中国科学技术大学计算机学院

《数字电路实验》报告

实验题目：编码器和译码器

学生姓名：张如凡

学生学号：PB20051054

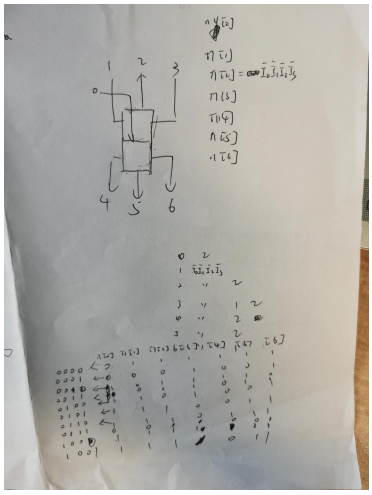
完成日期：10.28.2021

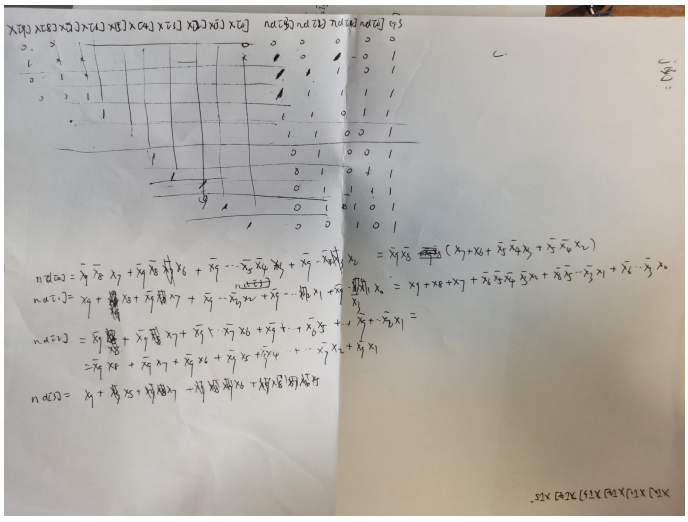
计算机实验教学中心制

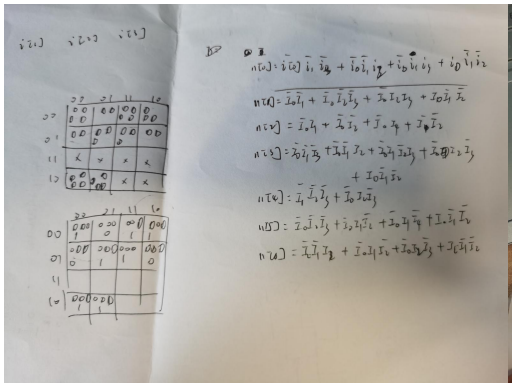
2020 年 09 月

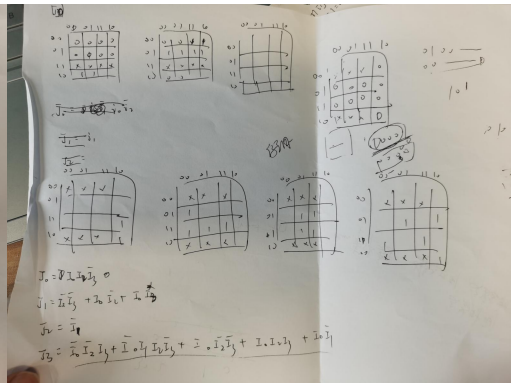
逻辑设计：

本实验主要是编码器和译码器的先后使用（对 0-9 这 10 个数字用普通编码器和优先编码器进行编码，再用不同的译码器进行使得 LED 和七段数码管正确点亮），因此主要的逻辑设计是译码器和编码器的设计（下面是设计编码器和译码器时的手稿图片）：









对设计思路进行解释：

1. 普通编码器（低电平有效）：输入为四位二进制码（0000-1001），输出为 8421 码，由此画出真值表，写出最简表达式。由于普通编码器的设计思路是一个输出只对应一种输入，且在化简表

达试时用到了无关项，因此输入不满足设计条件时会出错。

2. 译码器：两个译码器的输出结果不同，但设计思路一样，都是根据输入输出之间的逻辑关系。画出真值表后，写出最简表达式（用到了无关项）。注意用于优先编码器的译码器只需要增加电路将余三循环码转化为 8421 码。

3. 优先编码器（高电平有效）：跟普通编码器不同的时，输入时较高位的优先级较高，所以输入优先取决较高位，且输出是余三循环码。接下来与普通编码器一样，写出真值表、表达式。

