

**计算机与信息 学院实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验课程： | 计算机组成原理实验 | | | | | |
| 实验编号： | Project 4 | | | | | |
| 实验名称： | Verilog HDL数据流建模方法 | | | | | |
| 实验人员： | 学号 | 18111303044 | |  | |  |
| 姓名 | 邵一波 | |  | |  |
| 班级 | 18计算机类1班 | |  | |  |
| 实验日期： | 2019-09-30 | | | | | |
| 实验室： | 学苑南楼1幢305 | | | | | |
|  |  | | | | | |
| 实验评价： |  | | | | | |
| 实验成绩： | |  | 评价日期： |  | |
|  | 指导教师： | |  | | | |

# Verilog HDL常用建模方法

# 一、实验目的

## 1.掌握Verilog HDL的数据流建模方法。

## 2.掌握Verilog HDL的数据流建模方法的验证技术。

# 二、实验工具

## 1.Xilinx Vivado 2014.2软件。

## 2.Windows系统PC机。

# 三、实验要求

## 1. 通过“双控开关控制逻辑电路”实例演示学习Verilog HDL数据流建模方法和验证技术。

## 2.利用Verilog HDL的数据流建模方法对“4位数值比较器建模与仿真”建模和验证。

## 3.利用Verilog HDL的数据流建模方法对“3-8译码器”建模和验证。

## 4.利用Verilog HDL的数据流建模方法对“4-1多路选择器”建模和验证。

# 四、实验内容

## 1. “双控开关控制逻辑电路”数据流建模与验证

设计一个楼上、楼下开关的控制逻辑电路来控制楼梯上的路灯，使之在上楼前，用楼下开关打开电灯，上楼后，用楼上开关关灭电灯；或者在下楼前，用楼上开关打开电灯，下楼后，用楼下开关关灭电灯。

### (1)列真值表

设楼上开关为A，楼下开关为B，灯泡为Y。并设A、B闭合时为1，断开时为0；灯亮时Y为1，灯灭时Y为0。根据逻辑要求列出真值表。

表1 “双控开关控制逻辑电路”真值表

|  |  |
| --- | --- |
| A B | *Y* |
| 0 0  0 1  1 0  1 1 | 0  1  1  0 |

### (2)列输出方程



### (3)画电路图

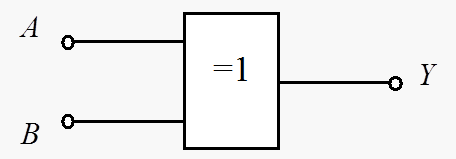
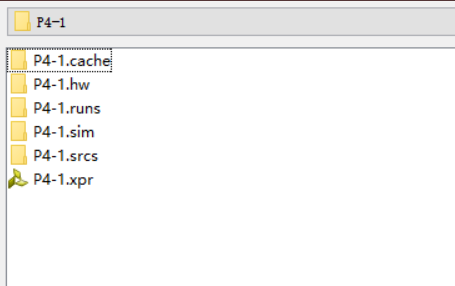


图1 逻辑电路图

### (4) 创建工程P4-1

****

### (5)数据流建模

module doubleswitch\_dataflow(a,b,y);

input wire a,b;

output wire y;

assign y=a^b;

endmodule

### (6) 仿真

`timescale 1ns/100ps

module test\_doubleswitch();

reg clk\_1Hz,a,b;

wire y;

doubleswitch\_dataflow u0(a,b,y);

always #100 clk\_1Hz = ~clk\_1Hz ;

initial

begin

clk\_1Hz=0;

a=0;b=0;

#200;

a=0;b=1;

#200;

a=1;b=0;

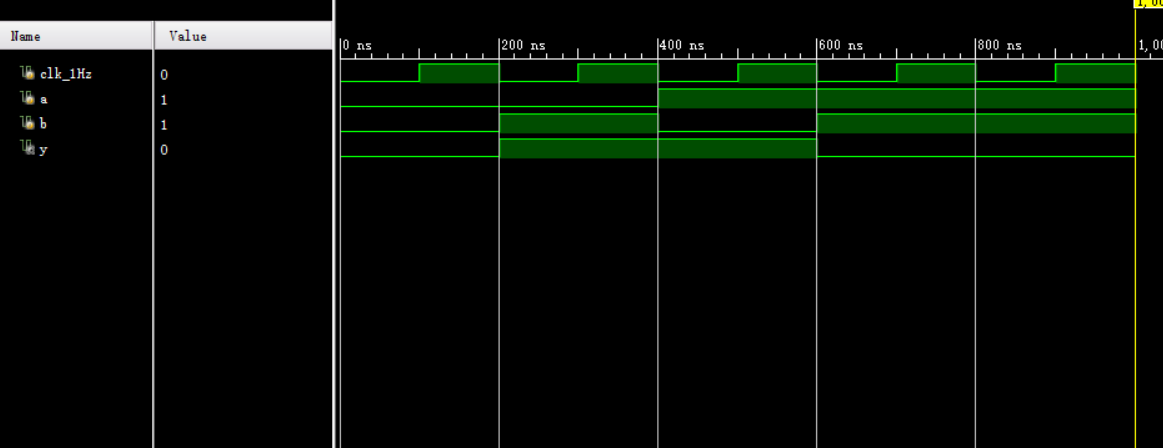
#200;

a=1;b=1;

