浙江大学实验报告

课程名称：电子电路实验　　指导老师：屈民军、唐奕　　成绩：_____

实验名称：常用组合电路模块的设计

一、实验目的和要求（必填）　　二、实验内容和原理（必填）

三、主要仪器设备（必填）　　四、操作方法和实验步骤

五、实验数据记录和处理　　六、实验结果与分析（必填）

七、讨论、心得

装

订

线

一、实验目的

1. 掌握用 Verilog HDL 描述数据选择器、加法器和比较器等电路模块。
2. 了解“自顶而下”的数字设计方法，掌握系统层次结构的设计。
3. 掌握模块调用的方法，掌握参数定义和参数传递的方法。
4. 掌握 ModelSim 的功能仿真的工作流程，进一步了解 Vivado 的工作流程。
5. 认识到文件管理的重要性。

二、实验内容与原理

（一）实验内容

1. 设计求两数之差的绝对值电路：

   电路输入 aIn、bIn 为 4 位无符号二进制数，电路输出 out 为两数之差的绝对值，即 out=|aIn-bIn|。

   要求用多层次结构设计电路，即调用数据选择器、加法器和比较器等基本模块来设计电路。

2. 设计模式比较器电路：

   电路的输入为两个 8 位无符号二进制数 a、b 和一个模式控制信号 m；电路的输出为 8 位无符号二进制数 y。当 m=0 时，y=MAX(a,b)　而当 m=1 时，则 y=MIN(a,b)

   要求用多层次结构设计电路，即调用数据选择器和比较器等基本模块来设计电路。

3. 设计 12s 定时器电路：

   要求：

   1） 两个 LED 指示灯。LED1 表示控制输出，LED0 表示定时结束时的报警指示灯。

   2） 两个按键：复位 reset —— 复位后两灯均处于熄灭状态，定时器处于初始状态

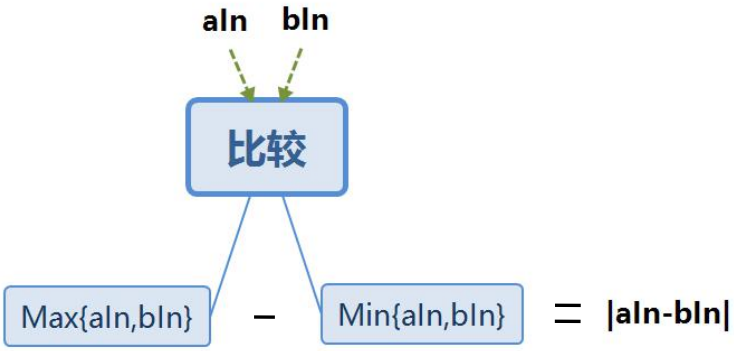
   启动 restart—— 启动后，LED1 点亮，定时开始，12s 后，LED1 熄灭，LED0 以每秒 5 次速度闪烁。

   3） 开发板输入始终为 100MHz

（二）实验原理

1. 两数之差的绝对值电路总体设计

(1) 设计思路框架：



由于减法是由加法实现，因此最后是由加法器实现 Max 与（-Min）的补码的加法：out=Max{aIn,bIn}+(~Min{aIn,bIn}+1)

(2) 原理框图：

注：comp ——数值比较器

mux_2to1 ——数据选择器（二选一）

full_adder ——全加器

该电路中

（comp）模块功能：比较 aIn、bIn 大小

（mux_2to1）功能：选出 Max{aIn,bIn} 与 Min{aIn,bIn}

（full_adder）模块：组合成四位加法器，得到最终结果 aIn、bIn 之差的绝对值，即 out=Max{aIn,bIn}+(~Min{aIn,bIn}+1)

