gdut

\_\_ \_计算机 \_\_学院 \_\_ **网络工程**\_\_专业 \_\_ **2** 班

学号\_3219005307 \_姓名\_\_戴嘉欣\_\_ 教师评定\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

实验题目\_ 基于Libero的数字逻辑设计仿真及验证实验\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| 序号 | 实验内容 | 完成情况 |
| 4 | 用EDA设计仿真基本门电路并烧录验证  用EDA设计仿真组合逻辑电路74HC148、74HC138并烧录验证 |  |
| 5 | 用EDA设计仿真组合逻辑电路74HC153、74HC85、74HC283、74HC4511并烧录验证  课件5-5-3EDA开发综合实例3 SmartDesign的使用 |  |
| 6 | 中华好声音表决电路  用EDA设计仿真时序逻辑电路74HC74、74HC112、74HC194并烧录验证 |  |
| 7 | 综合实验（至少做3道综合题），使用SmartDesign工具来设计 |  |
| 8 | 大考核：在规定时间内完成老师布置的题目并给老师现场检查 |  |

**实验报告**

## 基本门电路

一、实验目的

1、了解基于Verilog的基本门电路的设计及其验证。

2、熟悉利用EDA工具进行设计及仿真的流程。

3、学习针对实际门电路芯片74HC00、74HC02、74HC04、74HC08、74HC32、74HC86进行VerilogHDL设计的方法。

4、熟悉实验箱的使用和程序下载（烧录）及测试的方法。

二、实验环境及仪器

1、Libero仿真软件。

2、数字逻辑与系统设计实验箱。

3、Actel A3P060 FPGA芯片及Flash Pro5烧录器。

三、实验内容

1、掌握Libero软件的使用方法。

2、进行针对74系列基本门电路的设计，并完成相应的仿真实验。

3、参考教材中相应章节的设计代码、测试平台代码（可自行编程），完成74HC00、74HC02、74HC04、74HC08、74HC32、74HC86相应的设计、综合及仿真。

4、提交针对74HC00、74HC02、74HC04、74HC08、74HC32、74HC86的综合结果，以及相应的仿真结果。

四、实验结果和数据处理

1、所有模块及测试平台代码清单

//74HC00代码-与非

module hc00(a,b,y);

input[4:1] a,b;

output[4:1] y;

reg[4:1] y;

always@(a,b)

y=~(a&b);

endmodule

//74HC00测试平台代码

`timescale 1ns/1ns

module testhc00();

reg[4:1] pa,pb;

wire[4:1] y;

hc00 test74hc00(pa,pb,y);

initial

begin

pa=4'b0000;pb=4'b0001;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10 pb=pb<<1;

pa=4'b1111;pb=4'b0001;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10 pb=pb<<1;

end

endmodule

//74HC02代码-或非

module hc02(a,b,y);

input[4:1] a,b;

output[4:1] y;

reg[4:1] y;

always@(a,b)

y=~(a|b);

endmodule

//74HC02测试平台代码

`timescale 1ns/1ns

module testhc02();

reg[4:1] pa,pb;

wire[4:1] y;

hc02 test74hc02(pa,pb,y);

initial

begin

pa=4'b0000;pb=4'b0001;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10 pb=pb<<1;

pa=4'b1111;pb=4'b0001;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10 pb=pb<<1;

end

endmodule

//74HC04代码-非

module hc04(a,y);

input[6:1] a;

output[6:1] y;

reg[6:1] y;

always@(a)

y=~a;

endmodule

//74HC04测试平台代码

`timescale 1ns/1ns

module testhc04();

reg[6:1] pa;

wire[6:1] y;

hc04 test74hc04(pa,y);

initial

begin

pa=6'b000001;

#10 pa=pa<<1;//000010

#10 pa=pa<<1;//000100

#10 pa=pa<<1;//001000

#10 pa=pa<<1;//010000

#10 pa=pa<<1;//100000

#10;

end

Endmodule

//74HC08代码-与

module hc08(a,b,y);

input[4:1] a,b;

output[4:1] y;

reg[4:1] y;

always@(a,b)

y=a&b;

endmodule

//74HC08测试平台代码

`timescale 1ns/1ns

module testhc08();

reg[4:1] pa,pb;

wire[4:1] y;

hc08 test74hc08(pa,pb,y);

initial

begin

pa=4'b0000;pb=4'b0001;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10 pb=pb<<1;

pa=4'b1111;pb=4'b0001;

#10;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10;

end

endmodule

//74HC32代码-或

module hc32(a,b,y);

input[4:1] a,b;

output[4:1] y;

reg[4:1] y;

always@(a,b)

y=a|b;

endmodule

//74HC32测试平台代码

`timescale 1ns/1ns

module testhc32();

reg[4:1] pa,pb;

wire[4:1] y;

hc32 test74hc32(pa,pb,y);

initial

begin

pa=4'b0000;pb=4'b0001;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10;

pa=4'b1111;pb=4'b0001;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10;

end

endmodule

//74HC86代码-异或

module hc86(a,b,y);

input[4:1] a,b;

output[4:1] y;

reg[4:1] y;

always@(a,b)

y=a^b;

endmodule

//74HC86测试平台代码

`timescale 1ns/1ns

module testhc86();

reg[4:1] pa,pb;

wire[4:1] y;

hc86 test74hc86(pa,pb,y);

initial

begin

pa=4'b0000;pb=4'b0001;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10 pb=pb<<1;

#10;

pa=4'b1111;pb=4'b0001;

#10 pb=pb<<1;

#10 pb=pb<<1;

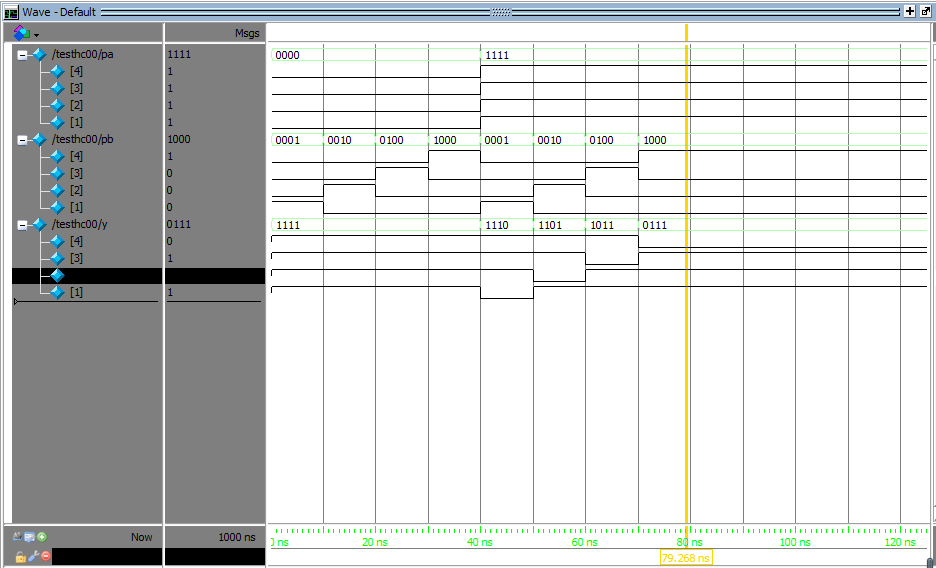
#10 pb=pb<<1;

#10;

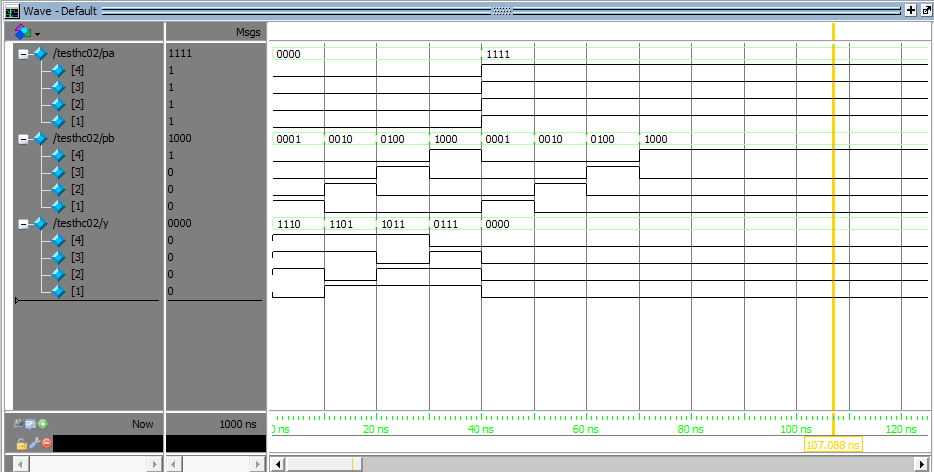
end

endmodule

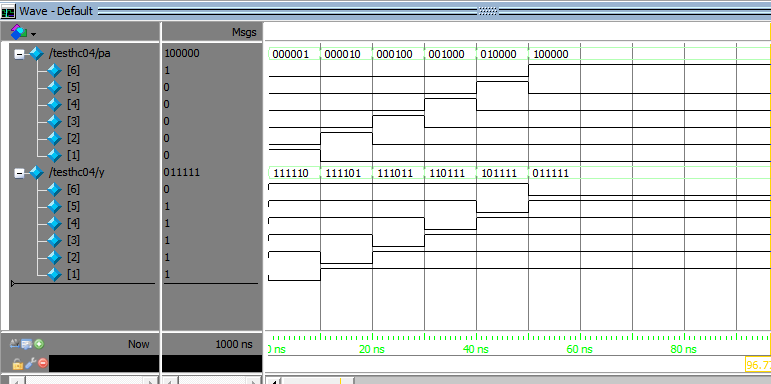
1. 第一次仿真结果（**截图，注明对应的门电路**）。（将波形窗口背景设为**白色**，调整窗口至合适大小，使波形能完整显示，对窗口**截图**。后面实验中的仿真使用相同方法处理）



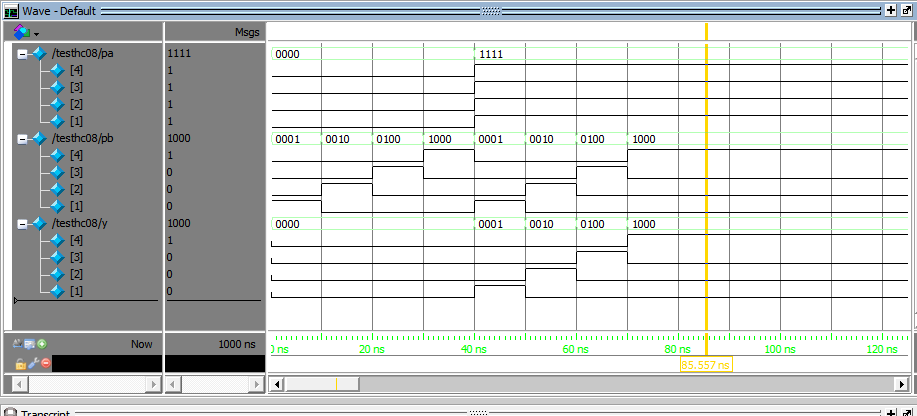
74HC00



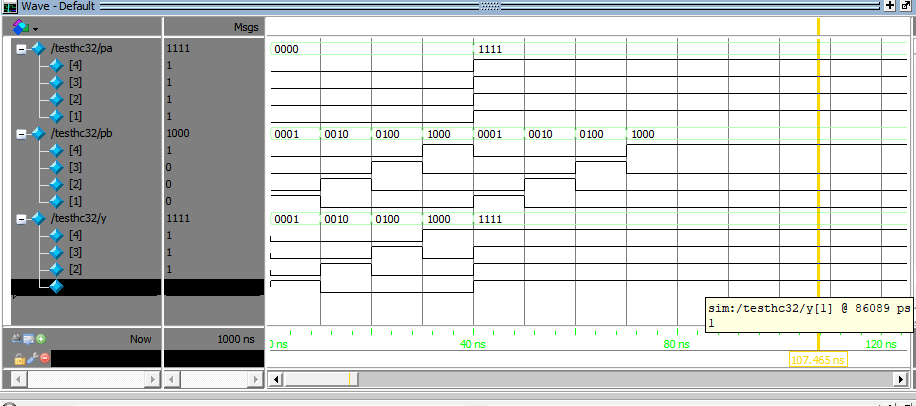
74HC02



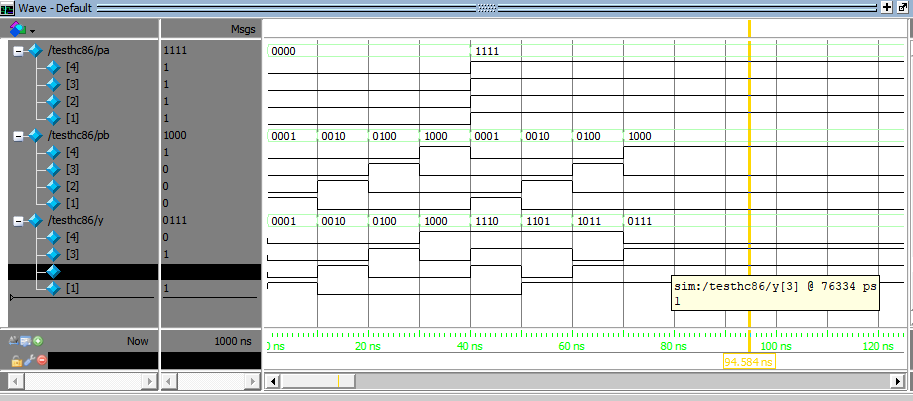
74HC04



74HC08

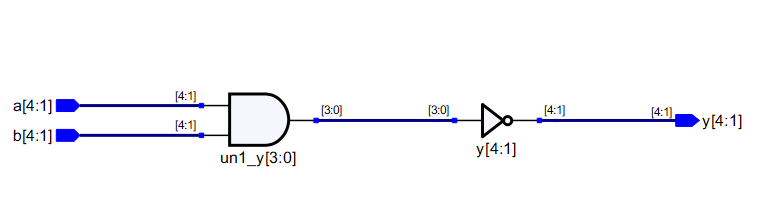


74HC32

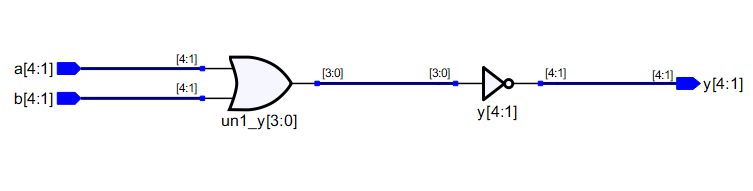


74HC86

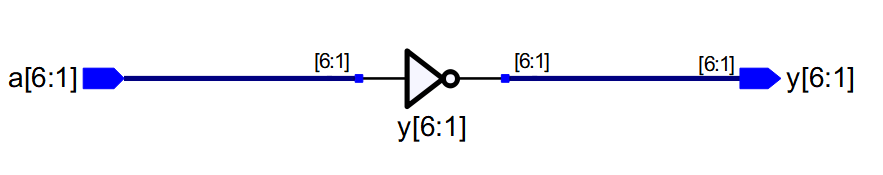
3、综合结果（**截图，注明对应的门电路**）。（将相关窗口调至合适大小，使RTL图能完整显示，对窗口截图，后面实验中的综合使用相同方法处理）



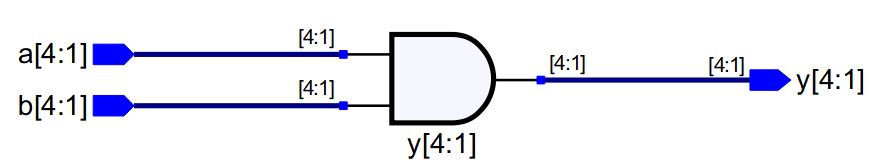
74HC00



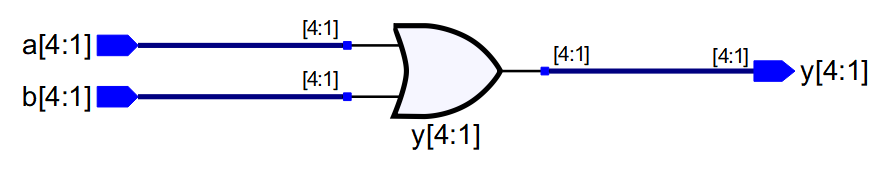
74HC02



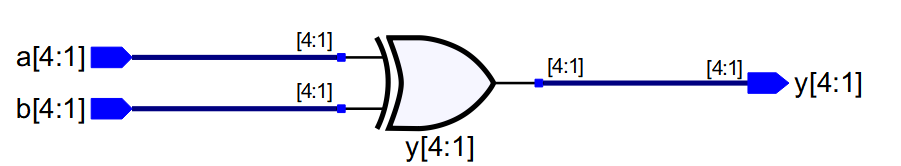
74HC04



74HC08

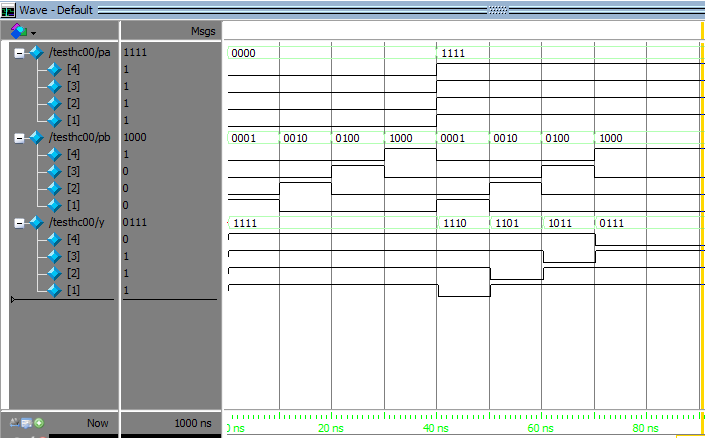


74HC32

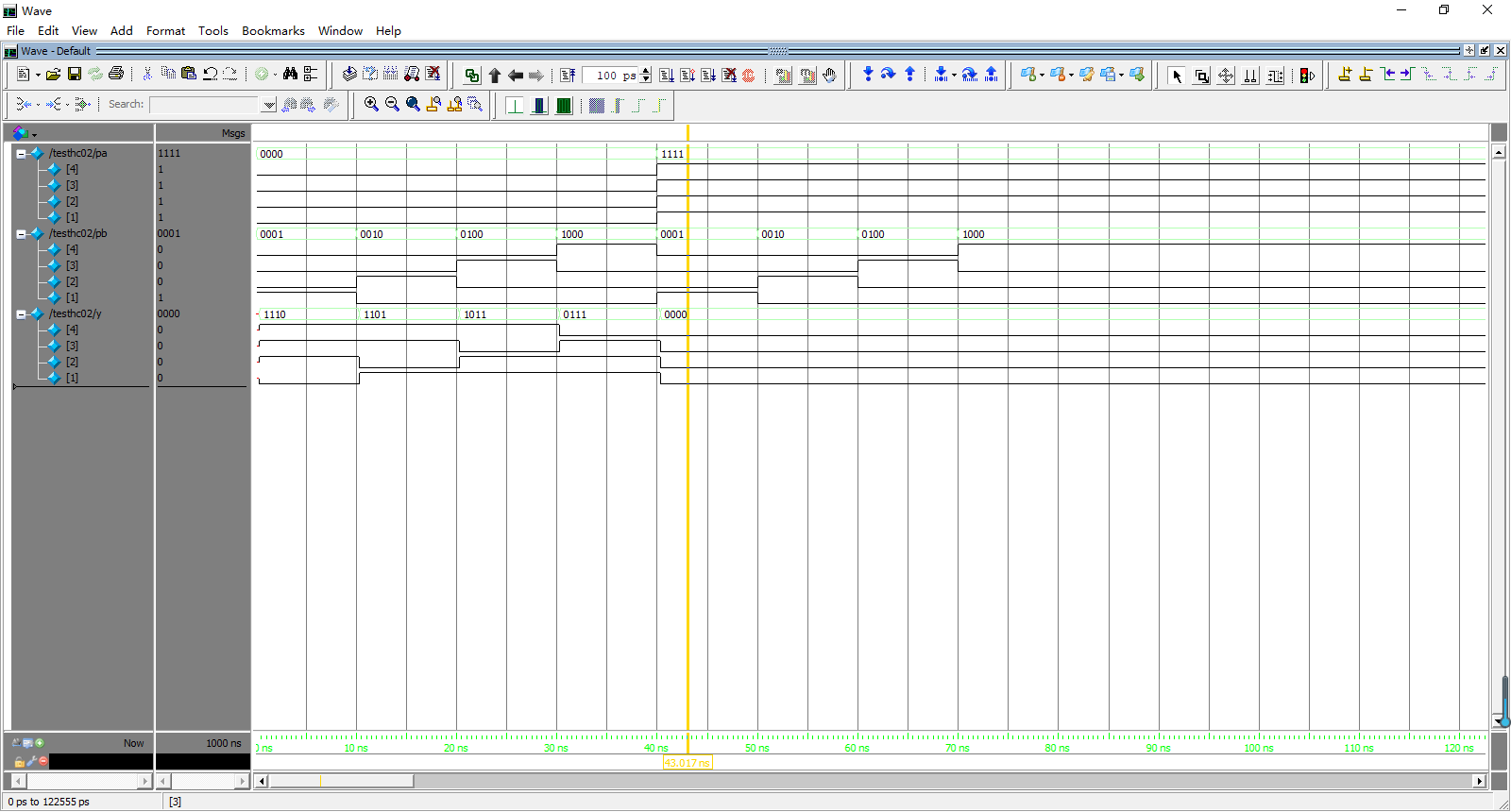


74HC86

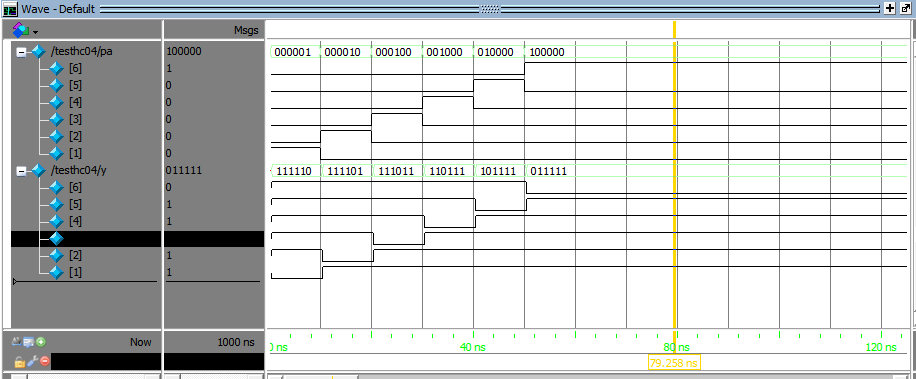
4、第二次仿真结果（综合后）（**截图，注明对应的门电路**）。回答输出信号是否有延迟，延迟时间约为多少？



