

MC851 - Projeto em Computação I

arRISCado - Entrega 4

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

1. Introdução

Este relatório representa a continuação do trabalho delineado no documento anterior, focando na conclusão do processador e na integração dos periféricos. Nosso objetivo primordial consiste na criação de um ambiente de execução funcional para o processador e seus periféricos em FPGA, destacando o desenvolvimento da cache L1 e a integração dos periféricos ao processador. Em última análise, buscamos alcançar um processador operacional, compatível com o conjunto de instruções RV32IMAC, e dotado de pelo menos 2 periféricos plenamente funcionais.

2. Planejamento

Considerando a fase avançada dos dois periféricos, optamos por concentrar nossos esforços na integração deles ao processador. Essa decisão foi marcada pela transição do Yosys para o GOWIN, necessitando adaptações para que o pipeline pudesse compilar e executar eficientemente na FPGA. Essa mudança proporcionou acesso a 8000 slices na placa, em comparação aos 1000 disponíveis na compilação do Yosys.

Para otimizar a abordagem das pendências identificadas, distribuímos tarefas entre os membros da equipe. João foi designado para implementar o divisor multíciclo, a lógica de stall e as bolhas de execução para branch taken. Ângelo e Kinder assumiram a responsabilidade de corrigir a execução das instruções JAL e JALR. Elton concentrou-se no desenvolvimento do periférico PWM, Iago liderou a implementação do cache, enquanto o Claudio adicionou o suporte para o conjunto de instruções restantes (RV32AC), ficou na parte de verificar se todas as instruções estavam funcionando como deveria, e assumiu a implementação dos códigos de exemplo e teste. A colaboração entre os membros foi crucial para superar desafios e desbloquear suas respectivas tarefas.

3. Desenvolvimento

a. Finalização do Pipeline

Inicialmente, João cuidou da implementação dos sinais de stall, destinados a serem utilizados em situações de resoluções de estado mais complexas, como no acesso à memória ou no cálculo de divisão inteira. Com a implementação bem-sucedida e validada dessa lógica, pudemos avançar para scripts mais sofisticados.

João e Ângelo colaboraram na implementação de um divisor multiciclo, integrando-o à ALU e à Pipeline. Quando o divisor inicia o cálculo, envia um sinal de stall que se propaga por toda a pipeline, afetando estágios anteriores e posteriores ao Execute. Ao concluir a divisão, o sinal de stall é zerado, permitindo o retorno ao funcionamento normal da pipeline.

Ângelo e Kinder corrigiram as instruções de Jump, como JAL e JALR, que anteriormente armazenavam endereços incorretos ao saltar para momentos anteriores do código. Essa correção foi possível graças a um comentário perspicaz do professor Rodolfo sobre a extensão de sinal nos imediatos durante a etapa de Decode das instruções.

Claudio ficou na parte de verificar todas as instruções e adicionar o suporte para as instruções atômicas e compactas. Havia erros com o sinal dos operandos na ALU, foi escolhido receber os operandos sem sinal, e ter registradores que sinalizaram esses operandos. Para cada instrução, foi escolhido se eles receberiam os operandos sem sinal (default) ou com sinal. Quanto às instruções compactas, o processador ao receber elas, trata elas na forma de instruções estendidas e o Iago complementou modificando o fetch para quando identificar instruções compactas, incrementar o PC em 2 ao invés de 4.

b. Implementação do Pipeline na FPGA

Com a mudança do Yosys pro Gowin, algumas mudanças foram feitas em diversas partes do código com o objetivo de garantir o funcionamento usando essa nova ferramenta de compilação. Ainda assim, no geral, a maior mudança foi a permissão de usar mais Slices e de uso de componentes antes inacessíveis pelo Yosys, como Adders e Multipliers inclusos na FPGA que antes eram sintetizadas em tempo de compilação, gastando memória desnecessariamente.

c. Ampliação do conjunto de instruções

Ângelo e Claudio assumiram a responsabilidade de garantir a execução das instruções presentes no conjunto RV32IMAC nos módulos de decodificação, execução e na ALU. Até o momento, as instruções do conjunto RV32C não foram totalmente implementadas, faltando alguns detalhes relacionados ao ciclo do PC, e as instruções de jump; e as instruções de *lb*, *sb*, *lh* e *sh* do conjunto RV32I também estão em falta.

No módulo de decode, o opcode das instruções foi declarado como "localparam" no início do módulo, visando aprimorar a legibilidade do código. Nesta etapa, também foram instanciados os sinais de controle associados a cada instrução da extensão M. Os sinais das instruções de Branch foram refinados e explicitados de maneira mais clara, facilitando a depuração e a compreensão dos erros.

No módulo execute, os sinais de controle AluOP e AluSrc são utilizados para determinar os operandos a serem enviados para a ALU, enquanto na ALU, o sinal AluControl é empregado para determinar a operação aritmética ou lógica a ser executada entre os dois operandos. Os possíveis valores para o AluControl também foram definidos como "localparam" no início do módulo da ALU. Para o caso da divisão, o módulo do divisor

multiciclo, quando ativado, dispara um sinal de Stall para que a pipeline fique parada enquanto ele calcula o resultado.

d. Integração dos Periféricos

Iago realizou a implementação de um módulo “Memory Management Unit” (MMU) na pipeline do processador, que realiza o chaveamento dos sinais para o dispositivo de memória que será lido ou escrito, a Memória ou o Gerenciador de Periféricos.

