计数器译码器实验报告

朱河勤 PB16030899

2017 年 11 月 10 日

目录

1 实验目的

1 了解编码器、计数器的工作原理及电路结构

2 能够根据实际要求，设计出相应的逻辑电路

3 学会使用Verilog语言设计时序逻辑电路

4 进一步熟悉FPGA及ISE、Vivado工具的使用

2 设备外设

时钟（50MHz or 100MHz）、按键（复位用）、拨动开关（4个）、LED灯（4个）、7段数码管（1个）

3 实验要求

1 实现一编码逻辑，将4个拨动开关的状态按16进制显示在7段数码管上

2 实现一带异步复位的30位计数器，每个时钟周期计数值加1，复位时计数器的值为：$0x3333_3333$

3 将计数器的[29:26]输出到4个LED上，同时做为编码器的输入，显示到七段数码管上

4 实验结果

成功满足要求，在50mhz时钟周期下增加1，实现了计数器，并将高4位信号输出到八段码，LED,使得每$1/50000000 * 2^{26}$约1.3S 改变状态，并有异步复位功能。

5 实验分析

1 八段码是共阳极的，所以在写verilog代码时应该输入低电平信号

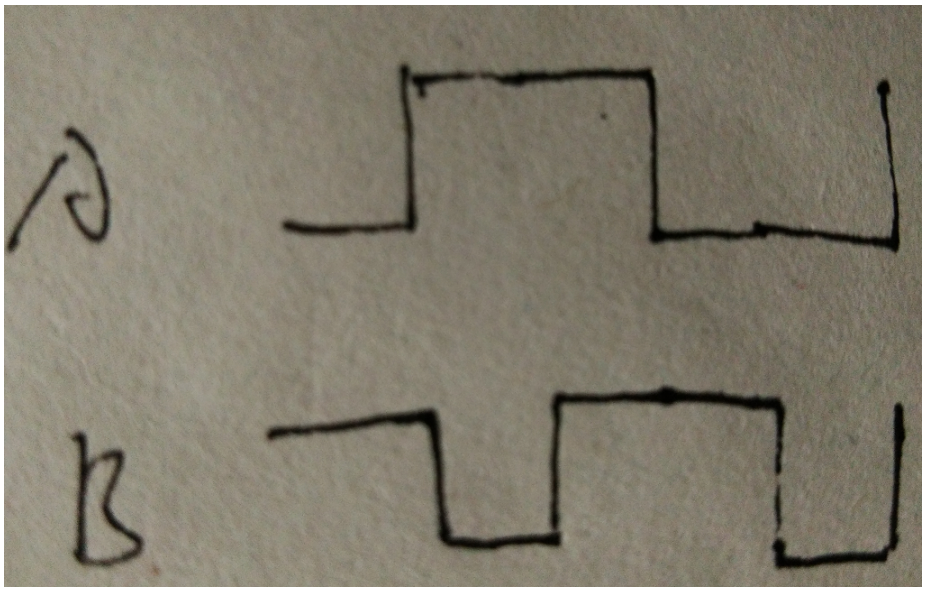
2 由于是递增的计数器，所以应该捕捉时钟下降沿

6 额外要求

6.1 代码注释

```
input [2:0] a;
output reg [7:0] y;
always@(a) //捕捉a的变化
begin
        case(a[2:0]) //case 语句
        3'b000:y=8'b00000001;
        3'b001:y=8'b00000010;
        3'b010:y=8'b00000100;
        3'b011:y=8'b00001000;
        3'b100:y=8'b00010000;
        3'b101:y=8'b00100000;
        3'b110:y=8'b01000000;
        3'b111:y=8'b10000000;
        default:y=8'b00000001;
        endcase //最好有default ，译码为0
end //此module 实现了3-8译码器，但是没有使能端，扩展输入端
endmodule
```

6.2 波形图



7 源代码

```
module top(
        input rst,clk,
        output [7:0] seg ,[2:0] AN,[3:0] LED
    );
    wire [29:0] pad;
    cnt2  cnt_ins(.out(pad),.clk(clk),.rst(~rst));
    assign AN=3'b000;
    assign LED[0]=pad[29],
           LED[1]=pad[28],
           LED[2]=pad[27],
           LED[3]=pad[26];
    decoder  dec_ins(.in(pad[29:26]),.out(seg));
```

```
module cnt2(
    input clk50m,reset,
    output reg [29:0] out
    );
    always @(posedge clk50m,posedge reset)
        if(reset) out<=8'h3333_3333;
        else out <=out+1;
endmodule
```

```
module decoder(
        input [3:0] in,
        output reg  [7:0] out
        );
        always@(*)
            begin
                case(in)
                0:out<=8'b1100_0000;
                1:out<=8'b1111_1001;
                2:out<=8'b1010_0100;
                3:out<=8'b1011_0000;
                4:out<=8'b1001_1001;
                5:out<=8'b1001_0010;
                6:out<=8'b1000_0010;
                7:out<=8'b1101_1000;
                8:out<=8'b1000_0000;
                9:out<=8'b1001_0000;
                10:out<=8'b1000_1000;
                11:out<=8'b1000_0011;
                12:out<=8'b1100_0110;
```

```
                13:out<=8'b1010_0001;
                14:out<=8'b1000_0110;
                15:out<=8'b1000_1110;
                endcase
            end
endmodule
```