Estudo e desenvolvimento de periféricos internos para trabalhar em conjunto com microarquiteturas RISC-V baseadas no ISA RV32I.

Luiz Fernando C. Silva (IC), Gustavo Della Colleta (PQ)
Universidade Federal de Itajubá.

Palavras-chave: Avalon. Barramento. GPIO. Platform Designer.

Introdução

O tema deste estudo envolve o desenvolvimento de periféricos internos para integração com microarquiteturas RISC-V baseadas na ISA RV32I. O objetivo principal é desenvolver a interconexão do módulo *core*, construído sob essa microarquitetura, com periféricos como memória de dados, GPIO (*General Purpose Input/Output*) e *Timer*, utilizando a tecnologia de barramento Avalon - o Avalon é um protocolo de barramento proprietário da Intel/Altera que interconecta diferentes blocos de *hardware*. A implementação é realizada por meio da ferramenta *Platform Designer*, disponível no software Quartus, a qual exige a utilização da tecnologia de barramento Avalon na criação de Socs (*System on-Chip*). O barramento, elemento central do sistema, é responsável pelas conexões lógicas, garantindo a comunicação entre os diferentes módulos do sistema.

A justificativa para a pesquisa se dá pela relevância do barramento Avalon na integração de periféricos com microarquiteturas modernas, como o RISC-V, o que permite criar sistemas eficientes e escaláveis, principalmente em ambientes embarcados. Assim, a utilização do *Platform Designer* facilita o desenvolvimento e gerenciamento dos sinais, otimizando o design do sistema.

O método utilizado compreende, inicialmente, o estudo da arquitetura RISC-V, seguido pelo aprofundamento no funcionamento dos periféricos, especialmente memória e GPIO. Por fim, o estudo e implementação do barramento Avalon se dá através do uso do *Platform Designer*, que permite a importação dos módulos e o gerenciamento dos sinais de comunicação de acordo com a tecnologia de barramento Avalon.

Metodologia

O trabalho teve início com um estudo geral sobre a tecnologia RISC-V e suas particularidades [1]. Em seguida, foi realizado o estudo das microarquiteturas compatíveis com o RISC-V, abrangendo as arquiteturas *single cycle* e *multi cycle* [2]. Com essa etapa concluída, os estudos avançaram para os periféricos de memória, incluindo o funcionamento da memória RAM (*Random Access Memory*) [3][5], com foco nas tecnologias DRAM (*Dynamic Random-Access Memory*) e SRAM (*Static Random-Access Memory*) [3], além da memória ROM (*Read-Only Memory*), abordando especificamente a memória *FLASH* [3]. Durante essa fase, foi dada especial atenção às memórias de instrução e de dados.

Após o estudo das memórias, a pesquisa se concentrou no periférico GPIO (*General Purpose Input/Output*), que permite a comunicação do microcontrolador com o meio externo [7][8][9]. O funcionamento do GPIO no hardware foi analisado por meio da ferramenta IP Catalog do software Quartus, que oferece uma implementação genérica do GPIO. Utilizando o recurso *Netlist Viewers*, foi possível verificar a estrutura interna do GPIO, que inclui buffers de entrada e saída de dados e o pino oe (*output enable*), responsável por controlar a direção desses dados.

Na etapa seguinte, procedeu-se à interligação dos periféricos com o núcleo (*core*) utilizando a tecnologia de barramento Avalon [10]. Foi necessário adaptar o código em Verilog do core baseado na arquitetura RISC-V, desenvolvido durante os estudos de iniciação científica.

Para compatibilizar o core com o barramento Avalon, foram criados sinais específicos no código, correlacionados à tecnologia Avalon, exemplificado na figura 1.

```
module cpu(
    input CLK, RST,
    input [31:0] ReadData,
    output [31:0] ALUResult, PC, WriteData,
    output MemWrite,

    // Avalon signals
    input avl_waitrequest,
    output [31:0] avl_address,
    output [31:0] avl_writedata,
    input [31:0] avl_readdata,
    output [3:0] avl_byteenable,
    output avl_read,
    output avl_write,
    output avl_chip_select
);
```

Figura 1: Criação dos sinais Avalon no módulo CPU

O Avalon é uma tecnologia de barramento utilizado para conectar diferentes blocos de hardware, amplamente utilizado em um sistema de chip (SoC). O mesmo, possui interface padronizada que simplifica a interligação entre

os módulos periféricos e o núcleo (*core*), uma das suas principais características e vantagens é sua flexibilidade, uma vez que o mesmo permite uma variedade de configurações de barramentos, como leitura e escrita, *chip select*, instruções. Além disso, o Avalon possui a subinterface Avalon-MM (*Memory-Mapped Interface*), que facilita a comunicação entre o processador e periféricos mapeados em memória - como pode se visualizar na figura 2 - uma vez que o núcleo acessa os periféricos como se fossem posições de memória.

