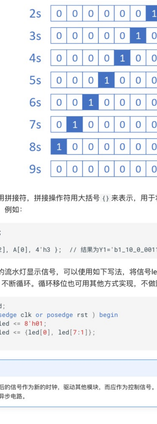
实验原理

1、流水灯实现

流水灯控制原理如下图，共8个灯，每次只亮1个灯，图中1代表亮灯，0代表不亮。第1s的时候最右边的灯亮，第2s的时候左移到下一个灯，不断移动，到第9s的时候又从最右边的灯开始亮。不断循环，需要间隔为1秒的流水灯切换控制信号，且间隔1由输入控制，有了间隔为1的流水灯切换控制信号后，还需要生成流水灯显示信号。



循环移位可以使用拼接符，拼接操作符用大括号 {} 来表示，用于将多个信号拼接成新的信号，信号间用逗号隔开，例如：

```
A = 4'b1010 ;
B = 1'b1 ;
Y1= {B, A[3:2], A[0], 4'h3 };  // 结果为Y1='b1_10_0_0011
```

要实现循环右移的流水灯显示信号，可以使用如下写法，将信号led的最低位移动到最高位，高7位往低位移动1位，不断循环。循环移位也可用其他方式实现，不做限制。

```
reg [7:0] led;
always @ (posedge clk or posedge rst ) begin
    if(rst) led <= 8'h01;
    else    led <= {led[0], led[7:1]};
end
```

注意

不要将计数器分频后的信号作为新的时钟，驱动其他模块。而应作为控制信号，分频信号作为新的时钟会使得整个电路有多个时钟，属异步电路。

2、计数器Verilog实现

计数器Verilog实现模板

```
reg [3:0] cnt;

wire cnt_end = ; // 结束条件
wire cnt_inc = ; // 自增条件

always @ (posedge clk or posedge rst) begin
    if(rst)          cnt <= 4'h0;       // 复位后计数器清零
    else if(cnt_end) cnt <= 4'h0;       // 计数结束后计数器清零
    else if(cnt_inc) cnt <= cnt + 4'h1; // 累加条件满足时，计数器累加
end
```

计数器Verilog实现示例，4bit计数器，button按下启动一次计数，计数0-9，到9停止计数，等待button再次按下重新启动。

```
module counter(
    input wire clk,
    input wire rst,
    input wire button,
    output reg [3:0] cnt
);

reg   cnt_inc;
wire  cnt_end = cnt_inc & (cnt==4'h9);

always @ (posedge clk or posedge rst) begin
    if(rst)          cnt_inc <= 1'b0;
    else if(button)  cnt_inc <= 1'b1;
    else if(cnt_end) cnt_inc <= 1'b0;
end

always @ (posedge clk or posedge rst) begin
    if(rst)          cnt <= 4'h0;
    else if(cnt_end) cnt <= 4'h0;
    else if(cnt_inc) cnt <= cnt + 4'h1;
end
endmodule
```

3、边沿检测与按键启停

同一个按键开关既能启动，又能暂停，不仅要判断按键是否按下，还要记录按键按下代表的状态。不能将button信号作为上升边触发的敏感信号。下面这种是很不规范的写法，敏感信号列表中的边沿触发除了时钟和复位信号，不能出现其他信号。

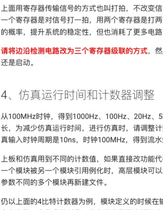
```
always @(posedge button)
begin

end
```

那要怎么实现信号的边沿检测？边沿检测将输入信号的高低电平变化转化为脉冲信号，有上升沿检测、下降沿检测以及双边沿检测。根据是否与时钟同步，分为同步检测与异步检测，实验仅考虑同步检测。对于同步检测，通过比较输入信号在相邻两个时钟周期内电平，从而判断是否发生了电平变化。

如下图所示，位置1的时候，输入信号为低电平，在下一个时钟上升沿，输入信号为高电平，输入信号有上升沿变化。同样分别对比位置3和4的信号，可以检测到信号的下降沿。如果信号变化太快，在一个时钟周期内就完成变化，确实会检测不到。实验当中，时钟频率是100MHz远高于按键按下的频率，不会丢失检测。

