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　5 月 17 日

Lab 6 – Multicycle CPU 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: 2020-05-17 **Instructor:** 洪奇军

1. Method and Experimental Steps

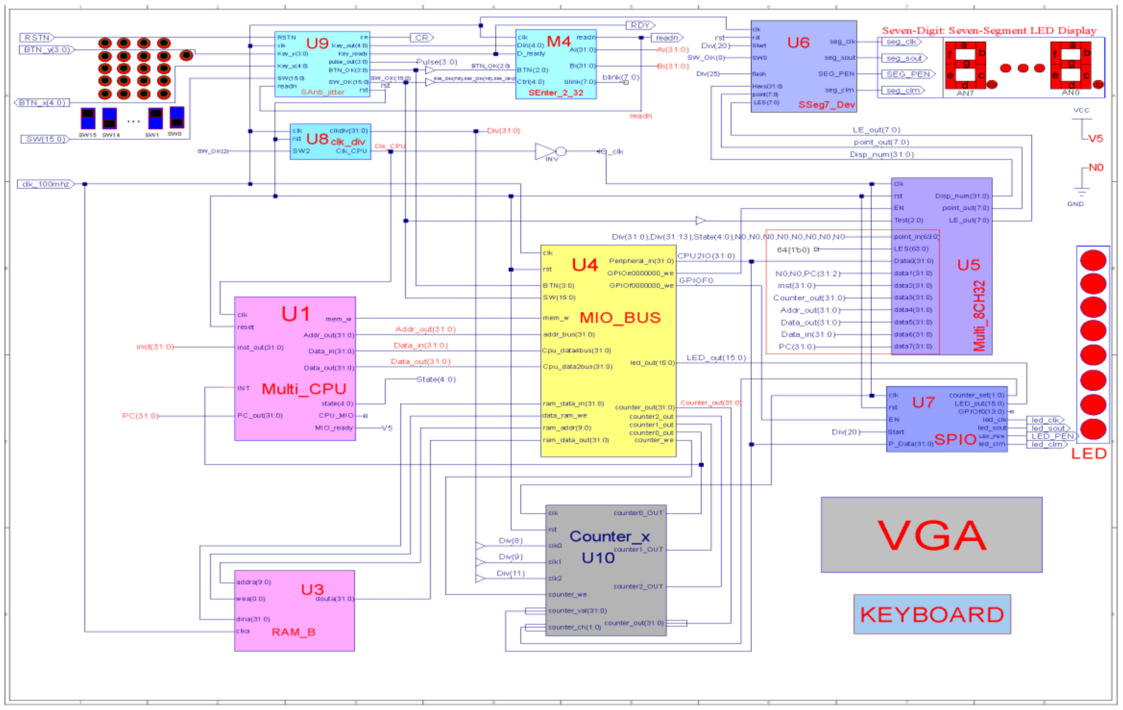


Figure 1 - topMod.sch

In lab 9, I designed a multicycle CPU using modules from previous experiments and ngc cores. The top level (pictured above) was designed off of lab 3, but a HDL implementation was used instead of a schematic. The CPU module is replaced with a multicycle CPU, and only a RAM was used. Supposedly, we were to implement a new coe file in the ram. This lab's purpose was to better understand the structure of a multicycle implementation and explore the different module's functions. I believe that within the next coming experiments, we will rebuild some of the modules from scratch instead of using ngc cores.
The .ucf for this program came from the provided courseware, and is linked to topMod.v. Synthesis had minimal warnings, and implementation was successful. A programmable file has been generated and is ready for testing on the SWORD board. The above diagram was retrieved from the lab PowerPoint. No schematic has been drawn for this experiment.