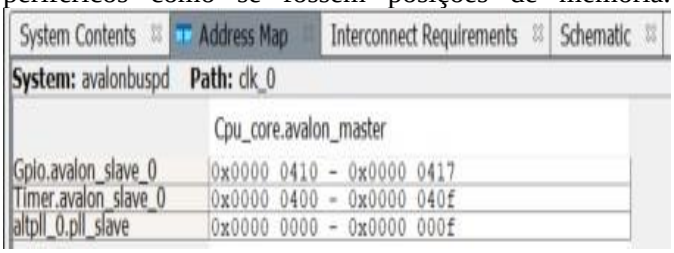

Figura 2: *Adress Map*, ferramenta para configuração dos endereços de memória.

Em seguida, a ferramenta *Platform Designer* foi utilizada para importar os periféricos. É importante notar que a ferramenta não foi originalmente desenvolvida para a tecnologia RISC-V, o que exige adaptações no código da CPU e nos sinais dos periféricos utilizados - isso é exemplificado na Figura 3, que mostra a importação do módulo GPIO. Entre os periféricos importados estão o módulo *DataMemory*, destinado à memória de dados, o módulo GPIO, que foi referenciado a partir de um projeto disponível no GitHub [11], conhecido por empregar a tecnologia de barramentos Avalon e o módulo CPU.

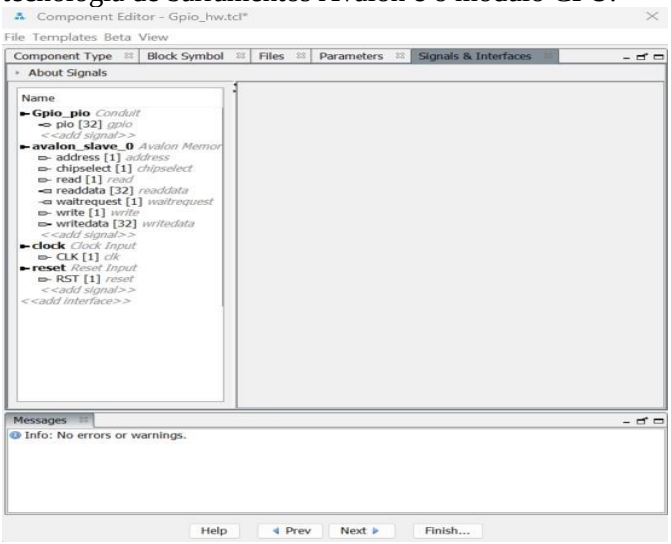

Figura 3: Adição do módulo GPIO, com os sinais compatíveis ao Avalon

Quanto ao módulo *Timer*, optou-se pelo módulo disponível no IP Catalog. A escolha dos módulos externos visou facilitar a manipulação dos sinais de barramento, já que os módulos nativos do IP Catalog no *Platform Designer* não permitem essa manipulação. Assim, todas as etapas foram conduzidas com o objetivo de garantir a interconexão eficiente dos periféricos ao core utilizando a tecnologia de barramento Avalon, como demonstrado na figura 4.

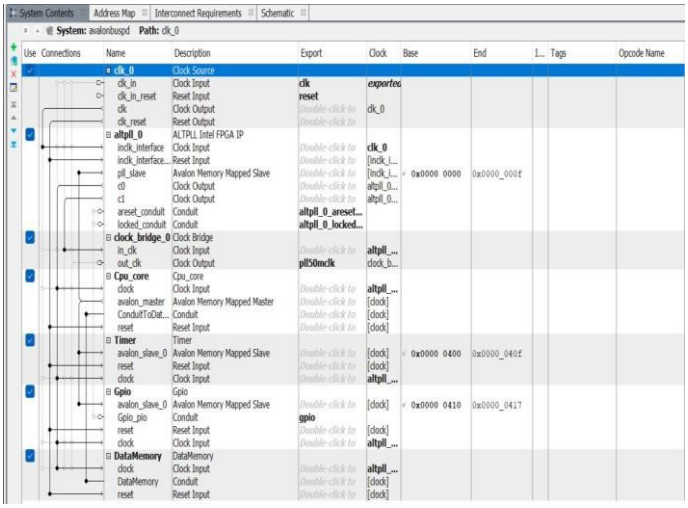

Figura 4: Resultado final da importação dos módulos utilizados com a adaptação para à interface Avalon

