**实验2 带控制端的8位运算器设计**

一、实验目的

1. 掌握加法器的设计方法。
2. 掌握测试文件的设计方法。
3. 掌握FPGA技术的层次化设计方法。
4. 掌握FPGA下载测试方法。

二、**实验主要仪器设备**

1. FPGA实验板
2. FPGA实验板配套软件，ModelSim仿真软件

三、设计任务与要求

**1. 基本任务及要求**

（1）用Verilog设计一个带低有效控制端的一位全加器，再利用级联方法构成带低有效控制端的8位加法器。

（2）用Verilog测试文件，实现ModelSim时序仿真。

（3）根据FPGA开发板，配置输入和输出管脚，生成下载文件，实现下载测试。

2. 扩展任务及要求

（1）用Verilog设计带低有效控制端的4位并行进位加法器，再利用层次设计方法构成带低有效控制端的8位并行加法器。

（2）用Verilog设计加减运算器，通过控制端完成8位加法和8位减法的转换。

（3）用Verilog测试文件，实现ModelSim时序仿真。

（4）根据FPGA开发板，配置输入和输出管脚，生成下载文件，实现下载测试。

四、实验内容与步骤

**1. 基本任务**

**(1) 带低有效控制端的1位全加器**

(a) 工作原理

首先写出带有控制端1位全加器的真值表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **en\_n** | **A** | **B** | **C** | **S** | **Cout** |
| 1 | x | x | x | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |

表 1 低有效控制端1位全加器真值表

其中en\_n为低有效使能端，其值为高电平时，S,Cout始终输出低电平。接下来可以根据*表 1*，使用verilog写出对应的程序。

(b) Verilog源程序

module full\_adder(

input A,

input B,

input C,

input en\_n,

output reg S,

output reg Cout

);

always @ (A,B,C,en\_n) begin

if (en\_n) {S,Cout} = 2'b00;

else begin

case ({A,B,C})

3'b000: {S,Cout} = 2'b00;

3'b001: {S,Cout} = 2'b10;

3'b010: {S,Cout} = 2'b10;

3'b011: {S,Cout} = 2'b01;

3'b100: {S,Cout} = 2'b10;

3'b101: {S,Cout} = 2'b01;

3'b110: {S,Cout} = 2'b01;

3'b111: {S,Cout} = 2'b11;

endcase

end

end

endmodule

(c) RTL视图

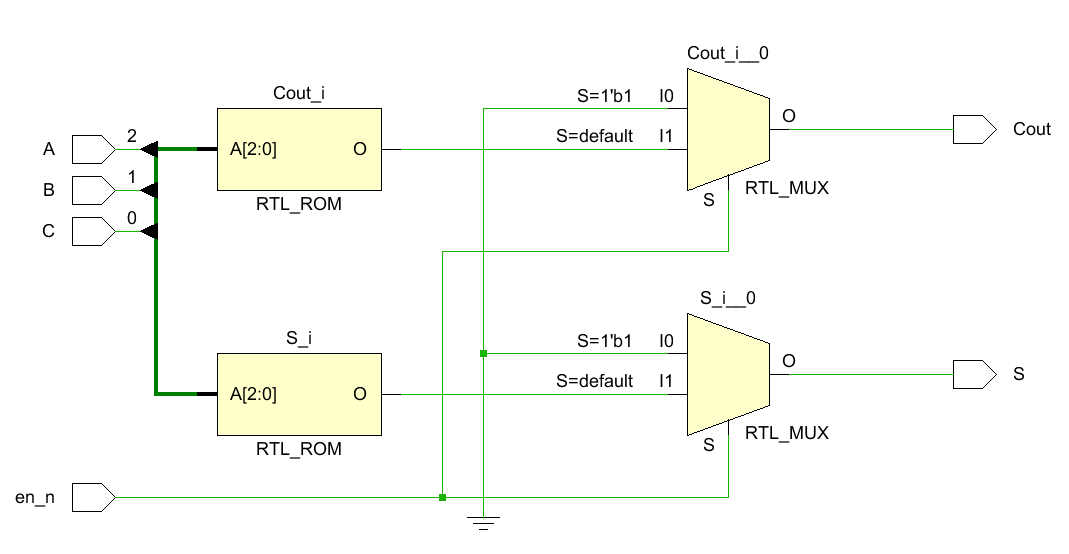


图 1 低有效控制端1位全加器RTL视图

*图 1*中有两个查找表模块（储存真值表的内存），分别对应真值表中Cout和S的值；还有两个多路选择器，用于使能端en\_n的控制。

(d) ModelSim源程序

`timescale 1ns / 100ps

module full\_adder\_tb;

reg A;

reg B;

reg C;

reg en\_n; // 低有效使能端

wire S;

wire Cout; // 进位输出

full\_adder uut (

.A(A),

.B(B),

.C(C),

.en\_n(en\_n),

.S(S),

.Cout(Cout)

);

initial begin

A = 0;

B = 0;

C = 0;

en\_n = 1; // 初始使能无效

// 测试使能功能

#10 en\_n = 0; // 使能有效

// 测试所有可能的输入组合

#10 A=0; B=0; C=0;

#10 A=0; B=0; C=1;

#10 A=0; B=1; C=0;

#10 A=0; B=1; C=1;

#10 A=1; B=0; C=0;

#10 A=1; B=0; C=1;

#10 A=1; B=1; C=0;

#10 A=1; B=1; C=1;

// 再次测试使能无效的情况

#10 en\_n = 1;

#10 A=1; B=1; C=1; // 输出应为0

// 结束仿真

#10 $finish;

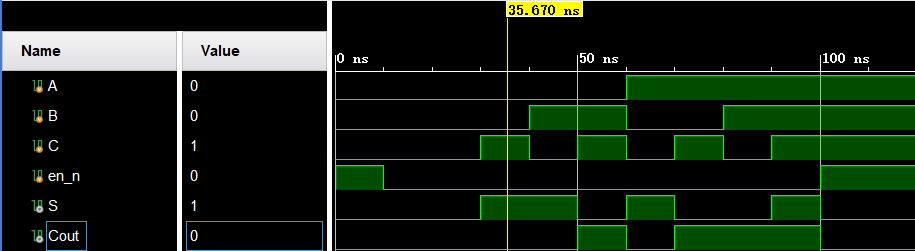
end

endmodule

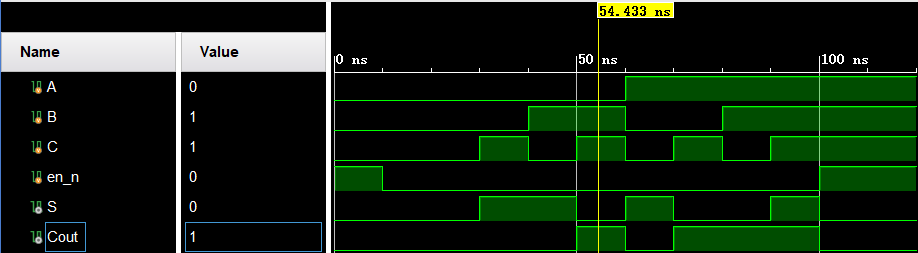
(d) ModelSim仿真结果

在使能输入en\_n（低有效）为低电平时：

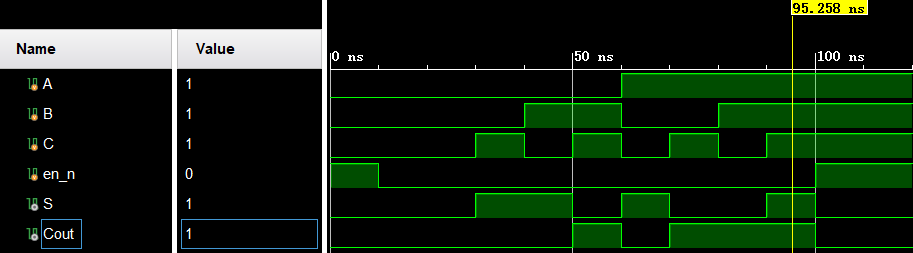
低位进位C为1时，本位和S为1，向高位进位Cout为0：



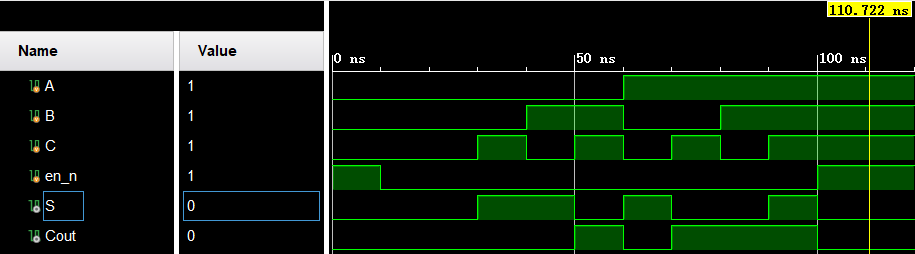
输入B和低位进位C为1，本位和为0，产生进位，Cout为1：



输入A,B为 1，产生进位1，Cout为1，同时加上低位进位C，本位和S为1：



当使能端en\_n（低有效）为1时，输出S，Cout均为0：



(e) 下载测试结果

管脚配置

#DIP\_SW:

