PCS-3225 Sistemas Digitais II — 12 de Junho de 2024 - v1

Barramentos

Aula 23 - Exercício

Bruno Albertini, Edson Gomi e Ricardo Lera

O objetivo deste exercício é demonstrar uma aplicação do protocolo (simplificado) AMBA de comunicação para um barramento entre CPU, RAM e ROM.

Três módulos de componentes, `cpu`, `ram` e `rom`, são conectados pelo módulo `mboard`. A CPU (componente ativo) pode se comunicar com as memórias (componentes reativos) por via dos barramentos de Endereço (`addr`), Dados (`rdata`, `wdata`) e Controle (`clk`, `ready`, `write`). Ela também possui dois sinais de controle externos (`rst`, `done`).

Esta CPU é uma versão simplificada do processador RISC-V, e é capaz de processar 4 instruções: ADD, LW, SW e EBREAK. Esta última instrução, em um processador completo, passa controle para o *debugger*. Vamos utilizá-la para setar o valor de `done` e pausar a execução.

Cada instrução requer 4 etapas para ser processada pela CPU (4 estados de máquina), cada uma durando pelo menos 1 período de *clock*. São estes: *fetch, decode, execute* e *write*, respectivamente. Nosso objetivo é implementar estes quatro estados utilizando os barramentos para se comunicar com as memórias, a fim de executar as instruções suportadas pelo sistema.

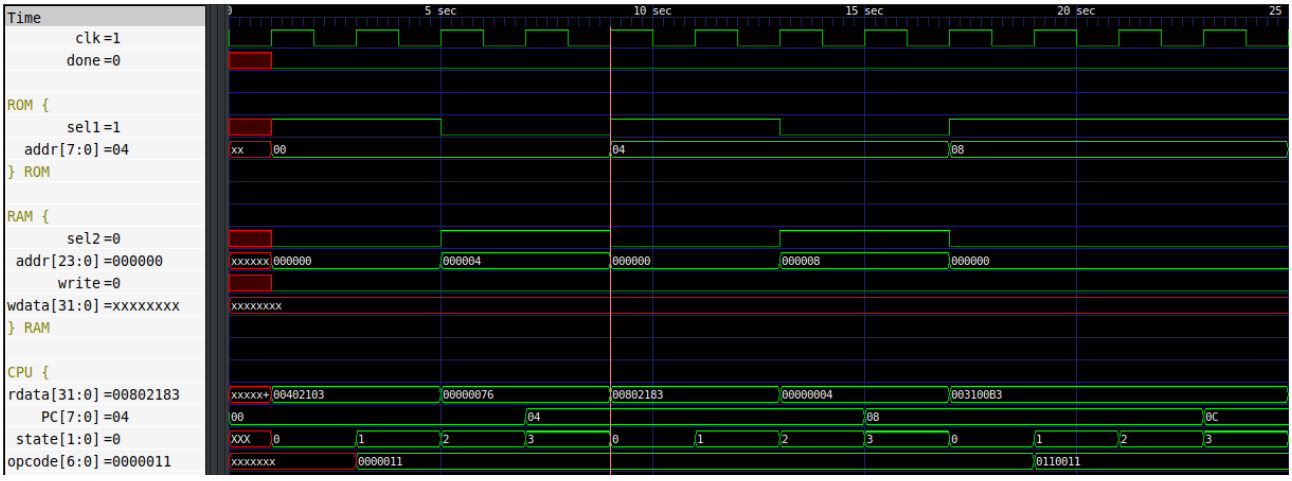

Figura 1: Testbench exemplificando o funcionamento do sistema

No ediscIplinas, foi disponibilizado um *template* `cpu.v`. O estado *decode* já foi preenchido. Complete, então, os outros três estados para que se adéquem às quatro operações suportadas pelo sistema (ADD, LW, SW, EBREAK).

Recurso importante: RISC-V Encoder/Decoder `https://luplab.gitlab.io/rvcodecjs/`

Também foram disponibilizados os arquivos `ram.v`, `rom.v` e `mboard.v`, que já estão completos, mas podem ser utilizados para testes (neste caso será necessário escrever um arquivo `rom.hex` em *assembly*). Envie para o juiz somente o módulo `cpu`.

Detalhes de Implementação:

- O barramento de endereço `addr` possui **32 bits**: os 8 menos significativos (`addr[7:0]`) endereçam a ROM, e os 24 mais significativos (`addr[31:8]`) endereçam a RAM.
- Como o barramento de dados é comum para todos os dispositivos, o decodificador de endereços (parte do módulo `mboard`) **seleciona a ROM quando addr < 0x00000100**, ou seja, quando **todos** os bits em `addr[31:8]` forem 0. Senão, ele seleciona a RAM.
- O endereçamento das memórias ROM e RAM é **por byte**. Portanto, o incremento usual do Program Counter é `PC = PC + 4`.
- Lembre de utilizar o atribuidor síncrono `<=` em máquinas de estado, para assegurar que seu programa responderá adequadamente.