**电子设计自动化（EDA）实验报告**

实验十一

1. 实验内容

实现点阵的汉字显示：张

1. 代码

module seven(clk,reset,row,col);

input clk;

input reset;

output reg [15:0] row; //行

output reg [3:0] col;//列

initial col=4'b0;

parameter [2:0]s1=000;

parameter [2:0]s2=001;

parameter [2:0]s3=010;

parameter [2:0]s4=011;

reg[6:0]data;

reg [2:0]current\_state;

reg [2:0]next\_state;

reg divi\_50hz;

reg [24:0] count\_reg1; //定义一个25位的计数值

always@(posedge clk)

if(count\_reg1==25'd24999) //（50Mhz / 50hz）/ 2 - 1

begin

divi\_50hz <= ~ divi\_50hz; //高低电平翻转

count\_reg1 <= 25'd0; //计数值清零

end

else

begin

count\_reg1 <= count\_reg1 + 25'd1;

end

always @(posedge clk)

begin

if (!reset) col<=4'b0;

else

begin //利用计数器产生列的 16 种编码：0000-1111

if(col<15) col<=col+1;

else col<=4'b0000;

end

end

//reg [2:0]count;

always@(posedge divi\_50hz)

begin

current\_state<=next\_state;

end

always@(posedge divi\_50hz)

begin

case(current\_state)

s1:next\_state<=s2;

s2:next\_state<=s3;

s3:next\_state<=s4;

s4:next\_state<=s1;

default:next\_state<=s1;

endcase

end

always@(posedge divi\_50hz or negedge reset)

begin

case(current\_state)

s1:

begin

if (!reset) row<=16'b0;

else

begin

case (col)

4'b0000: row<=16'b0001000001000000; //第 1 列

4'b0001: row<=16'b0001000010000000; //第 2 列

4'b0010: row<=16'b0001000100000000; //3

4'b0011: row<=16'b0001001111111111; //4

4'b0100: row<=16'b0001110000000010; //5

4'b0101: row<=16'b0011000010000010; //6

4'b0110: row<=16'b1101000010000010; //7

4'b0111: row<=16'b0001000010000010; //8

4'b1000: row<=16'b0001000010000010; //9

4'b1001: row<=16'b0001011111111110; //10

4'b1010: row<=16'b0001000010000010; //11

4'b1011: row<=16'b0001000010000010; //12

4'b1100: row<=16'b0001000010000010; //13

4'b1101: row<=16'b0001000010000010; //14

4'b1110: row<=16'b0001000000000010; //15

4'b1111: row<=16'b0000000000000000; //16

default:row<=16'b0000000000000000;

endcase

end

end

s2:

begin

if (!reset) row<=16'b0;

else

begin

case (col)

4'b0000: row<=16'b0000001000000000; //第 1 列

4'b0001: row<=16'b0000001000000000; //第 2 列

4'b0010: row<=16'b1000001000000000; //3

4'b0011: row<=16'b1000001000000000; //4

4'b0100: row<=16'b1000001000000000; //5

4'b0101: row<=16'b1000001000000000; //6

4'b0110: row<=16'b1000001000000000; //7

4'b0111: row<=16'b1111111111111110; //8

4'b1000: row<=16'b1000001000000000; //9

4'b1001: row<=16'b1000001000000000; //10

4'b1010: row<=16'b1000001000000000; //11

4'b1011: row<=16'b1000001000000000; //12

4'b1100: row<=16'b1000001000000000; //13

4'b1101: row<=16'b0000001000000000; //14

4'b1110: row<=16'b0000001000000000; //15

4'b1111: row<=16'b0000000000000000; //16

default:row<=16'b0000000000000000;

endcase

end

end

s3:

begin

if (!reset) row<=16'b0;

else

begin

case (col)

4'b0000: row<=16'b0011111111110000; //第 1 列

4'b0001: row<=16'b0010000000100000; //第 2 列

4'b0010: row<=16'b0010000000100000; //3

4'b0011: row<=16'b0011111111110010; //4

4'b0100: row<=16'b0000000000001100; //5

4'b0101: row<=16'b0011111111110000; //6

4'b0110: row<=16'b0010001000010000; //7

4'b0111: row<=16'b0010001001100000; //8

4'b1000: row<=16'b0011111111111111; //9

4'b1001: row<=16'b1010001010010000; //10

4'b1010: row<=16'b0110001001100000; //11

4'b1011: row<=16'b0011111111111111; //12

4'b1100: row<=16'b0010001011000000; //13

4'b1101: row<=16'b0010001000110000; //14

4'b1110: row<=16'b0010000000010000; //15

4'b1111: row<=16'b0000000000000000; //16

default:row<=16'b0000000000000000;

endcase

end

end

s4:

begin

if (!reset) row<=16'b0;

else

begin

case (col)

