MC851 - Projeto em Computação I

arRISCado - Entrega 1

Ângelo Renato Pazin Malaguti - 165429
Claudio dos Santos Júnior - 195727
Elton Cardoso do Nascimento - 233840
Gabriel Costa Kinder - 234720
Iago Caran Aquino - 198921
João Pedro de Moraes Bonucci - 218733

Planejamento e protótipo inicial de um processador RISC-V 32 bits

1. Introdução

Este trabalho tem como propósito principal o desenvolvimento de um processador RISC-V de 32 bits com a capacidade de interpretar o conjunto de instruções RV32IMAC e estabelecer comunicação com no mínimo dois periféricos. Constitui a primeira fase de um projeto dividido em quatro entregas planejadas, abarcando a definição de requisitos, especificação do sistema, modelagem do processador, validação e operacionalização. O presente documento detalha os passos executados em cada uma dessas etapas, apresentando tanto os resultados conquistados como os desafios enfrentados.

2. Planejamento e infraestrutura

No estágio inicial, nosso foco primordial foi projetar um processador RISC-V capaz de interpretar as instruções do conjunto RV32I, com a implementação de um pipeline completo. Adicionalmente, trabalhamos para configurar um ambiente de execução adequado para o processador, culminando na compilação do projeto para posterior execução em uma FPGA.

Para dar início a essa empreitada, conduzimos uma análise minuciosa tanto das ferramentas e ambientes recomendados, quanto dos manuais pertinentes à FPGA selecionada, o modelo Tang Nano 9k. A configuração do ambiente de desenvolvimento abrangeu a adoção do editor de texto VS Code, complementado pelas extensões Lushay Code, Verilog-HDL e WaveTrace, além da suíte de desenvolvimento OSS-CAD-Suite, especialmente o ICARUS

Verilog Compilation System. Paralelamente, utilizamos os materiais didáticos fornecidos durante aulas anteriores, compreendendo slides e anotações da disciplina MC732 (Projeto de Sistemas Computacionais) e MC404 (Organização Básica de Computadores e Linguagem de Montagem), amplamente apoiados por fontes bibliográficas indicadas para revisitar linguagem de máquina, a arquitetura e estabelecer o escopo do nosso processador.

3. Desenvolvimento

Após a definição dos requisitos e a especificação do sistema, prosseguimos com a modelagem do processador RISC-V, seguindo o paradigma de um pipeline de cinco estágios: busca, decodificação, execução, acesso à memória e escrita no registrador. Para cada estágio, elaboramos um módulo em Verilog que incorporou as funcionalidades necessárias para interpretar as instruções RV32I. Além disso, criamos módulos auxiliares para representar a memória de dados, a memória de instruções, o banco de registradores e a unidade de controle, assim como os barramentos entre cada etapa do pipeline. Distribuimos as tarefas entre três duplas: uma dedicada a testes, outra à gestão das fases de controle, decodificação e execução, e a terceira responsável pelas etapas de busca, acesso à memória e escrita no registrador. Entretanto, essa abordagem suscitou alguns problemas de comunicação, que, somados à nossa inexperiência com a linguagem de desenvolvimento escolhida, ocasionaram atrasos no planejamento inicial e demandaram revisões e retrabalhos consideráveis.

Na segunda iteração, optamos por concentrar nossos esforços na implementação completa do pipeline. Para isso, decidimos restringir o número total de instruções a serem implementadas de uma só vez, adotando uma abordagem de desenvolvimento incremental. Uma alteração adicional envolveu a criação de conjuntos de testes específicos para cada módulo, permitindo a validação de segmentos individuais de código e a interação entre os vários estágios. Também realizamos simplificações em estruturas auxiliares, como a remoção de módulos externos, incluindo o adder, register e multiplexer. Isso contribuiu significativamente para a melhoria da legibilidade do código.

Entretanto, para esta primeira entrega, decidimos remover uma representação mais realista da memória acessada pelo processador, originalmente pensada para funcionar como uma memória externa para todo o processador, e passamos a utilizar uma memória rom para a etapa de busca de instrução apenas.. Além disso, durante o pipeline não fizemos nenhum tratamento de hazards de controle, dados ou estruturais até o momento da entrega.

4. Aprendizado

- Ângelo

Entendimento mais aprofundado da sintaxe do Verilog, saber como funciona o design de um processador/Hardware na prática (só tive matérias teóricas sobre isso até o momento).

- Claudio

Primeira experiência de programação utilizando uma FPGA, foi necessário revisar tópicos de mc404 e mc732. Fora isso, primeira experiência com Verilog e com o assembly RISCV. Meu conhecimento sobre arquitetura de computadores e RISCV eram apenas teóricos, essa está sendo a primeira oportunidade de colocar esses conhecimentos em prática.

- Gabriel

Aprofundamento em conhecimentos e elaboração de programas em assembly de RISCV e do funcionamento de instruções específicas do conjunto RV32I. Aprendizagem sobre a elaboração de testes unitários que testem corretamente seus usos e edge cases.

- João

Contato inicial com ambiente de desenvolvimento em verilog e contato com FPGAs modernas. Além disso, esse primeiro mês foi útil para revisão de conceitos de arquitetura de computadores e instruções RISC.

- Iago

Possibilidade de dedicar tempo para estudar Verilog e construir uma CPU. Aprendendo sobre as sutilezas das ferramentas de síntese para FPGA que possuem recursos que devem ser evitados, como o uso de initial.

5. Resultados

Atualmente temos as etapas de busca de instruções e decodificação funcionando e integradas, além disso conseguimos validar as etapas de acesso a memória e escrita em registradores (writeback). No entanto, não conseguimos terminar a etapa de execução até a

entrega deste relatório nem o controle. Os códigos e testes podem ser encontrados em nosso repositório e executados com o programa de simulação preferido do leitor, no entanto utilizamos iverilog para validar os módulos com os testes.

6. Pendências

Para essa entrega deixamos como débito a integração entre as etapas de decodificação, execução e memória assim como a finalização do mecanismo de controle. Também pretendemos adicionar o controle de hazards conhecidos e combinar o código dos barramentos dentro dos próprios estágios de pipeline, melhorando a legibilidade e complexidade do circuito sem perder os benefícios de uma arquitetura com pipeline.

7. Referências

PATTERSON, David A.; Hennessy, John L. Computer Organization and Design RISC-V Edition. Elsevier, 2020.

PATTERSON, David; WATERMAN, Andrew. The RISC-V Reader: An Open Architecture Atlas. 1ª edição. Publicação livre, 2017.

Component Byte. Verilog Tutorial for Beginners. [s.d.]. Disponível em: <https://www.youtube.com/playlist?list=PLAC_jmBddcjTPEh1UV_ojRJmsx2D9sQXH>. Acesso em: 20 ago. 2023.