实验一 组合逻辑设计

**一、实验目的**

1 掌握Verilog语言和Vivado、Logisim开发平台的使用；

2 掌握基础组合逻辑电路的设计和测试方法。

**二、实验内容（用Logisim或Vivado实现）**

1 基础门电路（多输入门电路、复用器等）的设计和测试；

2 基础功能模块（编码器、译码器等）的设计与测试。

**三、实验要求**

1 掌握Vivado与Logisim开发工具的使用，掌握以上电路的设计和测试方法；

2 记录设计和调试过程（Verilog代码/电路图/表达式/真值表，Vivado仿真结果，Logisim验证结果等）；

3 分析Vivado仿真波形/Logism验证结果，注重输入输出之间的对应关系。

**四、实验过程及分析**

**1.多输入门电路**

代码：

设计文件：module first(

input a,b,c,d,e,

output y

);

assign y=~(a&~b&c&(d|e));

endmodule

仿真文件：module sim;

reg a,b,c,d,e;

wire y;

initial begin

a=0;b=0;c=0;d=0;e=0;

#5 {a,b,c,d,e}=5'b10000;

fork

forever #5 b=~b;

repeat(5) #10 c=~c;

forever #5 d=~d;e=~e;

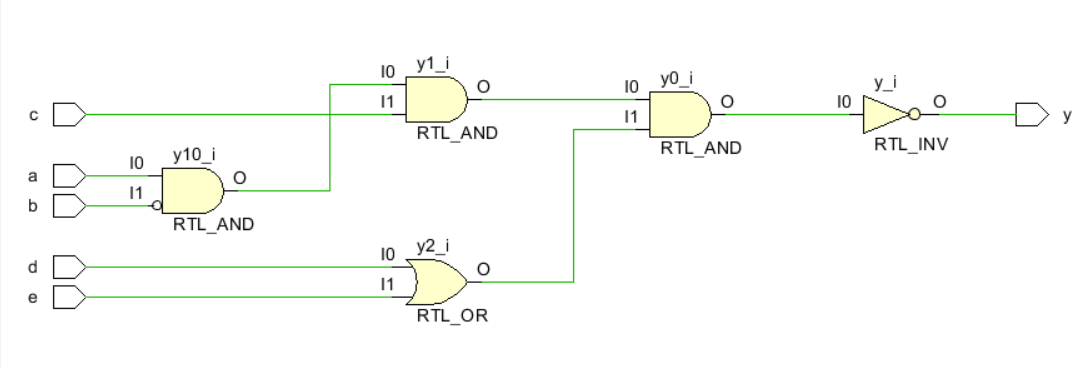
join

end

first s(a,b,c,d,e,y);

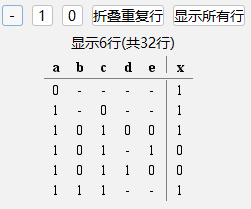
endmodule

电路图：

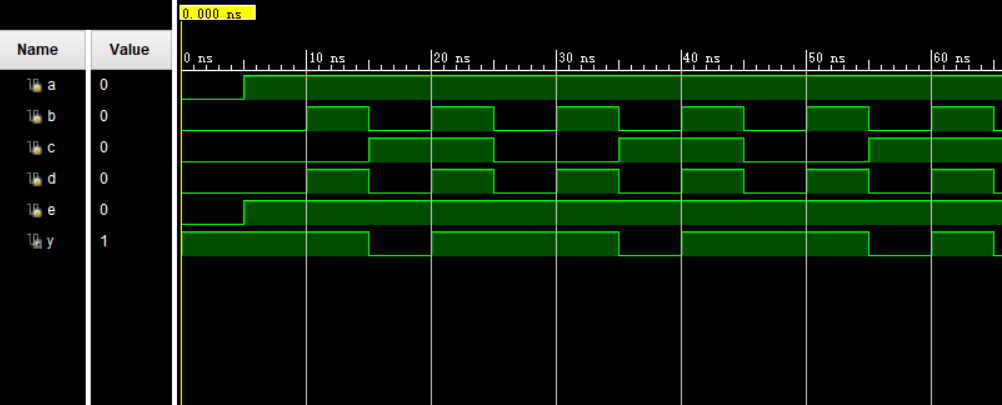


表达式：y=~(a·~b·c·(d+e))