理解各个编码器和译码器的逻辑，得到相关输出的表达式后，进行各个模块的设计（代码编写），最后进行整合。

核心代码：

普通编码器的代码

```
module normalcode0(
    input [9:0] x,
    output [3:0] nd,
    output GS
    );
    assign nd[0]=~(x[9]&x[8]);
    assign nd[1]=~(x[6]&x[7]&x[5]&x[4]);
    assign nd[2]=~(x[6]&x[7]&x[2]&x[3]);
    assign nd[3]= ~(x[9]&x[7]&x[5]&x[3]&x[1]);
    assign w=~(nd[0]|nd[1]|nd[2]|nd[3]);
    assign GS=~(x[0]&w);
endmodule
```

优先编码器的代码

```
module prioritycode(
    input [9:0] x,
    output [3:0] nd,
    output GS
    );
    assign nd[0]=~x[9]&~x[8]&(x[7]|x[6]|(~x[5]&~x[4]&x[3])|(~x[5]&~x[4]&x[2]));
```

```
    assign nd[1]= x[9]|x[8]|x[7]|((~x[6]&~x[5]&~x[4]&~x[3])&(x[2]|x[1]|x[0]));
    assign nd[2]=~x[9]&(x[8]|x[7]|x[6]|x[5]|x[4]|x[3]|x[2]|x[1]);
    assign nd[3]= (x[9]|x[8]|x[7]|x[6]|x[5]);
    assign GS=x[9]|x[8]|x[7]|x[6]|x[5]|x[4]|x[3]|x[2]|x[1]|x[0];
Endmodule
```

LED 码器的代码

```
module normaldecode(
    input [3:0] i,
    input EI,
    output [9:0] y
    );
    assign y[0]=~i[0]&~i[1]&~i[2]&~i[3]&EI;
    assign y[1]=~i[0]&~i[1]&~i[2]&i[3]&EI;
    assign y[2]=~i[0]&~i[1]&i[2]&~i[3]&EI;
    assign y[3]=~i[0]&~i[1]&i[2]&i[3]&EI;
    assign y[4]=~i[0]&i[1]&~i[2]&~i[3]&EI;
    assign y[5]=~i[0]&i[1]&~i[2]&i[3]&EI;
    assign y[6]=~i[0]&i[1]&i[2]&~i[3]&EI;
    assign y[7]=~i[0]&i[1]&i[2]&i[3]&EI;
    assign y[8]=i[0]&~i[1]&~i[2]&~i[3]&EI;
    assign y[9]=i[0]&~i[1]&~i[2]&i[3]&EI;
Endmodule
```

七段数码管的代码

```
module prioritydecode(
    input [3:0] I,
    input EI,
    output [7:0] n
    );
    assign
n[0]=~((~I[0]&~I[1]&~I[3])|(~I[0]&~I[1]&I[2])|(~I[0]&I[1]&I[3])|(I[0]&~I[1]&~I[2])|(~I[0]&I[1]&I[2]&~I[3]))|~E
I;
    assign n[1]=~((~I[0]&~I[1])|(~I[0]&~I[2]&~I[3])|(~I[0]&I[2]&I[3])|(I[0]&~I[1]&~I[2]))|~EI;
    assign n[2]=~((~I[0]&I[1])|(~I[0]&~I[2])|(~I[0]&I[3])|(~I[1]&~I[2]))|~EI;
    assign
n[3]=~((~I[0]&~I[1]&~I[3])|(~I[0]&~I[1]&I[2])|(~I[0]&I[1]&~I[2]&I[3])|(~I[0]&I[1]&~I[2]&I[3])|(~I[0]&I[2]&~I[3]
)|(I[0]&~I[1]&~I[2]))|~EI;
    assign n[4]=~((~I[1]&~I[2]&~I[3])|(~I[0]&I[2]&~I[3]))|~EI;
    assign n[5]=~((~I[0]&~I[2]&~I[3])|(~I[0]&I[1]&~I[2])|(~I[0]&I[1]&~I[3])|(I[0]&~I[1]&~I[2]))|~EI;
    assign n[6]=~((~I[0]&~I[1]&I[2])|(~I[0]&I[1]&~I[2])|(~I[0]&I[2]&~I[3])|(I[0]&~I[1]&~I[2]))|~EI;
Endmodule
```