Resultados e discussão

O desenvolvimento do projeto permitiu a aquisição de conhecimentos fundamentais para a integração de núcleos e periféricos descritos em Verilog com o barramento de comunicação Avalon, nativo do *Platform Designer*. Inicialmente, foi realizada a importação dos componentes externos, como o núcleo RISC-V, memória de dados, GPIO e posteriormente o periférico interno do *Platform Designer*, *TIMER*. Esse processo envolveu a análise dos arquivos de síntese e a criação dos sinais necessários para o correto funcionamento do sistema.

No caso específico do núcleo RISC-V, foi criado o barramento Avalon *Memory Mapped Master*, o qual foi responsável por controlar as operações de leitura e escrita nos periféricos conectados, como o GPIO e o *TIMER*, que foram configurados com a interface Avalon *Memory Mapped Slave*. A interface Avalon *Conduit* foi utilizada para conectar a memória de dados ao núcleo, visto que ela se comunica por um barramento genérico. Durante esse processo, foi necessária a criação manual de sinais essenciais, como *chip select*, *address*, *waitrequest* e *clock*. Outro aspecto importante foi a implementação de um circuito PLL (*Phase-Locked Loop*), adicionado para gerar o sinal de clock de forma sincronizada para todos os periféricos. Esse PLL foi obtido através do IP Catalog do *Platform Designer*, e permitiu garantir que os dispositivos operassem em uma frequência comum. Além disso, uma *clock bridge* foi utilizada para garantir a distribuição uniforme do sinal de *clock*, caso fosse

necessário compartilhar o mesmo *clock* entre diferentes dispositivos. A figura 5 representa o resultado final das interligações do núcleo *core*, periféricos e associações com o barramento Avalon.

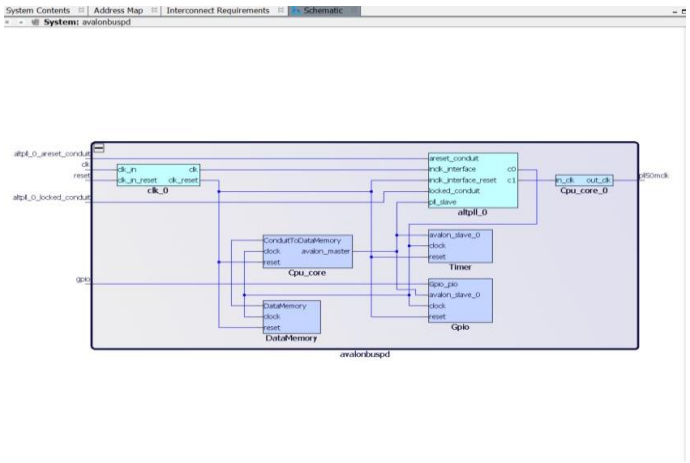

Figura 5: Esquemático para a visualização da lógica de interligação de módulos desenvolvida

Após a importação dos componentes, foi realizado o ajuste da faixa de endereços dos periféricos no *Address Map* do *Platform Designer*. Essa etapa foi crítica para garantir que cada periférico ocupasse o espaço de memória correto, evitando conflitos de endereçamento no sistema. Com os endereços definidos, foi gerado o HDL (*hardware description language*) do projeto, e, em seguida, foi criada uma nova instância de projeto no Quartus para importar os arquivos gerados.

Durante a fase de prototipação, enfrentamos um obstáculo relacionado à memória. Embora a integração com o barramento Avalon exija uma memória síncrona Antes de acessar o *netlist viewer*, foi necessário compilar o projeto. Durante essa compilação, o Quartus realiza diversas verificações essenciais, como a checagem da conectividade dos sinais, a síntese da lógica descrita no HDL, o mapeamento dos blocos lógicos para os recursos físicos do FPGA (processo de *fitting*), e a verificação de que o design atende aos requisitos de temporização. Essas etapas são fundamentais para assegurar que a arquitetura do sistema esteja correta e coesa, mesmo sem a execução de testes práticos.