怎么得到信号在相邻两个时钟周期的值？寄存器的存储功能可以帮助实现，且有多种方式。

- 方式一：只用一个寄存器，寄存器的输出是上一个时钟周期的值，恰好延迟一个时钟周期，再与signal当前的值做对比判断是上升沿还是下降沿，最终的输出直接与输入有关，可以检测到不足一个时钟周期的跳变，输入信号有跳变也能立即检测到，但也容易带来毛刺不稳定。



- 方式二：两个寄存器级联，第一个寄存器锁存住某个时钟上升沿到来时的输入电平，第二个寄存器锁存住再往前一个时钟沿到来时的输入电平。这两个寄存器的输出信号也是隔了一个时钟周期。最终输出是两个寄存器输出信号的逻辑操作，输出更稳定，但响应没有方式一快。信号signal有边沿跳变1个时钟周期后才能检测到。

- 对应的verilog示例代码如下

```
module edge_detect(
    input             clk,
    input             rst,
    input             signal,
    output            pos_edge
);

reg sig_r0, sig_r1;

always @(posedge clk)
begin
    if(rst) sig_r0 <= 1'b0;
    else    sig_r0 <= signal;
end

always @(posedge clk)
begin
    if(rst) sig_r1 <= 1'b0;
    else    sig_r1 <= sig_r0;
end

assign pos_edge = ~sig_r1 & sig_r0;
```

上面用寄存器传输信号的方式也叫打拍，不改变信号的变化过程，仅对信号延迟一个时钟周期，用一个寄存器是对信号打一拍，用两个寄存器是打两拍，级联的寄存器越多，越可以减少亚稳态发生的概率，提升系统的稳定性，但也消耗了更多电路资源。

请将边沿检测电路改为三个寄存器级联的方式，然后根据边沿检测的次数或状态判断当前是该停止还是启动。

4、仿真运行时间和计数器调整

从100MHz时钟，得到1000Hz、100Hz、20Hz、5Hz的流水灯频率，分频系数太大，仿真时间较长，为减少仿真运行时间，进行仿真时，请调整计数器的判断条件，将对应的计数上限值缩小。仿真输入时钟周期是10ns，时钟100MHz，得到流水灯频率是1000KHz、100KHz、20KHz、5KHz。

上板和仿真用到不同的计数量，如果直接改功能代码，改来改去容易出错，可用参数例化解决。当一个模块被另一个模块引用例化时，高层模块可以对低层模块的参数值进行改写，不用单独为只有参数不同的多个模块再新建文件。

仍以上面的4比特计数器为例，模块定义的时候在输入输出信号前，使用 `#()` 和 `parameter` 关键词定义参数，参数个数任意，定义的参数可在模块内直接使用。

```
module counter #(
    parameter CNT_MAX = 4'h9,
    parameter WIDTH   = 4
)
(
    input wire clk,
    input wire rst,
    input wire button,
    output reg [WIDTH - 1:0] cnt
);

reg   cnt_inc;
wire  cnt_end = cnt_inc & (cnt==CNT_MAX);

always @ (posedge clk or posedge rst) begin
    if(rst)          cnt_inc <= 1'b0;
    else if(button)  cnt_inc <= 1'b1;
    else if(cnt_end) cnt_inc <= 1'b0;
end

always @ (posedge clk or posedge rst) begin
    if(rst)          cnt <= 4'h0;
    else if(cnt_end) cnt <= 4'h0;
    else if(cnt_inc) cnt <= cnt + 4'h1;
end
endmodule
```

例化的时候，将所需要的参数传过去，这样在下板和仿真需要不同的频率，可以不用修改代码而只修改参数。

```
counter #(10000, 24) u_counter(
    .clk(clk),
    .reset(reset),
    .button(1),
    .cnt(cnt)
);
```

默认的仿真时间不够，输出的led信号没有看不到变化，需要延长仿真时间，具体操作请参考：仿真调试常用功能：3.延长仿真时间。

实验要求流水灯的间隔严格按照精度要求，在仿真中要进行信号周期测量。以分频器为例，用仿真视图上方的时钟沿工具（transition）选中输出时钟clk_out的两个相邻的上升沿，注意先用鼠标选中clk_out信号，再用 工具进行标示，请注意时间的小数位。