电子科技大学

University of Electronic Science and Technology of China

本科生实验报告

实验课程：复杂数字集成电路设计（挑战性课程）

实验名称：实验 2：二进制码和格雷码之间的转换

实验地点：无

学生姓名：周岳恒

学　　号：2021340105016

指导教师：廖永波

实验时间：2023 年 10 月 1 日

一、 实验目的

学会设计二进制码和格雷码的 verilog 描述的转换电路，理解二进制码和格雷码相互转换的原理。学会通过参数例化的方式作通用化设计，在模块外部就能控制信号宽度等内容。

二、 实验任务

编写 verilog 代码实现二进制码到格雷码，格雷码到二进制码的电路设计，并编写对应 testbench 进行仿真；用 wavedrom 绘制电路功能的波形图，绘制综合后的电路图。

三、 实验原理

二进制码到格雷码：

4 位：由于位数较少，可以用译码器的方式直接通过组合逻辑输出结果，比较简单。

N 位：布尔代数原理：$g_n=b_n$, $g_i=b_i \oplus b_{i+1}$ 从而 gray=bin⊕(bin>>1)，左边补 0，因此 $g_n=b_n \oplus 0=b_n$ 满足条件

格雷码到二进制码：

4 位：由于位数较少，可以用译码器的方式直接通过组合逻辑输出结果，比较简单。

N 位：布尔代数原理：$b_{n=} g_n$, $b_i = g_i \oplus b_{i+1}$ 从而使用 for 简化重复 N 次例化的过程，完成反馈连线。

四、 实验过程

二进制码到格雷码：

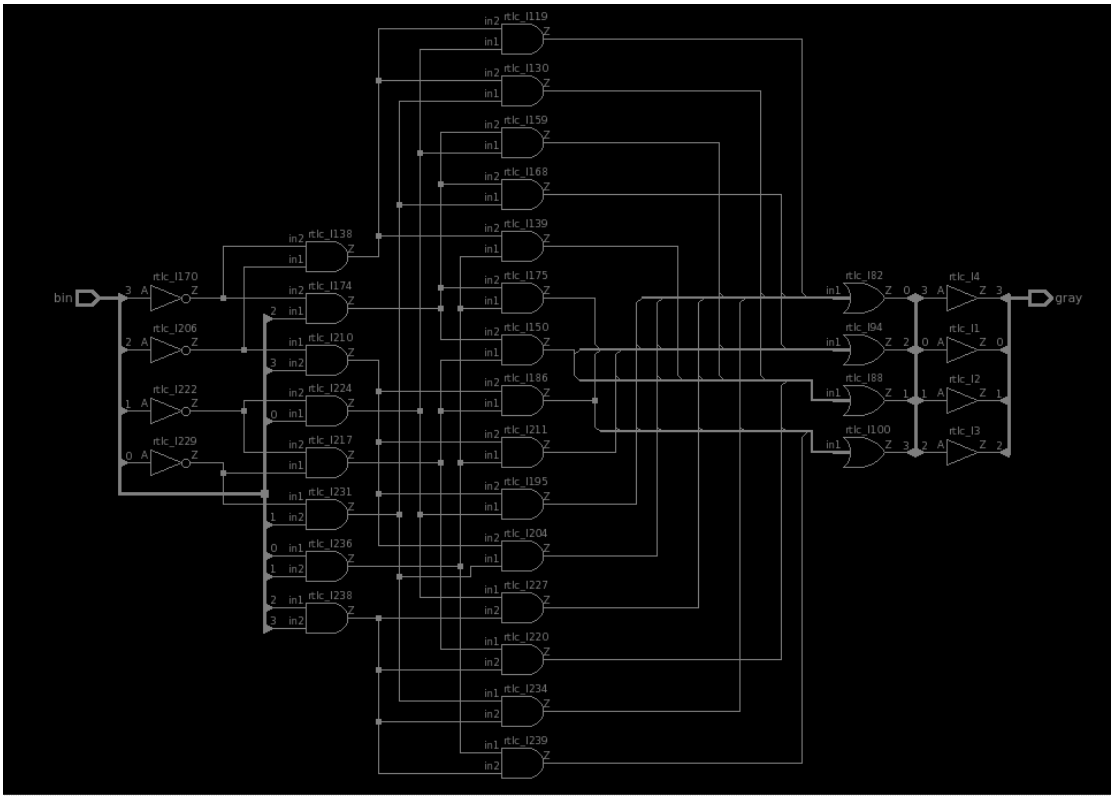
Verilog 代码：

```
module bin2gray_4(
    input [3:0] bin,
    output [3:0] gray
);
    reg [3:0] gray_reg;
    always @(*)
        case(bin)
            4'b0000: gray_reg = 4'b0000;
```

