浙江大學

本科实验报告

课程名称:	计算机体系结构
姓　　名:	王俊
学　　院:	海洋学院
专　　业:	海洋工程与技术
学　　号:	3170100186
指导教师:	翁恺

2020 年　10　月　10　日

Lab1——单周期 CPU 回顾

课程名称：	计算机体系结构			实验类型：	综合
实验项目名称：	单周期 CPU 回顾				
学生姓名：	王俊	专业：	海洋工程与技术	学号：	3170100186
同组学生姓名：	None			指导老师：	翁恺
实验地点：	曹光彪西-301			实验日期：	2020 年 10 月 10 日

一、实验目的和要求

Experiment Purpose:

- Understand the principles of CPU Controller and master methods of CPU Controller design
- Understand the principles of Datapath and master methods of Datapath design
- Understand the principles of Single-cycle CPU and master methods of Single-cycle CPU design
- master methods of program verification of CPU

Experiment Apparatus:

- Computer (Intel Core i5 or higher, 4GB RAM or higher) system
- Sword-V4 development board
- Xilinx ISE 14.4 and above development tools

Experimental Materials:

No

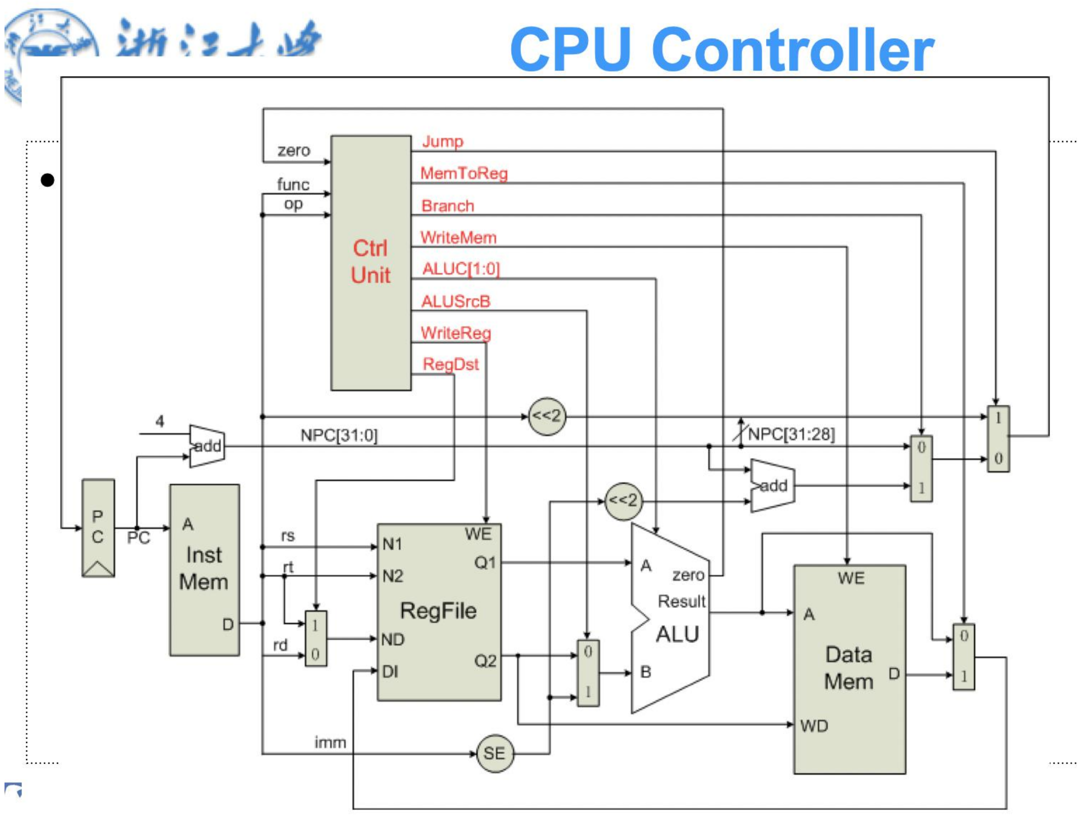
二、实验内容和原理

2.1.Experimental task:

- Design the CPU Controller of Single-cycle CPU
- Design the Datapath, bring together the basic units into Single-cycle CPU
- Verify the CPU with program and observe the execution of program

