奋斗的小孩之 altera 系列

第十九篇 序列检测机

对于每一个的小实验，我们都可以把它看作是一个小项目，逐步的去分析，设计，调试，最后完成功能。下面我们就开始我们的“小项目”。

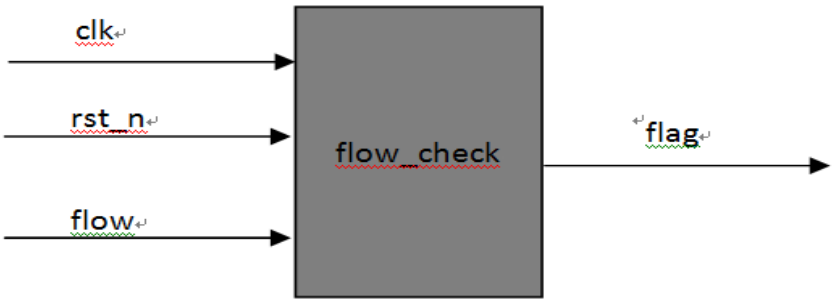
项目名称：序列检测机。

具体要求：

1. 检测序列为：1101。
2. 检测出序列后发出同步高脉冲。
3. 若输入序列中出现 ••••• 1101101 ••••• 则输出两个同步高脉冲。

同步高脉冲：一个时钟周期的高电平。

通过分析上述的“项目名称”和“具体要求”，我们可以设计出如下的架构：



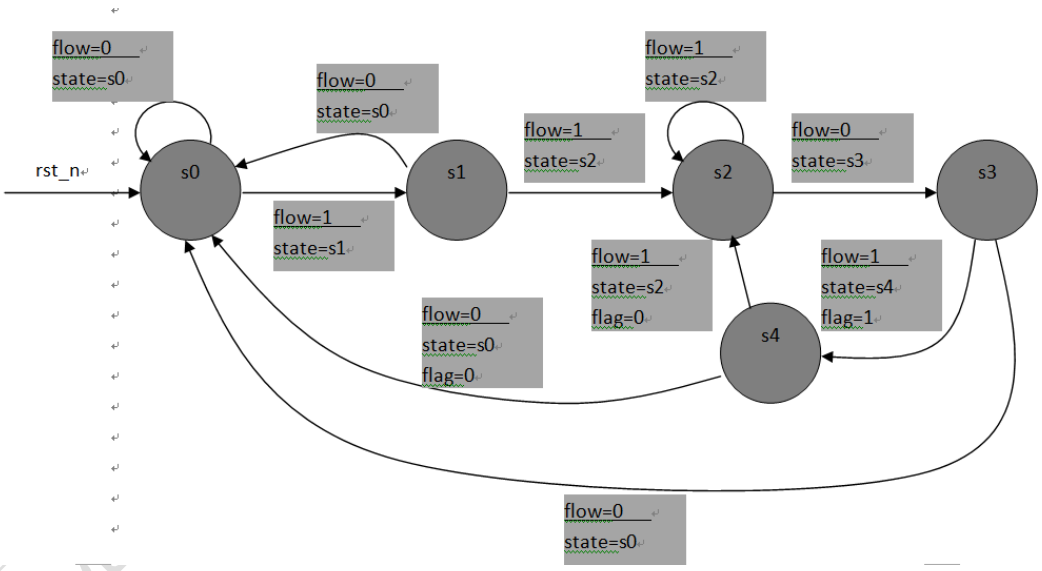
flow：序列输入

flag：同步高脉冲信号

要求 flow 输入的频率必须和驱动的频率相同。假设序列一秒钟输入一次，而系统一秒中检测 5 次，每一个值都会被检测 5 次。那么什么时候也不会出现 1101 了。

系统设计：

1. 工程的名称：flow_check。
2. 状态转移图如下：



设计代码如下：

```
/*
模块名称：flow_check
模块功能：检测出序列“1101”，发出同步高脉冲。
编写时间：2016-08-13
```

```
   作者：至芯科技----奋斗的小孩
   邮箱：zxopenhxs@126.com
   */
module flow_check (clk, rst_n, flow, flag);

   input clk;
   input rst_n;
   input flow;//输入序列

   output reg flag;

   reg [2:0] state;

   localparam s0 = 3'b000;
   localparam s1 = 3'b001;
   localparam s2 = 3'b010;
   localparam s3 = 3'b011;
   localparam s4 = 3'b100;

   always @ (posedge clk or negedge rst_n)
      begin
```

```
if (!rst_n)
   begin
      flag <= 1'b0;
      state <= s0;
   end
else
   begin
      case (state)
         s0 : begin//第一个‘1’
               if (flow == 1'b0)
                  state <= s0;
               else
                  state <= s1;
            end

         s1 : begin//第二个‘1’
               if (flow == 1'b0)
                  state <= s0;
               else
                  state <= s2;
            end
```

```
s2 : begin//’ 0 ‘
        if (flow == 1’b0)
            state <= s3;
        else
            state <= s2;
    end

s3 : begin//’ 1 ‘
        if (flow == 1’b0)
            begin
                state <= s0;
            end
        else
            begin
                state <= s4;
                flag <= 1’b1;
            end
    end
```

