**实验一：译码器和编码器**

实验时间：2017年10月30日 第八周 星期一

实验者：16307130194 陈中钰 16级 计算机科学技术学院

座位号：30

指导老师：唐志强

1. 实验目的

* 通过实现计算机系统中最常用的逻辑部件之一的译码器，来熟悉实验操作、设计流程
* 了解并掌握自顶向下的设计流程，通过实现模块化，了解分层设计思想
* 了解译码器原理和作用，并设计、实现译码器，完成对操作码的译码
* 了解编码器的分类、原理和作用，并设计、实现译码器，完成对电平信号的编码
* 了解并掌握de Morgan’s Law

1. 实验原理
   1. 74LS138：3-8译码器

* 输入：S[2:0]；输出：Y[7:0]
* 真值表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 输入 | | | 输出 | | | | | | | |
| S2 | S1 | S0 | Y7 | Y6 | Y5 | Y4 | Y3 | Y2 | Y1 | Y0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

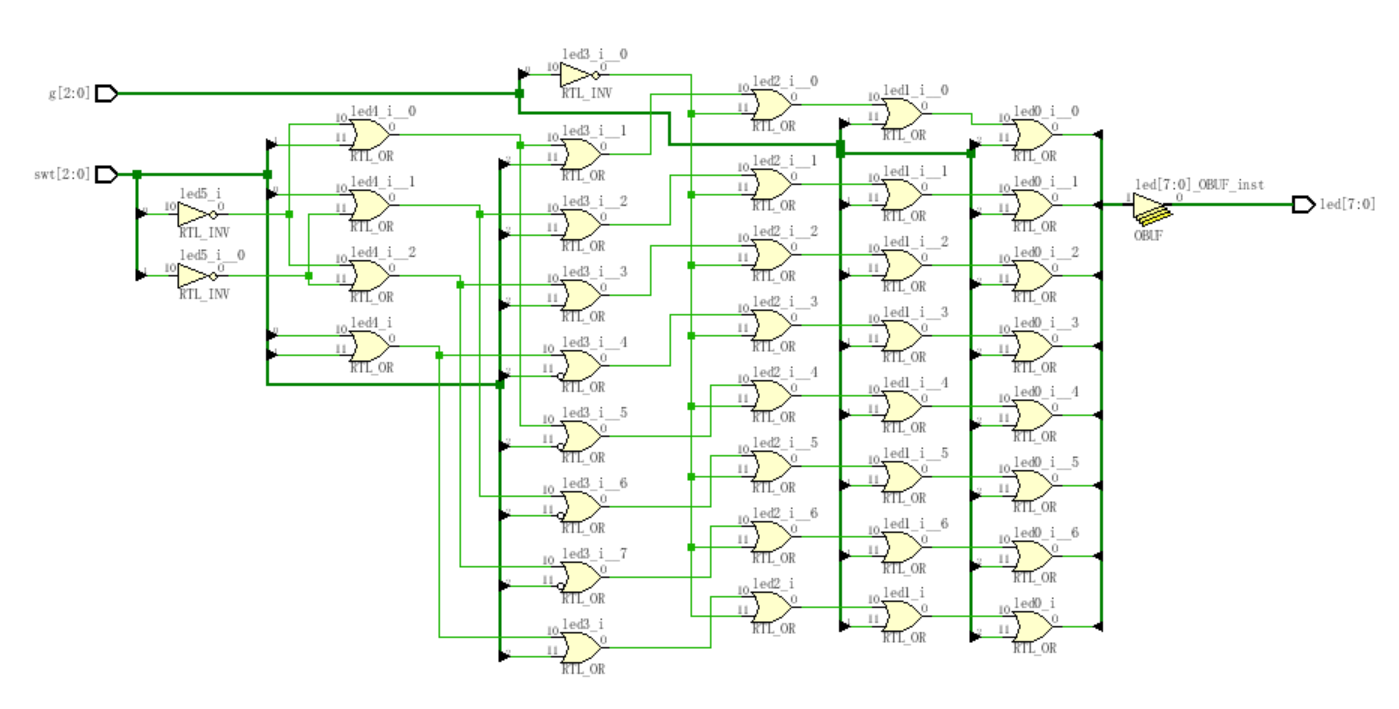
* K-Map（以Y0为例）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S0 S2S1 | 00 | 01 | 11 | 10 |
| 0 |  | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

