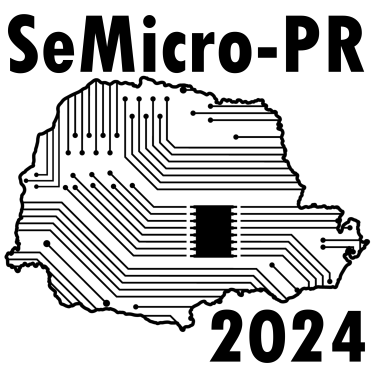

Ferramenta para Interconexão de Múltiplos Núcleos em Sistemas Integrados

Thiago H. Rausch, Eduardo M. D. Souza,
Wesley Grignani, Douglas R. Melo
LEDS, Universidade do Vale do Itajaí, Itajaí, Brasil
{thiagorausch, eduardomichel, wesley.grignani}@edu.univali.br, drm@univali.br

Resumo - Com o avanço da complexidade nos sistemas integrados (Systems-on-Chip - SoCs), integrar componentes utilizando redes-em-chip (Networks-on-Chip - NoCs) tornou-se um desafio significativo. A integração manual de núcleos gerentes e subordinados com NoCs é um processo complexo e propenso a erros. Para abordar esse problema, este trabalho propõe uma ferramenta que simplifica o processo de interconexão, automatizando a geração do módulo topo e das conexões e abstraindo detalhes complexos da integração. Foram desenvolvidas duas versões de um SoC, uma construída manualmente e outra empregando a ferramenta proposta. Os resultados obtidos demonstram que a ferramenta permite a obtenção de SoCs equivalentes com menor esforço manual e sem aumento no custo lógico.

I. INTRODUÇÃO

Atualmente, a construção de computadores frequentemente adota a abordagem de sistemas integrados (Systems-on-Chip - SoCs), que consiste na integração de diversos componentes de Propriedade Intelectual (Intellectual Property - IP), também conhecidos como núcleos, em um único substrato de silício. Para que esses IPs se comuniquem de maneira eficaz, é necessário um sistema de interconexão, historicamente realizado por barramentos [1].

Uma alternativa aos barramentos são as redes-em-chip (Networks-on-Chip - NoCs), que utilizam roteadores interligados para permitir a comunicação entre os núcleos de um SoC [2]. Contudo, conectar manualmente componentes a esses sistemas é uma tarefa trabalhosa que requer o conhecimento da arquitetura de interconexão utilizada. Além disso, a complexidade de integração aumenta à medida que mais componentes são interconectados.

Recentemente, [3] propuseram uma ferramenta para simplificar a conexão de componentes ao barramento Advanced Microcontroller Bus Architecture (AMBA), mas essa abordagem não contempla redes-em-chip. Dessa forma, este trabalho propõe o desenvolvimento de uma ferramenta que permita a integração de componentes com NoCs, oferecendo uma solução mais robusta para a interconexão em SoCs.

O restante do artigo está organizado da seguinte forma: a Seção II apresenta o referencial teórico, a Seção III discute os trabalhos relacionados, a Seção IV aborda o desenvolvimento da ferramenta proposta, a Seção V apresenta os resultados obtidos e a Seção VI apresenta a conclusão deste trabalho.

II. FUNDAMENTAÇÃO TEÓRICA

Os SoCs integram processadores, memórias, dispositivos de entrada/saída e outros componentes em um único substrato de silício, otimizando custos de produção e aumentando a eficiência do sistema. Para que os diversos componentes de IP em um SoC comuniquem-se efetivamente, é essencial um sistema de interconexão eficiente, responsável por transmitir comandos e dados entre iniciadores (gerentes) e alvos (subordinados). Os barramentos são a forma tradicional de interconexão em SoCs, proporcionando movimento de dados, endereçamento e controle de fluxo entre os componentes. Devido à sua simplicidade, são uma escolha viável para projetos com restrições de tempo e custo [1].

O AMBA, desenvolvido pela ARM, é a arquitetura de barramento mais adotada, fornecendo interfaces padronizadas como AXI, AXI-Lite e AXI-Stream para comunicação entre componentes de hardware em um SoC [4]. O protocolo AXI (Advanced eXtensible Interface) é fundamental para a comunicação de dados entre gerentes e subordinados, suportando canais distintos para solicitações de leitura e escrita, facilitando a integração de dispositivos [5].