真值表：



仿真波形：



分析：当a为0或者c为0时，输出y直接为0。当b为1时，输出y也直接为0。当以上三个条件都不成立（即a为1，b为0，c为1）时,输出y的值才取决于d和e，具体关系为y等于d和e的或非。

**2.二路复用器**

代码：

设计文件：module twowaymultiplexer (

input d0,d1,s,

output y );

assign y=(d0&~s)|(d1&s);

endmodule

仿真文件：module sim;

reg d0,d1,s;

wire y;

initial begin

{d0,d1,s}=3'b000;

#s {d0,d1,s}=3'b100;

fork

forever #5 d0=~d0;

forever #10 d1=~d1;

forever #15 s=~s;

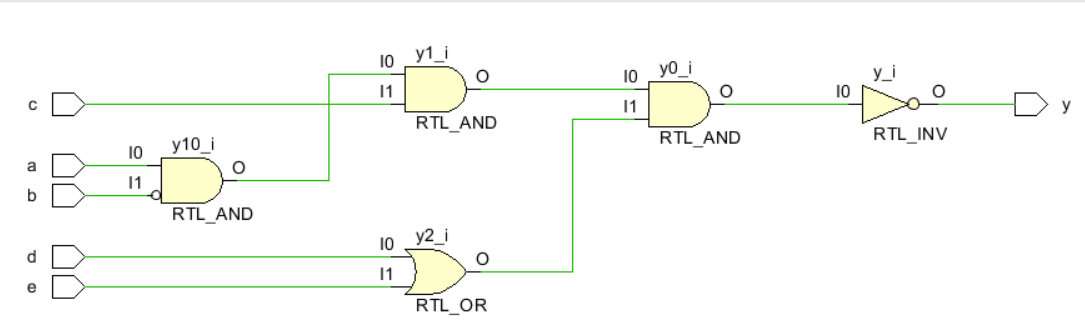
join

end

twowaymultiplexer m(d0,d1,s,y);

endmodule

电路图：

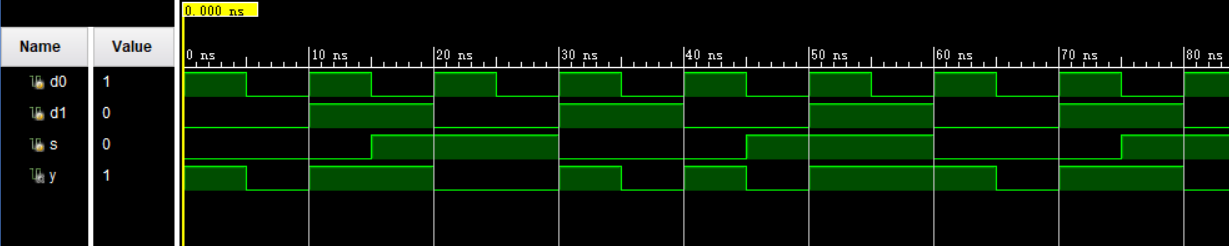


表达式：Y=D1·S+D0·~S

真值表：



仿真图形：



分析：当s的值为1时，输出y的值就等于输入d1的值；当s的值为0时，输出y的值就等于输入d0的值

**3.多路复用器**

代码：

设计文件：module Multiplexer(

input [3:0]D, [1:0]S,

output Y

);

wire Y1,Y2;

twowaymultiplexer f1(D[0],D[1],S[0],Y1),

f2(D[2],D[3],S[0],Y2),

f3(Y1,Y2,S[1],Y);

endmodule

仿真文件：module sim;

reg [3:0]D;

reg [1:0]S;

wire Y;

initial begin

D=4'b0000;S=2'b00;

end

always fork

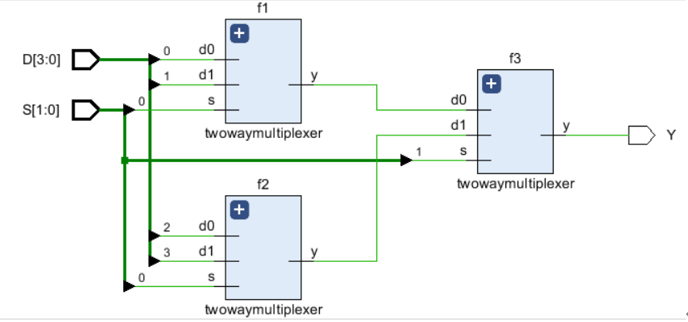
#1 D=D+1'b1;

