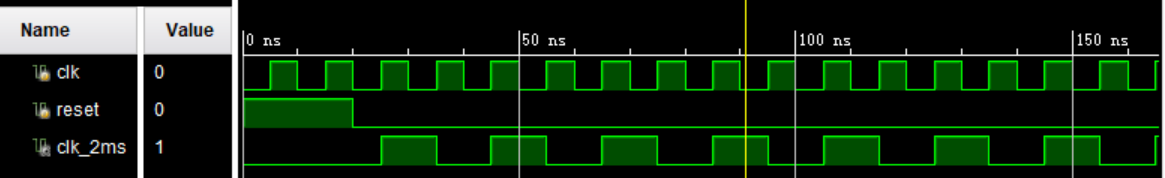
**实验报告 分频器**

一、二分频

二分频的分频器只需要在时钟上升沿到来时翻转分频后输出。

仿真结果如下：



代码如下：

module clk\_div(

input clk,

input reset,

output reg clk\_d

);

reg [19:0]count;

parameter maxcount=2;

always@(posedge clk or posedge reset)

begin

if(reset) begin clk\_d<=0;count<=0;end

else clk\_d<=~clk\_d;

end

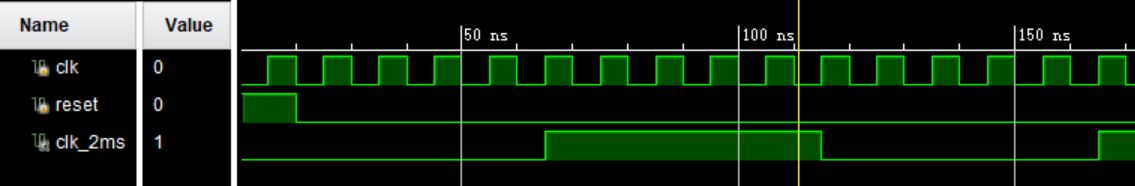
endmodule

二、十分频

分频器的意思就是将输入的时钟频率减小到原来的几分之一（除以几），，相应的，周期要扩大几倍（乘几），那么翻转分频后输出的间隔就应该是这个倍数的一半（十分频就应该是每5个时钟周期翻转分频后输出）。

这里计数器最大值maxcount=分频数N/2-1，其中分频数N为偶数。

仿真结果如下：



代码如下：

module clk\_div(

input clk,

input reset,

output reg clk\_d

);

reg [19:0]count;

parameter maxcount=4;

always@(posedge clk or posedge reset)

begin

if(reset) begin clk\_d<=0;count<=0;end

else if(count<maxcount) count<=count+1;

else if(count==maxcount)

begin count<=0;clk\_d<=~clk\_d;end

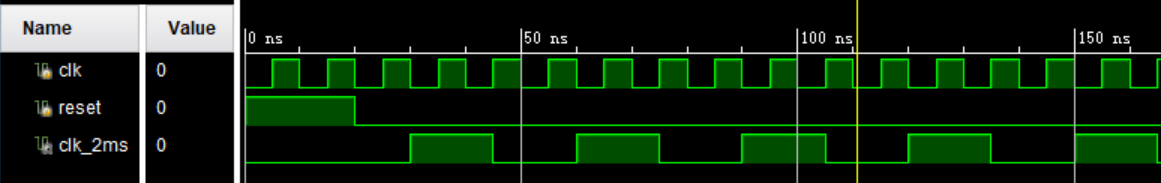
end

endmodule

\*三、三分频

上面的代码并不能实现像三分频这样的奇数分频（上式要求分频数N为偶数）。奇数分频不可能只用上升/下降沿实现计数翻转，必须将两者同时结合起来计数。也就是在之前代码的基础上将负边沿触发考虑进来。此时计数器最大值maxcount=分频数N-1，其中分频数N为大于1的正整数。

