《逻辑与计算机设计基础》实验报告

实验7：电子时钟

1. 实验目的

* 在Sword开发板上用七段数码管以HH-MM-SS格式显示当前时间。
* 使用某个拨位开关(可从sw0～sw15中任意指定一个，但应在文档中加以说明)控制时钟的计时/校准状态。即当开关状态为0时，时钟处于计时状态；当开关状态为1时，时钟处于校准状态。
* 当时钟处于校准状态时，当前时间被锁定。其中HH、MM、SS应有一组闪烁，相当于光标，其闪烁频率可任意设定，并与其他数码管有明显区别。
* 对Sword开发板右侧的小键盘进行按键定义，要求至少定义4个按键，包括+1、-1两个校准键和←、→两个方向键。其中校准键用于对时、分、秒进行校正；方向键用于移动光标。有关按键定义的方法请参考第三部分：键盘去抖模块。
* 每个时间值都有自己的取值范围，要求0≤HH≤23，0≤MM≤59，0≤SS≤59。该电路应能拒绝在校准时产生非法的时间值。

1. 实验任务

用Verilog HDL代码或原理图设计一个电子时钟。

三、实验设备

软件：ISE。

硬件：SWORD开发板、微型计算机。

四、实验原理和电路图

①clock\_1s.v文件：

module clock\_1s(

input clk,

output reg clk\_1s,

output reg [31:0] cnt

);

always @(posedge clk) begin

if (cnt<50\_000\_000)

cnt <= cnt + 1;

else begin

cnt <= 0;

clk\_1s <= ~clk\_1s;

end

end

endmodule

②count\_s.v文件：

module count\_s(

input clk1,

input sw1,

input [3:0] btn\_out,

output reg [7:0] cnt1,

output reg [1:0] co1,

output reg [1:0] btn\_pos

);

reg [7:0] cnt\_temp;

reg [3:0] i;

initial begin

cnt1 <= 0;

cnt\_temp <= 0;

co1 <= 0;

btn\_pos <= 1;

i <= 0;

end

always @(posedge clk1) begin

if (sw1 == 0) begin

btn\_pos <= 1;

if (cnt1 == 8'b1011\_1011) begin

cnt1 <= cnt\_temp;

i <= 0;

end

else if (cnt1 != 8'b1011\_1011) begin

if( (cnt1[3:0] >= 7) && (cnt1[7:4] >= 5) ) begin

co1 <= co1 + 1;

end

if (co1 == 2'b10)

co1 <= 0;

if( (cnt1[3:0] >= 9) && (cnt1[7:4] >= 5) ) begin

cnt1 <= 0;

end

else begin

if (cnt1[7:4] >= 5)

cnt1[3:0] <= cnt1[3:0] + 1;

else begin

if (cnt1[3:0] >= 9) begin

cnt1[3:0] <= 0;

cnt1[7:4] <= cnt1[7:4] + 1;

end

else

cnt1[3:0] <= cnt1[3:0] + 1;

end

end

end

end

//校准

if (sw1 == 1) begin

//左移

if (btn\_out == 4'b1110) begin

if (btn\_pos == 1) begin

btn\_pos <= 2;

end

if (btn\_pos == 2) begin

btn\_pos <= 3;

end

if (btn\_pos == 3) begin

btn\_pos <= 1;

end

end

//右移

if (btn\_out == 4'b1101) begin

if (btn\_pos == 1) begin

btn\_pos <= 3;

end

if (btn\_pos == 2) begin

btn\_pos <= 1;

end

if (btn\_pos == 3) begin

btn\_pos <= 2;

end

end

if (btn\_pos == 1) begin

//闪烁

if (btn\_out == 4'b1111) begin

if (i < 1) begin

if (i == 0)

cnt\_temp <= cnt1;

cnt1 <= 8'b1011\_1011;

i <= i + 1;

end

else begin

cnt1 <= cnt\_temp;

i <= 0;

end

end

//+1

if (btn\_out == 4'b1011) begin

if (cnt1 == 8'b1011\_1011) begin

cnt1 <= cnt\_temp;

i <= 0;

end

if (cnt1 != 8'b1011\_1011) begin

if( (cnt1[3:0] >= 9) && (cnt1[7:4] >= 5) )

cnt1 <= 0;

else begin

if (cnt1[7:4] >= 5) begin

cnt1[3:0] <= cnt1[3:0] + 1;

cnt1[7:4] <= 5;