74HC00

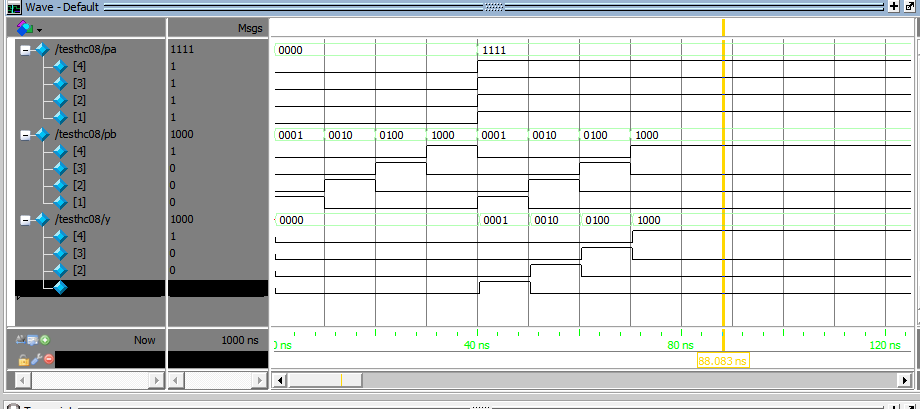


74HC02



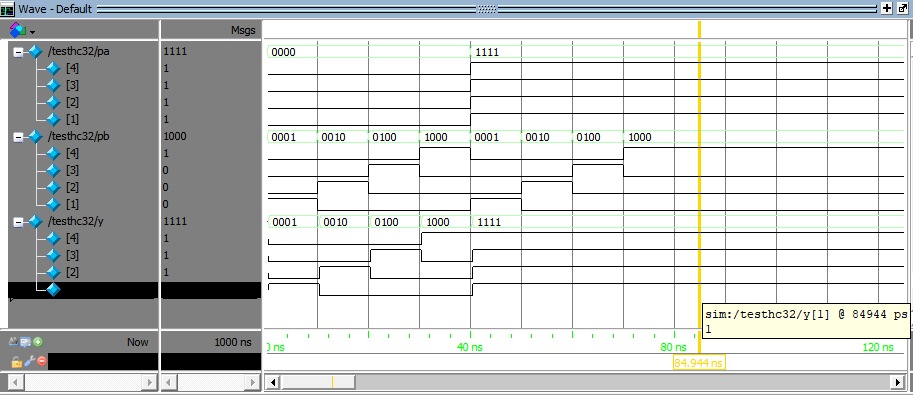
74HC04

延迟了0.3ns



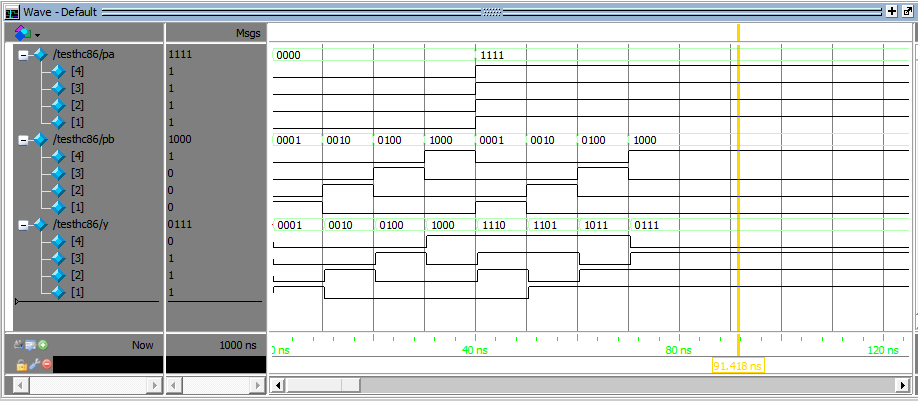
74HC08

延迟了0.3ns



74HC32

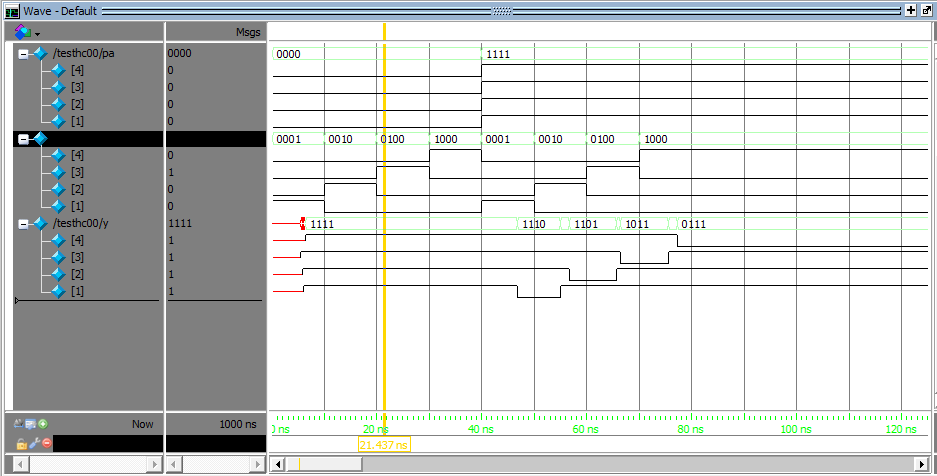
延迟了0.3ns



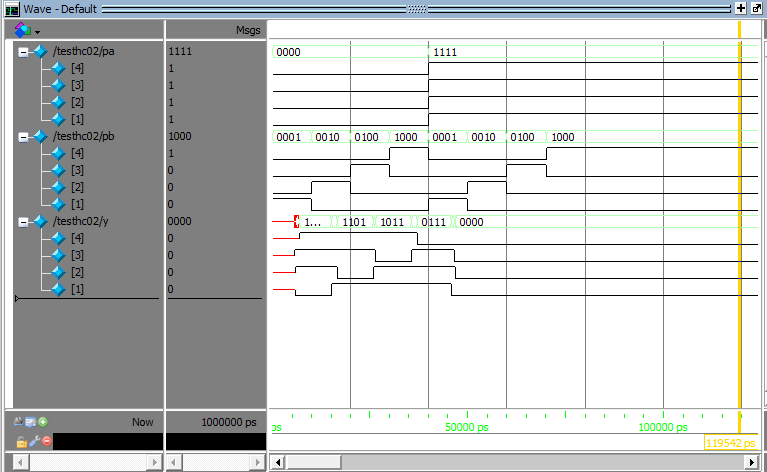
74HC86

延迟了0.3ns

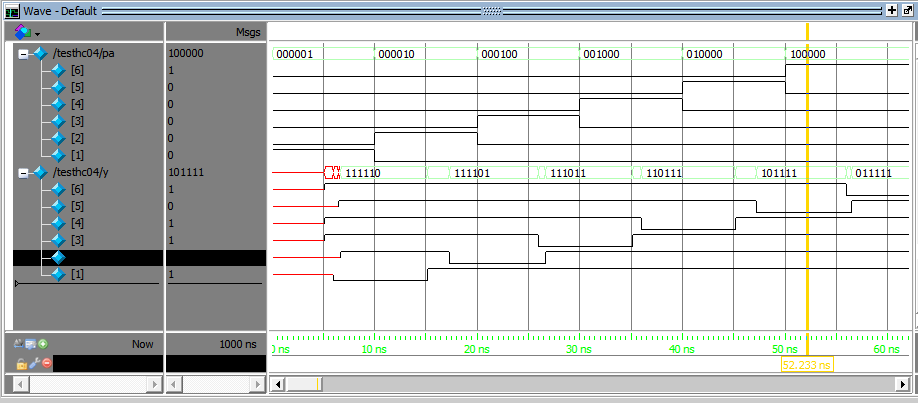
5、第三次仿真结果（布局布线后）（**截图，注明对应的门电路**）。回答输出信号是否有延迟，延迟时间约为多少？分析是否有出现竞争冒险。



74HC00

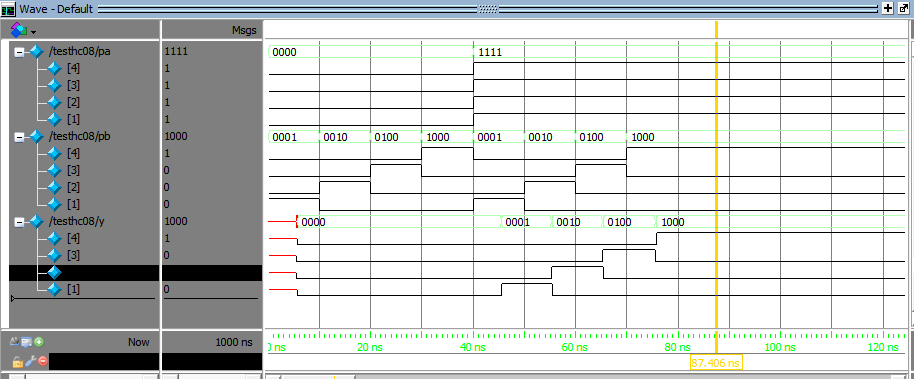


74HC02



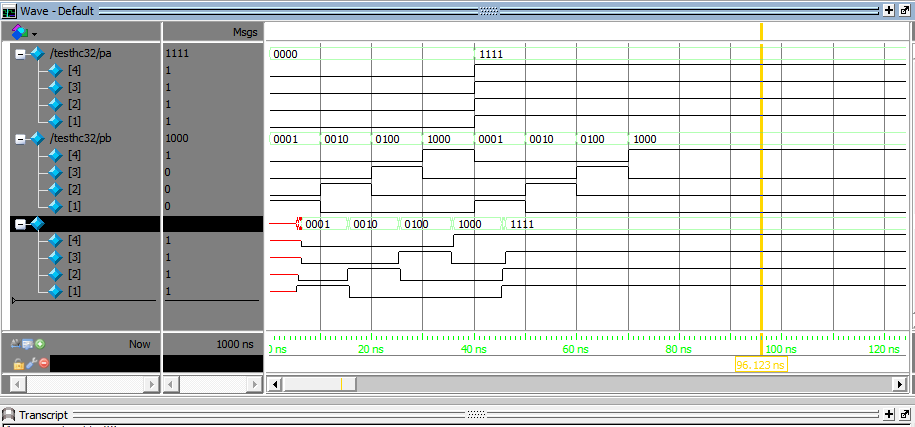
74HC04

延迟6.631ns



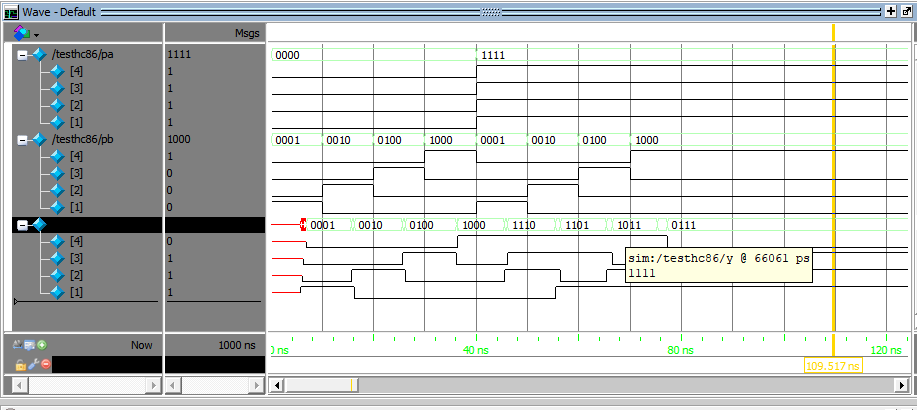
74HC08

延迟5.782ns



74HC32

延迟6.402ns



74HC86

延迟6.955ns

6、烧录验证：典型实验数据拍照、录像**（课堂给老师检查，课后全班统一刻光盘）**。

## 组合逻辑电路

一、实验目的

1、了解基于Verilog的组合逻辑电路的设计及其验证。

2、熟悉利用EDA工具进行设计及仿真的流程。

3、熟悉实验箱的使用和程序下载（烧录）及测试的方法。

4、学习针对实际组合逻辑电路芯片74HC148、74HC138、74HC153、74HC85、74HC283、74HC4511进行VerilogHDL设计的方法。

二、实验环境及仪器

1、Libero仿真软件。

2、数字逻辑与系统设计实验箱。

3、Actel A3P060 FPGA芯片及Flash Pro5烧录器。

三、实验内容

1、掌握Libero软件的使用方法。

2、进行针对74系列基本组合逻辑电路的设计，并完成相应的仿真实验。

3、参考教材中相应章节的设计代码、测试平台代码（可自行编程），完成74HC148、74HC138、74HC153、74HC85、74HC283、74HC4511相应的设计、综合及仿真。

4、74HC85测试平台的测试数据要求：进行比较的A、B两数，分别为本人学号的末两位，如“89”，则A数为“1000”，B数为“1001”。若两数相等，需考虑级联输入（级联输入的各种取值情况均需包括）；若两数不等，则需增加一对取值情况，验证A、B相等时的比较结果。

5、74HC4511设计成扩展型的，即能显示数字0~9、字母a~f（此部分暂时不用烧录）。

6、提交针对74HC148、74HC138、74HC153、74HC85、74HC283、74HC4511的综合结果，以及相应的仿真结果。

四、实验结果和数据处理

1、所有模块及测试平台代码清单

//74HC148代码

module hc148(EI,In,Out,GS,EO);

input[7:0] In;

input EI;

output[2:0] Out;

output GS,EO;

reg[2:0] Out;

reg GS,EO;

integer I;

always@(EI or In)

if(EI)

begin

Out=3'b111;GS=1;EO=1;

end

else

if(In==8'b11111111)

begin

Out=3'b111;GS=1;EO=0;

end

else

begin

for(I=0;I<8;I=I+1)

begin

if(~In[I])

begin

Out=~I;GS=0;EO=1;

end

end

end

endmodule

//74HC148测试平台代码

`timescale 1ns/1ns

module testhc148();

reg ei;

reg[7:0] turn;

wire[7:0]in=~turn;

wire[2:0] out;

wire eo,gs;

hc148 test74hc148(ei,in,out,eo,gs);

initial

begin

ei=1;turn=8'b1;

repeat(8)

#10 turn=turn<<1;

ei=0;turn=8'b1;

repeat(8)

#10 turn=turn<<1;

end

endmodule

//74HC138代码

//74HC138测试平台代码

//74HC153代码

module hc153(DateOut,DateIn,Sel,Enable);

input[3:0]DateIn;

input[1:0]Sel;

input Enable;

output reg DateOut;

always@(Enable or Sel or DateIn)

if(Enable) DateOut=0;

else DateOut=DateIn[Sel];

endmodule

//74HC153测试平台代码

`timescale 1ns/1ns

module testhc153();

wire out;

reg[3:0]in;

reg[1:0]sel;

reg ei;

hc153 test74hc153(out,in,sel,ei);

initial

begin

ei=0;sel=0;in=4'b1010;

repeat(4)

#10 sel=sel+1;

ei=1;sel=0;in=4'b1010;

repeat(4)

#10 sel=sel+1;

end

endmodule

//74HC85代码

module hc85(A,B,I,Q);

input[3:0] A,B;

input[2:0] I;

output[2:0] Q;

integer i;

reg[2:0] Q;

always@(A,B,I)

begin

if(A==B)

begin

if(I[1]) Q=3'b010;

else if(I==3'b000) Q=3'b101;

else if(I==3'b001) Q=3'b001;

else if(I==3'b100) Q=3'b100;

else if(I==3'b101) Q=3'b000;

end

else

begin

for(i=0;i<4;i=i+1)

if(A[i]>B[i]) Q=3'b100;

else if(A[i]<B[i]) Q=3'b001;

end

end

endmodule

//74HC85测试平台代码

`timescale 1ns/1ns

module testhc85();

reg[3:0] a,b;

reg[2:0] i;

wire[2:0] q;

hc85 test74hc85(a,b,i,q);

initial

begin

a=4'b1111;b=4'b0111;

#10 a=4'b0111;b=4'b1111;

#10 a=4'b0100;b=4'b0011;

#10 a=4'b0000;b=4'b0111;

#10 a=4'b0010;b=4'b0001;

#10 a=4'b0000;b=4'b0011;

#10 a=4'b0001;b=4'b0000;

#10 a=4'b0000;b=4'b0001;

#10 b=4'b0000;i=3'b000;

#10 i=3'b001;

#10 i=3'b100;

#10 i=3'b101;

#10 i=3'b111;

#10 i=3'b110;

#10 i=3'b010;

#10 i=3'b011;

end

endmodule

//74HC283代码

module HC283(DateA,DateB,Cin,Sum,Cout);

input [3:0]DateA,DateB;

input Cin;

output [3:0]Sum;

output Cout;

reg [4:0]Buf;

assign{Cout,Sum}=Buf;

always@(DateA or DateB or Cin)

Buf=DateA+DateB+Cin+5'd0;

endmodule

//74HC283测试平台代码

`timescale 1ns/100ps

module test\_283;

reg [3:0]a, b;

reg cin;

wire [3:0]sum;

wire cout;

HC283 u(a, b, cin, sum, cout);

initial

begin

a=0; b=0;

repeat(10)

begin

#10 a=$random; b=$random;

end

end

initial

begin

cin=0;

#50 cin=1;

end

endmodule

//74HC4511代码

module hc4511(Out,In,LE,BI\_N,LT\_N);

output[7:0] Out;

input[3:0] In;

input LE,BI\_N,LT\_N;

reg [7:0]Buf;

assign Out=Buf;

always@(In or LE or BI\_N or LT\_N)

begin

if(!LT\_N) Buf=8'b11111111;

else if(!BI\_N) Buf=8'b00000000;

else if(LE) Buf=Buf;

else

case(In)

4'd0:Buf=8'b00111111;

4'd1:Buf=8'b00000110;

4'd2:Buf=8'b01011011;

4'd3:Buf=8'b01001111;

4'd4:Buf=8'b01100110;

4'd5:Buf=8'b01101101;

4'd6:Buf=8'b01111101;

4'd7:Buf=8'b00000111;

4'd8:Buf=8'b01111111;

4'd9:Buf=8'b01101111;

4'd10:Buf=8'b01110111;

4'd11:Buf=8'b01111100;

4'd12:Buf=8'b00111001;

4'd13:Buf=8'b01011110;

4'd14:Buf=8'b01111001;

4'd15:Buf=8'b01110001;

default:;

endcase

end

endmodule

//74HC4511测试平台代码

`timescale 1ns/100ps

module testhc4511;

reg [3:0]in;

wire [7:0]out;

reg LE, BI\_N, LT\_N;

hc4511 test74hc4511(out, in, LE, BI\_N, LT\_N);

initial

begin

LE=0; BI\_N=0; LT\_N=1;

in=0;

#20 BI\_N=1;

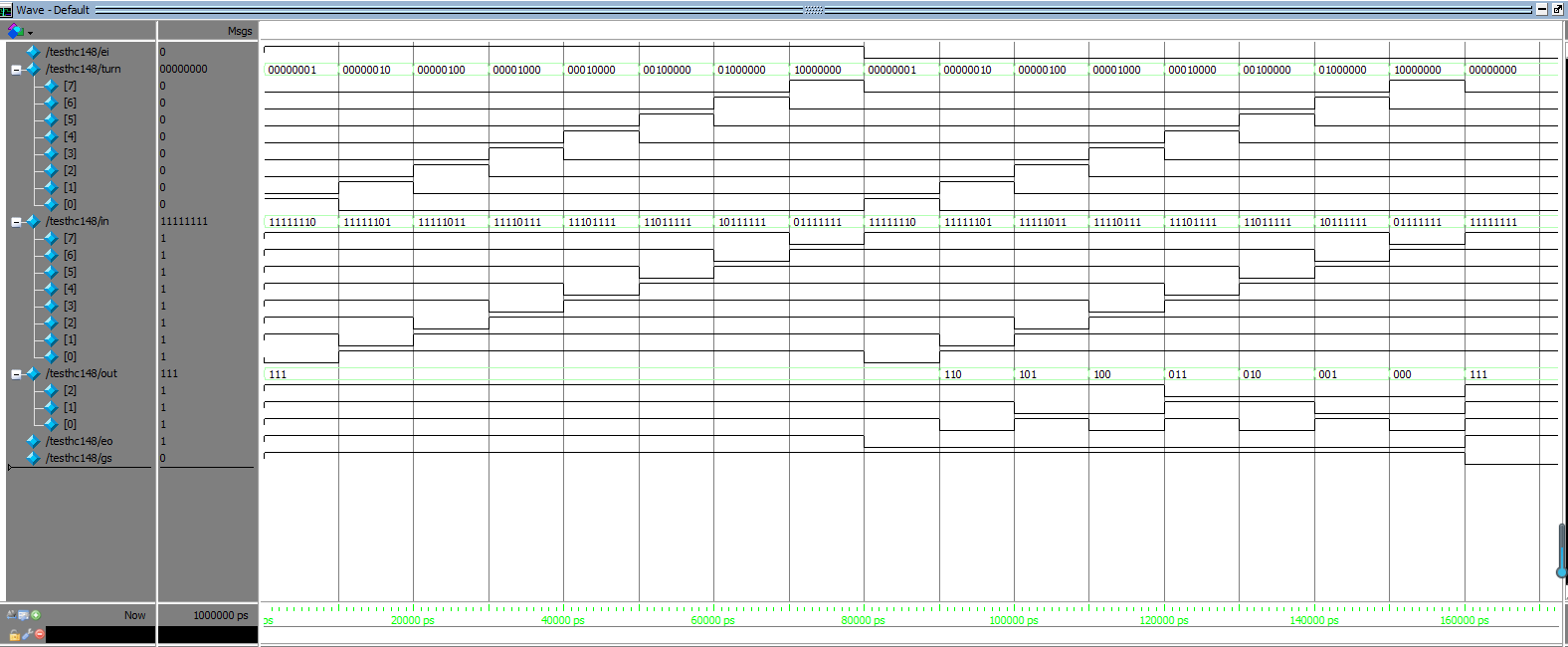
repeat(16)

#20 in=in+1;

