一、投票表决器

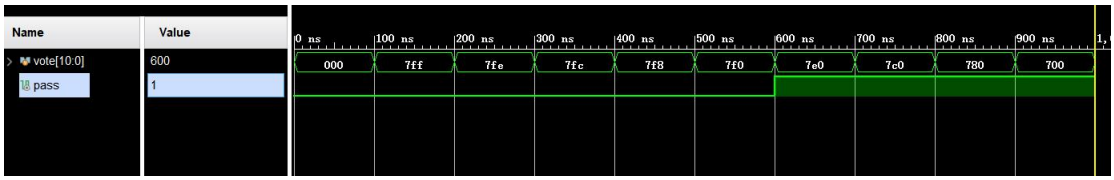
verilog 代码：

```
23  module voter11(pass, vote);
24      output pass;
25      input [10:0] vote;
26      reg [3:0] sum=0;
27      integer i;
28      assign pass=(sum>=6)?1:0;
29      always @(vote)
30      begin
31      for(i=0;i<11;i=i+1) begin
32          if(vote[i]) sum <= sum+1;
33      end
34      end
35  endmodule
```

仿真代码：

```
23  module test(
24
25      );
26      reg[10:0] vote=0;
27      wire pass;
28      voter11 u(pass,vote);
29      initial begin
30          #100 vote = 11'b1111_1111_111;
31          #100 vote = 11'b1111_1111_110;
32          #100 vote = 11'b1111_1111_100;
33          #100 vote = 11'b1111_1111_000;
34          #100 vote = 11'b1111_1110_000;
35          #100 vote = 11'b1111_1100_000;
36          #100 vote = 11'b1111_1000_000;
37          #100 vote = 11'b1111_0000_000;
38          #100 vote = 11'b1110_0000_000;
39          #100 vote = 11'b1100_0000_000;
40          #100 vote = 11'b1000_0000_000;
41          #100 vote = 11'b0000_0000_000;
42      end
43
44  endmodule
```

仿真结果：



整个项目见附录 -----> 

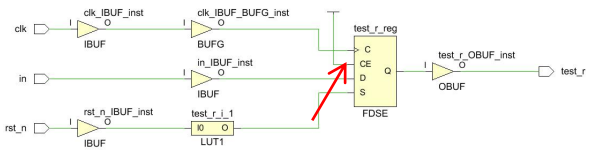
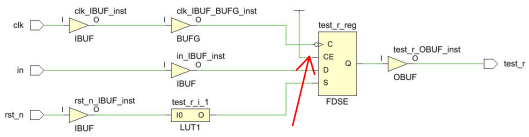
二、代码及生成电路差异

<table>
<tr><td colspan="2">下降沿与上升沿比较</td></tr>
<tr><td>上升沿</td><td>下降沿</td></tr>
<tr><td>Verilog 代码：

```
23 module tihuidaima(
24 input clk,
25 input rst_n,
26 input in,
27 output reg test_r
28     );
29 always @(posedge clk) begin
30     if(!rst_n) begin
31         test_r <=1;
32     end
33     else begin
34         test_r <=in;
35     end
36 end
37 endmodule
```

</td><td>Verilog 代码：

```
23 module tihuidaima(
24 input clk,
25 input rst_n,
26 input in,
27 output reg test_r
28     );
29 always @(negedge clk) begin
30     if(!rst_n) begin
31         test_r <=1;
32     end
33     else begin
34         test_r <=in;
35     end
36 end
37 endmodule
```

