

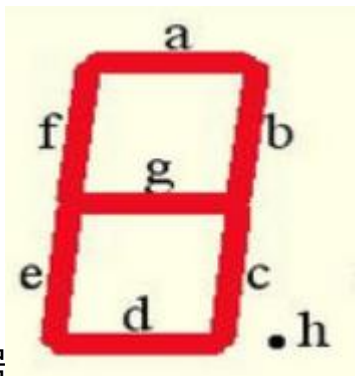
## 实验报告

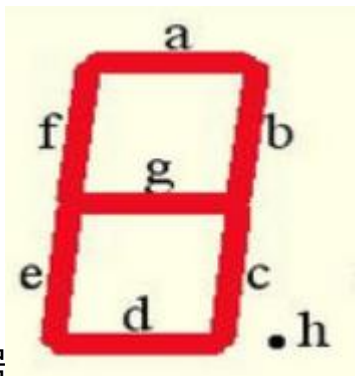
### ·实验目的：

- 通过本次实验，达成以下目标
  - 能够根据实际要求，设计出译码逻辑电路
  - 掌握从真值表、卡诺图到 Verilog 代码的整个设计过程
  - 进一步熟悉 FPGA 及 ISE 工具的使用
- 本实验用到的外设包括
  - 拨动开关（4 个）
  - LED 灯（8 个）
  - 7 段数码管
- 实现一编码逻辑，将 4 个拨动开关的状态按 16 进制显示在 7 段数码管上
- 实现一 3-8 译码器，将低 3 位开关状态做为编号，与编号对应的 LED 灯亮，其余灯灭

### ·实验内容

将低四位 switch 从高到低设置为 A,B,C,D。



根据 ，与 ABCD 的关系列出真值表，并用卡诺图化简，的下一表达式。（数字管为低电平有效）

```

12  a = ~A & ~B & ~C & D | ~A & B & ~C & ~D | A & ~B & C & D | A & B & ~C & D;
13  b = B & C & ~D | A & C & D | A & B & ~D | A & B & C | ~A & B & ~C & D;
14  c = A & B & ~D | A & B & C | ~A & ~B & C & ~D;
15  d = B & C & D | ~A & ~B & ~C & D | ~A & B & ~C & ~D | A & ~B & C & ~D;
16  e = ~A & D | ~B & ~C & D | ~A & B & ~C;
17  f = ~A & C & D | ~A & ~B & D | ~A & ~B & C | A & B & ~C & D;
18  g = ~A & ~B & ~C | ~A & B & C & D | A & B & ~C & ~D;

```

led 控制与上述同理，表达式为

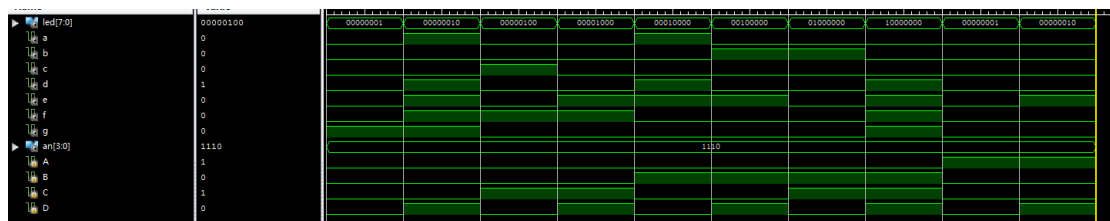
```

19  led[0] = ~B & ~C & ~D;
20  led[1] = ~B & ~C & D;
21  led[2] = ~B & C & ~D;
22  led[3] = ~B & C & D;
23  led[4] = B & ~C & ~D;
24  led[5] = B & ~C & D;
25  led[6] = B & C & ~D;
26  led[7] = B & C & D;

```

## ·实验结果

仿真图如下：



## ·实验分析

通过本实验，进一步学习了 verilog 的语法，知道了 nexys 板上个按键与 ucf 文件中值得对应关系，同时，实现 3-8 译码器复习了相关知识。

## ·意见建议

这次 ppt 中的要求存在二义性，交代有些不清。

建议:若理论课能够加强 verilog 语法的讲解，那么问题会少很多很多，我们可以将更多的精力放在实验的思想上，而不是语法的学习上。

·附录：

源代码如下：

```
module top(A,B,C,D,led,a,b,c,d,e,f,g,an);
```

```
input          A,B,C,D;
```

```
output reg [7:0] led;
```

```
output reg      a,b,c,d,e,f,g;
```

```
output [3:0] an;
```

```
assign an[0]=0;
```

```
assign an[1] = 1;
```

```
assign an[2] = 1;
```

```
assign an[3] = 1;
```

```
always @(A or B or C or D) begin
```

```
    a = ~A & ~B & ~C & D | ~A & B & ~C & ~D | A & ~B &  
C & D | A & B & ~C & D;
```

```
    b = B & C & ~D | A & C & D | A & B & ~D | A & B & C |  
~A & B & ~C & D;
```

```
    c = A & B & ~D | A & B & C | ~A & ~B & C & ~D;
```

```
d = B & C & D | ~A & ~B & ~C & D | ~A & B & ~C & ~D  
| A & ~B & C & ~D;
```

```
e = ~A & D | ~B & ~C & D | ~A & B & ~C;
```

```
f = ~A & C & D | ~A & ~B & D | ~A & ~B & C | A & B &  
~C & D;
```

```
g = ~A & ~B & ~C | ~A & B & C & D | A & B & ~C & ~D;
```

```
led[0] = ~B & ~C & ~D;
```

```
led[1] = ~B & ~C & D;
```

```
led[2] = ~B & C & ~D;
```

```
led[3] = ~B & C & D;
```

```
led[4] = B & ~C & ~D;
```

```
led[5] = B & ~C & D;
```

```
led[6] = B & C & ~D;
```

```
led[7] = B & C & D;
```

```
end
```

```
endmodule
```

```

1  module top(A,B,C,D,led,a,b,c,d,e,f,g,an);
2  input      A,B,C,D;
3  output reg [7:0] led;
4  output reg  a,b,c,d,e,f,g;
5  output [3:0] an;
6  assign an[0]=0;
7  assign an[1] = 1;
8  assign an[2] = 1;
9  assign an[3] = 1;
10
11  always @(A or B or C or D) begin
12      a = ~A & ~B & ~C & D | ~A & B & ~C & ~D | A & ~B & C & D | A & B & ~C & D;
13      b = B & C & ~D | A & C & D | A & B & ~D | A & B & C | ~A & B & ~C & D;
14      c = A & B & ~D | A & B & C | ~A & ~B & C & ~D;
15      d = B & C & D | ~A & ~B & ~C & D | ~A & B & ~C & ~D | A & ~B & C & ~D;
16      e = ~A & D | ~B & ~C & D | ~A & B & ~C;
17      f = ~A & C & D | ~A & ~B & D | ~A & ~B & C | A & B & ~C & D;
18      g = ~A & ~B & ~C | ~A & B & C & D | A & B & ~C & ~D;
19      led[0] = ~B & ~C & ~D;
20      led[1] = ~B & ~C & D;
21      led[2] = ~B & C & ~D;
22      led[3] = ~B & C & D;
23      led[4] = B & ~C & ~D;
24      led[5] = B & ~C & D;
25      led[6] = B & C & ~D;
26      led[7] = B & C & D;
27  end
28
29  endmodule

```

原理图略。