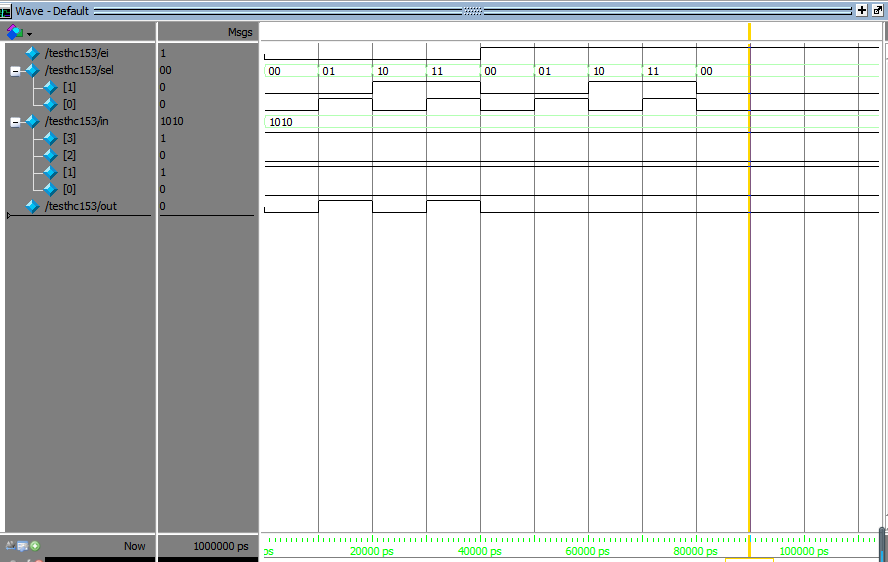
end

endmodule

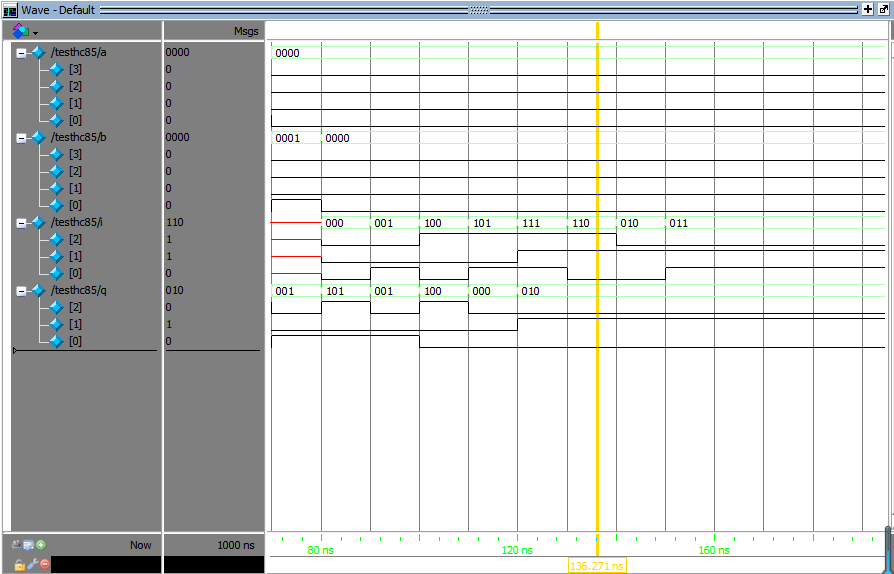
2、第一次仿真结果（**截图，注明对应的模块**）



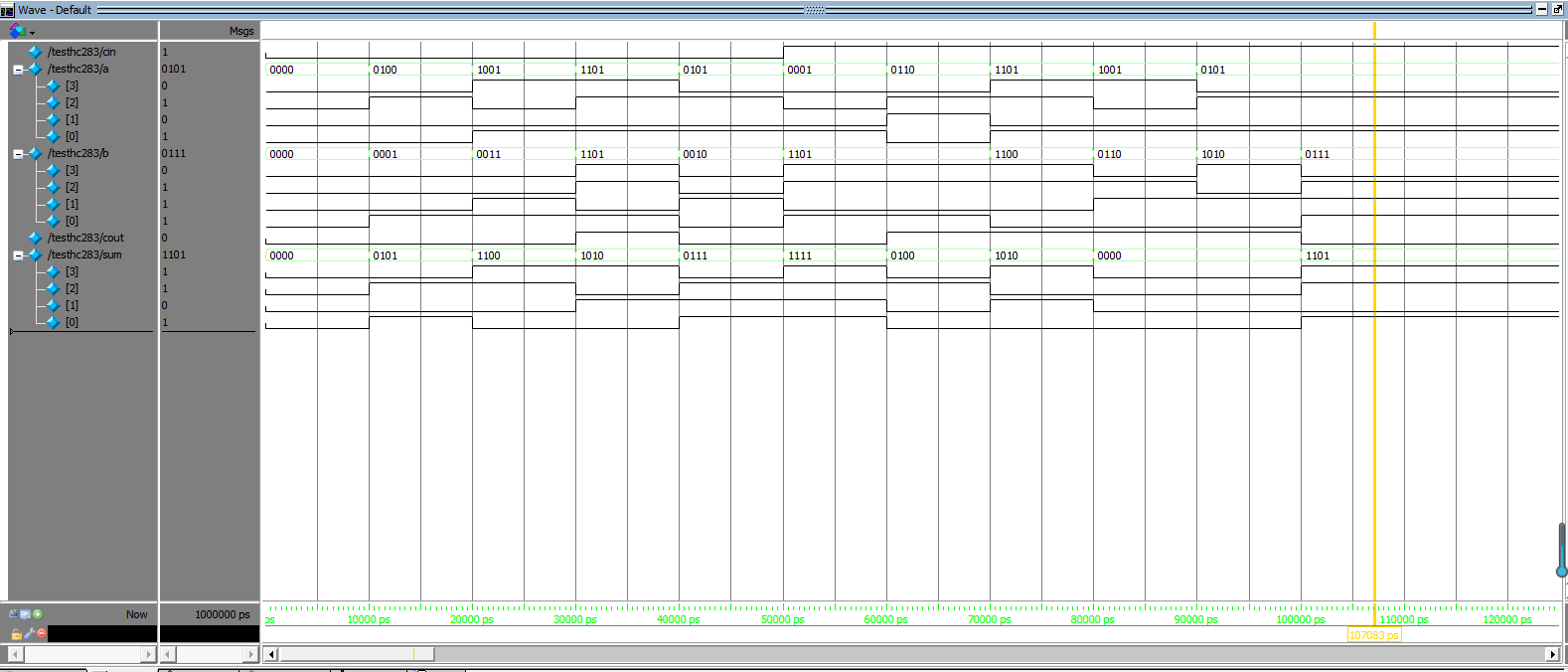
74HC148



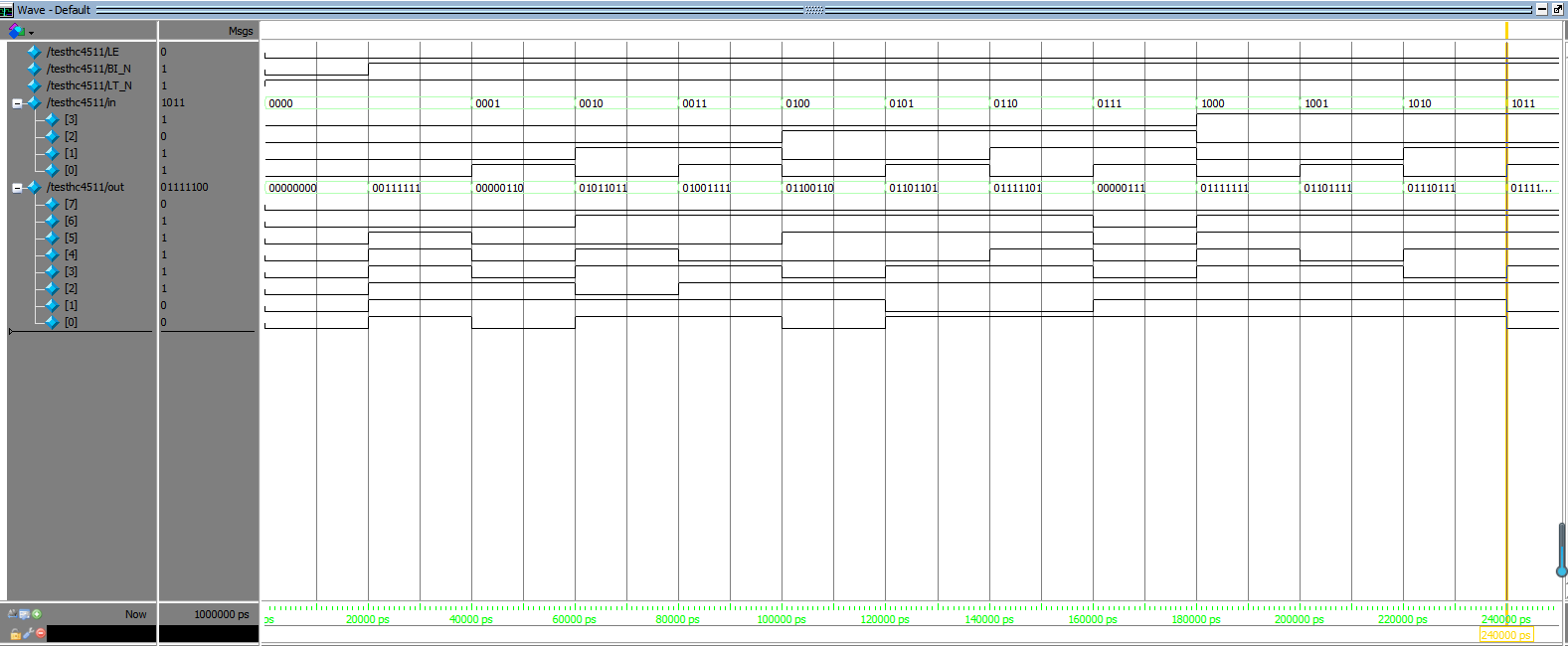
74HC153

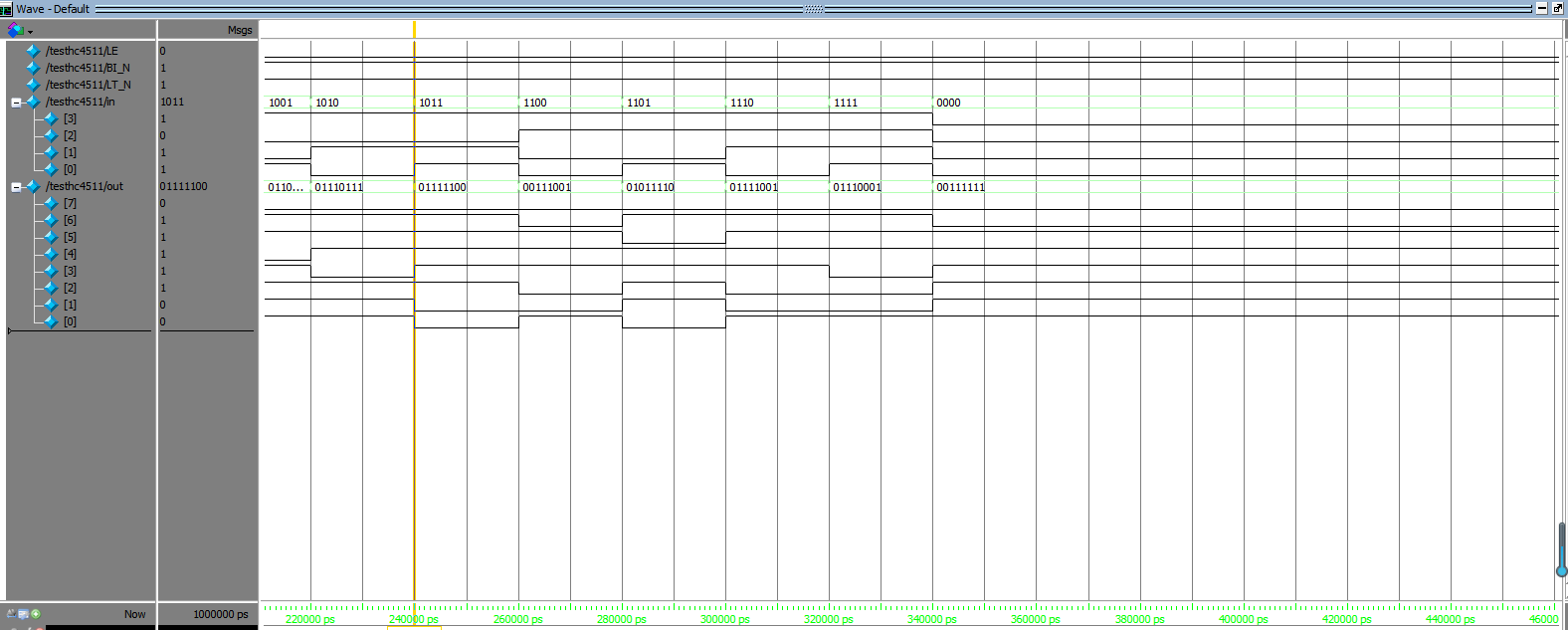


74HC85



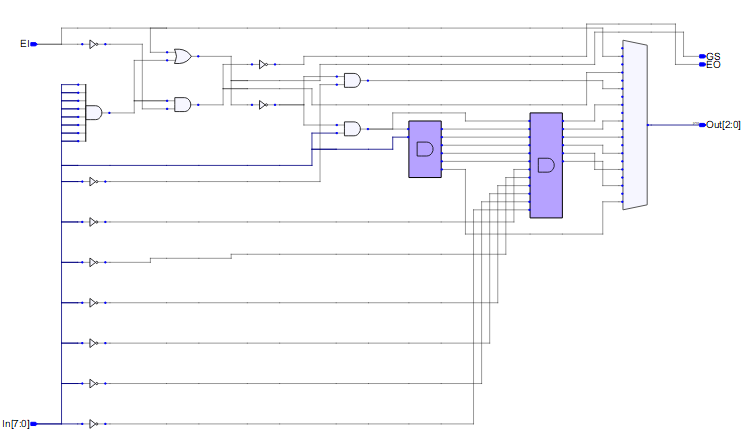
74HC283



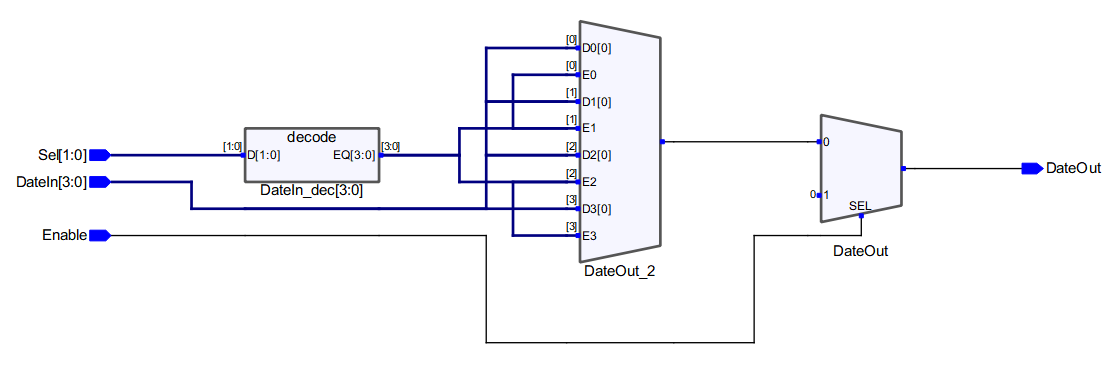


74HC4511

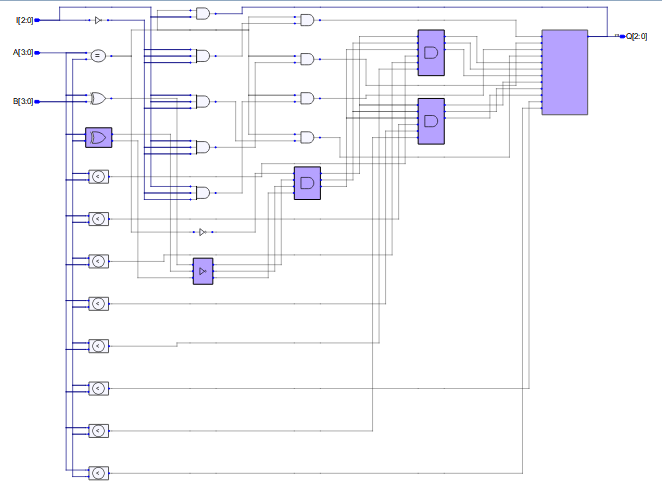
3、综合结果（**截图，注明对应的模块**）



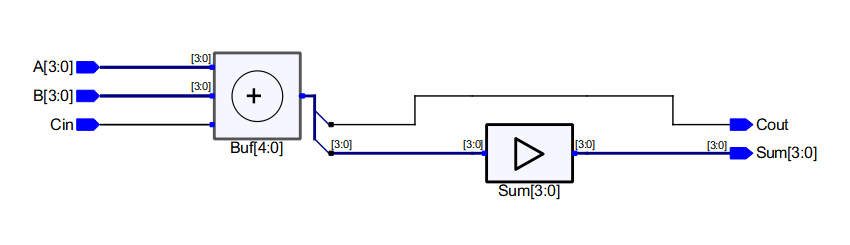
74HC148



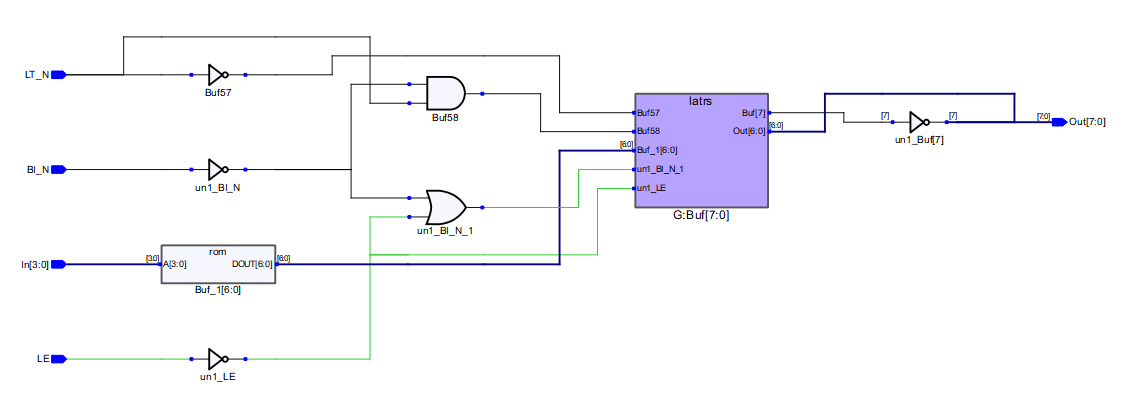
74HC153



74HC85

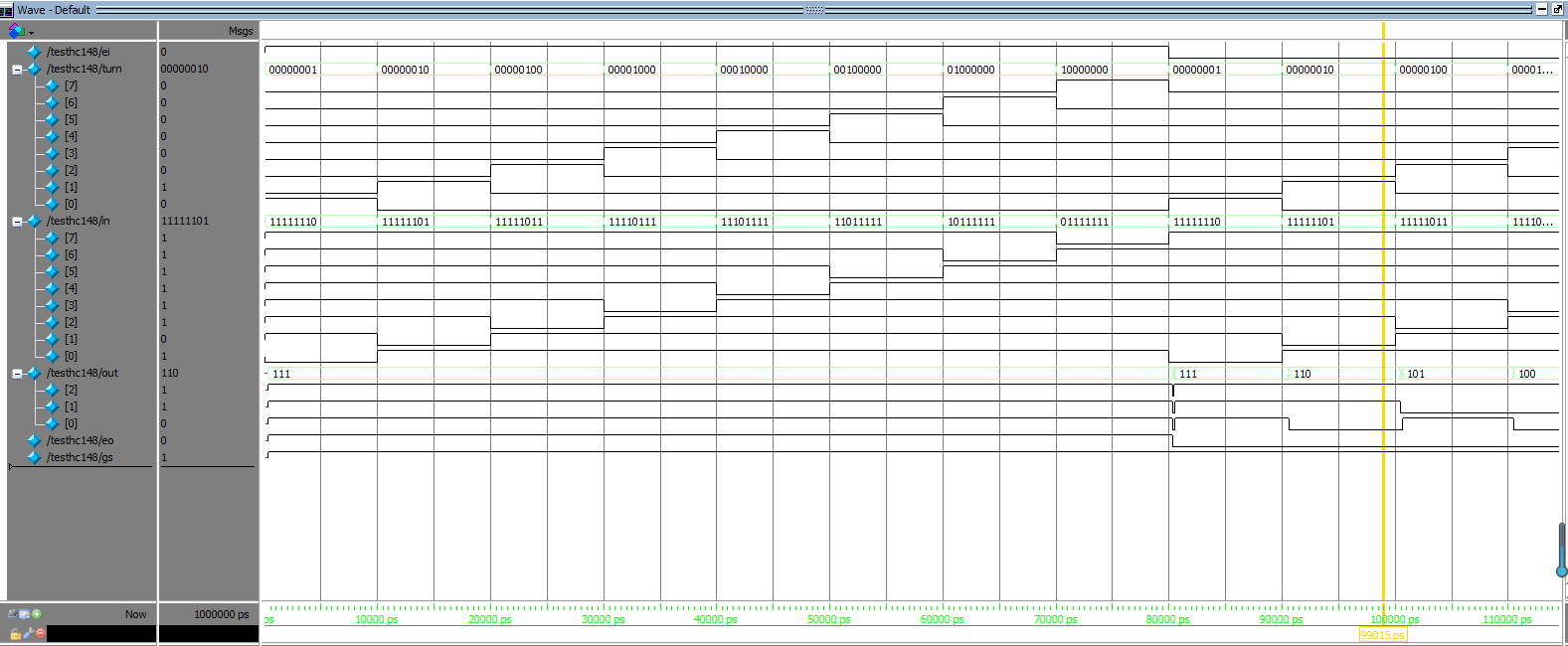


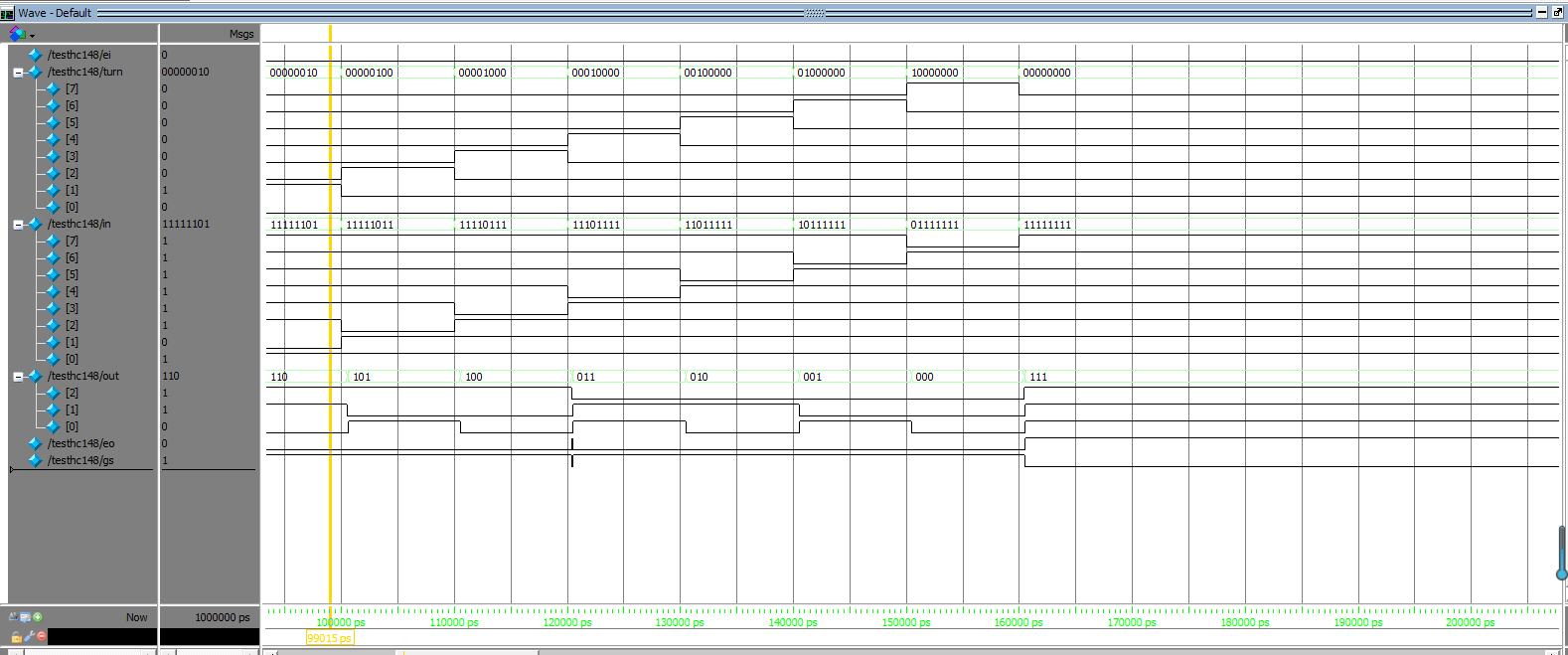
74HC283



74HC4511

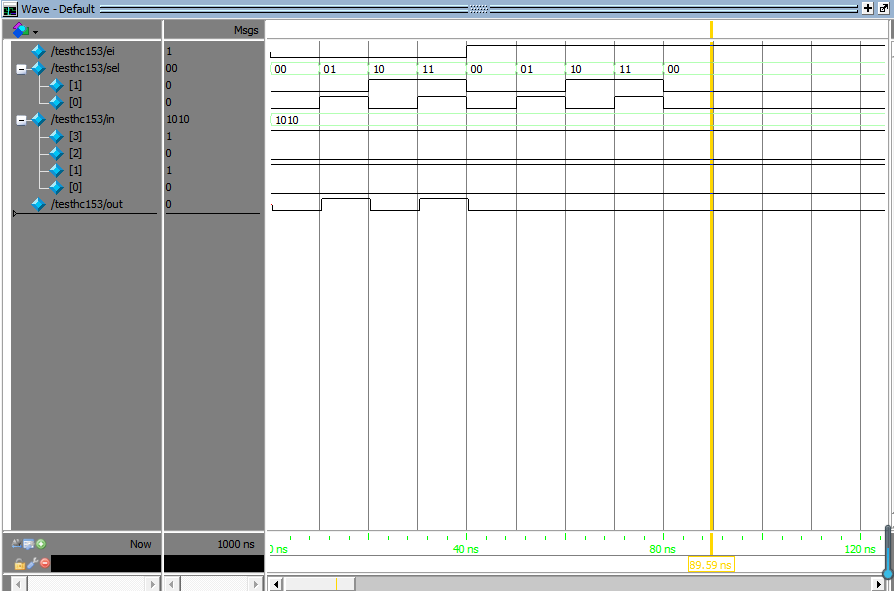
4、第二次仿真结果（综合后）（**截图，注明对应的模块**）。回答输出信号是否有延迟，延迟时间约为多少？