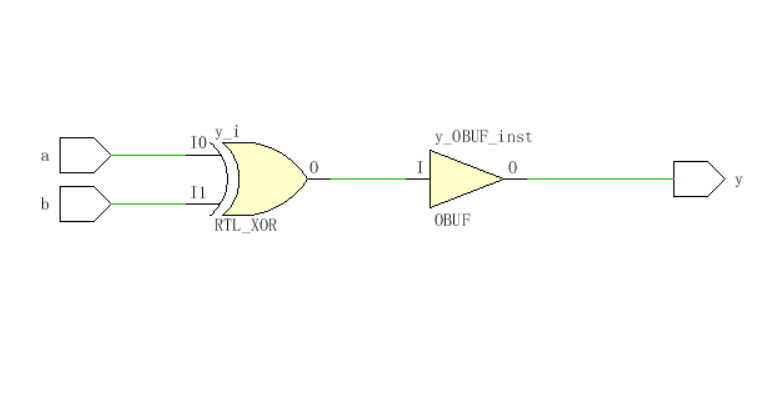
end

endmodule

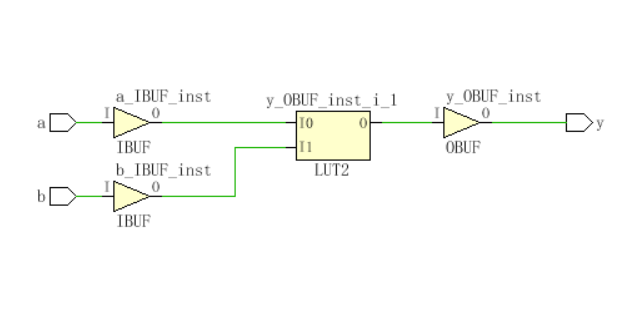
### (7)仿真，查看波形

****

### (8)进行RTL分析，查看电路原理图

****

### (9)进行综合，查看综合后电路原理图

****

## 2. “4位数值比较器”建模与验证

1位数值比较器的功能:比较两个1位二进制数*Ai*和*Bi*的大小，比较结果有三种情况，即：*Ai*＞*Bi*、*Ai*＜*Bi*、 *Ai*＝*Bi*。

输入：Ai、Bi。

输出： *Ai*＞*Bi，* 令Li＝1； Ai ＝*Bi* 令Gi =1； Ai＜Bi令Mi＝1。

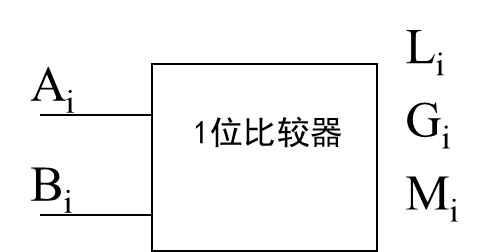
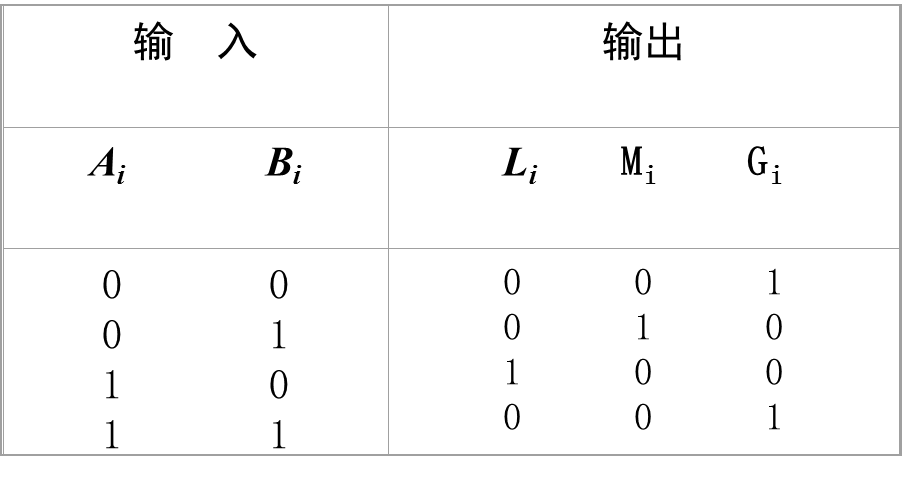