Com o aumento do número de núcleos em SoCs, os barramentos podem se tornar um gargalo. Como alternativa, as NoCs surgem como uma solução para interconectar múltiplos núcleos, utilizando roteadores e canais ponto a ponto. As NoCs são caracterizadas por atributos como topologia, controle de fluxo, roteamento, arbitragem, memorização e chaveamento, que determinam a organização e o funcionamento da rede-em-chip [6].

A XINA é uma NoC desenvolvida para aplicações em ambientes espaciais, onde a tolerância a falhas é crucial. A XINA oferece parametrização de controladores entre os modelos de Moore e Mealy e implementa técnicas de detecção e correção de falhas, como TMR (Triple Modular Redundancy) e Hamming ECC (Error Correcting Code), aumentando a confiabilidade em ambientes hostis [7].

III. TRABALHOS RELACIONADOS

Esta seção descreve as principais ferramentas para geração de SoCs que se assemelham ao propósito deste trabalho.

GRLIB [8] é uma biblioteca de IPs que facilita o projeto de SoCs, fornecendo uma ampla gama de componentes reutilizáveis, como processadores e controladores de periféricos. O LEON, um processador desenvolvido pela ESA, e o NOEL-V, um processador RISC-V open-source, são exemplos de IPs disponíveis no GRLIB.

Rocket Chip Generator [9] é uma ferramenta para criar processadores RISC-V personalizados, com módulos como caches e unidades de execução. Após a configuração, pode gerar automaticamente o design do processador em VHDL ou Verilog, que pode ser sintetizado para FPGA ou ASIC.

PULPissimo [10], parte do projeto PULP (Parallel Ultra-Low-Power), é um SoC open-source focado em eficiência energética, ideal para sistemas embarcados. Sua arquitetura modular permite flexibilidade na adaptação para diferentes dispositivos, sendo baseado na arquitetura RISC-V.

HeMPS [11] é uma plataforma que permite a configuração e prototipação de sistemas integrados baseados em múltiplos núcleos em diversos cenários, com opções para personalização de processadores e barramentos, além de integração de aceleradores externos.

O trabalho de [3] consiste em uma ferramenta baseada em scripts Python para configurar sistemas integrados, permitindo a personalização de processadores, barramentos AMBA AXI4-Lite e integração de IPs.

Este trabalho se distingue ao buscar a integração utilizando a rede-em-chip XINA, com o propósito de conectar núcleos baseados no barramento AMBA AXI.

IV. DESENVOLVIMENTO

Neste trabalho, foi desenvolvida uma ferramenta para simplificar o processo de construção de um SoC utilizando rede-em-chip, interfaces de rede e IPs gerentes e subordinados AMBA AXI. Para verificar o funcionamento, foram desenvolvidos dois SoCs funcionalmente equivalentes que utilizam a XINA e sua interface de rede: um implementado manualmente e outro empregando a ferramenta.

A. *Materiais*

A rede-em-chip XINA, juntamente com sua interface de rede, foi utilizada para interconexão dos IPs. Os IPs empregados consistem em geradores de tráfego gerentes e medidores de tráfego subordinados. O desenvolvimento foi realizado utilizando VHDL 2008, com síntese e simulação na ferramenta AMD/Xilinx Vivado 2020.1. O script foi desenvolvido em Python 3.10, e os arquivos de configuração utilizados incluem templates de texto e arquivos no formato *ini*.

B. *Geração manual de um SoC*

A Figura 1 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 apresentada uma NoC 2×2 utilizando dois IPs gerentes e dois IPs subordinados, interconectadas por 4 roteadores XINA por suas respectivas interfaces de rede.

Gerar manualmente um SoC com NoC é trabalhoso, requer conhecimento específico da arquitetura e aumenta a possibilidade de erros devido à configuração manual de parâmetros e conexões entre IPs. Cada IP pode ter requisitos de interface e protocolo específicos, exigindo alto nível de detalhe na configuração. Com o aumento do número de IPs e da complexidade do SoC, o esforço para garantir o funcionamento dos componentes cresce significativamente.