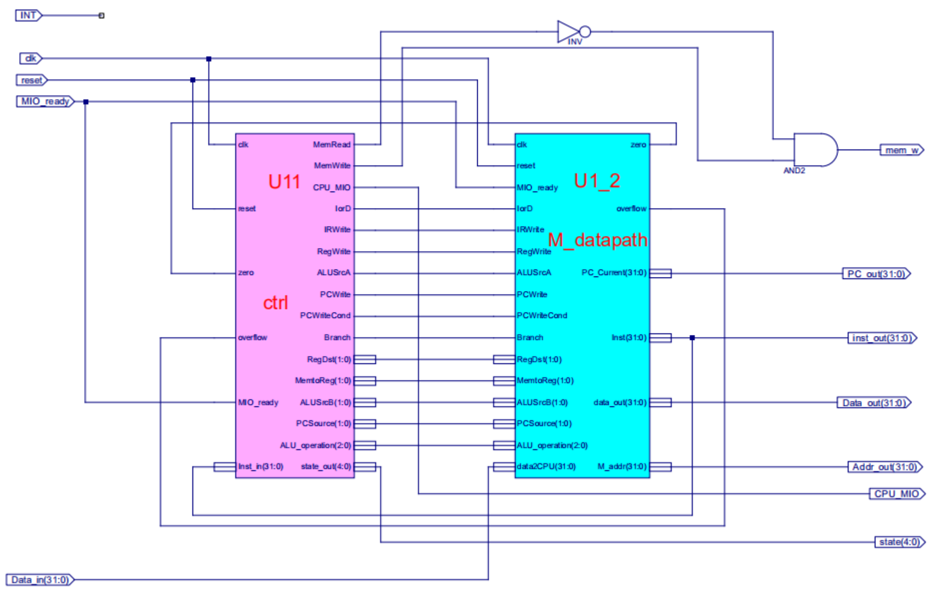
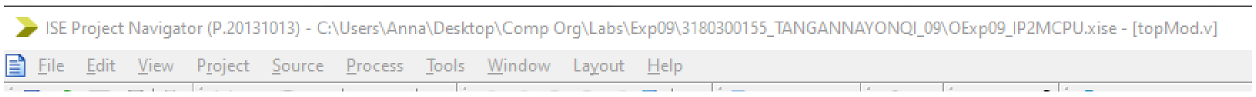


Figure 2 – Multi_CPU.sch

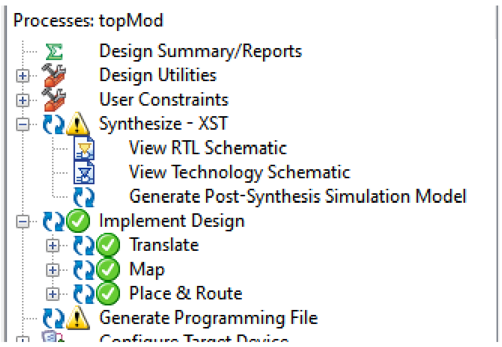
The multicycle CPU is made up of two modules, the datapath and the controller. All the code and schematics here were retrieved from the provided courseware. We use a RISC implementation, where memory is accessed through specific instructions rather than as a part of most instructions. The datapath processes the instruction and performs memory accesses based on the what is provided by the PC. It also takes care of the different control signals that were explored in chapter 5. The control module tells the datapath what to do to by asserting and deasserting the control signals. The above diagram was also retrieved from the provided courseware.



ISE Project Navigator (P.20131013) - C:\Users\Anna\Desktop\Comp Org\Labs\Exp09\3180300155_TANGANNAYONQI_09\OExp09_IP2MCPU.xise - [topMod.v]

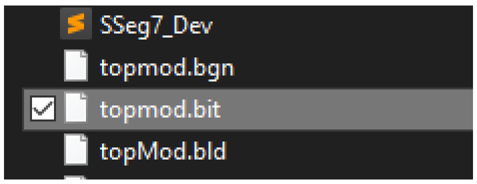
File Edit View Project Source Process Tools Window Layout Help