#2 S=S+1'b1;

join

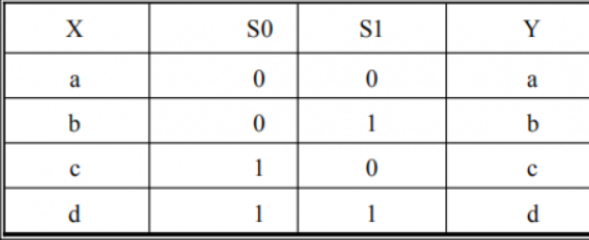
Multiplexer m(D,S,Y);

endmodule

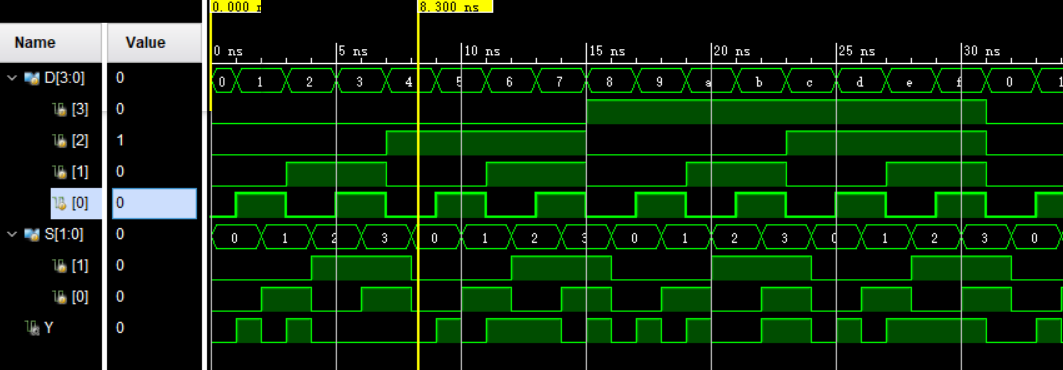
电路图：

表达式:Y=D[0]·(~S[0]·~S[1])+D[1]·(S[0]·~S[1])+D[2]·(~S[0]·S[1])+D[3]·(S[0]·S[1])

真值表：



仿真图形：



分析：输出Y与输入数组D之间的关系就是多选一，具体选择输入数组中的哪一个取决于数组S，S[0]构成低位和S[1]构成高位所组成的二位二进制数转化成十进制数N后，在输入数组D中选择对应十进制数N的输入值将其作为输出Y。比如在仿真波形中看到一开始S[0]和S[1]都为0，对应的N为0，所以这一段时期之内Y的值就等于D[N]也就是D[0]。

**4.编码器**

代码：

设计文件：module encoder(

input wire [7:0]I,

output wire [2:0]Y

);

assign Y = {I[4]|I[5]|I[6]|I[7], I[2]|I[3] |I[6] |I[7], I[1]|I[3]|I[5]|I[7]} ;

仿真文件：module sim;

reg [7:0]I;

wire [2:0]Y;

integer i;

initial begin

I=8'b00000001;i=0;

end

always begin

I=8'b00000000;i=(i+1)%8;

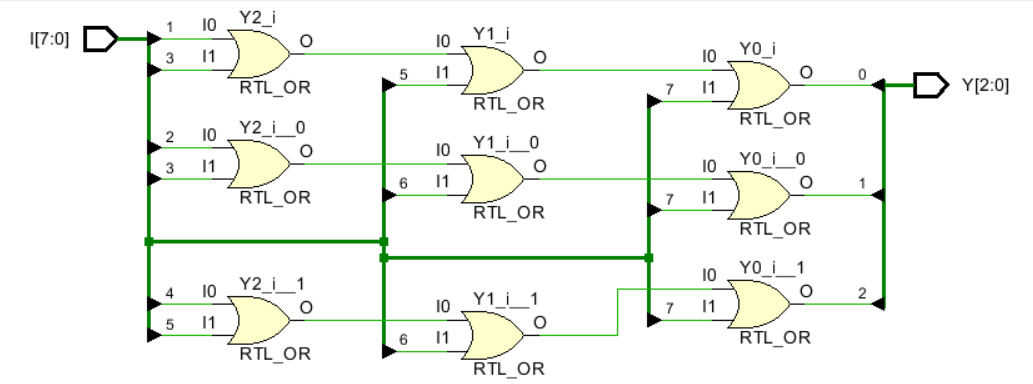
#10 I[i]<=1;

