组合逻辑实验报告

17 May 2014

目录

实验一 多路选择器设计 3
实验目的：掌握组合逻辑的基本设计方法；掌握多路选择器的基本原理。 3
实验原理： 3
（1） 二选一多路选择器 3
（2） 四选一多路选择器 3
（3） 二选一多路选择器的竞争冒险分析 3
关键代码 ： 4
文件清单 5
仿真结果 6
综合情况 6
硬件调试情况 6
实验二 译码器设计 7
实验目的： 7
实验原理： 7
（1） 三八译码器 7
（2） 数码管七段译码器 7
关键代码 8
文件清单 9
仿真情况 10

综合情况..... 10
硬件调试情况..... 10
实验三 **4** 位加法器设计 10
实验目的：..... 10
实验原理：..... 10
1、 一位全加器..... 10
2、 4bits 逐次进位加法器..... 10
3、 4bits 超前进位全加器..... 11
关键代码..... 11
文件清单..... 15
仿真结果..... 16
综合情况..... 16
硬件调试情况..... 16

实验一 多路选择器设计

实验目的：掌握组合逻辑的基本设计方法；掌握多路选择器的基本原理。

实验原理：

（1） 二选一多路选择器

如图 1 所示为一个二选一多路选择器，开关由一根控制线 s 控制。s 选择两个输入中的的一个作为输出，即输出 y 的逻辑值和被选中的那个输入的逻辑值相同。由表 1 所示的真值表，可以看到，当 s = 0 时，y = a；当 s = 1 时，

y = b。我们可以得到 y 的逻辑方程：$y = \overline{s} \cdot a + s \cdot b$

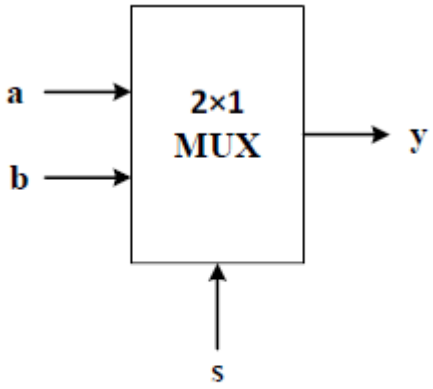


图 1 二选一多路选择器

表 1 二选一多路选择器真值表

s	b	a	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

（2） 四选一多路选择器

四选一多路选择器如图 4 所示。开关由两个控制线 s_0 和 s_1 控制。这两位控制选择 4 个输入中的一个作为输出。

（3） 二选一多路选择器的竞争冒险分析

给二选一多路选择器的各个逻辑门设置相同的传输延时，通过程序仿真观察竞争冒险现象。

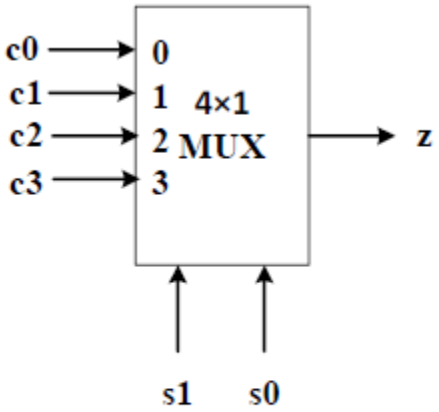


图 4 四选一多路选择器

关键代码 ：

2 选 1

```
module mux(a,b,s,y);
        input   a,b,s;
        output y;
        wire a,b,s,y;
        assign y=(~s&&a)||(s&&b);
endmodule
```

4 选 1

```
module mux(c0,c1,c2,c3,s1,s0,z);
        input wire c0,c1,c2,c3,s1,s0;
        output wire z;
        assign z=(~s1&&~s0)&&c0 || (~s1&&s0)&&c1 ||(s1&&~s0)&&c2 || (s1&&s0)&&c3;
endmodule
```