Embora o projeto tenha enfrentado um obstáculo relacionado à memória - visto que a integração com o barramento Avalon exige uma memória síncrona e apenas uma memória assíncrona foi sintetizada, impedindo então testes reais -, foi possível prosseguir com a validação da lógica implementada. Com o projeto compilado e verificado pelo Quartus, utilizamos o *netlist viewer* para inspecionar visualmente as conexões entre os periféricos e o núcleo, garantindo que a lógica projetada estava implementada conforme esperado. A figura 6 representa o circuito gerado, e por meio dessa visualização foi possível assegurar a coesão das conexões entre os módulos, corroborando o funcionamento teórico da lógica desenvolvida.

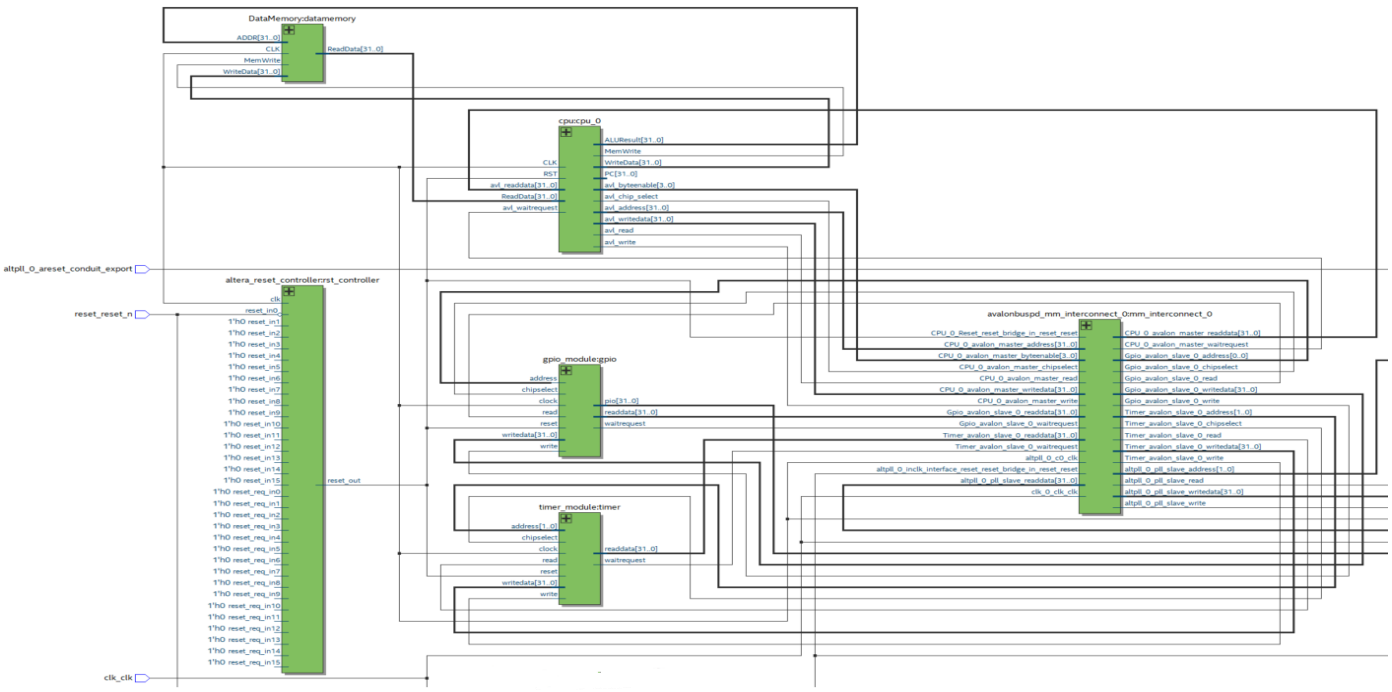

Figura 6: Visualização do projeto após a compilação por meio do netlist viewer

Conclusões

Nesta iniciação científica, foi realizado um estudo aprofundado sobre a tecnologia RISC-V, com foco em sua arquitetura ISA RV32I, que inclui o conjunto de instruções comprimidas, a ausência de registradores de status e os modos privilegiados. Além disso, foram analisadas as microarquiteturas compatíveis, como a de ciclo único (*single cycle*), que utiliza um *clock* baseado na instrução mais lenta, e a de múltiplos ciclos (*multi cycle*), onde cada instrução é executada em vários ciclos de *clock*. Esse estudo também abrangeu periféricos, como a memória e o GPIO, explorando suas tecnologias, funcionamento e, no caso do GPIO, aprofundando-se até a parte do funcionamento lógico do *hardware*.

