COE608 Computer Organization and Architectures
Winter 2017
Lab 6: The Complete CPU (Overall Project)

Due Date: Lab6 Part I & II -Week 12 (During the Lab Session)
Bonus -Week 13

1. Overview

In this final lab project, a complete CPU will be implemented whose main components datapath and control have been designed and implemented in the previous labs 4b and 5. Students are to combine the control unit and data-path with a reset circuit (more on this below). When complete, the over-all design will be able to implement the features described in the *CPU specification document*. Students are encouraged to consult this specification document while proceeding to test the CPU. The instruction memory unit (VHDL) and overall CPU testing block diagrams files to complete this lab are provided as follows: 1) The files and specifics for testing and setting up the CPU for simulation can be found in a document located in …/courses/coe608/labs/lab6/* and 2) for bonus marks involving CPU hardware implementation and emulation, the specifications and documentation can be found in …/courses/coe608/labs/bonus/*. The rest of this lab presents the reset circuit needed for the CPU.

2. Part I - CPU Reset Circuitry

In order for the CPU developed here to work properly it must incorporate a reset circuit. The block diagram of the reset circuit is illustrated in Figure 1.

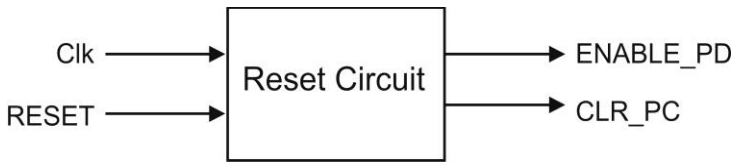

Figure 1: Reset Circuit

The reset circuit works as explained here. When RESET signal goes high, ENABLE_PD goes low that forces the control unit into state T0 and CLR_PC goes high, which clears the Program Counter. We know that the CPU program starts in memory at location 0x00000000.

When RESET goes low, ENABLE_PD & CLR_PC remains low & high respectively for 4 clock cycles. This allows the data surrounding the CPU to stabilize before its operation begins. The reset circuit is required to keep track (count) of the three clock cycles (T0, T1, and T2). This reset circuit can either be implemented asynchronously or synchronously. The synchronous waveform is shown in Figure 2.

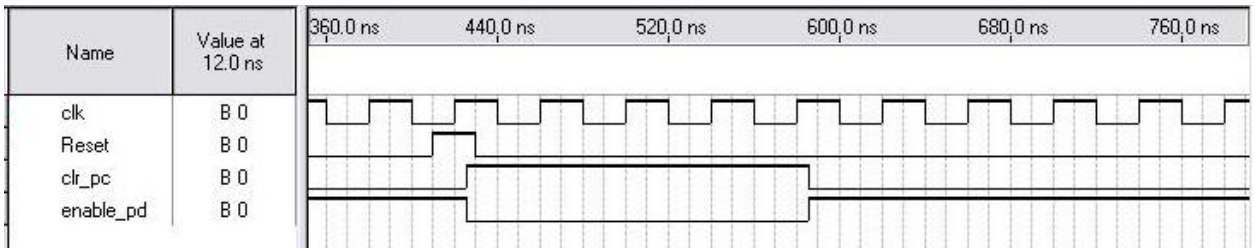

Figure 2: Reset Circuit Operation

3. VHDL Implementation

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.std_logic_unsigned.ALL;

ENTITY reset_circuit IS
        PORT
        (
                Reset :         IN STD_LOGIC;
                Clk :           IN STD_LOGIC;
                Enable_PD :     OUT STD_LOGIC;
                Clr_PC :        OUT STD_LOGIC
        );
END reset_circuit;

ARCHITECTURE description OF reset_circuit IS BEGIN
-- you fill in what goes here.
END description;
```

Students are free to implement the ARCHITECTURE section however they see fit.

4. Part I - What to Hand In

Students must submit the following to obtain full marks for Lab 6, Part I:

- A hard-copy listing of your VHDL source code implementation.
- A hard-copy printout of the timing simulation results for the reset circuit.

The lab instructor will quiz you on both Part I (reset circuit) and Part II (final CPU) for the Lab 6 demo.

5. Part II - The Complete CPU System

Once the reset circuit is implemented, all the CPU sub-systems are complete and the CPU can be assembled. The final CPU will consist of one instance of the data-path, the control unit, and the reset circuit. Interconnecting them appropriately is up to the students, however supporting files for VHDL interconnection and setup may be found in the course directory …/courses/coe608/labs/lab6/ in the cpu1 module. To help better understand the expectations and functionality of the final CPU, students are encouraged to refer to the **CPU Testing** document available in the course directory and website. This document will help you to:

- Generate the system's instruction memory unit (MegaCore RAM block (.mif) implementation)
- Assemble a top level working file (with reset circuit, instruction memory, a datapath, and control unit)
- Include and map other supporting files
- Simulate and demo your working CPU for Lab 6
- And optionally emulate the CPU for the Lab 6 Bonus

Students are advised to also refer to the CPU Specification document to ensure that all operations and specifications have been fulfilled by their final CPU.

6. Part II - What to Hand in

To obtain full marks for the complete CPU project (i.e. Lab6), students must demonstrate the correct operation of CPU circuit through simulation. This means that students have to demonstrate the timing simulation results that show the CPU correctly loading ALL the instructions and data, performing addition, load upper immediate, etc.

In addition to this, students must submit the VHDL code for their CPU, as well as timing simulation results for the complete CPU. To properly simulate the CPU operation, the *CPU_testing* document should be consulted, and Memory Module files provided should be used. Your lab supervisor will quiz you during the demo for both Parts I and II.

7. Bonus Project

To obtain bonus marks, students are required to demonstrate the operation of the processor through emulation on the DE2 boards found in the laboratory. To properly implement the CPU, the **CPU_testing** document should be consulted and system memory, seven-segment, display-unit, decoder, and all the other VHDL file provided in the following course directory should be used.
…/courses/coe608/labs/bonus/*

The following problems are provided as a bonus. Solving them will result in additional marks being added to your course mark.

Problem 1:
The processor developed throughout this course includes various branch and jump operations used for conditional statements. Using the built in mnemonics/ instruction set, find a way to implement the following:

```
IF(a == b){
        branch1();
}
else if(a > b){
        branch2();
}
else{
        continue;
}
end IF;
```

Problem 2:
Implement the following code using your CPU and its instruction set:

```
a = 1;
for(i = 1; i < 6; i++){
        a = a*2^i
}
```

To obtain the bonus, the assembly code for both of the problems above must be submitted, and the programs in question must be demonstrated on the DE2 boards in the laboratory.