end

else begin

if (cnt1[3:0] >= 9) begin

cnt1[3:0] <= 0;

cnt1[7:4] <= cnt1[7:4] + 1;

end

else

cnt1[3:0] <= cnt1[3:0] + 1;

end

end

//cnt\_temp <= cnt1;

end

end

//-1

if (btn\_out == 4'b0111) begin

if (cnt1 == 8'b1011\_1011) begin

cnt1 <= cnt\_temp;

i <= 0;

end

if (cnt1 != 8'b1011\_1011) begin

if( (cnt1[3:0] <= 0) && (cnt1[7:4] <= 0) )

cnt1 <= 8'b01011001;

else begin

if (cnt1[7:4] <= 0) begin

cnt1[3:0] <= cnt1[3:0] - 1;

cnt1[7:4] <= 0;

end

else begin

if (cnt1[3:0] <= 0) begin

cnt1[3:0] <= 9;

cnt1[7:4] <= cnt1[7:4] - 1;

end

else begin

cnt1[3:0] <= cnt1[3:0] - 1;

end

end

end

//cnt\_temp <= cnt1;

end

end

//-1完

end //btn\_pos == 1

if ( (btn\_pos == 2) || (btn\_pos == 3) ) begin

if (cnt1 == 8'b1011\_1011) begin

cnt1 <= cnt\_temp;

end

end

end//校准完

end

endmodule

③count\_min.v文件：

module count\_min(

input clk1,

input [1:0] ci1,

input sw1,

input [3:0] btn\_out,

input [1:0] btn\_pos,

output reg [7:0] cnt2,

output reg co2

);

reg [7:0] cnt\_temp;

reg [3:0] i;

initial begin

cnt2 <= 0;

cnt\_temp <= 0;

co2 <= 0;

i <= 0;

end

always @(posedge clk1) begin

if (sw1 == 0) begin

if (cnt2 == 8'b1011\_1011) begin

cnt2 <= cnt\_temp;

i <= 0;

end

else if (cnt2 != 8'b1011\_1011) begin

co2 <= 0;

if (ci1 == 2'b01) begin

if( (cnt2[3:0] >= 9) && (cnt2[7:4] >= 5) )

co2 <= 1;

end

if (ci1 == 2'b10) begin

if( (cnt2[3:0] >= 9) && (cnt2[7:4] >= 5) ) begin

cnt2 <= 0;

end

else begin

if (cnt2[7:4] >= 5)

cnt2[3:0] <= cnt2[3:0] + 1;

else begin

if (cnt2[3:0] >= 9) begin

cnt2[3:0] <= 0;

cnt2[7:4] <= cnt2[7:4] + 1;

end

else

cnt2[3:0] <= cnt2[3:0] + 1;

end

end

end

end//cnt2

end//sw1 == 0完

//校准

if (sw1 == 1) begin

if (btn\_pos == 2) begin

//闪烁

if (btn\_out == 4'b1111) begin

if (i < 1) begin

if (i == 0)

cnt\_temp <= cnt2;

cnt2 <= 8'b1011\_1011;

i <= i + 1;

end

else begin

cnt2 <= cnt\_temp;

i <= 0;

end

end//btn\_out == 4'b1111

//+1

if (btn\_out == 4'b1011) begin

if (cnt2 == 8'b1011\_1011) begin

cnt2 <= cnt\_temp;

i <= 0;

end

if (cnt2 != 8'b1011\_1011) begin

if( (cnt2[3:0] >= 9) && (cnt2[7:4] >= 5) )

cnt2 <= 0;

else begin

if (cnt2[7:4] >= 5) begin

cnt2[3:0] <= cnt2[3:0] + 1;

cnt2[7:4] <= 5;

end

else begin

if (cnt2[3:0] >= 9) begin

cnt2[3:0] <= 0;

cnt2[7:4] <= cnt2[7:4] + 1;

end

else begin

cnt2[3:0] <= cnt2[3:0] + 1;

end

end

end//

end//cnt2 != 8'b1011\_1011

end//btn\_out == 4'b1011

//-1

if (btn\_out == 4'b0111) begin

if (cnt2 == 8'b1011\_1011) begin

cnt2 <= cnt\_temp;

i <= 0;

end

if (cnt2 != 8'b1011\_1011) begin

if( (cnt2[3:0] <= 0) && (cnt2[7:4] <= 0) )

cnt2 <= 8'b01011001;