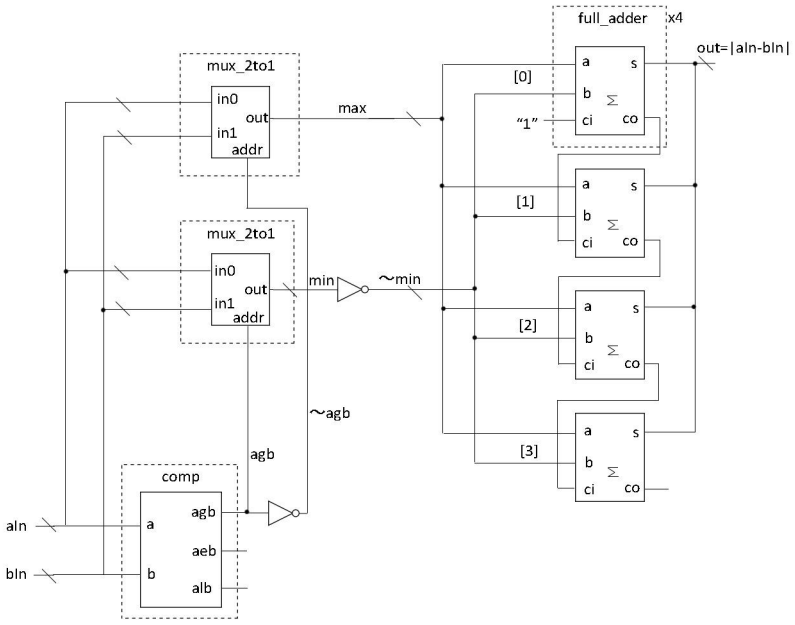


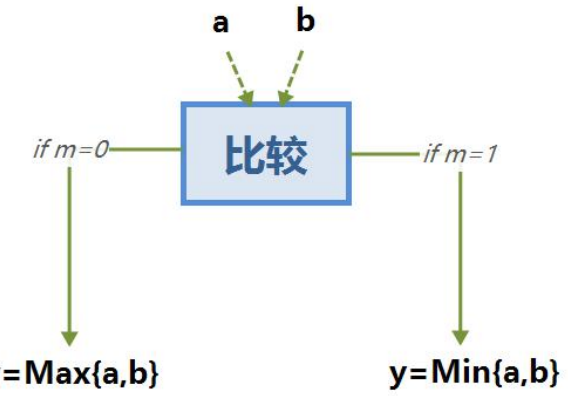
图 1 两数之差的绝对值电路

(3) 顶层模块 Verilog HDL 描述

```
module abs_dif(aIn,bIn,out);
  input[3:0] aIn,bIn;
  output[3:0] out;

    // 比较器实例comp_inst 比较aIn、bIn大小  输出agb
    wire agb;
    comp #(.n(4)) comp_inst(
        .a(aIn),
        .b(bIn),
        .agb(agb),
        .aeb(),
        .alb());

    // 两个数据选择器 mux1、 mux2
    wire[3:0] max,min;

    // 输出Max{aIn,bIn}
    mux_2to1 #(.n(4)) mux1(
        .out(max),
        .in0(aIn),
        .in1(bIn),
        .addr(~agb));

    // 输出Min{aIn,bIn}
    mux_2to1 #(.n(4)) mux2(
        .out(min),
        .in0(aIn),
        .in1(bIn),
        .addr(agb));

    // 四位加法器，实现out=max+(~min+1)
    wire[2:0] c;
    full_adder adder0(.a(max[0]),.b(~min[0]),.s(out[0]),.ci(1'b1),.co(c[0]));
    full_adder adder1(.a(max[1]),.b(~min[1]),.s(out[1]),.ci(c[0]),.co(c[1]));
    full_adder adder2(.a(max[2]),.b(~min[2]),.s(out[2]),.ci(c[1]),.co(c[2]));
    full_adder adder3(.a(max[3]),.b(~min[3]),.s(out[3]),.ci(c[2]),.co());
endmodule
```