C. *Geração automática de um SoC*

A Figura 2 apresenta a visão geral de um SoC gerado utilizando a ferramenta proposta. 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 os arquivos de configuração e gera automaticamente o módulo topo, com todas as conexões realizadas.

O script em Python também gera automaticamente uma matriz de interconexão com base nas informações do arquivo de configuração e nas definições para a XINA. Isso garante que a rede-em-chip tenha o tamanho correto e que os roteadores sejam configurados e conectados aos IPs conforme especificado. O SoC gerado por este método é funcionalmente equivalente ao obtido por implementação manual. No entanto, como toda a parte de interconexão é criada utilizando lógica generativa, o nível superior apresenta uma abstração maior dos componentes, agrupando-os em blocos relacionados às partes da interconexão que contêm todos os roteadores e interfaces de rede necessárias.

V. RESULTADOS

A ferramenta desenvolvida em Python tem tamanho total de 15,46 Kb. A Figura 3 mostra a saída do terminal após a execução do script. O usuário precisa preencher os arquivos `XINAFTSettings.ini`, que define as configurações da NoC XINA, e `RouterSettings.ini`, que especifica quais IPs ou interfaces de rede serão conectados a cada roteador.

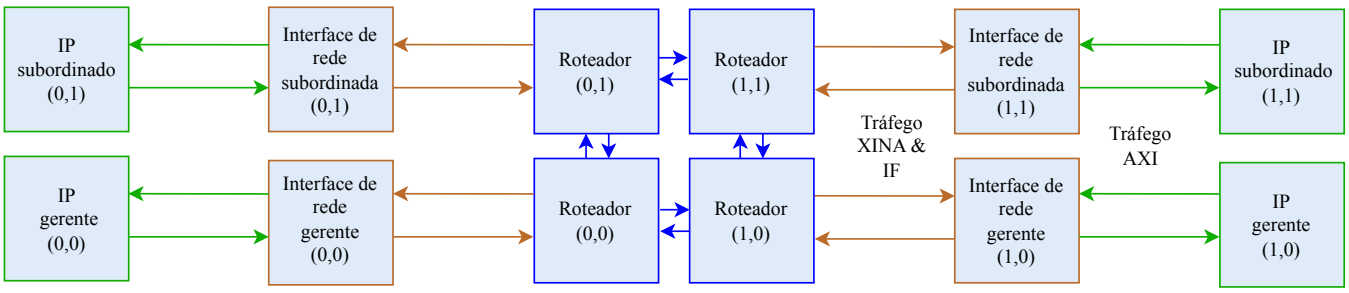

Fig. 1: *Visão geral de um SoC 2×2 elaborado manualmente.*

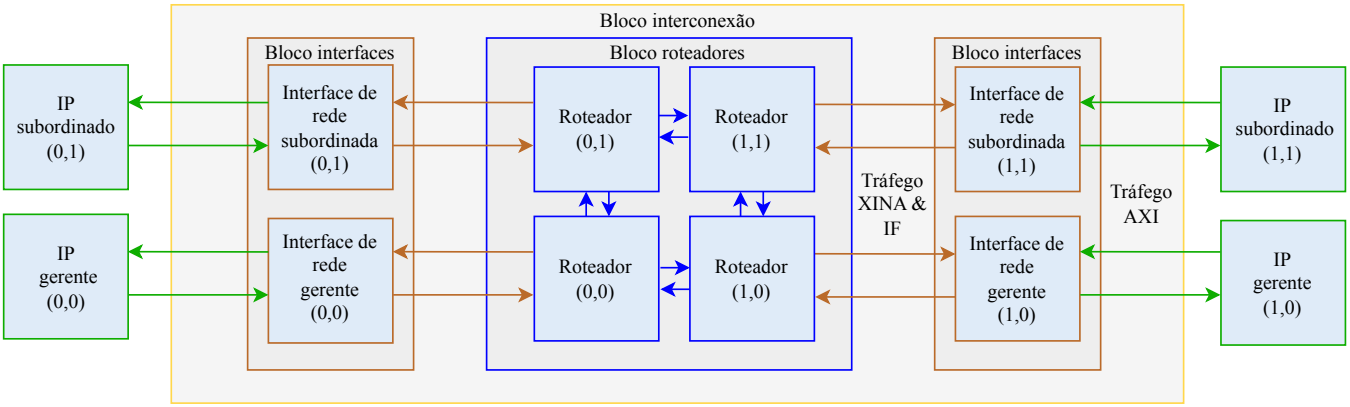

Fig. 2: *Visão geral de um SoC 2×2 elaborado automaticamente.*

```
bash RunHeadless.sh
Requirement already satisfied: pip in ./.venv/lib/python3.10/site-packages (24.
2)
Requirement already satisfied: configparser in ./.venv/lib/python3.10/site-pack
ages (from -r RequirementsHeadless.txt (line 1)) (7.1.0)
Virtual environment created and packages installed successfully.
XINAFTSettings:
 {'flow_ft_c': '0', 'routing_ft_c': '0', 'arbitration_ft_c': '0', 'buffering_ft
_c': '0', 'rows_c': '2', 'cols_c': '2', 'flow_mode_c': '0', 'routing_mode_c': '
0', 'arbitration_mode_c': '0', 'buffer_mode_c': '0', 'buffer_depth_c': '4', 'da
ta_width_c': '32'}
RouterSettings:
 {'Router_0_0': {'type': 'Manager', 'p_SRC_X': '0000000000000000', 'p_SRC_Y': '