可得反相Y0’=S0’S1’S2’，那么Y0=S0+S1+S2

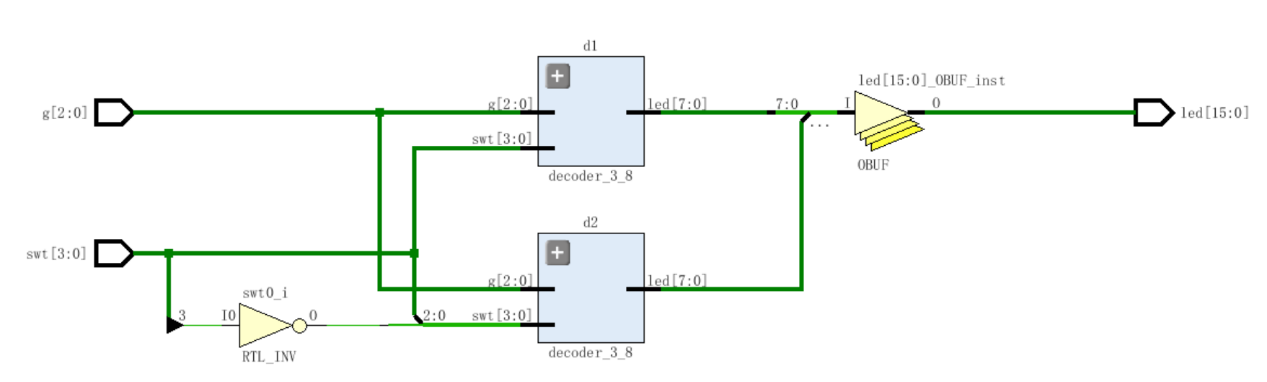
* bool代数式

|  |  |  |  |
| --- | --- | --- | --- |
| Y0=S0+S1+S2 | Y1=S0’+S1+S2 | Y2=S0+S1’+S2 | Y3=S0’+S1’+S2 |
| Y4=S0+S1+S2’ | Y5=S0’+S1+S2’ | Y6=S0+S1’+S2’ | Y7=S0’+S1’+S2’ |

* 逻辑电路图（带有三个控制开关）
  1. 利用上述3-8译码器和与非门，设计一个4-16译码器
* 采用的是分层设计的思想
* 用3-8译码器构造4-16译码器和用2-4译码器构造3-8译码器原理类似，相比于3-8译码器，多一个A[3]输入，共4个输入，若A[3]为0，与3-8译码器的输出D[7:0]组合，可以有8种输出，当A[3]为1时，还能组合出另外8种，一共16种的输出
* 真值表

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 输入S | | | | 第二个3-8译码器的输出Y | | | | | | | | 第一个3-8译码器的输出Y | | | | | | | |
| S3 | S2 | S1 | S0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | | | | | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 1 | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | | | | | | | |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

