Milestone 3 — Computer Architecture

Design of Pipelined RISC-V Processor

Hai Cao

rev 1.0.0

Contents

Abstract

This document provides an overview of the tasks, expectations, and requirements for the second milestone in the Computer Architecture course. It details the specific components and specifications students must follow to successfully design three models of pipelined RV32I processor. For any errors found or suggestions for enhancement, contact the TA via email at cxhai.sdh221@hcmut.edu.vn using the subject line "**[CA203 FEEDBACK]**".

1 Objectives

- Review understanding of pipeline techniques
- Design a pipelined RV32I processor
- Explore the Branch Prediction technique

2 Overview

In this milestone, students are tasked with designing at least 3 models of pipelined RV32I processor, as discussed in lectures. To enable communication between your custom processor (soft-core) and external peripherals, some modifications to the standard processor design are necessary. The design must adhere to the specifications outlined below and will be tested against both student-created testbenches and a comprehensive testbench provided by the TA.

2.1 Processor Specification

- **Top-level module:** `pipelined.sv`
- **I/O ports:**

Signal name	Width	Direction	Description
`i_clk`	1	input	Global clock, active on the rising edge.
`i_rst_n`	1	input	Global low active reset.
`o_pc_debug`	32	output	Debug program counter.
`o_insn_vld`	1	output	Instruction valid.
`o_mispred`	1	output	Mispredict.
`o_io_ledr`	32	output	Output for driving red LEDs.
`o_io_ledg`	32	output	Output for driving green LEDs.
`o_io_hex0..7`	7	output	Output for driving 7-segment LED displays.
`o_io_lcd`	32	output	Output for driving the LCD register.
`i_io_sw`	32	input	Input for switches.
`i_io_btn`	4	input	Input for buttons.

2.2 General Design Guidelines

2.2.1 Directory structure

The project should maintain a well-organized directory hierarchy for efficient management and submission:

```
.
|-- 00_src          # Verilog source files
|-- 01_bench       # Testbench files
|-- 02_test        # Testing files
|   |-- asm        # Assembly test code
|   `-- dump       # Binary/hex dump files
|-- 10_sim         # Simulation files
|-- 20_syn         # Synthesis files
|   `-- quartus
|       |-- run    # Makefile for synthesis
|       `-- src    # Source files specific to synthesis
`-- 99_doc         # Documentation files
```

3 Pipelined Models

3.1 Nonforwarding

In this model, students will design a pipelined processor without any bypassing or forwarding network. They will need to implement a hazard detection mechanism to inject nop whenever necessary.

3.2 Forwarding

Building on the previous model, students will now need to implement the forwarding network to enhance the processor's performance. They can choose to insert nop or predict not-taken when handling branch instructions.

3.3 Dynamic Prediction

Based on the previous model, a branch target buffer (BTB) is integrated. For the predictor, due to the simplicity but clear enhancement of the two-bit scheme, students should choose between two of the two-bit schemes.

3.4 G-share

Instead of a two-bit predictor, this model replaces it with a branch history table (BHT) and a pattern register to create a G-share predictor.

3.5 Requirements

Students will need to design three out of the five models below, with the first two being mandatory.

For models without branch prediction, the pin `o_mispred` should be asserted (1'b1). Otherwise, students need to wire this pin to the module which is capable of detecting whether the predicted PC is correct. This signal has to accompany its PC to indicate which instruction it's for – `pc_debug` and `mispred` travel through stage flipflops to WB stage.

Also, `insn_vld` will work the same. By doing so, students can measure their processors' performance, for this signal will be deasserted whenever NOP is injected. Remember to keep this signal stick to its relative PC.

4 Verification

A comprehensive testbench is crucial for verifying the functionality of your processor. Your testbench should cover a wide range of scenarios and corner cases to ensure thorough testing. The TA will provide a final, comprehensive testbench for further validation one week before the presentation. You are expected to create your testbenches and verify your design before this point.

5 I/O System Conventions

For consistent operation and testing, adhere to the following conventions for setting up with DE2 boards and interacting with the I/O system.

- **LEDs** Use the output ports `o_io_ledr` and `o_io_ledg` to control the red and green LEDs, respectively. These can be used for status indicators or debugging.

  `o_io_ledr`

Bits	Usage
`31 - 17`	(Reserved)
`16 - 0`	17-bit data connected to the array of 17 red LEDs in order.

  `o_io_ledg`

Bits	Usage
`31 - 8`	(Reserved)
`7 - 0`	8-bit data connected to the array of 8 green LEDs in order.

