

**本科实验报告**

课程名称： 计算机组成原理实验

课程编号： 08060038

学生姓名： 邝庆璇

学号： 2016051598

学院： 信息科学与技术学院

系： 计算机系

专业： 计算机科学与技术

指导教师： 王传胜 梁倬骞

教师单位： 暨南大学计算机系

开课时间：2019~ 2020学年度 第1期

**暨南大学教务处**

**2019年12月05日**

数字逻辑实验 **课程实验项目目录**

学生姓名：邝庆璇 学号：2016051598

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 实验项目编号 | 实验项目名称 | \*实验项目类型 | 成绩 | 指导教师 |
| 1 | **0806003801** | **验证逻辑门** | **验证** |  | **王传胜 梁倬骞** |
| 2 | **0806003802** | **基本逻辑门电路之间的转换** | **验证** |  | **王传胜 梁倬骞** |
| 3 | **0806003803** | **超前进位全加器** | **验证** |  | **王传胜 梁倬骞** |
| 4 | **0806003804** | **译码器** | **验证** |  | **王传胜 梁倬骞** |
| 5 | **0806003805** | **数字比较器** | **验证** |  | **王传胜 梁倬骞** |
| 6 | **0806003806** | **七段数码管** | **验证** |  | **王传胜 梁倬骞** |
| 7 | **0712000807** | **计数器** | **验证** |  | **王传胜 梁倬骞** |
| 8 | **0712000808** | **寄存器** | **验证** |  | **王传胜 梁倬骞** |
| 9 | **0712000809** | **时序逻辑电路综合设计** | **验证** |  | **王传胜 梁倬骞** |
| 12 |  |  |  |  |  |
| 13 |  |  |  |  |  |
| 14 |  |  |  |  |  |
| 15 |  |  |  |  |  |
| 16 |  |  |  |  |  |
| 17 |  |  |  |  |  |
| 18 |  |  |  |  |  |

\*实验项目类型：演示性、验证性、验证性、设计性实验。

\*此表由学生按顺序填写。

**暨南大学本科实验报告专用纸**

课程名称 数字逻辑实验 成绩评定

实验项目名称 Verilog门级建模 指导教师 王传胜 梁倬骞

实验项目编号 0806003801 实验项目类型 验证 实验地点 N126

学生姓名 邝庆璇 学号 2016051598

学院 信息科学技术 系 计算机科学 专业 计算机科学与技术

实验时间 2019 年 9 月 30 日 上 午

一、实验目的

1、学会使用Verilog HDL进行门级建模；

2、初步学会怎么用Verilog HDL编写仿真程序；

3、初步学会怎么使用Vivado软件；

4、初步学会验证基本逻辑门的逻辑功能的方法；

5、初步学会验证怎么使用EG0-1实验板。

二、实验内容

编写Verilog程序，验证3输入的与门、与非门、或门、或非门、异或门、同或门

三、实验程序

exp1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/09/30 11:28:50

// Design Name:

// Module Name: exp-1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module exp1(

input in1,

input in2,

input in3,

output out1,

output out2,

//output out3,

output out4,

output out5,

output out6,

output out7

);

and funcAnd(out1, in1, in2, in3);

or funcOr(out2, in1, in2, in3);

xor funcXor(out4, in1, in2, in3);

nand funcNand(out5, in1, in2, in3);

nor funcNor(out6, in1, in2, in3);

xnor funcXnor(out7, in1, in2, in3);

endmodule

四、仿真程序

sim\_1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/09/30 11:57:26

// Design Name:

// Module Name: sim\_1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module sim\_1(

output out1,

output out2,

//output out3,

output out4,

output out5,

output out6,

output out7

);

reg[2:0] in;

initial

begin

in = 3'b00;

end

always #10

begin

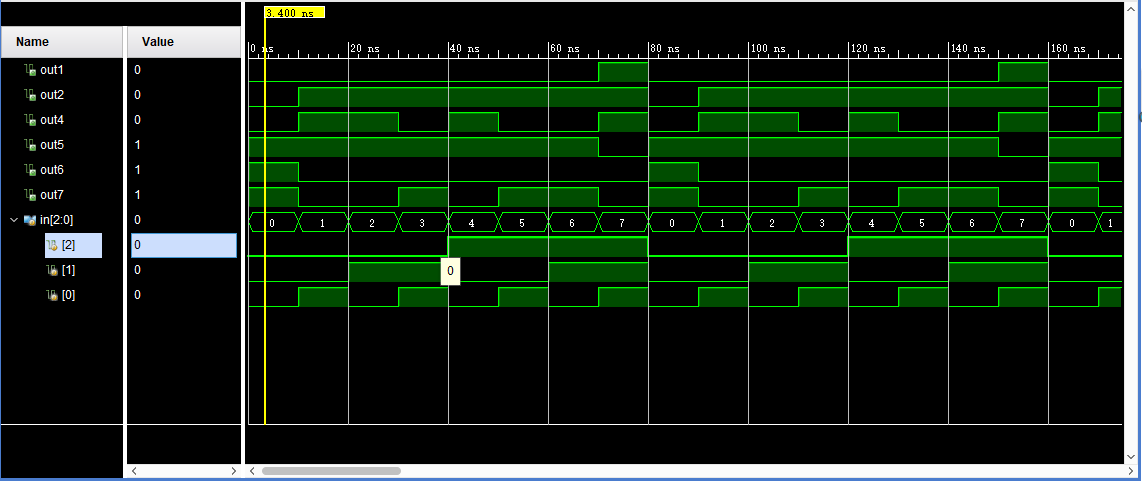
in = in + 1;

end

exp1 exp1(in[2],in[1], in[0], out1, out2, out4, out5, out6, out7);

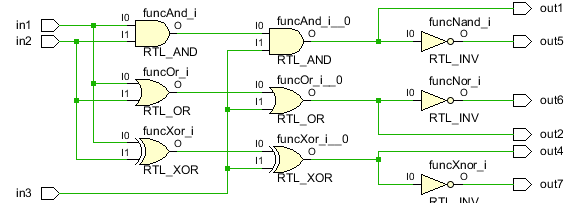
endmodule

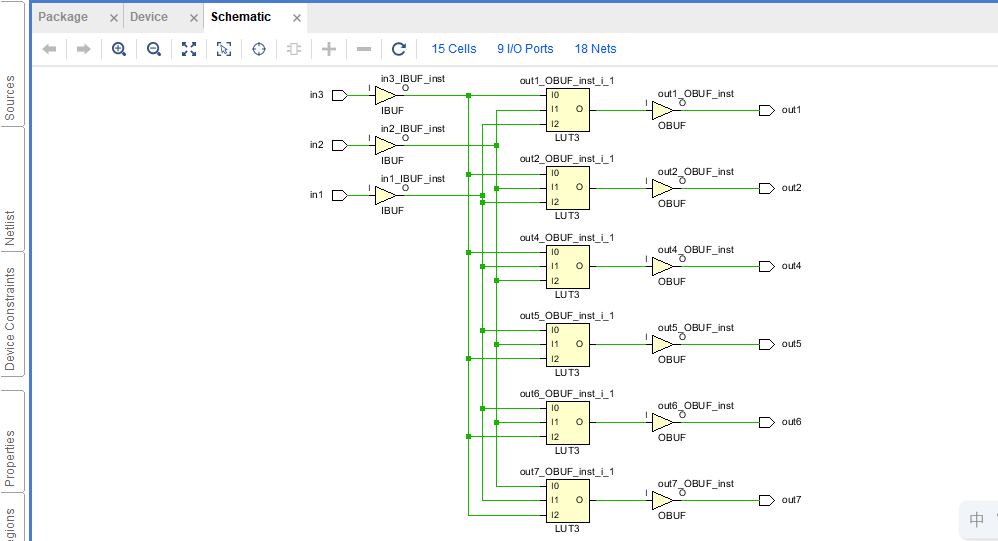
五、仿真结果



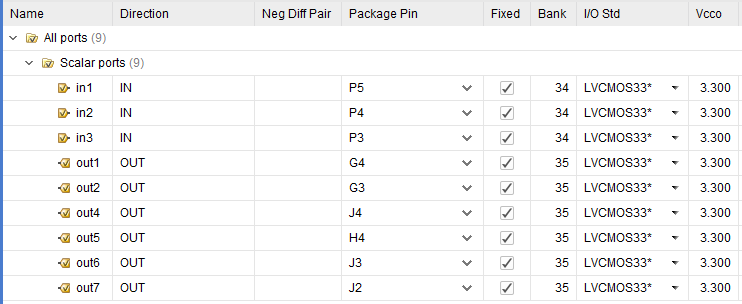
六、系统网表

RTL Analysis



Synthesis

七、系统约束



八、实验结果

示例：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 输入 | In1 | | In2 | | In3 | | |
| 输入的值 | 0 | | 2 | | 0 | | |
| 输出 | out1 | out2 | out4 | out5 | | out6 | out7 |
| 逻辑运算 | And | Or | Xor | Nand | | Nor | xnor |
| 输出的值 | 0 | 1 | 1 | 1 | | 0 | 0 |



九、实验体会

此次试验属于初步的实践，学习了门级建模，熟悉了vivado软件的使用及verilog编程的语法，有效地为数字逻辑试验和学习作准备。感悟：刚开始使用时，使用软件进行建模的流程相对繁琐，但熟悉之后会比较顺利，毕竟初期的实验难度不大。

**暨南大学本科实验报告专用纸**

课程名称 数字逻辑实验 成绩评定

实验项目名称 基本逻辑门电路之间的转换 指导教师 王传胜 梁倬骞

实验项目编号 0806003802 实验项目类型 验证 实验地点 N126

学生姓名 邝庆璇 学号 2016051598

学院 信息科学技术 系 计算机科学 专业 计算机科学与技术

实验时间 2019 年 10 月 14 日 上 午

一、实验目的

掌握用Verilog HDL 进行门级建模；

熟悉用Verilog HDL写仿真程序；

熟悉使用Vivado软件；

学会验证基本逻辑门电路之间的转换方法；

熟悉使用EGO-1实验板。

二、实验内容

用与非门（NAND）来实现异或（XOR）运算

三、实验程序

exp1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/10/14 12:11:56

// Design Name:

// Module Name: exp\_1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module exp\_1(

input in1,

input in2,

output out

);

nand (temp1, in1, in2);

nand (temp2, in1, temp1);

nand (temp3, in2, temp1);

nand (out, temp2, temp3);

endmodule

四、仿真程序

sim\_1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/10/14 12:22:02

// Design Name:

// Module Name: sim\_1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module sim\_1(

output out

);

reg[1:0] in;

initial

begin

in = 3'b00;

end

always #10

begin

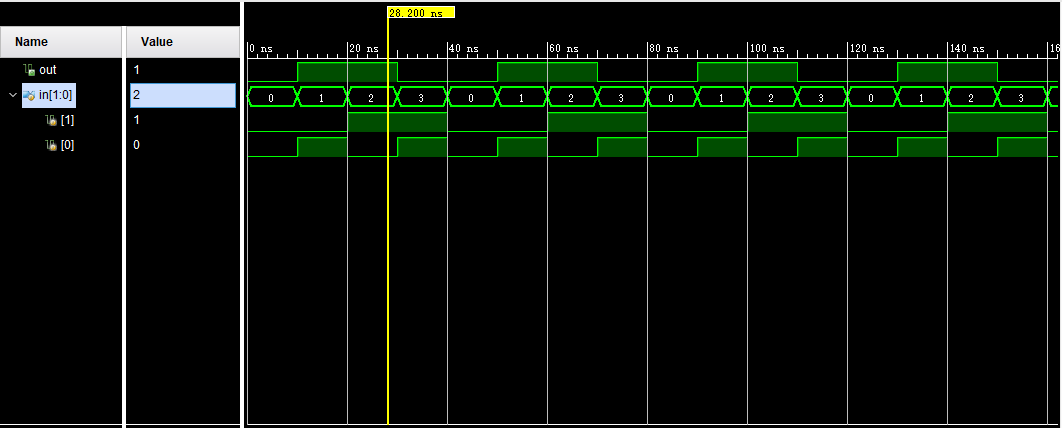
in = in + 1;

end

exp\_1 exp\_1(in[1], in[0], out);