仿真结果如下：



代码如下：

module clk\_div(

input clk,

input reset,

output reg clk\_d

);

reg [19:0]count;

parameter maxcount=2;

always@(posedge clk or posedge reset or negedge clk)

begin

if(reset) begin clk\_d<=0;count<=0;end

else if(count<maxcount) count<=count+1;

else if(count==maxcount)

begin count<=0;clk\_d<=~clk\_d;end

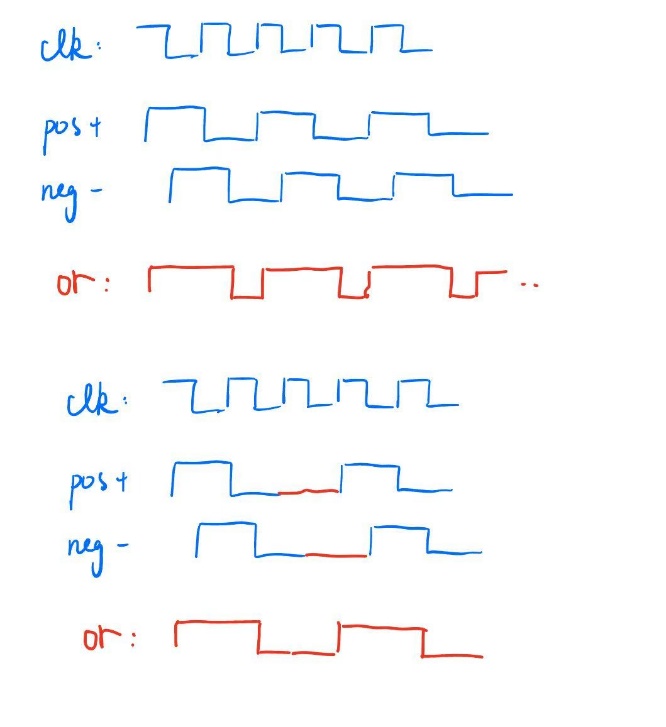
end

endmodule

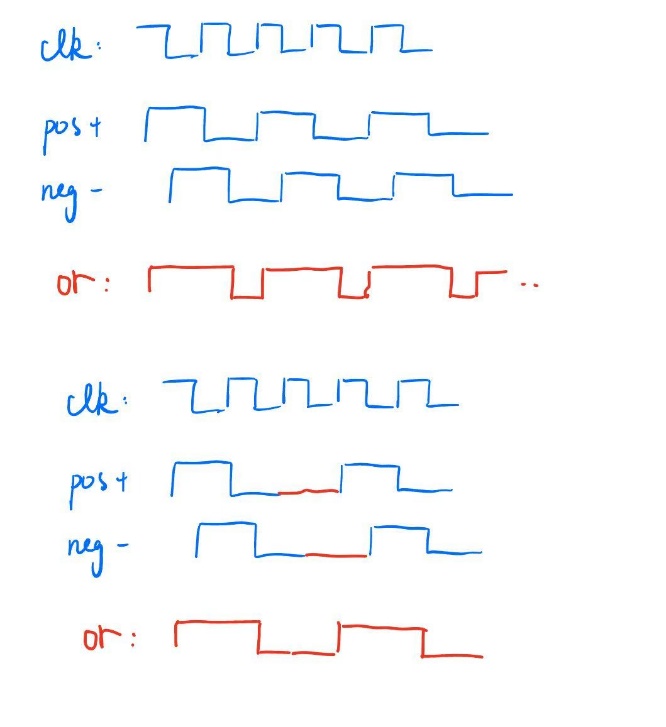
但是，这样写仿真可以跑，但是综合会报错。原因是在always块的敏感列表always@(posedge clk or posedge reset or negedge clk)中，同时指定了时钟clk的上升沿和下降沿，这会使综合工具无法确定唯一的时钟触发沿。实际硬件中，大多数时序逻辑元件（如触发器）基于单一的时钟沿触发工作，这种模糊的时钟定义不符合综合规则，所以综合不通过。

既然如此，我们分别设置两个输出，分别用正边沿和负边沿触发，然后将两个输出相加。想想是可行的，因为正负边沿触发刚好相差半个时钟周期，再加上一个偶数分频便可实现奇数分频。

先用二分频试一下：



将正/负边沿触发的二分频信号相或，结果的高电平符合三分频的要求，但低电平时间少了一个时钟周期，我们尝试加回来：



这就是我们想要的三分频信号了。

代码如下：

module clk\_div(

input clk0,

input reset,

output clk\_out

);

reg [30:0]count;

reg [30:0]count1;

parameter maxcount=2;

always@(posedge clk0 or posedge reset)

begin

if(reset) count<=0;

else if(count<maxcount) count<=count+1;

else if(count==maxcount)

count<=0;

end

reg clk1;

always @(\*) begin

case(count)

2'b00: clk1 = 1;

2'b01: clk1 = 0;

