浙江大学

本科实验报告

课程名称：	计算机组成
姓　名：	TANG ANNA YONGQI
学　院：	计算机科学与技术学院
专　业：	计算机科学与技术（中加班）留学生
学　号：	3180300155
生活照：	
指导教师：	刘海风，洪奇军

2020 年　4 月 10 日

Lab 5 – Datapath Design

Name: Anna Yongqi Tang **ID:** 3180300155 **Major:** 计算机科学与技术（中加班）留学生
Course: Computer Organization
Date: 2020-04-10 **Instructor:** 洪奇军

1. Method and Experimental Steps

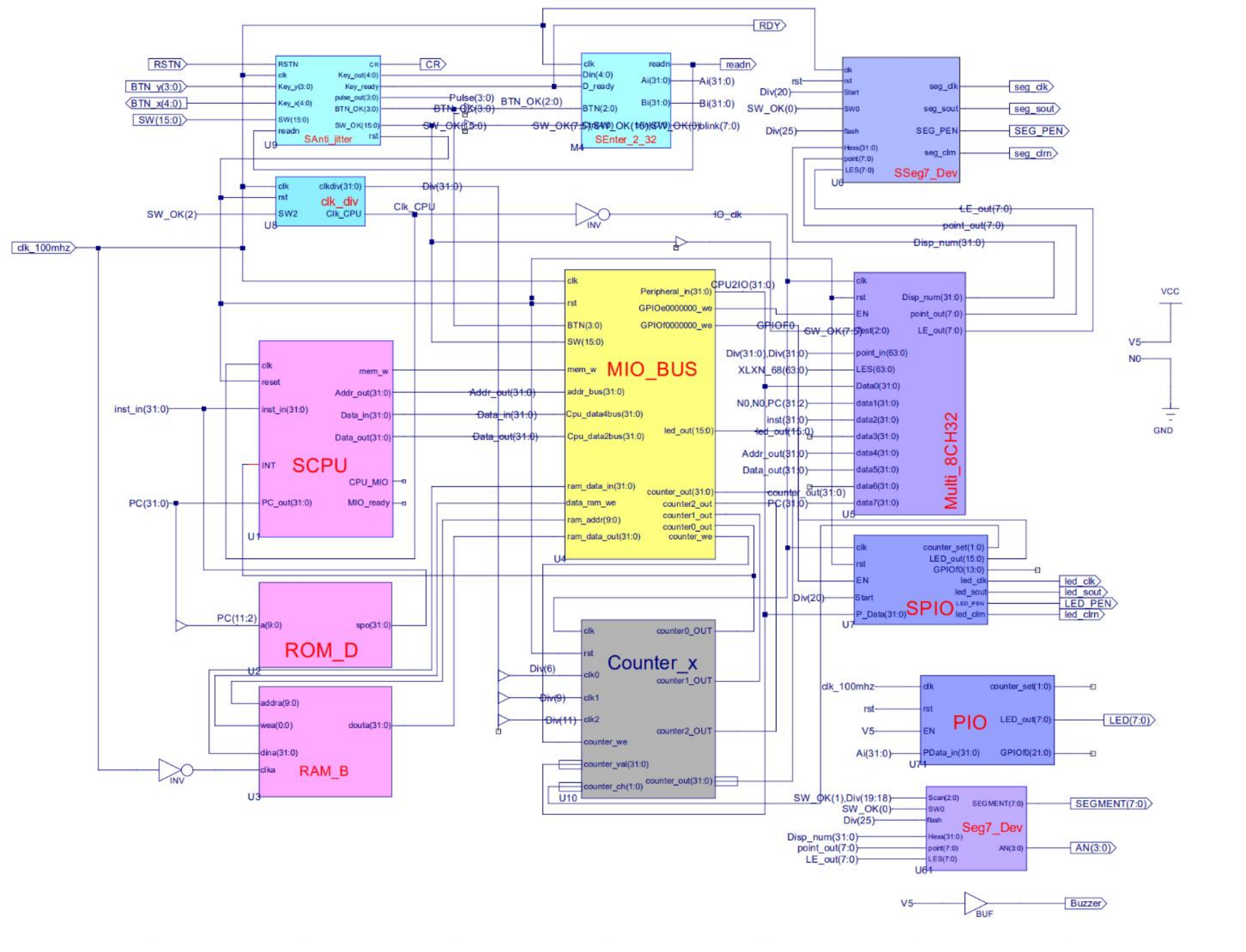


Figure 1 - topMod.sch

This depicts the completion of lab 5. The purpose of this week's experiment was to explore the different components of a datapath, and better understand how a processor works in a single cycle implementation. In this lab, I took lab 4 and I modified the datapath unit of the SCPU. In the datapath