2.2.Computer Architecture

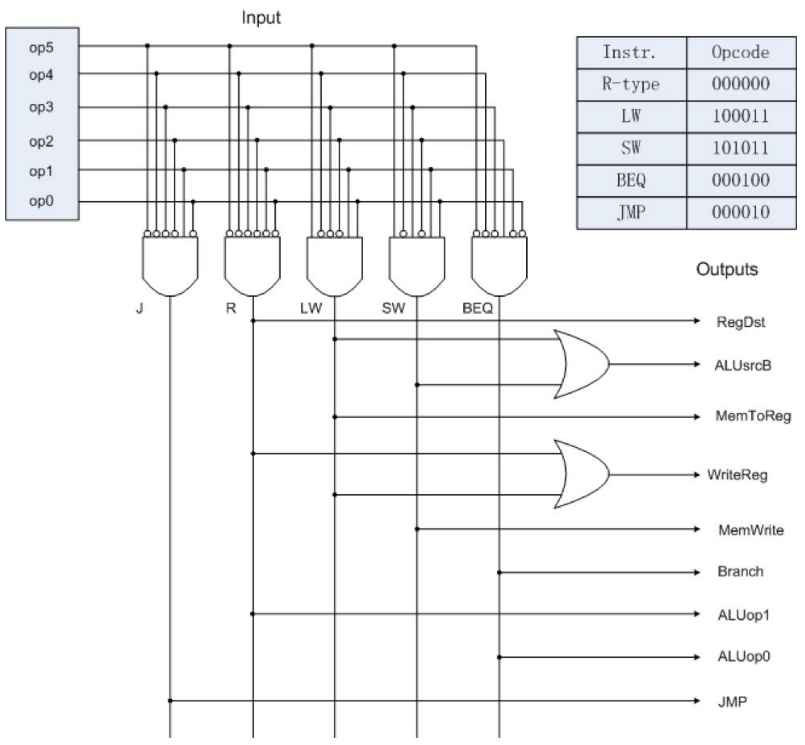
2.2.1.CPU controller



- **Output of CPU Controller:**

NO	Signal	Meaning when signal is 1	Meaning when signal is 0
1	**RegDst**	**rd**	**rt**
2	**ALUsrcB**	**Extended-Imm.**	**register**
3	**MemtoReg**	**Mem. output**	**ALU output**
4	**RegWrite**	**Write to Register file, the addr is up to RegDst, the data is up to MemtoReg**	**Not write to Register file**
5	**MemWrite**	**Write to Memory**	**Not write to memory**
6	**Branch**	**Branch addr.**	**Not branch**
7	**Jump**	**Jump addr.**	**Branch addr**
8	**ALUC[1:0]**	**ALU operation code**	

● The principle of CPU Controller



Instr.	Opcode
R-type	000000
LW	100011
SW	101011
BEQ	000100
JMP	000010

● Basic Units of Single-cycle CPU

•CPU Controller

•ALU and ALU Controller

•Register file

•Instruction Mem. and Data Mem.

•others: Register, adder, sign-extend Unit, shifter, multiplexor。

- **Memory**

•Instr. Mem.

–Single Port Block Memory

–Read only, Width:32

–Rising Edge Triggered

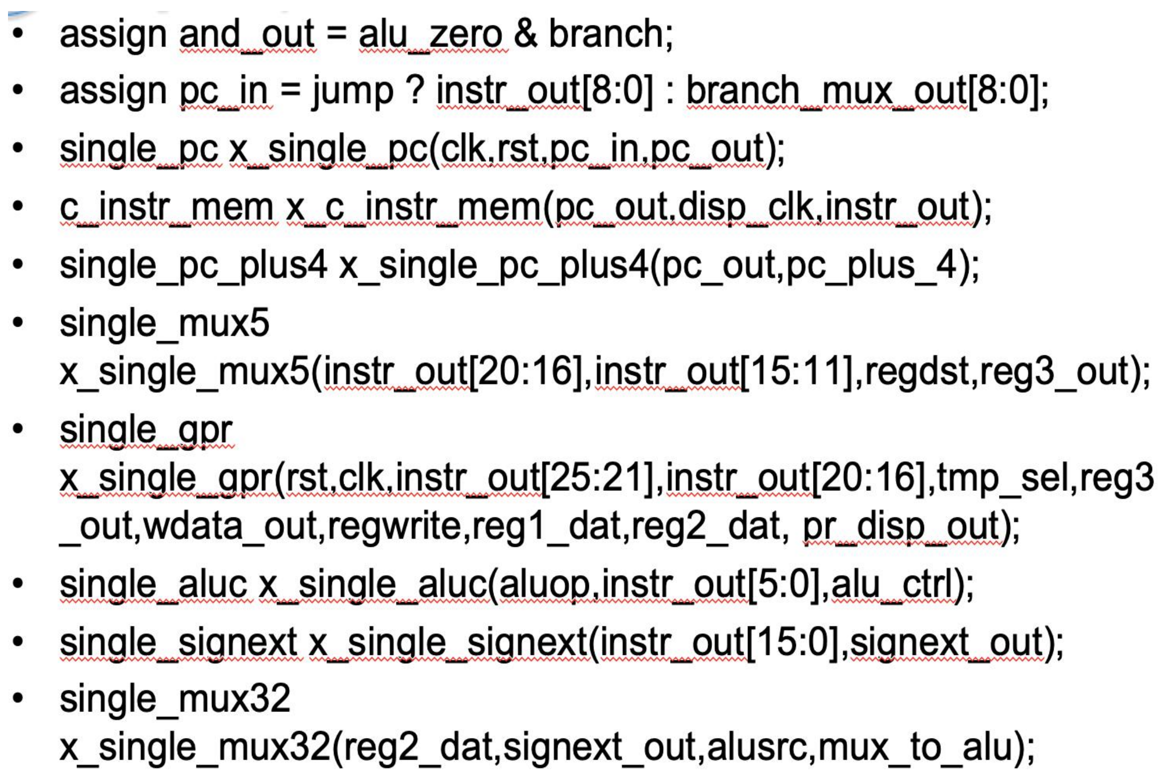
•Data Mem.