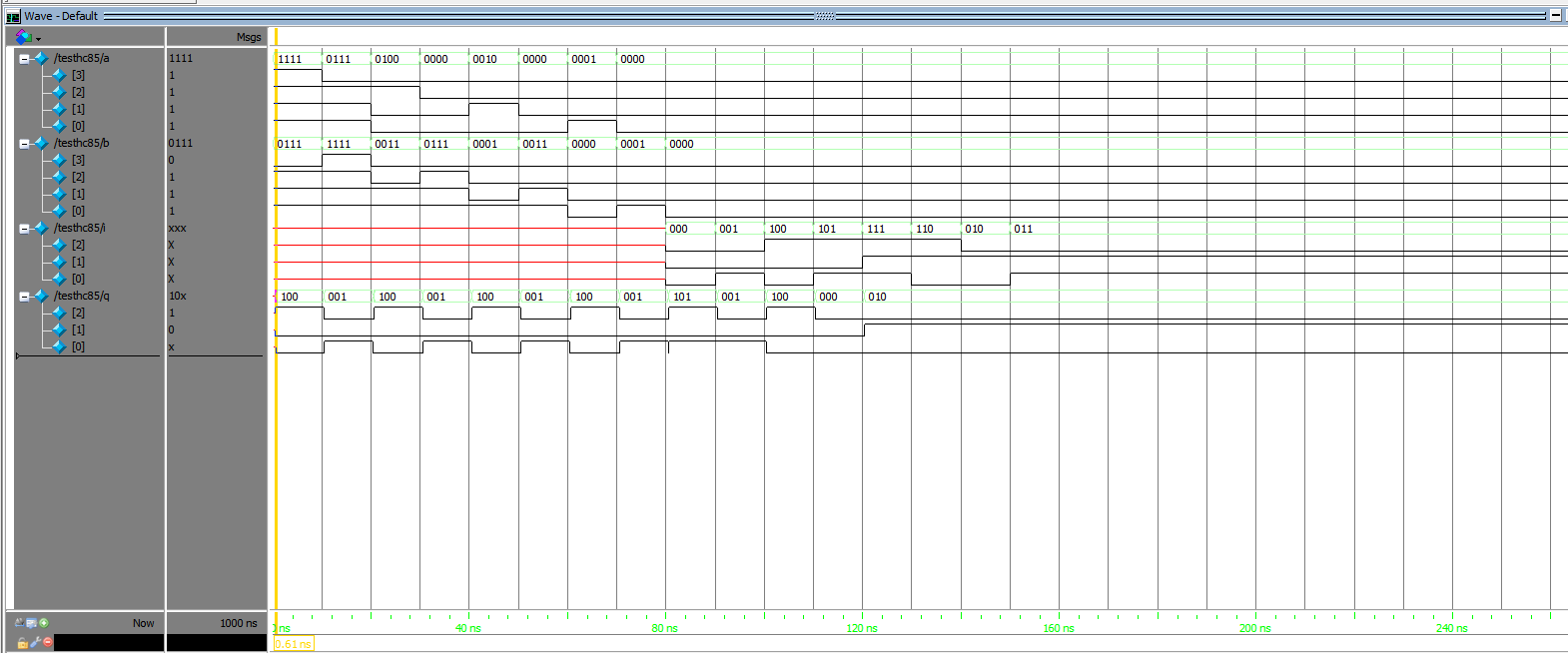
74HC148

延迟300ps



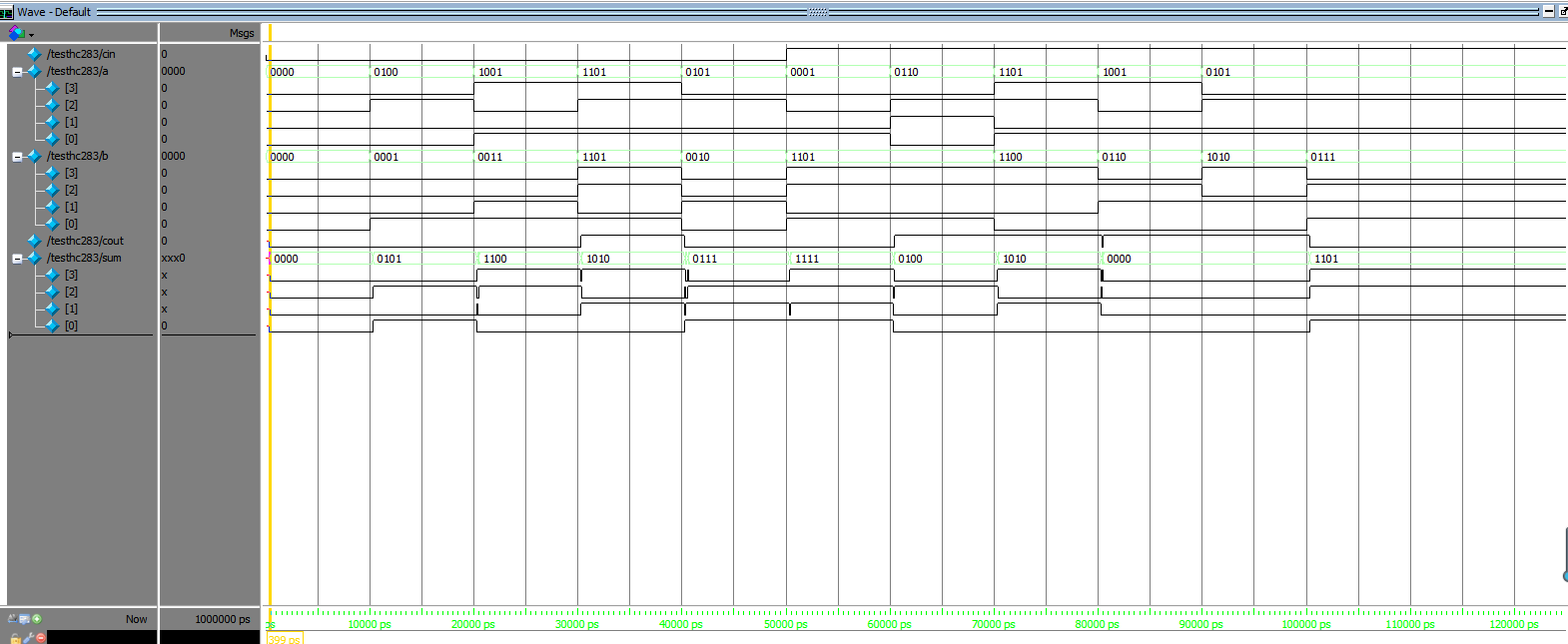
74HC153

延迟0.5ns



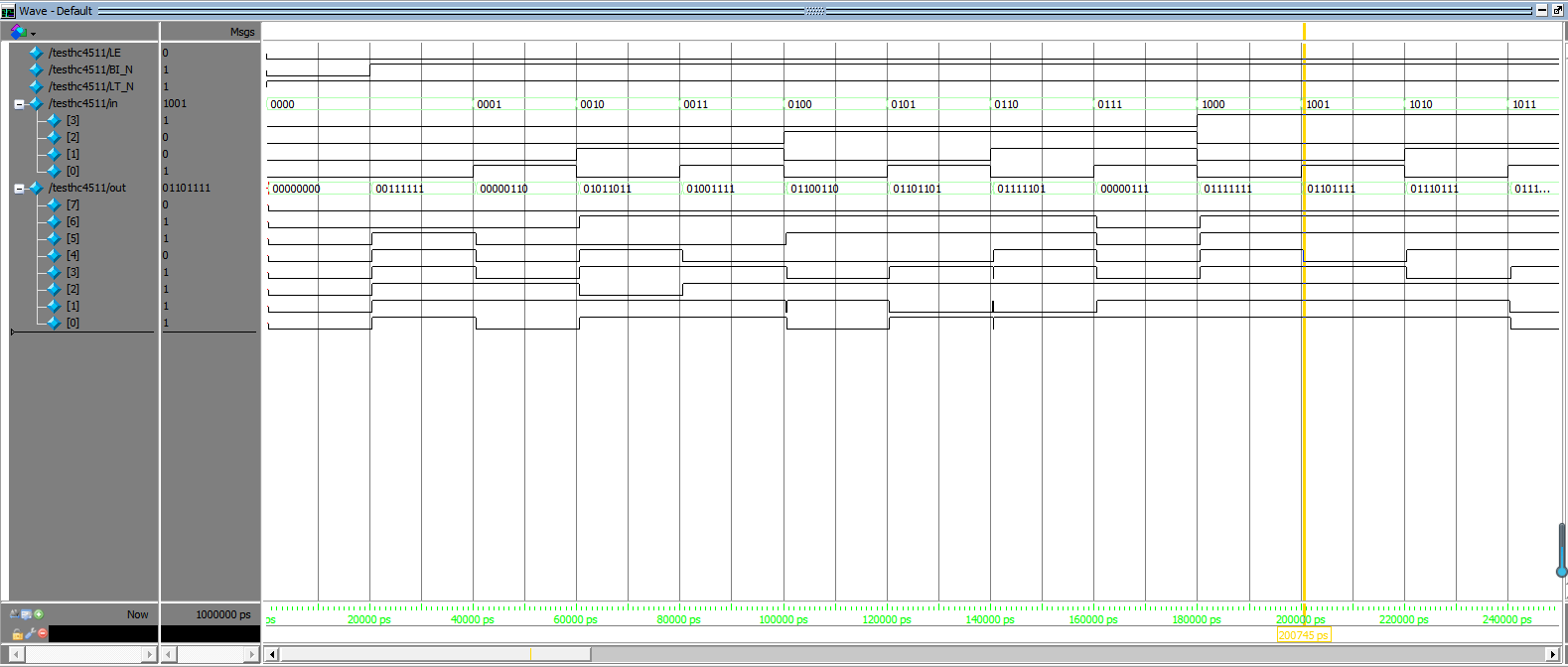
74HC85

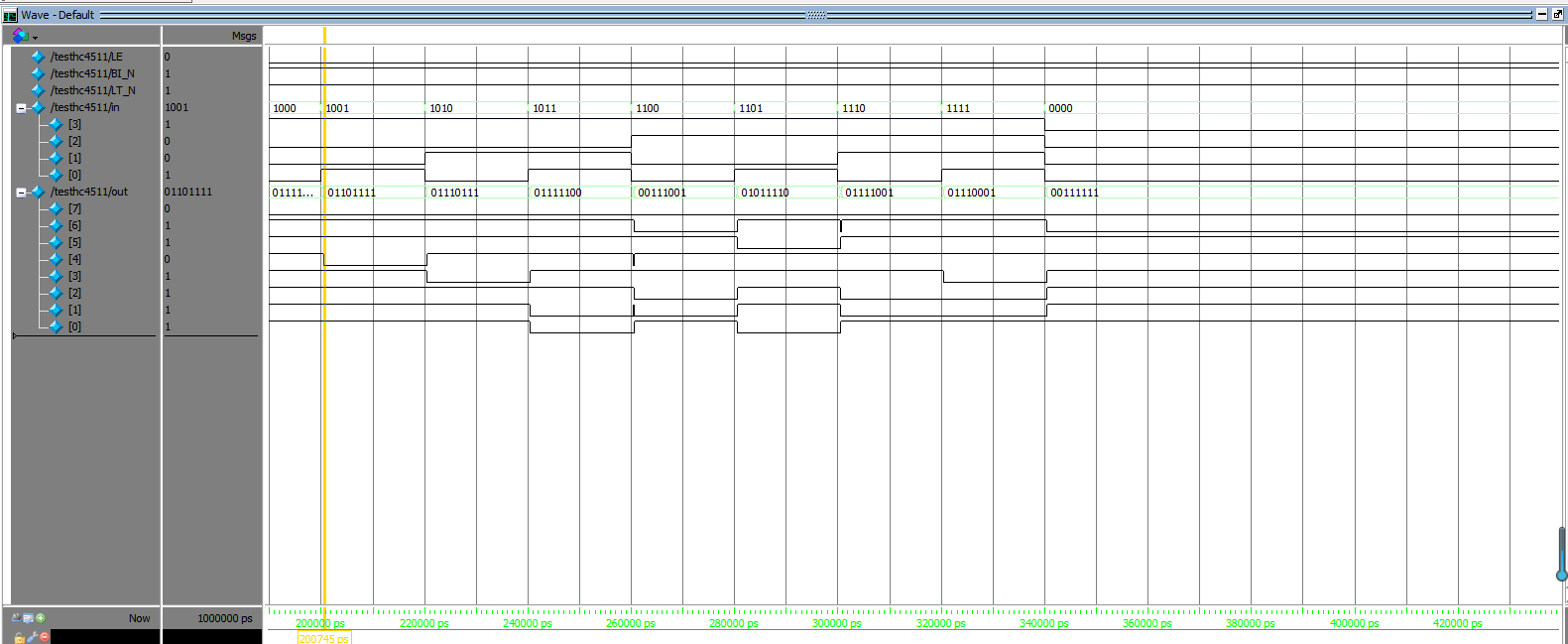
延迟0.61ns



74HC283

延迟400ps

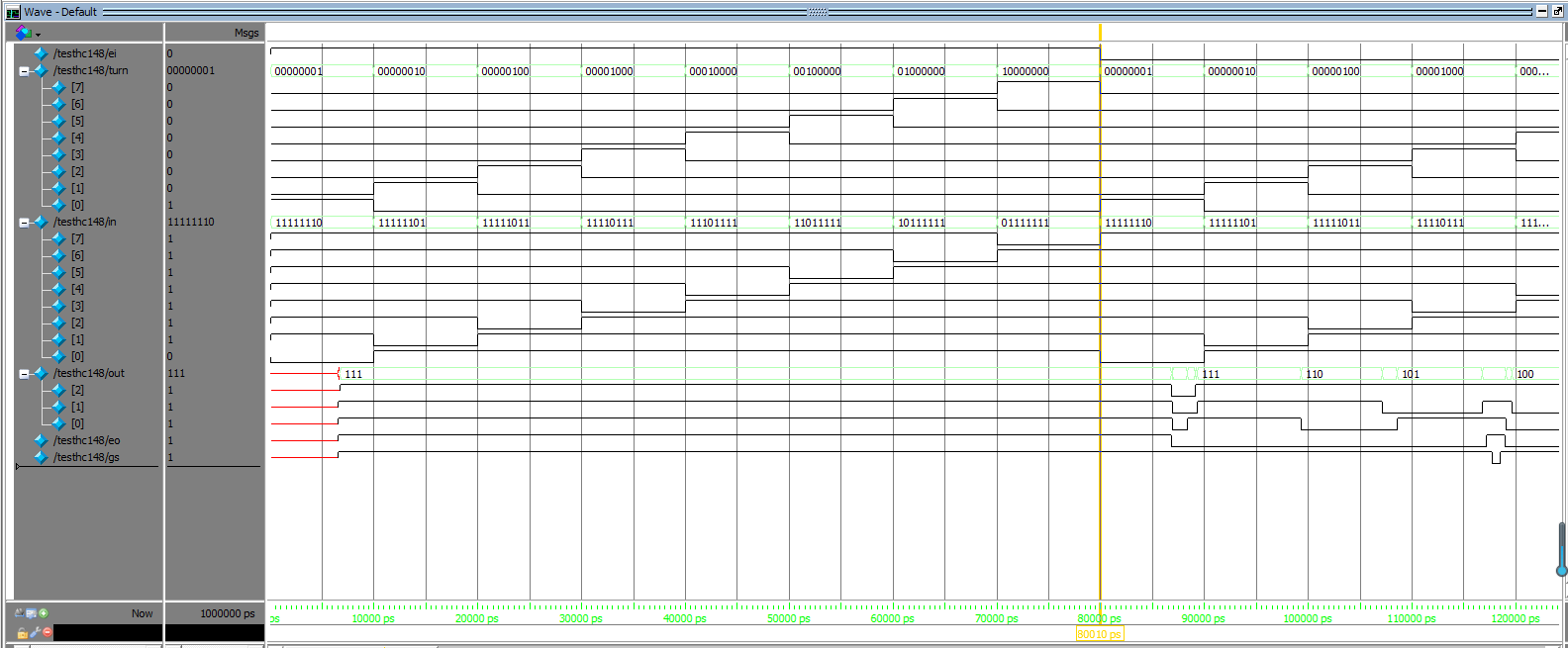


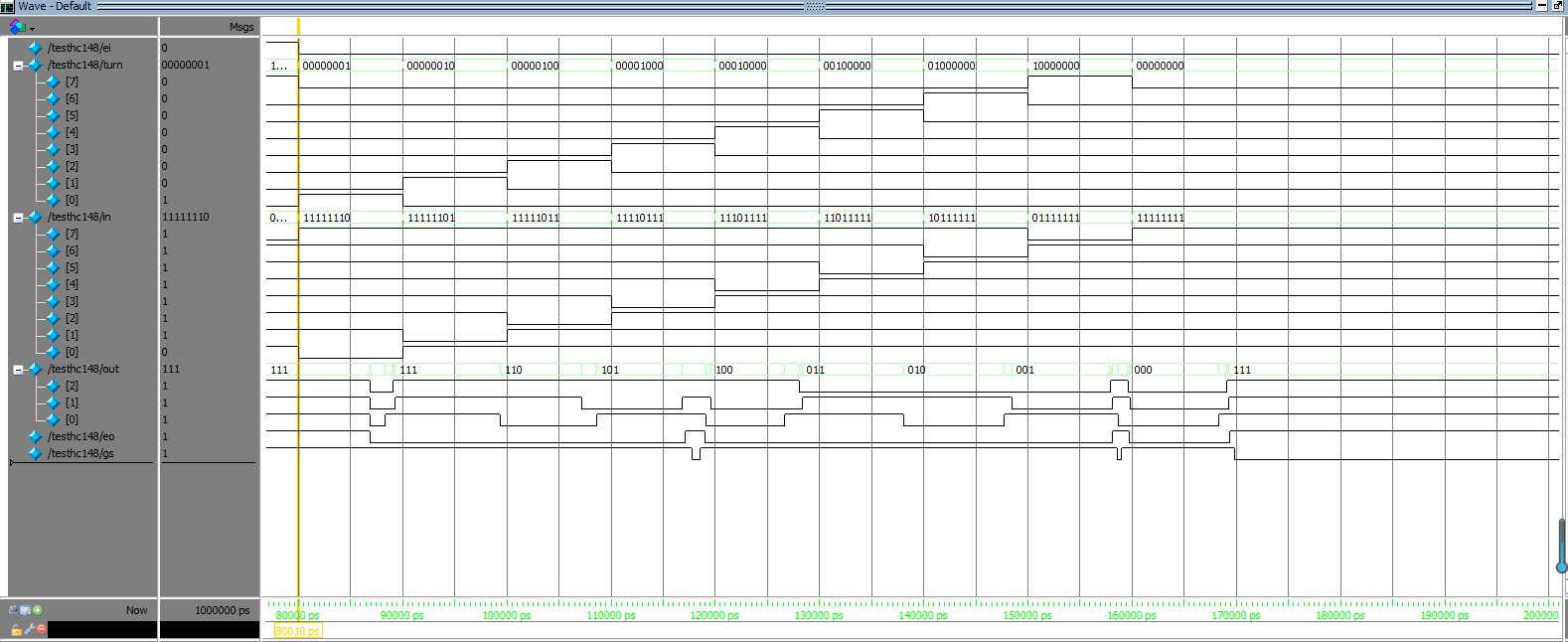


74HC4511

延迟500ps

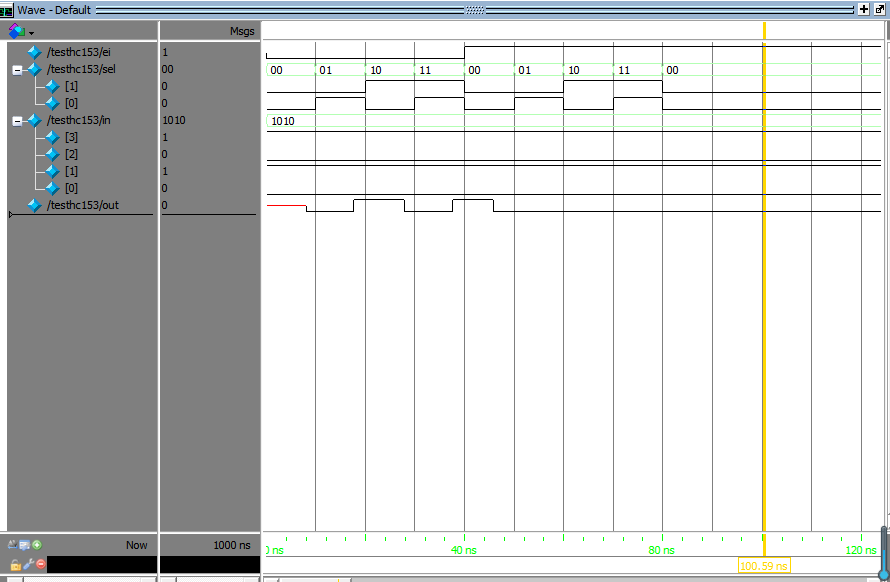
5、第三次仿真结果（布局布线后）（**截图，注明对应的模块**）。回答输出信号是否有延迟，延迟时间约为多少？分析是否有出现竞争冒险。