set\_property -dict {PACKAGE\_PIN T3 IOSTANDARD LVCMOS33} [get\_ports B] ;#B

set\_property -dict {PACKAGE\_PIN J3 IOSTANDARD LVCMOS33} [get\_ports A] ;#A

#LEDS

set\_property -dict {PACKAGE\_PIN B24 IOSTANDARD LVCMOS33} [get\_ports S]

set\_property -dict {PACKAGE\_PIN B20 IOSTANDARD LVCMOS33} [get\_ports Cout]

#Touch Button

set\_property -dict {PACKAGE\_PIN T2 IOSTANDARD LVCMOS33} [get\_ports en\_n] ;#BTN1

set\_property -dict {PACKAGE\_PIN M1 IOSTANDARD LVCMOS33} [get\_ports C] ;#BTN2

set\_property CFGBVS VCCO [current\_design]

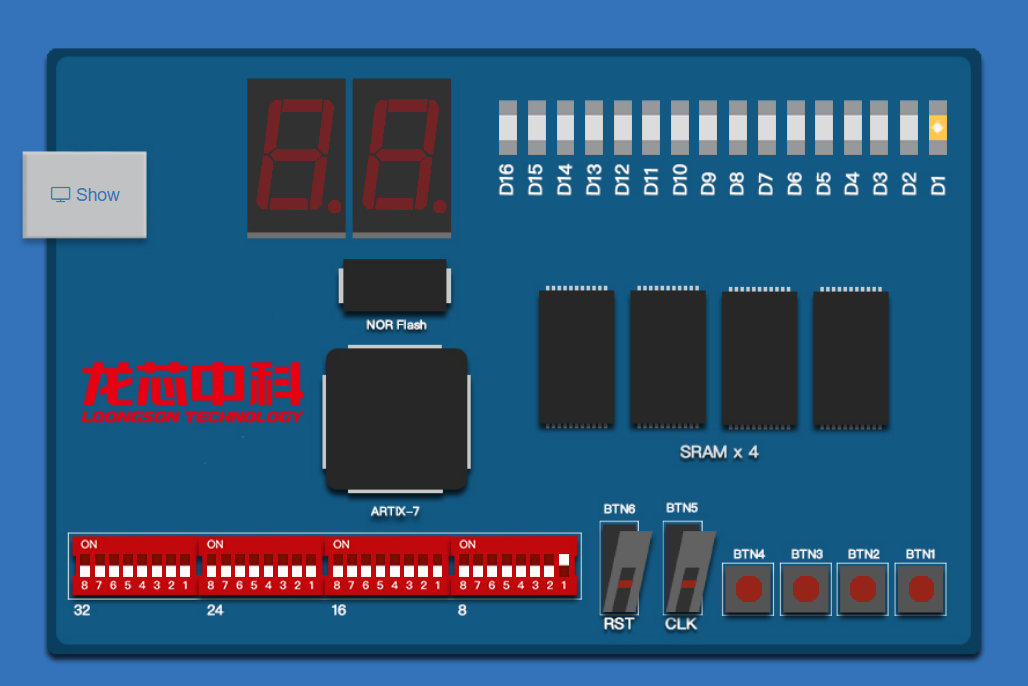
set\_property CONFIG\_VOLTAGE 3.3 [current\_design]

set\_property BITSTREAM.GENERAL.COMPRESS TRUE [current\_design]