图2 1位数值比较器示意图

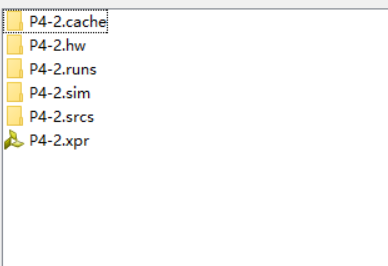
表2 1位数值比较器真值表



1位数值比较器逻辑表达式：



### (1)创建工程P4-2

****

### (2)1位数值比较器建模

【请完成任务】

module cmpi(ai,bi,li,mi,gi);

input wire ai,bi;

output wire li,mi,gi;

assign li=ai&(~bi);

assign gi=ai ~^ bi;

assign mi=(~ai) & bi;

endmodule

### (3)4位数值比较器建模

1位数值比较器只能对两个1位二进制数进行比较。而实用的比较器一般是多位的，而且考虑低位的比较结果。 下面以4位为例讨论这种数值比较器的结构及工作原理。

比较方法：从最高位开始比较，依次逐位进行。

输入：

* A=A3A2A1A0 ；
* B=B3B2B1B0 。

输出：

* A>B：L=1；
* A=B：G=1；
* A<B： M=1。

逻辑分析：

1)A3＜B3：A＜B，M3＝1，L＝G＝0。

2)A3＝B3，即G3＝1，A2＜B2：A＜B，M2＝1，L＝G＝0。

3)A3＝B3，A2=B2，即G3=G2＝1，A1＜B1： A＜B，M1＝1，L＝G＝0。

4) A3＝B3，A2=B2，A1=B1，即G3=G2＝G1=1，A0＜B0：A＜B，M0＝1，L＝G＝0。

5) A3＝B3，A2=B2，A1=B1， A0=B0，即G3=G2＝G1=G0=1：A=B,G=1。

6) M=G=0：L=1。

比照上述表达式，可写出：



设计出对应的模块：

module cmp(a,b,l,m,g);

parameter N=4;

input wire [N-1:0]a;

input wire [N-1:0]b;

output wire l,m,g;

wire [N-1:0]mi; //内部cmpi实例mi输出信号

wire [N-1:0]gi;//内部cmpi实例gi输出信号

cmpi cmp0(.ai(a[0]),.bi(b[0]),.mi(mi[0]),.gi(gi[0]));

cmpi cmp1(.ai(a[1]),.bi(b[1]),.mi(mi[1]),.gi(gi[1]));

cmpi cmp2(.ai(a[2]),.bi(b[2]),.mi(mi[2]),.gi(gi[2]));

cmpi cmp3(.ai(a[3]),.bi(b[3]),.mi(mi[3]),.gi(gi[3]));

assign m=mi[3]|(gi[3]&mi[2])|(gi[3]&gi[2]&mi[1])|( gi[3]&gi[2]&gi[1]&mi[0]);

assign g= gi[3]&gi[2]&gi[1]&gi[0];

assign l=!(m|g);

endmodule

### (4)编写Test Bench

`timescale 1ns/100ps

module test\_cmp();

reg clk\_1Hz;

reg [3:0]a;

reg [3:0]b;

wire l,m,g;

cmp u0(a,b,l,m,g);

always #100 clk\_1Hz = ~clk\_1Hz ;

initial

begin

clk\_1Hz=0;

a=4'b0000;b=4'b0000;

#200;

a=4'b0010;b=4'b1010;

#200;

a=4'b0100;b=4'b0101;

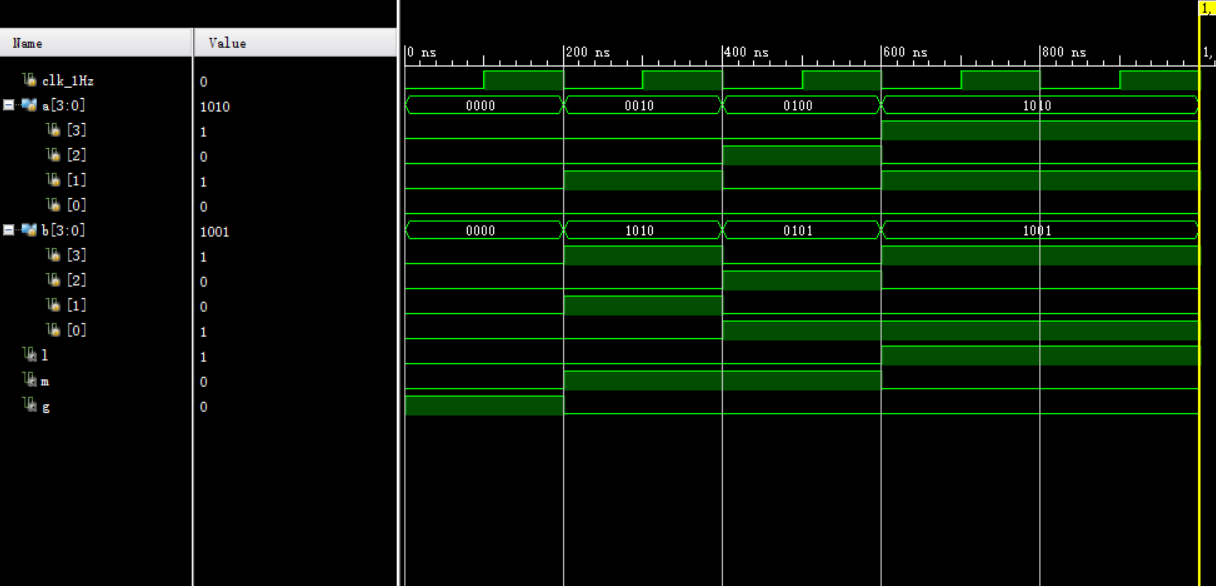
#200;

a=4'b1010;b=4'b1001;

