos, Wesley Grignani, and Douglas Rossi Melo. Plataforma de integração de componentes para sistemas embarcados em aplicações espaciais. *Anais do Computer on the Beach*, 14:444–446, 2023.

[4] Frank Vahid and Tony Givargis. *Embedded System Design: A Unified Hardware/Software Introduction.* John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2001. ISBN 0471386782.

[5] ARM. AMBA Specification, 2024. Disponível em: https://developer.arm.com/Architectures/AMBA.

[6] ARM. AMBA AXI Protocol Specification, 2024. Disponível em: https://developer.arm.com/documentation/ihi0022/latest.

[7] ARM. AMBA AXI-Stream Protocol Specification, 2024. Disponível em: https://developer.arm.com/documentation/ihi0051/latest/.

[8] Cesar Albenes Zeferino and Altamiro Amadeu Susin. SoCIN: a parametric and scalable network-on-chip. In *Proceedings of the 16th symposium on integrated circuits and system design*, pages 169–174. IEEE, 2003.

[9] Cesar Albenes Zeferino, Frederico GME Santo, and Altamiro Amadeu Susin. Paris: a parameterizable interconnect switch for networks-on-chip. In *Proceedings of the 17th symposium on integrated circuits and system design*, pages 204–209, 2004.

[10] Marcelo Daniel Berejuck and Cesar Albenes Zeferino. Adding mechanisms for QoS to a network-on-chip. In *Proceedings of the 22nd Annual Symposium on Integrated Circuits and System Design: Chip on the Dunes*, pages 1–6, 2009.

[11] Sidnei Baron, Michelle Silva Wangham, and Cesar Albenes Zeferino. Security mechanisms to improve the availability of a network-on-chip. In *2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS)*, pages 609–612. IEEE, 2013.

[12] Cesar Albenes Zeferino, Jaison Valmor Bruch, Thiago Felski Pereira, Márcio Eduardo Kreutz, and Altamiro Amadeu Susin. Avaliação de desempenho de rede-em-chip modelada em systemc. In *Proceedings of the 27rd Congress of Brazilian Computer Society-WPerformance*, pages 559–578, 2007.

[13] Jaison Valmor Bruch and Cesar Albenes Zeferino. Evaluation of architectural alternatives to reduce power consumption in a network-on-chip. In *2nd Workshop on Circuits and Systems Design (WCAS 2012)*, 2012.

[14] Douglas Rossi Melo, Cesar Albenes Zeferino, Luigi DiLillo, and Eduardo Bezerra. Maximizing the inner resilience of a network-on-chip through router controllers design. *Sensors*, 2019.

[15] Douglas Rossi Melo. *Interconnection Architecture for Dependable Multi-core Systems.* PhD thesis, Universidade Federal de Santa Catarina, 2020.

[16] Douglas Rossi Melo, Cesar Albenes Zeferino, Eduardo Augusto Bezerra, and Luigi Dilillo. Design and evaluation of implementation impact on a fault-tolerant network-on-chip router. In *2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1–6. IEEE, 2021.

[17] Gustavo S Mafra, Thiago H Rausch, Douglas Almeida Santos, Luigi Dilillo, Eduardo Augusto Bezerra, and Douglas Rossi Melo. Assessing the reliability of a network-on-chip through physical validation. In *LASSS/LACW 2022-Joint 3rd IAA Latin American Symposium on Small Satellites and 5th IAA Latin American CubeSat Workshop*, 2022.

[18] Frontgrade Gaisler. GRLIB IP Library User's Manual, 2024. Disponível em: https://www.gaisler.com/.

[19] Krste ASANOVIĆ, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, Sagar Karandikar, Ben Keller, Donggyu Kim, John Koenig, Yunsup Lee, Eric Love, Martin Maas, Albert Magyar, Howard Mao, Miquel Moreto, Albert Ou, David A. Patterson, Brian Richards, Colin Schmidt, Stephen Twigg, Huy Vo, and Andrew Waterman. The rocket chip generator. Technical Report UCB/EECS-2016-17, EECS Department, University of California, Berkeley, Apr 2016. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html.

[20] PULP. Pulpissimo, 2021. Disponível em: https://github.com/pulp-platform/pulpissimo.

[21] Gaph. HeMPS, 2016. Disponível em: https://www.inf.pucrs.br/hemps/index.html.

[22] Douglas A Santos, Lucas M Luza, Maria Kastriotou, Carlo Cazzaniga, Cesar A Zeferino, Douglas R Melo, and Luigi Dilillo. Characterization of a risc-v system-on-chip under neutron radiation. In *2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1–6. IEEE, 2021.