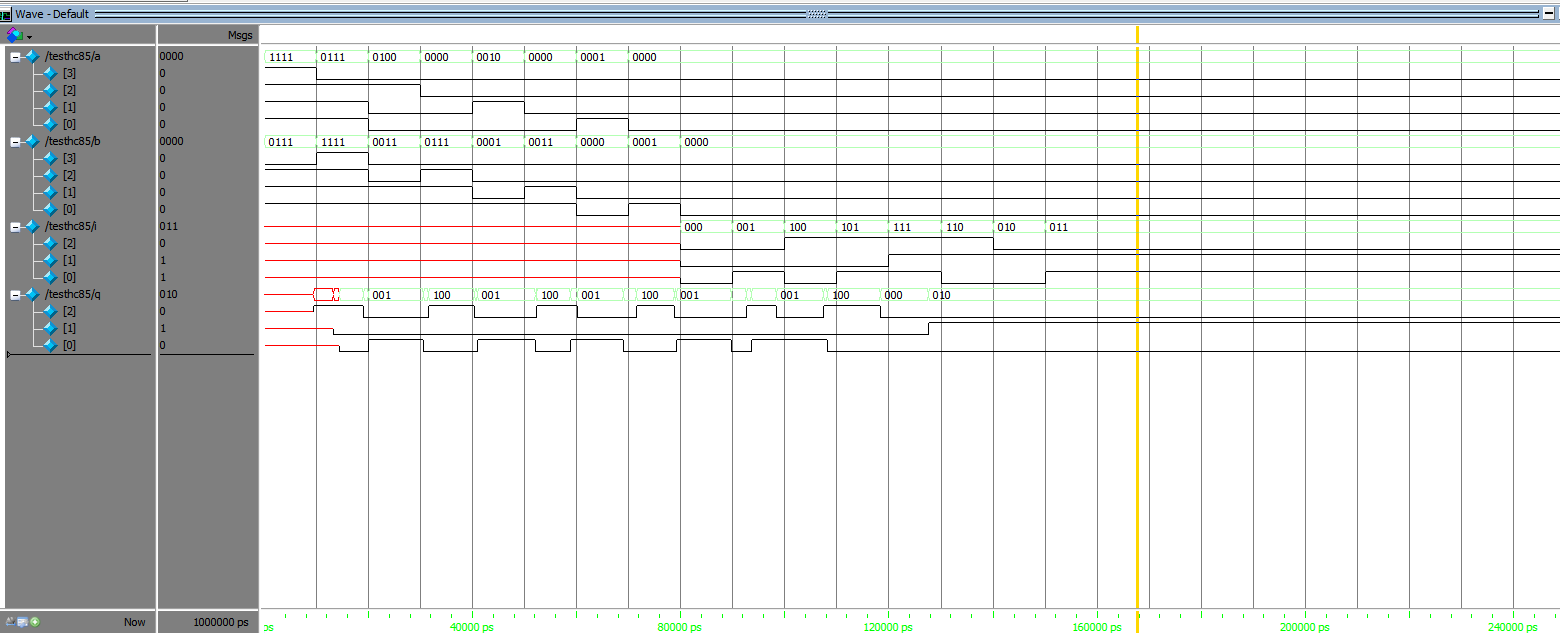
74HC148

延迟6689ps



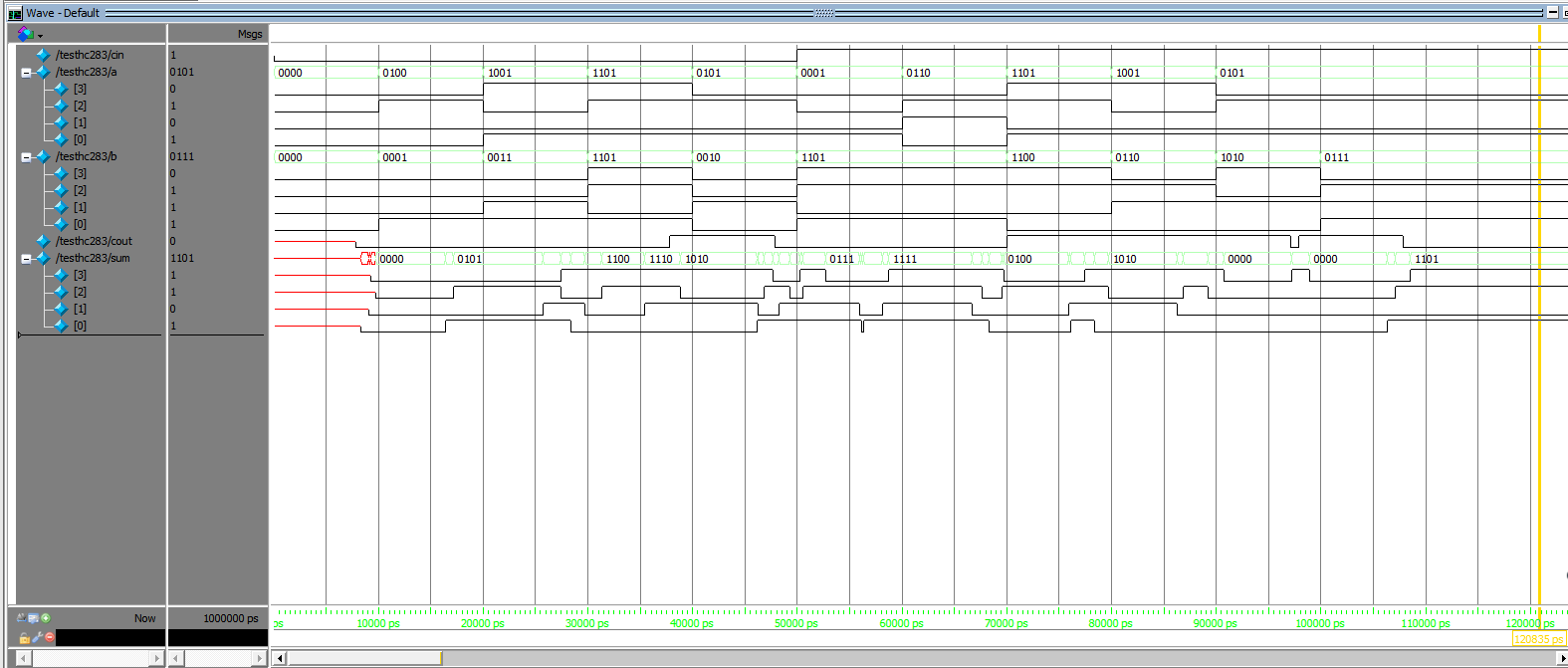
74HC153

延迟8.219ns



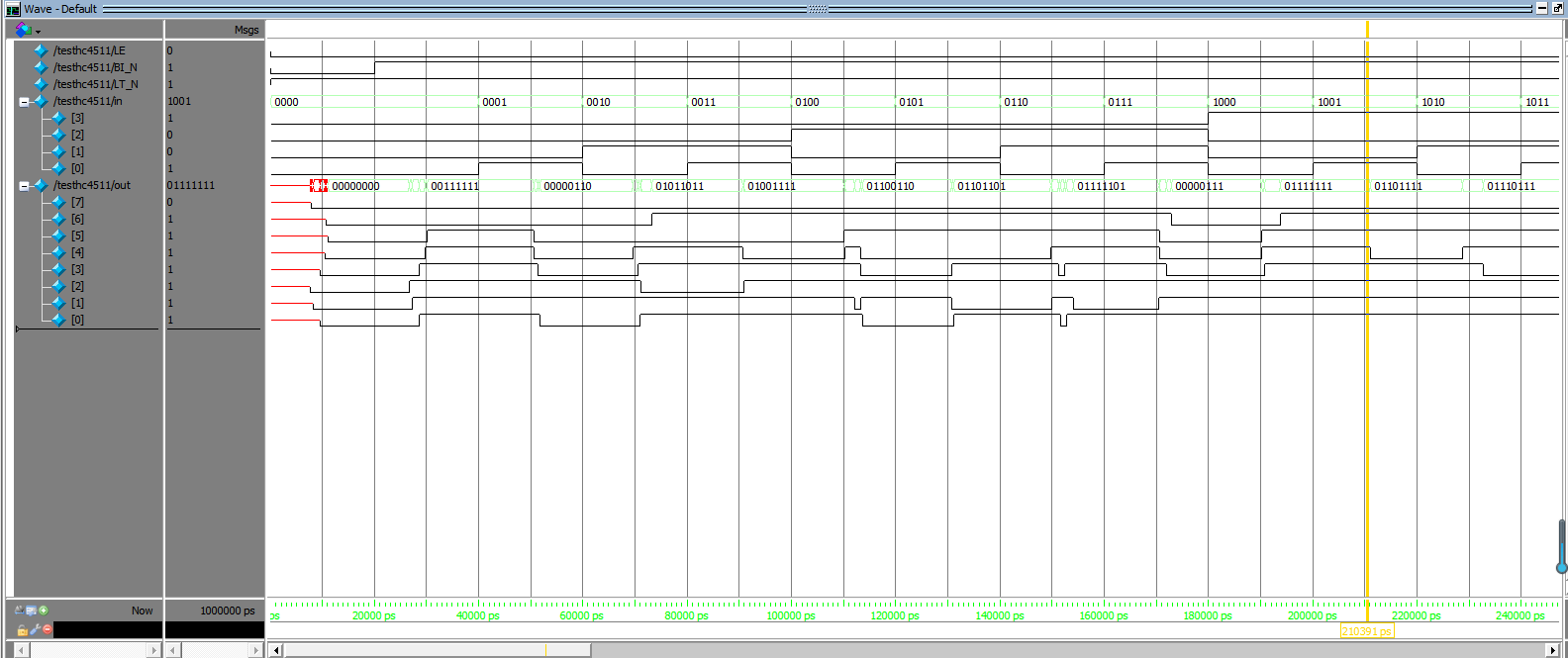
74HC85

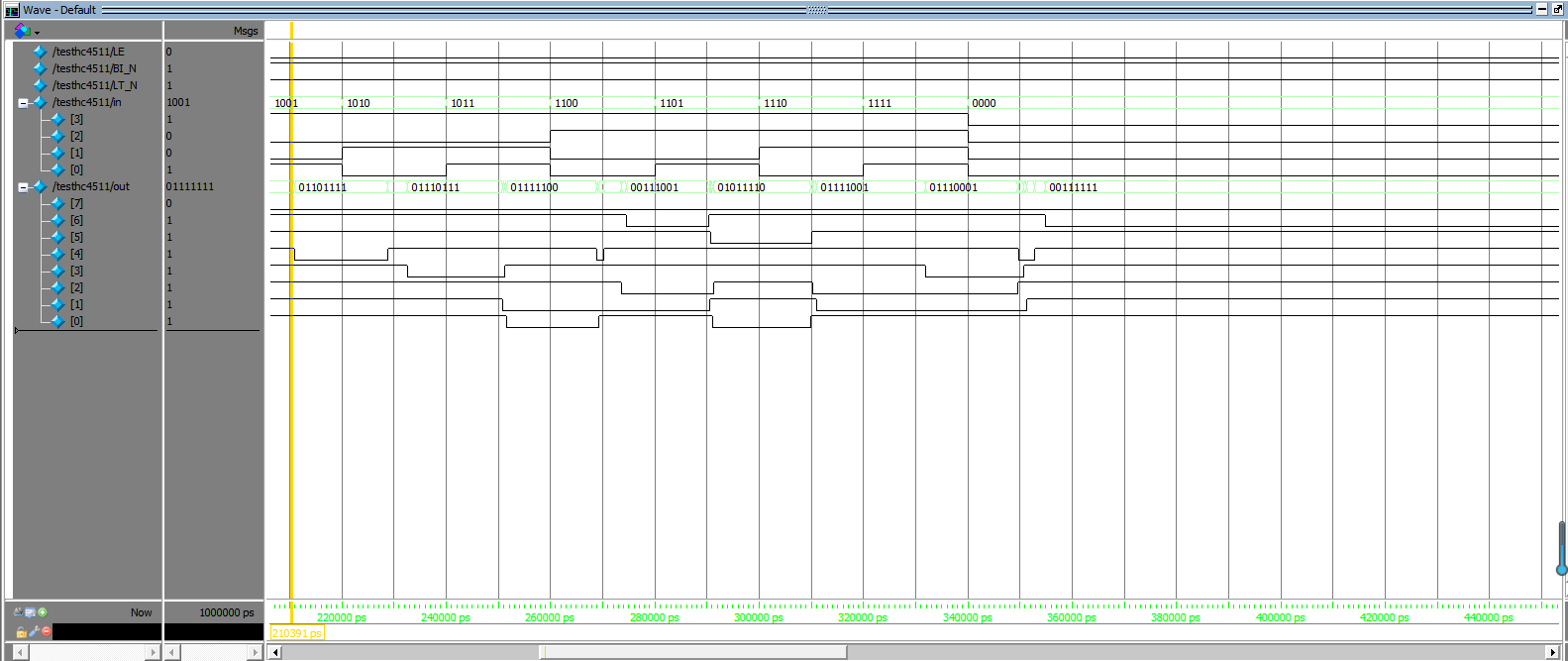
延迟14497ps



74HC283

延迟9753ps





74HC4511

延迟11092ps

6、烧录验证：典型实验数据拍照、录像**（课堂给老师检查，课后全班统一刻光盘）**。

## 时序逻辑电路

一、实验目的

1、了解基于Verilog的时序逻辑电路的设计及其验证。

2、熟悉利用EDA工具进行设计及仿真的流程。

3、熟悉实验箱的使用和程序下载（烧录）及测试的方法。

4、学习针对实际时序逻辑电路芯片74HC74、74HC112、74HC194、74HC161进行VerilogHDL设计的方法。

二、实验环境及仪器

1、Libero仿真软件。

2、数字逻辑与系统设计实验箱。

3、Actel A3P060 FPGA芯片及Flash Pro5烧录器。

三、实验内容

1、熟练掌握Libero软件的使用方法。

2、进行针对74系列时序逻辑电路的设计，并完成相应的仿真实验。

3、参考教材中相应章节的设计代码、测试平台代码（可自行编程），完成74HC74、74HC112、74HC161、74HC194相应的设计、综合及仿真。

4、提交针对74HC74、74HC112、74HC161、74HC194的综合结果，以及相应的仿真结果。

四、实验结果和数据处理

1、所有模块及测试平台代码清单

//74HC74代码

module hc74(Sn,Rn,clk,D,Q,Qn);

input Sn,Rn,clk,D;

output Q,Qn;

reg Q,Qn;

always@(posedge clk,negedge Sn,negedge Rn)

begin

case({Sn,Rn})

'b01: begin Q=1;Qn=0;end

'b10: begin Q=0;Qn=1;end

'b00: begin Q=1;Qn=1;end

'b11: begin Q=D;Qn=~D;end

endcase

end

endmodule

//74HC74测试平台代码

`timescale 1ns/1ns

module testhc74();

reg s, r, clk, d;

wire q, qn;

hc74 test74hc74(s, r, clk, d, q, qn);

task clock;

repeat(10)

begin

d = 0; #3;

d = 1; #3;

end

endtask

always

begin

clk = 0;#2;

clk = 1;#2;

end

initial

begin

s = 0;r = 1;clock;

#5 s = 1;r = 0;clock;

#5 s = 0;r = 0;clock;

#5 s = 1;r = 1;clock;

end

endmodule

//74HC112代码

module hc112(Sn,Rn,clk,J,K,Q,Qn);

input Sn,Rn,clk,J,K;

output Q,Qn;

reg Q,Qn;

always@(negedge clk,negedge Sn,negedge Rn)

begin

case({Sn,Rn})

2'b01:Q<=1,Qn<=0;

2'b10:Q<=0,Qn<=1;

2'b00:Q<=1,Qn<=0;

2'b11:

begin

case({J,K})

2'b00:Q<=Q,Qn<=Qn;

2'b01:Q<=0,Qn<=1;

2'b10:Q<=1,Qn<=0;

2'b11:Q<=~Q,Qn<=~Qn;

endcase

end

endcase

end

endmodule

//74HC112测试平台代码

`timescale 1ns/1ns

module testhc112();

reg sn, rn, clk, j, k;

wire q, qn;

hc112 test74hc112(sn, rn, clk, j, k, q, qn);

always

#5 clk = ~clk;

task clock;

repeat(3)

begin

j = 0;k = 1;#20;

j = 1;k = 0;#20;

j = 0;k = 0;#20;

j = 1;k = 1;#20;

end

endtask

initial

begin

clk = 0;

sn=0;rn=0;clock;

sn=0;rn=1;clock;

sn=1;rn=0;clock;

sn=1;rn=1;clock;

end

endmodule

//74HC161代码

module HC161(MR,CP,CEP,CET,PE,D,Q,TC);

input MR,CP,CEP,CET,PE;

input [3:0]D;

output [3:0]Q;

output TC;

reg [3:0]Q;

assign TC = &{CET, Q};

always @(negedge MR, posedge CP) begin

if(!MR)

Q<=0;

else if(CEP & CET & PE)

Q<=Q+1;

else if(!PE)

Q<=D;

end

endmodule

//74HC161测试平台代码

`timescale 1ns/1ns

module bench;

reg CP,CEP,CET,a,PEN;

reg [3:0]Dn;

wire TC;

wire [3:0]Qn;

HC161 test(a,CP,CEP,CET,PEN,Dn,Qn,TC);

parameter clock\_period=20;

always #(clock\_period/2) CP=~CP;

initial

begin

CP=0;

CEP=1; CET=1; a=0; PEN=0;

#10 a<=1;

#10 Dn<='b0010;

end

always @(negedge CP) begin

if(Qn=='b1111) begin

#10 a<=0;

#10 Dn<=0;

end

if(Qn=='b0010) PEN<=1;

end

endmodule

//74HC194代码

module HC194(MR,S,Clk,D\_SR,D\_SL,D,Q);

input MR,Clk,D\_SR,D\_SL;

input[1:0] S;

input[0:3] D;

output[0:3] Q;

reg[0:3] Q;

always @(posedge Clk)

begin

if(~MR) Q=0;

else

begin

case(S)

1:Q={D\_SR,Q[0:2]};

2:Q={Q[1:3],D\_SL};

3:Q=D;

default: Q=Q;

endcase

end

end

endmodule

//74HC194测试平台代码

`timescale 1ns/1ns

module testbench();

reg mr,clk,d\_sr,d\_sl;

reg[1:0] s;

reg[0:3] d;

wire[0:3] q;

HC194 hc194(mr,s,clk,d\_sr,d\_sl,d,q);

initial

begin

mr=0;

#20 mr=1;

end

initial

begin

s=3; d=4'b1011;

repeat(6)

begin

#50 s=$random;

end

end

initial

begin

clk=0;

repeat(60)

#5 clk=~clk;

end

initial

begin

d\_sr=0; d\_sl=1;

repeat(6)

begin

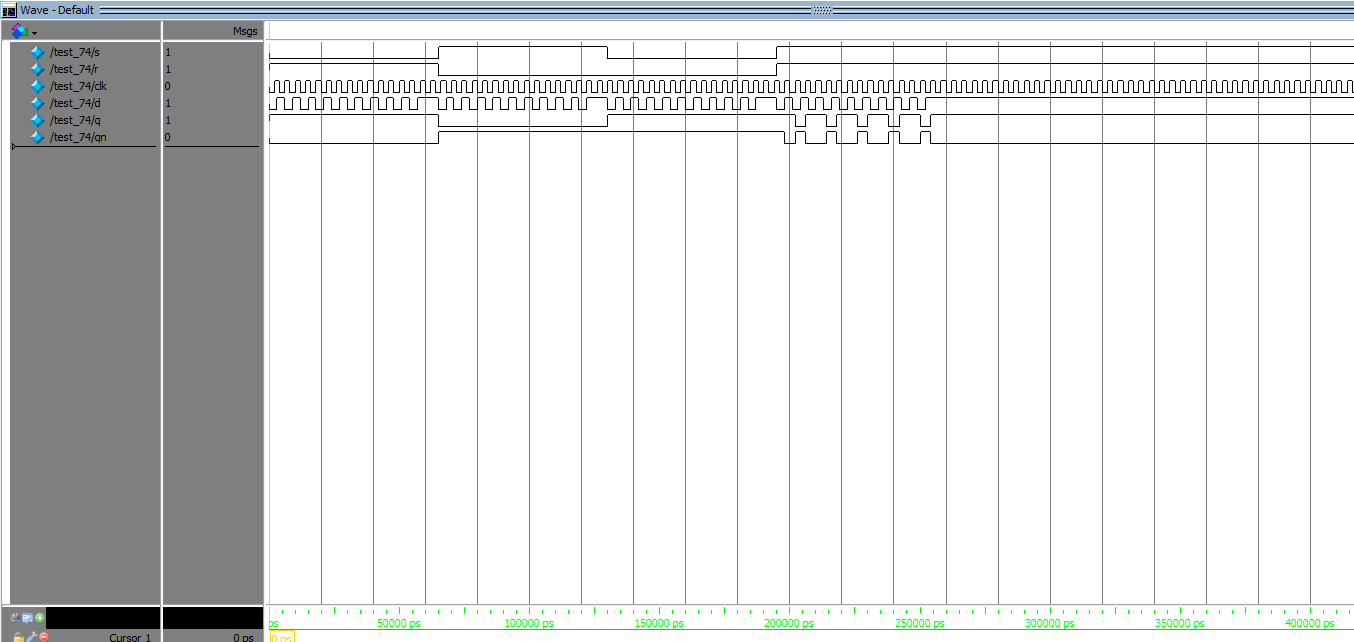
d\_sr=$random; d\_sl=$random;