that we have constructed this week, it consists of a PC, adders to calculate target addresses, MUXes to handle different signals, a sign extender, a register file and an ALU which were from the last lab. There are five MUXes and seven different control signals to handle instruction processing. Instruction memory and data memory were not implemented in this week's lab. The .ucf for this program came from the provided courseware, and is linked to topMod.sch. Synthesis had minimal warnings, and implementation was successful. A programmable file has been generated and is ready for testing on the SWORD board.

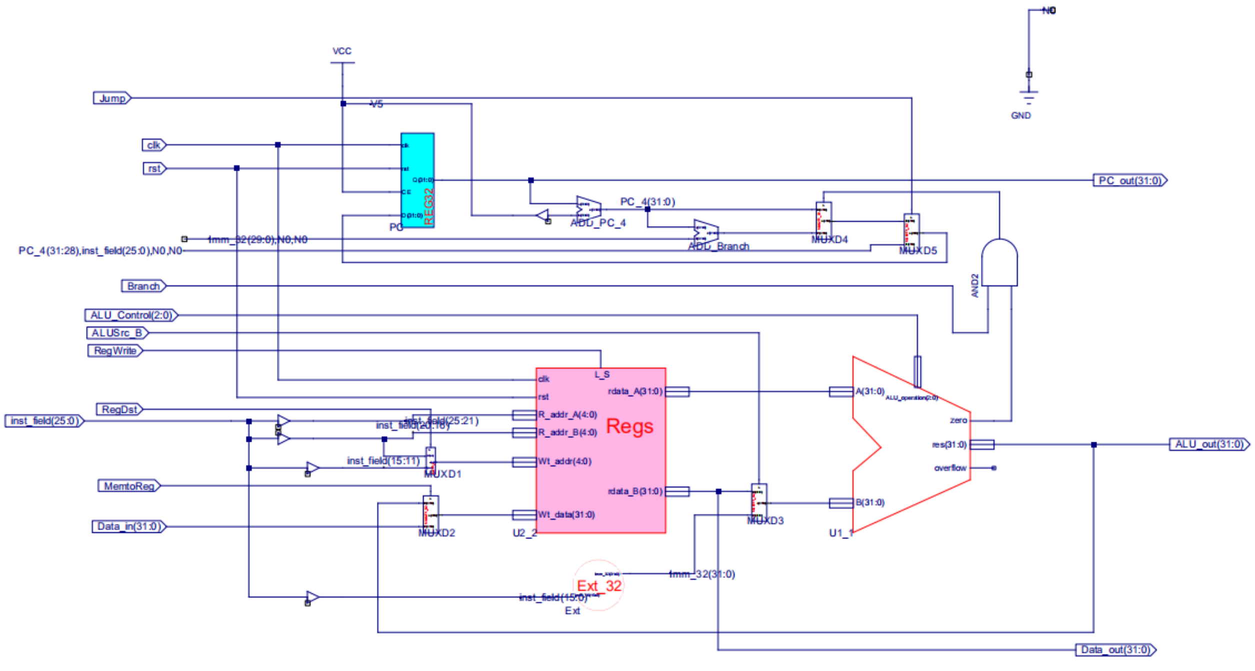
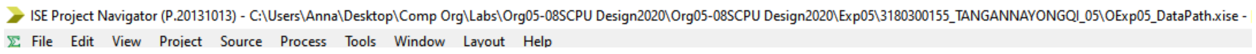