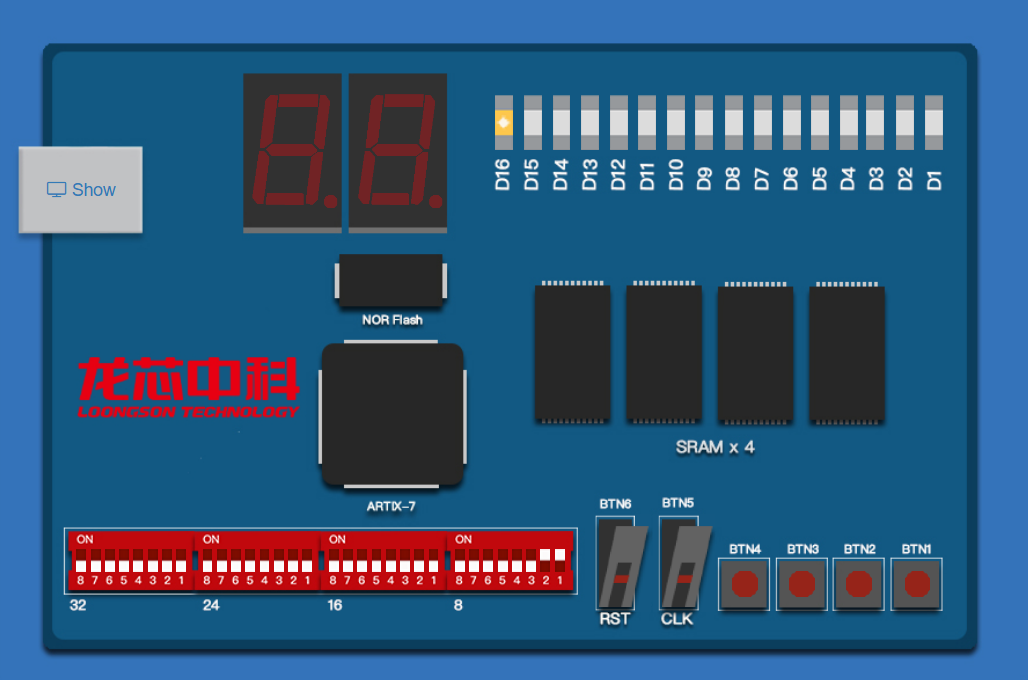
测试图

使能en\_n（低有效）为0：

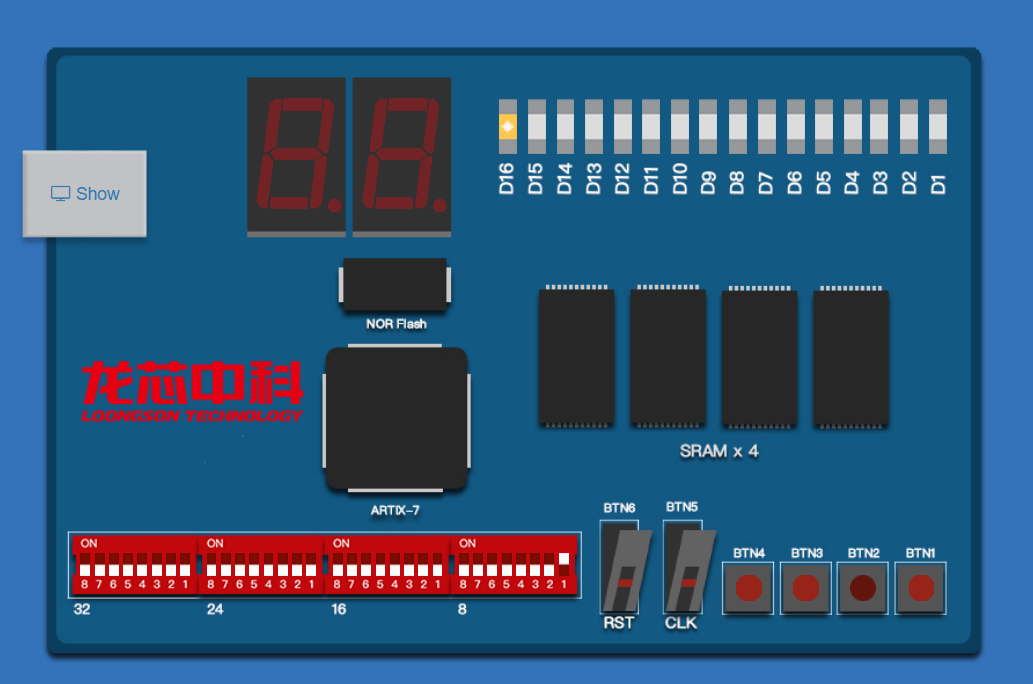
输入A为0，低位进位C为0，B为1，本位和S为1，Cout为0：



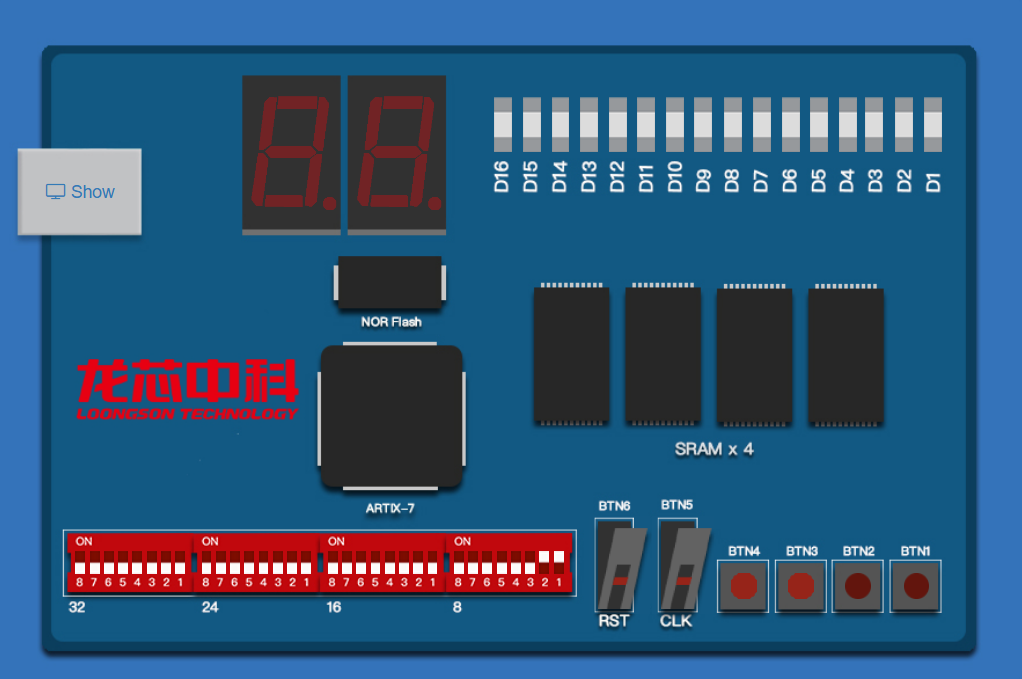
输入A，B均为1，低位进位C为0，本位和S为0，向高位进位Cout为1：



输入A为0，低位进位C为1，B为1，本位和S为0，Cout为1：



使能输入en\_n（低有效）为1，所有输出均为0：



**(2) 由1位全加器级联8位加法器**

(a) 工作原理

8位加法器就是将8个全加器按照位级进行级联。假设加法器的两个输入为8位的A和B，分别表示加数A和加数B（每个是8位二进制数），Cin是输入的初始进位（通常为0）。

第一位（最低位）加法：将A[0]和B[0]与Cin（初始为0）一起输入到第一个全加器中。第一个全加器产生S[0]（最低位的和）和Cout\_0（第一位的进位）。

后续位的加法：从第二位到第七位，每个全加器接收前一个全加器的进位作为输入，同时接收当前位A[i]和B[i]的输入。每个全加器的输出会生成当前位的和S[i]，并将进位输出Cout\_i传递到下一位的全加器。

第八位（最高位）加法：在最后一位，全加器将A[7]、B[7]和来自第七位的进位输入进行加法，最终生成S[7]和最后的进位Cout（即最终的进位输出）。

要实现数码管显示十进制十位、个位数字，首先要使用**双倍加三法**把8位二进制输出转换成2个十位、个位BCD码，然后接入数码管。并把所有功能封装成一个模块以便在开发板中操作。

(b) Verilog源程序

// adder8.v

module adder8(

input [7:0] A,

input [7:0] B,

input Cin,

input en\_n,

output [7:0] S,

output Cout

);

wire [6:0] carry; // 中间进位线

// 第0位例化

full\_adder u0 (

.A(A[0]),

.B(B[0]),

.C(Cin),

.en\_n(en\_n),

.S(S[0]),

.Cout(carry[0])

);

// 第1~6位例化

genvar i;

generate

for (i = 1; i < 7; i = i + 1) begin : FA\_GEN

full\_adder u (

.A(A[i]),

.B(B[i]),

.C(carry[i-1]),

.en\_n(en\_n),

.S(S[i]),

.Cout(carry[i])

);

end

endgenerate

// 第7位例化

full\_adder u7 (

.A(A[7]),

.B(B[7]),

.C(carry[6]),

.en\_n(en\_n),

.S(S[7]),

.Cout(Cout)

);

endmodule

// binary2bcd.v

module binary2bcd(

input [7:0] binary, // 8位二进制输入（0-255）

output reg [3:0] tens, // 十位BCD输出

output reg [3:0] ones // 个位BCD输出

);

integer i;

reg [11:0] bcd; // 12位宽，可以存储0-255的BCD码

always @(\*) begin

bcd = 12'b0; // 初始化BCD寄存器

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

// 如果BCD的每一位大于等于5，加3

if (bcd[3:0] >= 5) bcd[3:0] = bcd[3:0] + 3;

if (bcd[7:4] >= 5) bcd[7:4] = bcd[7:4] + 3;

if (bcd[11:8] >= 5) bcd[11:8] = bcd[11:8] + 3;

// 左移一位

bcd = bcd << 1;

bcd[0] = binary[i]; // 将输入的二进制位移入

end

// 提取十位和个位的BCD值

tens = bcd[7:4];

ones = bcd[3:0];

end

endmodule

// dpy.v

module dpy(

input [3:0] bcd,

output reg [6:0] seg

);

always @(\*) begin

case (bcd)//TODO 确定引脚对应位置

4'b0000: seg = 7'b1111110; // 0

4'b0001: seg = 7'b0110000; // 1

4'b0010: seg = 7'b1101101; // 2

4'b0011: seg = 7'b1111001; // 3

4'b0100: seg = 7'b0110011; // 4

4'b0101: seg = 7'b1011011; // 5

4'b0110: seg = 7'b1011111; // 6

4'b0111: seg = 7'b1110000; // 7

4'b1000: seg = 7'b1111111; // 8

4'b1001: seg = 7'b1110011; // 9

default: seg = 7'b0000000; // 熄灭

endcase

end

endmodule

// top.v

module top(

input [7:0] A,

input [7:0] B,

input Cin,

input en\_n,

output [6:0] seg\_tens, // 十位数码管

output [6:0] seg\_ones, // 个位数码管

output [7:0] S,

output dp\_ones, // 小数点

output dp\_tens, // 小数点

output Cout

);

wire [3:0] tens, ones;

// 实例化8位加法器

adder8 u\_adder(

.A(A),

.B(B),

.Cin(Cin),

.en\_n(en\_n),

.S(S),

.Cout(Cout)