end

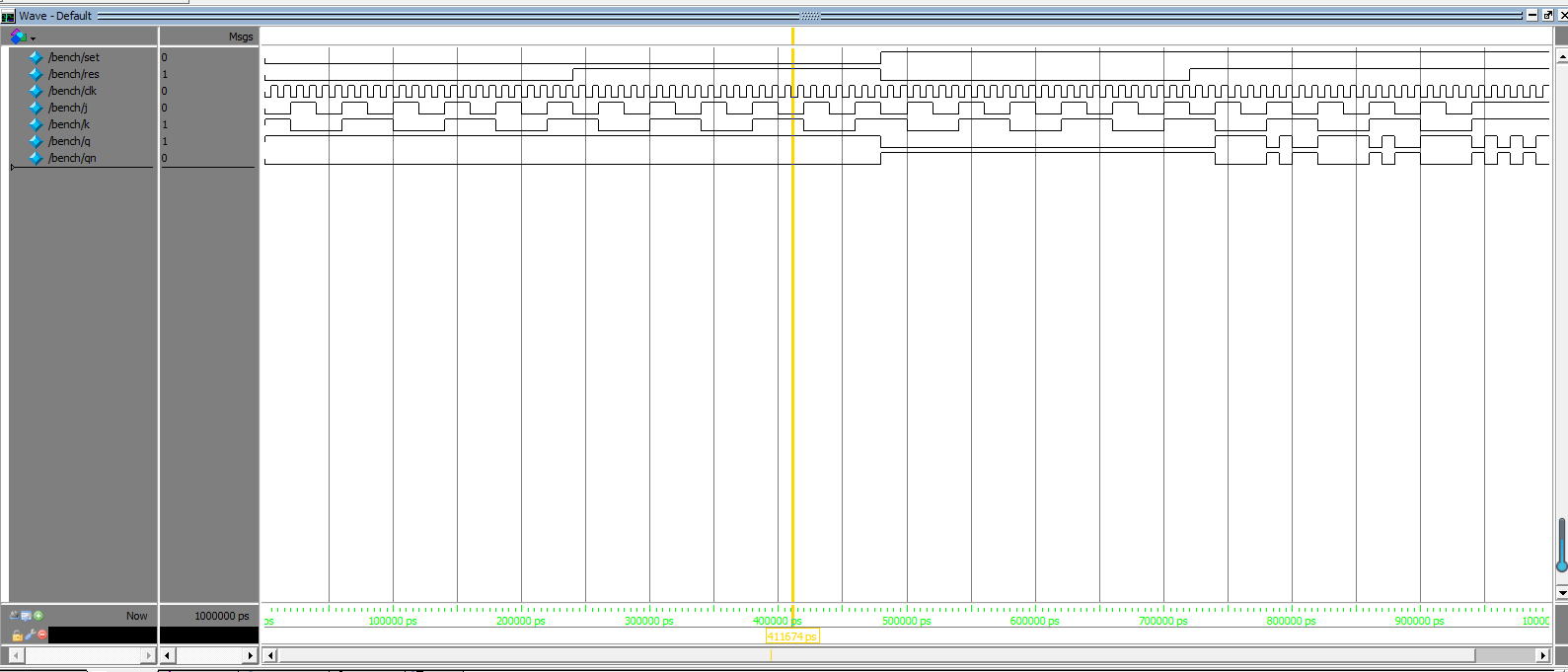
end

endmodule

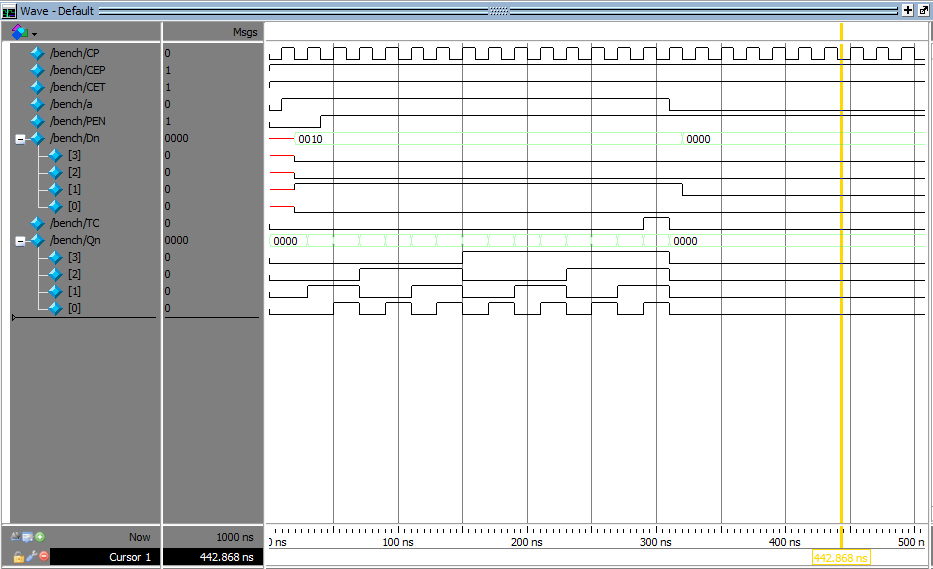
2、第一次仿真结果（**截图，注明对应的模块**）



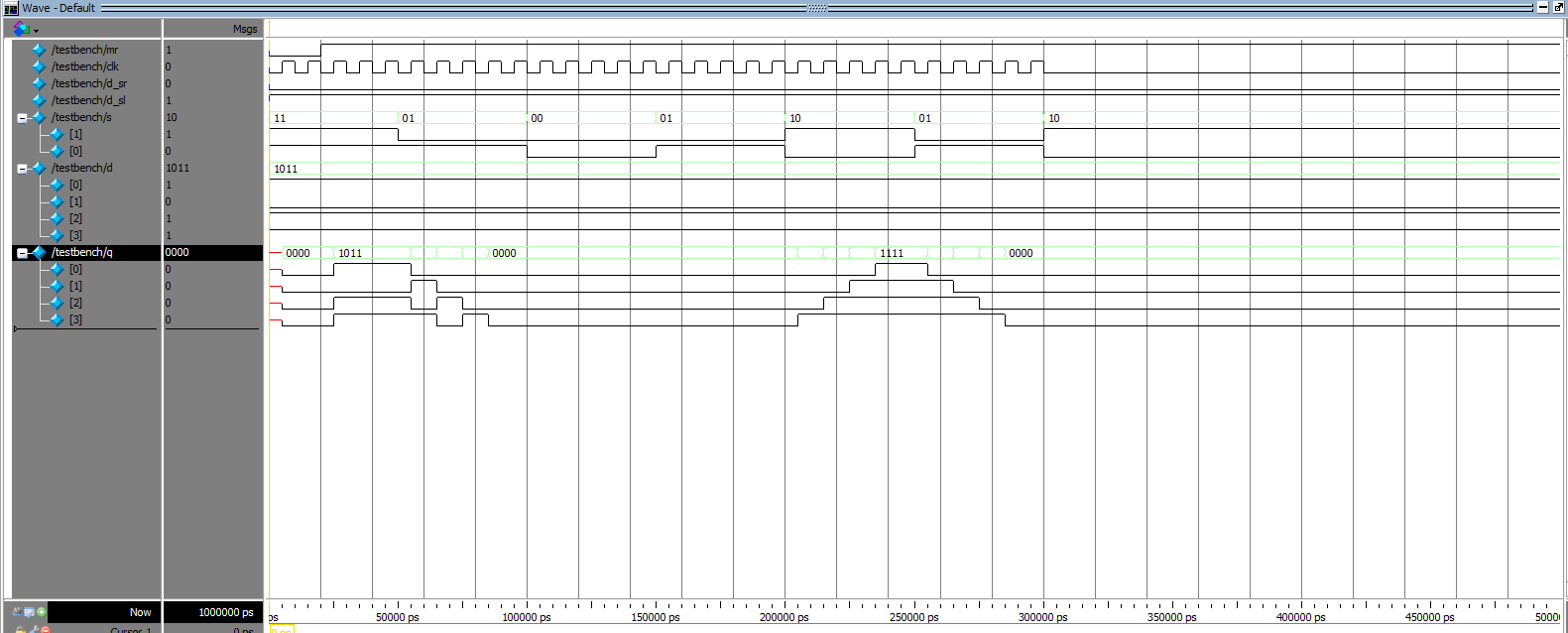
74HC74



74HC112

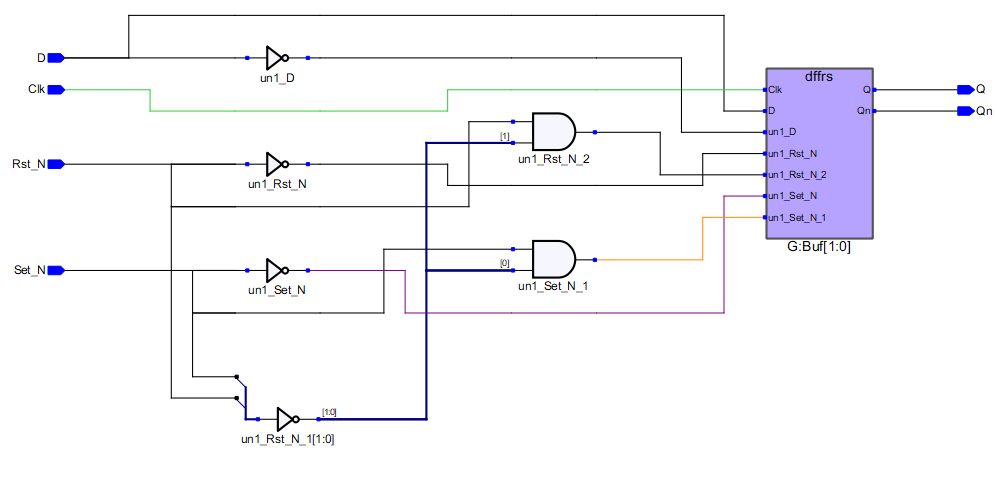


74HC161

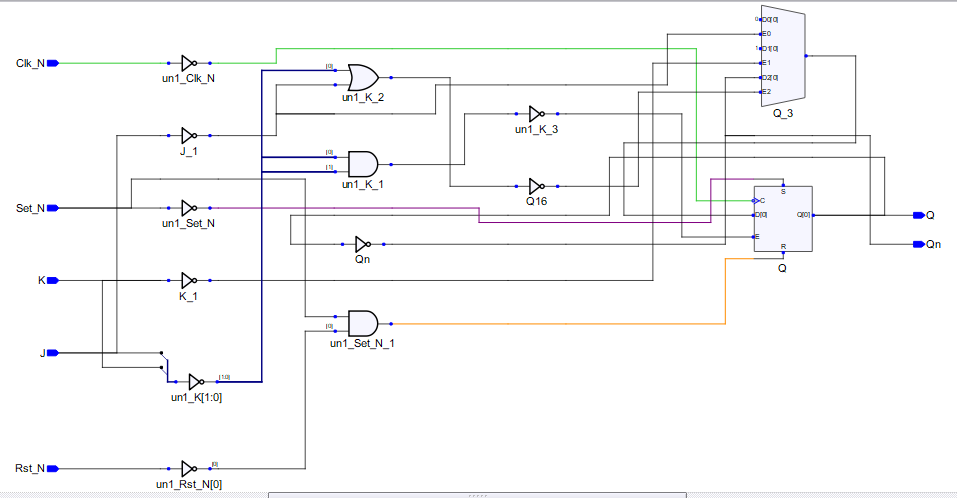


74HC194

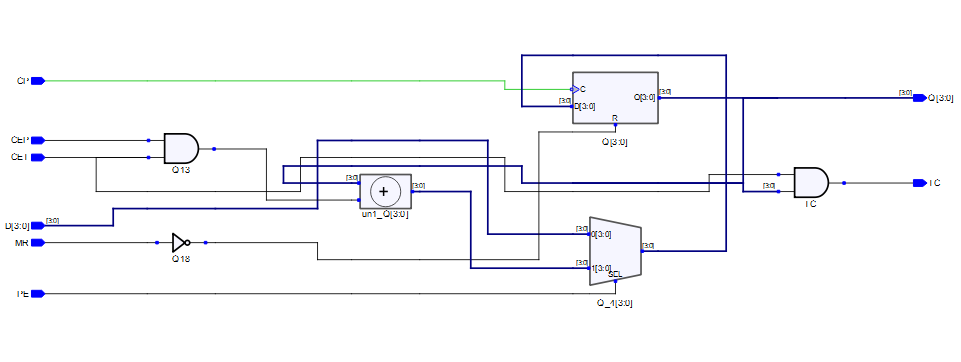
3、综合结果（**截图，注明对应的模块**）



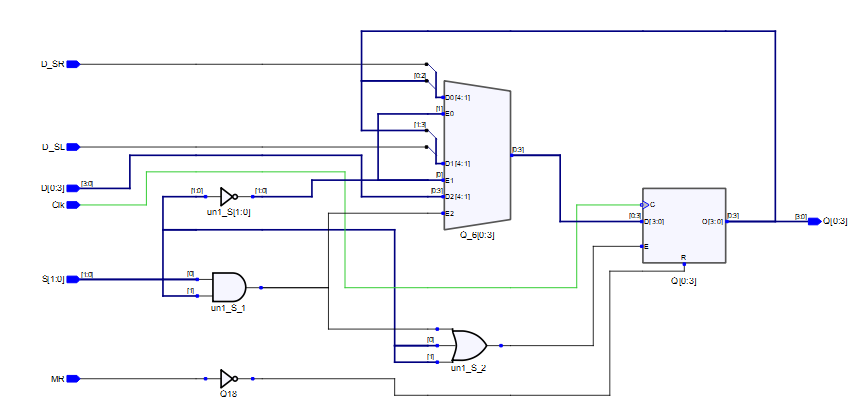
74HC74



74HC112

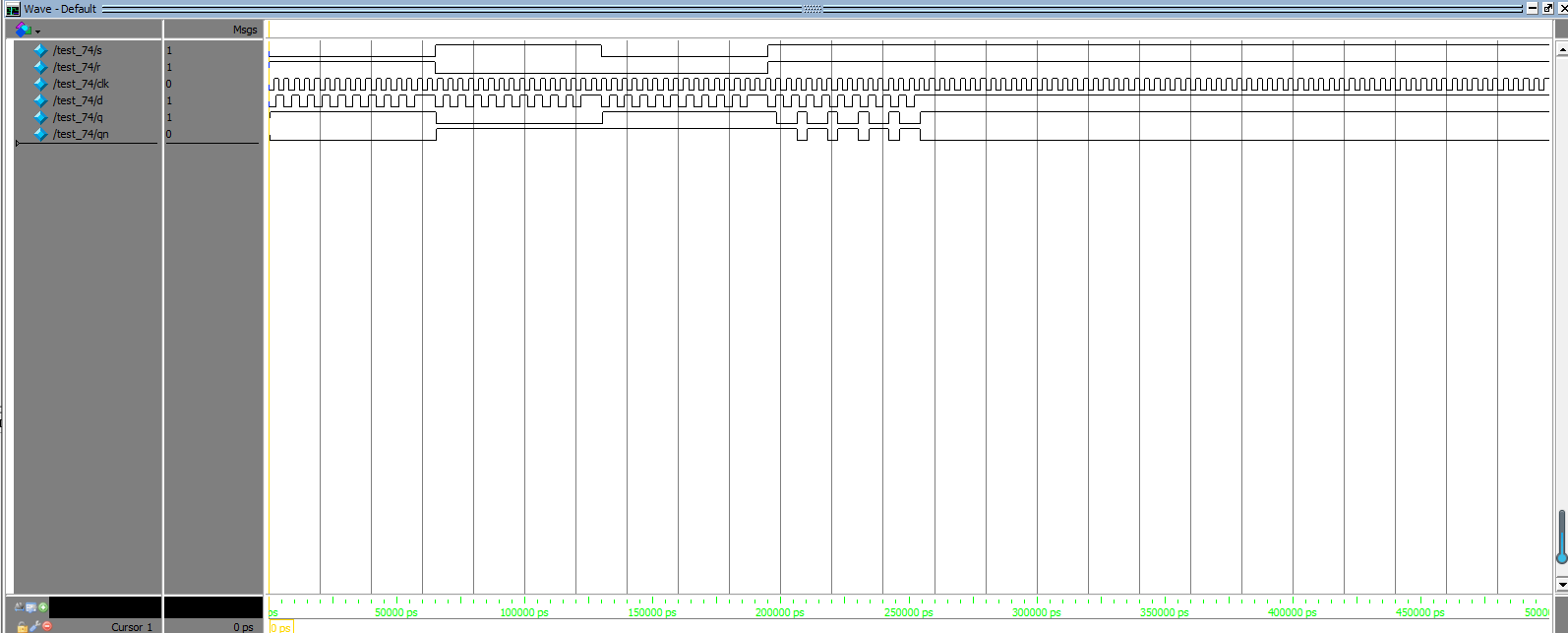


74HC161



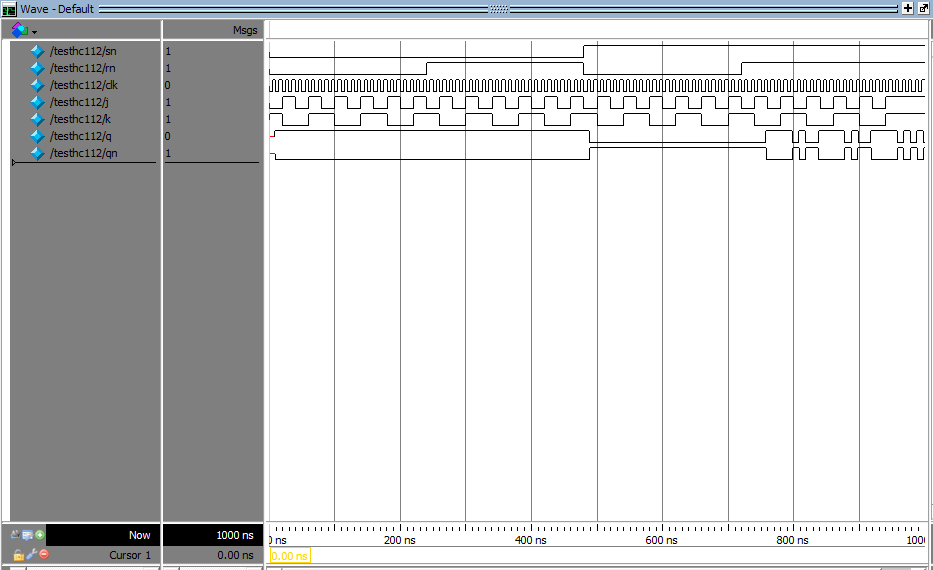
74HC194

4、第二次仿真结果（综合后）（**截图，注明对应的模块**）。

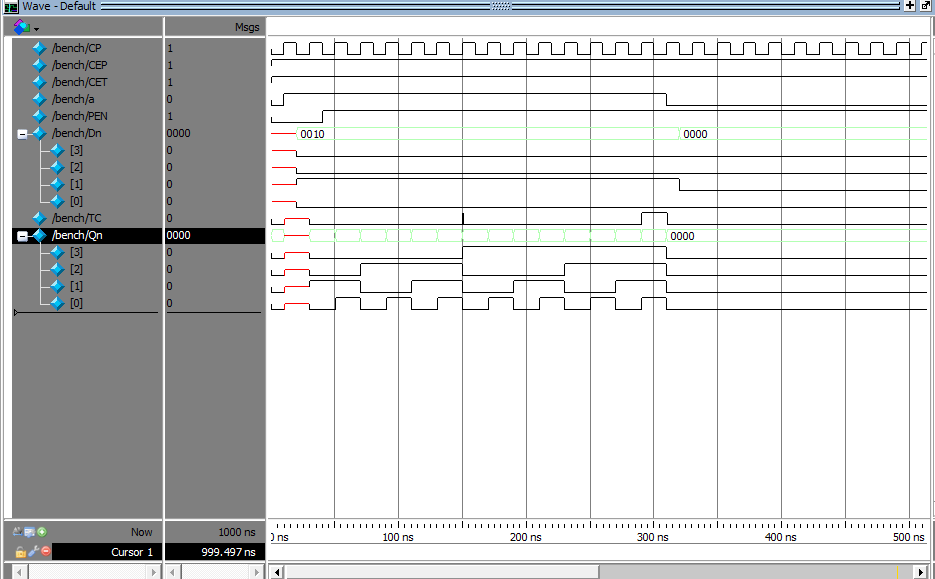


74HC74

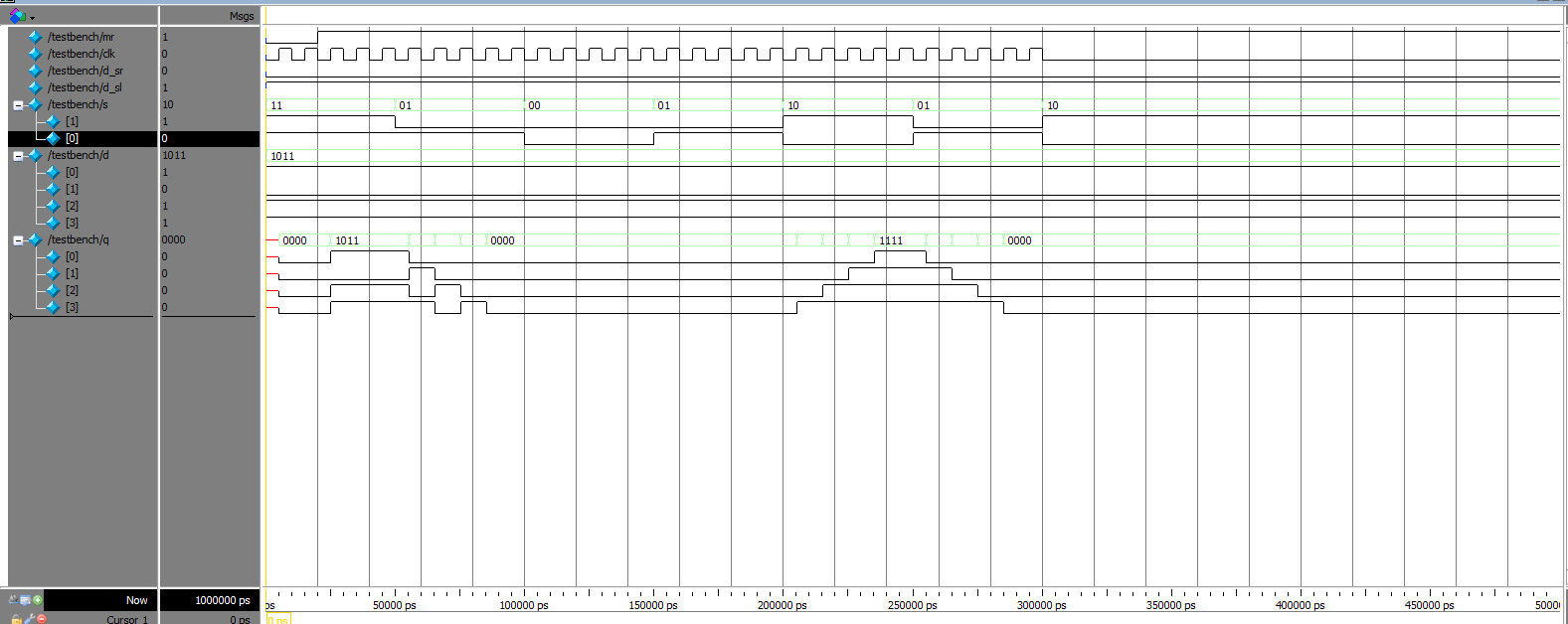
延迟400ps



74HC112

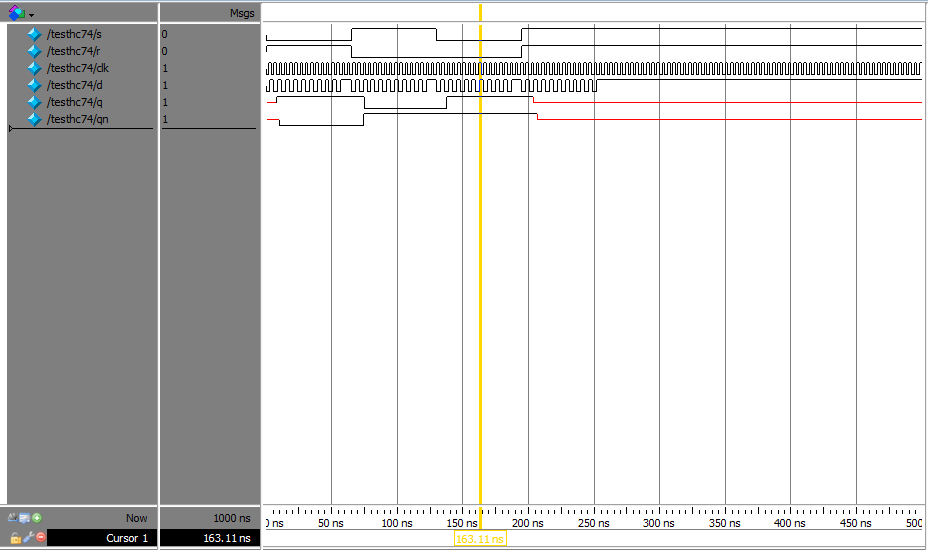


74HC161

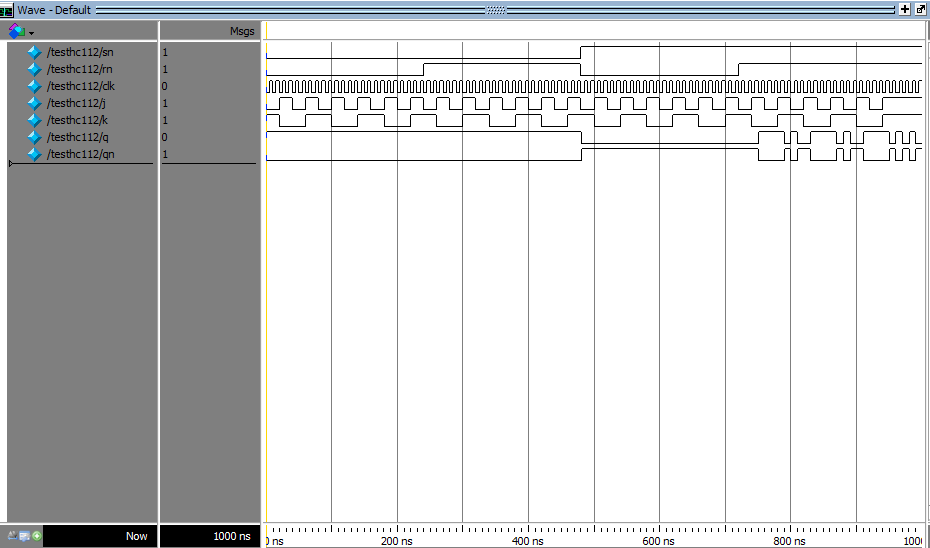


74HC194

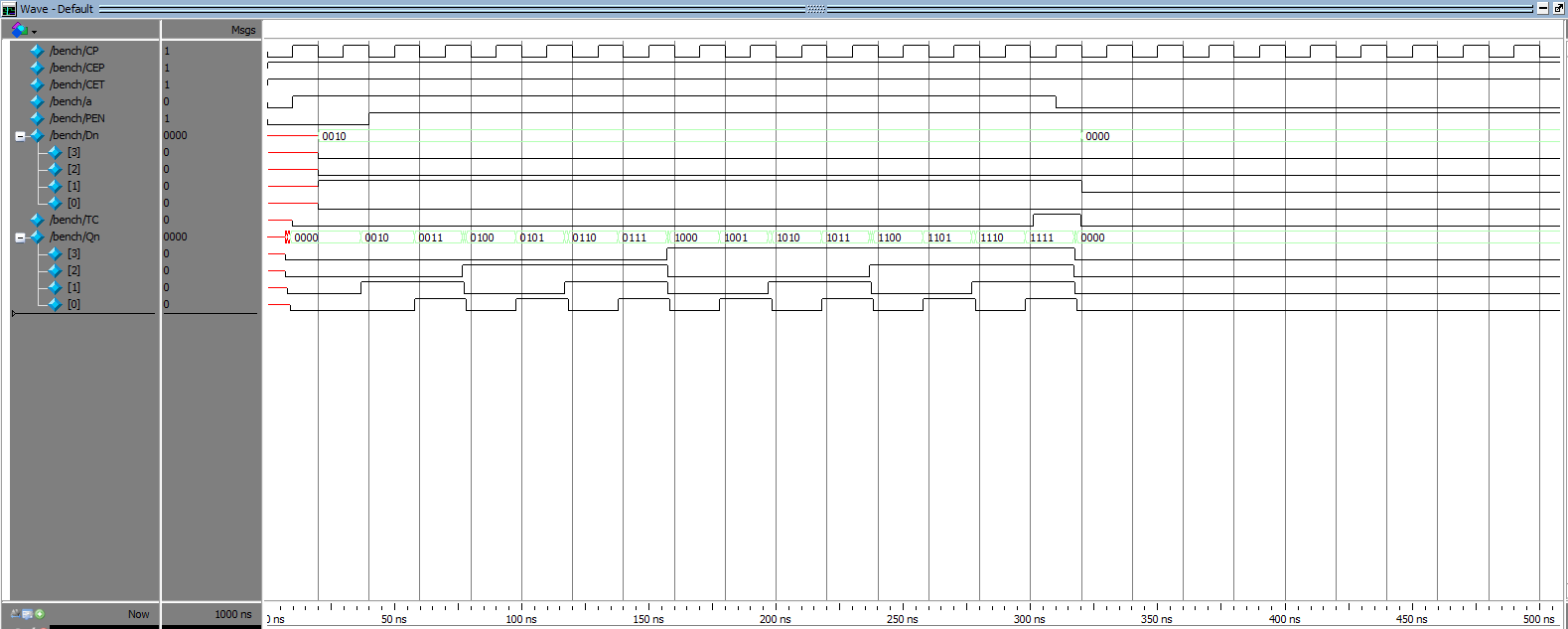
5、第三次仿真结果（布局布线后）（**截图，注明对应的模块**）。



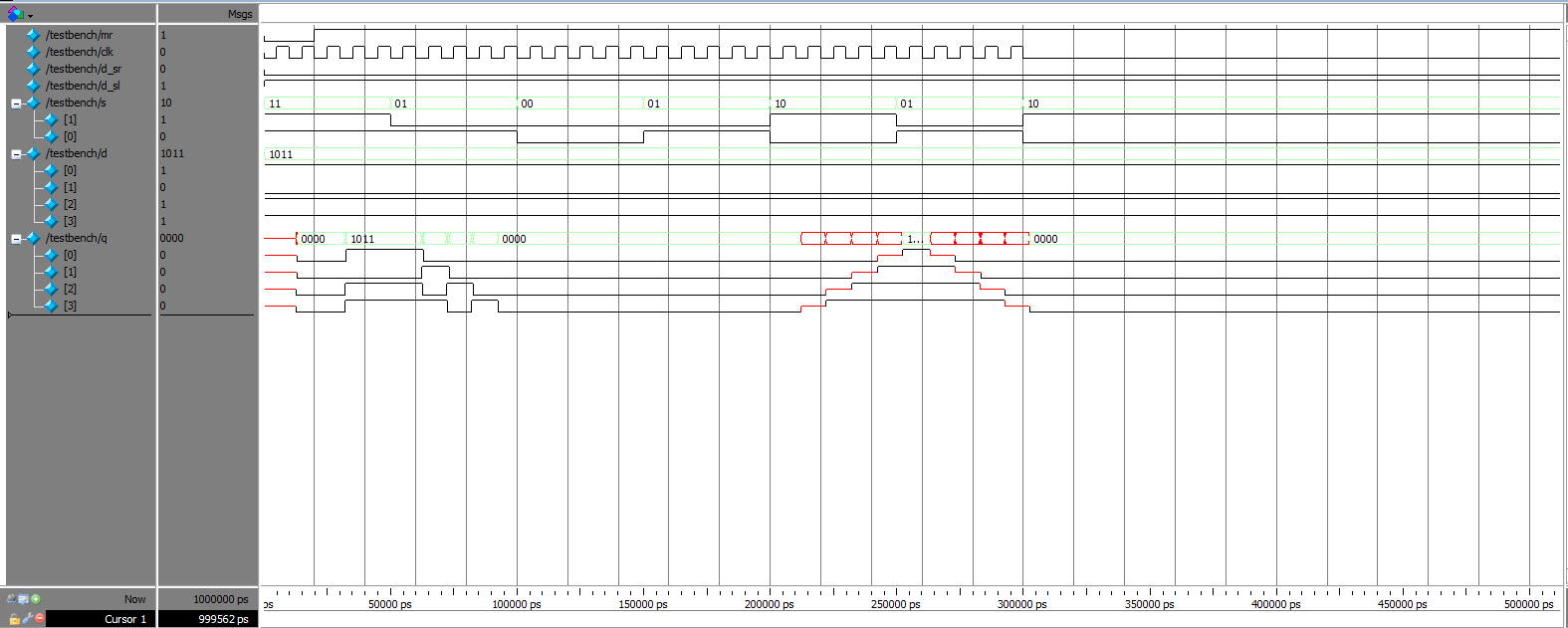
74HC74



74HC112



74HC161



74HC194

6、烧录验证：典型实验数据拍照、录像**（课堂给老师检查，课后全班统一刻光盘）**。

## 综合实验

一、实验目的

1、进一步熟悉利用EDA工具进行设计及仿真的流程。

2、熟悉利用EDA工具中的图形化设计界面进行综合设计。

3、熟悉芯片烧录的流程及步骤。

4、掌握分析问题、解决问题的综合能力，通过EDA工具设计出能解决实际问题的电路。

二、实验环境

1、Libero仿真软件。

2、数字逻辑与系统设计实验箱。

3、Actel A3P060 FPGA芯片及Flash Pro5烧录器。

三、实验内容

按老师要求在以下题目中选做，并按要求完成实验报告。

综合实验1——编码器扩展实验：当按下小于等于9的按键后，显示数码管显示数字，当按下大于9的按键后，显示数码管不显示数字。若同时按下几个按键，优先级别的顺序是9到0。本实验需要两个编码器74HC148、一个数码显示译码器74HC4511、一个共阴极8段显示数码管LN3461Ax和一个数值比较器74HC85。利用Libero SmartDesign图形化设计工具，采用图文混合设计方法进行设计。

综合实验2——译码器扩展实验：设计一个电路，通过改变输入，令显示数码管的4个数位轮流显示数字。本实验需要一个3-8译码器74HC138、一个数码显示译码器74HC4511、一个共阴极8段显示数码管LN3461Ax。将译码器74HC138的输入显示在数码管LN3461Ax上，并利用译码器74HC138的输出控制数码显示译码器74HC4511的工作（、或LE中任一个）。

综合实验3——有符号比较器实验：设计一个电路，比较两个8位有符号数的大小，判定是否满足大于等于的关系。方法：直接利用Libero工具提供的比较器IP核，实现一个有符号比较器。

综合实验4——4位有符号二进制加法器。

综合实验5——二——十进制码转换电路：设计一个能实现8位二进制码转换为12位8421BCD码的电路。

综合实验6——利用状态机实现一个简单自动售货机控制电路（顶层结构框图如图7-23）。该电路有两个投币口（1元和5角），商品2元一件，不设找零。In[0]表示投入5角，In[1]表示投入1元，D\_out表示是否提供货品。分别用Mealy和Moore型实现。

综合实验7——与寄存器结合的有限状态机：将寄存器逻辑（利用时钟信号同步进行赋值）与Mealy或Moore状态机组合起来，可以得出两种解决方案。

综合实验8——跑马灯：设计要求：共8个LED灯连成一排，用以下3种模式来显示，模式选择使用两个按键进行控制。

① 模式1：先点亮奇数灯，即1、3、5、7灯亮，然后偶数灯，即2、4、6、8灯亮，依次循环，灯亮的时间按时钟信号的二分频设计。

② 模式2：按照1、2、3、4、5、6、7、8的顺序依次点亮所有灯；然后再按1、2、3、4、5、6、7、8的顺序依次熄灭所有灯，间隔时间按时钟信号的八分频设计。