Figure 2 – Data_path.sch

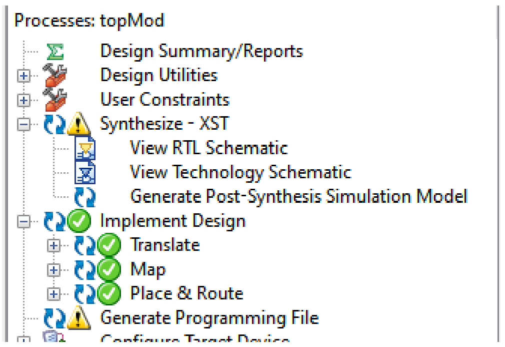
The above picture is a schematic of the datapath module of the SCPU. As we can see, there are several control signals and MUXes used. In this implement, the memory read and write signals are omitted. The datapath operates differently depending on the type of instruction fetched. The control signals have different assertions depending on instruction class. These signals determine which components are used, and how data flows in the path.



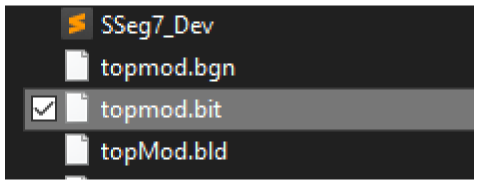
ISE Project Navigator (P.20131013) - C:\Users\Anna\Desktop\Comp Org\Labs\Org05-08SCPU Design2020\Org05-08SCPU Design2020\Exp05\3180300155_TANGANNAYONGQI_05\OExp05_DataPath.xise -

File Edit View Project Source Process Tools Window Layout Help

Figure 3 – 3180300155_TANGANNAYONGQI_05

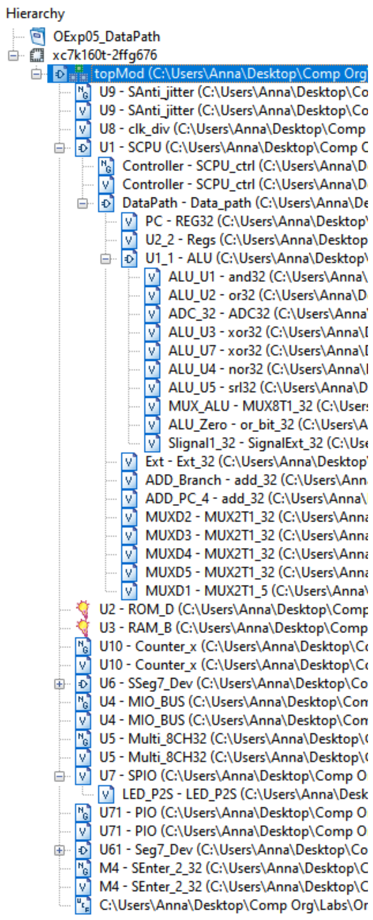


Processes: topMod
- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
  - View RTL Schematic
  - View Technology Schematic
  - Generate Post-Synthesis Simulation Model
- Implement Design
  - Translate
  - Map
  - Place & Route
- Generate Programming File