4'b0000: row<=16'b0000000000000000; //第 1 列

4'b0001: row<=16'b0111111111100000; //第 2 列

4'b0010: row<=16'b0100000001000000; //3

4'b0011: row<=16'b0100000001000000; //4

4'b0100: row<=16'b0111111111100000; //5

4'b0101: row<=16'b0000000000001000; //6

4'b0110: row<=16'b1000011000010000; //7

4'b0111: row<=16'b1011101000100000; //8

4'b1000: row<=16'b1000001001000000; //9

4'b1001: row<=16'b1000001010000100; //10

4'b1010: row<=16'b1000001100000010; //11

4'b1011: row<=16'b1111111111111100; //12

4'b1100: row<=16'b1000001000000000; //13

4'b1101: row<=16'b1000001000000000; //14

4'b1110: row<=16'b0000001000000000; //15

4'b1111: row<=16'b0000000000000000; //16

default:row<=16'b0000000000000000;

endcase

end

end

default:

begin

if (!reset) row<=16'b0;

else

begin

case (col)

4'b0000: row<=16'b0010000000010000; //第 1 列

4'b0001: row<=16'b0010000000010000; //第 2 列

4'b0010: row<=16'b0010000000010000; //3

4'b0011: row<=16'b0010001111010000; //4

4'b0100: row<=16'b0010001001010000; //5

4'b0101: row<=16'b1111101001010000; //6

4'b0110: row<=16'b0010101001010000; //7

4'b0111: row<=16'b0010111111111111; //8

4'b1000: row<=16'b0010101001010000; //9

4'b1001: row<=16'b1111101001010000; //10

4'b1010: row<=16'b0010001001010000; //11

4'b1011: row<=16'b0010001111010000; //12

4'b1100: row<=16'b0010000000010000; //13

4'b1101: row<=16'b0010000000010000; //14

4'b1110: row<=16'b0101110001011000; //15

4'b1111: row<=16'b0010000000010000; //16

default:row<=16'b0000000000000000;

