Concordia University
Department of Electrical and Computer Engineering
COEN 316 Computer Architecture
Lab 5 - Datapath/Control Unit Integration and system testing
Fall 2021

Introduction

The datapath designed in the previous lab contains 10 control signals. The function of the control unit, which is the focus of this lab, is to produce the correct values for all the control signals at the proper point in time. Tables 1 and 2 lists the 10 control signals and summarizes their operation.

Table 1: Single bit control signals.

Control signal	value = 0	value = 1
reg_write	do not write into register file	write into register file
reg_dst	rt is the destination register	rd is the destination register
reg_in_src	d_out of data_cache is the d_in to the register file	ALU output is the d_in to the register file
alu_src	out_b of register file (rt) is the y input of the ALU	sign extended immediate is the y input of the ALU
add_sub	ALU operation = addition	ALU operation = subtraction
data_write	do not write into data cache	write into data cache

Table 2: Two bit control signals.

Control signal	value = 00	value = 01	value =10	value = 11
logic_func	AND	OR	XOR	NOR
func	load upper immediate	set less	arithmetic	logic
branch_type	no branch	beq	bne	bltz
pc_sel	no jump (PC+1, or PC+target address if branch condition is true)	jump (PC = target address)	jump register (PC = rs)	not used

Table 3 lists the 20 instructions implemented by the CPU together with the values of the 6 bit opcode field and the 6 bit func field (contained within the instruction as per Figure 1 of Lab 4) together with the 10 control signals. Table 3 is **partially** completed, you are to **complete** the table by deriving the values of the 10 control signals based upon Tables 1 and 2 and knowledge of which control signals need to be activated during a particular instruction in order to achieve correct execution of the instruction. Refer to your datapath of Lab 4 to assist in completing the table.

Table 3: 20 instructions with opcode and function fields and control signals. [1]

Inst.	op	func	reg_write	reg_dst	reg_in_src	alu_src	add_sub	data_write	logic_func	func	branch_type	pc_sel
lui	001111		1	0	1	1	0 (don't care)	0	00 (don't care)	00	00	00
add	000000	100000	1	1	1	0	0	0	00	10	00	00
sub	000000	100010										
slt	000000	101010										
addi	001000		1	0	1	1	0	0	00	10	00	00
slti	001010											
and	000000	100100	1	1	1	0	1	0	00	11	00	00
or	000000	100101										
xor	000000	100110										
nor	000000	100111										
andi	001100											
ori	001101											
xori	001110											
lw	100011		1	0	0	1	0	0	10 (don't care)	10	00	00
sw	101011											
j	000010											
jr	000000	001000										
bltz	000001											
beq	000100		0	0	0	0	0	0	00	00	01	00
bne	000101											

Note that Table 3 has some entries which are “don’t care” values which have arbitrarily assigned with values.

Procedure

Complete Table 3 by deriving all the remaining values of the control signals. Design the control unit using VHDL. The control unit in this implementation is a combinational logic circuit whose inputs are the opcode and function fields of the instruction and the outputs are the 10 control signals. Test your control unit (through simulation) to ensure that it generates the correct values for the 10 control signal for each of the 20 instructions. You may either use a VHDL process for the control unit which will be added to your datapath designed in Lab 4, or you may design the control unit as a separate entity and use it as an additional component to be added with a port map statement to your existing datapath. Alternatively, the datapath of Lab 4 may be added as a component together with an instance of your control unit to create a new top level entity (with name cpu).

Use the following VHDL entity specification for the final CPU design:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_signed.all;

entity  cpu  is
port(reset : in std_logic;
     clk   : in std_logic;
     rs_out, rt_out : out std_logic_vector(3 downto 0);

    -- output ports from register file

     pc_out : out std_logic_vector(3 downto 0); -- pc reg
     overflow, zero : out std_logic);


end cpu;
```

You will note that in the cpu entity, the top-level ports consist of the out_a (rs) and out_b (rt) ports of the register file and the PC register. As was done in previous labs, only the low-order 4 bits of these registers shall be constrained to LEDs on the Nexys A7 FPGA board. The overflow and zero output ports will be constrained to available LEDs. There is an asynchronous reset (which will be constrained to a switch input) and a clock input (constrained to a switch on the FPGA board).

It is important that you use the above entity specification, as it is the one which will be used during the lab test.

Test your complete CPU by writing different test programs into the I-cache and **verify correct operation of your CPU through simulation. Explain your test programs.**

Requirements

1. Modelsim simulation results for both the control unit and the complete CPU showing execution of one example of each class of instruction: arithmetic and logic with and without immediate operands, conditional branches, unconditional jumps, and memory access instructions.

2. The VHDL code for the complete CPU (icluding the control unit VHDL code).

3. Synthesis and implementation log files (runme.log) as generated by Xilinx Vivado.

4. You are not required to demonstrate your downloaded design to the lab TA as this is the last lab of the session and there is no subsequent lab session two weeks henceforth. Although you are not required to demonstrate the working design, you are required to submit as part of the written lab report the following:

• the runme.log files for both the synthesis and implementatin runs as created by the Xilinx Vivado software.

• a listing of the directory contents containing the CPU .bit file generated by the Xilinx Vivado software for Lab 5. Include in the listing the present working directory (obtained with the Linux 'pwd' command). For example:

```
ted@deadflowers impl_1 2:18pm >pwd
/nfs/home/t/ted/VIVADO/COEN316_Nexsys_Board/Lab5/cpu/cpu.runs/
impl_1
ted@deadflowers impl_1 2:18pm >ls -al cpu.bit
-rw------- 1 ted ted 3825888 Nov 10 14:47 cpu.bit
ted@deadflowers impl_1 2:18pm >
```

Report Submission

The due date for Lab 5 shall be announced by Moodle and email. Submit a **hardcopy** printout of your lab report to the mailbox of T. Obuchowicz in room EV5.139. Please ask the receptionist at the front desk in EV5.139 to place it in my mailbox . CLEARLY INDICATE ON THE FRONT COVER OF YOUR LAB REPORT YOUR LAB SECTION. **It is in the best interest of each student to complete Lab 5 prior to performing the lab quiz (week of Nov. 29- Dec. 3, 2021) as the lab quiz will involve a demo of the CPU for a provided machine code program.**

References

1. *Computer Architecture, From Microprocessors to Supercomputers*, Behrooz Parhami, Oxford University Press, ISBN 0-19-515455-x, 2006, p251.

Ted Obuchowicz
November 14, 2016.

Revised: Nov. 7, 2017.

Revised: Nov. 11, 2021: Updated for Xilinx Vivado and Nexys A7 FPGA board. T. Obuchowicz.