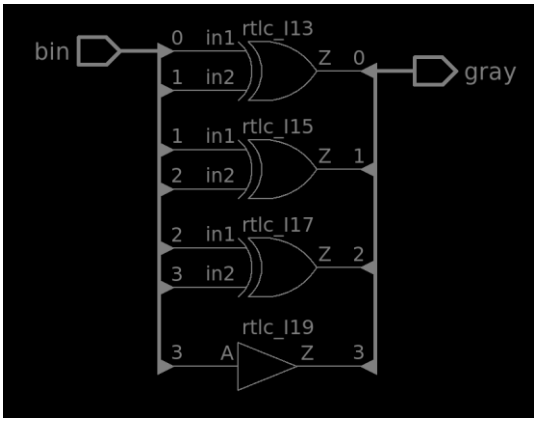
```
            4'b0001: gray_reg = 4'b0001;
            4'b0010: gray_reg = 4'b0011;
            4'b0011: gray_reg = 4'b0010;
            4'b0100: gray_reg = 4'b0110;
            4'b0101: gray_reg = 4'b0111;
            4'b0110: gray_reg = 4'b0101;
            4'b0111: gray_reg = 4'b0100;

            4'b1000: gray_reg = 4'b1100;
            4'b1001: gray_reg = 4'b1101;
            4'b1010: gray_reg = 4'b1111;
            4'b1011: gray_reg = 4'b1110;
            4'b1100: gray_reg = 4'b1010;
            4'b1101: gray_reg = 4'b1011;
            4'b1110: gray_reg = 4'b1001;
            4'b1111: gray_reg = 4'b1000;
            default: gray_reg = 4'b0000;
        endcase

    assign gray = gray_reg;
endmodule




module bin2gray_N
#(
    parameter N = 4
)(
    input [N-1:0] bin,
    output [N-1:0] gray
);
    assign gray = bin ^ (bin >> 1);
endmodule
```

电路原理图：

4 位：



N 位：



波形图：

格雷码到二进制码：

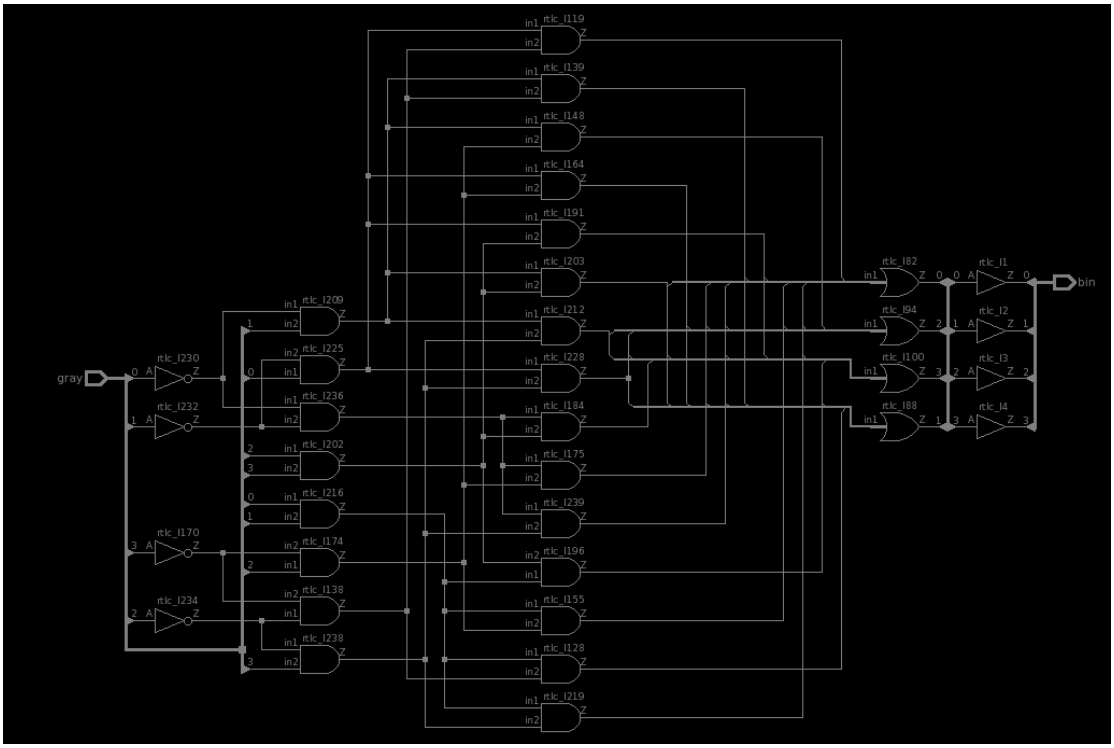
verilog 代码：

```
module gray2bin_4(
    input [3:0] gray,
    output [3:0] bin
);
    reg [3:0] bin_reg;
```