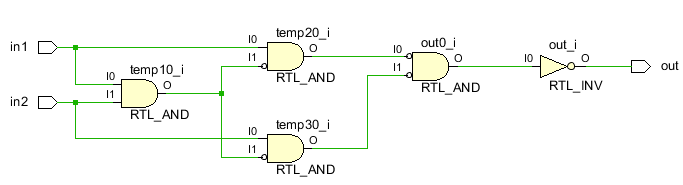
Endmodule

五、仿真结果

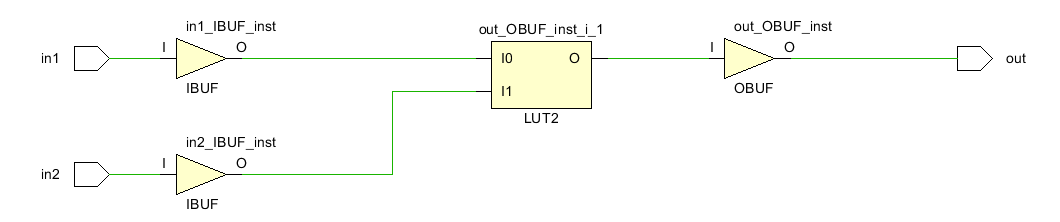


六、系统网表

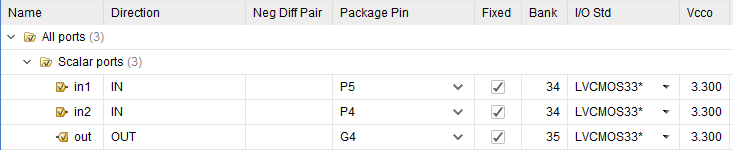
RTL Analysis



Synthesis



七、系统约束

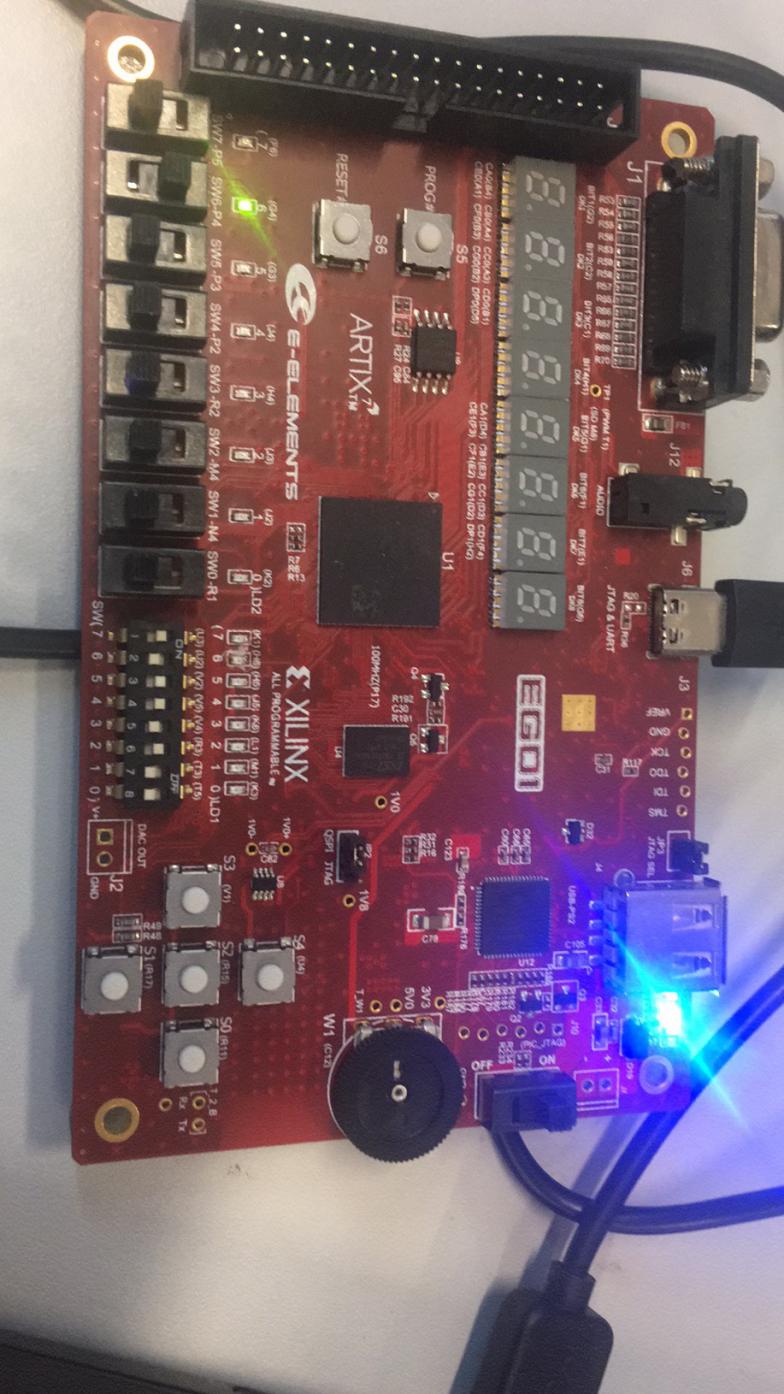


八、实验结果

示例：

|  |  |  |
| --- | --- | --- |
| 输入 | In1 | In2 |
| 输入端口 | P5 | P4 |
| 输入的值 | 0 | 1 |
| 输出 | out | |
| 输出端口 | G4 | |
| 输出的值 | 1 | |

（示例照片见下页）



九、实验体会

通过此次试验，进一步地联系了门级建模，也更加熟悉vivado的编程及使用流程（包括模拟simulation，综合synthesis，根据硬件设备指定I/O Ports，生产比特流generate bitstream，到最后把程序放到设备里面），还有对每一步结果分析。

**暨南大学本科实验报告专用纸**

课程名称 数字逻辑实验 成绩评定

实验项目名称 超前进位加法器 指导教师 王传胜 梁倬骞

实验项目编号 0806003803 实验项目类型 验证 实验地点 N126

学生姓名 邝庆璇 学号 2016051598

学院 信息科学技术 系 计算机科学 专业 计算机科学与技术

实验时间 2019 年 10 月 21 日 上 午

一、实验目的

学会用Verilog HDL 进行数据流建模；

掌握用Verilog HDL 写仿真程序；

掌握使用Vivado 软件；

学会超前进位全加器的原理；

掌握使用EGO-1实验板。

二、实验内容

用Verilog HDL 实现3位的超前进位加法器。

\*注意：必须使用超前进位实现！！！不能{c4,sum}=a+b+c0

三、实验程序

exp1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/10/21 12:43:16

// Design Name:

// Module Name: exp\_1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module exp\_1(

input [2:0] A,

input [2:0] B,

input C\_in,

output C\_out,

output [2:0] S

);

wire [2:0] G = A & B;

wire [2:0] P = A ^ B;

// 2为高位，0为低位, 210

wire C1, C2; // 从1开始的A1+B1, A2+B2所产生的进位

assign C1 = G[0] + (P[0] & C\_in); // 记得要加括号呀！

assign C2 = G[1] + (P[1] & C1);

assign C\_out = G[2] + (P[2] & C2);

assign S[0] = P[0] ^ C\_in;

assign S[1] = P[1] ^ C1;

assign S[2] = P[2] ^ C2;

endmodule

四、仿真程序

sim\_1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/10/21 12:58:48

// Design Name:

// Module Name: sim\_1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module sim\_1(

output C\_out,

output [2:0] S

);

reg [5:0] in;

reg [2:0] A;

reg [2:0] B;

reg inc;

initial

begin

in = 6'b000000;

assign A = in[5:3];

assign B = in[2:0];

assign inc = 1;

end

always #10

begin

in = in + 1;

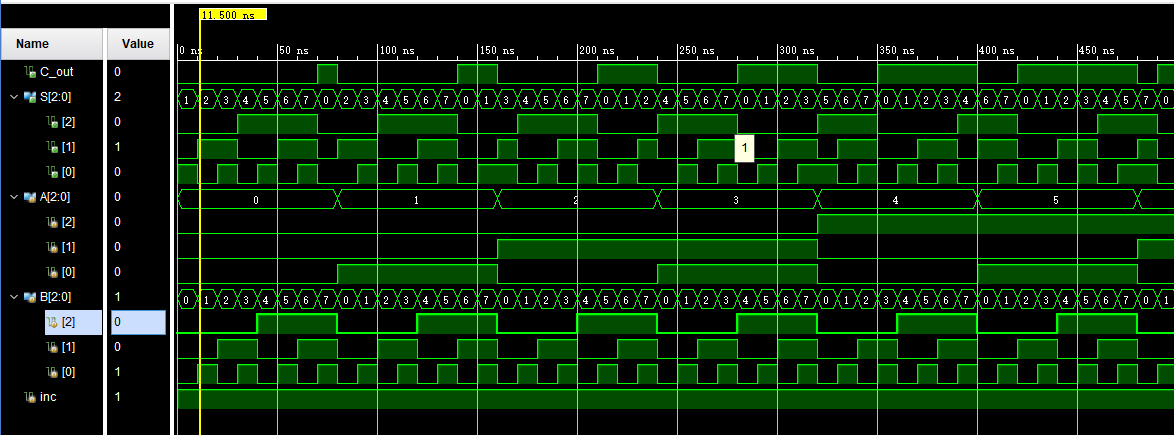
inc = inc + 1;

end

exp\_1 exp\_1(A, B, inc, C\_out, S);