s4 : begin//检测出序列“1101”后的第一个输入

```
                if (flow == 1'b0)
                    begin
                        state <= s0;
                        flag <= 1'b0;
                    end
                else
                    begin
                        flag <= 1'b0;
                        state <= s2;
                    end
            end

        default : state <= s0;

    endcase
  end
end

endmodule
```

解析：

s2 状态：如果可以进入 s2 状态，证明序列中已经有了连续的两

个‘1’。若下一个输入为‘0’，应该进入下一个状态。若下一个输入为‘1’，那么序列中出现了三个连续的‘1’，我们可以认为后面的两个‘1’是我们所要检测序列“1101”中前面两个‘1’。

s4 状态：如果可以进入 s4 状态，证明序列中已经有了“1101”。若下一个输入为‘0’，应该回到‘s0’状态。若下一个输入为‘1’，那么正好可以和检测出的“1101”中最后一个‘1’一起被看作我们所要检测“1101”的前两个‘1’（项目要求：输入序列中出现 ····· 1101101 ····· 则输出两个同步高脉冲。）。

激励代码如下：

```
/*
模块名称：flow_check_tb
模块功能：为 flow_check 模块提供激励信号
编写时间：2016-08-12
作者：至芯科技----奋斗的小孩
邮箱：zxopenhxs@126.com
*/
`timescale 1ns/1ps

module flow_check_tb;

    reg clk;
```

```
reg rst_n;
reg flow;

wire flag;

initial begin
   clk = 1'b1;
   rst_n =1'b0;
   flow = 1'b0;
   # 200.1
   rst_n = 1'b1;
   # 10//使时钟上升沿和输入序列的中间在同一时刻
   flow = {$random}%2;
   forever begin//一直执行
      # 20 //输入序列的频率和驱动时钟一致
      flow = {$random}%2;
   end
end

initial begin//上电之后 2000ns 停止仿真
   # 2000
```

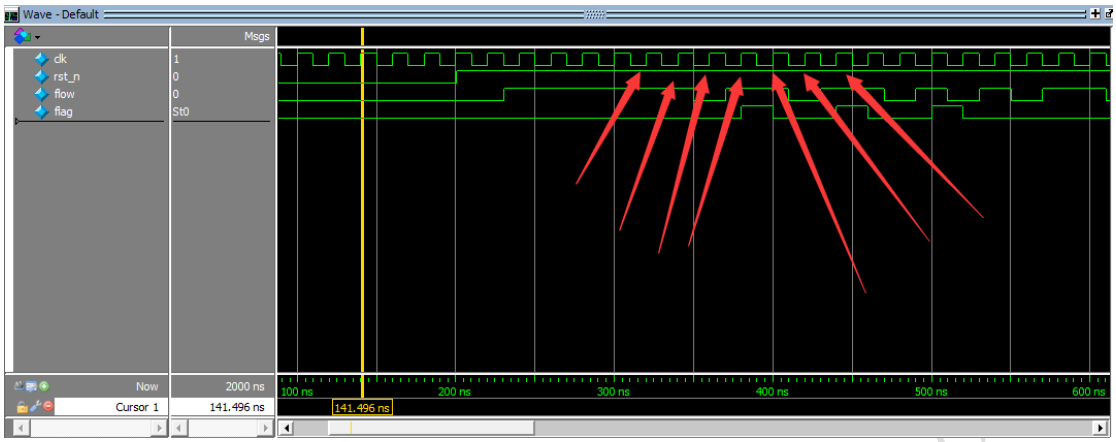
```
    $stop;
end

always # 10 clk = ~clk;

flow_check flow_check_dut(
        .clk(clk),
        .rst_n(rst_n),
        .flow(flow),
        .flag(flag)
    );

endmodule
```

解析：

$random 函数调用时返回一个 32 位的随机数，它是一个带符号的整形数。

```
rand=$random % 60; //产生一个在 -59—59 范围的随机数。
rand={$random} % 60; //通过{}产生 0—59 范围的随机数。
```

仿真波形如下：



从仿真出来的波形来看，当出现“1101”时，flag 拉高了一个时钟周期。当出现“1101101”时，flag 拉高了两次。

设计正确。如果小伙伴的电路原理和笔者的不一样，请自行更改设计。如果还是有不明白的小伙伴可以发邮件到我邮箱或者加群询问。

制作人:奋斗的小孩　　fpga 交流群:282124839