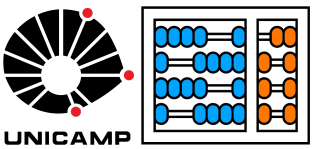

MC851 Projeto em Computação

Relatório Final

Resumo: planejamento e desenvolvimento de um processador RISC-V de 32 bits, com pipeline, que suporte o conjunto RV32I.

Equipe "RISC-VI":
RA 169374, Daniel Paulo Garcia
RA 182783, Lucca Costa Piccolotto Jordão
RA 185447, Paulo Barreira Pacitti
RA 198435, Guilherme Tavares Shimamoto
RA 216116, Gabriel Braga Proença
RA 221859, Mariana Megumi Izumizawa

1. Introdução

O *design* e desenvolvimento de um sistema computacional pode ser muito complexo, e muito caro também. Graças a tecnologias de HDL (*Hardware Description Language*) e *chips* FPGA (*Field Programmable Gate Arrays*), podemos prototipar *chips* de forma barata e rápida. Mesmo com experiências com RISC-V em MC404 (Organização Básica de Computadores e Linguagem de Montagem), com FPGA e HDL em MC613 (Laboratório de Circuitos Digitais) e MC732 (Projeto de Sistemas Computacionais), construir um SoC (*System-on-a-Chip)*, a equipe entende o projeto como um desafio bastante complexo e têm estudado e revisado conceitos dessas disciplinas. Nesta terceira entrega, apresentamos o andamento do terceiro mês de desenvolvimento realizado de um sistema computacional com arquitetura RISC-V de 32 bits que têm suporte ao conjunto RV32I.

Foram implementados dois chips: o RISC-6™ A-500, com a RV32I (apenas com load/store de words), memória integrada e periféricos, demonstrado na demo presencialmente, e, o RISC-6™ A-1000, chip incompleto com a implementação com um novo design de unidade de memória com caches L1. O código da implementação do sistema está localizado no GitHub, onde a branch `plan-b` contém o código relativo ao chip RISC-6™ A-500 completo, e a branch `memory-control` contém o código ao chip incompleto RISC-6™ A-1000. Para execução da demo demonstrada em sala de aula, deve ser aberto o projeto contido na pasta `mc851_gowin_project/`, na branch `plan-b`, no Gowin IDE.

2. Planejamento

O planejamento do nosso projeto, dividido em quatro entregas, reflete uma abordagem progressiva e adaptativa no desenvolvimento de um sistema computacional com base na arquitetura RISC-V.

Iniciamos com o design de um processador com pipeline RISC-V de cinco estágios. A equipe foi dividida em três duplas para desenvolver os estágios do pipeline: Gabriel e Lucca ficaram responsáveis pelos estágios IF e ID, Mariana e Paulo pelo estágio EX, e Daniel e Guilherme pelos estágios MEM e WB.

Devido à falta de familiaridade inicial com Verilog, optamos por implementar módulos simples como a ALU e o RegFile e, em seguida, integrá-los aos respectivos estágios. Determinamos que cada módulo implementado deveria ter um testbench associado, essencial para prevenir dificuldades na integração dos módulos e facilitar o debugging.

Durante o primeiro mês, nossas reuniões semanais revelaram a necessidade de melhorar a comunicação e a colaboração. Inicialmente, utilizamos mensagens de grupo no WhatsApp, o que levou à perda de documentação importante e dificuldades no acompanhamento das atividades.

Para aprimorar nossa colaboração, passamos a realizar reuniões duas vezes por semana e criamos um servidor no Discord com canais organizados por tópicos. Além disso, começamos a usar o GitHub para gerenciar pull requests, comentários e rastrear issues.

Decidimos focar na finalização da implementação da instrução ADDI, na montagem de uma planilha de sinais e instruções e na pesquisa e integração de um periférico. O grupo se dividiu novamente em duplas para as tarefas, com Guilherme e Mariana focando nos testes de

integração com um periférico (Serial Console) e no desenvolvimento da instrução BEQ, enquanto Lucca e Gabriel trabalharam na planilha de sinais e instruções, e Paulo e Daniel focaram na implementação da MMU e na refatoração da CPU.

Após avaliar os resultados da segunda entrega, decidimos focar na integração de periféricos, no desenvolvimento de instruções dos tipos I e R, na implementação da cache L1 e no controle de acesso à memória, além da implantação do sistema na placa FPGA.
A equipe foi reorganizada em novas duplas: Guilherme e Mariana ficaram responsáveis pela integração dos periféricos e pela implantação na placa FPGA, Lucca e Gabriel pelos testes das instruções dos tipos I e R, e Paulo e Daniel pela implementação das caches e controle de acesso à memória principal.

A última fase do projeto foi dedicada a finalizar as pendências remanescentes para a entrega final, concentrando-se na integração e refinamento de todos os componentes desenvolvidos até então.

Esse planejamento evidencia uma evolução contínua, onde cada fase construiu sobre os resultados e aprendizados da anterior, permitindo uma abordagem iterativa e adaptável para o desenvolvimento do nosso sistema computacional.

3. Resultados

Ao finalizar nosso projeto, entregamos um resultado significativo na área de arquitetura de computadores. Conseguimos desenvolver uma CPU de cinco estágios capaz de executar instruções aritméticas e lógicas dos tipos I, R, B e instruções de load e store de words. É importante notar que, apesar de as instruções de load e store estarem implementadas, elas ainda não produzem efeitos práticos devido a limitações na memória, como não possuir data forwarding. Conseguimos integrar com sucesso periféricos ao processador, ampliando assim a funcionalidade e a aplicabilidade do nosso sistema. Esses periféricos incluem LEDs e botões, ambos acoplados ao processador, oferecendo interatividade e feedback direto do hardware.