else begin

if (cnt2[7:4] <= 0) begin

cnt2[3:0] <= cnt2[3:0] - 1;

cnt2[7:4] <= 0;

end

else begin

if (cnt2[3:0] <= 0) begin

cnt2[3:0] <= 9;

cnt2[7:4] <= cnt2[7:4] - 1;

end

else begin

cnt2[3:0] <= cnt2[3:0] - 1;

end

end

end

end//cnt2 != 8'b1011\_1011

end//btn\_out == 4'b0111

end//btn\_pos == 2

if ( (btn\_pos == 1) || (btn\_pos == 3) ) begin

if (cnt2 == 8'b1011\_1011) begin

cnt2 <= cnt\_temp;

end

end

end//sw1 == 1

end//always

endmodule

④count\_hour.v文件：

module count\_hour(

input clk1,

input ci2,

input sw1,

input [3:0] btn\_out,

input [1:0] btn\_pos,

output reg [7:0] cnt3

);

reg [7:0] cnt\_temp;

reg [3:0] i;

initial begin

cnt3 <= 0;

cnt\_temp <= 0;

i <= 0;

end

always @(posedge clk1) begin

if (sw1 == 0) begin

if (cnt3 == 8'b1011\_1011) begin

cnt3 <= cnt\_temp;

i <= 0;

end

else if (cnt3 != 8'b1011\_1011) begin

if (ci2 == 1) begin

if ( (cnt3[3:0] >= 3) && (cnt3[7:4] >= 2) )

cnt3 <= 0;

else begin

if(cnt3[7:4] >= 2)

cnt3[3:0] <= cnt3[3:0] + 1;

else begin

if (cnt3[3:0] >= 9) begin

cnt3[3:0] <= 0;

cnt3[7:4] <= cnt3[7:4] + 1;

end

else

cnt3[3:0] <= cnt3[3:0] + 1;

end

end

end

end // cnt3 != 8'b1011\_1011

end//sw1 == 0

//校准

if (sw1 == 1) begin

if (btn\_pos == 3) begin

//闪烁

if (btn\_out == 4'b1111) begin

if (i < 1) begin

if (i == 0)

cnt\_temp <= cnt3;

cnt3 <= 8'b1011\_1011;

i <= i + 1;

end

else begin

cnt3 <= cnt\_temp;

i <= 0;

end

end//btn\_out == 4'b1111

//+1

if (btn\_out == 4'b1011) begin

if (cnt3 == 8'b1011\_1011) begin

cnt3 <= cnt\_temp;

i <= 0;

end

if (cnt3 != 8'b1011\_1011) begin

if( (cnt3[3:0] >= 3) && (cnt3[7:4] >= 2) )

cnt3 <= 0;

else begin

if (cnt3[7:4] >= 2) begin

cnt3[3:0] <= cnt3[3:0] + 1;

cnt3[7:4] <= 2;

end

else begin

if (cnt3[3:0] >= 9) begin

cnt3[3:0] <= 0;

cnt3[7:4] <= cnt3[7:4] + 1;

end

else begin

cnt3[3:0] <= cnt3[3:0] + 1;

end

end

end

end // cnt3 != 8'b1011\_1011

end // btn\_out == 4'b1011

//-1

if (btn\_out == 4'b0111) begin

if (cnt3 == 8'b1011\_1011) begin

cnt3 <= cnt\_temp;

i <= 0;

end

if (cnt3 != 8'b1011\_1011) begin

if( (cnt3[3:0] <= 0) && (cnt3[7:4] <= 0) )

cnt3 <= 8'b00100011;

else begin

if (cnt3[7:4] <= 0) begin

cnt3[3:0] <= cnt3[3:0] - 1;

cnt3[7:4] <= 0;

end

else begin

if (cnt3[3:0] <= 0) begin

cnt3[3:0] <= 9;

cnt3[7:4] <= cnt3[7:4] - 1;

end

else begin

cnt3[3:0] <= cnt3[3:0] - 1;

end

end

end

end//cnt3 != 8'b1011\_1011

end//btn\_out == 4'b0111

end//btn\_pos == 3

if ( (btn\_pos == 1) || (btn\_pos == 2) ) begin

if (cnt3 == 8'b1011\_1011) begin

cnt3 <= cnt\_temp;