SSeg7_Dev
topmod.bgn
✓ topmod.bit
topMod.bld

Figure 4 - .bit file generation *Figure 5 - .bit file generated in directory*

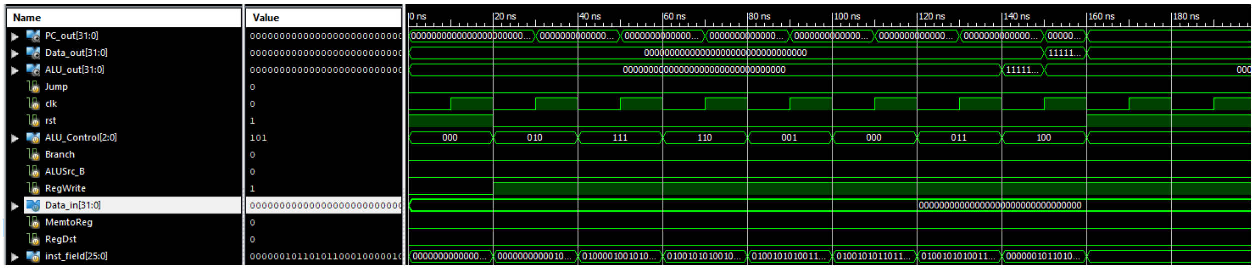


```
Hierarchy
  OExp05_DataPath
  xc7k160t-2ffg676
    topMod (C:\Users\Anna\Desktop\Comp Org
      U9 - SAnti_jitter (C:\Users\Anna\Desktop\Co
      U9 - SAnti_jitter (C:\Users\Anna\Desktop\Co
      U8 - clk_div (C:\Users\Anna\Desktop\Comp
      U1 - SCPU (C:\Users\Anna\Desktop\Comp C
        Controller - SCPU_ctrl (C:\Users\Anna\D
        Controller - SCPU_ctrl (C:\Users\Anna\D
        DataPath - Data_path (C:\Users\Anna\De
          PC - REG32 (C:\Users\Anna\Desktop\
          U2_2 - Regs (C:\Users\Anna\Desktop
          U1_1 - ALU (C:\Users\Anna\Desktop\
            ALU_U1 - and32 (C:\Users\Anna\
            ALU_U2 - or32 (C:\Users\Anna\D
            ADC_32 - ADC32 (C:\Users\Anna\
            ALU_U3 - xor32 (C:\Users\Anna\I
            ALU_U7 - xor32 (C:\Users\Anna\I
            ALU_U4 - nor32 (C:\Users\Anna\I
            ALU_U5 - srl32 (C:\Users\Anna\D
            MUX_ALU - MUX8T1_32 (C:\User:
            ALU_Zero - or_bit_32 (C:\Users\A
            SIignal1_32 - SignalExt_32 (C:\Use
          Ext - Ext_32 (C:\Users\Anna\Desktop\
          ADD_Branch - add_32 (C:\Users\Ann
          ADD_PC_4 - add_32 (C:\Users\Anna\
          MUXD2 - MUX2T1_32 (C:\Users\Anna
          MUXD3 - MUX2T1_32 (C:\Users\Anna
          MUXD4 - MUX2T1_32 (C:\Users\Anna
          MUXD5 - MUX2T1_32 (C:\Users\Anna
          MUXD1 - MUX2T1_5 (C:\Users\Anna\
      U2 - ROM_D (C:\Users\Anna\Desktop\Comp
      U3 - RAM_B (C:\Users\Anna\Desktop\Comp
      U10 - Counter_x (C:\Users\Anna\Desktop\Co
      U10 - Counter_x (C:\Users\Anna\Desktop\Co
      U6 - SSeg7_Dev (C:\Users\Anna\Desktop\Co
      U4 - MIO_BUS (C:\Users\Anna\Desktop\Con
      U4 - MIO_BUS (C:\Users\Anna\Desktop\Con
      U5 - Multi_8CH32 (C:\Users\Anna\Desktop\
      U5 - Multi_8CH32 (C:\Users\Anna\Desktop\
      U7 - SPIO (C:\Users\Anna\Desktop\Comp O
        LED_P2S - LED_P2S (C:\Users\Anna\Desk
      U71 - PIO (C:\Users\Anna\Desktop\Comp O
      U71 - PIO (C:\Users\Anna\Desktop\Comp O
      U61 - Seg7_Dev (C:\Users\Anna\Desktop\Co
      M4 - SEnter_2_32 (C:\Users\Anna\Desktop\C
      M4 - SEnter_2_32 (C:\Users\Anna\Desktop\C
      C:\Users\Anna\Desktop\Comp Org\Labs\Or
```

Figure 6 - file hierarchy

2. Simulations and Observations

This lab requires a MIPS program to be designed and tested on the datapath. The DEMO program and datapath testing will be performed later. The error observed in the ALU from last lab has been fixed, and I have re-simulated it using the Verilog test fixture that I have previously produced. I have also simulated the datapath by adjusting the control signals for each instruction that it currently supports.