);

// 实例化二进制到BCD转换器

binary2bcd u\_bcd(

.binary(S),

.tens(tens),

.ones(ones)

);

// 实例化十位数码管解码器

dpy u\_tens\_decoder(

.bcd(tens),

.seg(seg\_tens)

);

// 实例化个位数码管解码器

dpy u\_ones\_decoder(

.bcd(ones),

.seg(seg\_ones)

);

// 关闭小数点

assign dp\_tens = 1'b0;

assign dp\_ones = 1'b0;

endmodule

(c) RTL视图

将8个上述1位全加器级联成一个8位加法器，其RTL视图如*图 2*所示：

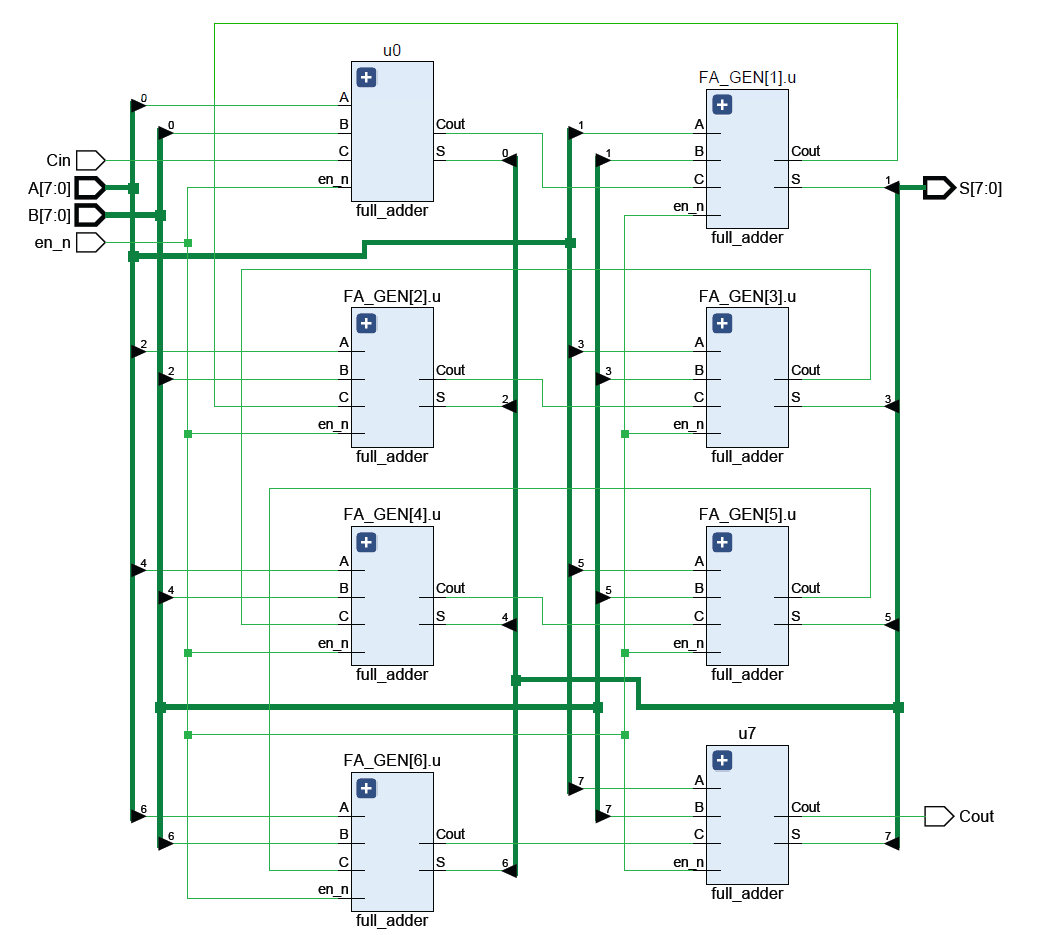


图 2 级联方法构成的带低有效控制端8位加法器RTL视图

(d) ModelSim源程序

`timescale 1ns / 100ps

module adder8\_tb;

// 输入信号

reg [7:0] A, B;

reg Cin;

reg en\_n;

// 输出信号

wire [7:0] S;

wire Cout;

// 实例化被测模块

adder8 uut (

.A(A),

.B(B),

.Cin(Cin),

.en\_n(en\_n),

.S(S),

.Cout(Cout)

);

initial begin

// 初始化

A = 8'b00000000; B = 8'b00000000; Cin = 0; en\_n = 1; // 禁用

#10;

// case 1: en\_n=1（禁用加法器）

A = 8'b00000001; // 1

B = 8'b00000010; // 2

Cin = 1;

#10;

// case 2: en\_n=0（启用），无进位

en\_n = 0;

A = 8'b00000001; // 1

B = 8'b00000010; // 2

Cin = 0;

#10;

// case 3: en\_n=0，有进位

A = 8'b11111111; // 255

B = 8'b00000001; // 1

Cin = 0;

#10;

// case 4: 全为1，有初始进位

A = 8'b11111111; // 255

B = 8'b11111111; // 255

Cin = 1;

#10;

// case 5: 随机测试

A = 8'd60; // 60 -> 00111100

B = 8'd167; // 167 -> 10100111

Cin = 1;

#10;

// case 6: 改变使能，重新禁用

en\_n = 1;

A = 8'd255; // 11111111

B = 8'd1; // 00000001

Cin = 1;

#10;

$finish;

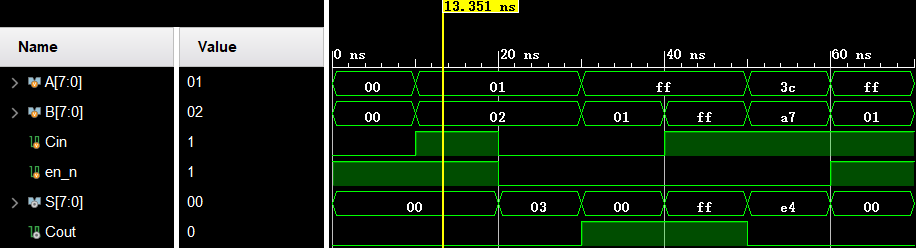
end

endmodule

(d) ModelSim仿真结果

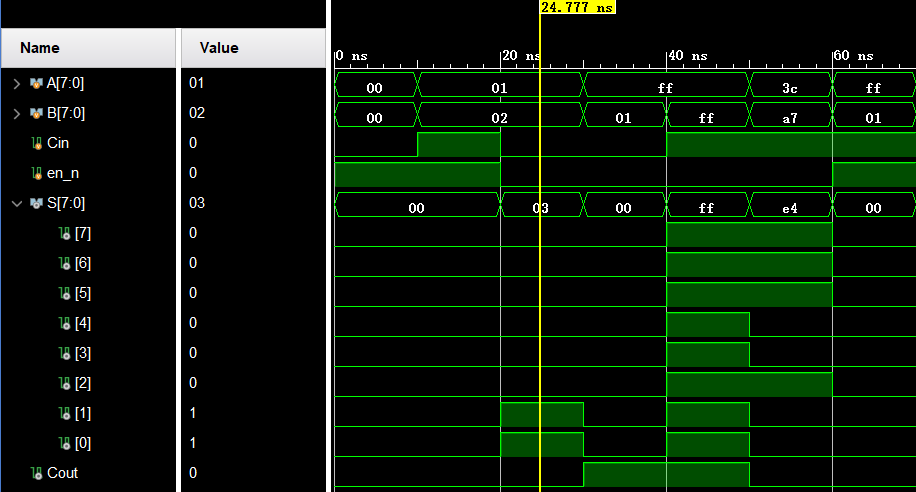
在使能端en\_n（低有效）为1时，加法器输出全低：

当输入A为1（8’b00000001），B为2（8’b00000010）时，本位和S和向高位进位Cout为0：



在使能端en\_n（低有效）为0时：

当输入A为1（8’b00000001），B为2（8’b00000010）时，本位和S为3（0’b0000011） :