–Single Port Block Memory

–Read and Write, Width:32, Read After Write

–Rising Edge Triggered

- **Single-cycle CPU Top Module**



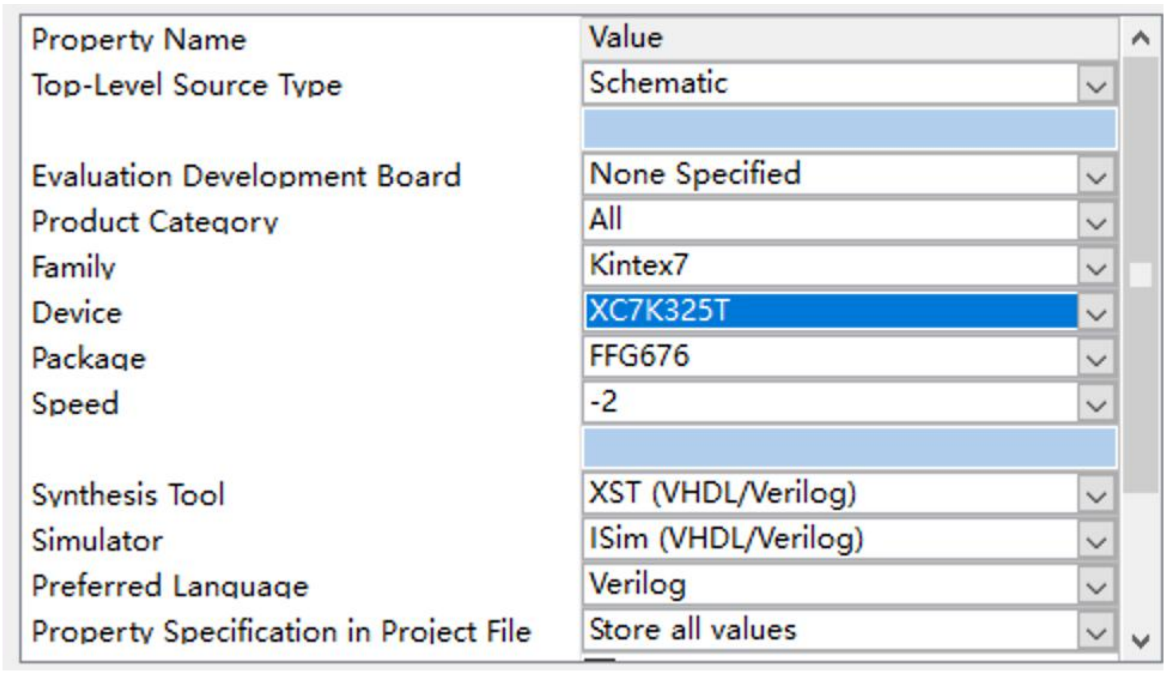
```
• assign and_out = alu_zero & branch;
• assign pc_in = jump ? instr_out[8:0] : branch_mux_out[8:0];
• single_pc x_single_pc(clk,rst,pc_in,pc_out);
• c_instr_mem x_c_instr_mem(pc_out,disp_clk,instr_out);
• single_pc_plus4 x_single_pc_plus4(pc_out,pc_plus_4);
• single_mux5
  x_single_mux5(instr_out[20:16],instr_out[15:11],regdst,reg3_out);
• single_gpr
  x_single_gpr(rst,clk,instr_out[25:21],instr_out[20:16],tmp_sel,reg3
  _out,wdata_out,regwrite,reg1_dat,reg2_dat, pr_disp_out);
• single_aluc x_single_aluc(aluop,instr_out[5:0],alu_ctrl);
• single_signext x_single_signext(instr_out[15:0],signext_out);
• single_mux32
  x_single_mux32(reg2_dat,signext_out,alusrc,mux_to_alu);
```

- single_alu x_single_alu(reg1_dat,mux_to_alu,alu_ctrl,alu_zero,alu_out);
- c_dat_mem x_c_dat_mem(alu_out[8:0],disp_clk,reg2_dat,mem_dat_out,memwrite);
- single_mux32 x_single_mux32_2(alu_out,mem_dat_out,memtoreg,wdata_out);
- single_add x_single_add(signext_out,{{23'b0},pc_plus_4},branch_addr_out);
- single_mux32 x_single_mux32_3({{23'b0},pc_plus_4},branch_addr_out,and_out,branch_mux_out);
- single_ctl x_single_ctl(rst,instr_out[31:26],regdst,jump,branch,memread, memtoreg,aluop,memwrite,alusrc,regwrite);

三、实验过程

3.1.Change the property of project:

FPGA 芯片从 160T 升级为 325T,芯片的引脚(UCF)完全兼容。建立设计工程时要选择“XC7K325T”,其他不变，原来工程只要此处或部分 UCF 就可以使用，如下图 ：

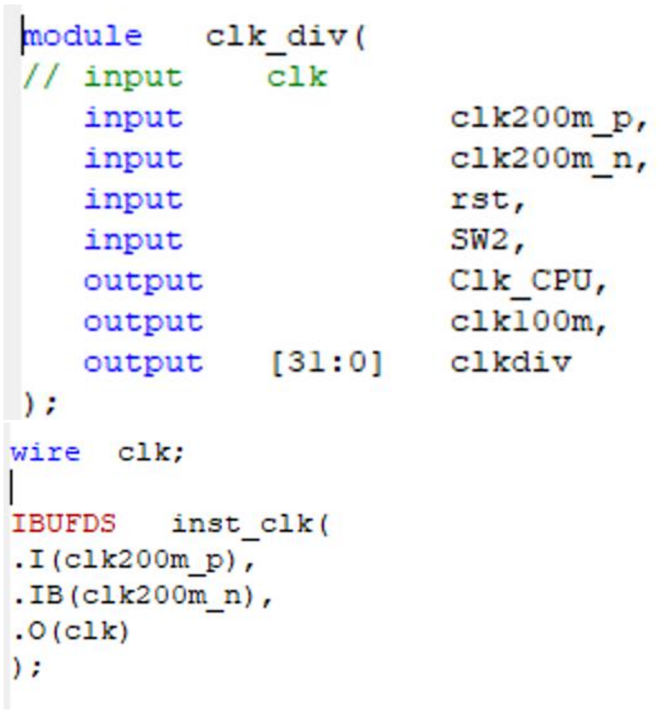


Property Name	Value
Top-Level Source Type	Schematic
Evaluation Development Board	None Specified
Product Category	All
Family	Kintex7
Device	XC7K325T
Package	FFG676
Speed	-2
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values

3.2.Change the clock module

1）时钟改为更可靠的双端口差分时钟模式，主频为 200MHz，具体使用以分频器为例说明如下：（为更清楚看清流水线的执行过程，我们的试验中更需要的是手动（按键）单步时钟，而不是连续时钟，因此这个模块可以不用）

更改后的时钟模块：

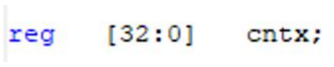


```
module   clk_div(
// input    clk
   input             clk200m_p,
   input             clk200m_n,
   input             rst,
   input             SW2,
   output            Clk_CPU,
   output            clk100m,
   output    [31:0]  clkdiv
);
wire  clk;

IBUFDS   inst_clk(
.I(clk200m_p),
.IB(clk200m_n),
.O(clk)
);
```

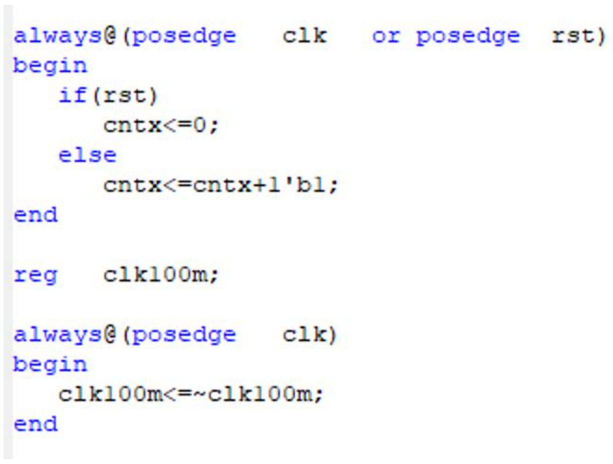
Input: clk200m_p 和 clk200m_n 两个双时钟

Output: CPU 使用的时钟 和 原来使用的 clk100m 和 分频后的时钟 [31:0] clkdiv

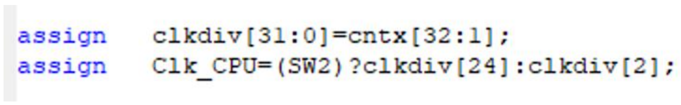
- **code:**



```
reg   [32:0]   cntx;
```

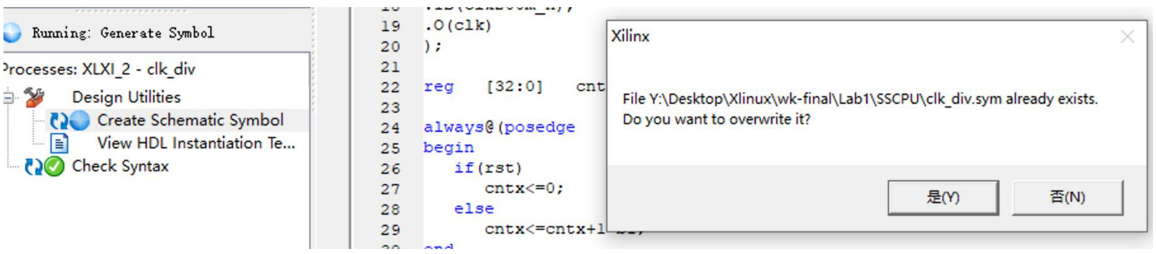


```
always@(posedge   clk   or posedge   rst)
begin
   if(rst)
      cntx<=0;
   else
      cntx<=cntx+1'b1;
end

reg   clk100m;

always@(posedge   clk)
begin
   clk100m<=~clk100m;
end
```



```
assign   clkdiv[31:0]=cntx[32:1];
assign   Clk_CPU=(SW2)?clkdiv[24]:clkdiv[2];
```

