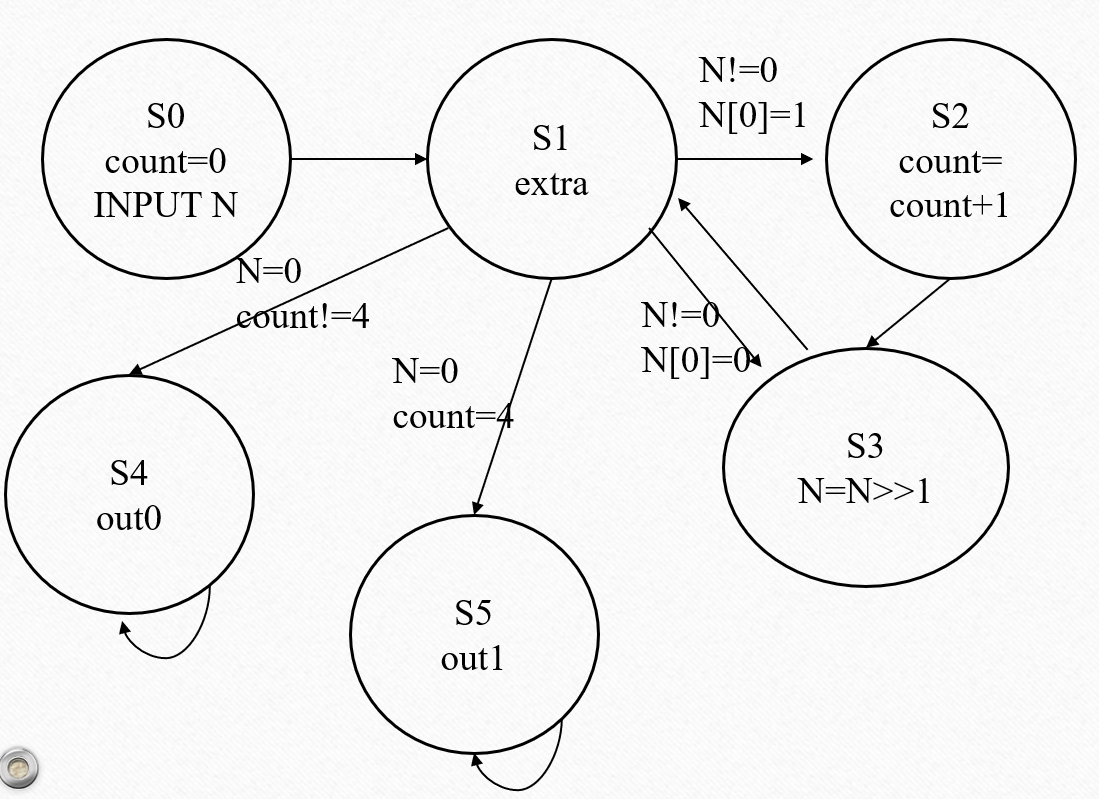