(e) 下载测试结果

管脚配置

用拨码开关表示两个输入A，B，BTN1表示en\_n（低有效）使能输入，BTN2表示初识进位Cin

#DIP\_SW B：

set\_property -dict {PACKAGE\_PIN T3 IOSTANDARD LVCMOS33} [get\_ports {B[0]}]

set\_property -dict {PACKAGE\_PIN J3 IOSTANDARD LVCMOS33} [get\_ports {B[1]}]

set\_property -dict {PACKAGE\_PIN M2 IOSTANDARD LVCMOS33} [get\_ports {B[2]}]

set\_property -dict {PACKAGE\_PIN P1 IOSTANDARD LVCMOS33} [get\_ports {B[3]}]

set\_property -dict {PACKAGE\_PIN P4 IOSTANDARD LVCMOS33} [get\_ports {B[4]}]

set\_property -dict {PACKAGE\_PIN L5 IOSTANDARD LVCMOS33} [get\_ports {B[5]}]

set\_property -dict {PACKAGE\_PIN L3 IOSTANDARD LVCMOS33} [get\_ports {B[6]}]

set\_property -dict {PACKAGE\_PIN N6 IOSTANDARD LVCMOS33} [get\_ports {B[7]}]

#DIP\_SW A：

set\_property -dict {PACKAGE\_PIN M6 IOSTANDARD LVCMOS33} [get\_ports {A[0]}]

set\_property -dict {PACKAGE\_PIN N7 IOSTANDARD LVCMOS33} [get\_ports {A[1]}]

set\_property -dict {PACKAGE\_PIN M7 IOSTANDARD LVCMOS33} [get\_ports {A[2]}]

set\_property -dict {PACKAGE\_PIN L7 IOSTANDARD LVCMOS33} [get\_ports {A[3]}]

set\_property -dict {PACKAGE\_PIN M5 IOSTANDARD LVCMOS33} [get\_ports {A[4]}]

set\_property -dict {PACKAGE\_PIN K3 IOSTANDARD LVCMOS33} [get\_ports {A[5]}]

set\_property -dict {PACKAGE\_PIN J1 IOSTANDARD LVCMOS33} [get\_ports {A[6]}]

set\_property -dict {PACKAGE\_PIN L2 IOSTANDARD LVCMOS33} [get\_ports {A[7]}]

#LEDS

set\_property -dict {PACKAGE\_PIN B24 IOSTANDARD LVCMOS33} [get\_ports {S[0]}]

set\_property -dict {PACKAGE\_PIN E21 IOSTANDARD LVCMOS33} [get\_ports {S[1]}]

set\_property -dict {PACKAGE\_PIN A24 IOSTANDARD LVCMOS33} [get\_ports {S[2]}]

set\_property -dict {PACKAGE\_PIN D23 IOSTANDARD LVCMOS33} [get\_ports {S[3]}]

set\_property -dict {PACKAGE\_PIN C22 IOSTANDARD LVCMOS33} [get\_ports {S[4]}]

set\_property -dict {PACKAGE\_PIN C21 IOSTANDARD LVCMOS33} [get\_ports {S[5]}]

set\_property -dict {PACKAGE\_PIN E20 IOSTANDARD LVCMOS33} [get\_ports {S[6]}]

set\_property -dict {PACKAGE\_PIN B22 IOSTANDARD LVCMOS33} [get\_ports {S[7]}]

set\_property -dict {PACKAGE\_PIN B20 IOSTANDARD LVCMOS33} [get\_ports Cout]

#Touch Button

set\_property -dict {PACKAGE\_PIN T2 IOSTANDARD LVCMOS33} [get\_ports en\_n] ;#BTN1

set\_property -dict {PACKAGE\_PIN M1 IOSTANDARD LVCMOS33} [get\_ports Cin] ;#BTN2

set\_property CFGBVS VCCO [current\_design]

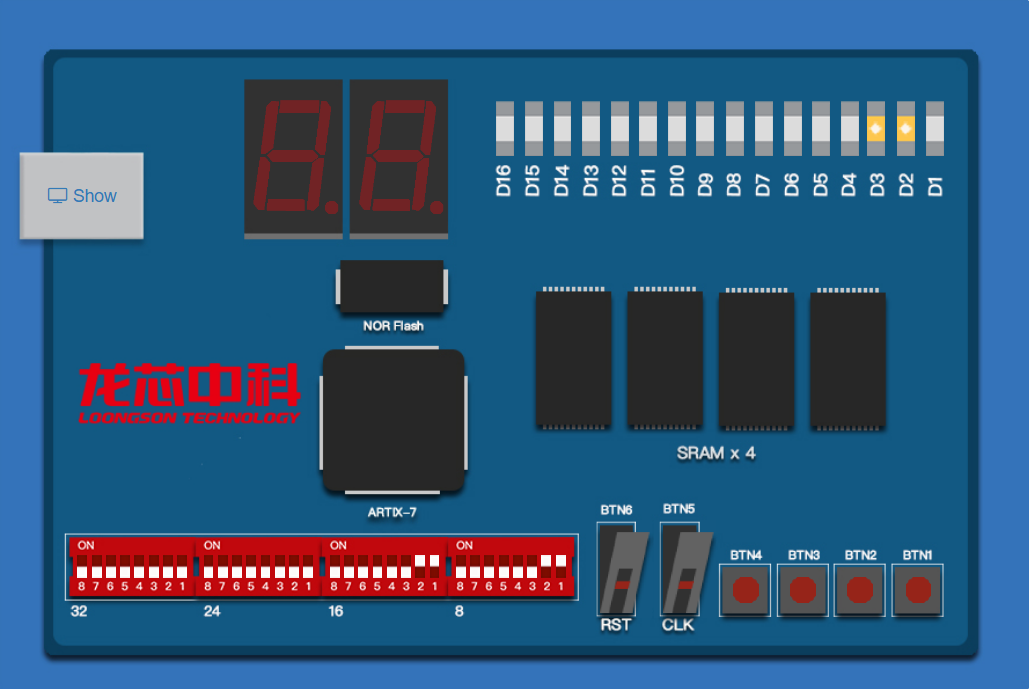
set\_property CONFIG\_VOLTAGE 3.3 [current\_design]

set\_property BITSTREAM.GENERAL.COMPRESS TRUE [current\_design]