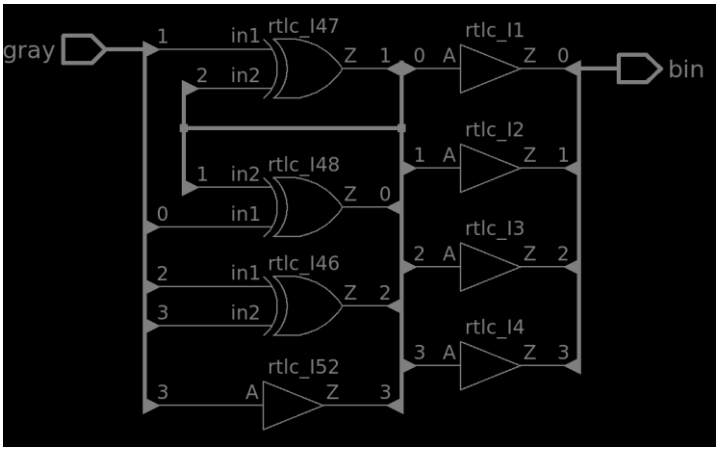
```
    always @(*)
        case(gray)
           4'b0000: bin_reg = 4'b0000;
           4'b0001: bin_reg = 4'b0001;
           4'b0011: bin_reg = 4'b0010;
           4'b0010: bin_reg = 4'b0011;
           4'b0110: bin_reg = 4'b0100;
           4'b0111: bin_reg = 4'b0101;
           4'b0101: bin_reg = 4'b0110;
           4'b0100: bin_reg = 4'b0111;

           4'b1100: bin_reg = 4'b1000;
           4'b1101: bin_reg = 4'b1001;
           4'b1111: bin_reg = 4'b1010;
           4'b1110: bin_reg = 4'b1011;
           4'b1010: bin_reg = 4'b1100;
           4'b1011: bin_reg = 4'b1101;
           4'b1001: bin_reg = 4'b1110;
           4'b1000: bin_reg = 4'b1111;
            default: bin_reg = 4'b0000;
        endcase

    assign bin = bin_reg;
endmodule




module gray2bin_N
#(
    parameter N = 4
)(
    input [N-1:0] gray,
    output [N-1:0] bin
);
    reg [N-1:0] bin_reg;
    integer i;
    always @(*) begin
        bin_reg[N-1] = gray[N-1];
        for(i = N-2; i >= 0; i = i - 1)
            bin_reg[i] = gray[i] ^ bin_reg[i+1];
    end
    assign bin = bin_reg;
endmodule
```