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

Figure 4 - .bit file generation



SSeg7_Dev
topmod.bgn
topmod.bit
topMod.bld

Figure 5 - .bit file generated in directory

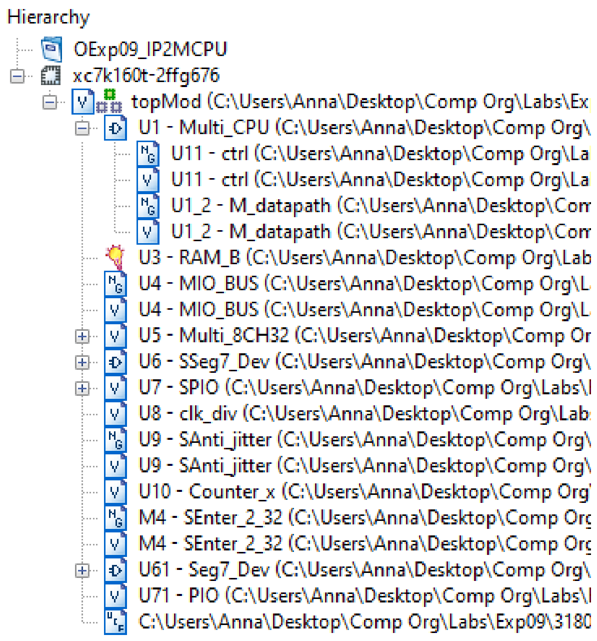

Hierarchy
OExp09_IP2MCPU
xc7k160t-2ffg676
topMod (C:\Users\Anna\Desktop\Comp Org\Labs\Ex
U1 - Multi_CPU (C:\Users\Anna\Desktop\Comp Org\
U11 - ctrl (C:\Users\Anna\Desktop\Comp Org\La
U11 - ctrl (C:\Users\Anna\Desktop\Comp Org\La
U1_2 - M_datapath (C:\Users\Anna\Desktop\Com
U1_2 - M_datapath (C:\Users\Anna\Desktop\Com
U3 - RAM_B (C:\Users\Anna\Desktop\Comp Org\Lab
U4 - MIO_BUS (C:\Users\Anna\Desktop\Comp Org\L
U4 - MIO_BUS (C:\Users\Anna\Desktop\Comp Org\L
U5 - Multi_8CH32 (C:\Users\Anna\Desktop\Comp Or
U6 - SSeg7_Dev (C:\Users\Anna\Desktop\Comp Org\
U7 - SPIO (C:\Users\Anna\Desktop\Comp Org\Labs\
U8 - clk_div (C:\Users\Anna\Desktop\Comp Org\Lab
U9 - SAnti_jitter (C:\Users\Anna\Desktop\Comp Org\
U9 - SAnti_jitter (C:\Users\Anna\Desktop\Comp Org\
U10 - Counter_x (C:\Users\Anna\Desktop\Comp Org
M4 - SEnter_2_32 (C:\Users\Anna\Desktop\Comp Org
M4 - SEnter_2_32 (C:\Users\Anna\Desktop\Comp Org
U61 - Seg7_Dev (C:\Users\Anna\Desktop\Comp Org\
U71 - PIO (C:\Users\Anna\Desktop\Comp Org\Labs\
C:\Users\Anna\Desktop\Comp Org\Labs\Exp09\3180

Figure 6 - file hierarchy

2. Simulations and Observations

This lab requires a MIPS program (demo.coe) to be tested on the multicycle CPU. The demo will be performed later in a future exercise. Since all modules were implemented with ngc cores, no simulation was performed for this lab.

3. Conclusion

This week's lab was just a brief overview and introduction to a multicycle CPU implementation. It is quite similar in structure to the single cycle implementation, but a lot more efficient in runtime and hardware. I look forward to simulating the datapath and comparing the results to what was produced in a single cycle implementation. I am also looking forward to rebuilding some of the modules used from scratch and examining the differences in hardware.