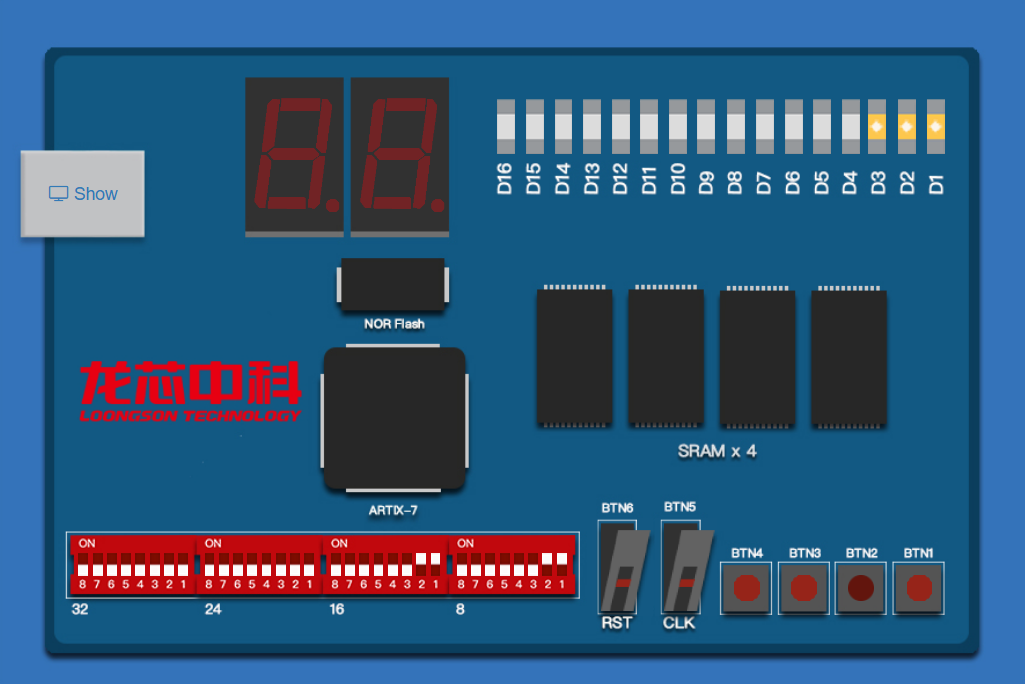
测试图**（数码管在最下方展示！）**

使能输入en\_n为0：

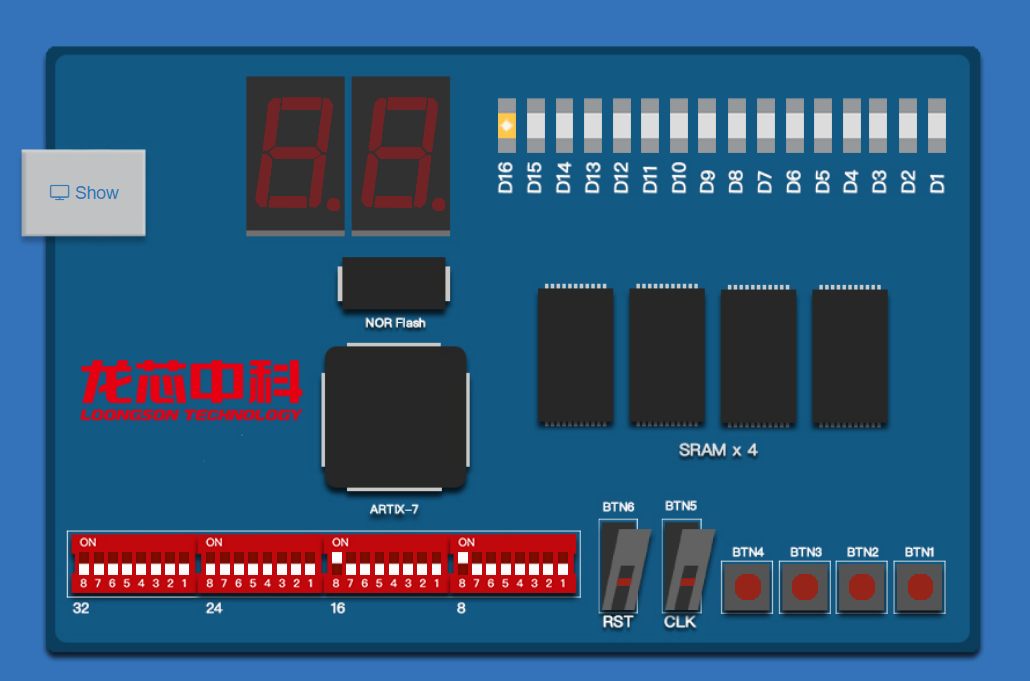
输入A，B为3（0b11），低位进位C为0，输出S为6（0b110）:



输入A，B为3（0b11），低位进位C为1，输出S为7（0b111）:

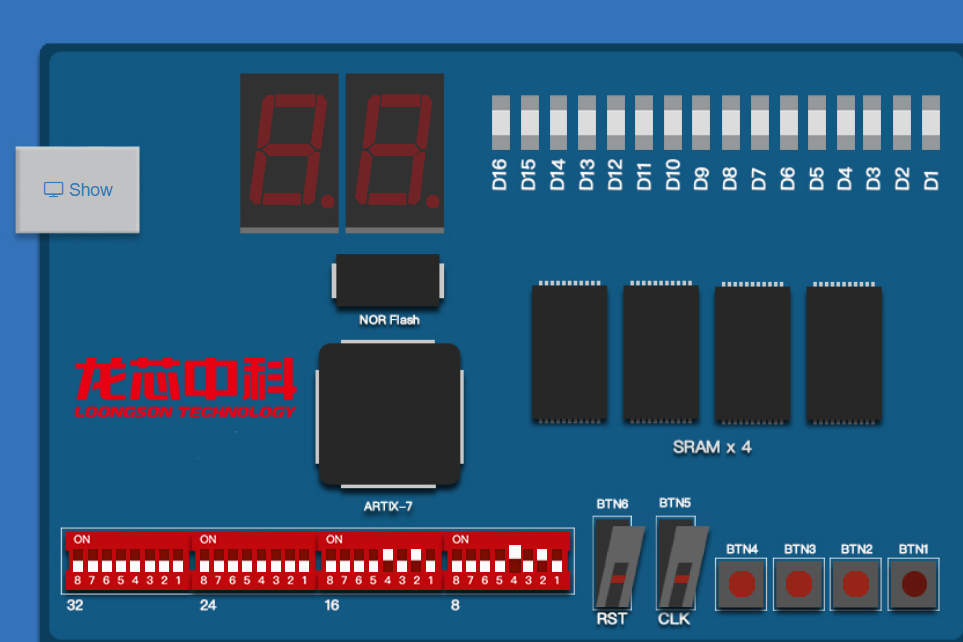


输入A，B均为128（8'b10000000）,会产生进位Cout为1：



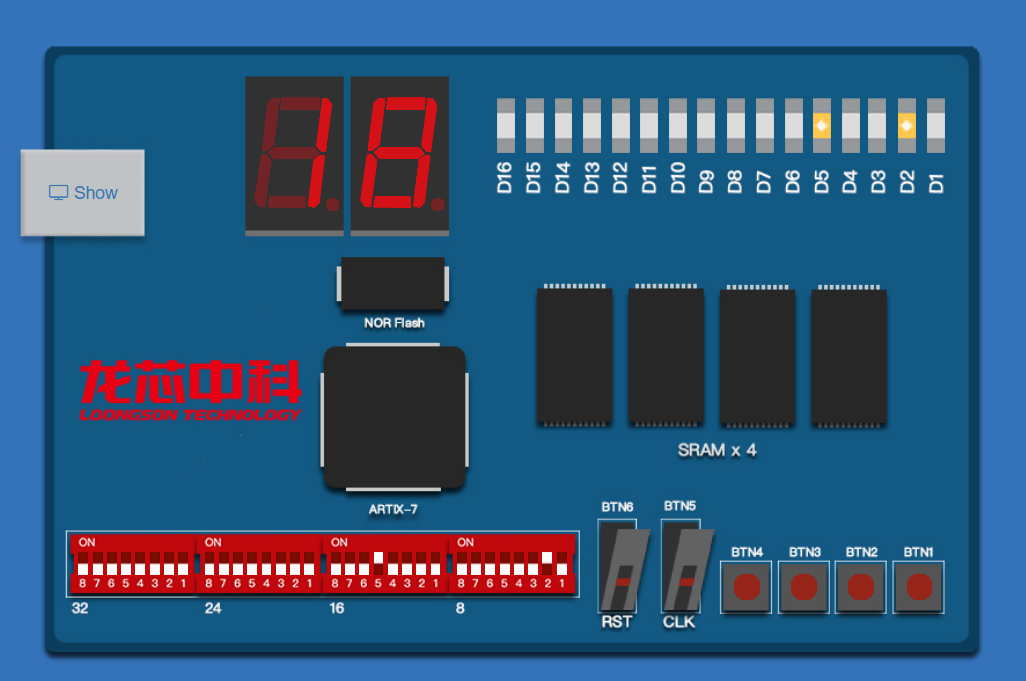
使能输入en\_n（低有效）为1：

所有输出均为0：



**加上数码管显示功能**：

显示了16 + 2 = 18

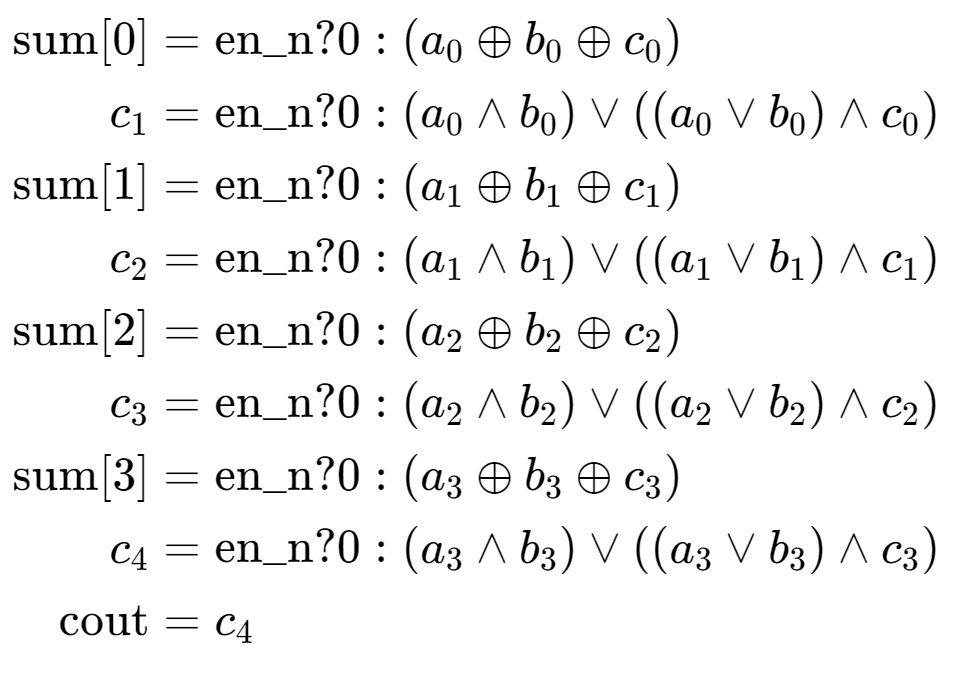


**2. 扩展任务**

（1）

(a) 工作原理

4位并行加法器逻辑函数：



(b) Verilog源程序

module sync\_adder4(

input [3:0] a,

input [3:0] b,

input cin,

input en\_n, // 低有效控制端

output [3:0] sum,

output cout

);

// 内部进位信号

wire [4:0] c;

// 当en\_n为低电平时，加法器工作；否则输出全0

assign c[0] = en\_n ? 1'b0 : cin;

// 并行进位逻辑

assign sum[0] = en\_n ? 1'b0 : (a[0] ^ b[0] ^ c[0]);

assign c[1] = en\_n ? 1'b0 : ((a[0] & b[0]) | ((a[0] | b[0]) & c[0]));

assign sum[1] = en\_n ? 1'b0 : (a[1] ^ b[1] ^ c[1]);

assign c[2] = en\_n ? 1'b0 : ((a[1] & b[1]) | ((a[1] | b[1]) & c[1]));

assign sum[2] = en\_n ? 1'b0 : (a[2] ^ b[2] ^ c[2]);

assign c[3] = en\_n ? 1'b0 : ((a[2] & b[2]) | ((a[2] | b[2]) & c[2]));

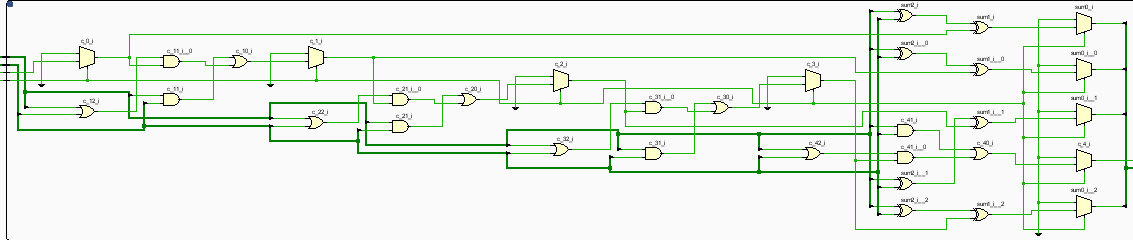
assign sum[3] = en\_n ? 1'b0 : (a[3] ^ b[3] ^ c[3]);

assign c[4] = en\_n ? 1'b0 : ((a[3] & b[3]) | ((a[3] | b[3]) & c[3]));

assign cout = c[4];

endmodule

(c) RTL视图



（2）

(a) 工作原理

输入：两个8位数 a, b，一个1位操作码 op（控制加减），一个进位输入 cin。但在设计中我们通过 op 控制进位自动为1或0，因此外部 cin 实际不需要了，也可以直接移除。而减法用加上他的补码实现。

(b) Verilog源程序

module adder8\_sync(

input [7:0] a, // 8位输入a

input [7:0] b, // 8位输入b

input op, // 进位输入(op = cin；0:加法，1:减法)

input en\_n, // 低有效控制端

output [7:0] sum, // 8位和输出

output cout // 进位输出

);

wire [7:0] b\_mux;

wire [3:0] sum\_low, sum\_high;

wire carry\_low, carry\_out;

// 取反b（op=1时）实现减法

assign b\_mux = op ? ~b : b;

// 第一个4位加法器（低4位）

sync\_adder4 u\_adder\_low (

.a(a[3:0]),

.b(b\_mux[3:0]),

.cin(op), // 加法cin=0，减法cin=1

.en\_n(en\_n),

.sum(sum[3:0]),

.cout(carry\_low)

);

// 第二个4位加法器（高4位）

sync\_adder4 u\_adder\_high (

.a(a[7:4]),

.b(b\_mux[7:4]),

.cin(carry\_low),

.en\_n(en\_n),

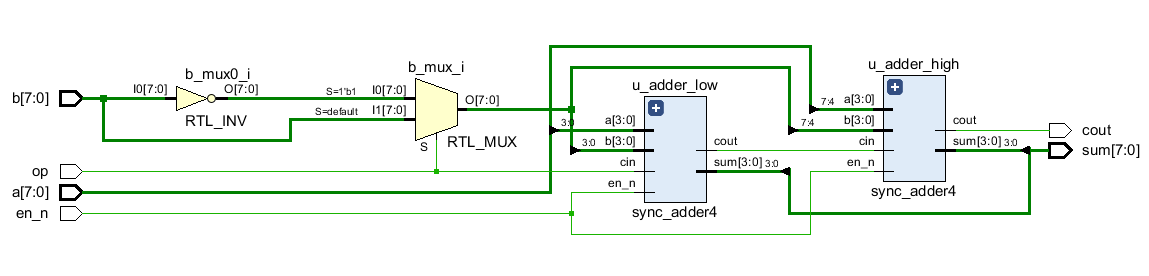
.sum(sum[7:4]),

.cout(cout)

);

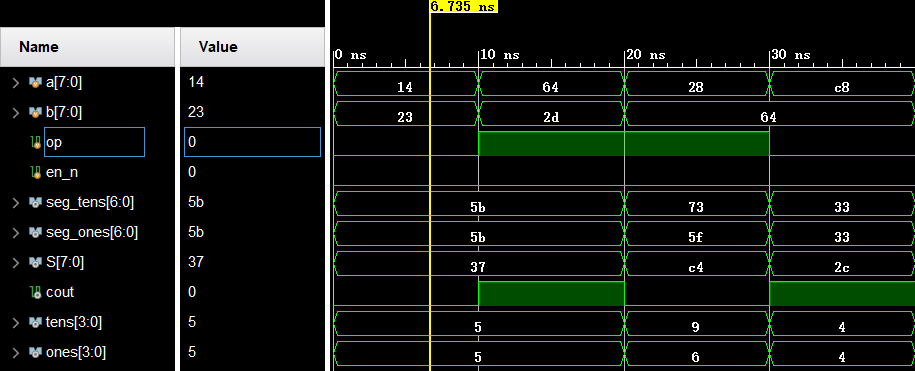
endmodule

(c) RTL视图

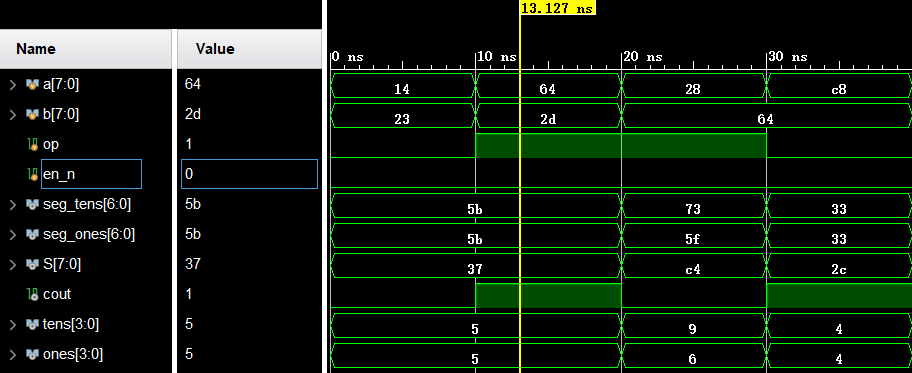


（3）对上述模块再加上数码管显示进行封装，进行仿真：

op为0，代表对A，B进行加法，20 + 35 = 55，十位输出5，个位是5，S是0x37h



op为1，代表对A,B进行减法，100 – 45 = 55，十位输出5，个位是5，S是0x37h



ModelSim源程序：

`timescale 1ns/100ps

module top\_plus\_tb;

