Politecnico di Torino
III Facoltà di Ingegneria

Lab 3 Report
Integrated Systems Architecture

Master degree in Computer Engineering

Authors: ISA36

Nicole Dai Prà s274501, Leonardo Izzi s278564

January 10, 2021

Many thanks to Prof. Mariagrazia Graziano for providing us with this template.

Contents

CHAPTER 1

Introduction

In this lab we had to develop a scalar, in-order, RISC-V-lite processor. To the base unit, as a second task, we had to add a special functional unit to compute the absolute value of a number.

As required, there is a GitHub repository available at the following link: `https://github.com/leoizzi/isa_labs/tree/main/lab3`.

The processors are separated in two different folders, in `riscv` there is the base model while in `riscv_abs` there is the extended one. The folders share the same structure, hence it is reported only once:

- `sim`: it contains all the simulation-related files, such as the data to initialize the RAM and the ROM, as well as the simulation script.
- `src`: it contains the processor's source files to be synthesized and simulated.
- `syn`: it contains the synthesis script and the synthesis results.
- `tb`: it contains all the testbenches developed to test the units and the processor itself.

CHAPTER 2

Processor architecture

In this chapter we will describe how we have implemented the processor. It is based on the structure studied during the lectures. At a glance, it has 5 stages, the control unit is hardwired and it does not support forwarding.

2.1 IF stage

In this stage new instructions to be executed are fetched from the ROM. Its structure is pretty simple: there is the program counter register (PC) that is updated at every clock cycle either with the target address, that is the address of a jump, or with $PC + 4$. In case of a stall, the enable signal of the register is de-asserted.

The current PC value is used to access a small ROM which is used only during simulations as it cannot be synthesized with the processor. The current PC value, $PC + 4$ and the ROM output are fed to the IF/ID registers.

2.2 ID stage

Here the decode of the instruction fetched in the previous cycle is performed. The register file, as well as the immediate generator, are in this stage. The register file is accessed regardless of the instruction type, and the control unit decides how and if the immediate should be extended. These values, along with the instruction PC, its subsequent value and the destination register address, are propagated to the ID/EXE registers.

The register file is capable of supporting two read and a single write per clock cycle. The write operation is done by the WB stage on the falling edge of the clock signal. This means that it is possible to reduce the number of stalls by one, because when an instruction writes in the RF from the WB stage its result can be immediately accessed by the instruction in the ID stage.

To allow the control unit to check whether there is the need for a stall, the source registers are sent to it.

2.3 EXE stage

The ALU supports two inputs, A and B. Based on the instruction, two muxes select which values have to be taken as operands. For the A port, the choice is between the PC, $PC + 4$, 0 or the value coming from the RF's port 1. For the B port, the choice is between the immediate, the RF's port 2

or 0. In the ALU, all the functional units perform the computation in parallel, then through a mux the result is selected. The functional units present in the ALU are:

- the adder;
- the shifter;
- the logicals (AND and XOR);
- the set-less-than;
- the absolute value (only for `risc_abs`).

Moreover, there is also a comparator that checks if the result is 0. In the affirmative case, it assert the `zero` signal.

To support branches and jumps there is a second adder, that takes as inputs the PC and the immediate value shifted to the left by one. This value represent the target address of the IF stage. However, it is selected by the IF stage only if two conditions are true: in the control word the bit `JMP_ENABLE` is set to 1, and if

- the instruction is a `jal` (detected by looking at the mux control signals, that is $A = NPC$ and $B = 0$)
- the instruction is a `beq`, hence if the `zero` signal is asserted.

Finally, to support `store` operations, there is a direct connection between the second register port output and the EXE/MEM registers.

2.4 MEM stage

All instructions but `lw` and `sw` bypass the memory and their data is directly propagated to the MEM/WB registers. The two aforementioned instructions, instead, are able to respectively read or write the RAM, which as for the ROM is not synthesized but only simulated.

The RAM has a port dedicated to read operations and one dedicated to write operations, but only one of them can be executed at any time. This has been done only to simplify the testing, since in this way there is no risk to misuse the tristate transistors. To distinguish between read and write operations a dedicated `wr` signal has been added.

Since the given test file assumes that the memory has been preloaded with the `.data` segment, this RAM on reset reads its initialization content from a file.

2.5 WB stage

In this stage there is simply a mux that, based on the instruction, selects if in the RF should be written the data coming either from the ALU or the memory. The CU keeps track of whether the instruction is allowed to write content in the RF or not, since the RF has an additional signal that enables the write operation.

2.6 Control unit

The control unit is hardwired, as based on the instruction read in the IF stage it loads all the control signals inside the registers and outputs them at the right time. This is achieved by pipelining the control word in the same way as the datapath.

In the ID stage it detects possible data hazards by looking at the data source addresses and the destination address in the ID/EXE and EXE/MEM pipelining registers. If an hazard is detected, the control unit stalls the processor by inserting NOPs in the proper stages.

The control unit is also notified whenever the datapath is about to jump, because it needs to flush the IF and ID stages, as their work is not needed anymore.

CHAPTER 3

Synthesis

For the synthesis we have followed the standard flow used in the previous laboratory sessions, hence we will not repeat it here.

We synthesized both the processors. The critical path lies in the EXE's adder, hence they have a very similar critical path and requires 2.7 ns to be traversed.

Even the area occupation is similar, as the functional unit added to `riscv_abs` is rather small. Indeed, the total area difference between the two versions is $\sim 2\%$. Below are reported the area results.

```
riscv

****************************************
Report : area
Design : core
Version: O-2018.06-SP4
Date   : Sun Jan 10 18:18:58 2021
****************************************

Number of ports:                          561
Number of nets:                          7720
Number of cells:                         6242
Number of combinational cells:           4783
Number of sequential cells:              1446
Number of macros/black boxes:               0
Number of buf/inv:                        839
Number of references:                      36

Combinational area:               5745.334081
Buf/Inv area:                      597.435990
Noncombinational area:            6538.811763
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 12284.145844
Total area:                 undefined
```

```
riscv_abs

****************************************
Report : area
Design : core
Version: O-2018.06-SP4
Date   : Sun Jan 10 18:44:01 2021
****************************************

Number of ports:                          659
Number of nets:                          7997
Number of cells:                         6452
Number of combinational cells:           4989
Number of sequential cells:              1448
Number of macros/black boxes:               0
Number of buf/inv:                        962
Number of references:                      36

Combinational area:               5945.366083
Buf/Inv area:                      667.659989
Noncombinational area:            6547.855762
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 12493.221846
Total area:                 undefined
```

If interested, the reader can find the actual results in the respective `syn` folders.