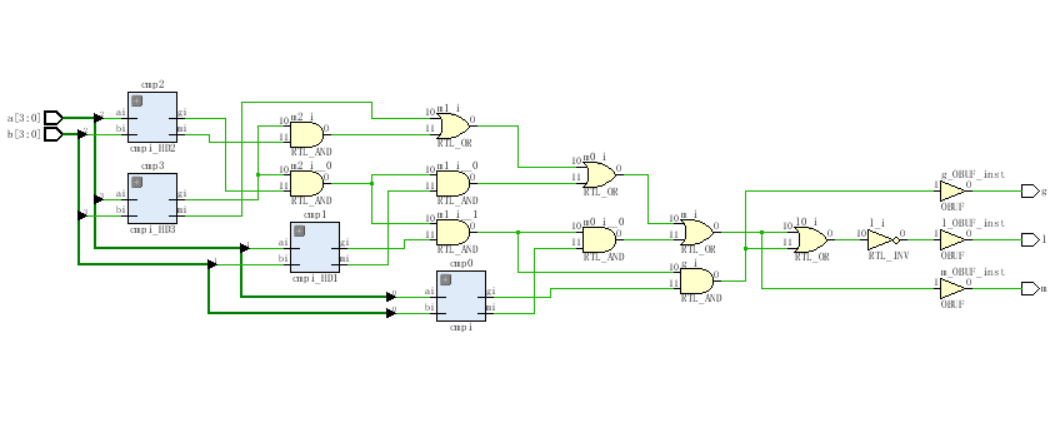
end

endmodule

### (5)仿真，查看波形

****

### (6)进行RTL分析，查看电路原理图

****

### (7)进行综合，查看综合后电路原理图

## 3. “3-8译码器”建模与验证

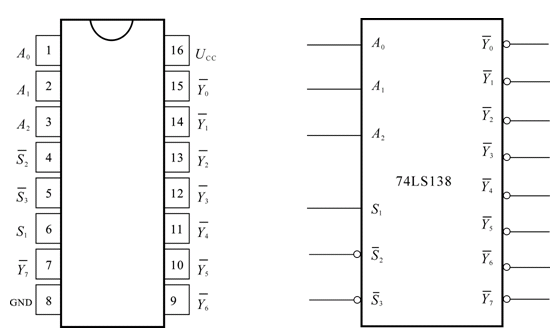
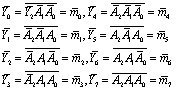


图3 74LS138管脚排列图

如果输入的是n位二进制代码，则译码器应该有2 n个输出端。所以2位二进制译码器有4个输出端，又可以称为2线－4线译码器；3位二进制译码器有8个输出端，可以称为3线－8线译码器；4位二进制译码器有16个输出端，可以称为4线－16线译码器等等。图4.13是集成3线－8线译码器74LS138的管脚排列图。图3是集成3线－8线译码器74LS138的逻辑图。在图4中A2、A1、A0是三个输入端，#Y7 ~ #Y0是八个输出端。S1、#S2、#S3是三个控制端。G1高电平有效， G2A、G2B低电平有效，即当S1=1、 #S2＝ #S3 ＝0时，控制门与门输出高电平，这个高电平把八个与非门打开，译码器正常工作，否则译码器不能正常译码，所有输出端都输出高电平。当译码器正常工作时（即与门输出1时）可以由逻辑图写出表达式，如式所示。



可以看出，译码器的输出#Y0 ~ #Y7正好是A2、A1、A0三个变量的 全部最小项，所以这种译码器又可以称为最小项译码器。只是74LS138的 输出是最小项的“非”。 根据表达式可以列出74LS138的功能表，如表3所示。

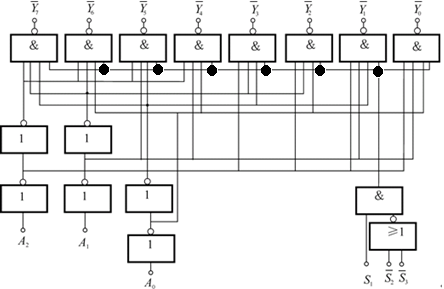
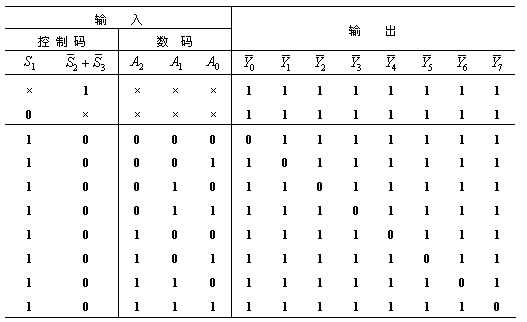