end

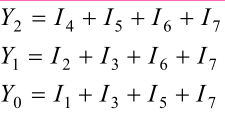
encoder m(I,Y);

endmodule

电路图：



表达式：

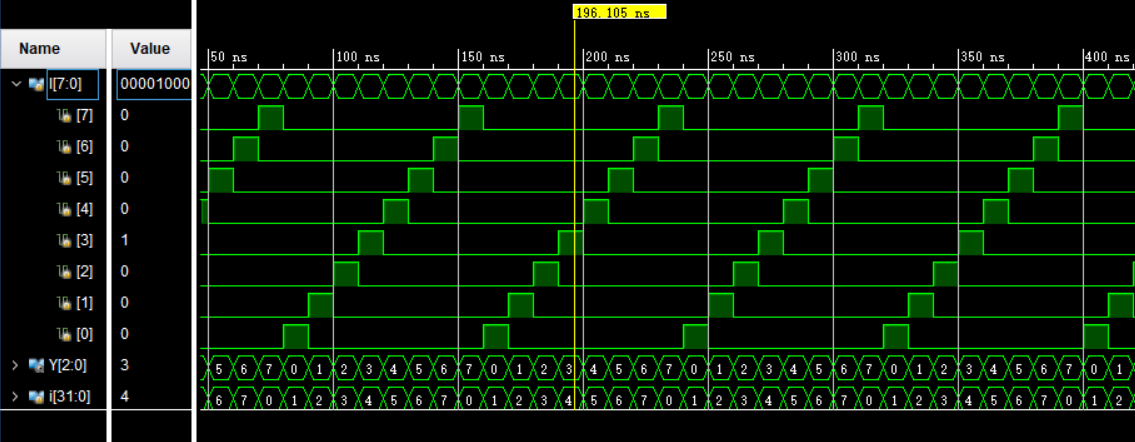


**逻辑表达式**

真值表：



仿真波形：



分析：输入数组I中仅有一个下标对应的数组成员值为1，这个下标值转化为三位二进制数以此存进输出数组Y中，就得到了输出数组Y的值。比如当I[3]为1时，3对应的三位二进制数为011，将011以此存进输出数组Y中，最终结果是Y[2]为0，Y[1]为1，Y[0]为1。

**5. 8-3优先编码器**

代码：

设计文件：module encoder(

input [7:0]I,

output reg [2:0]Y

);

integer i,j;

always@(I) begin

begin:one

for(i=7;i>0;i=i-1)

if(I[i]==1)

disable one;

end

Y=i;

end

endmodule

仿真文件：module sim;

reg [7:0]I;

wire [2:0]Y;

integer i;

encoder m(I,Y);

initial begin

I=8'b00000001;i=0;

end

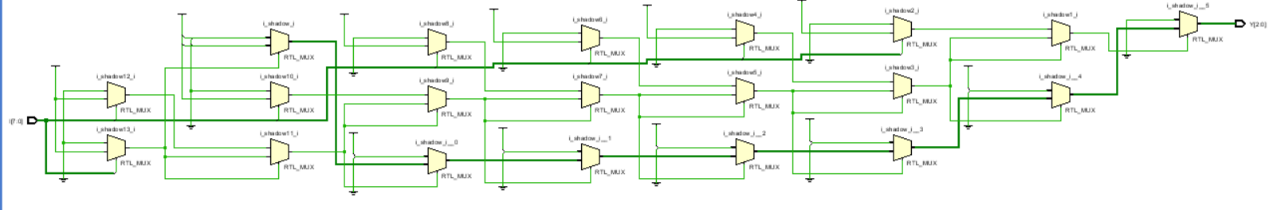
always begin

#1 I=I+1;