- **Check the syntax and Create Schematic Symbol**



Running: Generate Symbol

Processes: XLXI_2 - clk_div

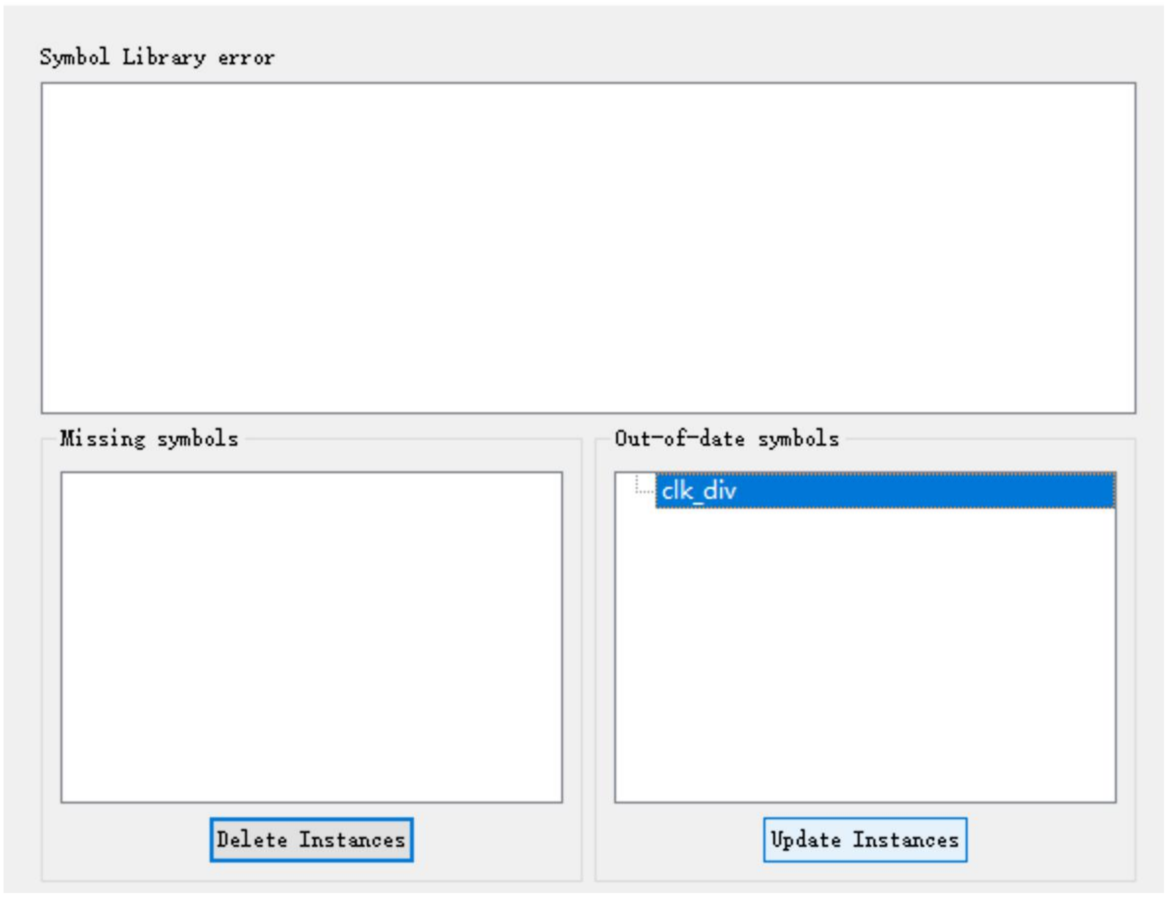
- Design Utilities
  - Create Schematic Symbol
  - View HDL Instantiation Te...
- Check Syntax

```
.O(clk)
);

reg    [32:0]    cnt

always@(posedge
begin
    if(rst)
        cntx<=0;
    else
        cntx<=cntx+1
```

Xilinx

File Y:\Desktop\Xlinux\wk-final\Lab1\SSCPU\clk_div.sym already exists.
Do you want to overwrite it?