0000000000000000', 'prefix': 'm_0_0', 'ip_connection': 'TrafficGeneratorForMana
gerNI'}, 'Router_0_1': {'type': 'Subordinate', 'p_SRC_X': '0000000000000000', '
p_SRC_Y': '0000000000000001', 'prefix': 's_0_1', 'ip_connection': 'SubordinateA
dder'}, 'Router_1_0': {'type': 'Manager', 'p_SRC_X': '0000000000000001', 'p_SRC
_Y': '0000000000000000', 'prefix': 'm_1_0', 'ip_connection': 'TrafficGeneratorF
orManagerNI'}, 'Router_1_1': {'type': 'Subordinate', 'p_SRC_X': '00000000000000
01', 'p_SRC_Y': '0000000000000001', 'prefix': 's_1_1', 'ip_connection': 'Subord
inateAdder'}}
VHDL file '../../Ips/Generated/NIs.vhd' updated.
VHDL file '../../Ips/xina-IF/hdl/dependencies/xina/rtl/ft/xina_ft.vhd' updated.
VHDL file '../../Ips/Generated/NIs_routers_and_IPs.vhd' updated.
```

Fig. 3: *Saída do terminal para execução do script.*

Após a etapa de configuração, basta executar um script bash que cria o ambiente virtual Python 3.10 e executa o programa. Esse programa interpreta os dados dos arquivos `.ini` e gera automaticamente os arquivos VHDL do projeto: `NIs.vhd`, que contém a matriz parametrizável de interfaces de rede; `xina_ft.vhd`, que configura a XINA com os parâmetros fornecidos; e `NIs_routers_and_IPs.vhd`, que insere todos os IPs ao bloco de interconexão.

As Figuras 4 e 5 apresentam o diagrama RTL (Register Transfer Level) do SoC gerado manualmente e pela ferramenta, ambos obtidos pelo Vivado e seguindo a mesma lógica de cores que as Figuras 1 e 2, com adição dos sinais de clock e reset na cor magenta. Os sistemas são compostos por dois IPs geradores de tráfego, dois IPs medidores de tráfego, suas respectivas interfaces de rede e quatro roteadores da XINA com matriz de tamanho 2×2.

A Figura 6 ilustra o interior do bloco. Nessa visão, as conexões AXI entram no bloco e conectam-se ao bloco de interfaces de rede, onde o tráfego é interpretado e convertido em tráfego XINA. O tráfego é então roteado pela XINA, retorna ao bloco de interfaces de rede, é novamente interpretado e liberado como tráfego AXI para os IPs. 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 um único bloco.