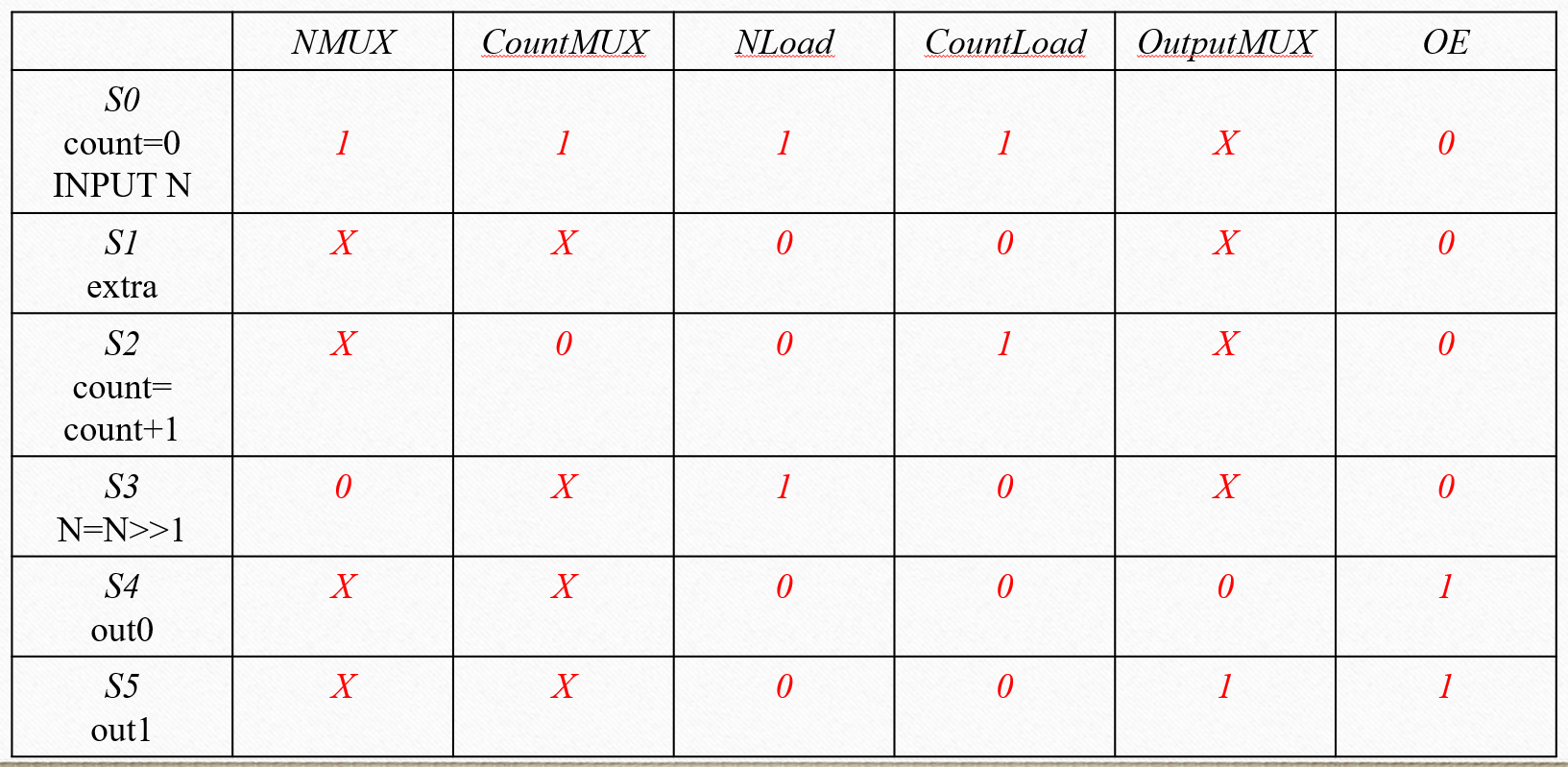
**实验报告 Lab 6**