- **Update instances:**



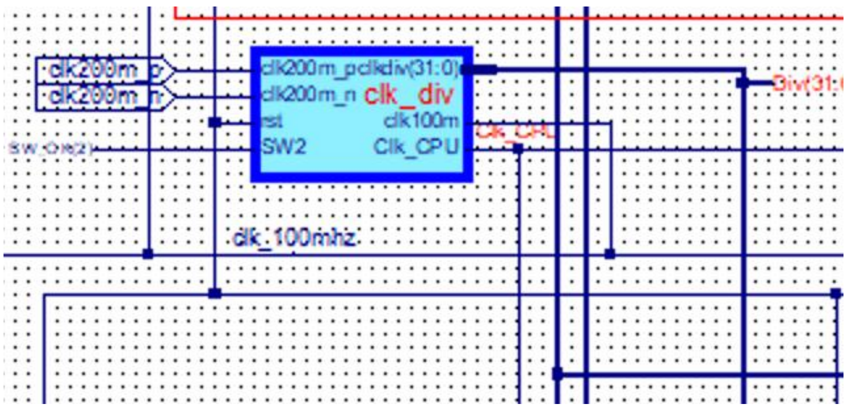
Symbol Library error

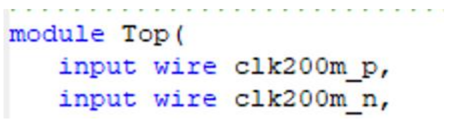
Missing symbols

Out-of-date symbols

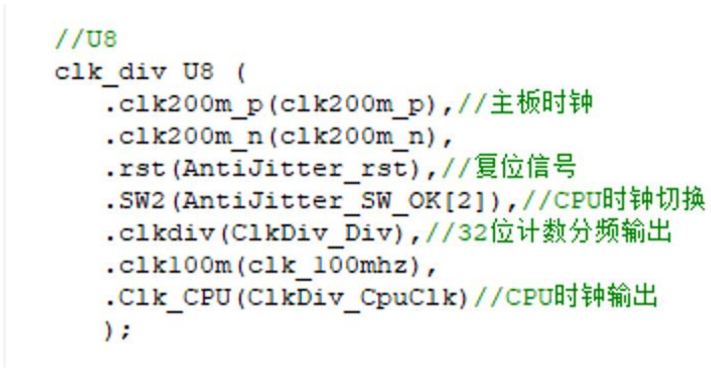
clk_div

3.3.Redesign the data path



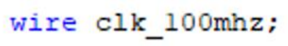


```
module Top(
    input wire clk200m_p,
    input wire clk200m_n,
```



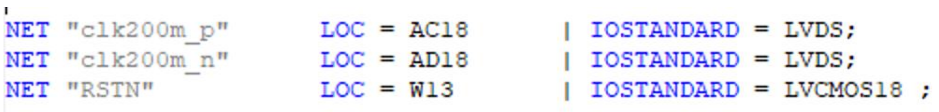
```
   //U8
   clk_div U8  (
      .clk200m_p(clk200m_p),//主板时钟
      .clk200m_n(clk200m_n),
      .rst(AntiJitter_rst),//复位信号
      .SW2(AntiJitter_SW_OK[2]),//CPU时钟切换
      .clkdiv(ClkDiv_Div),//32位计数分频输出
      .clk100m(clk_100mhz),
      .Clk_CPU(ClkDiv_CpuClk)//CPU时钟输出
      );
```

并添加 wire clk_100mhz:

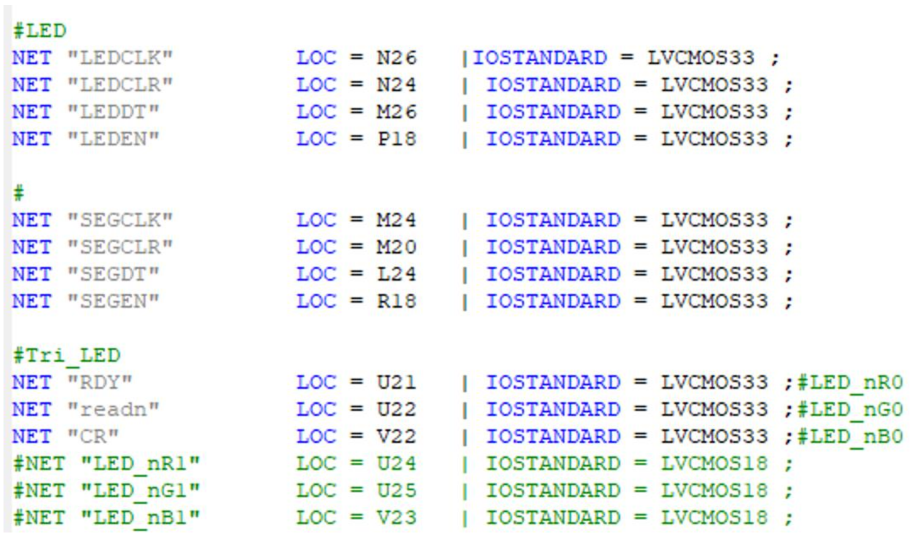


```
 wire clk_100mhz;
```

3.3.change the ucf file

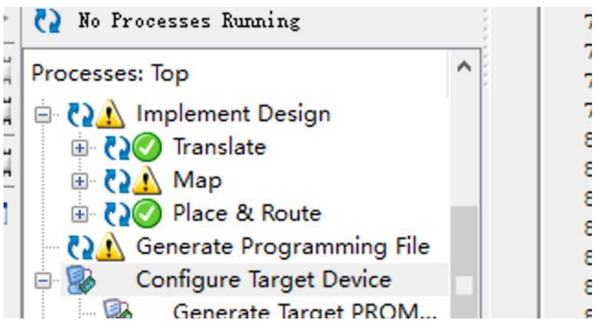


```
NET "clk200m_p"       LOC = AC18       | IOSTANDARD = LVDS;
NET "clk200m_n"       LOC = AD18       | IOSTANDARD = LVDS;
NET "RSTN"            LOC = W13        | IOSTANDARD = LVCMOS18 ;
```



```
#LED
NET "LEDCLK"         LOC = N26    |IOSTANDARD = LVCMOS33 ;
NET "LEDCLR"         LOC = N24    | IOSTANDARD = LVCMOS33 ;
NET "LEDDT"          LOC = M26    | IOSTANDARD = LVCMOS33 ;
NET "LEDEN"          LOC = P18    | IOSTANDARD = LVCMOS33 ;

#
NET "SEGCLK"         LOC = M24    | IOSTANDARD = LVCMOS33 ;
NET "SEGCLR"         LOC = M20    | IOSTANDARD = LVCMOS33 ;
NET "SEGDT"          LOC = L24    | IOSTANDARD = LVCMOS33 ;
NET "SEGEN"          LOC = R18    | IOSTANDARD = LVCMOS33 ;

#Tri_LED
NET "RDY"            LOC = U21    | IOSTANDARD = LVCMOS33 ;#LED_nR0
NET "readn"          LOC = U22    | IOSTANDARD = LVCMOS33 ;#LED_nG0
NET "CR"             LOC = V22    | IOSTANDARD = LVCMOS33 ;#LED_nB0
#NET "LED_nR1"       LOC = U24    | IOSTANDARD = LVCMOS18 ;
#NET "LED_nG1"       LOC = U25    | IOSTANDARD = LVCMOS18 ;
#NET "LED_nB1"       LOC = V23    | IOSTANDARD = LVCMOS18 ;
```

3.4.Regenerate the bit file

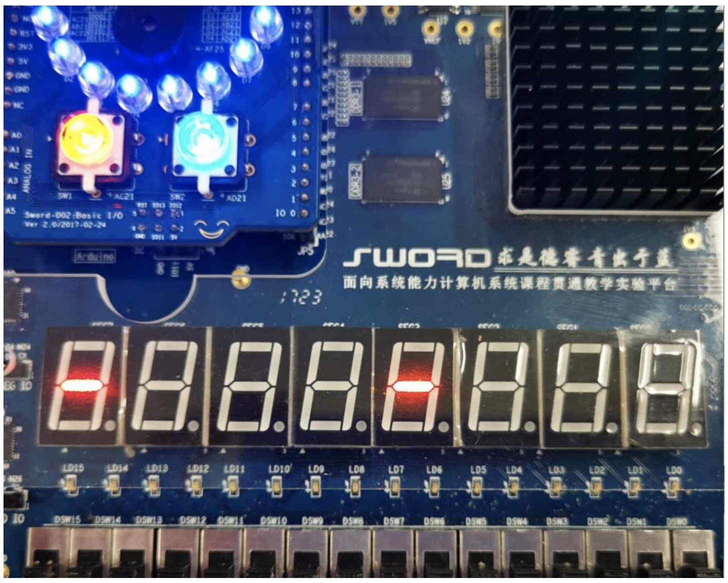


No Processes Running

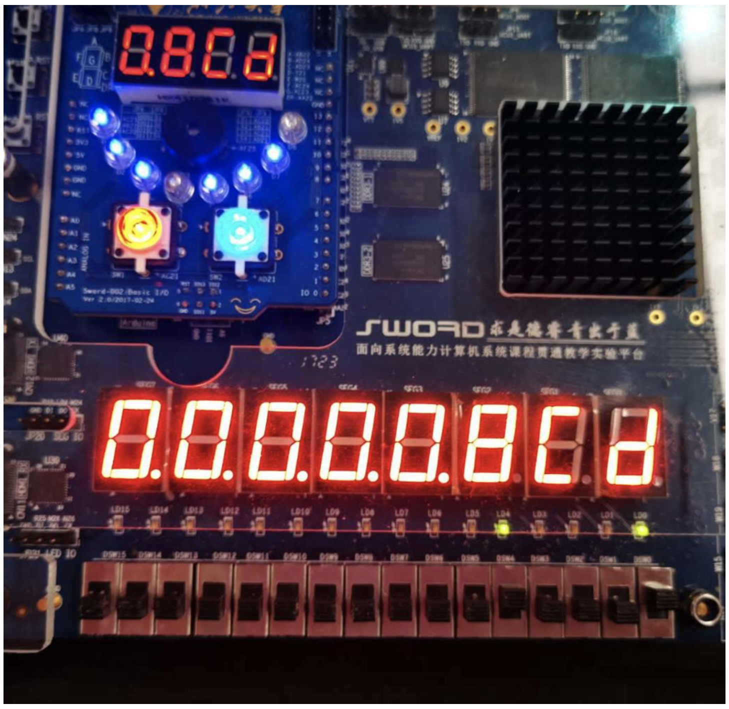
Processes: Top
- Implement Design
  - Translate
  - Map
  - Place & Route
- Generate Programming File
- Configure Target Device
  - Generate Target PROM...