Name	Value
PC_out[31:0]	0000000000000000000000000
Data_out[31:0]	0000000000000000000000000
ALU_out[31:0]	0000000000000000000000000
Jump	0
clk	0
rst	1
ALU_Control[2:0]	101
Branch	0
ALUSrc_B	0
RegWrite	1
Data_in[31:0]	0000000000000000000000000
MemtoReg	0
RegDst	0
inst_field[25:0]	0000001011010110001000001

Figure 7 – Datapath Simulation

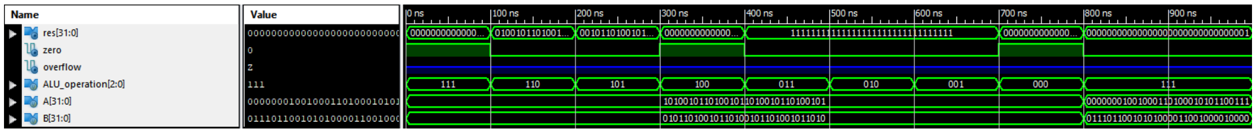


Figure 8 – Revised ALU Simulation

Datapath Simulation

```
module Data_path_Data_path_sch_tb();

// Inputs
   reg Jump;
   reg clk;
   reg rst;
   reg [2:0] ALU_Control;
   reg Branch;
   reg ALUSrc_B;
   reg RegWrite;
   reg [31:0] Data_in;
   reg MemtoReg;
   reg RegDst;
   reg [25:0] inst_field;

// Output
   wire [31:0] PC_out;
   wire [31:0] Data_out;
   wire [31:0] ALU_out;

// Bidirs

// Instantiate the UUT
   Data_path UUT (
            .PC_out(PC_out),
            .Jump(Jump),
            .clk(clk),
            .rst(rst),
            .ALU_Control(ALU_Control),
            .Branch(Branch),
            .Data_out(Data_out),
            .ALUSrc_B(ALUSrc_B),
            .RegWrite(RegWrite),
            .ALU_out(ALU_out),
            .Data_in(Data_in),
            .MemtoReg(MemtoReg),
            .RegDst(RegDst),
            .inst_field(inst_field)
   );
// Initialize Inputs
      initial begin
            `define signals
{RegDst,ALUSrc_B,MemtoReg,Jump,Branch,RegWrite,ALU_Control}
                  Branch = 0;
                  clk = 0;
                  rst = 1;
                  ALU_Control = 0;
                  ALUSrc_B = 0;
                  RegWrite = 0;
                  Jump = 0;
                  MemtoReg = 0;
                  inst_field = 0;
                  Data_in = 0;
                  RegDst = 0;
                  #20;

                  rst=0;
                  //add
                  `signals=12'b100000001010;
                  inst_field=26'b00000000000100000000100000;
                  #20;
                  //slt
                  `signals=12'b100000001111;
                  inst_field=26'b01000010010101000000101010;
                  #20;
                  //sub
                  `signals=12'b100000001110;
                  inst_field=26'b01001010100101100000100010;
                  #20;
                  //or
                  `signals=12'b100000001001;
                  inst_field=26'b01001010100110000000100101;
                  #20;
                  //and
                  `signals=12'b100000001000;
                  inst_field=26'b01001010110110100000100100;
                  #20;
                  //xor
                  `signals=12'b100000001011;
                  inst_field=26'b01001010100111000000100110;
                  #20;
                  //nor
                  `signals=12'b100000001100;
                  inst_field=26'b00000010110100100000100111;
                  #20;
                  //srl
                  `signals=12'b111000001101;
                  inst_field=26'b00000010110101100010000010;
                  rst=1;
      end
      always begin
            clk=0;#10;
            clk=1;#10;
      end
endmodule
```

3. Conclusion

In this week's lab and theoretical section of this course, we got to explore and build a datapath from scratch and look at how different instruction types process data from the register file in a single cycle. I also got to look at how read and write in memory and the register file worked. We haven't gotten the chance to implement memory into this datapath yet, but I'm looking forward to that. I'm also looking forward to implementing this onto actual hardware, and physically witnessing how computer processes work. Last summer, I took Yale Patt's Computing System's course and I'm glad that I got to revisit this concept again. It was cool to see the differences in the course of action of the LC-3 and the kintex7. I'm looking forward to experimenting with the further continuations of this lab.

4. Source Code

All modules and components were either retrieved from previous labs or directly taken from the given files.