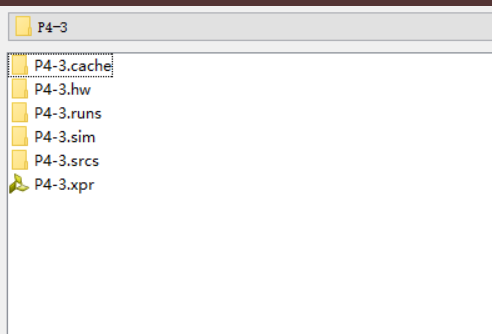
图4 3线－8线译码器74LS138的逻辑图

表3 3-8译码器真值表



请完成74LS138译码器的建模与验证。

### 创建工程P4-3



### “3-8译码器”建模

module decoder3\_8( S1 ,Y ,S2 ,A ,S3 );

input wire S1,S2,S3 ;

input wire [2:0] A ;

output wire [7:0] Y ;

assign Y =

({S1 & ~(~S2 | ~S3),A}==4'b1000)?8'b1111\_1110:

({S1 & ~(~S2 | ~S3),A}==4'b1001)?8'b1111\_1101:

({S1 & ~(~S2 | ~S3),A}==4'b1010)?8'b1111\_1011:

({S1 & ~(~S2 | ~S3),A}==4'b1011)?8'b1111\_0111:

({S1 & ~(~S2 | ~S3),A}==4'b1100)?8'b1110\_1111:

({S1 & ~(~S2 | ~S3),A}==4'b1101)?8'b1101\_1111:

({S1 & ~(~S2 | ~S3),A}==4'b1110)?8'b1011\_1111:

({S1 & ~(~S2 | ~S3),A}==4'b1111)?8'b0111\_1111:

8'b1111\_1111;

endmodule

### 编写Test Bench

`timescale 1ns/100ps

module test\_decoder3\_8();

reg clk\_1Hz;

reg S1,S2,S3 ;

reg [2:0] A ;

wire [7:0] Y ;

decoder3\_8 u0( S1 ,Y ,S2 ,A ,S3);

always #50 clk\_1Hz = ~clk\_1Hz ;

initial

begin

clk\_1Hz=0;

S1 = 0;S2 = 1;S3 = 1;A = 3'b000;

#100;

S1 = 1;S2 = 0;A = 3'b000;

#100;

S2 = 1;A = 3'b000;

#100;

A = 3'b001;

#100;

A = 3'b010;

#100;

A = 3'b011;

#100;

A = 3'b100;

#100;

A = 3'b101;

#100;

A = 3'b110;

#100;

A = 3'b111;