O periférico LED foi projetado para operar de forma binária, permitindo acender ou apagar os LEDs através da escrita de valores binários. Esta funcionalidade, embora aparentemente simples, demonstra a capacidade do nosso sistema de interagir com elementos de hardware externos de maneira eficaz. A implementação dessa característica requer uma compreensão clara de como os sinais são gerenciados e processados dentro da arquitetura do nosso processador, refletindo nossa habilidade em manipular e controlar dispositivos físicos através de instruções de software.

Por outro lado, o periférico botão foi configurado para operar em um modo de somente leitura, com um buffer que indica se ele foi pressionado recentemente. Essa funcionalidade é importante para fornecer um meio de entrada para o usuário, permitindo interações diretas com o sistema. A implementação de um buffer para o botão requer não apenas a capacidade de ler o estado do hardware, mas também de manter esse estado acessível para o processador.

A inclusão desses periféricos no projeto acrescenta uma dimensão prática e interativa ao nosso sistema, demonstrando nossa capacidade de integrar componentes de hardware de maneira eficiente e eficaz. Embora o foco principal do projeto tenha sido a CPU e a cache, a habilidade de acoplar e gerenciar periféricos como LEDs e botões evidencia uma compreensão abrangente do design de sistemas computacionais e uma aplicação prática dessa compreensão

Um aspecto notável do nosso projeto é a resolução da maioria dos hazards de dados, embora ainda persista o desafio do hazard "Use after load". Realizamos a inclusão de uma cache L1 de 512 linhas por 32 bits, totalizando 2 KiB, operando com mapeamento direto e uma política de write-back e write-allocate.

Contudo, o projeto conclui sem a implementação de um barramento de memória unificado. Este componente seria fundamental para mapear endereços físicos aos dispositivos correspondentes e fornecer à Unidade de Gerenciamento de Memória (MMU) as flags de permissão de acesso para cada região de memória. Isso seria essencial para a funcionalidade completa do Mapeamento de Memória de Entrada/Saída (MMIO) e periféricos. Além disso, a máquina de estados da MMU necessitaria de ajustes para operar eficientemente com essa arquitetura.

Este resultado, embora não inclua todos os aspectos inicialmente planejados, como o barramento de memória unificado e as modificações na MMU, representa um avanço notável e significativo em nosso conhecimento e habilidades técnicas. Demonstramos nossa capacidade de enfrentar e superar desafios complexos na área de design e programação de hardware, e o projeto como um todo servirá como uma base sólida de aprendizado e experiência para futuros empreendimentos na área de arquitetura de computadores.

4. Aprendizados e desafios

Nossa experiência com o projeto de implementação de um processador utilizando a arquitetura RISC-V e a linguagem HDL Verilog foi extremamente enriquecedora e desafiadora. Nós percebemos que nosso principal aprendizado centrou-se na compreensão detalhada do funcionamento de pipelines em processadores e no processo de decodificação de instruções. Essa compreensão aprofundada foi crucial para nos ajudar a desenvolver nossos conhecimentos práticos em design e implementação de sistemas computacionais, especialmente no que tange ao desenvolvimento em Verilog e ao entendimento completo dos datapaths em CPUs.

Um dos maiores desafios que enfrentamos foi a integração de vários componentes do sistema, como os estágios, módulos e a unidade de gerenciamento de memória (MMU) com a CPU, particularmente nas instruções de Load/Store. Além disso, adaptar-nos ao paradigma de desenvolvimento em hardware e gerenciar a simultaneidade inerente ao Verilog representou uma barreira significativa para nós. Outro grande desafio foi a fase inicial do projeto, que envolveu a configuração do ambiente e a complexidade de síntese e carregamento do processador na FPGA.

Nosso trabalho se concentrou principalmente na decodificação de instruções e na elaboração de testbenches correspondentes. Também dedicamos uma atenção especial ao

desenvolvimento do CPU, mais especificamente à sua Pipeline, módulos internos como o register file e a forwarding unit, e na execução de códigos na FPGA.

Nos comentários adicionais, refletimos sobre as dificuldades pessoais que alguns de nós enfrentamos, como os impactos de uma greve, que afetaram nossa colaboração como equipe. Sugerimos que abordagens mais simples nas fases iniciais do projeto poderiam ter sido mais eficazes.

No feedback que queríamos passar ao professor, destacamos o desafio que o projeto representou para um único semestre, sugerindo a necessidade de um apoio teórico e prático mais robusto, além de orientações mais claras sobre o uso de ferramentas específicas, como FPGA e implementações de referência.

Se pudéssemos fazer o projeto novamente, dedicaríamos mais tempo ao componente de memória desde o início, dada a sua complexidade e a necessidade de frequentes refatorações. Além disso, consideraríamos adotar um design mais simples no início, talvez focando em uma CPU multiciclo em vez de uma pipeline completa, para facilitar o desenvolvimento e a implementação.

Essa jornada educacional foi intensa para nós, marcada por aprendizados significativos em hardware e programação. No entanto, também evidenciou a necessidade de um planejamento e suporte mais estruturados em projetos de grande complexidade.

Referências

1. D. A. Patterson and J. L. Hennessy, *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2017.
2. Repositório com código do sitema computacional: https://github.com/izumizawa/mc851