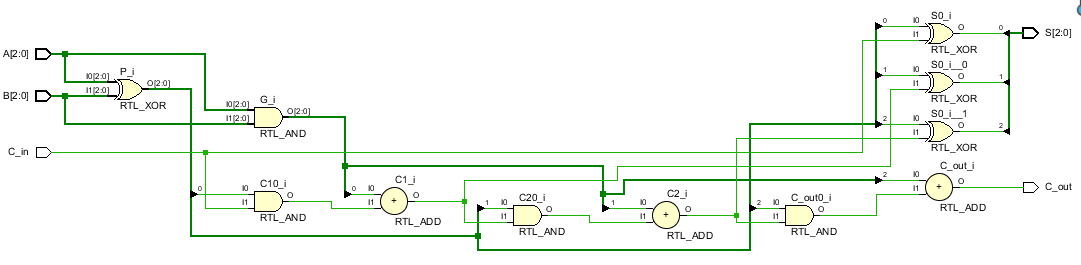
endmodule

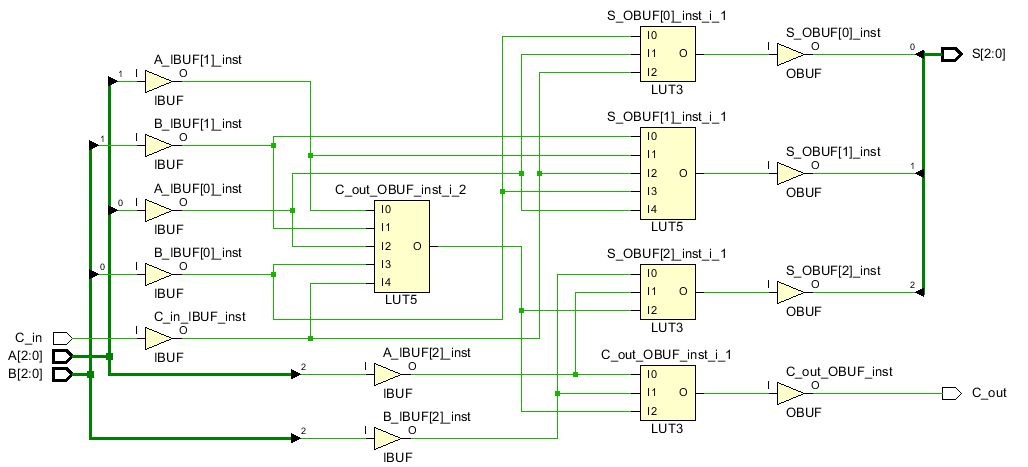
五、仿真结果



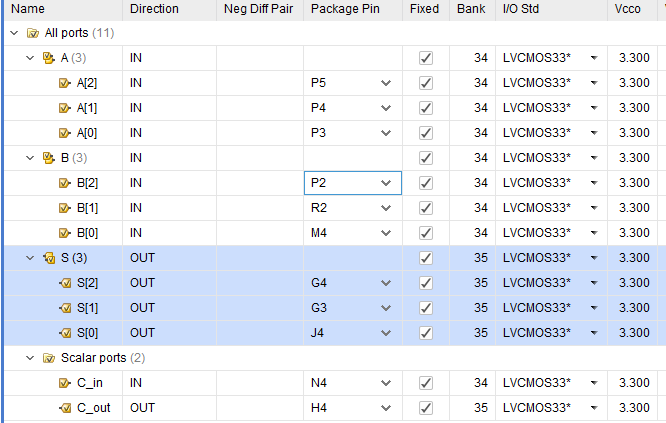
六、系统网表

RTL Analysis



Synthesis

七、系统约束

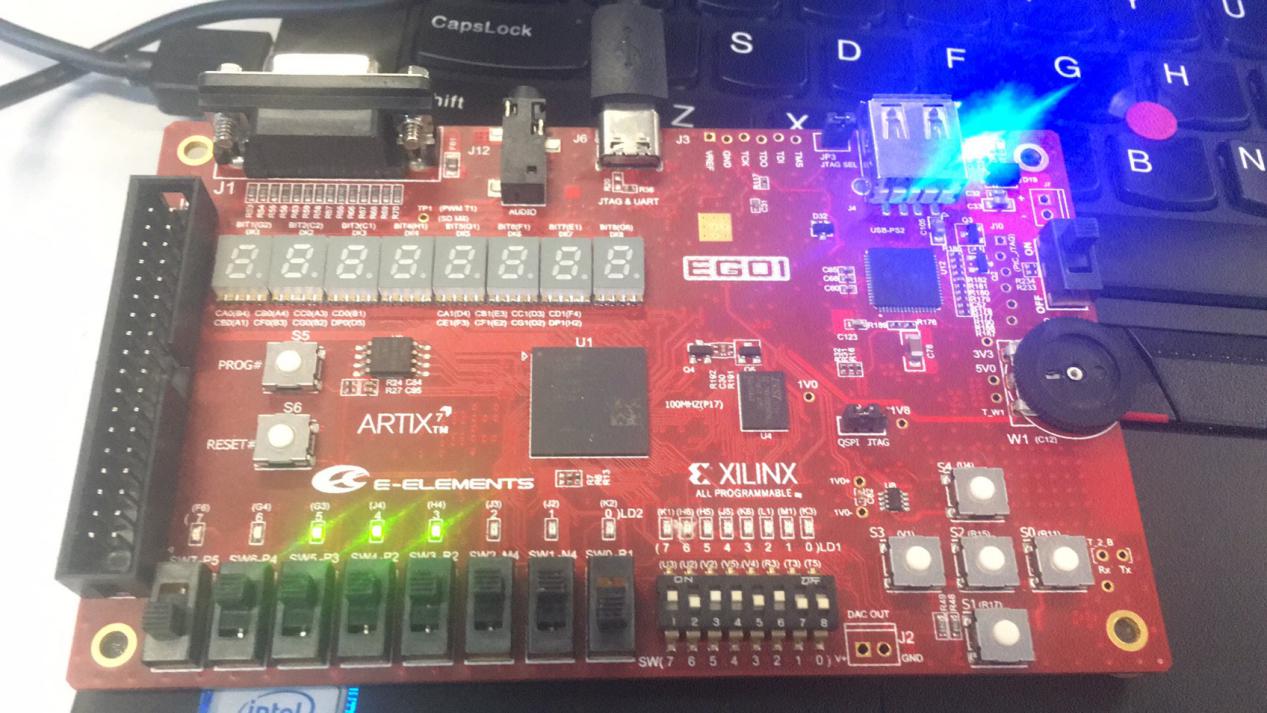


八、实验结果

示例（3+7+1=11=8+3）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 输入 | A (A2 A1 A0) | B (B2 B1 B0) | | C\_in |
| 输入端口 | P5 P4 P3 | P2 R2 M4 | | N4 |
| 输入的值 | 0 1 1 | 1 1 1 | | 1 |
| 输出 | S (S2 S1 S0) | | C\_out | |
| 输出端口 | G4 G3 J4 | | H4 | |
| 输出的值 | 0 1 1 | | 1 | |

（结果照片见下一页）



九、实验体会

通过此次试验，不但学习并练习了Verilog HDL进行数据流建模，熟悉了Verilog的各种操作符，而且还复习了超前进位全加器的原理。诚然，在编码过程当中，仍存在对Verilog语法及Vivado软件使用不熟练，导致走了一点弯路（比如没意识到实验程序exp1.v不能用while, for等语句，仿真文件里才能用），但在请教老师同学后迅速解决了问题，并完成了实验。

**暨南大学本科实验报告专用纸**

课程名称 数字逻辑实验 成绩评定

实验项目名称 译码器 指导教师 王传胜 梁倬骞

实验项目编号 0806003804 实验项目类型 验证 实验地点 N126

学生姓名 邝庆璇 学号 2016051598

学院 信息科学技术 系 计算机科学 专业 计算机科学与技术

实验时间 2019 年 10 月 28 日 上 午

一、实验目的

掌握用 Verilog HDL 进行数据流建模；

深入理解译码器的原理；

学会用 Verilog HDL 实现74LS138；

二、实验内容

用Verilog HDL 语言实现74LS138的功能。

三、实验程序

exp1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/10/28 12:13:17

// Design Name:

// Module Name: exp\_1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module exp\_1(

input [2:0] G,

input [2:0] A,

output [7:0] Y

);

wire E\_tmp = (G[0] & ~(G[1]) & ~(G[2]));

wire A0\_not = ~A[0];

wire A1\_not = ~A[1];

wire A2\_not = ~A[2];

assign Y[0] = ~(A0\_not & A1\_not & A2\_not & E\_tmp);

assign Y[1] = ~(A[0] & A1\_not & A2\_not & E\_tmp);

assign Y[2] = ~(A0\_not & A[1] & A2\_not & E\_tmp);

assign Y[3] = ~(A[0] & A[1] & A2\_not & E\_tmp);

assign Y[4] = ~(A0\_not & A1\_not & A[2] & E\_tmp);

assign Y[5] = ~(A[0] & A1\_not & A[2] & E\_tmp);

assign Y[6] = ~(A0\_not & A[1] & A[2] & E\_tmp);

assign Y[7] = ~(A[0] & A[1] & A[2] & E\_tmp);

endmodule

四、仿真程序

sim\_1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/10/28 12:16:20

// Design Name:

// Module Name: sim\_1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module sim\_1(

output [7:0] Y

);

reg [2:0] G;

reg [2:0] A;

initial

begin

assign A = 3'b000;

assign G = 3'b001;

end

always #20

begin

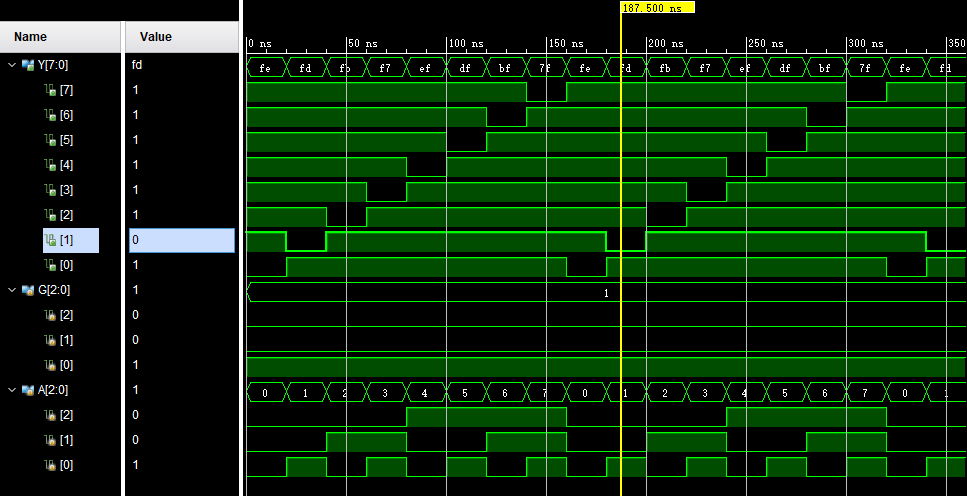
assign A = A + 1;

end

exp\_1 exp\_1(G, A, Y);