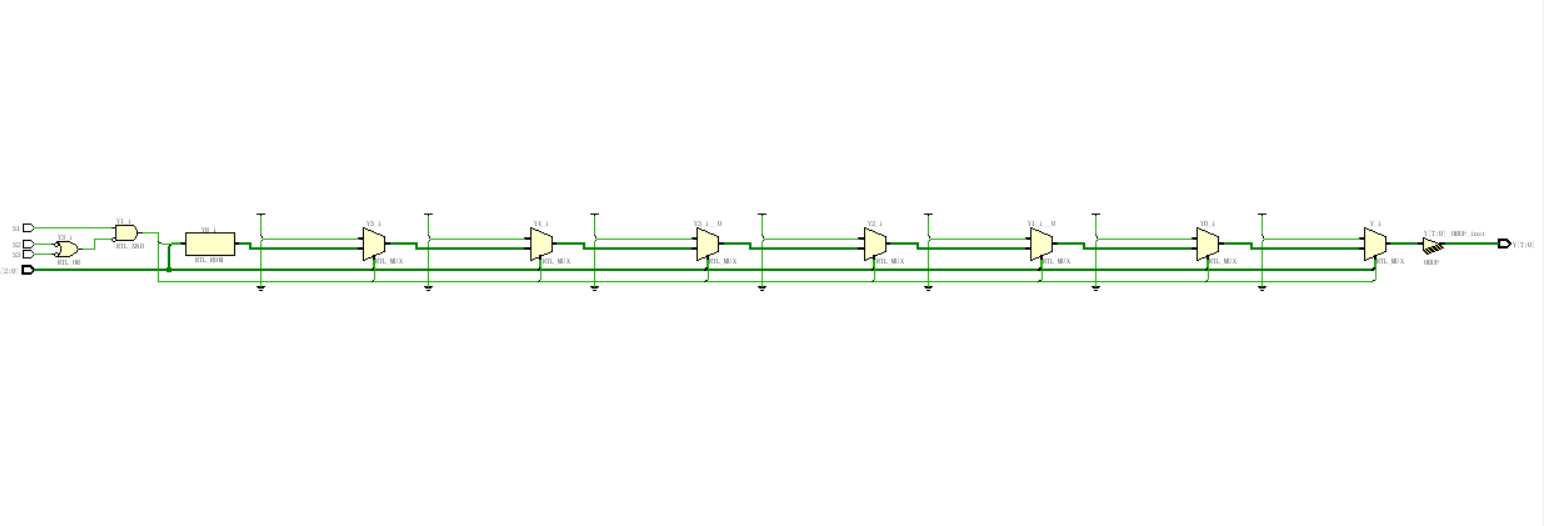
end

endmodule

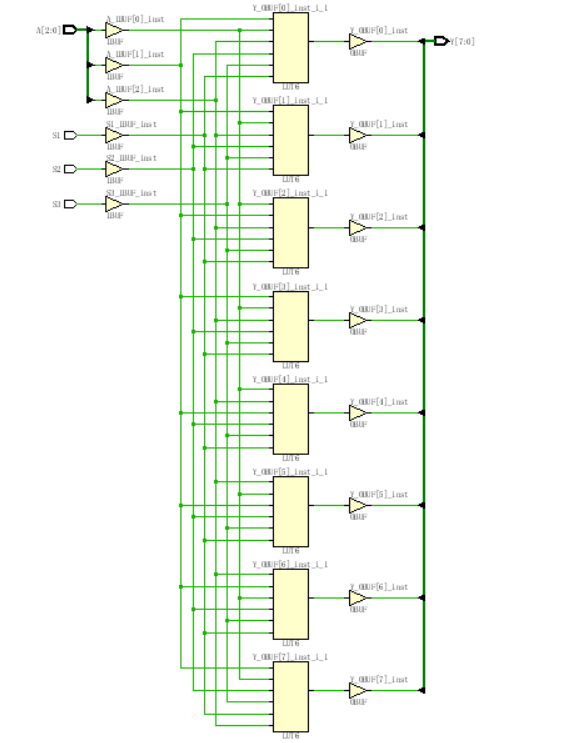
### 仿真，查看波形



### 进行RTL分析，查看电路原理图



### 进行综合，查看综合后电路原理图



## 4. “4-1多路选择器”建模与验证

多路选择器常见分类有4选1数据选择器（如图5所示）、8选1数据选择器（型号为74151、74LS151、74251、74LS152）、16选1数据选择器（可以用两片74151连接起来构成）等之分。多路选择器还包括总线的多路选择、模拟信号的多路选择等，相应的器件也有不同的特性和使用方法。

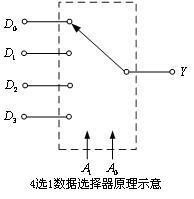
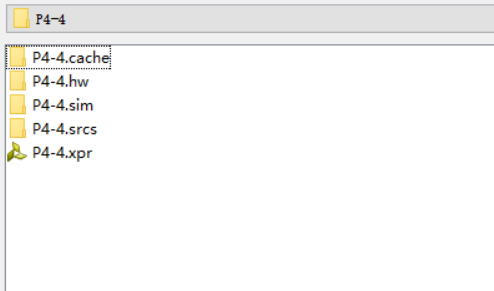


图5 4选1数据选择器原理示意图

请完成4选1数据选择器的建模与验证。

### 创建工程P4-4



### “4-1多路选择器”建模

|  |  |  |
| --- | --- | --- |
| **A1** | **A2** | **Y** |
| 0 | 0 | **D0** |
| 0 | 1 | **D1** |
| 1 | 0 | **D2** |
| 1 | 1 | **D3** |

module selector4(A,D,Y);

input wire [3:0]D;

input wire [1:0]A;

output wire Y;

assign Y =

(!A[1] & !A[0])?D[0]:

(!A[1] & A[0])?D[1]:

(A[1] & !A[0])?D[2]:

D[3];

endmodule

### 编写Test Bench

`timescale 1ns/100ps

module test\_selector4();

reg clk\_1Hz;

reg [3:0]D;

reg [1:0]A;

wire Y;

selector4 u0(A,D,Y);

always #25 clk\_1Hz = ~clk\_1Hz ;

initial

begin

clk\_1Hz=0;

D = 4'b0001;A = 2'b00;

#50;

D = 4'b0001;A = 2'b01;

#50;

D = 4'b0001;A = 2'b10;

#50;

D = 4'b0001;A = 2'b11;

#50;

D = 4'b0010;A = 2'b00;

#50;

D = 4'b0010;A = 2'b01;

#50;

D = 4'b0010;A = 2'b10;

#50;

D = 4'b0010;A = 2'b11;

#50;

D = 4'b0100;A = 2'b00;

#50;

D = 4'b0100;A = 2'b01;

#50;

D = 4'b0100;A = 2'b10;

#50;

D = 4'b0100;A = 2'b11;

#50;

D = 4'b1000;A = 2'b00;

#50;

D = 4'b1000;A = 2'b01;

#50;

D = 4'b1000;A = 2'b10;

#50;

D = 4'b1000;A = 2'b11;