一、状态图、控制字

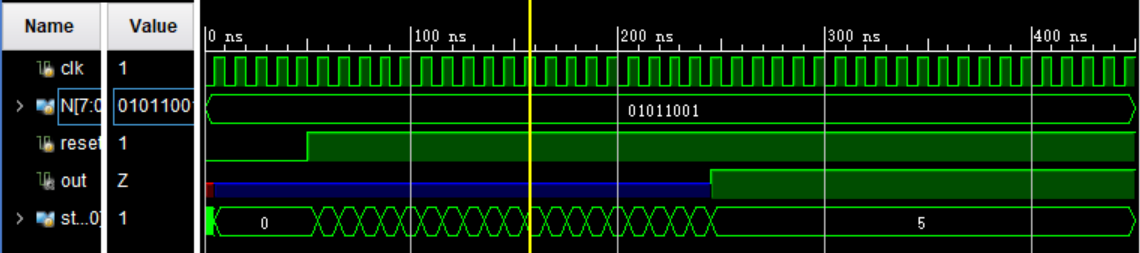




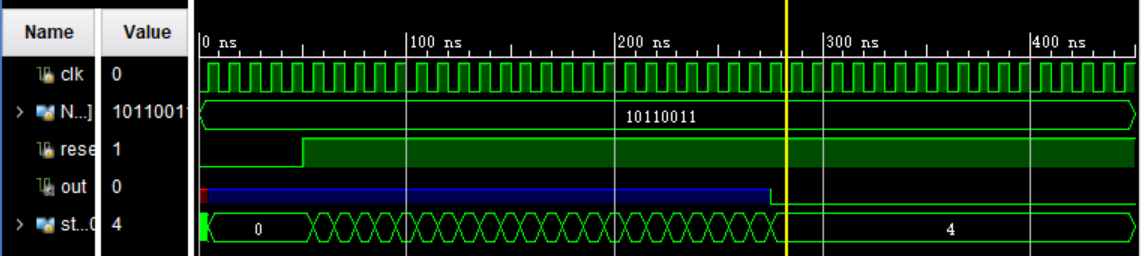
二、代码（见附页）

三、仿真

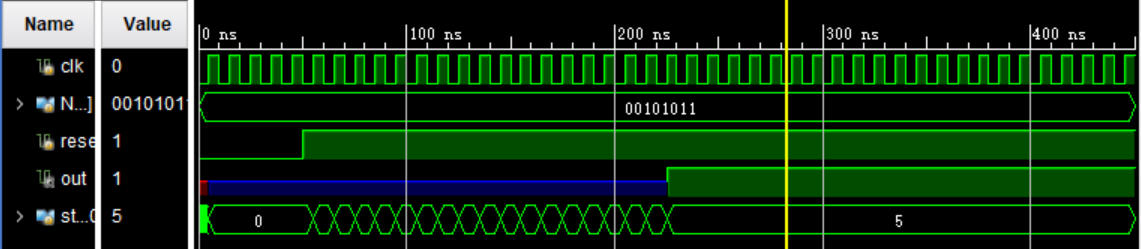
01011001：灯亮



10110011：灯灭



00101011：灯亮



四、板子现象

|  |  |
| --- | --- |
|  |  |
| 01101001，亮 | 01000001，灭 |
|  |  |
| 10000101，灭 | 00110101，亮 |

**附页** 代码

module microprocessor(

input clk,

input [7:0] N,

input reset,

output out

);

wire NMUX;

wire CountMUX;

wire NLoad;

wire CountLoad;

wire OutputMUX;

wire OE;

wire n\_0;

wire c\_4;

wire n0\_1;

Control control\_inst(

.clk(clk),

.reset(~reset),

.n\_0(n\_0),

.c\_4(c\_4),

.n0\_1(n0\_1),

.NMUX(NMUX),

.CountMUX(CountMUX),

.NLoad(NLoad),

.CountLoad(CountLoad),

.OutputMUX(OutputMUX),

.OE(OE)

);

Datapath datapath\_inst(

.clk(clk),

.N(N),

.reset(~reset),

.NMUX(NMUX),

.CountMUX(CountMUX),

.NLoad(NLoad),

.CountLoad(CountLoad),

.OutputMUX(OutputMUX),

.OE(OE),

.n\_0(n\_0),

.c\_4(c\_4),

.n0\_1(n0\_1),

.out(out)

);

Endmodule

module Control(

input clk,

input reset,

input n\_0,

input c\_4,

input n0\_1,

output reg NMUX,

output reg CountMUX,

output reg NLoad,

output reg CountLoad,

output reg OutputMUX,

output reg OE

);

parameter S0 = 0;

parameter S1 = 1;

parameter S2 = 2;

parameter S3 = 3;

parameter S4 = 4;

parameter S5 = 5;

reg [2:0] state, next\_state;

// 状态转移逻辑

always@(\*) begin

if (reset) begin // 统一为高电平复位

next\_state = S0;

end else begin

case (state)

