Entrega 1 - Relatório

Grupo - Popullacho (Placa 4)

Alexandre Ladeira	212328
Gabriel Pallotta Cardenete	216392
Igor Fernando Mandello	236769
Júlia Alves De Arruda	238077
Lucas Hideki Carvalho Dinnouti	220792
Vinícius Waki Teles	257390

Organização

Para facilitar e acelerar o desenvolvimento nos separamos em 2 grupos, sendo o grupo A composto polo Alexandre, Gabriel e Lucas, e grupo B pelo Igor, Vinícius e Júlia.

Em termos gerais, o grupo A ficou encarregado de configurar o ambiente para execução do código Verilog em simulador, enquanto que o grupo B ficou encarregado de preparar o ambiente de execução na FPGA.

Desenvolvimento do projeto

Para a execução sem a placa, inicialmente testamos o Verilator, mas tivemos problemas com a execução de testes e geração de arquivos VCD (visualização de sinais de onda). Então, decidimos utilizar o Icarus Verilog, que se mostrou mais simples do que o Verilator, e também permitiu a execução de testes, com a escrita automática de arquivos VCD para debugging.

Para permitir a implementação concorrente dos diferentes módulos e garantir uma mesma interface, nós criamos um diagrama compartilhado, inicialmente com um overview da arquitetura, e depois conforme foi evoluindo com o código, se tornou uma especificação do nosso projeto, contendo uma documentação das entradas e saídas de cada módulo (adicionamos no repositório uma imagem dele em seu estado atual).

Depois, pensando na primeira entrega, encarregamos o grupo A da implementação dos componentes de ALU, decodificação de instrução e banco de registradores, que podem ser executados e testados no simulador, e o grupo B ficou responsável pela implementação do módulo de interface com a memória flash da placa, e módulos de controle de memória e fluxo de execução (program counter).

Para testar os diferentes módulos do processador no ambiente de simulador, criamos testes unitários em arquivos *testbench*, utilizando asserções para validar os valores de saídas de cada módulo para diferentes entradas. Por exemplo, para o módulo de decoder, criamos um teste de decodificação para cada instrução. Além disso, também criamos uma CPU com um *mock* da memória, para permitir a execução da CPU completa no ambiente com simulador. A execução desses testes atualmente pode ser feita a partir do script test.sh presente na pasta raíz do repositório.

Criamos um programa de testes com algumas instruções simples de operações aritméticas, de forma a testar a integração com a placa. Esse programa se encontra no nosso repositório. Utilizamos o site https://riscvasm.lucasteske.dev para gerar a representação em binário das instruções a partir do código assembly, de forma a possibilitar que ele seja carregado na memória flash ou na memória *mock*. Uma melhoria para o futuro é termos um script no nosso repositório para realizar a conversão para binário automaticamente para programas de teste, utilizando a toolchain de compilação do RISC-V.

Também testamos usando a memória Flash da placa. Utilizando o programa em binário descrito acima, averiguamos se a placa estava lendo as instruções em ordem através dos 6 LEDs disponíveis, representando 6 bits de nossa escolha da instrução atual.

Conectamos o *program counter* e o registrador x1 com os 6 leds da placa (1 para o PC e os outros 5 para o registrador). Tivemos um problema com essa conexão direta com os LEDs, que não permitia a síntese do circuito na FPGA. O problema se deu por conta da tentativa de síntese de um D-Latch, que caia na verificação de “DFF-Legalize”. Descobrimos que o que estava causando o problema era a falta de um valor *default* no switch-case da ALU. A princípio, colocar um clock no circuito da ALU resolveu o problema, mas decidimos removê-lo ao descobrir que o problema poderia ser resolvido adicionando o caso *default* (result = 0).

Lições aprendidas

Tivemos diversos aprendizados durante esse projeto. O mais básico deles foi a sintetização em FPGA, a qual não era conhecida por ninguém no grupo.

Especificamente, aprendemos técnicas de simulação e teste e as diferentes abordagens para validação. O simulador *Verilator* possibilitou testes mais dinâmicos sem a necessidade da placa. Apesar da utilidade desse ambiente de simulação, também foi importante investirmos tempo nos testes com a placa, pois nela o paradigma é outro e nem tudo funciona da mesma forma. Na placa, utilizamos os LEDs como forma de debug e os botões como input adicional. Também tivemos o apoio da extensão digital JS que nos permitiu pré-visualizar uma aproximação do circuito a ser sintetizado.

Em relação ao projeto em si, aprendemos a importância dos valores padrão nos *switch-cases* já que durante a sintetização um D-Latch era adicionado para salvar o valor anterior.

Próximos passos

Os próximos passos para o projeto seriam a unificação dos 2 módulos de CPU que atualmente estão separados (CPU mock que roda no simulador e CPU que roda na placa), a implementação das barreiras de pipeline e o suporte às instruções faltantes (load, store e instruções branch).