2. 模式比较器电路总体设计

(1) 设计思路框架：



(2) 原理框图：

该电路中

（comp）模块功能：比较 a、b 大小

（Xnor gate）模块：同或门，输出 addr 控制二选一模块

（mux_2to1）功能：输出 Max{a,b} 与 Min{a,b}，实现最终功能

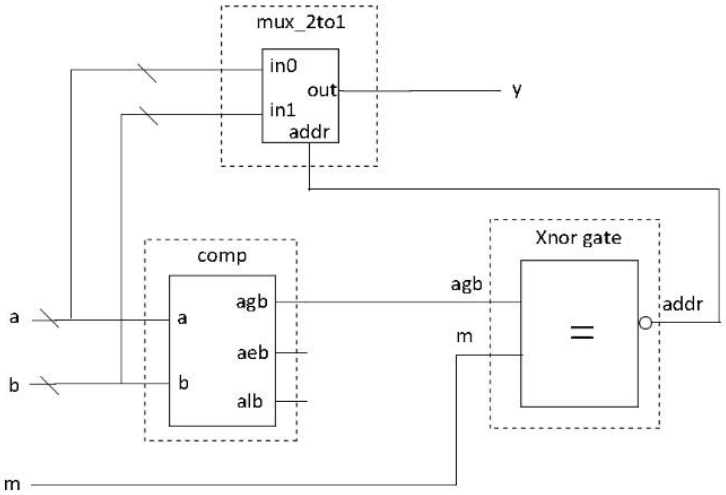


图 2 模式比较器电路

(3) 功能表

m	y
0	Max{a,b}
1	Min{a,b}

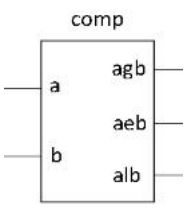
(4) 顶层模块 Verilog HDL 描述

```
module ModeComparator(a,b,m,y);
  output[7:0] y;
  input[7:0] a,b;
  input m;

    // 比较器模块，比较a、b
    wire agb;
    comp #(.n(8)) comp_inst(
        .a(a),
        .b(b),
        .agb(agb),
        .aeb(),
        .alb());
    //  同或门，输出addr
    wire addr;
    assign addr=~(m^agb);

    // addr控制数据选择器，选出目标
    mux_2to1 #(.n(8)) mux(
        .out(y),
        .in0(a),
        .in1(b),
        .addr(addr));
endmodule
```

3. **各功能模块设计**

(1) 比较器

① 原理框图



对输入值 a、b 比较大小

② 功能表

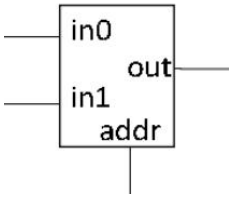
	agb	aeb	alb
a>b	1	0	0
a=b	0	1	0
a<b	0	0	1

③ 代码描述

```
module comp(a,b,agb,aeb,alb);

  // n为被比较数据的位数
  parameter n=1;
  input[n-1:0] a,b;      //被比较数据：a、b
  output agb,aeb,alb;
    assign agb=(a>b);    //  当 a>b  agb=1
    assign aeb=(a==b);   //  当 a=b  aeb=1
    assign alb=(a<b);    //  当 a<b  alb=1
endmodule
```

(2) 数据选择器

① 原理框图



对输入值 in0 和 in1 实现二选一功能

② 功能表

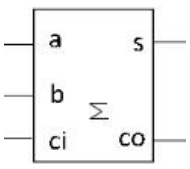
addr	out[n-1:0]
0	in0[n-1:0]
1	in1[n-1:0]

③ 代码描述

```
module comp(a,b,agb,aeb,alb);

  // n为被比较数据的位数
  parameter n=1;
  input[n-1:0] a,b;      //被比较数据：a、b
  output agb,aeb,alb;
    assign agb=(a>b);    //  当 a>b  agb=1
    assign aeb=(a==b);   //  当 a=b  aeb=1
    assign alb=(a<b);    //  当 a<b  alb=1
endmodule
```