end

end

end//sw1 == 1

end

endmodule

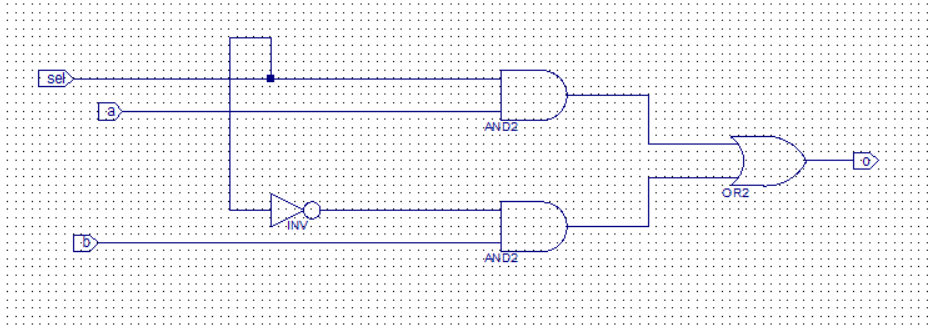


图7-1 mux2t1.sch(1位2选一多路选择器)

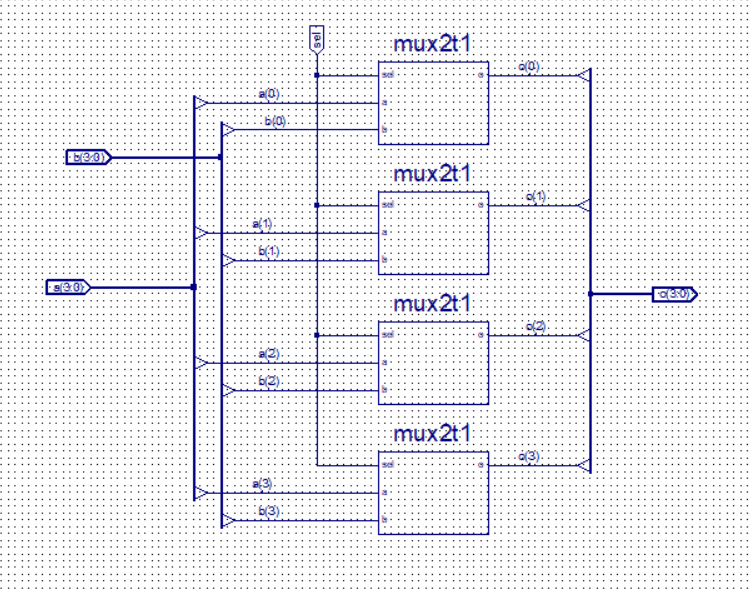


图7-2 mux2t1\_4.sch（4位2选1多路选择器）

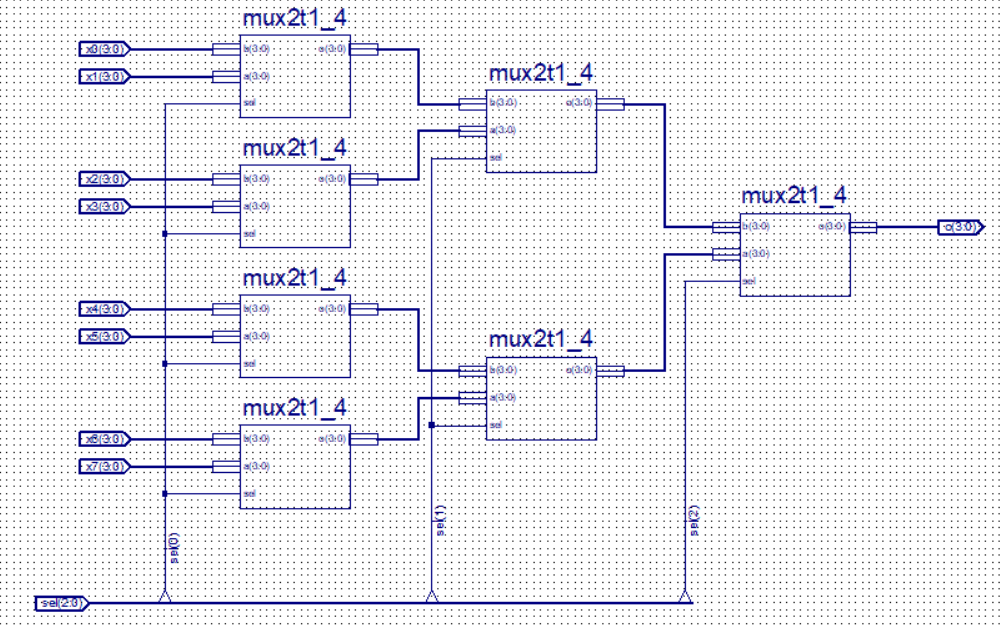


图7-3 mux8t1\_4.sch（4位8选1多路选择器）

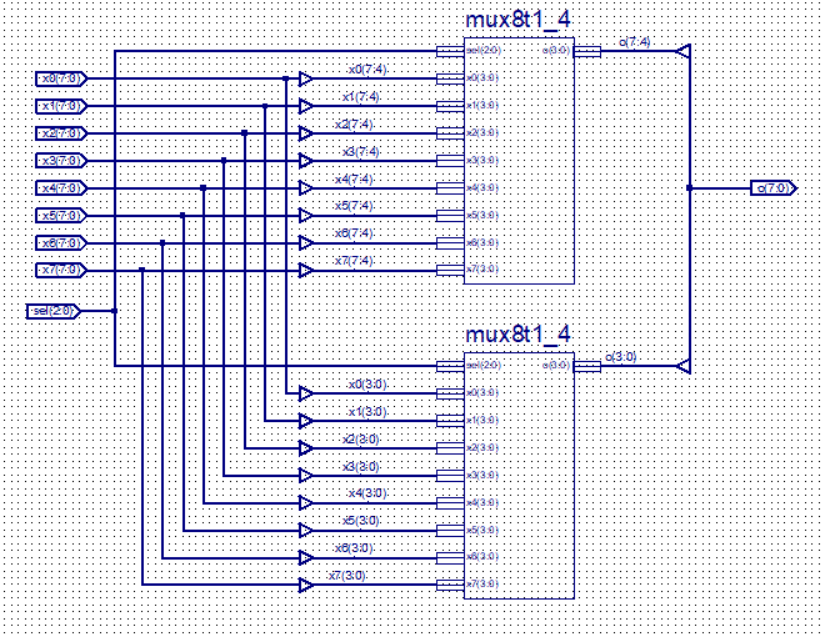


图7-4 mux8t1\_8.sch（8位8选1多路选择器）

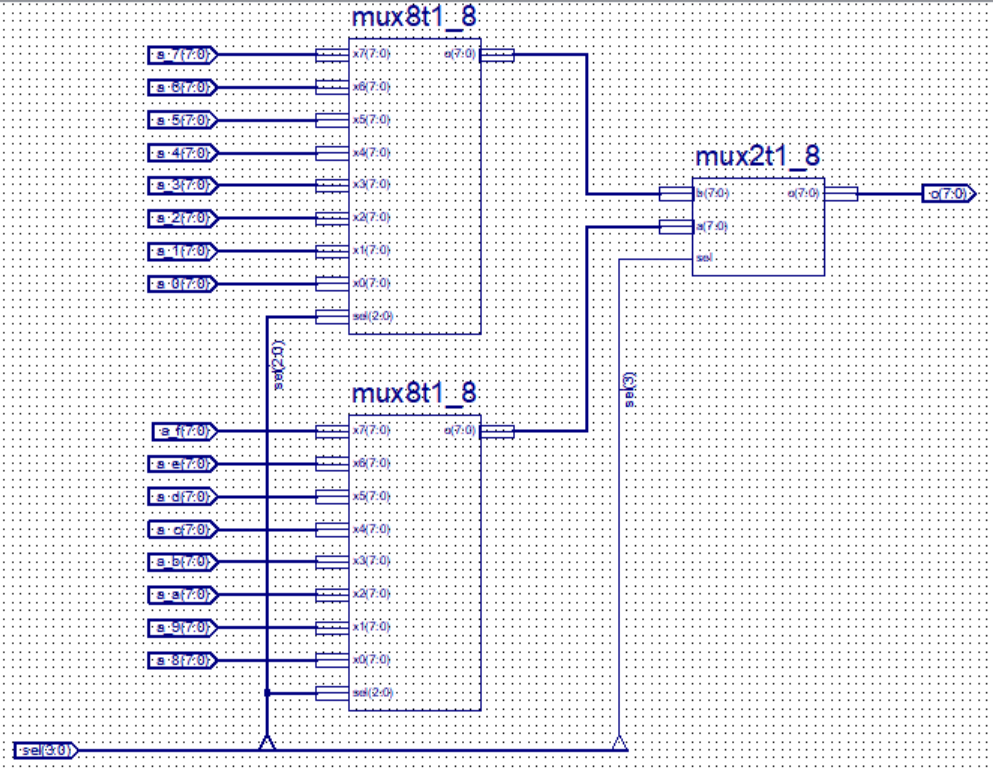


图7-5 mux16t1\_8.sch（8位16选1多路选择器）

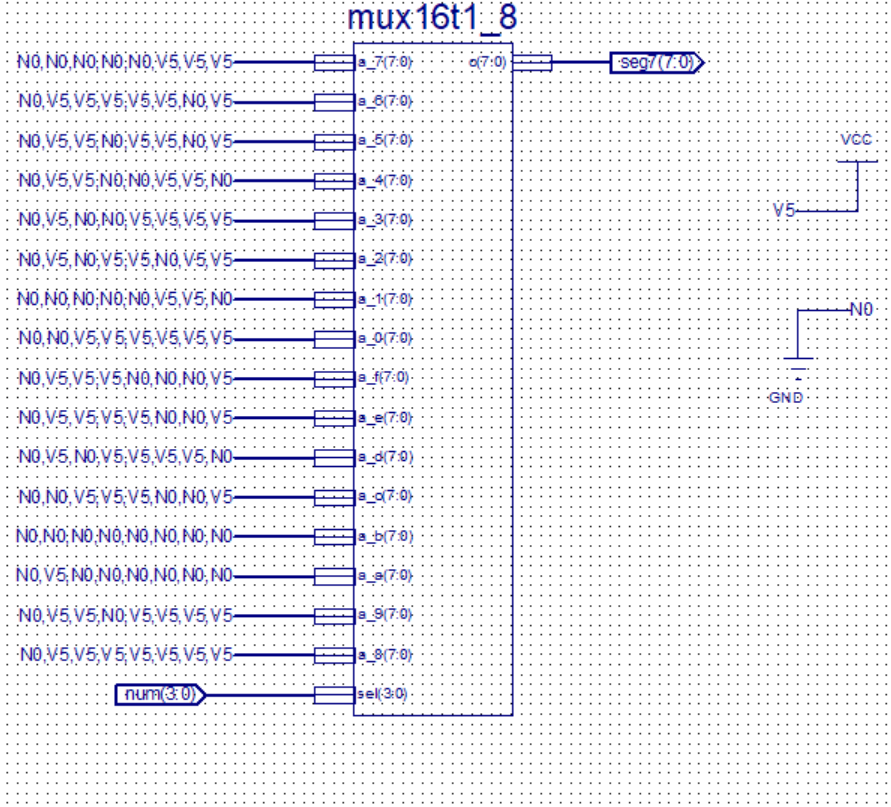


图7-6 seg7\_decoder.sch（七段译码器电路）

⑤Sword\_Anti\_jitter.v文件：