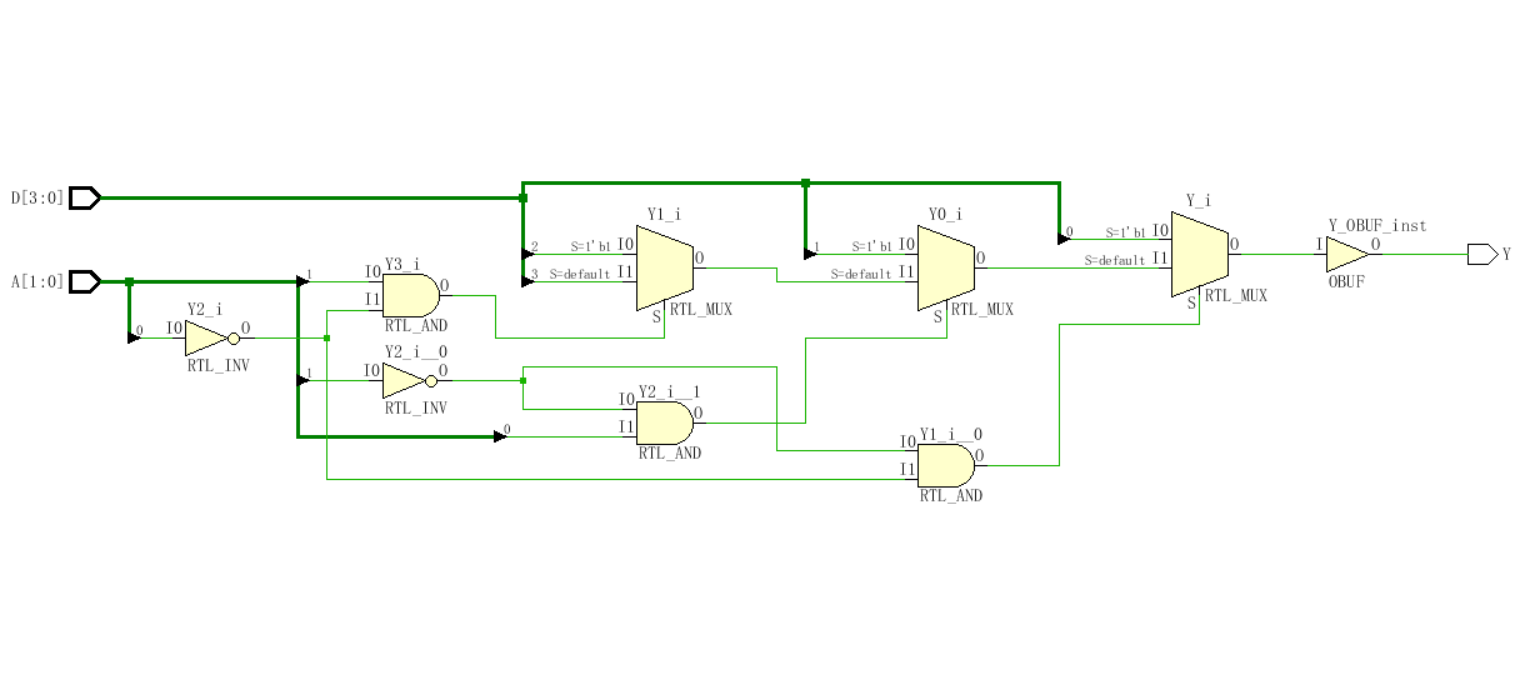
end

endmodule

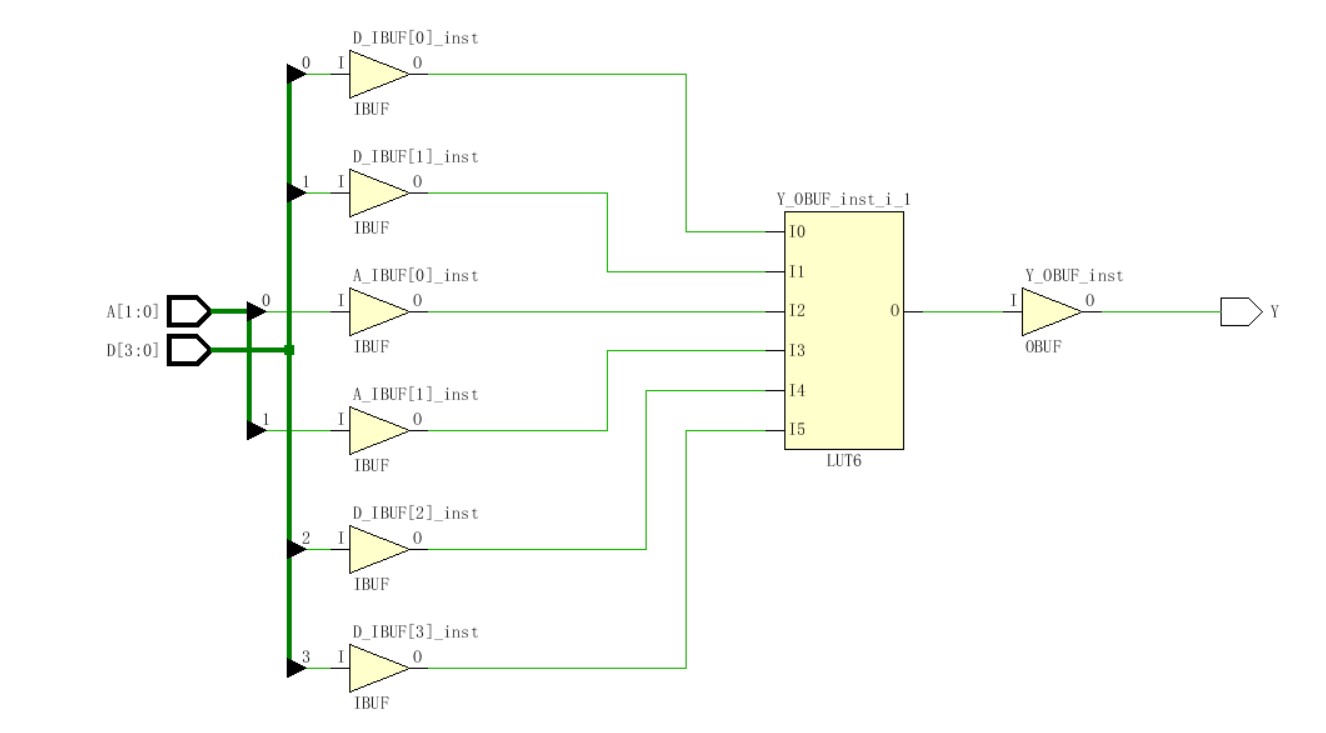
### 仿真，查看波形



### 进行RTL分析，查看电路原理图



### 进行综合，查看综合后电路原理图



# 五、实验思考

## 1.数据流建模的特点是什么？有何优点和缺点？

**【答】**

**特点：**数据流建模是根据数据在寄存器之间流动和处理的过程对电路进行描述。

**优点：**能从更高的抽象层次建模，将设计的重点放在电路的功能上，能有有效提高设计的效率。

**缺点：**在系统运行时会多一层解析，理论上会降低系统的速度

## 2. 数据流建模中多条assgin连续赋值语句的赋值符号左侧可以是什么？右边可以是什么？

**【答】** 左侧必须是一个标量或向量线网，或者是标量或向量线网的拼接，右侧是操作数。

## 3.数据流建模中多条assgin连续赋值语句的顺序对建模有影响吗？它们之间是顺序的关系还是并行的关系？

**【答】** 无影响，并行。

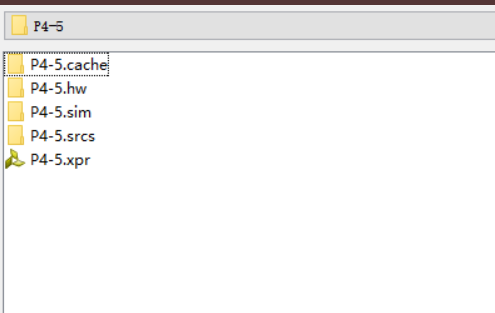
## 4.已知“4位超前进位加法器”输出方程如下：



试用数据流建模方法进行描述，并仿真、查看RTL原理图及综合后原理图。

**【答】**

### (1)建立工程P4-5



### (2)“4位超前进位加法器” 数据流建模

module parallel\_adder(a,b,cin,s,cout);

parameter N=4;//二进制位数

input wire [N-1:0]a;//加数

input wire [N-1:0]b;//被加数

input wire cin;//进位输入

output wire [N-1:0]s;//和

output wire cout;//进位输出

wire[N:0] g,p,c;