endcase

end

end

endcase

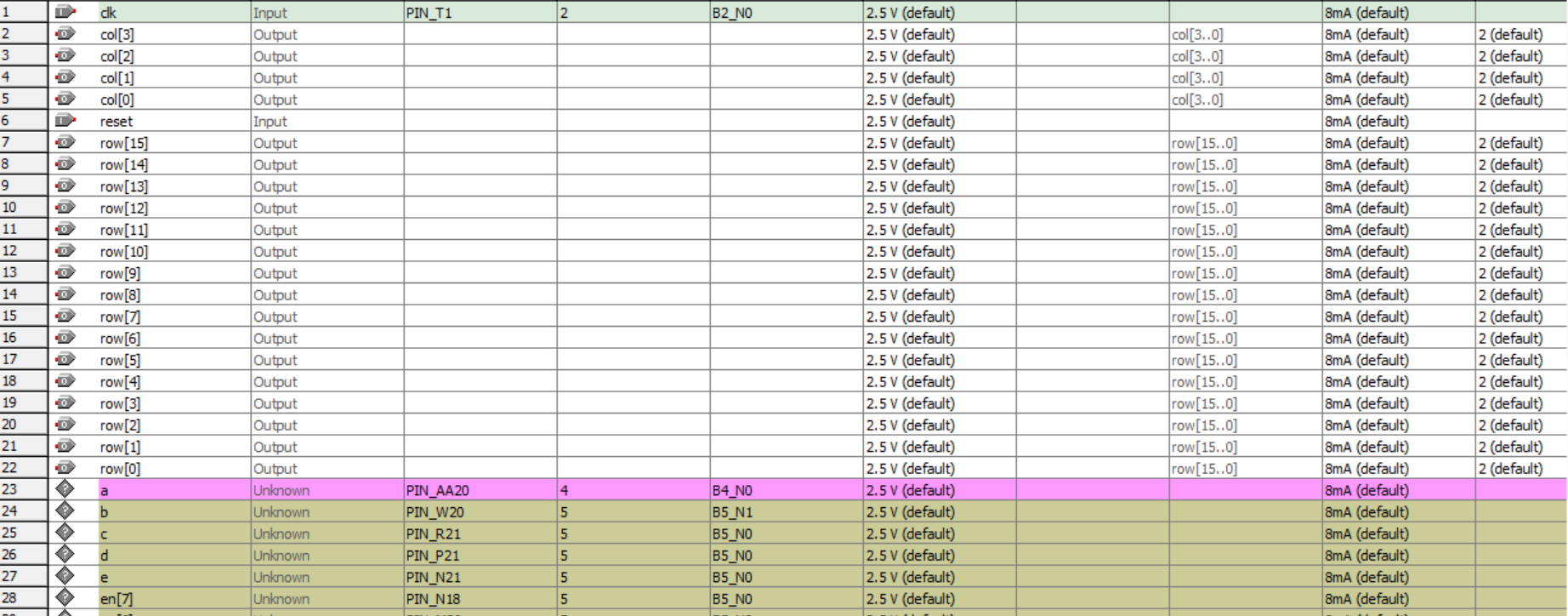
end

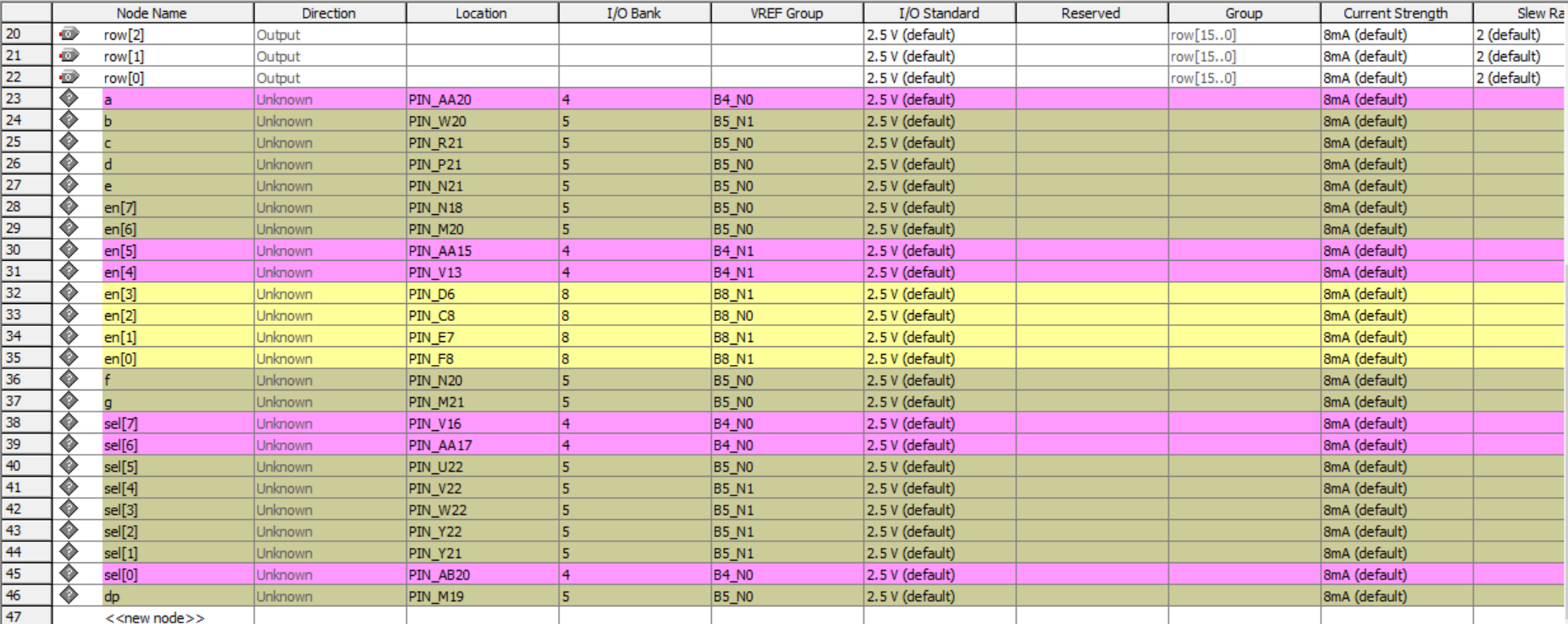
endmodule

1. RTL



1. 引脚绑定





1. 结果

在点阵上显示了“张”