</td></tr>
<tr><td>综合电路图：
 
</td><td>综合电路图：
 
</td></tr>
<tr><td colspan="2">分析：

使用上升沿和下降沿触发所生成的电路唯一的区别就是触发器的触发</td></tr>
</table>

方式不同，上升沿触发器电路图没有小圈，代表触发方式为上升沿触发；下降沿电路图触发器有小圈，代表触发方式为下降沿触发。

使用上升沿或下降沿触发优点：

1、提高触发器的可靠性。

2、增强抗干扰能力，触发器仅仅取决于 CLK 信号的下降沿（或上升沿）到达时刻输入信号的状态，而在此之前和之后输入状态的变化对触发器的状态没有影响。

3、可以有效防止空翻现象。

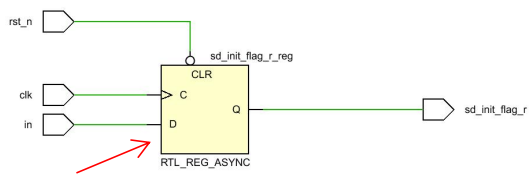
有无 else 等完整语句影响

有 else 等完整语句

Verilog 代码：

```
module tihuidaima(
input clk,
input rst_n,
input in,
output reg sd_init_flag_r
    );
always @(posedge clk or negedge rst_n) begin
    if(!rst_n) begin
        sd_init_flag_r <= 1'b0;
    end
    else begin
        if(in)
            sd_init_flag_r <= 1;
        else
            sd_init_flag_r <= 0;
    end
end
endmodule
```

RTL 电路图：

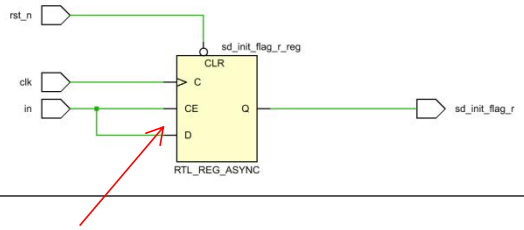


无 else 等完整语句

Verilog 代码：

```
module tihuidaima(
input clk,
input rst_n,
input in,
output reg sd_init_flag_r
    );
always @(posedge clk or negedge rst_n) begin
    if(!rst_n) begin
        sd_init_flag_r <= 1'b0;
    end
    else begin
        if(in)
            sd_init_flag_r <= 1;
    end
end
endmodule
```

RTL 电路图：



无 else 等完整语句的缺点：

1、输入状态可能多次变化，容易产生毛刺，增加了下一级电路的不确定性；

2、在大部分 FPGA 的资源中，可能需要比触发器更多的资源去实现；

同步和异步比较

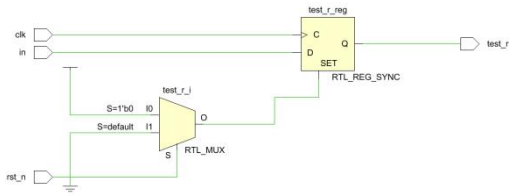
同步：

```
23  module tihuidaima(
24  input clk,
25  input rst_n,
26  input in,
27  output reg test_r
28      );
29  always @(posedge clk) begin
30      if(!rst_n) begin
31          test_r <=1;
32      end
33      else begin
34          test_r <=in;
35      end
36  end
37  endmodule
```

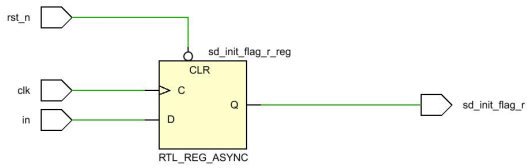
异步：

```
23  module tihuidaima(
24  input clk,
25  input rst_n,
26  input in,
27  output reg sd_init_flag_r
28      );
29  always @(posedge clk or negedge rst_n) begin
30      if(!rst_n) begin
31          sd_init_flag_r <= 1'b0;
32      end
33      else begin
34          if(in)
35              sd_init_flag_r <= 1;
36          else
37              sd_init_flag_r <= 0;
38      end
39  end
40  endmodule
```

RTL 电路图:



RTL 电路图：



分析：

同步时序电路只有一个时钟信号，置位信号依靠时钟信号，因此置位信号发生变化时并不是立即变化的。 而异步时序电路有多个时钟信号，复位信号不依靠时钟信号，因此复位信号一旦发生变化就能立即变换。

优缺点：

同步：

1、电路稳定性强，抗干扰能力强。

2、但需要更多的资源，如上图多需要一个选择器。

异步：

1、无需额外的逻辑资源，且复位信号不需要时钟信号。

2、抗干扰能力较弱，易受外界干扰，人为电路设计较复杂