module Sword\_Anti\_jitter(

input clk,

input wire [3:0] btn,

output reg [3:0] btn\_out

);

reg [31:0] counter;

reg [4:0] btn\_temp;

always @(posedge clk) begin

btn\_temp <= btn;

if (btn\_temp != btn) begin

counter <= 32'h0000\_0000;

end

else if (counter < 1000000) begin

counter<=counter+1;

end

else begin

btn\_out <= btn;

end

end

endmodule

⑤top.v文件：

module top(

input clk,

input s,

input [3:0] btn,

output seg\_data,

output seg\_clk,

output seg\_clr,

output seg\_pen

);

wire clk\_1sec;

wire [31:0] clk\_div;

wire [1:0] co1;

wire co2;

wire [23:0] cnt;

wire [3:0] btn\_out;

wire [1:0] btn\_pos;

wire [63:0] seg7\_64bits;

clock\_1s U1(.clk(clk), .clk\_1s(clk\_1sec), .cnt(clk\_div));

count\_s U2(.clk1(clk\_1sec), .sw1(s), .btn\_out(btn\_out), .cnt1(cnt[7:0]), .co1(co1), .btn\_pos(btn\_pos));

count\_min U3(.clk1(clk\_1sec), .ci1(co1), .sw1(s), .btn\_out(btn\_out), .btn\_pos(btn\_pos), .cnt2(cnt[15:8]), .co2(co2));

count\_hour U4(.clk1(clk\_1sec), .ci2(co2), .sw1(s), .btn\_out(btn\_out), .btn\_pos(btn\_pos), .cnt3(cnt[23:16]));