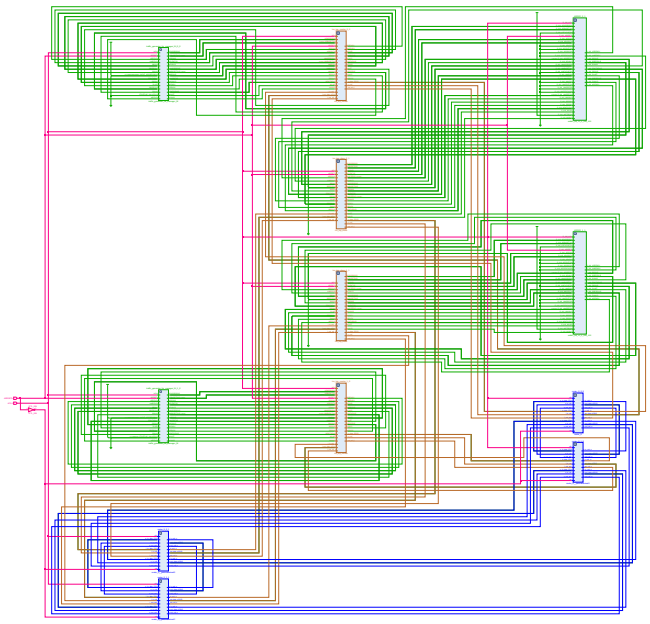

Fig. 4: *Diagrama RTL do SoC 2×2 gerado manualmente.*

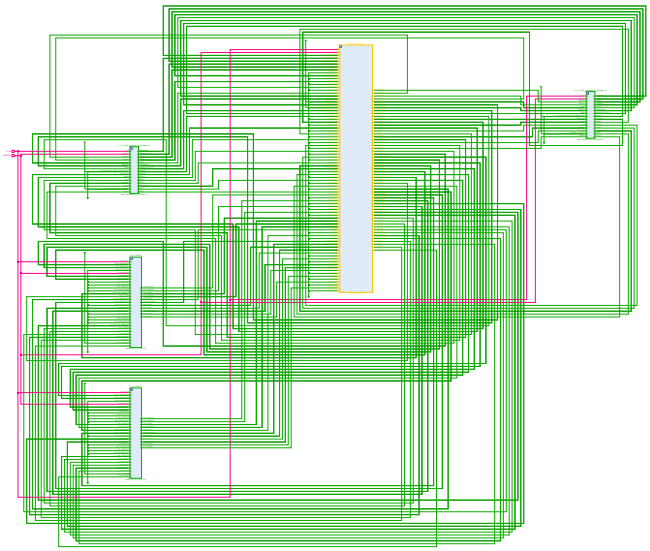

Fig. 5: *Diagrama RTL do SoC 2×2 gerado automaticamente.*

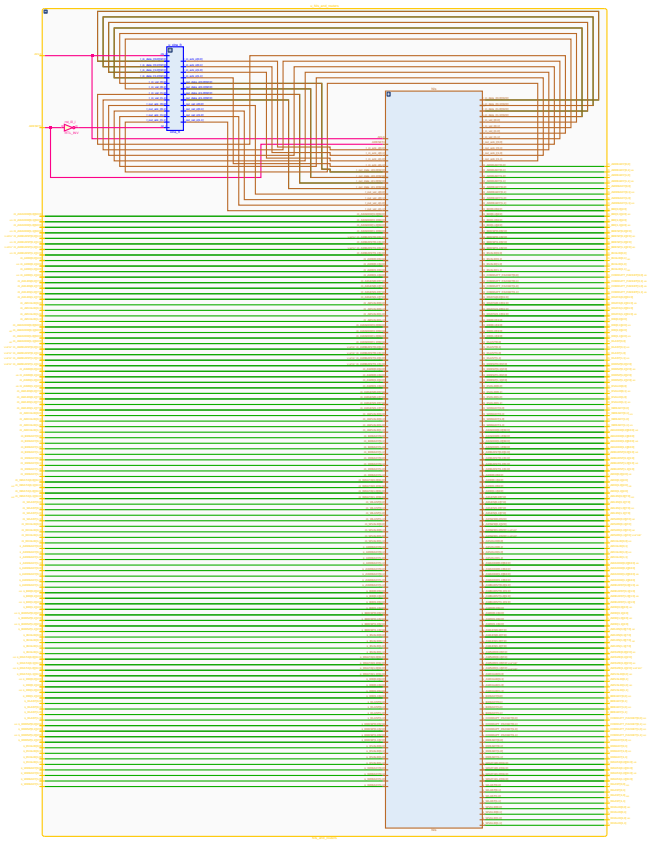

Fig. 6: *Diagrama RTL do bloco de interconexão.*

A síntese dos circuitos com o Vivado resultou na utilização de 3301 Look-Up Tables (LUTs), 987 Flip-Flops (FFs) e 842 Look-Up Table RAMs (LUTRAM). A frequência máxima obtida foi de 184,36 MHz, considerando o Field-Programmable Gate Array (FPGA) Zynq-7000 como dispositivo alvo. Como a versão gerada manualmente e a gerada pela ferramenta apresentam o mesmo circuito, a utilização de recursos e frequência máxima permaneceu a mesma.

A validação por simulação do SoC foi realizada no Vivado, por meio de uma transação de escrita com retorno. Nesse processo, o gerente envia um dado ao subordinado, que realiza uma operação de soma e retorna o resultado.

VI. CONCLUSÃO

Este trabalho apresentou o desenvolvimento de uma ferramenta que simplifica o processo de integração de componentes em SoCs utilizando redes-em-chip. A ferramenta visa abstrair a complexidade das conexões e reduzir a possibilidade de erros, facilitando a integração de IPs gerentes e subordinados com a NoC XINA. Os resultados demonstraram que é possível obter SoCs funcionalmente equivalentes com um esforço de desenvolvimento menor ao utilizar a ferramenta proposta, em comparação com a abordagem manual.

Como trabalho futuro, pretende-se expandir a ferramenta integrando uma interface gráfica que simplifique ainda mais o processo de configuração, evitando a necessidade de interação direta com arquivos de configuração. Além disso, a inclusão de suporte a outros protocolos de comunicação e a possibilidade de integrar diferentes tipos de IPs podem ampliar a versatilidade da ferramenta.