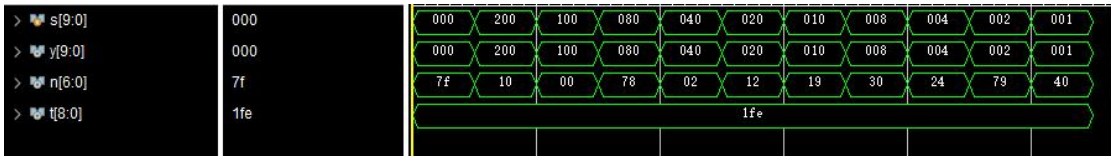
余三码转 8421 码的代码：

```
assign i[0]=j[3]&j[1]&~j[0];
assign i[1]=(~j[1]&~j[0])|(j[3]&~j[1])|(j[0]&j[3]);
assign i[2]=j[0];
assign i[3]=(~j[3]&~j[1]&j[0])|(~j[0]&j[1]&j[2]&~j[3])|(j[3]&~j[1]&~j[0])|(j[0]&j[1]&j[3])|(j[3]&~j[2]);
```

仿真结果：

普通编码器：

优先编码器：



结果分析：

对应相应输入的输出结果是正确的，能够使相应的 LED 正确点亮，七段数码管正确显示数字。可以看到普通编码器电路和优先编码器电路对应相同的输入（指的是输入代表的数字相同），其输出结果也是相同的。这与我们的设计初衷相符，更加证明了我们电路设计的正确性。

实验总结：

1. 通过本次实验，掌握了使用 vivado 设计电路，进行仿真，下载测试的方法。

2. 通过自己设计编码器的译码器，更加深刻的理解的它们的逻辑和电路结构。

3. 在实验中，提高了自己使用 verilog 描述电路的技能，使得课堂的学习得到了实践。

意见/建议：

希望在做第一个实验之前，助教能发一个视频教程（就是助教用 vivado 做一个简单的实验完整演示一下），我觉得这样能节约同学很多时间，因为我在弄懂如何用 vivado 做实验花了很多时间。

设计和测试文件：

普通编码器电路

```
module sum1(
    input [9:0] s,
    output [9:0] y,
    output [6:0] n,
    output [8:0] t
    );
   wire  [3:0] z;
   wire GS;
   assign t=9'b1_1111_1110;
   normalcode0 M0 (s[9:0],z[3:0],GS);
   normaldecode M1(z[3:0],GS,y[9:0]);
   prioritydecode M2(z[3:0],GS,n[6:0]);
endmodule
```

```
module sum_1_tb(
    );
    reg [9:0] s;
    wire [9:0] y;
    wire [6:0] n;
    wire [8:0] t;
    sum1 M3(s[9:0],y[9:0],n[6:0],t[8:0]);
    initial begin
        s=10'b1111_1111_11;
        #10 s=10'b1111_1111_10;
        #10 s=10'b1111_1111_01;
        #10 s=10'b1111_1110_11;
        #10 s=10'b1111_1101_11;
        #10 s=10'b1111_1011_11;
        #10 s=10'b1111_0111_11;
        #10 s=10'b1110_1111_11;
        #10 s=10'b1101_1111_11;
        #10 s=10'b1011_1111_11;
        #10 s=10'b0111_1111_11;
        #10 $finish;
    end
endmodule
```

优先编码器

```
module sum3(
    input [9:0] s,
    output [9:0] y,
    output [6:0] n,
    output [8:0] t
    );
   wire  [3:0] z;
   wire GS;
   assign t=9'b1_1111_1110;
   prioritycode M0 (s[9:0],z[3:0],GS);
   normalcode1 M1(z[3:0],GS,y[9:0]);
   prioritydecode2 M2(z[3:0],GS,n[6:0]);
endmodule
```

```
module sum3_tb(
    );
    reg [9:0] s;
    wire [9:0] y;
    wire [6:0] n;
    wire [8:0] t;
    sum3 M3(s[9:0],y[9:0],n[6:0],t[8:0]);
    initial begin
        s=10'b0000_0000_00;
        #10 s=10'b1000_0000_00;
        #10 s=10'b0100_0000_00;
        #10 s=10'b0010_0000_00;
        #10 s=10'b0001_0000_00;
        #10 s=10'b0000_1000_00;
        #10 s=10'b0000_0100_00;
        #10 s=10'b0000_0010_00;
        #10 s=10'b0000_0001_00;
        #10 s=10'b0000_0000_10;
        #10 s=10'b0000_0000_01;
        #10 $finish;
    end
endmodule
```