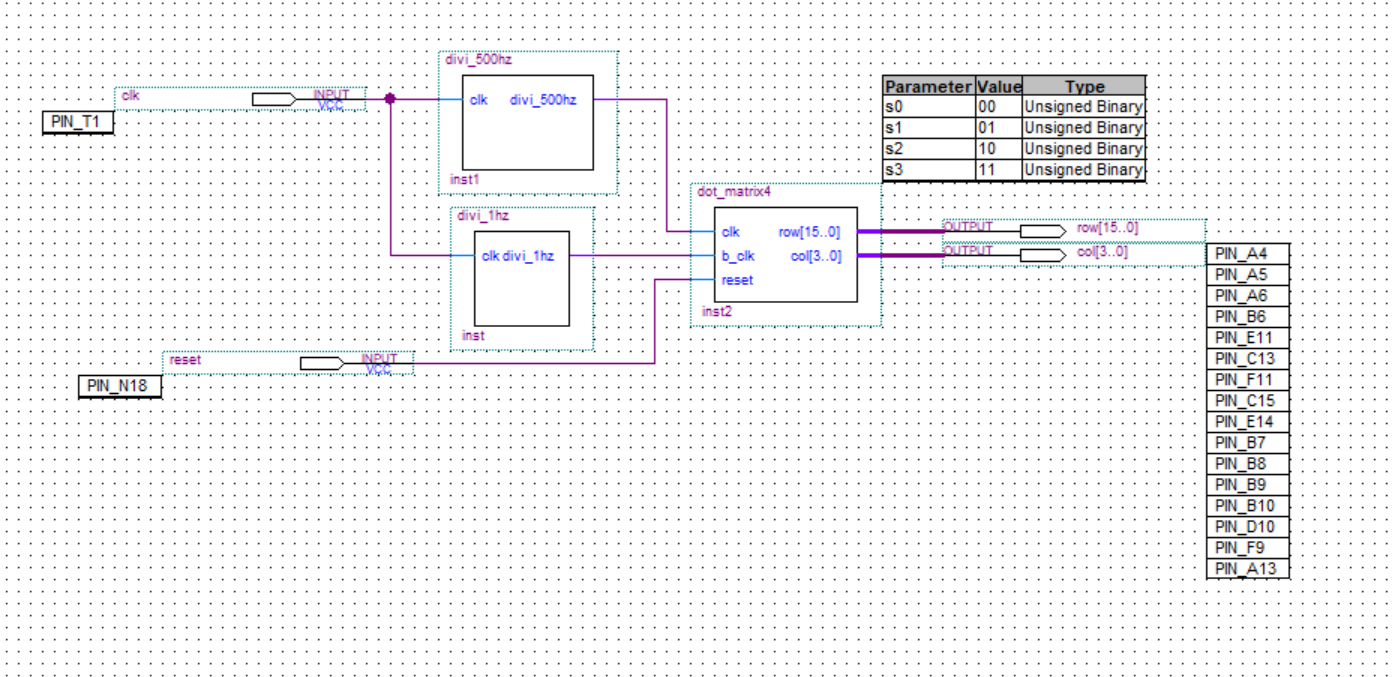
扩展：

1. 实验内容

点阵的四个字循环显示：在干嘛呀

1. 代码

顶层文件



分频

module divi\_1hz(clk,divi\_1hz);

input clk;

output reg divi\_1hz;

reg [24:0] count\_reg1; //定义一个25位的计数值

always@(posedge clk)

if(count\_reg1==25'd4999999) //（50Mhz / 5hz）/ 2 - 1

begin

divi\_1hz <= ~ divi\_1hz; //高低电平翻转

count\_reg1 <= 25'd0; //计数值清零

end

else

begin

count\_reg1 <= count\_reg1 + 25'd1;

end

endmodule

点阵

module dot\_matrix4(clk,b\_clk,reset,row,col);

input clk,b\_clk,reset;

output reg [15:0] row; //行

output reg [3:0] col;//列

initial col=4'b0;

parameter [1:0]s0=2'b00;

parameter [1:0]s1=2'b01;

parameter [1:0]s2=2'b10;

parameter [1:0]s3=2'b11;

reg [6:0]data;

reg [2:0]current\_state;

reg [2:0]next\_state;