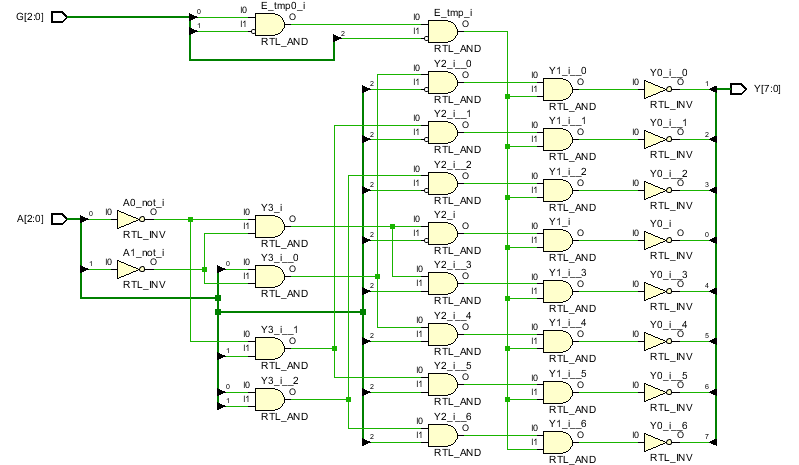
Endmodule

五、仿真结果

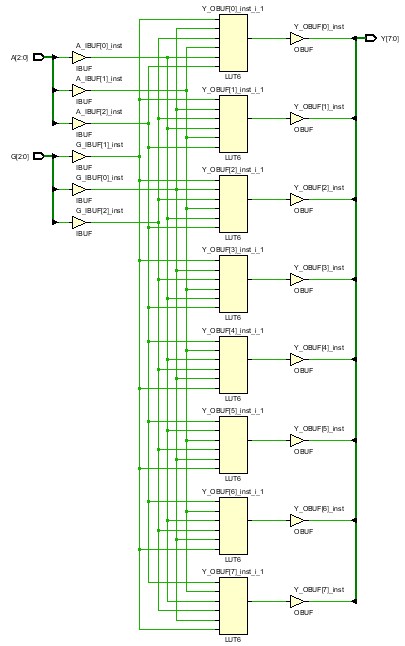


六、系统网表

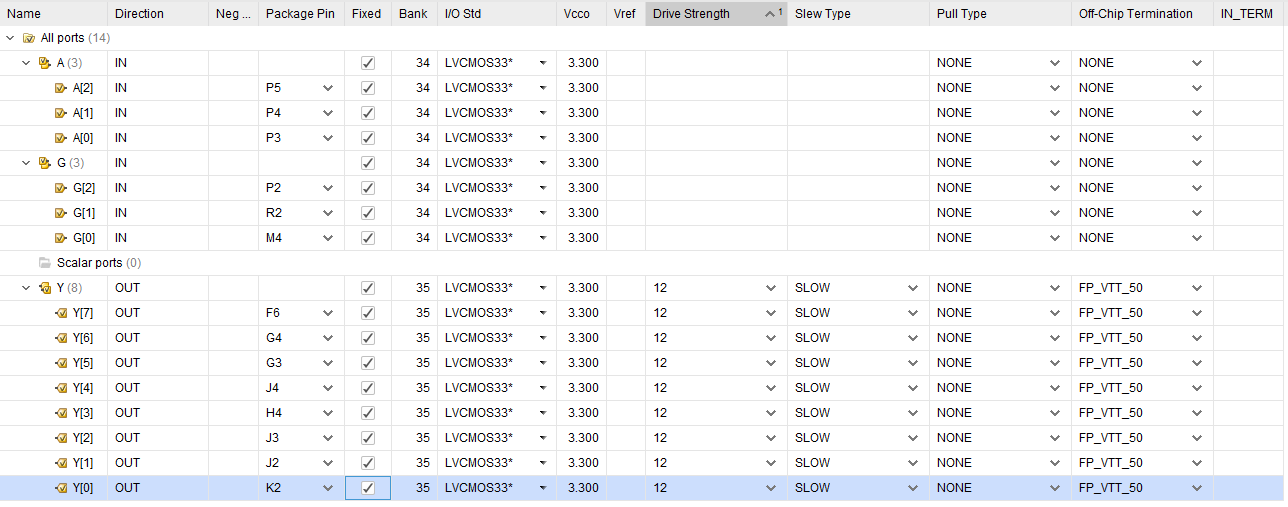
RTL Analysis



Synthesis



七、系统约束

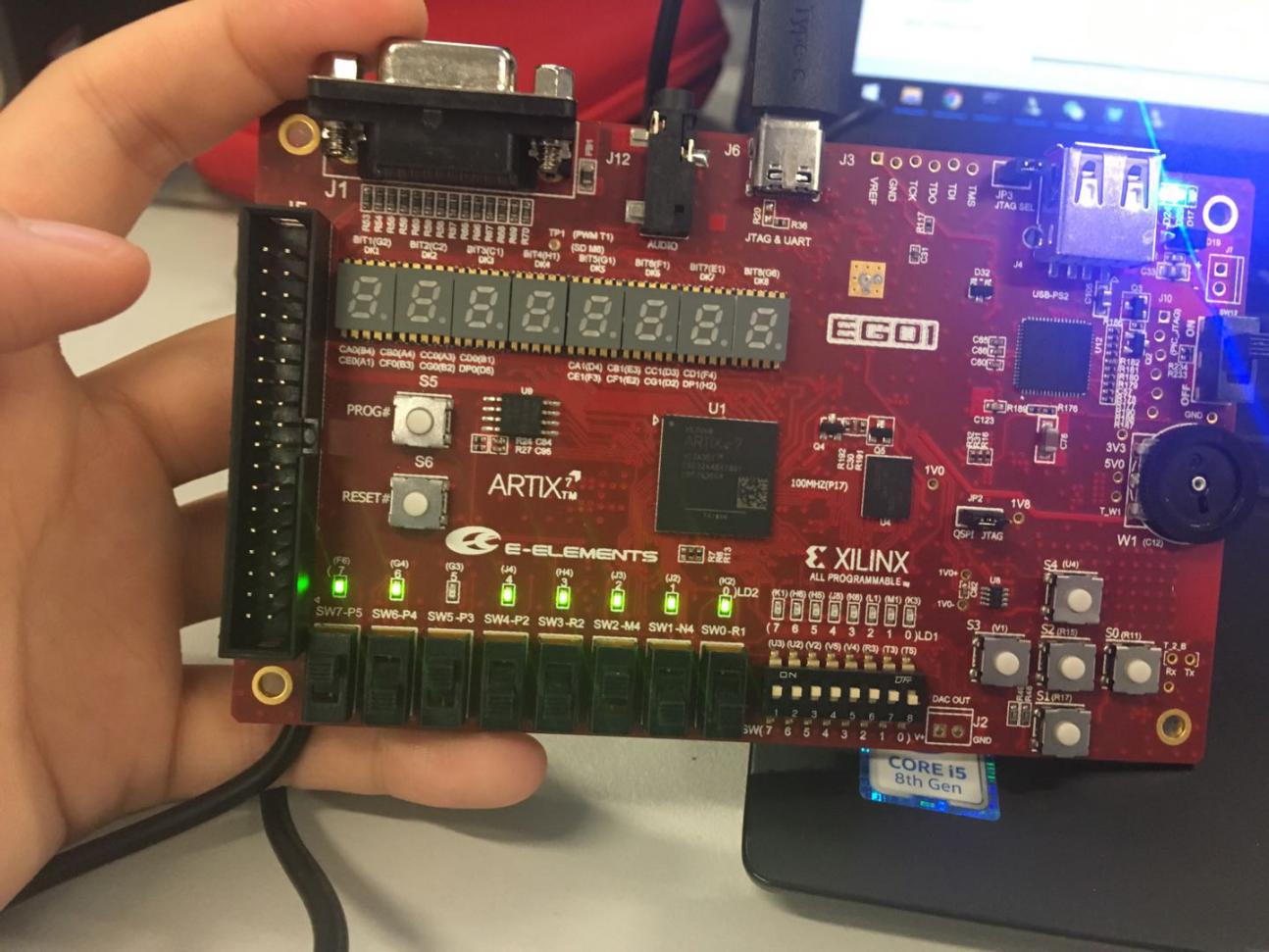


八、实验结果

示例（A=101, G=001）：

|  |  |  |
| --- | --- | --- |
| 输入 | A (A2 A1 A0) | G (G3 G2 G1) |
| 输入端口 | P5 P4 P3 | P2 R2 M4 |
| 输入的值 | 1 0 1 | 0 0 1 |
| 输出 | Y (Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0) | |
| 输出端口 | F6 G4 G3 J4 H4 J3 J2 K2 | |
| 输出的值 | 1 1 0 1 1 1 1 1 | |

（照片见下页）



九、实验体会

在做之前的经验积累铺垫下，此次实验完成得比较顺利。但因为一开始的时候是照着图而不是真值表来编码，而图的线路比较复杂（看起来），所以很容易乱；然后改成看真值表，就容易找得到规律了。另外，要注意细心一点儿。

**暨南大学本科实验报告专用纸**

课程名称 数字逻辑实验 成绩评定

实验项目名称 数字比较器 指导教师 王传胜 梁倬骞

实验项目编号 0806003805 实验项目类型 验证 实验地点 N126

学生姓名 邝庆璇 学号 2016051598

学院 信息科学技术 系 计算机科学 专业 计算机科学与技术

实验时间 2019 年 11 月 04 日 上 午

一、实验目的

掌握Verilog HDL进行数据流建模；

学会使用层次建模的方式来设计整个系统；

学会用Verilog HDL实现数字比较器

二、实验内容

用Verilog HDL 数据流模式实现由级联2位数值比较器

三、实验程序

exp1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/04 11:28:52

// Design Name:

// Module Name: exp\_1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module exp\_1(

input [1:0] A,

input [1:0] B,

input [2:0] I, // I[2]: A>B; I[1]: A<B; I[0]: A==B

output [2:0] F // F[2]: A>B; F[1]: A<B; F[0]: A==B

);

// A > B

assign F[2] = ((A[1] > B[1]) || (A[1] == B[1] && A[0] > B[0]) || (A[1] == B[1] && A[0] == B[0] && I[2] > 0));

// A < B

assign F[1] = ((A[1] < B[1]) || (A[1] == B[1] && A[0] < B[0]) || (A[1] == B[1] && A[0] == B[0] && I[1] > 0));

// A == B

assign F[0] = (A[1] == B[1] && A[0] == B[0] && I[0] > 0);

endmodule

四、仿真程序

sim\_1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/04 11:37:29

// Design Name:

// Module Name: sim\_1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module sim\_1(

output [2:0] F

);

reg [1:0] A;

reg [1:0] B;

reg [3:0] AB;

reg [2:0] I;

initial

begin

AB = 4'b0000;

A = AB[3:2];

B = AB[1:0];

I = 3'b001;

end

always #20

begin

AB = AB + 1;

A = AB[3:2];

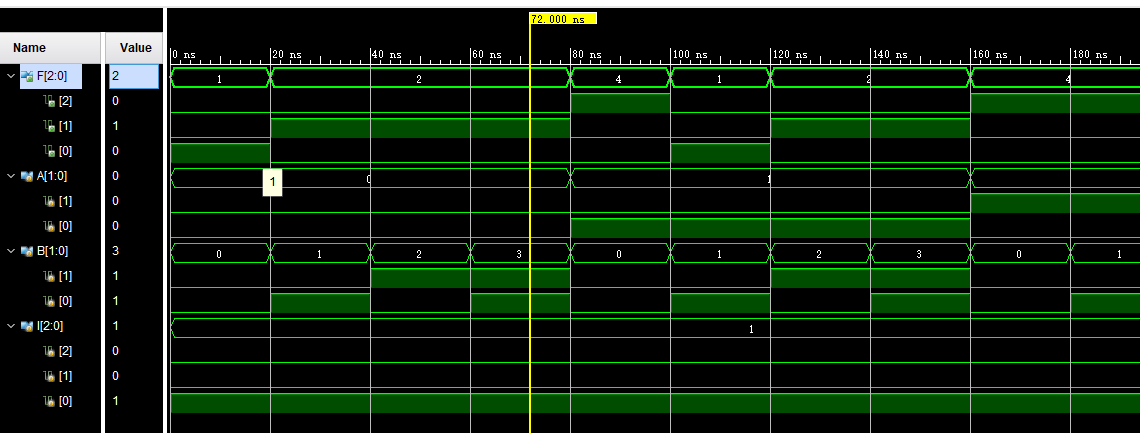
B = AB[1:0];

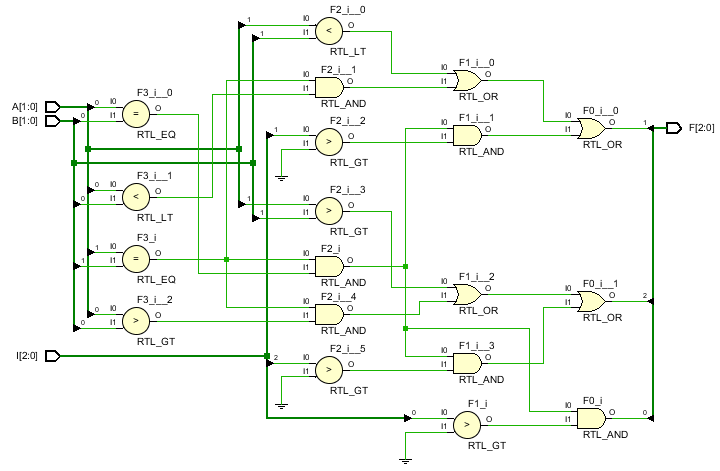
end

exp\_1 exp\_1(A, B, I, F);

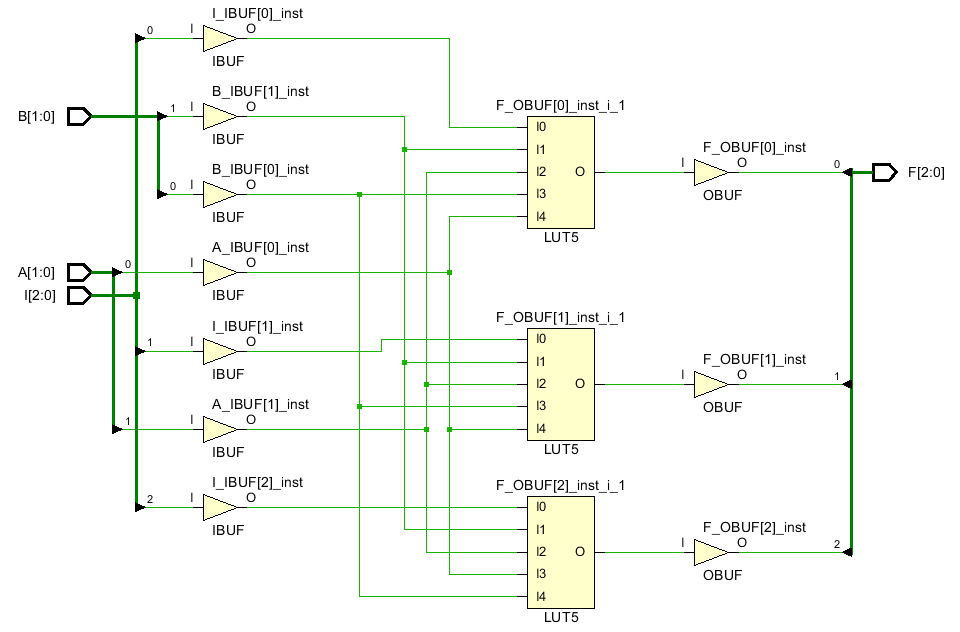
endmodule

五、仿真结果

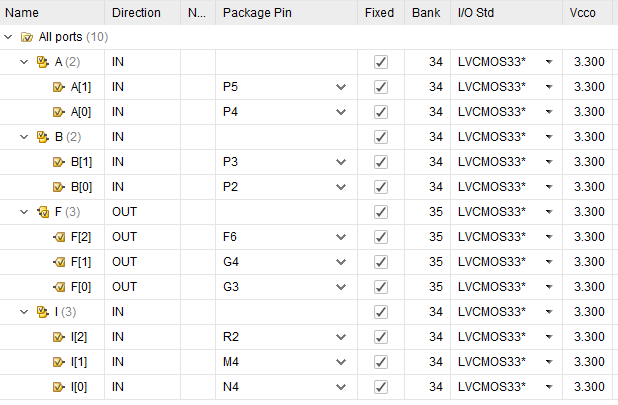


六、系统网表

RTL Analysis

Synthesis

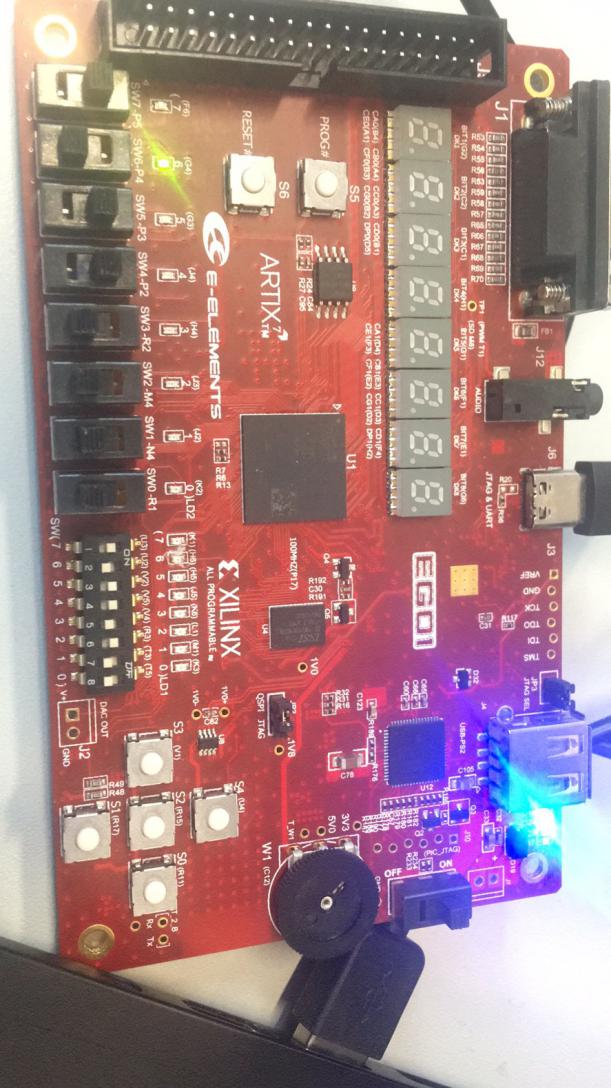
七、系统约束



八、实验结果

示例（A=10, B=11, I=100）：

|  |  |  |  |
| --- | --- | --- | --- |
| 输入 | A (A1 A0) | B(B1 b0) | I |
| 输入的值 | 1 0 | 11 | 100 |
| 输出 | F (F2 F2 F0) | | |
| 输出的值 | 010 | | |