- **Seven-Segment** Utilize `o_io_hex0..7` to display numerical values or messages. Each port has 7 bits in total, so four of them can represent a 32-bit data as shown below. To control each seven-segment display, stores a byte at the corresponding address, such as `SB` at `0x7022` will change the value of HEX2, while `SH` at the same location will affect both HEX2 and HEX3. For the case of misaligned address, your assumption is critical.

Address	**0x7020**
Bits	**Usage**
`31`	(Reserved)
`30 - 24`	7-bit data to HEX3.
`23`	(Reserved)
`22 - 16`	7-bit data to HEX2.
`15`	(Reserved)
`14 - 8`	7-bit data to HEX1.
`7`	(Reserved)
`6 - 0`	7-bit data to HEX0.
Address	**0x7024**
Bits	**Usage**
`31`	(Reserved)
`30 - 24`	7-bit data to HEX7.
`23`	(Reserved)
`22 - 16`	7-bit data to HEX6.
`15`	(Reserved)
`14 - 8`	7-bit data to HEX5.
`7`	(Reserved)
`6 - 0`	7-bit data to HEX4.

- **LCD Display** Manage more complex visual output through `o_io_lcd`. To drive LCD properly, visit this link to investigate the specification of LCD HD44780.

Bits	**Usage**
`31`	ON
`30 - 11`	(Reserved)
`10`	EN
`9`	RS
`8`	R/W
`7 - 0`	Data.

- **Switches** Use `i_io_sw` to receive input from external switches, which can be used for user interaction or control signals.

Bits	**Usage**
`31 - 18`	(Reserved)
`17`	Reset.
`16 - 0`	17-bit data from SW16 to SW0 respectively.

6 Applications

Develop an application that utilizes the designed processor, demonstrating its capabilities and practical use. The complexity and innovation of the application will impact your grading. Simple applications might include basic input/output handling, while more advanced applications could involve complex calculations or data processing.

Below are some example programs with its expected score:

1 pt	Design a stopwatch using seven-segment LEDs as the display.
1 pt	Convert a hexadecimal number to a decimal number and display on seven-segment LEDs.
1.5 pts	Convert a hexadecimal number to its decimal and binary forms and display on LCD.
2 pts	Input 3 2-D coordinates of A, B, and C. Determine which point, A or B, is closer to C using LCD as the display.

7 Rubric

Your project will be evaluated based on the following criteria:

For Undergraduate

1. **Baseline Submission – 5 pts:** Nonforwarding and Forwarding
   If your design successfully passes all test cases, you will receive 5 points. Submit your source code to the server for evaluation. In-person presentation is not required. If you choose not to present, indicate this on the Presentation sheet.

2. **Demonstration – 2 pts:** For an additional 2 points, you may demonstrate your project. Set up your RISC-V processor on the DE2 board and execute a pre-compiled program. The level of creativity and sophistication demonstrated will be considered in determining the score.

3. **Branch Prediction – 2 pts:** For an additional 2 points, you may implement Branch Prediction. The complexity and accuracy of your prediction algorithm will be evaluated. Higher scores will be awarded for more sophisticated and reliable methods.

4. **Advanced or Alternative Design – 4 pts:** For an additional 4 points, you may undertake a substaintial modification of your processor. The level of innovation and functionality incorporated into your changes will be assessed. Higher scores will be awarded for more substantial and effective modifications.

For Graduate

1. **Baseline Submission – 5 pts:** Nonforwarding, Forwarding, and Always-Taken If your design successfully passes all test cases, you will receive 5 points. Submit your source code to the server for evaluation. In-person presentation is not required. If you choose not to present, indicate this on the Presentation sheet.

2. **Branch Prediction – 2 pts:** For an additional 2 points, you may implement Branch Prediction. The complexity and accuracy of your prediction algorithm will be evaluated. Higher scores will be awarded for more sophisticated and reliable methods.

3. **Cache – 2 pts:** If you integrate a Cache into your RISC-V design, you can earn up to 2 additional points. The sophistication of the cache and its performance will be assessed to determine the number of points awarded.

4. **Advanced or Alternative Design – 4 pts:** For an additional 4 points, you may undertake a substaintial modification of your processor. The level of innovation and functionality incorporated into your changes will be assessed. Higher scores will be awarded for more substantial and effective modifications.

7.1 Report

A comprehensive project report must be submitted, detailing the design process, challenges faced, and solutions implemented. Refer to the report guidelines on Google Drive. The report should be clear and concise, with sections for introduction, methodology, results, and conclusions. Visual aids such as diagrams and charts are encouraged to illustrate key points.