end

endmodule

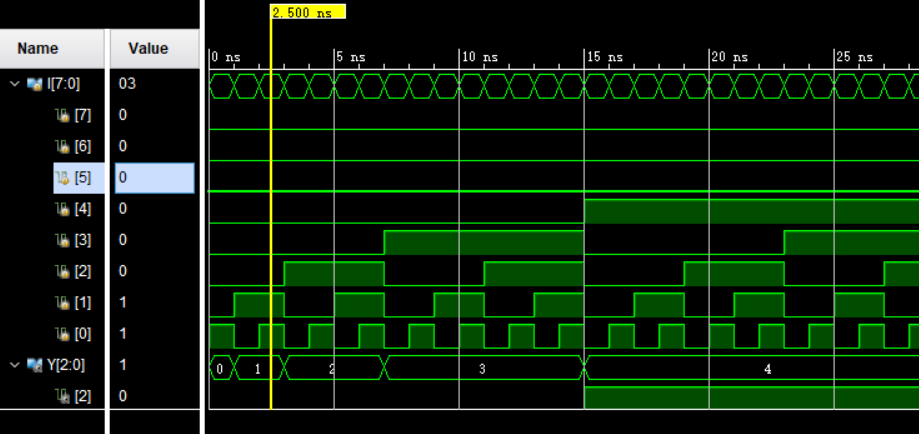
电路图：



真值表：



仿真图形：



分析：输出数组Y构成的二进制数就是输入数组I中最高位1的下标值，因此，我们利用变量i设计for循环从输入数组最高位开始逐渐往低位找1，i充当下标值初始值为7，每经过一个循环i的值就减1，当找到数组中的1时立刻disable退出循环，再让Y等于I即可。比如输入为0x08,从输入最高位往低位找数组中的1，当i到4时找到目标，退出for循环，让Y等于4即可。

**6.译码器**

代码：

设计文件：module decoder(

input [1:0]A,

output reg [3:0]Y

);

always@(A) begin

Y=4'b0000;

Y[A]=1;

end

endmodule

仿真文件：module sim;

reg [1:0]I;

wire [3:0]Y;

decoder m(I,Y);

initial begin

I=0; end

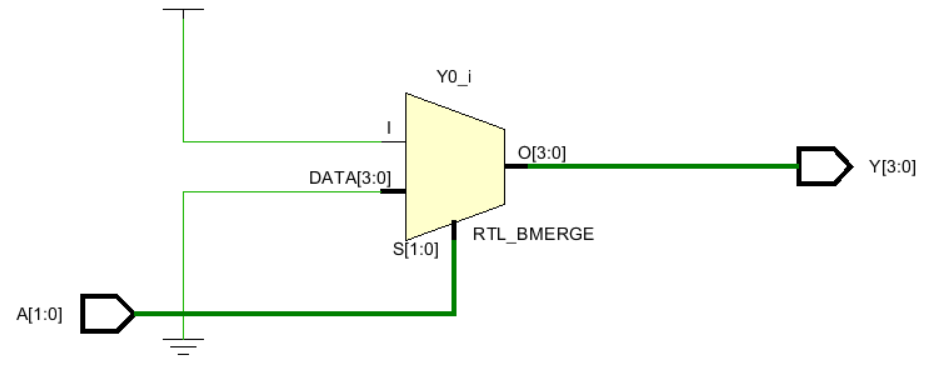
always begin

#1 I=I+1;

end

endmodule

电路图：



表达式：

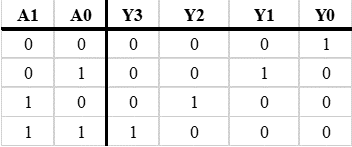
Y0=~A1·~A0;

Y1=~A1·A0;

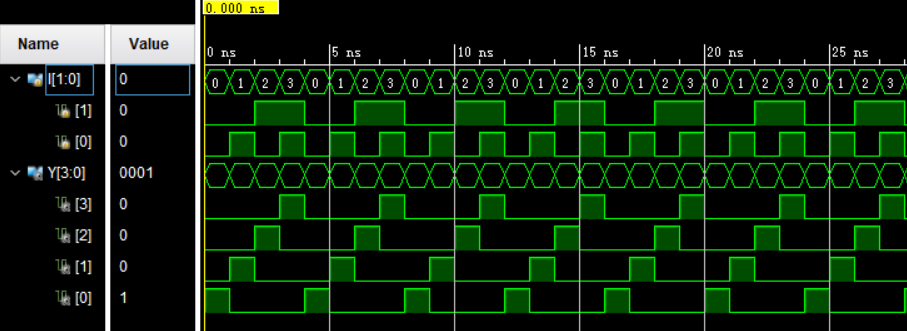
Y2=A1·~A0;