4. Source Code

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

Top Module - topMod.v

```
module topMod(
    input RSTN,
    input [3:0] BTN_y,
    input [4:0] BTN_x,
    input [15:0] SW,
    input clk_100mhz,
    output CR,
    output RDY,
    output readn,
    output seg_clk,
    output seg_sout,
    output seg_clrn,
    output SEG_PEN,
    output led_clk,
    output led_sout,
    output LED_PEN,
    output led_clrn,
    output [7:0] SEGMENT,
    output [3:0] AN,
    output [7:0] LED,
    output Buzzer
  );

    wire V5, N0;

    assign V5 = 1'b1;
    assign N0 = 1'b0;
    assign Buzzer = 1'b1;

    wire Clk_CPU, mem_w, data_ram_we, IO_clk, GPIOE0, GPIOF0, counter0_out,
 counter1_out, counter2_out, counter_we;
    wire[1:0] counter_set;
    wire[3:0] BTN_OK, Pulse;
    wire[4:0] Key_out, state;
    wire[7:0] point_out, LE_out, blink;
    wire[9:0] ram_addr;
    wire[15:0] SW_OK, LED_out;
    wire[31:0] inst, PC, Addr_out, Data_in, Data_out, ram_data_in, ram_data_out,
 CPU2IO, Counter_out, Div, Disp_num, Ai, Bi;

    assign IO_clk = ~Clk_CPU;

    Multi_CPU U1(
         .clk(Clk_CPU),
         .reset(rst),
         .inst_out(inst),
         .INT(counter0_out),
         .PC_out(PC),
         .mem_w(mem_w),
         .Addr_out(Addr_out),
         .Data_in(Data_in),
         .Data_out(Data_out),
         .state(state),
         .CPU_MIO(),
         .MIO_ready(V5)
     );

     RAM_B U3(
         .addra(ram_addr),
         .wea(data_ram_we),
         .dina(ram_data_in),
         .clka(clk_100mhz),
         .douta(ram_data_out)
     );
```

```
MIO_BUS U4(
    .clk(clk_100mhz),
    .rst(rst),
    .BTN(BTN_OK),
    .SW(SW_OK),
    .mem_w(mem_w),
    .Cpu_data2bus(Data_out),
    .addr_bus(Addr_out),
    .ram_data_out(ram_data_out),
    .led_out(LED_out),
    .counter_out(Counter_out),
    .counter0_out(counter0_out),
    .counter1_out(counter1_out),
    .counter2_out(counter2_out),
    .Cpu_data4bus(Data_in),
    .ram_data_in(ram_data_in),
    .ram_addr(ram_addr),
    .data_ram_we(data_ram_we),
    .GPIOf0000000_we(GPIOF0),
    .GPIOe0000000_we(GPIOE0),
    .counter_we(counter_we),
    .Peripheral_in(CPU2IO)
);

Multi_8CH32 U5(
    .clk(IO_clk),
    .rst(rst),
    .EN(GPIOE0),
    .Test(SW_OK[7:5]),
    .point_in({Div, Div[31:13], state, N0, N0, N0, N0, N0, N0, N0, N0}),
    .LES(64'b0),
    .Data0(CPU2IO),
    .data1({N0,N0,PC[31:2]}),
    .data2(inst),
    .data3(Counter_out),
    .data4(Addr_out),
    .data5(Data_out),
    .data6(Data_in),
    .data7(PC),
    .point_out(point_out),
    .LE_out(LE_out),
    .Disp_num(Disp_num)
);

SSeg7_Dev U6(
    .clk(clk_100mhz),
    .rst(rst),
    .Start(Div[20]),
    .SW0(SW_OK[0]),
    .flash(Div[25]),
    .Hexs(Disp_num),
    .point(point_out),
    .LES(LE_out),
    .seg_clk(seg_clk),
    .seg_sout(seg_sout),
    .SEG_PEN(SEG_PEN),
    .seg_clrn(seg_clrn)
);

SPIO U7(
    .clk(IO_clk),
    .rst(rst),
    .Start(Div[20]),
    .EN(GPIOF0),
    .GPIOf0(),
    .P_Data(CPU2IO),
    .counter_set(counter_set),
    .LED_out(LED_out),
    .led_clk(led_clk),
    .led_sout(led_sout),
    .led_clrn(led_clrn),
    .LED_PEN(LED_PEN)
);

clk_div U8(
    .clk(clk_100mhz),
    .rst(rst),
    .SW2(SW_OK[2]),
    .clkdiv(Div),
    .Clk_CPU(Clk_CPU)
);

SAnti_jitter U9(
    .clk(clk_100mhz),
    .RSTN(RSTN),
    .readn(readn),
    .Key_y(BTN_y),
    .Key_x(BTN_x),
    .SW(SW),
    .Key_out(Key_out),
    .Key_ready(RDY),
    .pulse_out(Pulse),
    .BTN_OK(BTN_OK),
    .SW_OK(SW_OK),
    .CR(CR),
    .rst(rst)
);

Counter_x U10(
    .clk(IO_clk),
    .rst(rst),
    .clk0(Div[8]),
    .clk1(Div[9]),
    .clk2(Div[10]),
    .counter_we(counter_we),
    .counter_val(CPU2IO),
    .counter_ch(counter_set),
    .counter0_OUT(counter0_out),
    .counter1_OUT(counter1_out),
    .counter2_OUT(counter2_out),
    .counter_out(Counter_out)
);

SEnter_2_32 M4(
    .clk(clk_100mhz),
    .BTN(BTN_OK[2:0]),
    .Ctrl({SW_OK[7:5],SW_OK[15],SW_OK[0]}),
    .D_ready(RDY),
    .Din(Key_out),
    .readn(readn),
    .Ai(Ai),
    .Bi(Bi),
    .blink(blink)
);

Seg7_Dev U61(
    .Scan({SW_OK[1],Div[19:18]}),
    .SW0(SW_OK[0]),
    .flash(Div[25]),
    .Hexs(Disp_num),
    .point(point_out),
    .LES(LE_out),
    .SEGMENT(SEGMENT),
    .AN(AN)
);

PIO U71(
    .clk(IO_clk),
    .rst(rst),
    .EN(GPIOF0),
    .counter_set(),
    .GPIOf0(),
    .PData_in(CPU2IO),
    .LED_out(LED)
);

endmodule
```