九、实验体会

在做之前的经验积累铺垫下，此次实验完成得比较顺利。但因为一开始的时候是照着图而不是真值表来编码，而图的线路比较复杂（看起来），所以很容易乱；然后改成看真值表，就容易找得到规律了。另外，要注意细心一点儿。

**暨南大学本科实验报告专用纸**

课程名称 数字逻辑实验 成绩评定

实验项目名称 七段数码管 指导教师 王传胜 梁倬骞

实验项目编号 0806003806 实验项目类型 验证 实验地点 N126

学生姓名 邝庆璇 学号 2016051598

学院 信息科学技术 系 计算机科学 专业 计算机科学与技术

实验时间 2019 年 11 月 11 日 上 午

一、实验目的

掌握用 Verilog HDL 进行数据流建模；

学会在EGO-1实验板上使用七段数码管；

学会用 Verilog HDL 实现用七段数码管实现16进制数；

二、实验内容

编写Verilog 程序，使得在七段数码管中显示0~9，A~F这16进制数。

三、实验程序

exp1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/11 13:11:15

// Design Name:

// Module Name: exp\_11

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module exp\_11(

input [3:0] in\_num,

input [3:0] in\_chip,

output reg [6:0] out\_num,

output [3:0] out\_chip

);

assign out\_chip = in\_chip;

always @(in\_num)

case (in\_num)

4'b0000 : out\_num = 7'b1111110;

4'b0001 : out\_num = 7'b0110000;

4'b0010 : out\_num = 7'b1101101;

4'b0011 : out\_num = 7'b1111001;

4'b0100 : out\_num = 7'b0110011;

4'b0101 : out\_num = 7'b1011011;

4'b0110 : out\_num = 7'b1011111;

4'b0111 : out\_num = 7'b1110000;

4'b1000 : out\_num = 7'b1111111;

4'b1001 : out\_num = 7'b1111011;

4'b1010 : out\_num = 7'b1110111;

4'b1011 : out\_num = 7'b0011111;

4'b1100 : out\_num = 7'b1001110;

4'b1101 : out\_num = 7'b0111101;

4'b1110 : out\_num = 7'b1001111;

4'b1111 : out\_num = 7'b1000111;

default : begin

out\_num = 'bx;

$display("not match");

end

endcase

endmodule

四、仿真程序

sim\_1.v

endmodule

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/11 13:14:02

// Design Name:

// Module Name: sim\_11

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module sim\_11(

output [6:0] out\_num,

output [3:0] out\_chip

);

reg [3:0] in\_num;

reg [3:0] in\_chip;

initial

begin

in\_chip = 4'b1010;

in\_num = 4'b0000;

end

always #20

begin

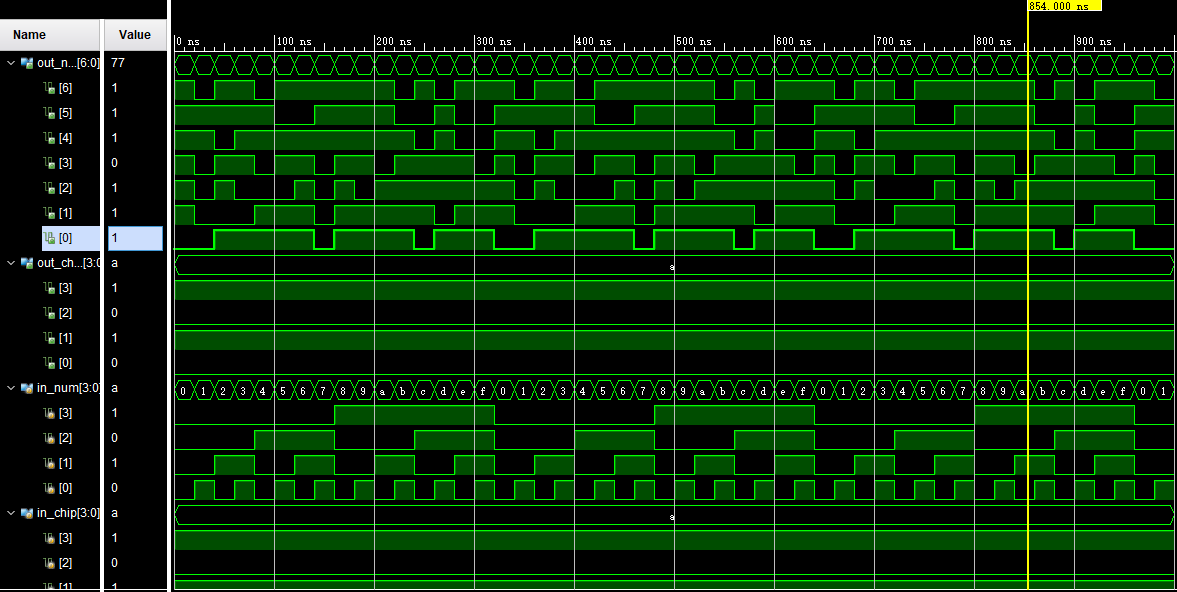
in\_num = in\_num + 1;

end

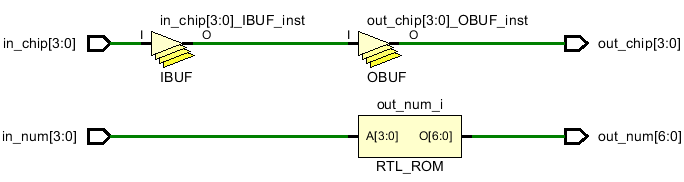
exp\_11 exp\_11(in\_num, in\_chip, out\_num, out\_chip);

endmodule

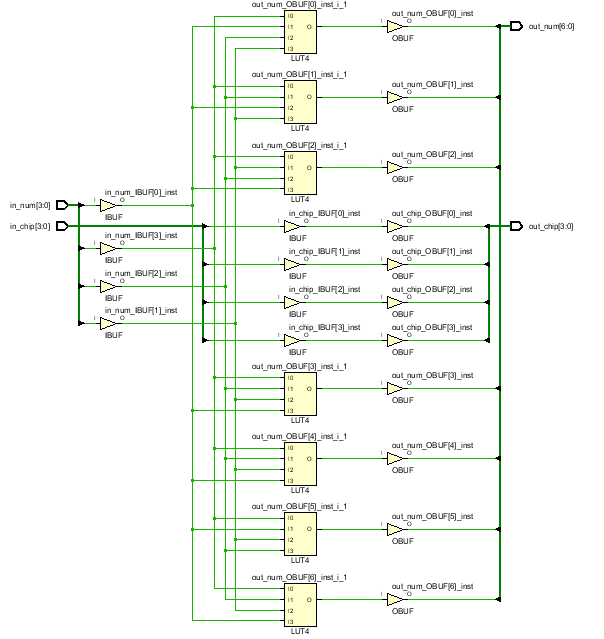
五、仿真结果



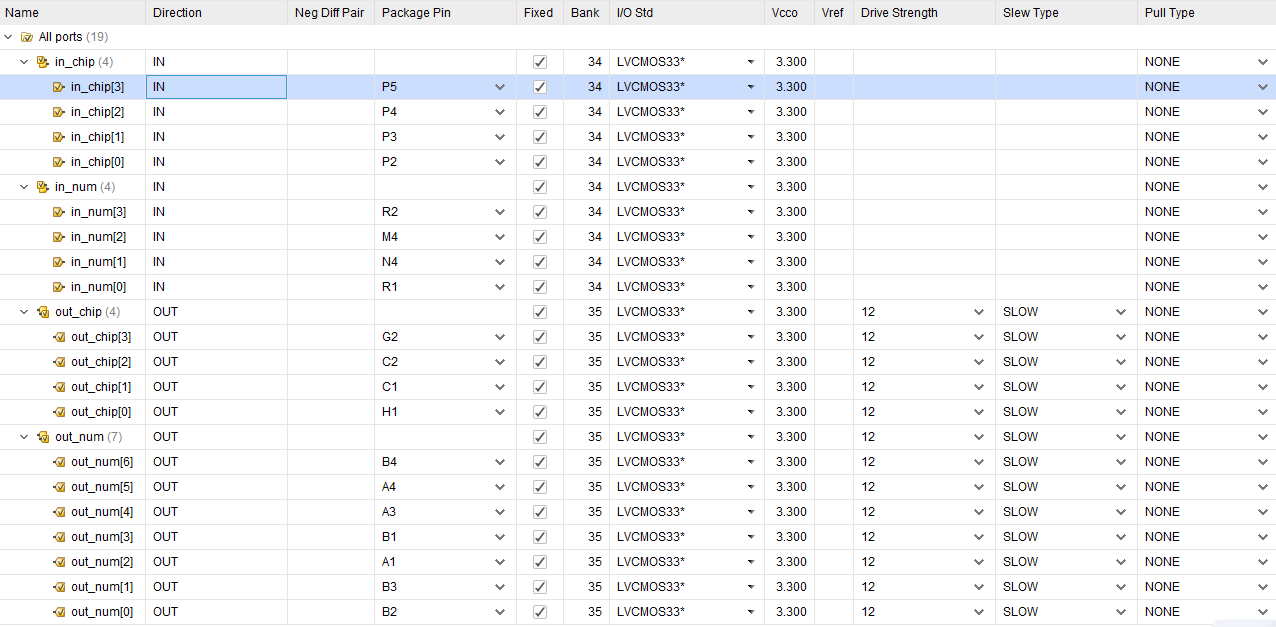
六、系统网表

RTL:

Synthesis：



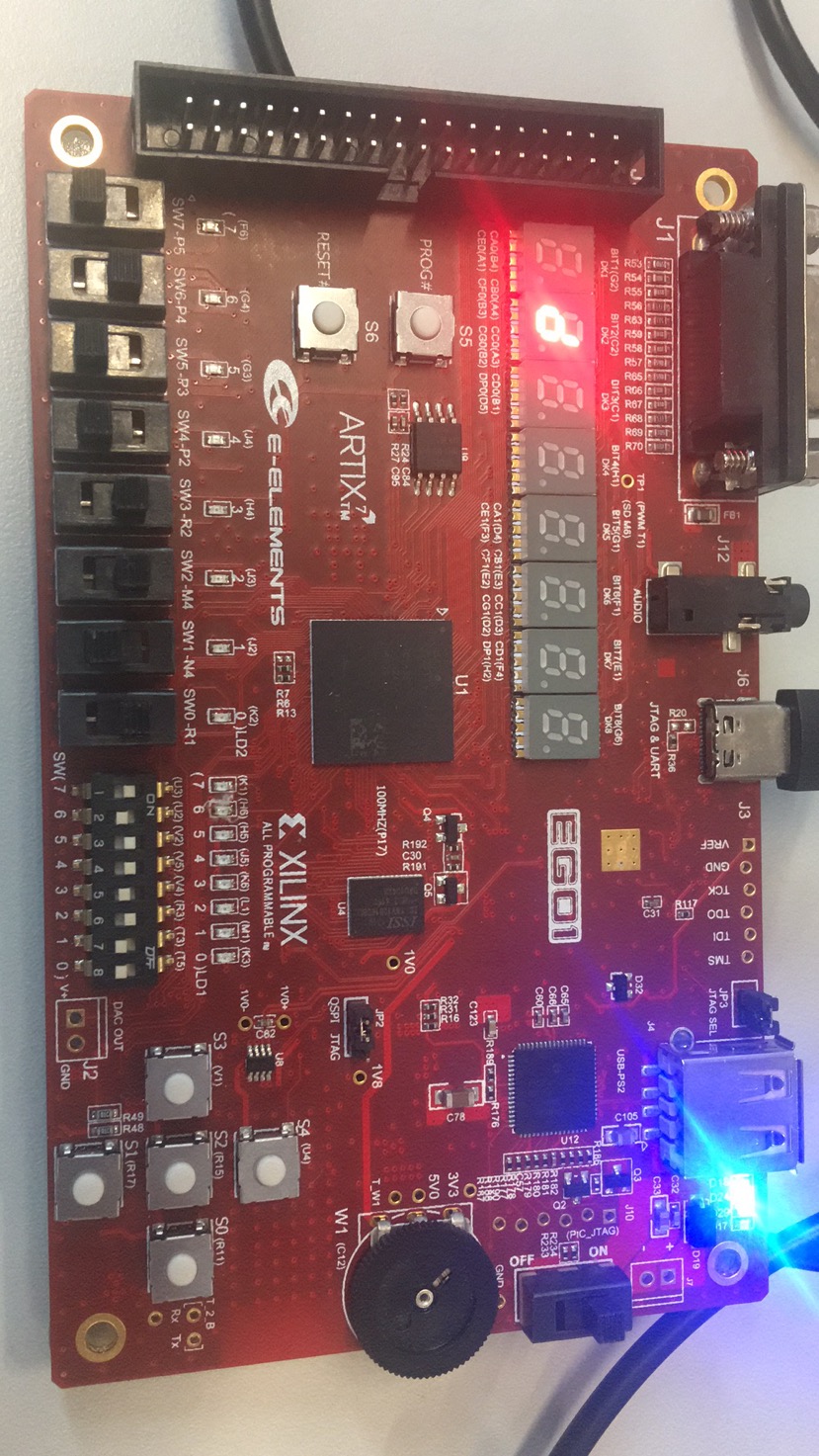
七、系统约束



八、实验结果

示例（A=101, G=001）：

|  |  |  |
| --- | --- | --- |
| 输入 | in\_chip （片选） | in\_num |
| 输入的值 | 0100 | 1101 |
| 输出 | out\_chip | out\_num |
| 输出的值 | 0100 | D |



九、实验体会

在做之前的经验积累铺垫下，此次实验完成得比较顺利。但因为一开始的时候是照着图而不是真值表来编码，而图的线路比较复杂（看起来），所以很容易乱；然后改成看真值表，就容易找得到规律了。另外，要注意细心一点儿。

**暨南大学本科实验报告专用纸**

课程名称 数字逻辑实验 成绩评定

实验项目名称 计数器 指导教师 王传胜 梁倬骞

实验项目编号 0806003807 实验项目类型 验证 实验地点 N126

学生姓名 邝庆璇 学号 2016051598

学院 信息科学技术 系 计算机科学 专业 计算机科学与技术

实验时间 2019 年 11 月 17 日 上 午

一、实验目的

学会使用 Verilog HDL 进行行为建模；

学会的 Verilog HDL 行为建模来实现计数器；

学会用 Verilog HDL 行为建模来实现分频器；

二、实验内容

用Verilog HDL 实现2位的十六进制计数器，并用七段数码管来显示十六进制数00~FF。

三、实验程序

top.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/12/03 23:51:03

// Design Name:

// Module Name: top

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module top(

input clk,

input clr,

output [6:0] a2g,

output [1:0] an

);

wire clk48, clk190;

wire [7:0] data;

div div(clk, clk48, clk190);

counter counter(clk48, clr, data);

seg\_2 seg\_2(data, clk190, clr, a2g, an);

endmodule

div.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/12/03 23:51:03

// Design Name:

// Module Name: div

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module div(

input clk,

output clk48,

output clk190

);

reg[24:0] q = 25'b0;

always @(posedge clk) begin

q <= q + 1;

end

assign clk190 = q[18];

assign clk48 = q[24];

// assign clk190 = clk;

// assign clk48 = clk;

endmodule

counter.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/12/03 23:46:46

// Design Name:

// Module Name: counter

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module counter(

input clk48,

input clr,

output reg [7:0] data

);

reg isFirst = 1;

always @(posedge clk48 or posedge clr) begin

if (clr == 1)

data <= 0;

else begin

if (isFirst) begin

data <= 0;

isFirst = 0;

end

data <= data + 1;

end

end

endmodule

seg\_2.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/12/03 23:29:37

// Design Name:

// Module Name: seg\_2

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module seg\_2(

input [7:0] data,

input clk190,

input clr,

output reg [6:0] a2g,

output reg [1:0] an

);

reg s = 0; // 位

reg [3:0] digit;

reg isFirst = 1;

always @(s) begin

case(s)

0: digit = data[3:0];

1: digit = data[7:4];

default: digit = data[3:0];

endcase

case(digit)

4'b0000: a2g = 7'b1111110;

4'b0001: a2g = 7'b0110000;

4'b0010: a2g = 7'b1101101;

4'b0011: a2g = 7'b1111001;

4'b0100: a2g = 7'b0110011;

4'b0101: a2g = 7'b1011011;

4'b0110: a2g = 7'b1011111;

4'b0111: a2g = 7'b1110000;

4'b1000: a2g = 7'b1111111;

4'b1001: a2g = 7'b1111011;

4'b1010: a2g = 7'b1110111;

4'b1011: a2g = 7'b0011111;

4'b1100: a2g = 7'b1001110;

4'b1101: a2g = 7'b0111101;

4'b1110: a2g = 7'b1001111;

4'b1111: a2g = 7'b1000111;

default:;

endcase

an = 2'b00;

an[s] = 1;

end

// 1位计数器

always @(posedge clk190 or posedge clr) begin

if (clr) s <= 0;

else s <= s + 1;

end

endmodule

四、仿真程序

sim\_1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/12/03 20:05:06

// Design Name:

// Module Name: sim\_1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module sim\_1(

output [6:0] a2g,

output [1:0] an

);

reg clk, clr;

reg isFirst = 1;

parameter clk\_period = 10;

initial begin

clk = 0;

clr = 1;

forever begin

#(clk\_period / 2) clk = ~clk;

if (isFirst == 1) begin

isFirst = 0;

clr = 0;

end

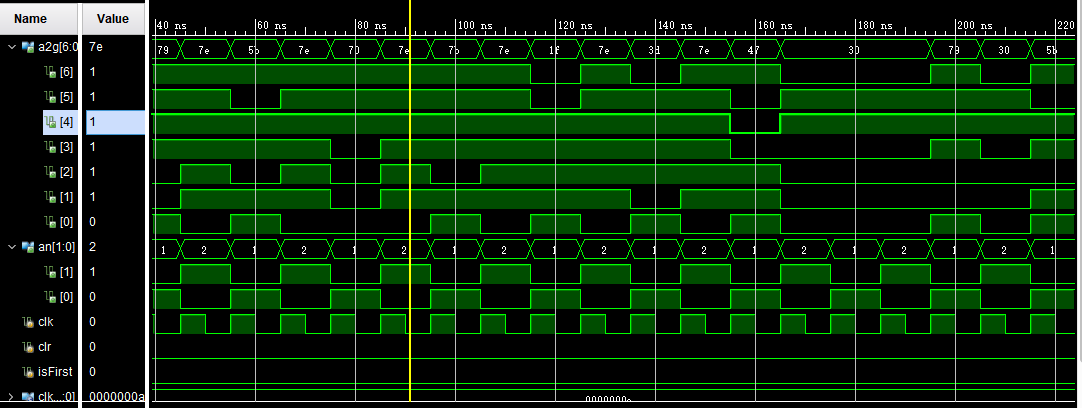
end

end

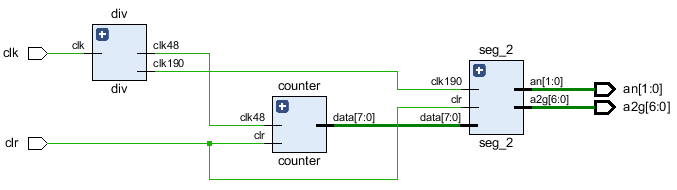
top top(.clk(clk), .clr(clr), .a2g(a2g), .an(an));

endmodule

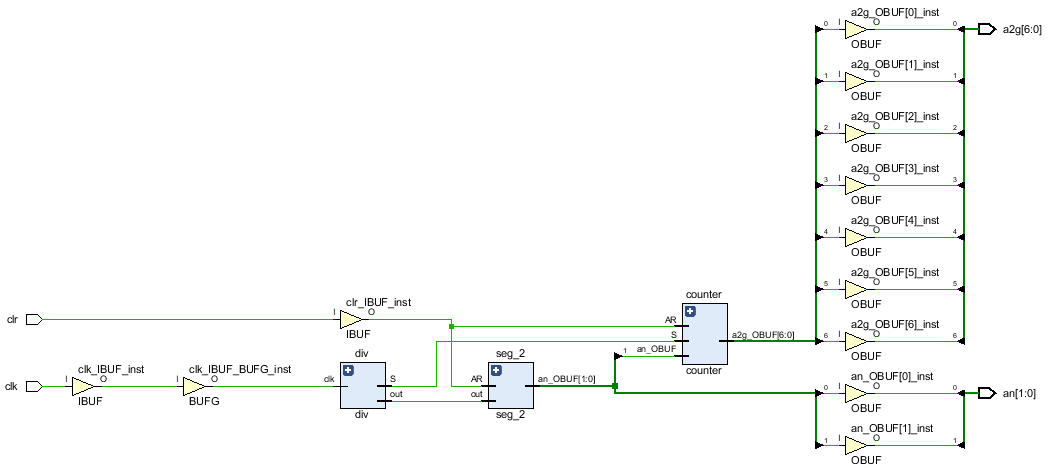
五、仿真结果



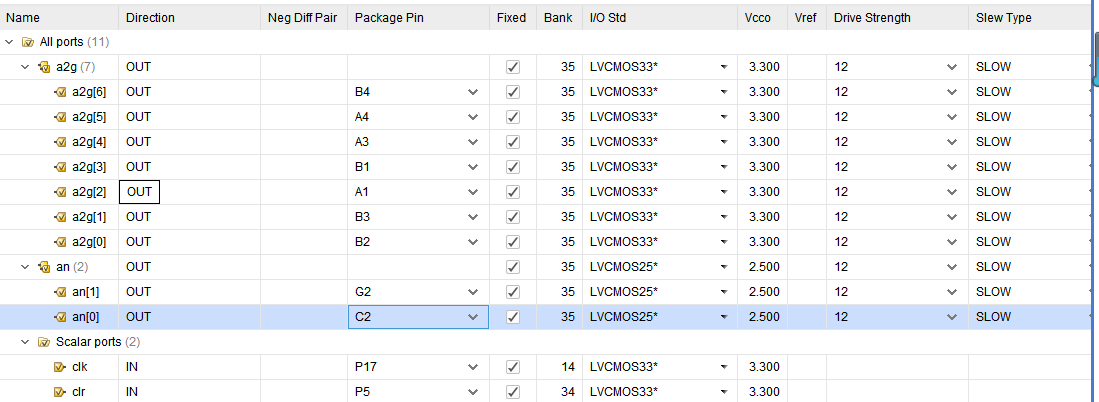
六、系统网表

RTL:

Synthesis：

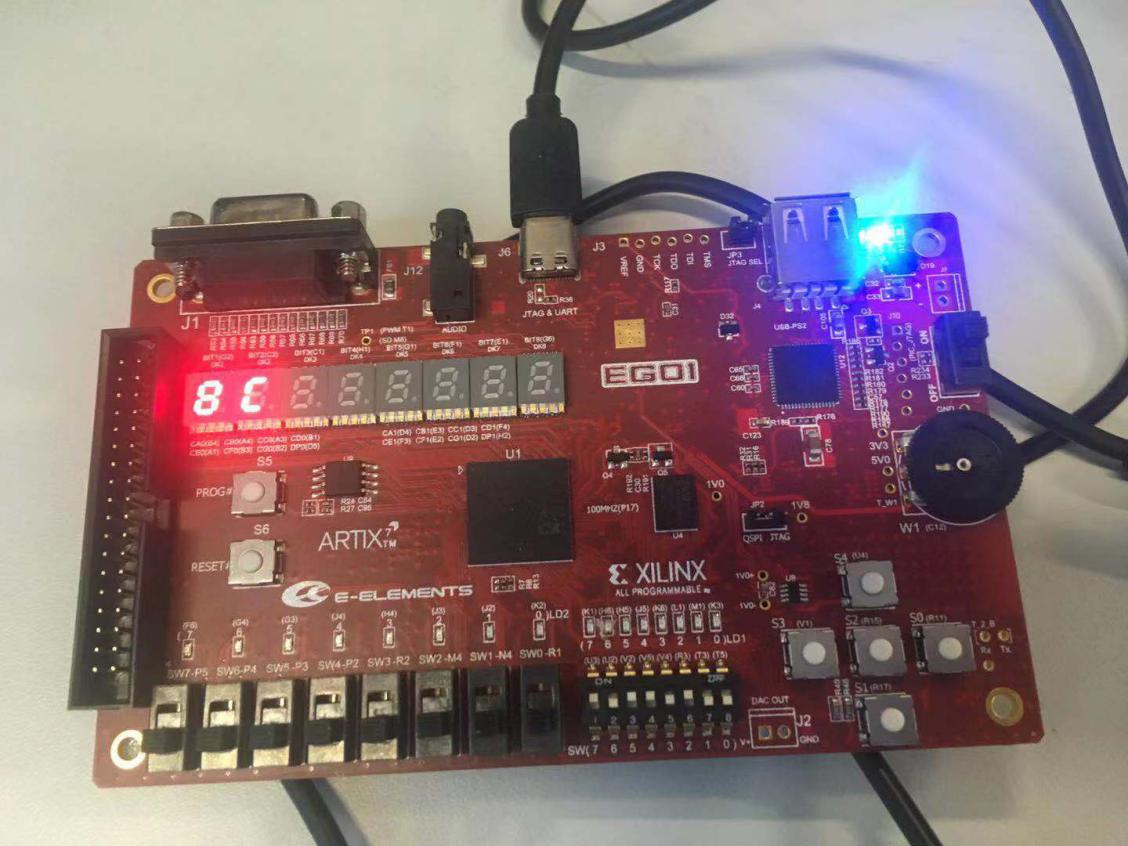


七、系统约束



八、实验结果

当clr=0时，就会按照设定的频率连续地进行累加。实验结果照片，就是累加到8C的情况：



九、实验体会

主要的难点在于需要更加习惯于时序逻辑的设计、以及verilog里多模块的编程；并且发现一些语法上概念的还需要熟练，在编程过程中有所耽搁，但也的确学习了。

**暨南大学本科实验报告专用纸**

课程名称 数字逻辑实验 成绩评定

实验项目名称 移位寄存器 指导教师 王传胜 梁倬骞

实验项目编号 0806003808 实验项目类型 验证 实验地点 N126

学生姓名 邝庆璇 学号 2016051598

学院 信息科学技术 系 计算机科学 专业 计算机科学与技术

实验时间 2019 年 11 月 28 日 上 午

一、实验目的

熟练使用 Verilog HDL 进行行为建模；

学会 Verilog HDL 行为建模来实现移位寄存器；

二、实验内容

先用Verilog HDL实现74LS194的功能，再利用74LS194实现8位数据双向移位寄存器。

三、实验程序

dataShift8.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/12/01 15:05:17

// Design Name:

// Module Name: dataShift8

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module dataShift8(

input notCR,

input clk,

input [1:0] g,

input [1:0] ds,

input [7:0] DATA,

output [7:0] Q

);

wire clk\_out;

clkdiv clkdiv(.clk\_in(clk), .clk(clk\_out));

wire [1:0] dsH, dsL; // 高4位/低4位的ds

assign dsH = {Q[3], ds[1]};

assign dsL = {ds[0], Q[4]};

shift\_194 shiftH(.notCR(notCR), .clk(clk\_out), .s(g), .ds(dsH), .data(DATA[7:4]), .q(Q[7:4]));

shift\_194 shiftL(.notCR(notCR), .clk(clk\_out), .s(g), .ds(dsL), .data(DATA[3:0]), .q(Q[3:0]));

endmodule

clkdiv.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/25 13:34:41

// Design Name:

// Module Name: clkdiv

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module clkdiv(

input clk\_in,

output clk

);

reg [24:0] q = 25'b0;

always @(posedge clk\_in)

begin

q = q + 1;

end

assign clk = q[24];

// assign clk = clk\_in;

endmodule

Shift\_194.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/12/01 14:21:14

// Design Name:

// Module Name: shift\_194

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module shift\_194(

input notCR,

input clk,

input [1:0] s, // 移位方向（高左移低右移）

input [1:0] ds, // 所移入的数字（左移高右移低）

input [3:0] data,

output reg [3:0] q

);

always @(posedge clk or posedge notCR)

begin

if (notCR != 1)

q = 4'b0000;

else

begin

case (s)

2'b00:;

2'b01: q = {ds[0], q[3:1]}; // right

2'b10: q = {q[2:0], ds[1]}; // left

2'b11: q = data;

default: q = 4'b0000;

endcase

end

end

endmodule

四、仿真程序

sim\_1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/12/01 16:50:08

// Design Name:

// Module Name: sim\_1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module sim\_1(

output [7:0] Q

);

reg notCR;

reg clk;

reg [1:0] g;

reg [1:0] ds;

reg [7:0] DATA;

reg isFirst;

parameter clk\_period = 10;

initial

begin

clk = 1'b0;

notCR = 1'b1;

g = 2'b11; // 先送数

ds = 2'b10;

DATA = 8'b00101101;

isFirst = 1'b1;

forever

begin

#(clk\_period / 2) clk = ~clk;

if (isFirst != 0)

begin

isFirst = 1'b0;

g = 2'b10; // 左移

end

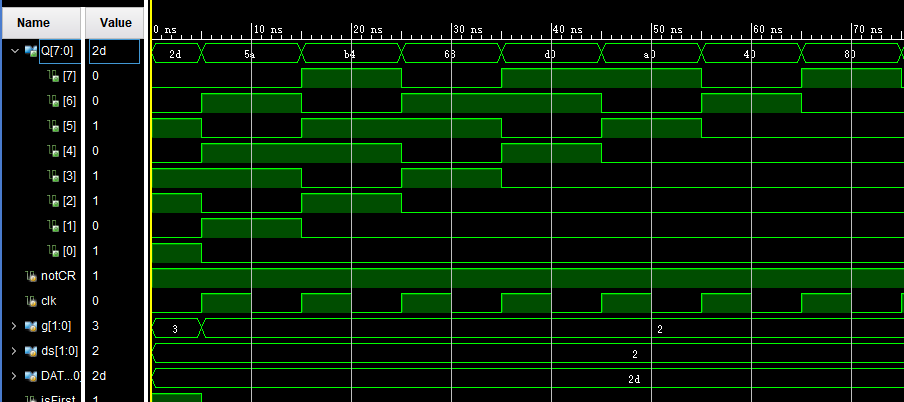
end

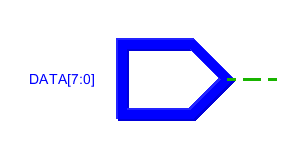
end

dataShift8 dataShift8(.notCR(notCR), .clk(clk), .g(g), .ds(ds), .DATA(DATA), .Q(Q));

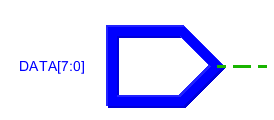
endmodule

五、仿真结果

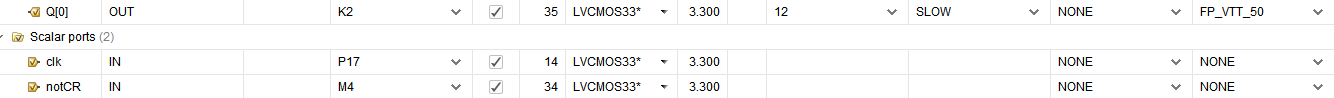
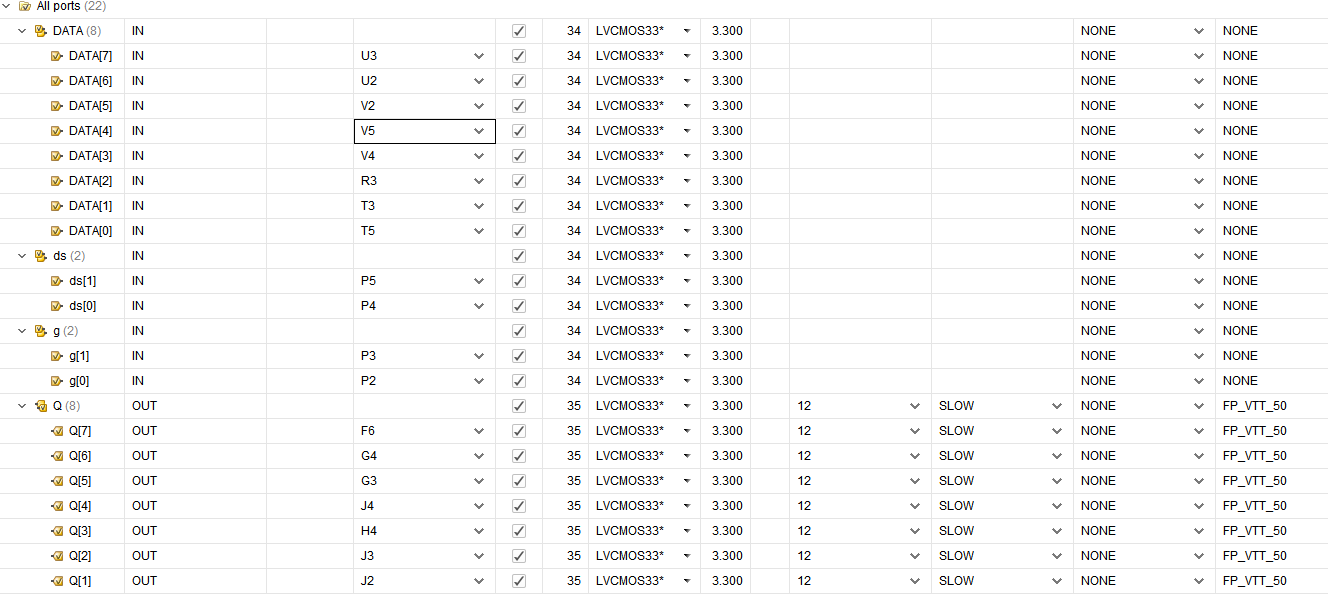