Y3=A1·A0;

真值表：



仿真图形：



分析：输入数组A的每一个成员值以此构成二进制数中的一位数，比如A[0]构成二进制数中的最低位，A[1]构成二进制数中的次第位，最终得到的二进制数N作为输出数组Y中寻址的下标，令Y[N]=1，Y中其余成员的值为0即可得到输出数组Y。比如当A的值为3的时候，Y[3]为1，Y中其余成员的值都为0，最终Y的值为0X08.

**7.3-8译码器**

代码：

设计文件：module decoder(

input [2:0]A,

output [7:0]Y

);

assign Y[0]=A[0]|A[1]|A[2];

assign Y[1]=~A[0]|A[1]|A[2];

assign Y[2]=A[0]|~A[1]|A[2];

assign Y[3]=~A[0]|~A[1]|A[2];

assign Y[4]=A[0]|A[1]|~A[2];

assign Y[5]=~A[0]|A[1]|~A[2];

assign Y[6]=A[0]|~A[1]|~A[2];

assign Y[7]=~A[0]|~A[1]|~A[2];

endmodule

仿真文件：module sim;

reg [2:0]I;

wire [7:0]Y;

decoder m(I,Y);

initial begin

I=0;

end

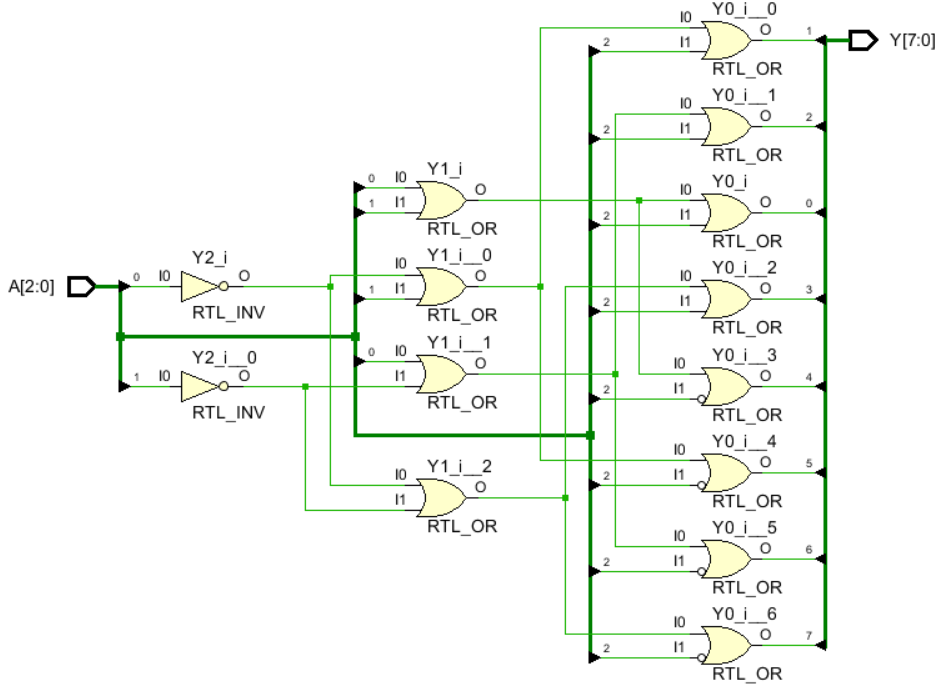
always begin

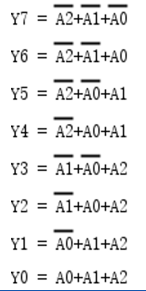
#1 I=I+1;