(3) 全加器

① 原理框图



实现一位加数 a、b 的二进制加法。s—本位和，co—进位，ci—低位的进位

② 真值表

a	b	ci	s	co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

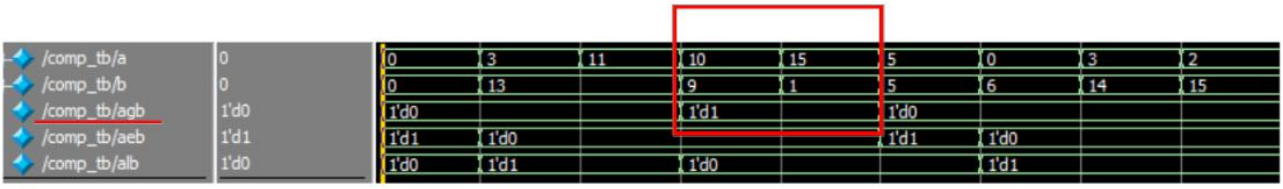
三、主要仪器设备

1. 装有 Modelsim 、vivado 软件的计算机
2. Nexys Video 开发板
3. 3.3V 的 DC 电源适配器

四、实验数据记录、处理

以下利用老师提供的测试代码分别对各个功能模块以及两个顶层模块进行仿真分析。

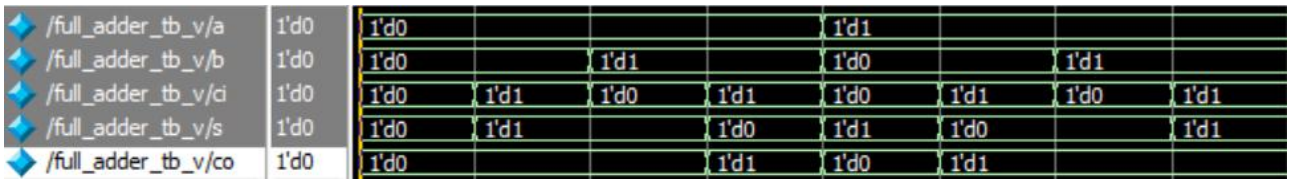
（一）比较器仿真



图中红色方框中为 a>b 的情况，可见 agb 为 1，符合最初的功能设计。其他情况 a=b,a<b 输出逻辑也正确。

（二）数据选择器

（三）全加器

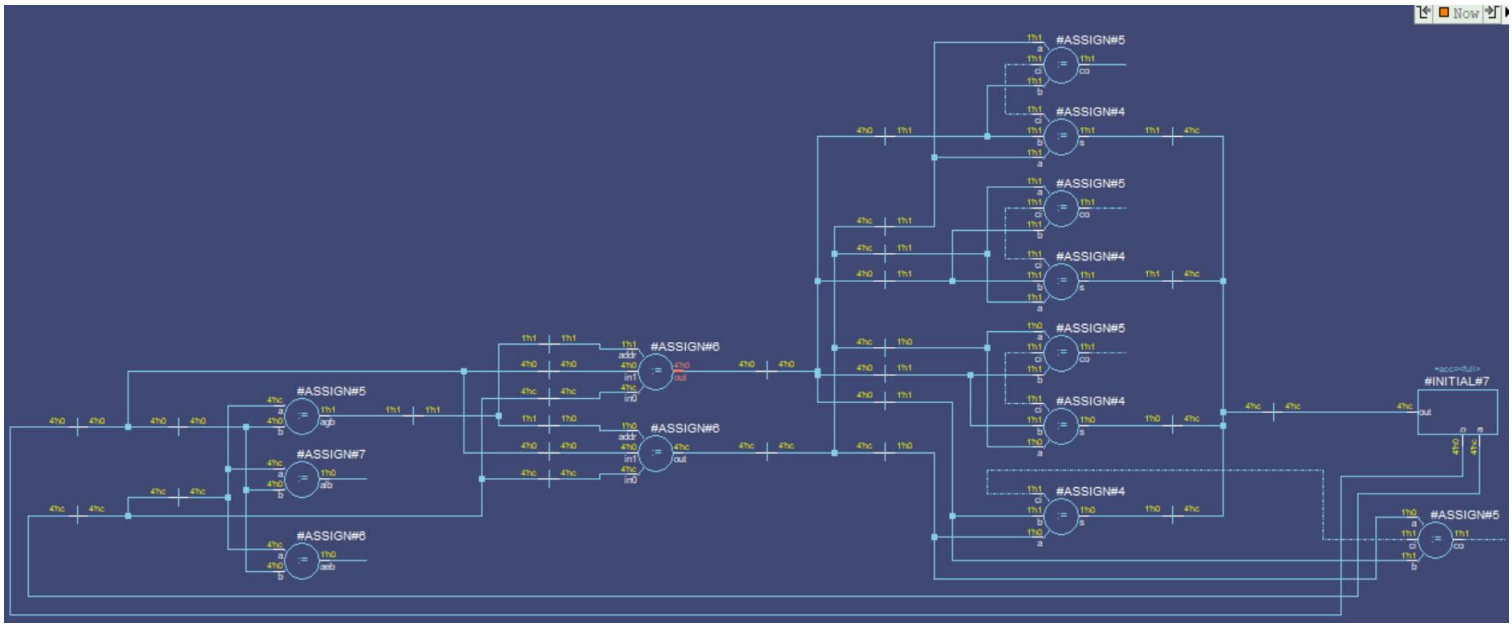


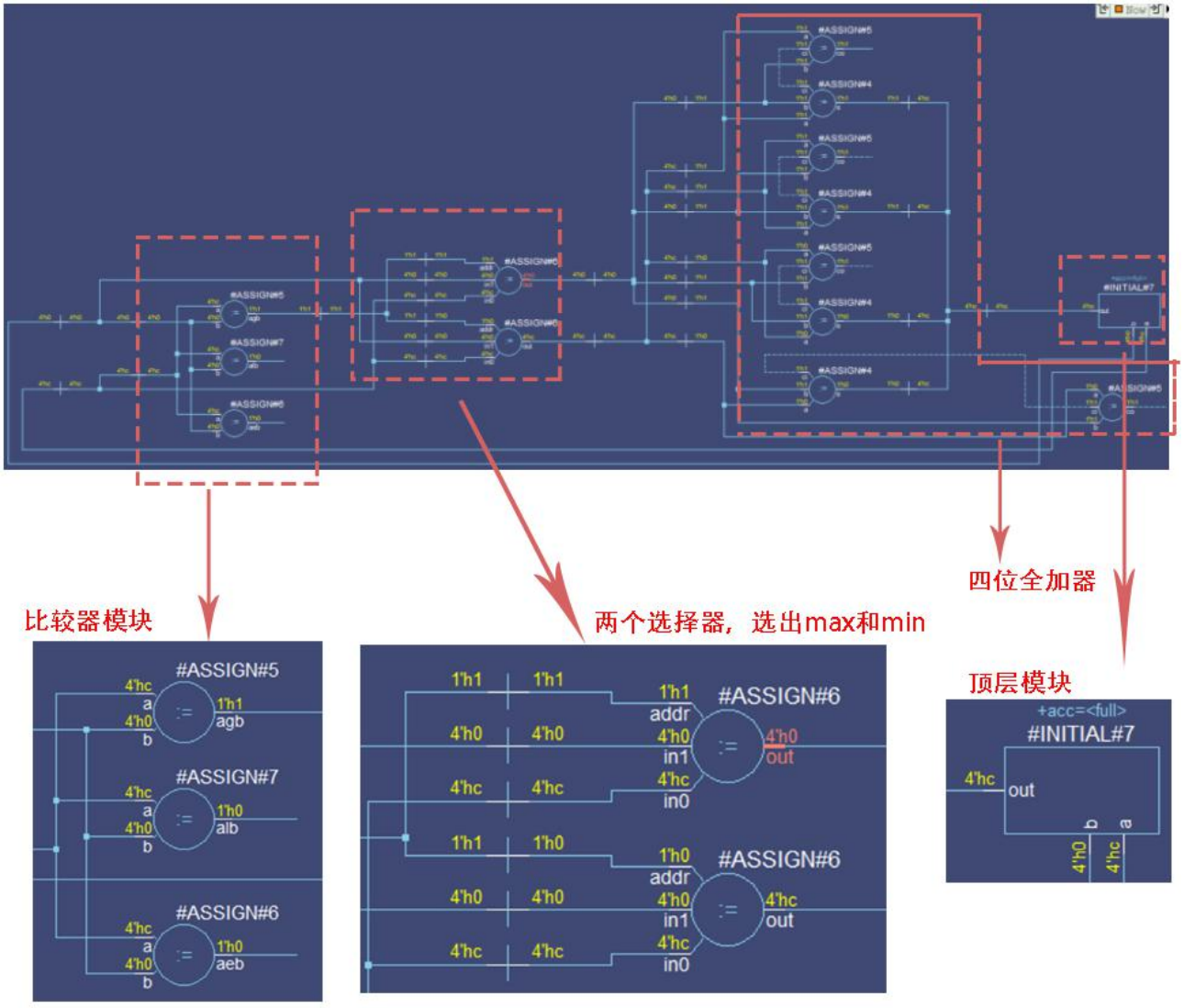
将波形中的测试数据填入下列表格，与最初设计的功能表一致，说明全加器功能无误，代码正确。