assign c[0]=cin;

assign p=a^b;

assign g=a&b;

assign c[1]=g[0]|(p[0]&c[0]);

assign c[2]=g[1]|(p[1]&(g[0]|(p[0]&c[0])));

assign c[3]=g[2]|(p[2]&(g[1]|(p[1]&(g[0]|(p[0]&c[0])))));

assign c[4]=g[3]|(p[3]&(g[2]|(p[2]&(g[1]|(p[1]&(g[0]|(p[0]&c[0])))))));

assign s=p^c[3:0];

assign cout=c[4];

endmodule

### (3)设计“4位超前进位加法器”Test Bench

`timescale 1ns/100ps

module test\_adder();

reg clk\_1Hz;

reg [3:0]a,b;

reg cin;

wire [3:0]s;

wire cout;

parallel\_adder u0(.a(a),.b(b),.cin(cin),.s(s),.cout(cout));

always #50 clk\_1Hz = ~clk\_1Hz ;

initial

begin

clk\_1Hz=0;

a=4'b0000;b=4'b0000;cin=1'b0;

#100;

a=4'b0000;b=4'b0001;cin=1'b0;

#100;

b=4'b1111;cin=1'b1;

#100;

a=4'b1101;b=4'b0100;

#100;

a=4'b0000;b=4'b1001;cin=1'b0;

#100;

a=4'b1000;b=4'b0110;

end

endmodule

### (4)仿真“4位超前进位加法器”，查看波形



### (5)分析 “4位超前进位加法器”，查看RTL原理图

### (6)综合“4位超前进位加法器”，查看原理图