Elton realizou a validação e correções para o funcionamento na FPGA do módulo PWM com a MMU e pipeline principal. Um pequeno erro continua pendente, onde são necessárias duas vezes a mesma instrução de escrita para escrever no periférico.

Elton e Kinder implementaram o segundo periférico, que realiza a leitura dos botões já integrados na FPGA. Cada botão é mapeado para um endereço de memória, que é incrementado em 1 sempre que ocorre um pulso. São utilizados os endereços com prefixo 010, sendo 010x...x0 o primeiro botão e 010x...x1 o segundo.

e. Implementação da cache L1

Iago ficou responsável pela elaboração da cache L1, cujo progresso pode ser encontrado na branch feat/add-cache2.0 do [Pull Request #78](#). O módulo, atualmente, está pronto para acessos de memória alinhados, embora ainda não tenha sido integrado à branch principal. Um problema que surgiu na leitura é que, por algum bug que não foi encontrado até o presente momento, ela só aceita uma leitura por Flash do Bitstream do código do processador, que claramente causa erros indesejados na execução de código. Por esse motivo, ela não foi integrada na branch principal.

f. Demo

O Kinder implementou uma demonstração com a execução de código em alto nível na FPGA através da UART. Os leds foram configurados para mostrar os valores nos registradores a0 (x10) e a1 (x11), enquanto que os botões foram configurados para alternar entre esses leds. Dessa forma é possível visualizar o resultado das operações na FPGA. Pelas funcionalidades adicionadas aos leds e aos botões a demo não foi incorporada na branch main, é possível testá-la e visualizá-la na branch [demo-uart](#). A demonstração foi realizada com sucesso na aula de 24/11/2023.

Elton e Kinder implementaram uma demonstração com todos os periféricos do processador. O módulo PWM é configurado com um duty cycle de 10 clocks, enquanto que a quantidade de clocks que fica ligado é incrementado com o pressionar de um botão e decrementado com o outro. A demonstração foi realizada com sucesso na aula de 01/12/2023 e o código está incorporado ao branch main do projeto.

4. Pendências

Abaixo segue uma lista de pendências que não foram possíveis de serem concluídas durante a disciplina:

- Finalização da Cache L1;
- Adição de bolhas em branch taken (lógica existe, mas gera alta impedância e teve que ser removida);
- Cálculo correto do PC para instruções compactas de Jump;

5. Aprendizados

Ao longo deste último mês conseguimos avançar nosso conhecimento no tópicos listados abaixo:

- Gerenciamento de Projeto: A experiência de coordenar a equipe na atribuição de tarefas específicas, como a implementação do divisor multíciclo, correção de instruções e desenvolvimento de periféricos, ressalta a importância do gerenciamento eficiente do projeto. A divisão de responsabilidades contribuiu para melhorar nosso progresso, maximizando as habilidades individuais dos membros.
- Adaptação a Novas Tecnologias: A transição do Yosys para o GOWIN destacou nossa capacidade de nos adaptarmos à evolução das tecnologias. A mudança exigiu ajustes no pipeline para compilar eficientemente na FPGA, ressaltando a importância de se manter atualizado e flexível diante das ferramentas disponíveis.
- Simulação e Testes: A validação do comportamento dos periféricos em simulação e os desafios enfrentados na ativação dos dispositivos de saída realçam a importância de testes e validações abrangentes. A utilização de diversas ferramentas, como Yosys, iverilog, GTKWave e o próprio Gowin, contribuiu para uma compreensão mais completa e refinada do nosso sistema. O uso dos LEDs integrados na FPGA foi ampliado para os testes com o hardware, verificando registradores e sinais específicos dentro e entre os estágios. O uso de dispositivos auxiliares, como um osciloscópio, também foi realizado.
- Estudo de Toolchain: A mudança de foco de Iago da cache para o estudo da toolchain da Gowin destaca a necessidade de entender e adaptar-se às ferramentas disponíveis para garantir uma implementação eficiente do código em um ambiente de software proprietário. Esse aprendizado evidencia nossa habilidade de se ajustar às demandas do ambiente de desenvolvimento.

- Desenvolvimento Incremental: O progresso incremental, como o desenvolvimento da cache L1, reflete a abordagem estruturada e progressiva que adotamos no projeto. Focar em módulos específicos antes da integração completa permitiu um desenvolvimento mais controlado e eficiente.
- Aprofundamento em Verilog: A utilização de "localparam" no início dos módulos para declarar opcodes e valores de controle destaca nossa busca contínua pela melhoria da legibilidade do código-fonte. Além disso, a implementação de estruturas complexas, como a cache, o gerenciador de periféricos e o pipeline com instruções multiciclo, impulsionou nosso aprofundamento nos conhecimentos de Verilog e de arquitetura.