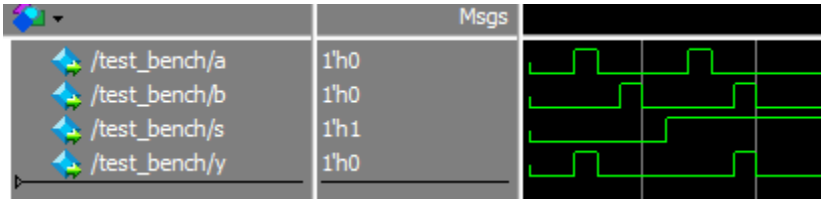
冒险分析

```
module mux(a,b,s,y);
    input  a,b,s;
    output y;
    wire a,b,s,y;
    assign #10 t1=~s;
    assign #10 t2=t1&&a;
    assign #10 t3=s&&b;
    assign #10 y=t2||t3;
endmodule
```

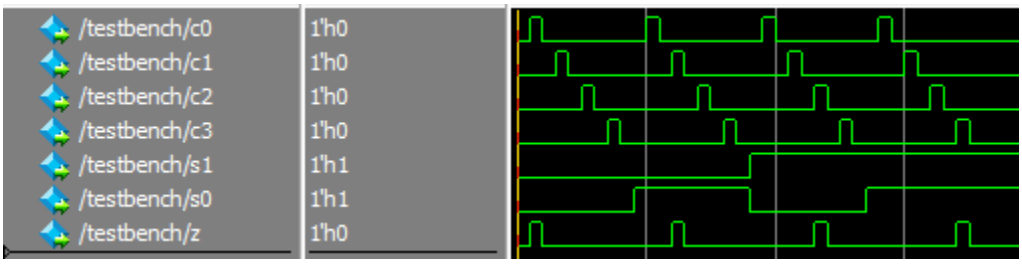
文件清单

MUX21.v	2 选 1 多路选择器
MUX21withdelay.v	2 选 1 多路选择器冒险分析
MUX41.v	4 选 1 多路选择器
testbench1.v	2 选 1 多路选择器 testbench
testbench2.v	4 选 1 多路选择器 testbench
testbench3.v	2 选 1 多路选择器冒险分析 testbench

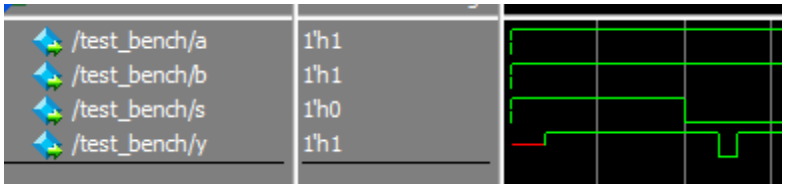
仿真结果



2 选 1 多路选择器功能可以正常实现。



4 选 1 多路选择器功能可以正常实现。



延时导致了冒险。

综合情况

2 选 1：	Total logic elements	1
	Worst-case	6.946 ns
4 选 1：	Total logic elements	2
	Worst-case	7.665 ns

硬件调试情况

调试顺利，未遇到困难。

实验二 译码器设计

实验目的：

掌握组合逻辑的基本设计方法；掌握译码器的设计原理。

实验原理：

（1） 三八译码器

三八译码器的框图如图 1 所示。真值表如表 1 所示。由真值表可以分析得到输出与输入的逻辑方程。

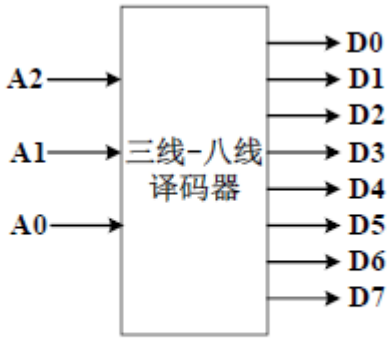


图 1 三八译码器框图

（2） 数码管七段译码器

7 个 LED 管排列成图 3 所示的模式，这样就形成了不同的位。DE2 开发板上有 8 个共阳极的 7 段数码管。在共阳极的情况下，FPGA 引脚输出 0 将点亮 LED，输出 1 则关掉 LED。表 2 所示的真值表给出了显示 0~F 的十六进制数所需的 a~g 的输出阴极值。