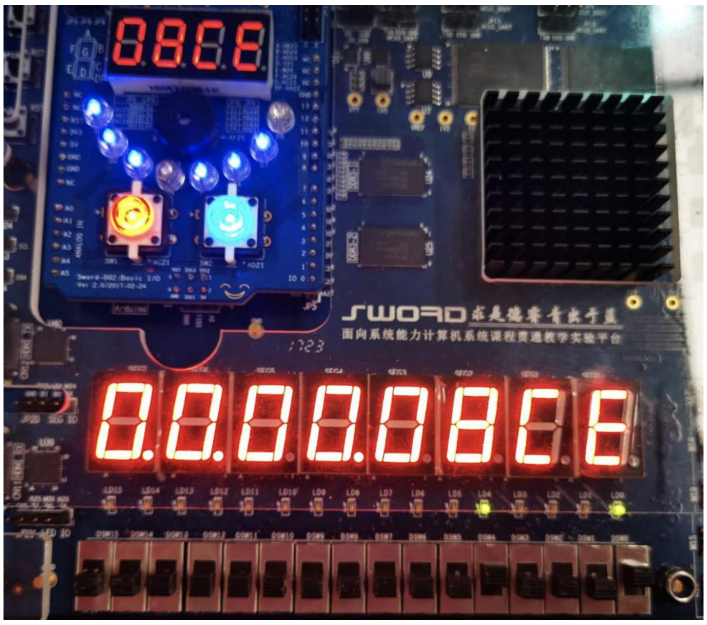
四、实验结果

LED Banner

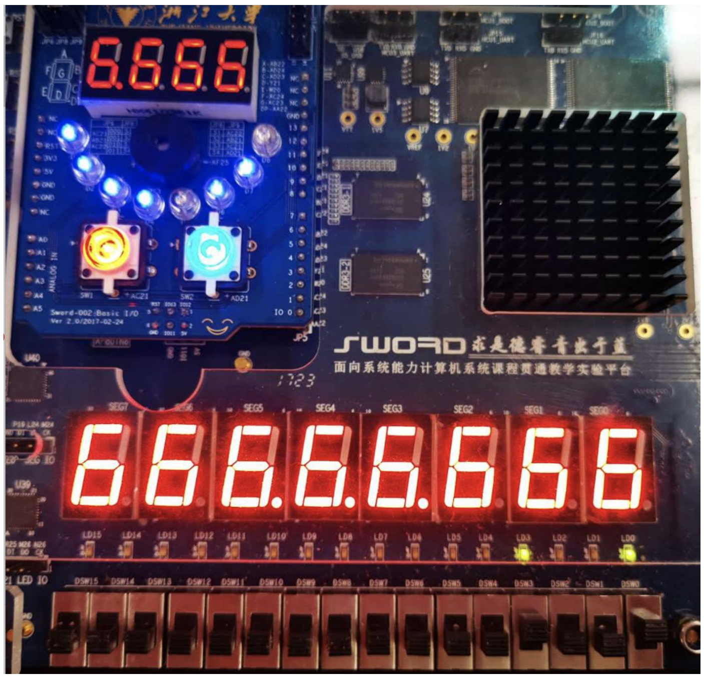


Timer

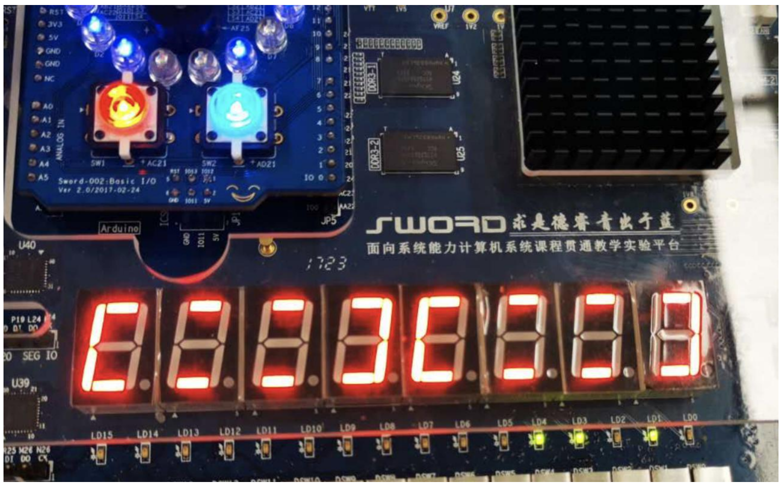




Number traversal



Rectangle change



——成功在新的板子上运行实现之前做的 SCPU