2'b10: clk1 = 0;

default:clk1 = 0;

endcase

end

always@(negedge clk0 or posedge reset)

begin

if(reset) count1<=0;

else if(count1<maxcount) count1<=count1+1;

else if(count1==maxcount)

count1<=0;

end

reg clk2;

always @(\*) begin

case(count1)

2'b00: clk2 = 1;

2'b01: clk2 = 0;

2'b10: clk2 = 0;

default:clk2 = 0;

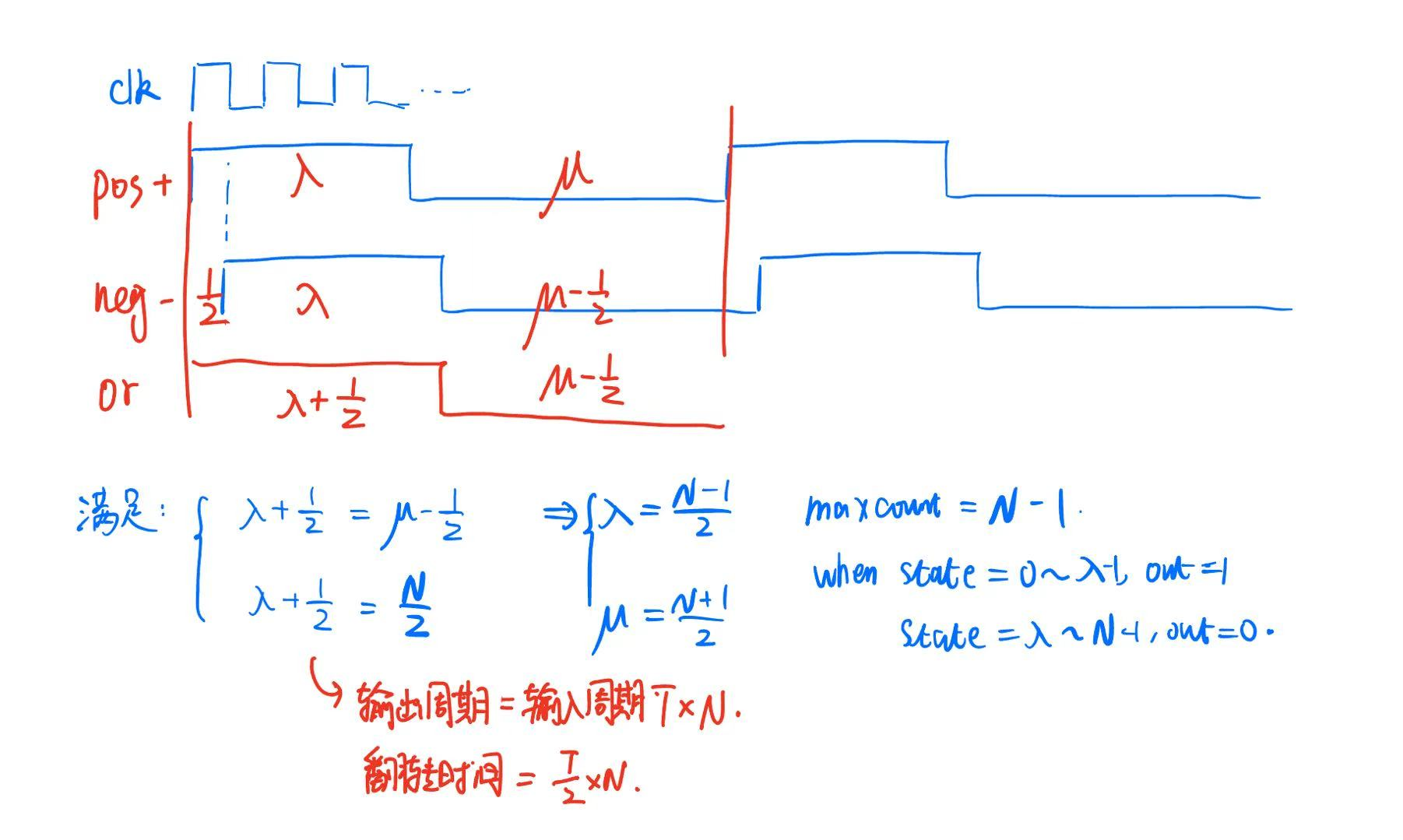
endcase

end

assign clk\_out=clk1|clk2 ;

endmodule

推广：实现奇数分频的各参数计算：



如：五分频