// 输入信号

reg [7:0] a, b;

reg op; // 操作：0-加法，1-减法

reg en\_n; // 低电平有效

// 输出信号

wire [6:0] seg\_tens, seg\_ones;

wire [7:0] S;

wire cout;

// BCD输出

wire [3:0] tens, ones; // 十位和个位BCD输出

// 实例化顶层模块

top uut (

.A(a),

.B(b),

.op(op),

.en\_n(en\_n),

.S(S),

.seg\_tens(seg\_tens),

.seg\_ones(seg\_ones),

.Cout(cout)

);

// 实例化二进制到BCD转换模块

binary2bcd u\_binary2bcd (

.binary(S),

.tens(tens),

.ones(ones)

);

// 测试过程

initial begin

$display("Time\tA\tB\tOP\tResult (seg)\tCOUT");

en\_n = 0; // 启用模块

// 加法测试：20 + 35 = 55

a = 8'd20;

b = 8'd35;

op = 0;

#10;

$display("%4t\t%3d\t%3d\tADD\t%7b %7b\t%b", $time, a, b, seg\_tens, seg\_ones, cout);

// 减法测试：100 - 45 = 55

a = 8'd100;

b = 8'd45;

op = 1;

#10;

$display("%4t\t%3d\t%3d\tSUB\t%7b %7b\t%b", $time, a, b, seg\_tens, seg\_ones, cout);

// 减法测试：40 - 100 = 196（负数补码表现）

a = 8'd40;

b = 8'd100;

op = 1;

#10;

$display("%4t\t%3d\t%3d\tSUB\t%7b %7b\t%b", $time, a, b, seg\_tens, seg\_ones, cout);

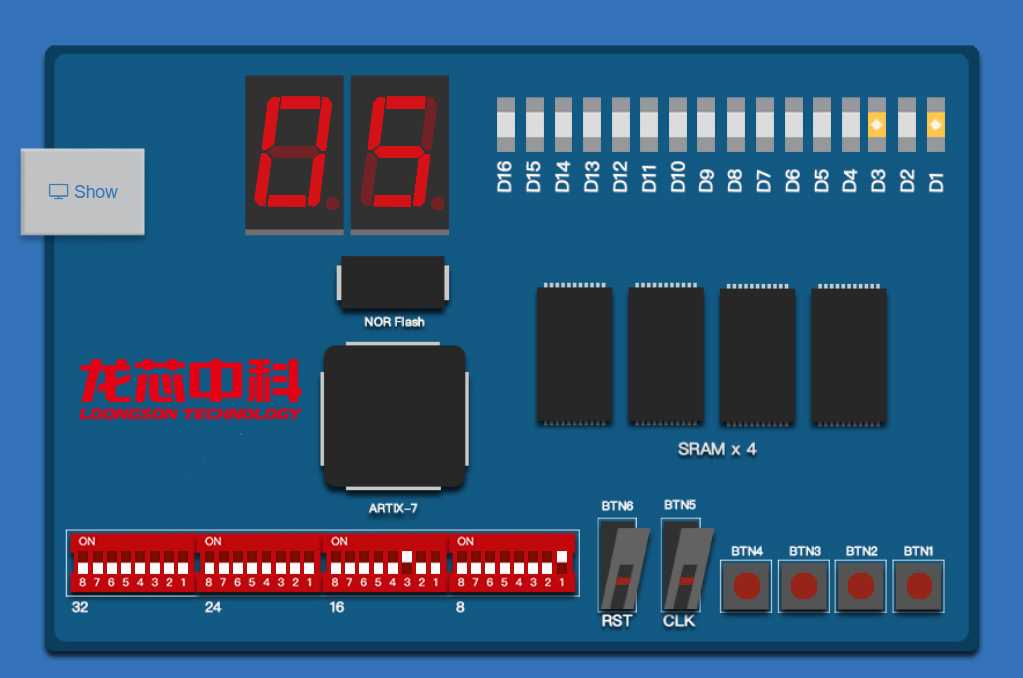
$stop;

end

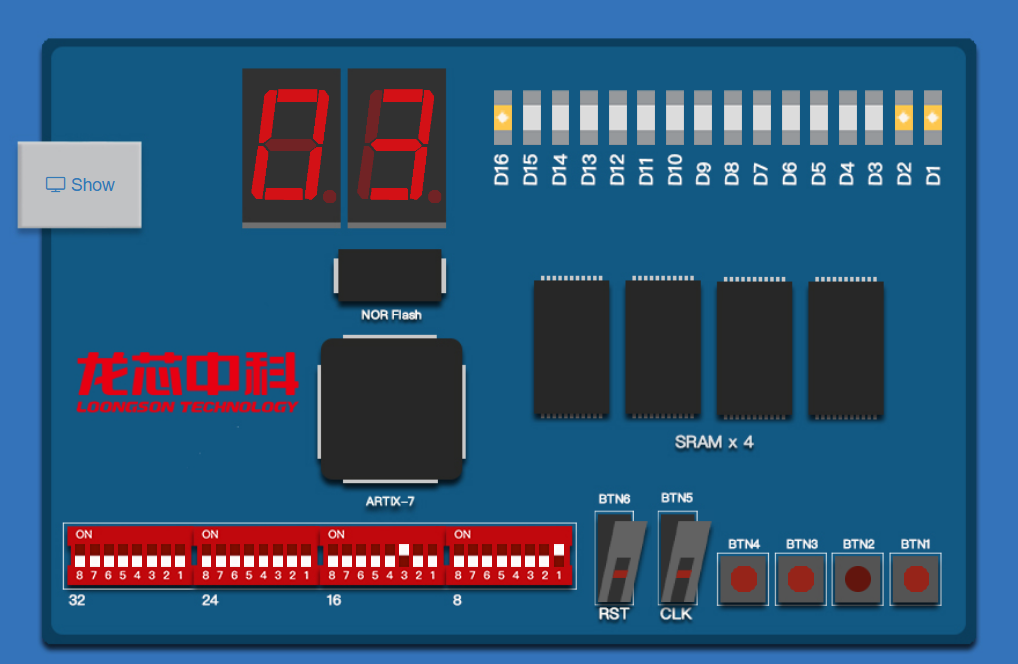
endmodule

（4）

BTN2为低电平，表示对A，B进行加法：4 + 1 = 5；



BTN2为高电平，表示对A,B进行减法：4 – 1 = 3：



五、实验过程中出现的故障现象、原因分析及解决的办法

故障现象：

绑定数码管引脚后，提交至实验平台发现并没有按想象中的亮灭。

原因分析：

首先检查仿真，观察8位二进制转BCD结果正确，接下来排查十位、各位对应的DPY模块输入输出都是正确的，那么问题只能出在约束文件的绑定上。

解决：

经过对.xdc文件DPY0,，DPY1的测试，发现引脚并不是按顺序排列的，根据调试出的顺序进行正确的绑定后，数码管正确显示。