//col count

always @(posedge clk)

begin

if (!reset) col<=4'b0;

else

begin //利用计数器产生列的 16 种编码：0000-1111

if(col<15) col<=col+1;

else col<=4'b0000;

end

end

//state count

always@(posedge b\_clk or negedge reset)

begin

if(!reset) current\_state<=s0;

else current\_state<=next\_state;

end

//state machine

always@(posedge b\_clk)

begin

case(current\_state)

s0:next\_state<=s1;

s1:next\_state<=s2;

s2:next\_state<=s3;

s3:next\_state<=s0;

default:next\_state<=s0;

endcase

end

//display

always @(posedge b\_clk or negedge reset)

begin

case(current\_state)

s0:begin if(!reset) row<=16'b0;//NI

else begin case (col)

4'b0000: row<=16'h0080; //第 1 列

4'b0001: row<=16'h0100;

4'b0010: row<=16'h0600;

4'b0011: row<=16'h1FFF;

4'b0100: row<=16'hE000;

4'b0101: row<=16'h0208;

4'b0110: row<=16'h0430;

4'b0111: row<=16'h18C0;

4'b1000: row<=16'hF002;

4'b1001: row<=16'h1001;

4'b1010: row<=16'h13FE;

4'b1011: row<=16'h1000;

4'b1100: row<=16'h1080;

4'b1101: row<=16'h1460;

4'b1110: row<=16'h1818;

4'b1111: row<=16'h0000; //16

default: row<=16'h0000;

endcase

end

end

s1:begin if(!reset) row<=16'b0;//HAO

else begin case (col)

4'b0000: row<=16'h0802; //第 1 列

4'b0001: row<=16'h0844;

4'b0010: row<=16'h0FA8;

4'b0011: row<=16'hF810;

4'b0100: row<=16'h0868;

4'b0101: row<=16'h0F86;

4'b0110: row<=16'h0000;

4'b0111: row<=16'h0100;

4'b1000: row<=16'h4102;

4'b1001: row<=16'h4101;

4'b1010: row<=16'h47FE;

4'b1011: row<=16'h4900;

4'b1100: row<=16'h5100;

4'b1101: row<=16'h6100;

4'b1110: row<=16'h0100;

4'b1111: row<=16'h0000; //16

default: row<=16'h0000;

endcase

end

end

s2:begin if(!reset) row<=16'b0;//SHENG

else begin case (col)

4'b0000: row<=16'h0102; //第 1 列

4'b0001: row<=16'h0202;

4'b0010: row<=16'h0C42;

4'b0011: row<=16'h7842;

4'b0100: row<=16'h0842;

4'b0101: row<=16'h0842;

4'b0110: row<=16'h0842;

4'b0111: row<=16'hFFFE;

4'b1000: row<=16'h0842;

4'b1001: row<=16'h0842;

4'b1010: row<=16'h0842;

4'b1011: row<=16'h0842;

4'b1100: row<=16'h0842;

4'b1101: row<=16'h0802;

4'b1110: row<=16'h0002;

4'b1111: row<=16'h0000; //16

default: row<=16'h0000;

endcase

end

end

s3:begin if(!reset) row<=16'b0;//HUO

else begin case (col)

4'b0000: row<=16'h0820; //第 1 列

4'b0001: row<=16'h0620;

4'b0010: row<=16'h047E;

4'b0011: row<=16'h3180;

4'b0100: row<=16'h0000;

4'b0101: row<=16'h0400;

4'b0110: row<=16'h247F;

4'b0111: row<=16'h2442;

4'b1000: row<=16'h2442;

4'b1001: row<=16'h7FC2;

4'b1010: row<=16'h4442;

4'b1011: row<=16'hC442;

4'b1100: row<=16'h447F;

4'b1101: row<=16'h0400;

4'b1110: row<=16'h0400;

4'b1111: row<=16'h0000; //16

default: row<=16'h0000;

endcase

end

end

default:begin if(!reset) row<=16'b0;//NI

else begin case (col)

4'b0000: row<=16'h0080; //第 1 列

4'b0001: row<=16'h0100;

4'b0010: row<=16'h0600;

4'b0011: row<=16'h1FFF;

4'b0100: row<=16'hE000;

4'b0101: row<=16'h0208;

4'b0110: row<=16'h0430;

4'b0111: row<=16'h18C0;

4'b1000: row<=16'hF002;

4'b1001: row<=16'h1001;

4'b1010: row<=16'h13FE;

4'b1011: row<=16'h1000;

4'b1100: row<=16'h1080;