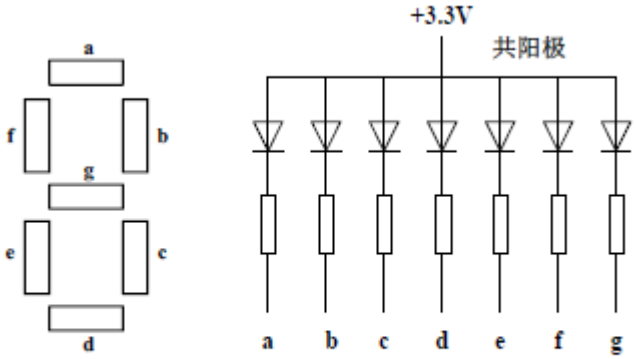


图 3 包含 7 个发光二极管（LED）的 7 段显示管

关键代码

38 译码器

```
module Ex2(A,D);

        input wire [2:0]A;

        output wire [7:0]D;

        assign D[7]=A[2]&&A[1]&&A[0];

        assign D[6]=A[2]&&A[1]&&~A[0];

        assign  D[5]=A[2]&&~A[1]&&A[0];

        assign D[4]=A[2]&&~A[1]&&~A[0];

        assign D[3]=~A[2]&&A[1]&&A[0];

        assign D[2]=~A[2]&&A[1]&&~A[0];

        assign D[1]=~A[2]&&~A[1]&&A[0];

        assign D[0]=~A[2]&&~A[1]&&~A[0];

endmodule
```

数码管七段译码器

```
module segment(X,H);
    input wire [3:0]X;
    output wire [6:0]H;
    assign H[0]=~X[3]&&~X[2]&&~X[1]&&X[0]||~X[3]&&X[2]&&~X[1]&&~X[0]||X[3]&&~X[2]&&X[1]&&X[0]||X[3]&&X[2]&&~X[1]&&X[0];
    assign H[1]=X[3]&&X[2]&&~X[1]&&~X[0]||~X[3]&&X[2]&&~X[1]&&X[0]||X[3]&&X[1]&&X[0]||X[2]&&X[1]&&~X[0];
    assign H[2]=X[3]&&X[2]&&~X[1]&&~X[0]||~X[3]&&~X[2]&&X[1]&&~X[0]||X[3]&&X[2]&&X[1];
    assign H[3]=~X[3]&&X[2]&&~X[1]&&~X[0]||~X[3]&&~X[2]&&~X[1]&&X[0]||X[2]&&X[1]&&X[0]||X[3]&&~X[2]&&X[1]&&~X[0];
    assign H[4]=~X[3]&&X[2]&&~X[1]||~X[3]&&X[0]||~X[2]&&~X[1]&&X[0];
    assign H[5]=X[3]&&X[2]&&~X[1]&&X[0]||~X[3]&&~X[2]&&X[0]||~X[3]&&X[1]&&X[0]||~X[3]&&~X[2]&&X[1];
    assign H[6]=X[3]&&X[2]&&~X[1]&&~X[0]||~X[3]&&X[2]&&X[1]&&X[0]||~X[3]&&~X[2]&&~X[1];
endmodule
```

文件清单

38.v　　38 译码器

7segment.v　数码管译码器

仿真情况

38 译码器和 7 段数码管译码器在硬件上的调试更加直观，故未进行仿真测试。

综合情况

38 译码器	Total logic elements	8
	Worst-case	6.757 ns
数码管译码器	Total logic elements	7
	Worst-case	7.378 ns