六、系统网表

RTL

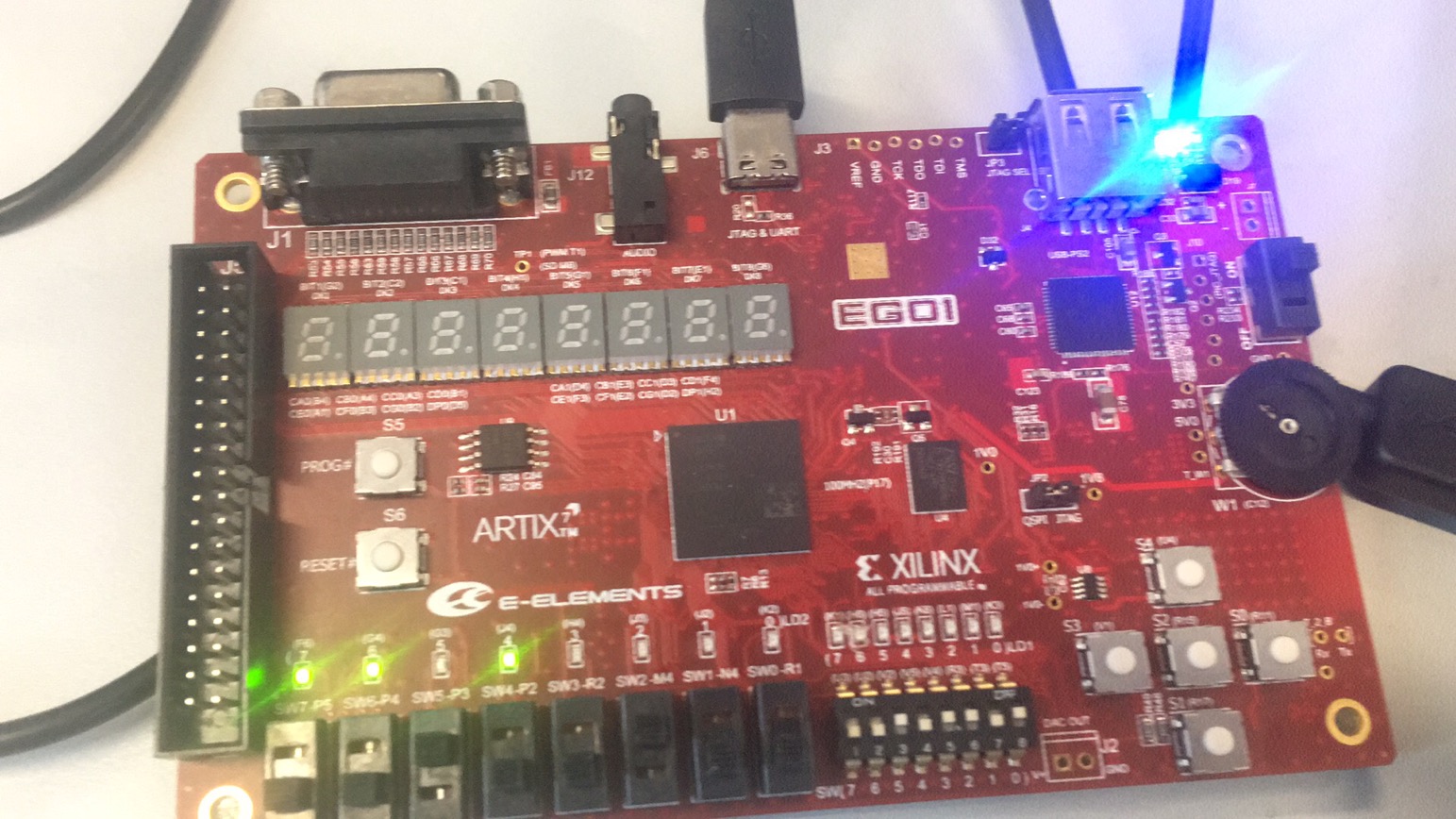
Synthesis

七、系统约束



八、实验结果

比如，把00101101进行左移，循环左移入0，右移到第4步，变成11010000：



九、实验体会

主要的难点在于需要更加习惯于时序逻辑的设计、以及verilog里多模块的编程；并且发现一些语法上概念的还需要熟练，在编程过程中有所耽搁，但也的确学习了。

**暨南大学本科实验报告专用纸**

课程名称 数字逻辑实验 成绩评定

实验名称 时序逻辑电路综合设计（交通信号灯） 指导教师 王传胜 梁倬骞

实验项目编号 0806003808 实验项目类型 验证 实验地点 N126

学生姓名 邝庆璇 学号 2016051598

学院 信息科学技术 系 计算机科学 专业 计算机科学与技术

实验时间 2019 年 12 月 01 日 上 午

一、实验目的

理解有限状态机FSM的原理；

学会Verilog HDL 来实现有限状态机FSM

熟练利用FSM解决问题

二、实验内容

实现交通信号灯。我们要为一个十字路口交通信号灯设计程序。该十字路口分为南北方向和东西方向，每个方向都有红、黄、绿三种颜色信号灯。下图给出了信号灯的状态表：



三、实验程序

top.v

``timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/12/03 17:05:25

// Design Name:

// Module Name: top

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module top(

input clk,

input clr,

output [5:0] lights

);

wire clk\_3;

clk\_div clkdiv(clk, clr, clk\_3);

traffic traffic(clk\_3, clr, lights);

endmodule

clk\_div.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/12/03 17:00:01

// Design Name:

// Module Name: clk\_div

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module clk\_div(

input clk,

input clr,

output clk\_3

);

reg [24:0] q;

always @(posedge clk or posedge clr) begin

if (clr == 1) q <= 0;

else q <= q + 1;

end

// assign clk\_3 = q[24];

assign clk\_3 = clk;

endmodule

traffic.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/12/02 17:09:54

// Design Name:

// Module Name: traffic

// Project Name:

// Target Devices:

// Tool Versions:

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module traffic(

input clk\_3,

input clr,

output reg [5:0] lights

);

reg [1:0] state;

reg [4:0] count;

// 4种状态

parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;

// 3HZ, 则SEC6: 6S, 18个时钟周期; SEC1: 1s, 3个时钟周期

parameter SEC6 = 5'b10001, SEC1 = 5'b00010;

// 高3位：南北方向灯色（高到低每位分别代表绿黄红）；低三位：西东

parameter LRR = 6'b001001, LRY = 6'b001010, LRG = 6'b001100,

LYR = 6'b010001, LYY = 6'b010010, LYG = 6'b010100,

LGR = 6'b100001, LGY = 6'b100010, LGG = 6'b100100;

// set status

always @(posedge clk\_3 or posedge clr) begin

if (clr == 1) begin

state <= S0;

count <= 0;

end

else begin

case (state)

S0:

if (count < SEC6) begin

state <= S0;

count <= count + 1;

end

else begin

state <= S1;

count <= 0;

end

S1:

if (count < SEC1) begin

state <= S1;

count <= count + 1;

end

else begin

state <= S2;

count <= 0;

end

S2:

if (count < SEC6) begin

state <= S2;

count <= count + 1;

end

else begin

state <= S3;

count <= 0;

end

S3:

if (count < SEC1) begin

state <= S3;

count <= count + 1;

end

else begin

state <= S0;

count <= 0;

end

default:

state <= S0;

endcase

end

end

// set lights according to current state

always @(\*) begin

case(state)

S0: lights = LGR;

S1: lights = LYR;

S2: lights = LRG;

S3: lights = LRY;

default: lights = LRR;

endcase

end

endmodule

四、仿真程序

sim\_1.v

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/12/03 17:09:59

// Design Name:

// Module Name: sim\_1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module sim\_1(

output [5:0] lights

);

reg clk, clr;

parameter clk\_period = 10;

initial begin

clr = 1'b0;

clk = 1'b0;

forever

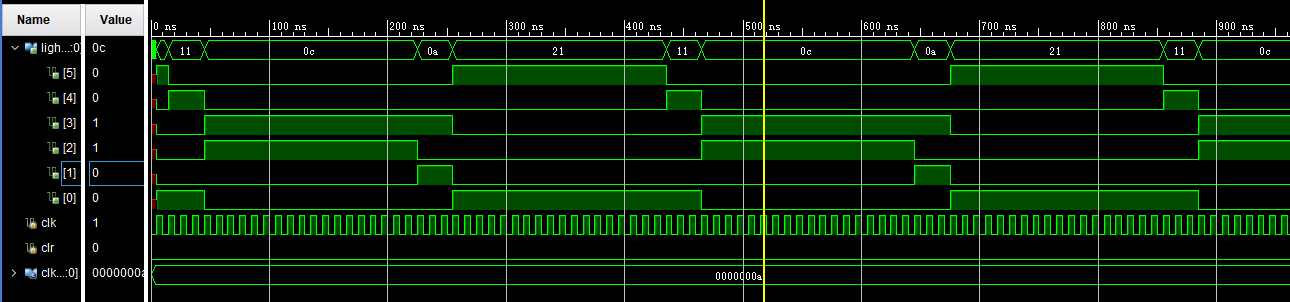
#(clk\_period / 2) clk = ~clk;

end

top top(clk, clr, lights);

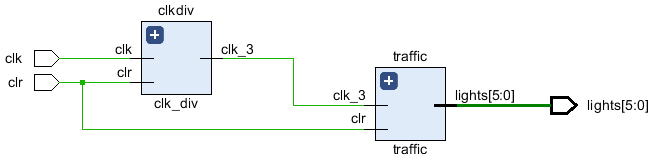
endmodule

五、仿真结果

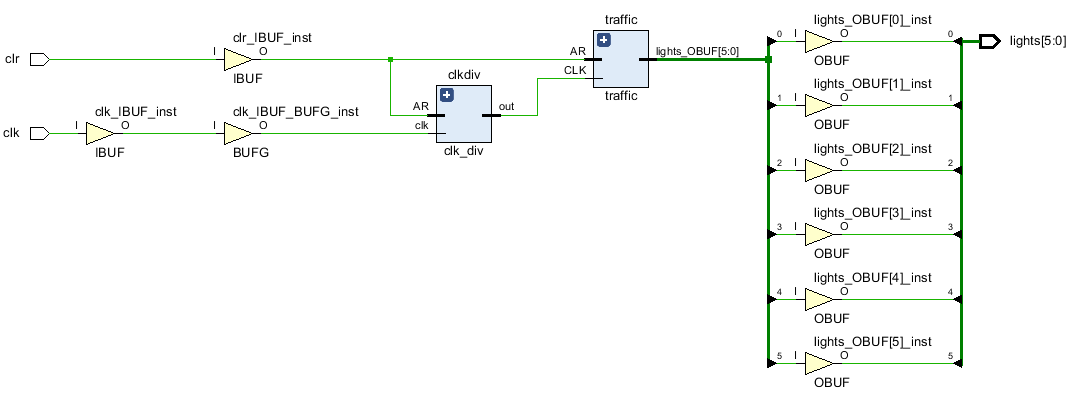


六、系统网表

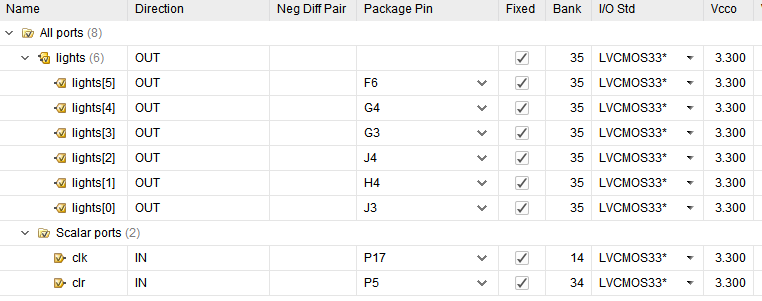
RTL:



Synthesis：

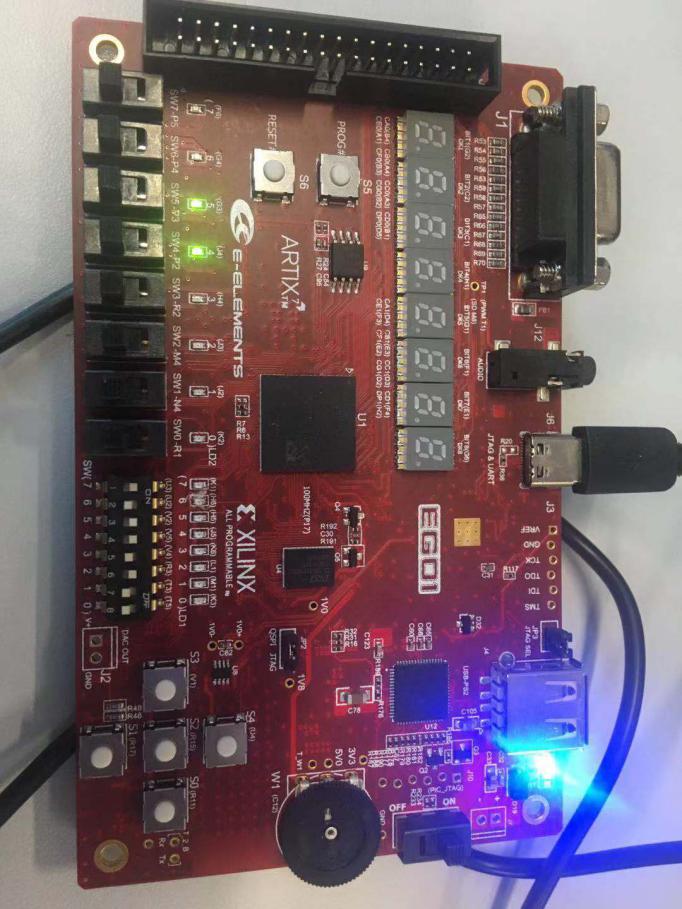


七、系统约束



八、实验结果

按照预期，按照6s或1s的时间间隔轮流着4个状态对应的4灯的情况：LGR, LYR, LRG, LRY。比如，以下就是LRG状态，表示南北为红(R. Red)，东西为绿(G, Green)



九、实验体会

经过前面的若干铺垫，对于时序逻辑已经比较熟悉，因此这次实验总体还算顺利。P.S.：这算是本实验课最后一个实验了，一个学期下来，对Verilog HDL数字逻辑的编程有了较好的学习，收获颇丰。感谢老师的帮助与指点！有缘再会！