4'b1101: row<=16'h1460;

4'b1110: row<=16'h1818;

4'b1111: row<=16'h0000; //16

default: row<=16'h0000;

endcase

end

end

endcase

end

endmodule

500hz分频

module divi\_500hz(clk,divi\_500hz);

input clk;

output reg divi\_500hz;

reg [24:0] count\_reg1; //定义一个25位的计数值

always@(posedge clk)

if(count\_reg1==25'd49999) //（50Mhz / 500hz）/ 2 - 1

begin

divi\_500hz <= ~ divi\_500hz; //高低电平翻转

count\_reg1 <= 25'd0; //计数值清零

end

else

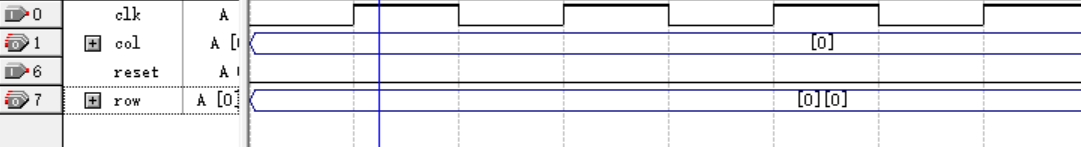
begin

count\_reg1 <= count\_reg1 + 25'd1;

end

endmodule

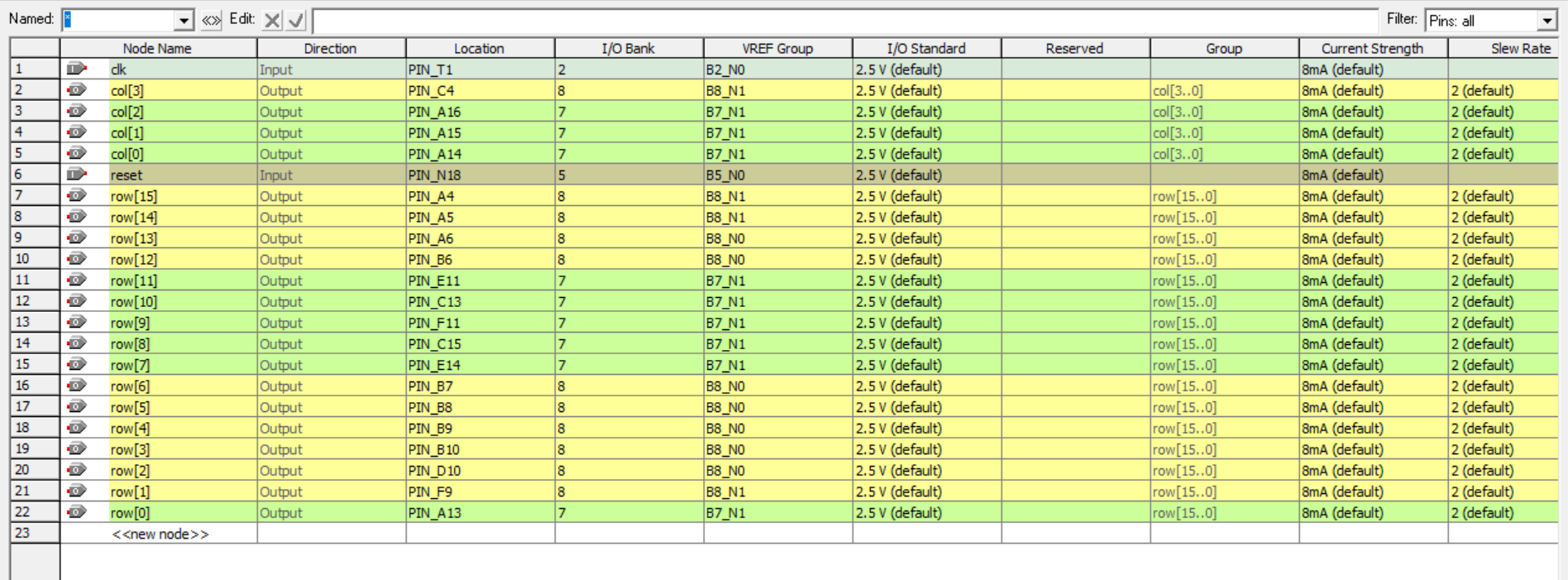
1. 仿真



1. RTL



1. 引脚绑定



1. 结果

在点阵上循环显示四个字