seg7\_decoder U5(.num(cnt[3:0]), .seg7(seg7\_64bits[7:0]));

seg7\_decoder U6(.num(cnt[7:4]), .seg7(seg7\_64bits[15:8]));

seg7\_decoder U7(.num(4'b1010), .seg7(seg7\_64bits[23:16]));

seg7\_decoder U8(.num(cnt[11:8]), .seg7(seg7\_64bits[31:24]));

seg7\_decoder U9(.num(cnt[15:12]), .seg7(seg7\_64bits[39:32]));

seg7\_decoder U10(.num(4'b1010), .seg7(seg7\_64bits[47:40]));

seg7\_decoder U11(.num(cnt[19:16]), .seg7(seg7\_64bits[55:48]));

seg7\_decoder U12(.num(cnt[23:20]), .seg7(seg7\_64bits[63:56]));

Sword\_Anti\_jitter U13(.clk(clk), .btn(btn), .btn\_out(btn\_out));

P2S #(.DATA\_BITS(64), .DATA\_COUNT\_BITS(6), .DIR(0))

U14(

.clk(clk\_div[1]),

.rst(1'b0),

.Start(clk\_div[20]),

.PData(~seg7\_64bits),

.sclk(seg\_clk),

.sclrn(seg\_clr),

.sout(seg\_data),

.EN(seg\_pen)

);

endmodule

五、实验步骤和仿真结果

1. 打开”Xilinx ISE 14.7”软件，新建一个名为”electronic\_clock”的工程。点击 Next 按钮进入下一页：Project Settings，选择所使用的芯片及综合、仿真工具。我们选用 Kintex7 XC7160T 芯片，采用FFG676 封装。另外，我们选择 Verilog 作为默认的硬件描述语言。
2. 在工程管理区将View设置为Implementatio，新建源文件，选择Verilog module，新建文件“clock\_1s.v”、“count\_s.v”、“count\_min.v”、“count\_hour.v”(具体内容可依顺序分别参见第四部分对应部分①到④)。
3. 在工程管理区将View设置为Implementatio，依次新建文件“mux2t1.sch”（图7-1）、“mux2t1\_4.sch”（图7-2）、“mux8t1\_4.sch”（图7-3）、“mux8t1\_8.sch”（图7-4）、“mux16t1\_8.sch”（图7-5），每依次完成一个原理图设计后就生成相应的符号。
4. 在工程管理区将View设置为Implementatio，新建文件“seg7\_decoder.sch”（图7-6），并在完成原理图设计后生成相应的符号（原理图中a\_a、a\_b有改动，改动之后分别对应显示“-”、“ ”）。
5. 将由老师提供的“P2S.ngc”、“P2S\_IO.v”拷贝添加到工程中。
6. 在工程管理区将View设置为Implementatio，新建源文件，选择Verilog module，新建文件“Sword\_Anti\_jitter.v”(具体内容可参见第四部分对应部分⑤)。
7. 新建源文件，选择 Verilog module，命名 top(具体内容可参见第四部分对应部分⑥)，完善代码后保存该文件。
8. 在工程管理区中单击鼠标右键，在弹出菜单中选择New Source命令，选择Implementation Constraints File，输入文件名：myucf，系统会生成一个空白的约束文件并打开，接下来我们就可以为设计添加各种约束,添加如下代码并保存该文件：

#系统时钟

NET "clk" LOC = AC18 | IOSTANDARD = LVCMOS18;

#七段码串行接口

NET "seg\_clk" LOC = M24 | IOSTANDARD = LVCMOS33;

NET "seg\_clr" LOC = M20 | IOSTANDARD = LVCMOS33;

NET "seg\_data" LOC = L24 | IOSTANDARD = LVCMOS33;

NET "seg\_pen" LOC = R18 | IOSTANDARD = LVCMOS33;

#拨位开关

NET "s" LOC = AA10 | IOSTANDARD = LVCMOS15;

# Buttons

NET "btn[0]" LOC = V18 |IOSTANDARD = LVCMOS18;

NET "btn[1]" LOC = V19 |IOSTANDARD = LVCMOS18;

NET "btn[2]" LOC = V14 |IOSTANDARD = LVCMOS18;

NET "btn[3]" LOC = W14 |IOSTANDARD = LVCMOS18;