* 逻辑电路图



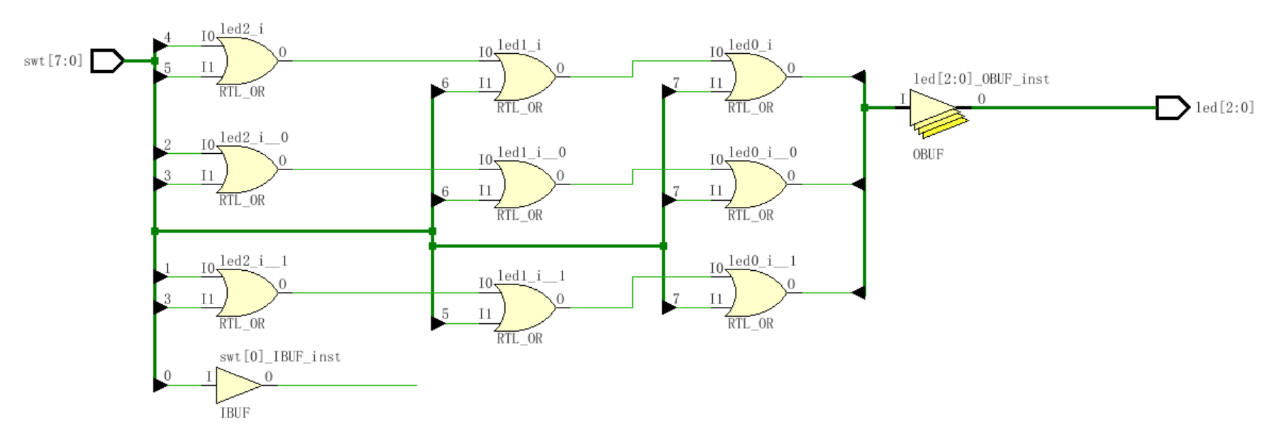
* 1. 8-3普通编码器
* 输入：I[7:0]；输出：F[2:0]
* 真值表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 输入 | | | | | | | | 输出 | | |
| I7 | I6 | I5 | I4 | I3 | I2 | I1 | I0 | F0 | F1 | F2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

* bool代数式（可以直接观察得到）

|  |  |  |
| --- | --- | --- |
| F0=I4+I5+I6+I7 | F1=I2+I3+I6+I7 | F2=I1+I3+I5+I7 |

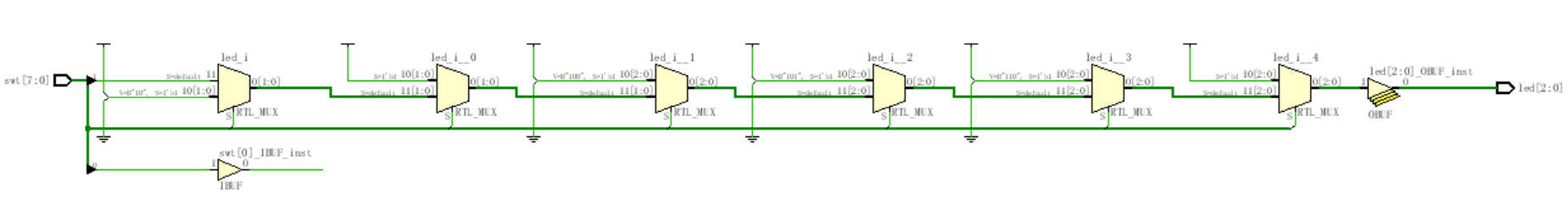
* 逻辑电路图



* 1. 8-3优先编码器
* 输入：I[7:0]；输出：F[2:0]
* 真值表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 输入 | | | | | | | | 输出 | | |
| I7 | I6 | I5 | I4 | I3 | I2 | I1 | I0 | F0 | F1 | F2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | X | X | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | X | X | X | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | X | X | X | X | 1 | 0 | 0 |
| 0 | 0 | 1 | X | X | X | X | X | 1 | 0 | 1 |
| 0 | 1 | X | X | X | X | X | X | 1 | 1 | 0 |
| 1 | X | X | X | X | X | X | X | 1 | 1 | 1 |

* 通过if else结构可以很好地表达“优先”的效果，而不需要求出bool代数式，描述会更为简洁、易懂
* 逻辑电路图



1. 实验内容
   1. 74LS138：3-8译码器

* 要求加入三个控制信号：G1，G2A，G2B，当G1=1，G2A=G2B=0时实现正常编码，否则Y[7:0]=7’b1111111。那么可以在原来的基础上给Y的每一位OR (~G1+G2A+G2B)，当满足条件时这一部分为0，对原结果没影响，否则这一部分为1，Y每一位结果都是1，为高电平
* 控制信号表

|  |  |  |
| --- | --- | --- |
| G1G2G2B | ~G1+G2A+G2B | 输出 |
| 100 | 0 | 不变 |
| 其他 | 1 | 全1 |