a	b	ci	s	co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

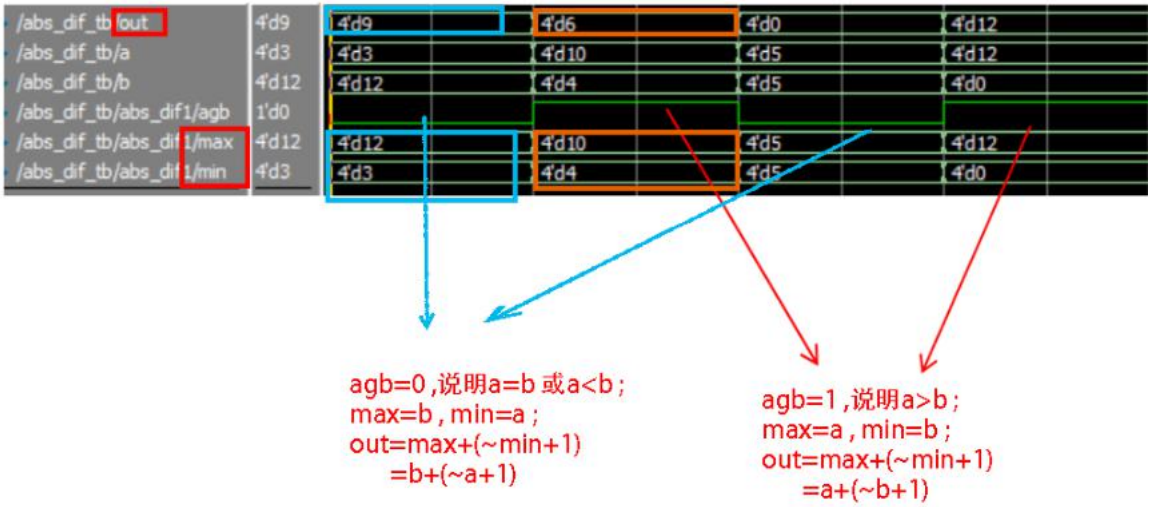
（四）两数之差的绝对值电路

1.数据流





2.波形图

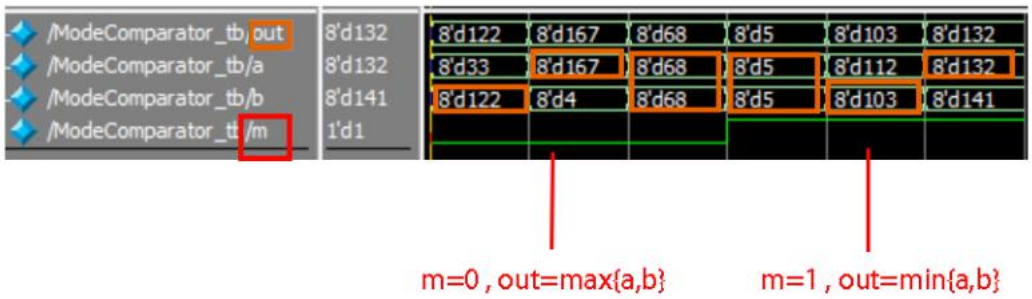


在测试代码中运用了 display ，所以可以在命令窗口看到仿真结果：

```
VSIM 37> run -all
# |  3 - 12 |=  9
# | 10 -  4 |=  6
# |  5 -  5 |=  0
# | 12 -  0 |= 12
```

计算结果正确，电路功能正确

（五）模式比较器



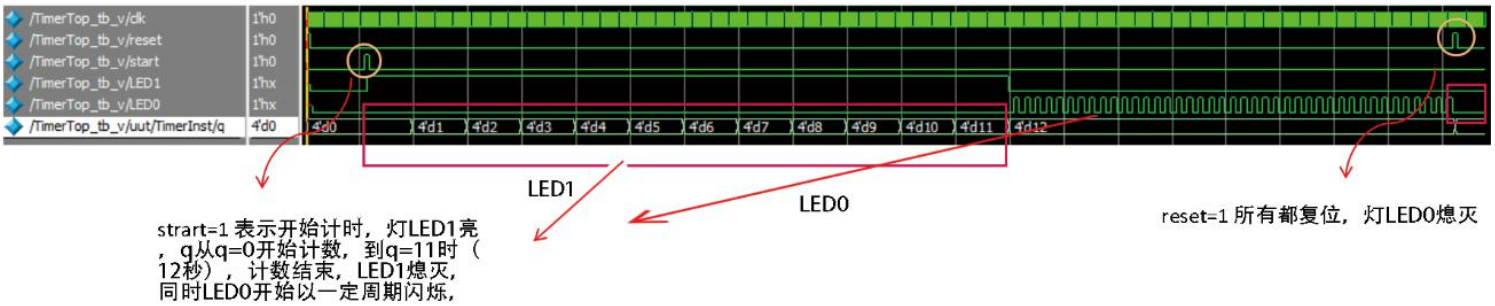
命令窗口也可直观看出仿真结果：

```
VSIM 45> run -all
# MAX(  33 , 122 )= 122
# MAX( 167 ,   4 )= 167
# MAX(  68 ,  68 )=  68
# MIN(   5 ,   5 )=   5
# MIN( 112 , 103 )= 103
# MIN( 132 , 141 )= 132
```

对比最初设计原理，仿真结果符合原理，电路功能正确。

Lab7　常用时序电路模块仿真分析

1. 12s 计时器



五、实验结果与分析

（一）对 Verilog HDL 语言的认识

1. 定义变量类型最自己来说是比较难的，在初学时很难分清 wire 和 reg 区别。理解下来就是 wire

表示直通，即只要输入有变化，输出马上无条件地反映；reg 表示一定要有触发，输出才会反映输入。直观点说，wire 对应于连续赋值，如 assign。reg 对应于过程赋值，如 always，initial。

2. 我认为 Verilog HDL 语言建立的硬件模块可分为有时钟源和无时钟源。有时钟源的意思是需要时钟信号作为操作最基本消耗单位，硬件模块才能执行。无时钟源的意思就是不需要消费时钟信号，硬件模块也可以被执行。

实验五由组合逻辑建立的硬件模块就是无时钟源的例子，其中涉及到的基本组合电路：比较器，加法器等等。

3. 其次，将预期功能从语言文字描述转化为代码描述这个建模过程也十分重要，因此也就需要自顶而下的结构设计过程，先设计好总体框架，再按照各个模块的真值表或者功能表设计小模块的电路。

（二）参数认识

1. parameter 经常用于定义延迟时间和变量位宽，可以提高程序的可读性和可维护性。

例如这次实验中，设计了比较器，数据选择器等，其中都定义了输入变量的位宽为 parameter n，在顶层模块中通过传递参数 n 可以更改变量的位宽，这样在别的顶层模块中也可以调用比较器等而不会因为变量位宽不同而重新再写一次比较器代码，从而提高工作效率。