S0: begin

next\_state = S1;

end

S1: begin

if ((n\_0 == 0) && (n0\_1 == 1))

next\_state = S2;

else if ((n\_0 == 0) && (n0\_1 == 0))

next\_state = S3;

else if ((n\_0 == 1) && (c\_4 == 0))

next\_state = S4;

else if ((n\_0 == 1) && (c\_4 == 1))

next\_state = S5;

end

S2: begin

next\_state = S3;

end

S3: begin

next\_state = S1;

end

S4: begin

next\_state = S4;

end

S5: begin

next\_state = S5;

end

default: begin

next\_state = S0;

end

endcase

end

end

// 状态寄存器更新逻辑

always@(posedge clk) begin

if (reset) begin

state <= S0;

end else begin

state <= next\_state;

end

end

// 输出逻辑

always@(\*)begin

case (state)

S0: begin

NMUX = 1;

CountMUX = 1;

NLoad = 1;

CountLoad = 1;

OutputMUX = 0;

OE = 0;

end

S1: begin

NMUX = 0; // 补充未赋值信号

CountMUX = 0;

NLoad = 0;

CountLoad = 0;

OutputMUX = 0;

OE = 0;

end

S2: begin

NMUX = 0; // 补充未赋值信号

CountMUX = 0;

NLoad = 0;

CountLoad = 1;

OutputMUX = 0;

OE = 0;

end

S3: begin

NMUX = 0; // 补充未赋值信号

CountMUX = 0;

NLoad = 1;

CountLoad = 0;

CountMUX = 0;

OE = 0;

end

S4: begin

NMUX = 0; // 补充未赋值信号

CountMUX = 0;

NLoad = 0;

CountLoad = 0;

OutputMUX = 0;

OE = 1;

end

S5: begin

NMUX = 0; // 补充未赋值信号

CountMUX = 0;

NLoad = 0;

CountLoad = 0;

OutputMUX = 1;

OE = 1;

end

//default: begin

// NLoad = 0;

// CountLoad = 0;

// OutputMUX = 0;

// OE = 0;

// end

endcase

end

endmodule

module Datapath(

input clk,

input [7:0]N,

input reset,

input NMUX,

input CountMUX,

input NLoad,

input CountLoad,

input OutputMUX,

input OE,

output n\_0,

output c\_4,

output n0\_1,

output out

);

wire [7:0] Nin;

wire [3:0] Cin;

wire [7:0] Nout;

wire [3:0] Cout;

wire [7:0] Nout1;

wire [3:0] Cout1;

wire OEin;

mux2 mux\_n(.A(Nout1),.B(N),.S(NMUX),.Y(Nin));

mux2 mux\_count(.A(Cout1),.B(0),.S(CountMUX),.Y(Cin));

RegN regn1(.clk(clk),.reset(reset),.load(NLoad),.in(Nin),.out(Nout));

RegC regc1(.clk(clk),.reset(reset),.load(CountLoad),.in(Cin),.out(Cout));

assign n\_0=~(Nout[7]|Nout[6]|Nout[5]|Nout[4]|Nout[3]|Nout[2]|Nout[1]|Nout[0]);

assign n0\_1=Nout[0];

assign c\_4=(~Cout[3])&Cout[2]&(~Cout[1])&(~Cout[0]);

assign Nout1=Nout>>1;

assign Cout1=Cout+1;

mux2 mux\_o(.A(0),.B(1),.S(OutputMUX),.Y(OEin));

OE OE1(.in(OEin),.en(OE),.out(out));

endmodule

module RegN(

input [7:0]in,

input clk,

input reset,

input load,

output reg [7:0]out

);

always @(posedge clk or posedge reset) begin

if (reset)

out <= 8'b0;

else if (load)

out <= in;

end

endmodule

module RegC(

input [3:0]in,

input clk,

input reset,

input load,

output reg [3:0]out

);

always @(posedge clk or posedge reset) begin

if (reset)

out <= 4'b0;

else if (load)

out <= in;

end

endmodule