* Verilog代码实现

module decoder\_3\_8(input [2:0] swt, input [2:0] g, output [7:0] led);

assign led[0]=swt[0] | swt[1] | swt[2]| ~g[0] | g[1] | g[2];

assign led[1]=~swt[0] | swt[1] | swt[2]| ~g[0] | g[1] | g[2];

assign led[2]=swt[0] | ~swt[1] | swt[2]| ~g[0] | g[1] | g[2];

assign led[3]=~swt[0] | ~swt[1] | swt[2]| ~g[0] | g[1] | g[2];

assign led[4]=swt[0] | swt[1] | ~swt[2]| ~g[0] | g[1] | g[2];

assign led[5]=~swt[0] | swt[1] | ~swt[2]| ~g[0] | g[1] | g[2];

assign led[6]=swt[0] | ~swt[1] | ~swt[2]| ~g[0] | g[1] | g[2];

assign led[7]=~swt[0] | ~swt[1] | ~swt[2]| ~g[0] | g[1] | g[2];

endmodule

* 1. 利用上述3-8译码器和与非门，设计一个4-16译码器
* 要构造出4-16译码器，则需要2个3-8译码器
* 需要对上述3-8译码器进行改造，添加多一个swt[3]的输入，作用与使能相似，输入两个3-8译码器的值是相反的，使其中一个正常工作，另一个则全亮，那么可以在原来的基础上给每一个输出OR swt[3]，那么，swt[3]=1的3-8译码器则输出全亮，而另一个3-8译码器的swt[3]=0，正常工作，那么所有的16个灯仅1个灯不亮，符合要求

module decoder\_4\_16(input [3:0] swt, input [2:0] g, output [15:0] led);

decoder\_3\_8 d1( {swt[3],swt[2:0]}, g[2:0], led[7:0]);

decoder\_3\_8 d2( {~swt[3],swt[2:0]}, g[2:0], led[15:8]);

endmodule

module decoder\_3\_8(input [3:0] swt, input [2:0] g, output [7:0] led);

assign led[0]=swt[0] | swt[1] | swt[2]| ~g[0] | g[1] | g[2]| swt[3];

assign led[1]=~swt[0]|swt[1] | swt[2]| ~g[0] | g[1] | g[2]| swt[3];

assign led[2]=swt[0]|~swt[1] | swt[2]| ~g[0] | g[1] | g[2]| swt[3];

assign led[3]=~swt[0]|~swt[1] | swt[2]| ~g[0] | g[1] | g[2]| swt[3];

assign led[4]=swt[0]|swt[1] | ~swt[2]| ~g[0] | g[1] | g[2]| swt[3];

assign led[5]=~swt[0]|swt[1] | ~swt[2]| ~g[0] | g[1] | g[2]| swt[3];

assign led[6]=swt[0]|~swt[1] | ~swt[2]| ~g[0] | g[1] | g[2]| swt[3];

assign led[7]=~swt[0]|~swt[1]| ~swt[2]| ~g[0] | g[1] | g[2]| swt[3];

endmodule

* 1. 8-3普通编码器

module encoder\_8\_3(input [7:0] swt,output [2:0] led);

assign led[0]=swt[4]|swt[5]|swt[6]|swt[7];

assign led[1]=swt[2]|swt[3]|swt[6]|swt[7];

assign led[2]=swt[1]|swt[3]|swt[5]|swt[7];

endmodule

* 1. 8-3优先编码器

module priority\_encoder\_8\_3(input [7:0] swt, output reg [2:0] led);

always@(swt)

begin

if(swt[7]) led=3'b111; else if(swt[6]) led=3'b110; else if(swt[5]) led=3'b101;

else if(swt[4]) led=3'b100; else if(swt[3]) led=3'b011; else if(swt[2]) led=3'b010;

else if(swt[1]) led=3'b001; else if(swt[0]) led=3'b000; else led=3'b000;

end

endmodule

1. 实验结论

* 实现了带3个控制的3-8译码器
* 使用3-8译码器构造了4-16译码器
* 实现了8-3普通编码器
* 实现了8-3优先编码器

1. 实验感想

* 能更熟练地使用Vivado，对Verilog语言更加熟悉了
* 对译码器、编码器的功能和设计更加了解
* 了解了分层设计、模块化的思想，为之后的实验设计打下基础（后续实验会经常用到模块化）