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) e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq (processos 313513/2021-0, 350208/2022-0, 408641/2023-1 e 350794/2023-5).

REFERÊNCIAS

[1] D. Greaves, *Modern System-on-Chip Design on Arm*. ARM Education Media, 2021.

[2] L. P. Tedesco, "Uma proposta para geração de tráfego e avaliação de desempenho para nocs," Master's thesis, Pontifícia universidade católica do rio grande do sul, 2005.

[3] C. N. Domingos, D. A. Santos, W. Grignani, and D. R. Melo, "Plataforma de integração de componentes para sistemas embarcados em aplicações espaciais," *Anais do Computer on the Beach*, vol. 14, pp. 444–446, 2023.

[4] ARM, "AMBA Specification," 2023. Disponível em: `https://developer.arm.com/Architectures/AMBA`.

[5] ARM, "AMBA AXI Protocol Specification," 2023. Disponível em: `https://developer.arm.com/documentation/ihi0022/latest`.

[6] C. A. Zeferino and A. A. Susin, "Socin: a parametric and scalable network-on-chip," in *16th Symposium on Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings.*, pp. 169–174, IEEE, 2003.

[7] D. R. Melo, C. A. Zeferino, L. Dilillo, and E. Bezerra, "Maximizing the inner resilience of a network-on-chip through router controllers design," *Sensors*, 2019.

[8] F. Gaisler, "GRLIB IP library user's manual 2023.2." Disponível em: `https://www.gaisler.com/`, 2023.

[9] K. Asanivic, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz, S. Karandikar, B. Keller, D. Kim, J. Koenig, Y. Lee, E. Love, M. Maas, A. Magyar, H. Mao, M. Moreto, A. Ou, D. A. Patterson, B. Richards, C. Schmidt, S. Twigg, H. Vo, and A. Waterman, "The rocket chip generator," Tech. Rep. UCB/EECS-2016-17, EECS Department, University of California, Berkeley, Apr 2016.

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

[11] Gaph, "Hemps." Disponível em: `https://www.inf.pucrs.br/hemps/index.html`, 2016.