1. 完成了设计后就可以进行综合。选中要综合的模块 top，在过程管理区中双击Implement Design，若上述操作无误，双击 Generate Programming File可以产生包含硬件信息的比特流文件。

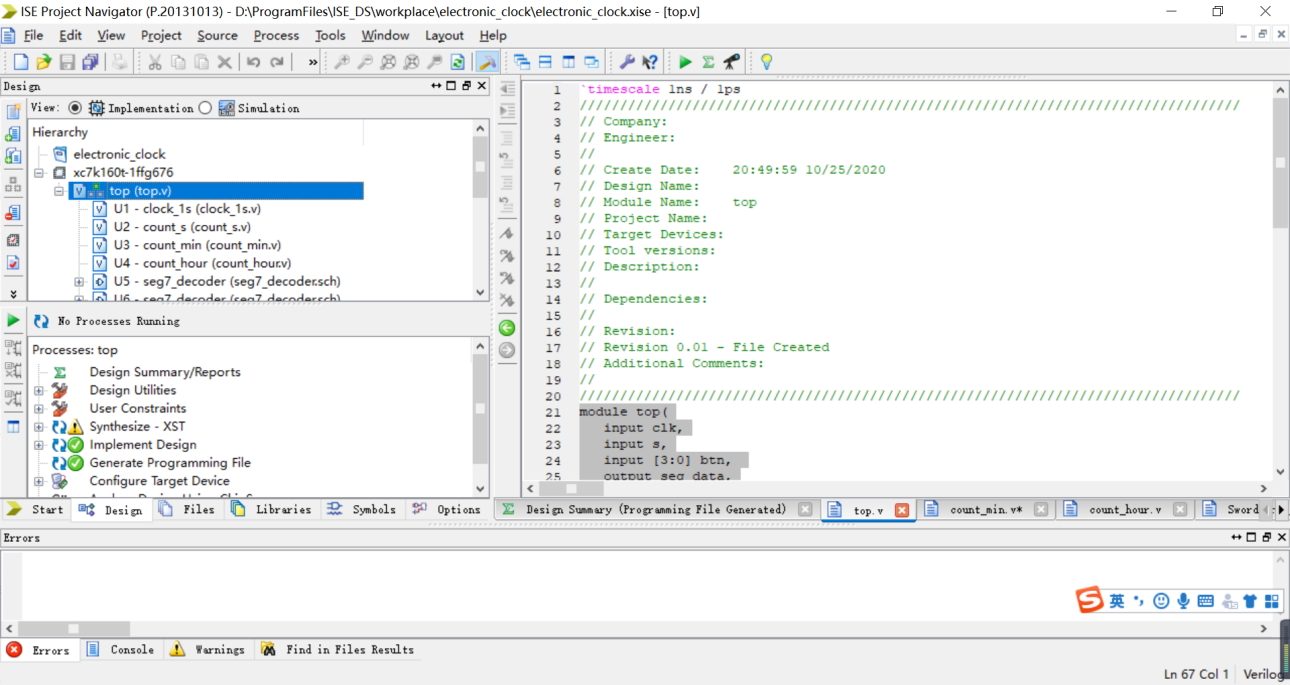


图7-7 综合结果

1. 等待生成比特流文件，将板卡正确连接到 PC 并打开电源，将比特文件下载到板卡上观察实验结果。在工程管理区中选中 top，然后在下方的 process：top 中双击 Configure Target Device，打开下载工具 IMPACT，遇到警告直接 OK 略过。然后系统会自动启动 IMPACT。先双击 IMPACT Flows 中的 Boundary Scan，然后在空白处单击右键，然后点击 Initialize Chain 初始化板卡，对弹出的提示点击 NO ，在接下来弹出的窗口选择 cancel，选中芯片，单击右键,选择 Launch File Assignment Wizard，在工程目录下找到生成的bit 文件，选中然后点击 open，下载 bit文件，如遇提示Attach SPI BPI PROM，选择NO，此时会回到软件主界面，选中芯片单击右键，选择 program，等待程序下载完成，尝试拨动开关，观察结果。

六、开发板运行结果记录

表 7-1

|  |  |
| --- | --- |
| DSW0 为 0  计时状态 | IMG_256 |
| DSW0 为 1  校准状态（秒闪烁） | IMG_256 |
| DSW0 为 1  校准状态（分闪烁） | IMG_256 |
| DSW0 为 1  校准状态（时闪烁） | IMG_256 |

-1