硬件调试情况

调试顺利，未遇到困难。

实验三 4 位加法器设计

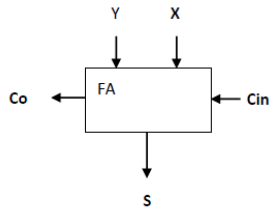
实验目的：

掌握运算组合逻辑设计方法；掌握加法器的基本设计原理；掌握逐次进位加法器和超前进位加法器的设计方法。

实验原理：

1、 一位全加器

一位全加器又称为(3,2)adder，示意图如右图

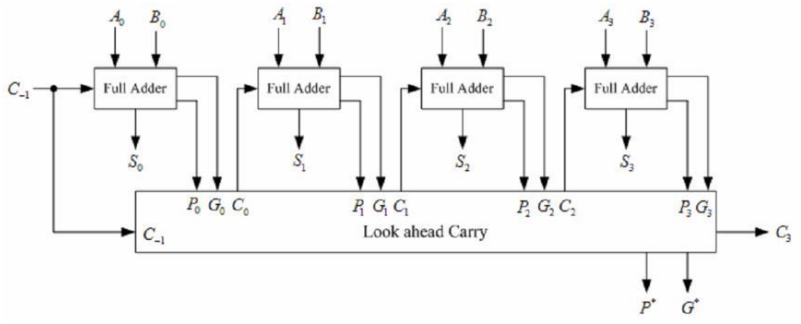


2、 4bits 逐次进位加法器

利用全加器可以构成多位二进制加法器。上图所示为四位二进制加法电路，由四个一位全加器级联而成，低一位的进位输出为高一位的进位输入。这种结构称为逐次进位加法器（Ripple Adder）。

3、 4bits 超前进位全加器

由于逐次进位加法器的进位信号是在各级之间逐次传递的，所以高位的输出必须等待低位的进位输入稳定后才有效，这就使得逐次进位加法器的延时较大，速度较慢。为了提高加法器的运算速度，要求对加法器的结构进行改进。于是就出现了超前进位加法器。下图给出了利用全加器和超前进位电路组成的四位二进制超前进位加法器的结构图。



关键代码

一位全加器

```
module FA(x,y,cin,s,co);

        input wire x,y,cin;

        output wire s,co;

        assign s=~x&&(~y&&cin||y&&~cin)||x&&(y&&cin||~y&&~cin);

        assign co=(x||y)&&cin||x&&y;

endmodule
```

4bit 加法器

```
module RippieAdder(a,b,cin,s,cout);

        input wire [3:0]a;

        input wire [3:0]b;
```

```
    input wire cin;
    output wire [3:0]s;
    output wire cout;
    wire [2:0]temp;
    FA fa0(a[0],b[0],cin,s[0],temp[0]);
    FA fa1(a[1],b[1],temp[0],s[1],temp[1]);
    FA fa2(a[2],b[2],temp[1],s[2],temp[2]);
    FA fa3(a[3],b[3],temp[2],s[3],cout);
endmodule
```