③ 模式3：按照1/8、2/7、3/6、4/5的顺序依次点亮所有灯，每次同时点亮两个灯；然后再按1/8、2/7、3/6、4/5的顺序熄灭相应灯，每次同时熄灭两个灯，灯亮的时间按时钟信号的四分频设计。

④ 模式4：自行设计。

综合实验9——四位数码管扫描显示电路的设计

设计要求：共4个数码管，连成一排，要求可以显示其中任意一个数码管。具体要求如下：

① 依次选通4个数码管，并让每个数码管显示相应的值，其结果由相应输入决定（通过扩展板按键输入）。

② 要求能在实验箱上演示出数码管的动态显示过程。必须使得4个选通信号DIG1、DIG2、DIG3、DIG4轮流被单独选通，同时，在段信号输入口加上希望在对应数码管上显示的数据，这样随着选通信号的变化，才能实现扫描显示的目的。

综合实验10——交通灯控制器

设计要求：实现一个常见的十字路口交通灯控制功能。一个十字路口的交通灯一般分为两个方向，每个方向具有红灯、绿灯和黄灯三种。实现一个常见的十字路口交通灯控制功能，具体要求如下：

① 十字路口包含A、B两个方向的车道。A方向放行一分钟（绿55秒，黄5秒），同时B方向禁行（红60秒）；然后A方向禁行1分钟（红60秒），同时B方向放行（绿55秒，黄5秒），示意图如图5-56所示。依此类推，循环往复。

② 实现正常的倒计时功能，用两组数码管作为A和B两个方向的倒计时显示。

③ 系统时钟1KHz。

综合实验11——键盘扫描器和编码器

设计要求：

① 检测是否有按键按下；

② 确定按键的位置；

③ 生成唯一按键编码；

④ 在数码管上显示相应的按键内容。

1. 实验结果和数据处理

综合实验4——4位有符号二进制加法器。

代码：

module Add\_prop\_gen(sum,c\_out,a,b,c\_in,shiftedcarry);

output[3:0] sum;

output[4:0] shiftedcarry;

output c\_out;

input[3:0]a,b;

input c\_in;

reg[3:0]carrychain;

wire[3:0] g=a&b;

wire[3:0] p=a^b;

always@(a,b,c\_in,p,g)

begin:carry\_generation

integer i;

carrychain[0]=g[0]+(p[0]&c\_in);

for(i=1;i<=3;i=i+1)

carrychain[i]=g[i]|(p[i]&carrychain[i-1]);

end

wire[4:0]shiftedcarry={carrychain,c\_in};

wire[3:0]sum=p^shiftedcarry;

wire c\_out=shiftedcarry[4];

endmodule

module mux2(D0,D1,Sel0,Result);

input D0,D1,Sel0;

output Result;

reg Result;

always@(D0,D1,Sel0)

begin

if(Sel0)

Result=D1;

else Result=D0;

end

endmodule

测试平台：

`timescale 1ns/10ps

module testbench;

reg[3:0] ina,inb;

reg cin;

wire s3,s2,s1,s0,cout;

wire[3:0]sum;

assign sum={s3,s2,s1,s0};

adder4\_of2 testadder(.a(ina),.b(inb),.c\_in(cin),.S3(s3),.S2(s2),.S1(s1),.S0(s0),.c\_out(cout));

initial

begin

ina=0;

repeat(200)

#40 ina=$random;

end

initial

begin

inb=0;

repeat(100)

#80 inb=$random;

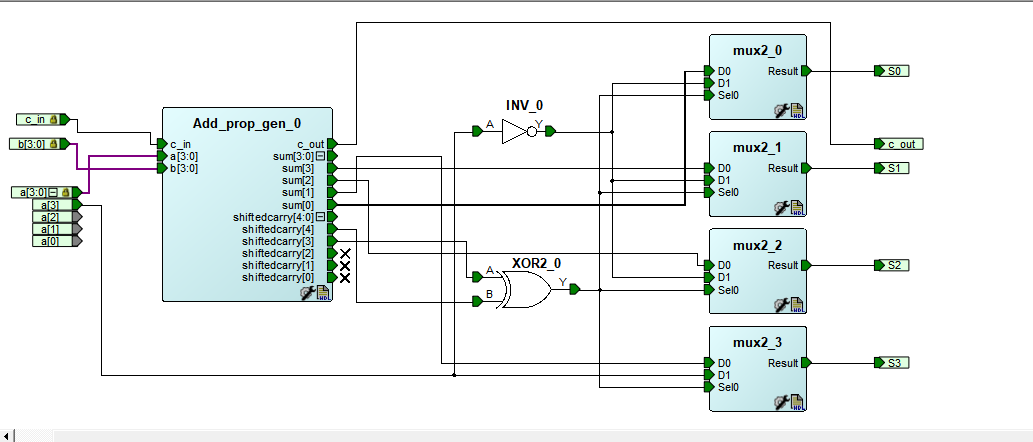
end

initial

begin cin=0;#200 cin=1;end

endmodule

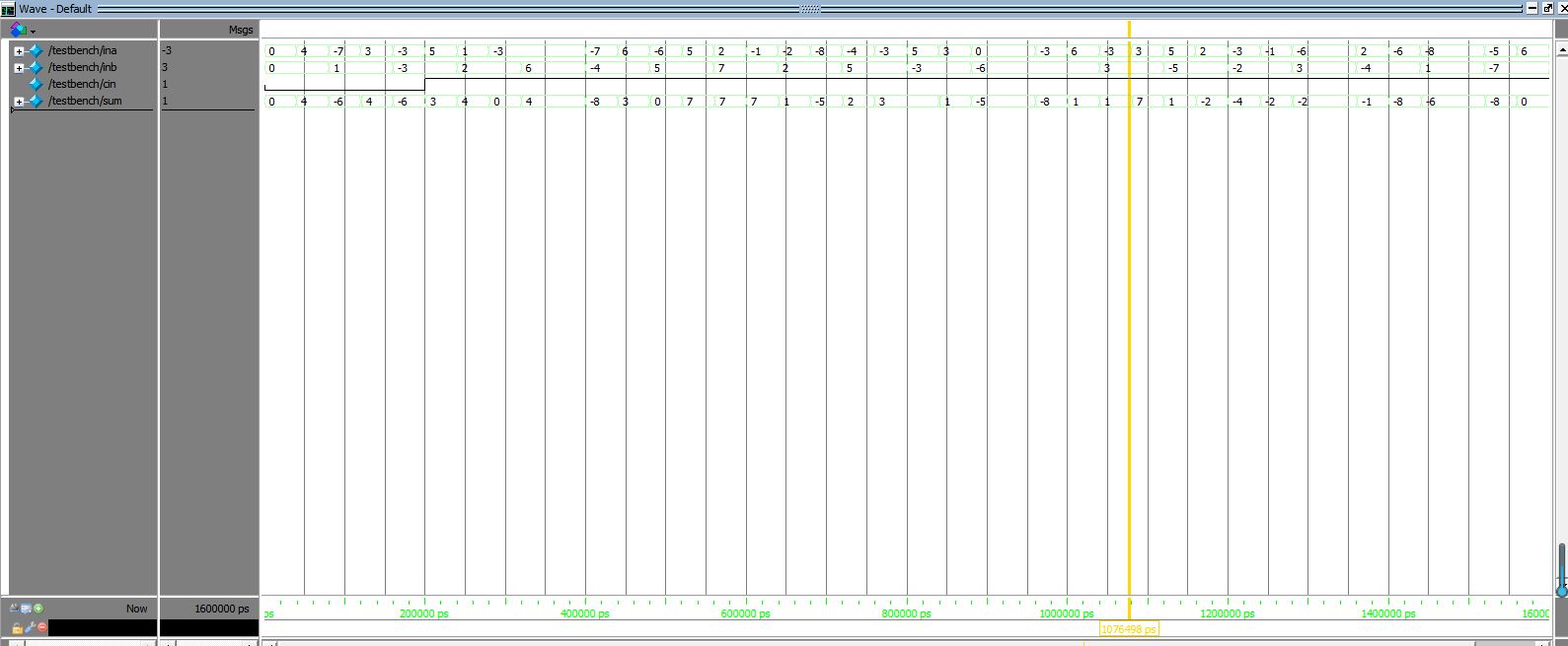
（1）SmartDesign的连线图



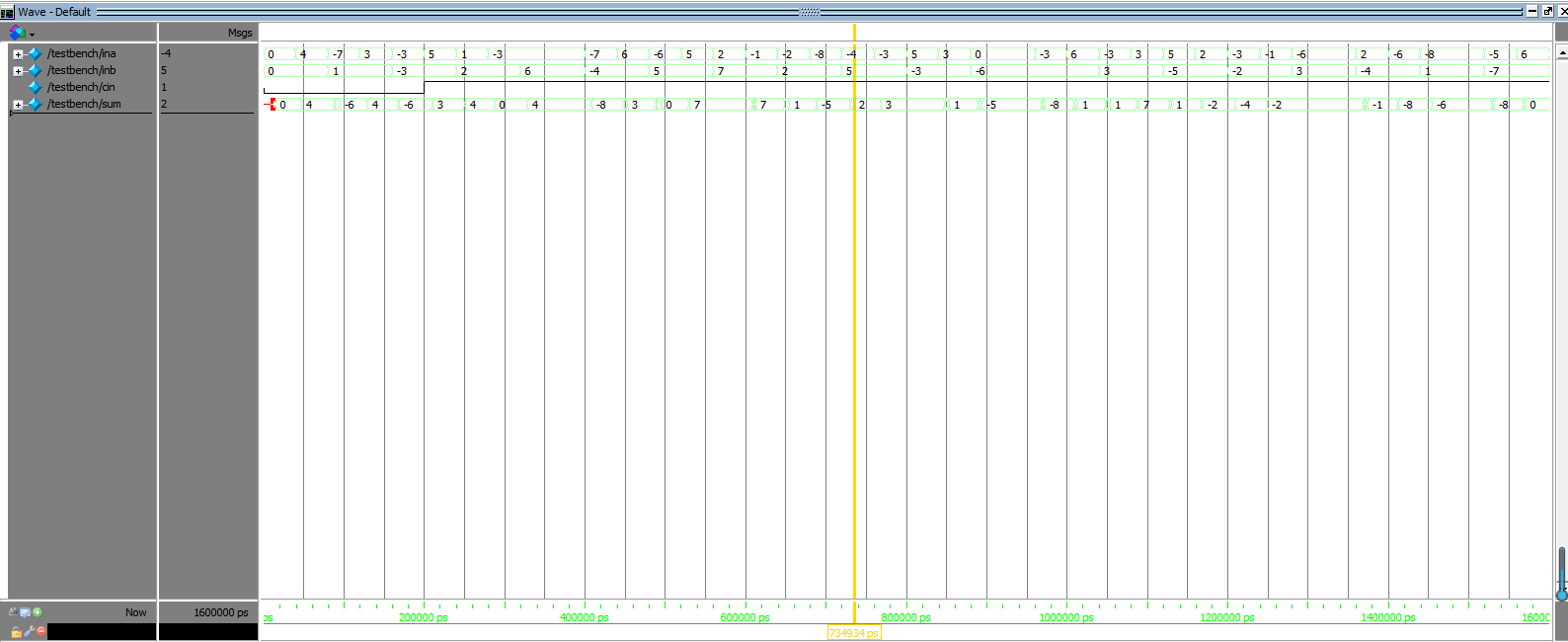
（2）功能仿真波形图



第一次

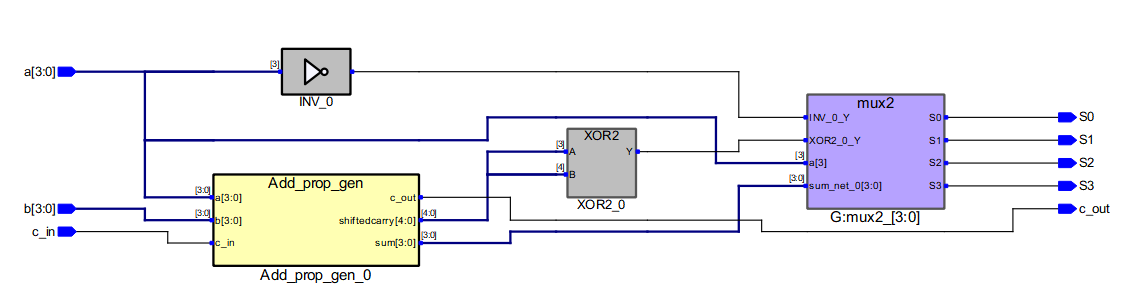


第二次

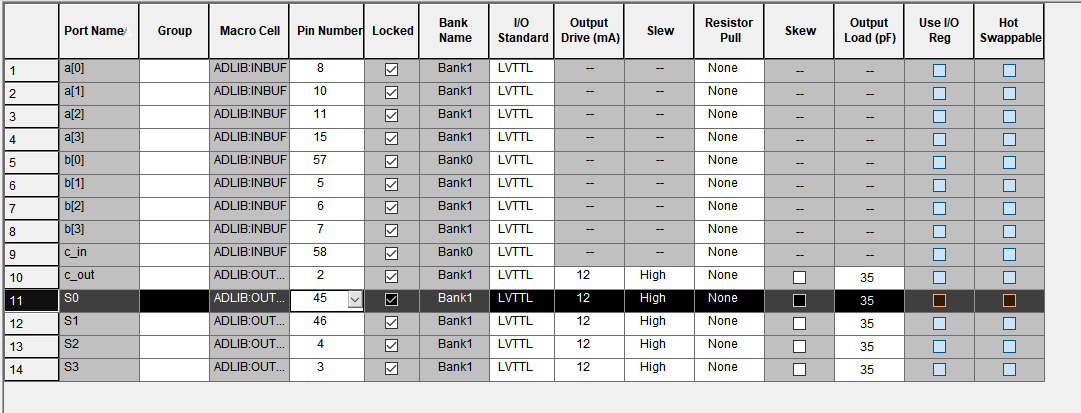


第三次

（3）综合结果RTL图



（4）引脚分配I/O Attribute Editor截图



综合实验10——交通灯控制器

设计要求：实现一个常见的十字路口交通灯控制功能。一个十字路口的交通灯一般分为两个方向，每个方向具有红灯、绿灯和黄灯三种。实现一个常见的十字路口交通灯控制功能，具体要求如下：

