NTNU

TDT4255 - Computer Design

Exercise 1

Peter Aaser, Anders Lima & Øyvind Robertsen

October 14, 2015

Abstract

This report outlines our solution to Exercise 1 - Simple Multi-cycle MIPS processor for the subject TDT4255 - Computer Design. The goal of the exercise is to implement a simple multi-cycle processor in VHDL. The processor should support a subset of the MIPS instruction set, a typical RISC instruction set.

This report gives a slightly more thorough description of the exercise task as well as an in-depth description of the technicalities of our implementation and descriptions of the design, implementation and testing processes.

The result of our work on this exercise is a functioning implementation of the requested processor architecture, which has been verified in software simulation by testbenches and tested on an FPGA.

Contents

1 Introduction

1.1 Exercise Description

The goal of this exercise, as described in the course compendium [1], is to implement a multi-cycle processor in VHDL, supporting a subset of the MIPS instruction set. Some modules (data / instruction memory and `hostcomm`-communication) is handed out by the course staff. A top level module (`MIPSSystem`) has also been provided, to facilitate testing on the FPGA. The top level system architecture is illustrated in figure 1.1.

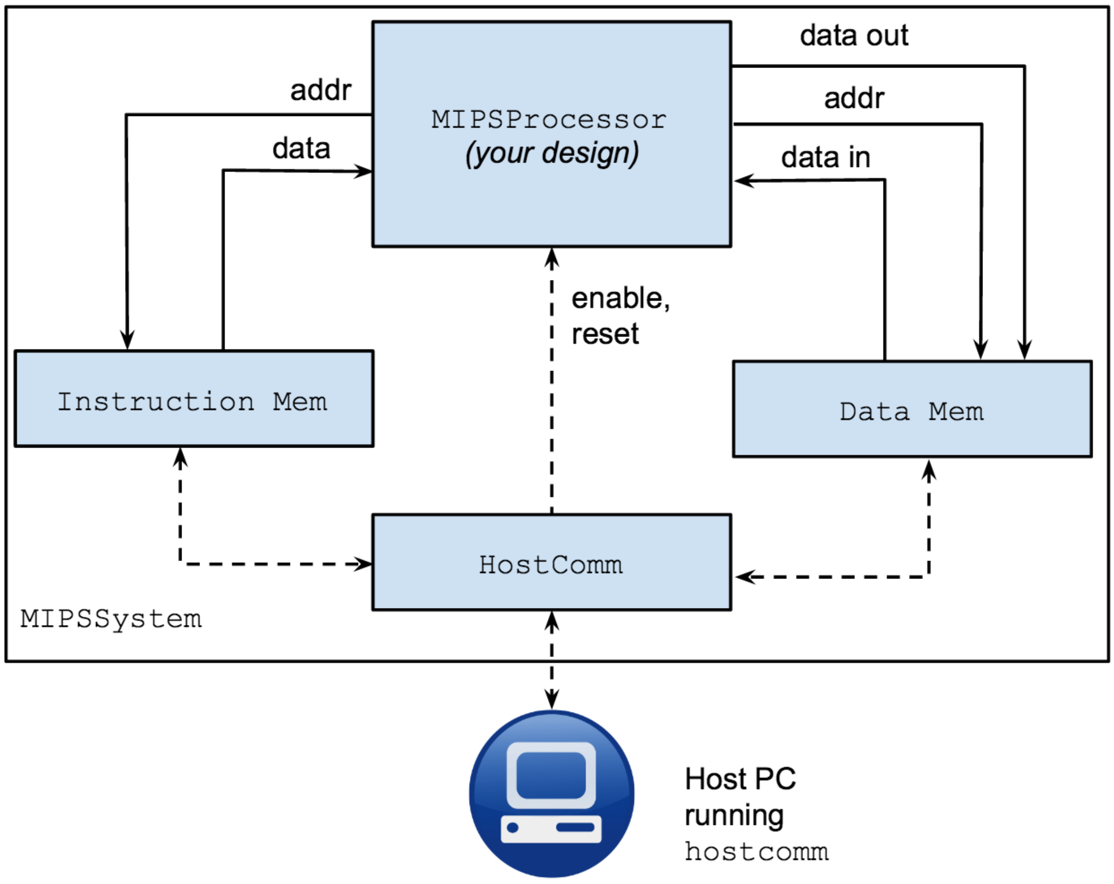

Figure 1.1: The top-level architecture. Figure taken from the compendium. [1, p. 48]

Our job in this exercise is to implement the MIPSProcessor module in figure 1.1 as a multi-cycle, MIPS-like processor. The following categories of MIPS instructions are required to be implemented:

- ALU operations - ADD, SUB, SLT, AND, OR
- Conditional branch - BEQ
- LOAD and STORE
- Load immediate - LDI
- Jump - JMP

1.2 Approach

We chose to start our work by sketching RTL designs based on the microarchitecture suggested by the course staff in the compendium [1, p. 45]. Based on these sketches, we wrote VHDL to implement each of the following modules: Program counter, Register and ALU. We also wrote testbenches for each of the modules, allowing us to test them independently from the system as a whole. These components were then wired together in the MIPSProcessor module. Signals going into and out of the MIPSProcessor were also routed to their appropriate submodules. The final piece of the puzzle is the Control module, which we implemented based on figure 4.2 in the compendium [1, p. 46].

2 Solution

3 | Results

3.1 Discussion

3.2 Discussion

4 | Conclusion

References

[1] TDT4255 course staff. Lab exercises in tdt4255 computer design. Technical report, NTNU, 2015.

[2] Mathias Ose. ma.thiaso.se. `http://ma.thiaso.se/`, 2014.