O desenvolvimento do barramento de comunicação utilizando o *Platform Designer* foi o principal resultados obtido. Esse processo proporcionou o aprendizado sobre o uso dessa ferramenta robusta e flexível, que permite a integração de tecnologias externas, não nativas do *Platform Designer*. Essa flexibilidade foi essencial para conectar núcleos e periféricos personalizados, como o RISC-V, ampliando as possibilidades de manipulação de microarquiteturas e de desenvolvimento de sistemas personalizados.

O estudo destacou a importância do barramento de comunicação, que se mostrou fundamental para interligar toda a lógica dos componentes, garantindo o funcionamento adequado do sistema como um todo. A aplicação do barramento Avalon nativo do *Platform Designer*, combinada com a adaptação de tecnologias externas, permitiu integrar com sucesso os periféricos e o núcleo RISC-V, demonstrando a flexibilidade e a eficiência dessa abordagem. Esse processo não apenas possibilitou uma comunicação eficiente entre os diferentes módulos, mas também evidenciou o potencial do barramento Avalon para escalar o sistema com novos periféricos e funcionalidades no futuro. Assim, o estudo ressalta a importância de uma arquitetura bem planejada e de ferramentas de desenvolvimento robustas, como o *Platform Designer*, para o sucesso na construção de sistemas personalizados e otimizados.

Agradecimentos

Agradeço à Universidade Federal de Itajubá (UNIFEI) pela oportunidade de realizar esta pesquisa e ao Programa Institucional de Bolsas de Iniciação Científica (PIBIC) pelo apoio financeiro fundamental para o desenvolvimento deste trabalho.

Referências

[1] PATTERSON, D.; WATERMAN, A. **Guia Prático RISC-V: Atlas de uma Arquitetura Aberta**. Primeira edição, 2017.

[2] HARRIS, S. L.; HARRIS, D. M. **Digital Design and Computer Architecture**. Burlington: Morgan Kaufmann, 2012.

[3] PATTERSON, D. A.; HENNESSY, J. L. **Computer Organization and Design**. Cambridge: Morgan Kaufmann, 1998.

[4] INSIGHTVITY. **Difference between program and data memory**. Disponível em: insightvity.com/difference-between-program-and-data-memory/#overview-ofprogram-and-data-memory. Acesso em: maio. 2024.

[5] GEEKSFORGEEKS. **Difference between RAM and ROM**. Disponível em: www.geeksforgeeks.org/difference-between-ram-and-rom/. Acesso em: maio. 2024.

[6] RESEARCHGATE. **Portal de pesquisas científicas**. Disponível em: www.researchgate.net/. Acesso em: maio. 2024.

[7] INTEL CORPORATION. **Intel MAX 10 General Purpose I/O User Guide**. Intel, 2021. Disponível em: www.intel.com/content/www/us/en/docs/programmable/683751/21-1/i-ooverview.html. Acesso em: maio. 2024.

[8] NERDY ELECTRONICS. **Introduction to GPIO**. Nerdy Electronics, 2024. Disponível em: nerdyelectronics.com/introduction-to-gpio/. Acesso em: maio. 2024.

[9] EMBETRONICX. **Understanding the microcontroller GPIO – GPIO working explained**. Disponível em: embetronicx.com/tutorials/tech_devices/understandingthe-microcontroller-gpio-gpio-working-explained/. Acesso em: maio. 2024.

[10] INTEL CORPORATION. **Introduction to the Avalon Interface Specifications**. Intel, 2021. Disponível em: Introduction to the Avalon® Interface Specifications (intel.com). Acesso em: 6 agosto. 2024.

[11] SHAHEER, S. **RISCV: 32-bit soft RISCV processor for FPGA applications**. Repositório no GitHub, 2024. Disponível em: github.com/ShaheerSajid/RISCV. Acesso em: agosto. 2024.

[12] LUO, W. **Build A Soft Core CPU - Part Three - NIOS II in Intel FPGA**. YouTube, 2023. Disponível em: Build A Soft Core CPU - Part Three - NIOS II in Intel FPGA - YouTube. Acesso em: agosto. 2024.