① 十字路口包含A、B两个方向的车道。A方向放行一分钟（绿55秒，黄5秒），同时B方向禁行（红60秒）；然后A方向禁行1分钟（红60秒），同时B方向放行（绿55秒，黄5秒），示意图如图5-56所示。依此类推，循环往复。

② 实现正常的倒计时功能，用两组数码管作为A和B两个方向的倒计时显示。

③ 系统时钟1KHz。

代码：

module one\_second\_clk(Reset,Clk,Cout);

input Reset,Clk;

reg[9:0]Count;

output reg Cout;

always@(posedge Clk)

if(Reset)

begin Count=0;Cout=0;end

else if(Count==999)

begin Count=0;Cout=1;end

else begin Count=Count+1;Cout=0;end

endmodule

module traffic\_control(Clk,Reset,Done\_NS,Done\_EW,Red1,Yellow1,Green1,Red2,Yellow2,Green2,Sload\_NS,Sload\_EW,State\_cnt);

input Clk,Reset;

input Done\_NS,Done\_EW;

output Red1,Yellow1,Green1,Red2,Yellow2,Green2;

output Sload\_NS,Sload\_EW;

output[3:0]State\_cnt;

parameter S0=4'b0001,S1=4'b0010,S2=4'b0100,S3=4'b1000;

reg[3:0]current\_state,next\_state;

reg Red1,Yellow1,Green1,Red2,Yellow2,Green2;

reg Sload\_NS,Sload\_EW;

assign State\_cnt=current\_state;

always@(posedge Clk,posedge Reset)

begin

if(Reset)current\_state<=S0;

else current\_state<=next\_state;

end

always@(current\_state,Done\_NS,Done\_EW)

begin:fsmtr

case(current\_state)

S0:begin

if(Done\_NS)next\_state<=S1;

else next\_state<=S0;

end

S1:begin

if(Done\_NS)next\_state<=S2;

else next\_state<=S1;

end

S2:begin

if(Done\_EW)next\_state<=S3;

else next\_state<=S2;

end

S3:begin

if(Done\_EW)next\_state<=S0;

else next\_state<=S3;

end

default:next\_state<=S0;

endcase

end

always@(\*)

begin

Sload\_NS<=1'b0;

Sload\_EW<=1'b0;

case(current\_state)

S0:begin

Green1<=1'b1;Yellow1<=1'b0;Red1<=1'b0;

Green2<=1'b0;Yellow2<=1'b0;Red2<=1'b1;

if(Done\_NS)

begin

Sload\_NS<=1'b1;

end

end

S1:begin

Green1<=1'b0;Yellow1<=1'b1;Red1<=1'b0;

Green2<=1'b0;Yellow2<=1'b0;Red2<=1'b1;

if(Done\_NS)

begin

Sload\_NS<=1'b1;

Sload\_EW<=1'b1;

end

end

S2:begin

Green1<=1'b0;Yellow1<=1'b0;Red1<=1'b1;

Green2<=1'b1;Yellow2<=1'b0;Red2<=1'b0;

if(Done\_EW)

begin

Sload\_EW<=1'b1;

end

end

S3:begin

Green1<=1'b0;Yellow1<=1'b0;Red1<=1'b1;

Green2<=1'b0;Yellow2<=1'b1;Red2<=1'b0;

if(Done\_EW)

begin

Sload\_NS<=1'b1;

Sload\_EW<=1'b1;

end

end

default:begin

Green1<=1'b1;Yellow1<=1'b0;Red1<=1'b0;

Green2<=1'b0;Yellow2<=1'b0;Red2<=1'b1;

Sload\_NS<=1'b1;

Sload\_EW<=1'b1;

end

endcase

end

endmodule

module timer\_EW(Clk,Reset,Ld,En,State,Q\_EW,Done\_EW);

input Clk,Reset,Ld,En;

input[3:0]State;

output[7:0]Q\_EW;

output Done\_EW;

reg[7:0] Q\_EW;

parameter red\_EW=8'h3b,green\_EW=8'h36,yellow\_EW=8'h04;

parameter St0=4'b0001,St1=4'b0010,St2=4'b0100,St3=4'b1000;

assign Done\_EW=~(|Q\_EW)&&En;

always@(posedge Clk)

if(Reset)Q\_EW<=green\_EW;

else if(Ld)

case(State)

St0:Q\_EW<=8'h00;

St1:Q\_EW<=green\_EW;

St2:Q\_EW<=yellow\_EW;

St3:Q\_EW<=red\_EW;

default:Q\_EW<=8'h00;

endcase

else if(En)Q\_EW<=Q\_EW-1;

endmodule

module timer\_NS(Clk,Reset,Ld,En,State,Q\_NS,Done\_NS);

input Clk,Reset,Ld,En;

input[3:0]State;

output[7:0]Q\_NS;

output Done\_NS;

reg[7:0] Q\_NS;

parameter red\_NS=59,green\_NS=54,yellow\_NS=4;

parameter St0=4'b0001,St1=4'b0010,St2=4'b0100,St3=4'b1000;

assign Done\_NS=(Q\_NS==0)&&En;

always@(posedge Clk)

if(Reset)Q\_NS<=green\_NS;

else if(Ld)

case(State)

St0:Q\_NS<=yellow\_NS;

St1:Q\_NS<=red\_NS;

St2:Q\_NS<=8'h00;

St3:Q\_NS<=green\_NS;

default:Q\_NS<=8'h00;

endcase

else if(En)Q\_NS<=Q\_NS-1;

endmodule

module BIN\_to\_BCD(Data,Units,Tens,Hundreds);

input[7:0]Data;

output[3:0]Units;

output[3:0]Tens;

output[3:0]Hundreds;

reg[3:0]Units,Tens,Hundreds;

reg[7:0]dat\_r;

always@(Data)

begin

dat\_r=Data;

Hundreds=dat\_r/100;

Tens=(dat\_r-Hundreds\*100)/10;

Units=dat\_r%10;

end

endmodule

module dymamic\_led(seg,sl,Clk,Reset,DataA,DataB,DataC,DataD);

output[7:0]seg;

output[3:0]sl;

input Clk,Reset;

input[3:0]DataA,DataB,DataC,DataD;

reg[7:0]seg\_reg;

reg[3:0]sl\_reg,disp\_dat;

reg[1:0]count;

assign seg=seg\_reg;

assign sl=sl\_reg;

always@(posedge Clk,posedge Reset)

begin

if(Reset)count=0;

else count=count+1;

end

always@(\*)

begin

case(count)

2'b00:begin

disp\_dat=DataA;

sl\_reg=4'b1110;

end

2'b01:begin

disp\_dat=DataB;

sl\_reg=4'b1101;

end

2'b10:begin

disp\_dat=DataC;

sl\_reg=4'b1011;

end

2'b11:begin

disp\_dat=DataD;

sl\_reg=4'b0111;

end

endcase

end

always@(disp\_dat)

begin

case(disp\_dat)

4'h0:seg\_reg=8'h3f;

4'h1:seg\_reg=8'h06;

4'h2:seg\_reg=8'h5b;

4'h3:seg\_reg=8'h4f;

4'h4:seg\_reg=8'h66;

4'h5:seg\_reg=8'h6d;

4'h6:seg\_reg=8'h7d;

4'h7:seg\_reg=8'h07;

4'h8:seg\_reg=8'h7f;

4'h9:seg\_reg=8'h67;

4'ha:seg\_reg=8'h77;

4'hb:seg\_reg=8'h7c;

4'hc:seg\_reg=8'h3c;

4'hd:seg\_reg=8'h5e;

4'he:seg\_reg=8'h79;

4'hf:seg\_reg=8'h71;

endcase

end

endmodule

测试平台：

`timescale 1ns/1ns

module tb\_fsm\_cnt;

reg clk,reset;

wire[7:0]seg;

wire[3:0]sl;

wire red1,green1,yellow1,red2,green2,yellow2;

parameter DELY=20;

always#(DELY/2)clk=~clk;

core\_traffic tb(.Red1(red1),.Red2(red2),.Reset(reset),.Clk(clk),.Green1(green1),.Green2(green2),.Yellow1(yellow1),.Yellow2(yellow2),.seg(seg),.sl(sl));

initial

begin

clk=0;reset=0;

#(DELY\*2)reset=1;

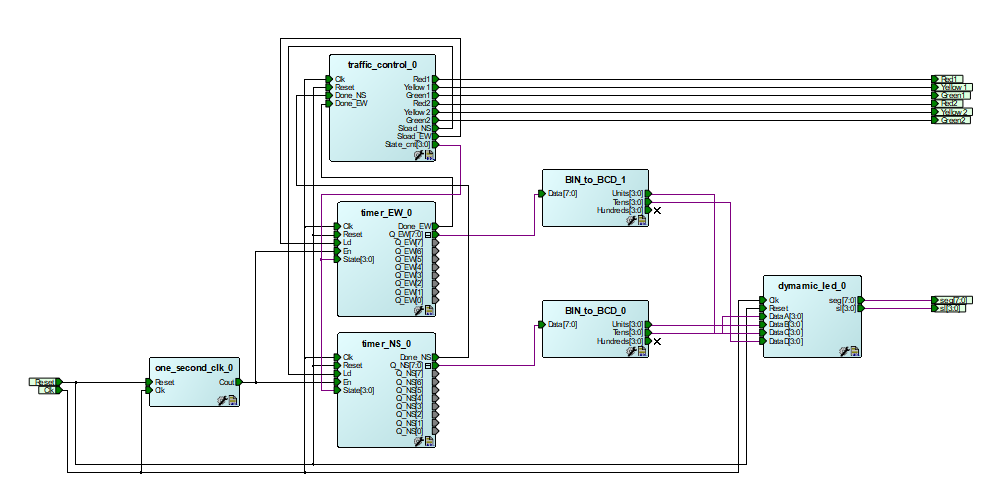
#DELY reset=0;

#(DELY\*500000) $finish;

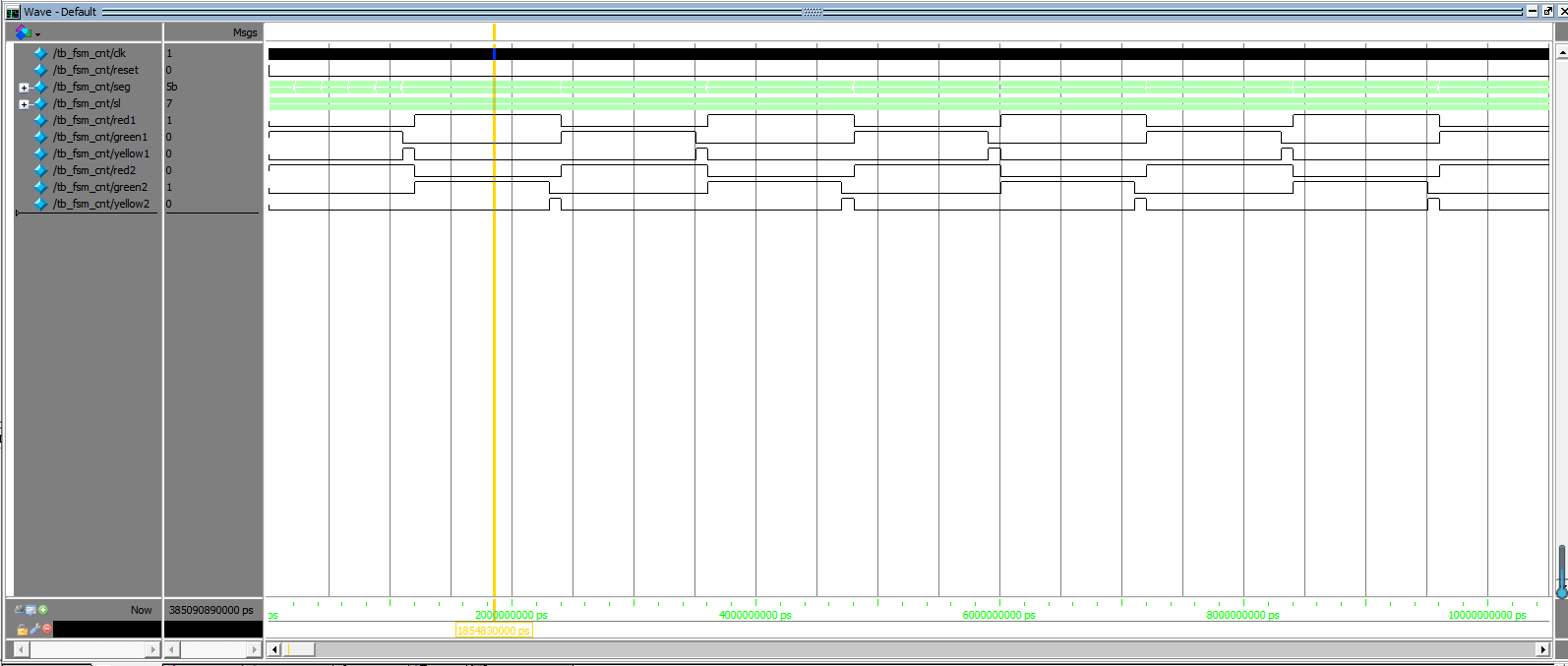
end

endmodule

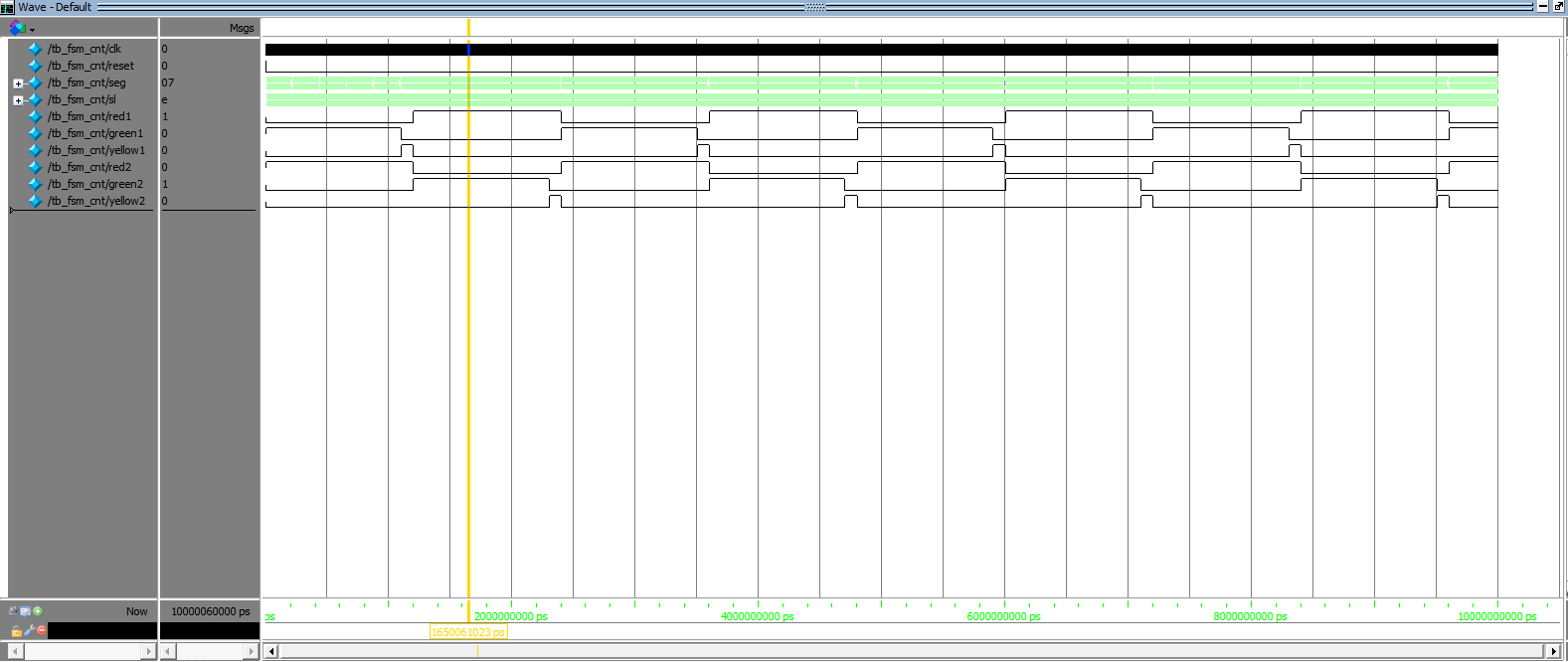
（1）SmartDesign的连线图



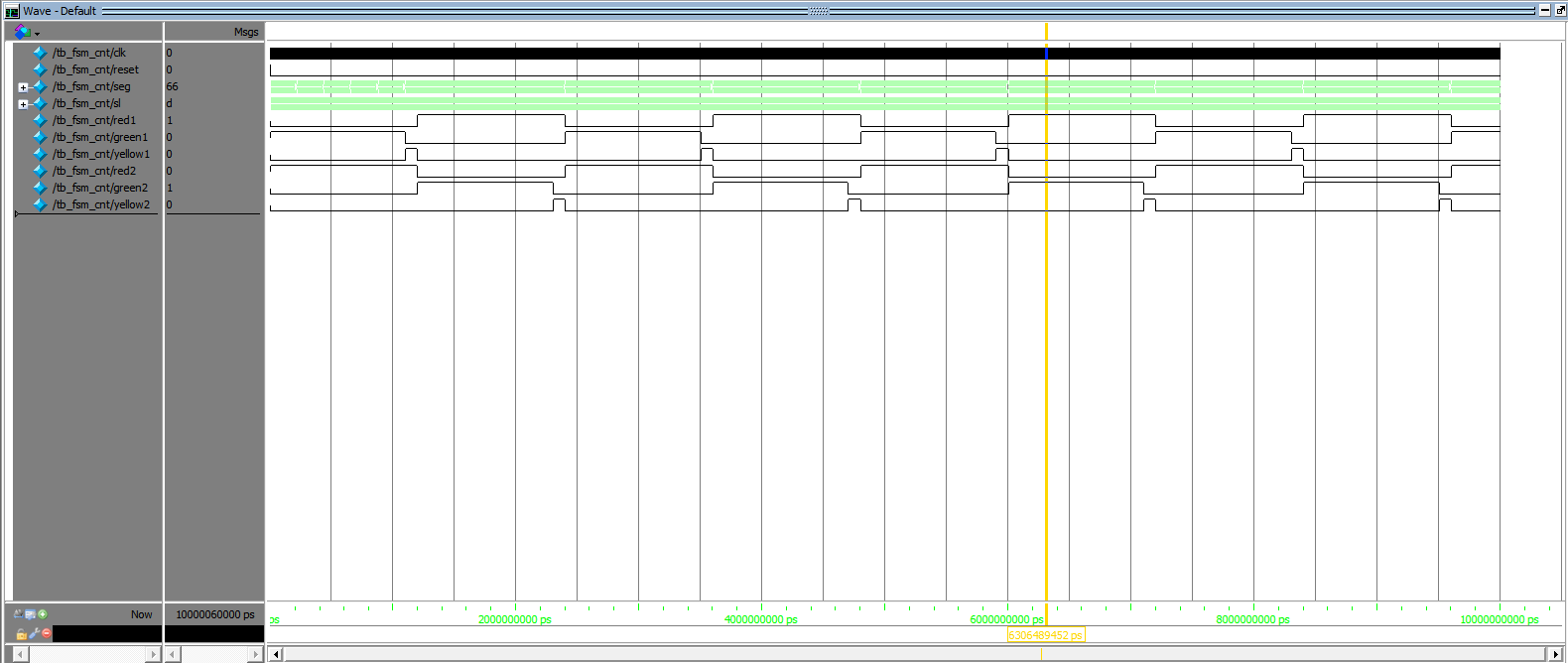
（2）功能仿真波形图



第一次

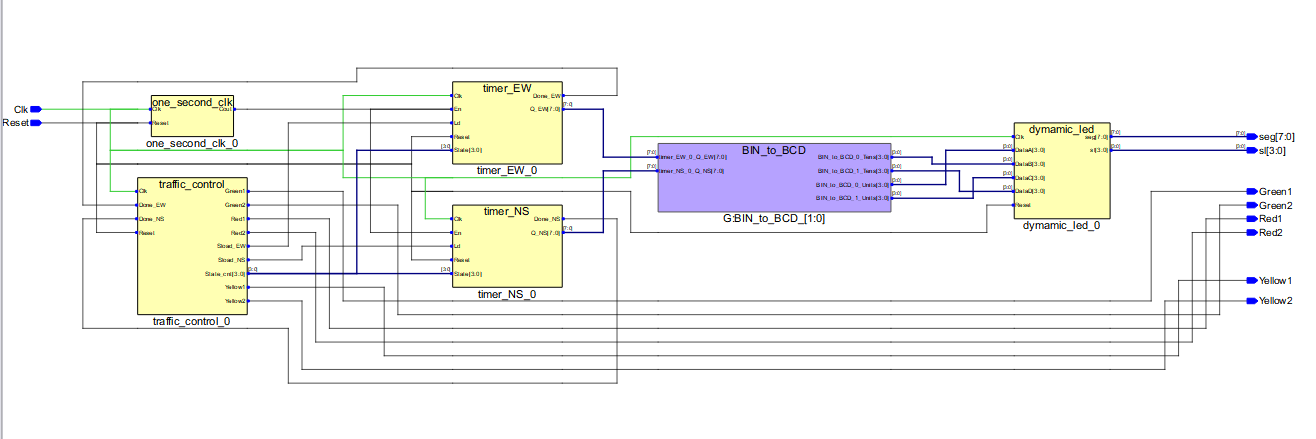


第二次



第三次

（3）综合结果RTL图



（4）引脚分配I/O Attribute Editor截图