2. 参数传递方法：

例子:

```
// 定义模块
module Eg（A，B）
parameter n=1，m=1； ……
模块定义参数时指定缺省值 如果没有上面传递下来的值就使用这个缺省值

// 顶层模块调用 Eg 模块
Eg  #（4,0） D1（...）； ……
// 其中 Eg 是子模块 module 名， #后面的是参数，进行位置对应映射，即 4 传递给 n， 0 传递给 m。
// 如果：
Eg  #（4） D1（...）； ……
// 里面的参数映射只有一个，即 4 给 n，而 m 使用缺省值 1。
// 但是为了程序可读性和规范性，应该采用名称匹配：
Eg  #（.n(4), .m(0)） D1（...）；……
```

（三）文件管理

此次实验中体会最深的是关于源代码的管理，因为 modelsim 和 vivado 使用的是同一系列源代码，如果将源文件拷贝到 vivado 文件夹里，如果 vivado 报错，并且在 vivado 中更改代码，可能更改的是 vivado 文件夹中的，源文件夹中的代码依然是错误的，用 modelsim 时依然用的是错误的文件。

（四）Modelsim

1. modelsim 编译器在对设计文件进行编译时，不会进行综合，不会与硬件对应，没有包含硬件信息，不会存在器件延时所以只是一种功能仿真。

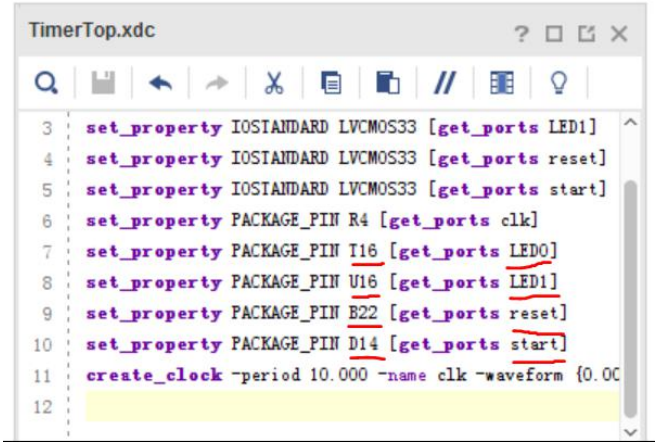
2. 在波形中如果对十六进制不熟悉，可以改成无符号位十进制，对我来说更加直观。

3. 观察变量的时候不能只把 testbench 中的变量添加进去，有时候子模块的一些变量更容易判断逻辑的正确与否，才是顶层模块中的关键之处，同时也更容易发现自己的错误在哪里。例如时序电路 12s 定时器中的 timer（计数模块）中的 q（用于计数）才是判断波形正确与否的关键。

4. 在时序电路中仿真时候，通过 testbench 中参数 sim 的传递可以改变分频比。在仿真时用小分频比的好处是便于仿真，可以减少仿真时间。如果依旧用实际电路中的大分频比，频率太高，仿真时可能看不见波形。

（五）Vivado

1. 在 Lab7 引脚约束的时候，把 LED 灯和 reset、restart 按键的引脚弄反了，把程序写进板子后，怎么按按键都没有反应（已经确定程序是正确的），就很疑惑，后来点开 XDC 文件，一眼就发现自己的引脚约束好像和书上的约束不同，于是直接改代码改正过来并且重新保存 XDC 文件。下图是更改后的文件