end

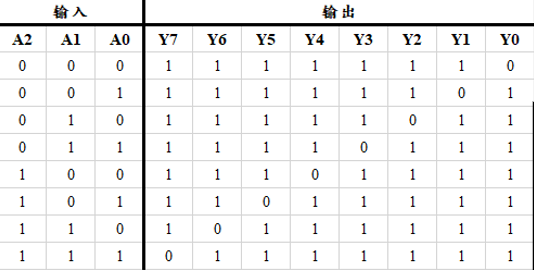
endmodule

电路图：

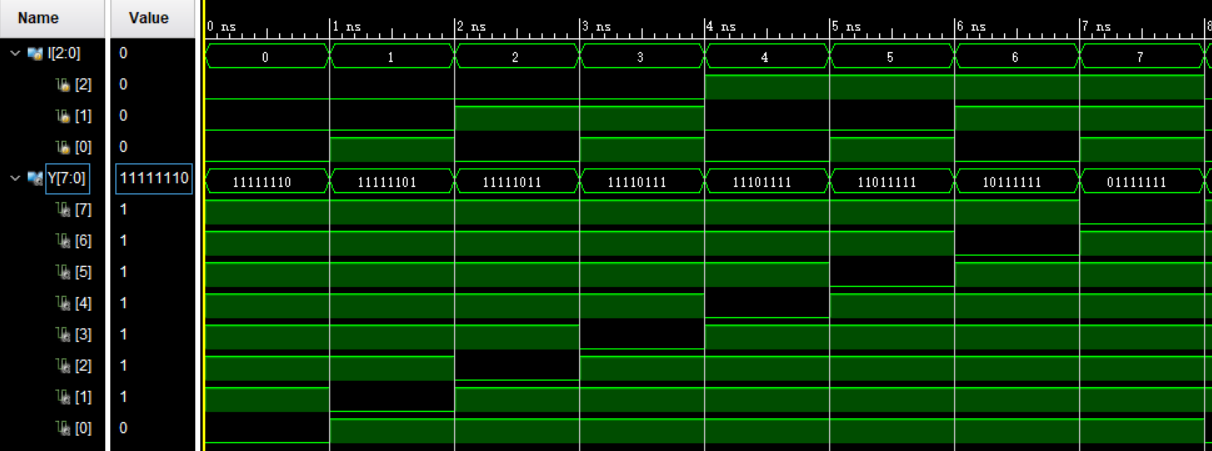


 表达式：

真值表：



仿真图形：

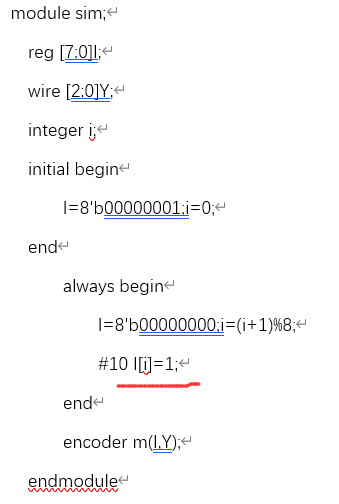


分析：输入数组A的每一个成员值以此构成二进制数中的一位数，比如A[0]构成二进制数中的最低位，A[1]构成二进制数中的次第位，最终得到的二进制数N作为输出数组Y中寻址的下标，令Y[N]=0，Y中其余成员的值为1即可得到输出数组Y。比如当A的值为3的时候，Y[3]为0，Y中其余成员的值都为1，最终Y的值为0xf7。

**五、调试和心得体会**

通过完成此次实验，我重新回顾了如何进行基础门电路和基础功能模块的设计，进一步熟悉了vavido的环境并掌握了如何运用verilog语言来开发硬件功能，对于仿真波形的理解也更加深刻。

比如在编写编码器的仿真文件时有一处赋值语句我写成了=，仿真后发现波形不对，一直没有输出，仔细打断点进行调试后发现输入没有问题，在块语句内部发现I[i]的值如预期一样被修改成预期的值，但是一旦再次执行这个块语句的时候我发现I[i]的值又变回了原样，在请教老师后发现这个地方应该使用<=，最后得到的仿真图形如预期所示



又比如在编写译码器的设计文件时，我直接对Y进行赋值操作，但是经过vavido报错后发现直接进行赋值操作的只能是reg类型的变量，于是我对Y的类型进行更改之后再进行仿真，得到的仿真结果是Y没有任何起伏变化，后来我进行调试也没有发现任何变化，于是我去查阅了上学期数电实验的资料，发现随着A的变化Y也需要变化，这种情况下需要添加always@(A),最终的预期仿真结果符合预期