电路原理图：

4 位：



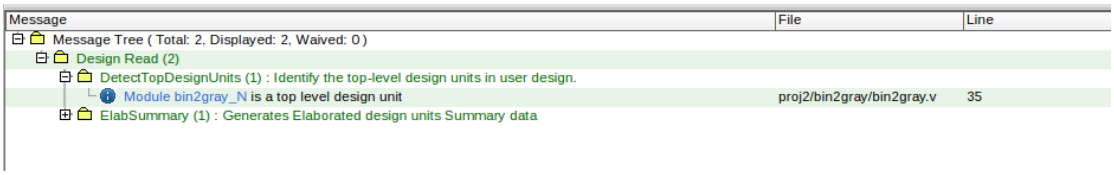
N 位：

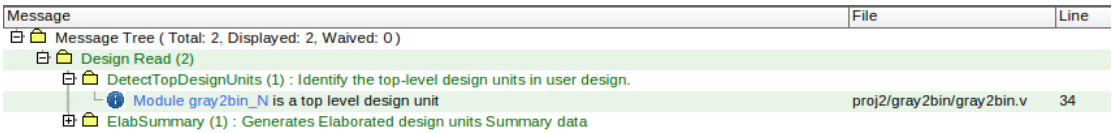


可以看到在计算 bin 时需要更早得到的 bin 的位作反馈输入，所以增加了 buffer。

波形图：

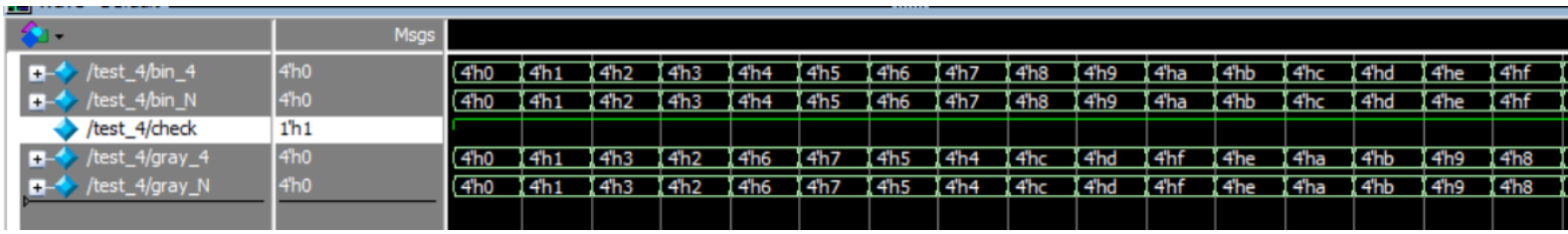
spyglass 报告：PASS





五、 仿真结果

二进制码转格雷码：



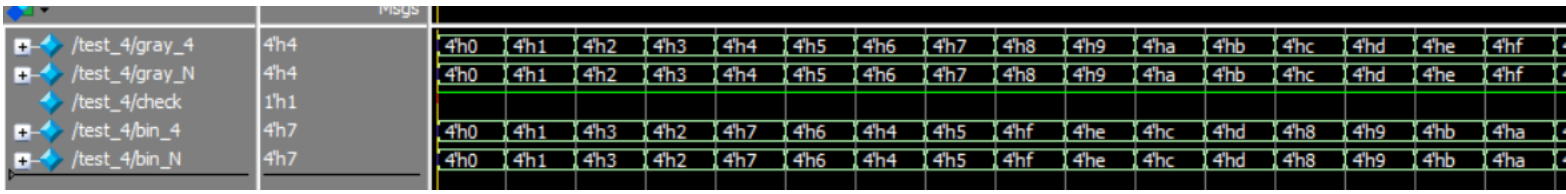
结果和预期的波形图一致。

testbench：

```
`timescale 1ps/1ps
module test_4#(
    parameter N=4
)(

);
    reg [3:0] bin_4;
    wire [3:0] gray_4;
    reg [N-1:0] bin_N;
    wire [N-1:0] gray_N;

    bin2gray_4 u_4 (
        .bin(bin_4),
        .gray(gray_4)
    );

    bin2gray_N #(
        .N(N)
    ) u_N (
        .bin(bin_N),
        .gray(gray_N)
    );
```

```
    initial begin
        bin_4 = 0;
        while(bin_4 <= 15)
            #10 bin_4 = bin_4 + 1;
    end

    initial begin
        bin_N = 0;
        while(bin_N <= 2**N-1)
            #10 bin_N = bin_N + 1;
    end

    wire check;
    assign check = bin_N == bin_4;

endmodule
```

格雷码转二进制码：



结果和预期的波形图一致。

testbench：

```
`timescale 1ps/1ps
module test_4#(
    parameter N=4
)(

);
    wire [3:0] bin_4;
    reg [3:0] gray_4;
    wire [N-1:0] bin_N;
    reg [N-1:0] gray_N;

    gray2bin_4 u_4 (
        .bin(bin_4),
        .gray(gray_4)
    );
```

```
    gray2bin_N #(
        .N(N)
    ) u_N (
        .bin(bin_N),
        .gray(gray_N)
    );

    initial begin
        gray_4 = 0;
        while(gray_4 <= 15)
            #10 gray_4 = gray_4 + 1;
    end

    initial begin
        gray_N = 0;
        while(gray_N <= 2**N-1)
            #10 gray_N = gray_N + 1;
    end

    wire check;
    assign check = gray_N == gray_4;

endmodule
```

六、 实验总结

本次实验我了解到二进制码和格雷码的转换原理，学习了通过移位运算方便二进制码转格雷码的原理，通过这种方法可以使代码更加简洁，可读性更好。

对于有限位的转换可以使用译码器穷举所有情况，虽然消耗逻辑资源较多，但是简单易懂。注意写代码时不要写错。

格雷码转二进制码的组合逻辑需要借用先前计算出的二进制高位结果，这和级联的加法器很像，但是使用 for 简化例化过程比较优雅。但是注意在写 rtl 不能用 for 综合，如果不是因为例化数位 N 不好完全例化完，就尽量不要使用 for 语句块。