因此将 XDC 保存到源文件夹 src 下，便于检查和更改，方便文件管理，有效地提高了效率。

2. 除了通过代码检查错误外还可以通过综合结果查看电路逻辑，易于发现错误

六、 讨论、心得

思考题：

1. 有符号数（补码）的两数之差的绝对值电路

(1) 设计思路

本电路依旧选用上述的选择器和比较器（无符号数的比较），因此在比较和选择前需要对有符号数的 a、b 进行处理。

考虑到计算机存储有符号数的方式，初步设计思路如下表格。

	max	min	out
a、b 同正负	Max{a,b}	Min{a,b}	max+(~min+1)
a、b 一正一负	Min{a,b}	Max{a,b}	

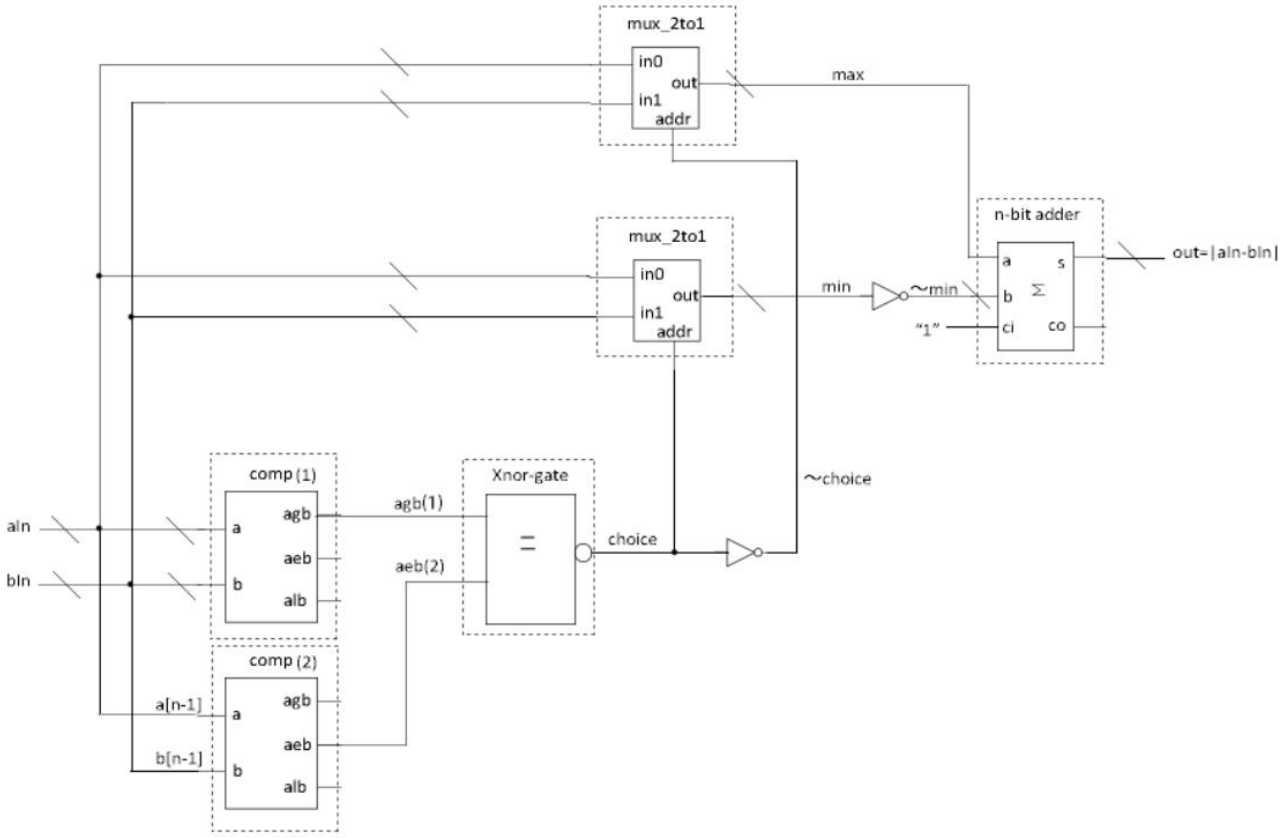
举例 （a、b 为有符号数）：

a、b 同正负： 1）同正： a = 0 0011 b=0 0001 则最大值 max 为 a

2）同负： a = 1 1111 （-1） b=1 1110 （-2） 最大值 max 为 a 即 Max{a,b}

a、b 一正一负： a = 1 1111 （-1） b=0 0001 （+1） 最大值 max 为 b 即 Min{a,b}

(2) 原理框图



(3) 各个模块功能

① comp(1)

比较 a、b 大小（把 a、b 当作无符号数比较）

② comp(2)

比较 a、b 的符号位 即 a[n-1]、b[n-1]的大小

③ Xnor-gate

输出 choice 控制选择器，选出 a、b 大小（分正负）

功能表

aeb(2)	agb(1)	choice	备注
0	0	1	a、b 一正一负，max 为 a
0	1	0	a、b 一正一负，max 为 b
1	0	0	a、b 同正负，max 为 b
1	1	1	a、b 同正负，max 为 a

④ 两个 mux_2to1

输出 max、min

⑤ n-bit adder

实现最终功能 out=|aIn-bIn| =max+(~min+1)

2. 用加法器实现比较器功能

aIn、bIn 为有符号数

为防止溢出，将 aIn、bIn 扩展一位从 aIn[n-1]、bIn[n-1] 到 aIn[n]、bIn[n]。即 aIn[n-1]、bIn[n-1]才是符号位

先用全加器实现：s=aIn-bIn；
然后 assign 描述：
big=(~s[n])&&(|s)
equal=~|s
small=~s[n]

aIn>bIn	s 为负	big=1
aIn=bIn	s 为 0	equal=1
aIn<bIn	s 为正	small=1

表 1 功能表

3. 参数传递方法见“五、实验结果与分析”。