4bit 超前加法器

```
module LookAheadAdder (a,b,cin,s,cout);
    input wire [3:0]a;
    input wire [3:0]b;
    input wire cin;
    output wire [3:0]s;
    output wire cout;
    wire [3:0]c;
    wire [3:0]G;
    wire [3:0]P;
    assign G[3]=a[3]&&b[3];
    assign G[2]=a[2]&&b[2];
```

```
assign G[1]=a[1]&&b[1];

assign G[0]=a[0]&&b[0];

assign P[3]=a[3]&&~b[3]||~a[3]&&b[3];

assign P[2]=a[2]&&~b[2]||~a[2]&&b[2];

assign P[1]=a[1]&&~b[1]||~a[1]&&b[1];

assign P[0]=a[0]&&~b[0]||~a[0]&&b[0];

wire [3:0]c3;

wire [2:0]c2;

wire [1:0]c1;

and(c3[0],P[3],P[2],P[1],P[0],cin);

and(c3[1],P[3],P[2],P[1],G[0]);

and(c3[2],P[3],P[2],G[1]);

and(c3[3],P[3],G[2]);

and(c2[0],P[2],P[1],P[0],cin);

and(c2[1],P[2],P[1],G[0]);

and(c2[2],P[2],G[1]);

and(c1[0],P[1],P[0],cin);

and(c1[1],P[1],G[0]);

and(c0,P[0],cin);

or(cout,c3[0],c3[1],c3[2],c3[3],G[3]);

or(c[2],c2[0],c2[1],c2[2],G[2]);
```

```
        or(c[1],c1[0],c1[1],G[1]);

        or(c[0],c0,G[0]);

        assign s[0]=~a[0]&&(~b[0]&&cin||b[0]&&~cin)||a[0]&&(b[0]&&cin||~b[0]&&~cin);

        assign s[1]=~a[1]&&(~b[1]&&c[0]||b[1]&&~c[0])||a[1]&&(b[1]&&c[0]||~b[1]&&~c[0]);

        assign s[2]=~a[2]&&(~b[2]&&c[1]||b[2]&&~c[1])||a[2]&&(b[2]&&c[1]||~b[2]&&~c[1]);

        assign s[3]=~a[3]&&(~b[3]&&c[2]||b[3]&&~c[2])||a[3]&&(b[3]&&c[2]||~b[3]&&~c[2]);

endmodule
```

8bit 加法器

```
module EightBitLAAder(a,b,cin,s,cout);

        input wire [7:0]a;

        input wire [7:0]b;

        input wire cin;

        output wire [7:0]s;

        output wire cout;

        wire temp;

        LookAheadAdder (a[3:0],b[3:0],cin,s[3:0],temp);

        LookAheadAdder (a[7:4],b[7:4],temp,s[7:4],cout);

endmodule
```

减法器

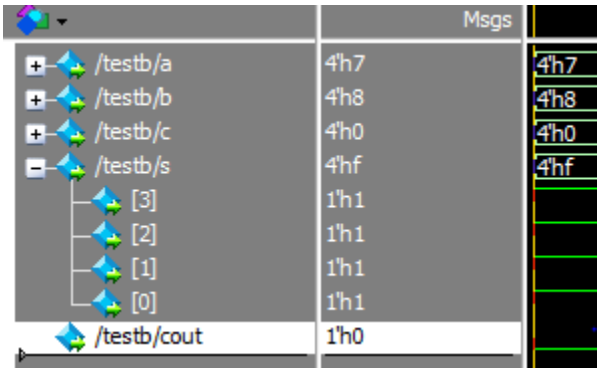
```
module Subtractor(A,B,S,sgn);

        input wire [3:0]A;
```

```
    input wire [3:0]B;
    output wire [3:0]S;
    output wire sgn;
    wire [3:0]reverse;
    wire [3:0]result;
    wire [3:0]result2;
    assign  reverse = B^(4'b1111);
    wire [3:0]temp;
    RippieAdder (A,reverse,4'b0001,result,sgn);
    assign temp=result^{4{~sgn}};
    RippieAdder (temp,4'b0000,~sgn,S, );
endmodule
```

文件清单

FA.v	一位全加器
LookAheadAdder.v	超前进位加法器
RippieAdder.v	4bit 加法器
Subtractor.v	减法器
8bitLAAdder.v	8bit 加法器
testbench.v	通用 testbench

仿真结果



加法器能正常工作。

综合情况

全加器	Total logic elements	2
	Worst-case	9.265 ns
逐次进位	Total logic elements	6
	Worst-case	10.730ns
超前进位	Total logic elements	12
	Worst-case	10.936 ns
8bit 加法器	Total logic elements	24
	Worst-case	12.598ns
减法器	Total logic elements	10
	Worst-case	11.473 ns

硬件调试情况

调试顺利，未遇到困难。