数字系统设计实验报告

姓名：卢泽熙　学号：3220102478　实验名称：常用组合电路模块设计和应用　指导老师：屈民军、唐奕

装 订 线

目录

一、实验目的

本实验作为数字系统设计的基本实验，重点在于了解系统层次结构设计方法，学习模块调用方法、参数定义和参数传递方法，同时掌握各种经典数字电路模块的设计过程。具体的实验目的包括以下几点：

（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、实验任务

整个实验包含以下两个子实验：

实验一任务要求

设计**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。

实验二任务要求

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**、**比较器**等基本模块来设计电路。

三、实验原理与设计思路

实验一

1. 总体电路设计

数字电路无法直接进行减法运算，而是利用补码的性质进行加法运算，因此该电路输出计算公式如下：

$$out = Max(aIn, bIn) + (\sim Min(aIn, bIn) + 1)$$

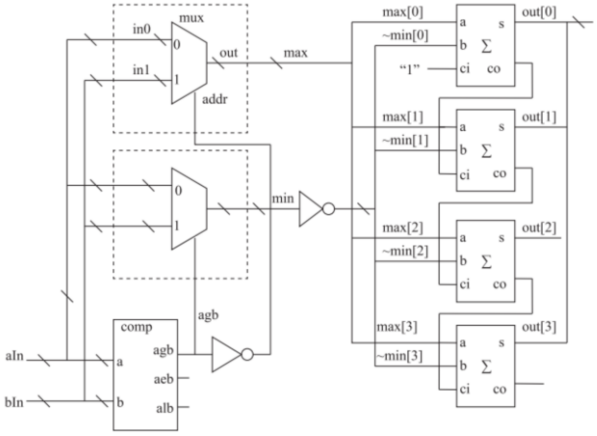
根据计算公式可以得到该数字电路的总体设计思路：

（1）首先利用数值比较器实现比较输入两数的大小，比较结果 agb 用于控制两个数据选择器；

（2）数据选择器分别选出输入两数的最大值 Max 和最小值 Min；

（3）最后由全加器组成的四位加法器实现 Max 与 -Min 的补码加法。

该电路的原理框图如下：



注意到数据选择器选出的 Min 连接反相器实现取补码操作；第一位全加器初始进位输入需置“1”，与公式吻合。

2. 模块的 Verilog HDL 描述

根据分析得到的原理框图，给出该电路的顶层设计 Verilog HDL 代码

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

  wire agb;              // Wire to hold the result of 'a greater than b' comparison
  // Instantiate the comparator module to compare aIn and bIn
  comp #(.N(4)) comp_inst(.a(aIn), .b(bIn), .agb(agb), .aeb(), .alb());

  wire [3:0] max, min;   // Wires to hold the maximum and minimum of aIn and bIn
  // Two multiplexers to determine 'max' and 'min' based on 'agb'
  mux_2to1 #(.N(4)) mux1(.out(max), .in0(aIn), .in1(bIn), .addr(~agb));
  mux_2to1 #(.N(4)) mux2(.out(min), .in0(aIn), .in1(bIn), .addr(agb));

  wire [2:0] c;          // Carry bits for the full adders
    // Four full adders to perform the subtraction (binary two's complement addition) to find the absolute difference
  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
```

根据顶层设计 Verilog HDL 描述，按照自顶而下的数字设计方法，可以确定各子模块的端口命名、参数定义等信息，从而完成各子模块的 Verilog HDL 设计，这里不再赘述。

实验二

1. 总体电路设计

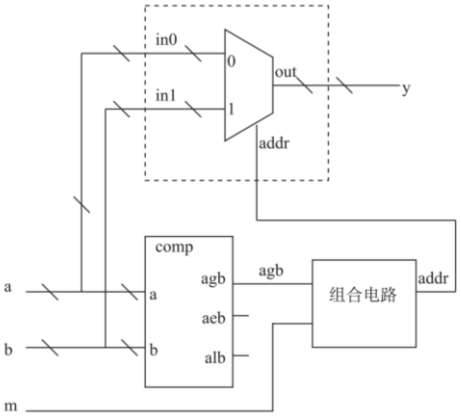
与实验 1 设计思路类似，利用比较器、数据选择器可以得到两个输入数据的最值；而模式控制信号 m 决定输出是最大值或是最小值。据此得到模式比较器的总体设计思路：

（1）首先利用数值比较器实现比较输入两数的大小，比较结果为 agb；

（2）agb 与模式控制信号 m 经过组合电路处理，输出 addr 作为数据选择器的地址传入数据；

（3）数据选择器根据传入的地址信号决定输出。

该电路的原理框图如下：



根据总体设计思路，分析产生 addr 组合电路需实现的功能，如下列表格所示：

m	*agb*	*addr*	*备注*
0	0	1	当 m=0，且 a≤b 时，y 取最大值 b
0	1	0	当 m=0，且 a＞b 时，y 取最大值 b
1	0	0	当 m=1，且 a≤b 时，y 取最大值 b
1	1	1	当 m=1，且 a＞b 时，y 取最大值 b

从表中可以看出，addr 产生电路实现的功能就是一个同或门。

2. 模块的 Verilog HDL 描述

根据分析得到的原理框图，给出该电路的顶层设计 Verilog HDL 代码：

```
module ModeComparator(a, b, m, y);
    parameter n = 8;
    input[n-1:0] a, b;
    input m;
    output[n-1:0] y;

    wire agb;                       // Wire to hold the result of 'a greater than b' comparison
    // Instantiate the comparator module to compare 'a' and 'b'
    comp #(.n(8)) comp_inst(.a(a), .b(b), .agb(agb), .aeb(), .alb());

    wire addr;                      // Wire to determine the mux address based on mode 'm'
    // Determine the address for mux; XOR 'agb' with mode 'm'
    assign addr = ~agb^m;

    wire[n-1:0] max, min;             // Wires to hold the maximum and minimum values of 'a' and 'b'
    // Instantiate a 2-to-1 mux to select the comparison output based on 'addr'
    mux_2to1 #(.n(8)) mux(.out(y), .in0(a), .in1(b), .addr(addr));

endmodule
```

各子模块在实验一中已经完成设计，可直接调用。

四、主要仪器设备

该实验需要用到以下实验设备：

（1）装有 Vivado、ModelSim SE 软件的计算机。

（2）Nexys Video 开发板或 Basys3 开发板。

五、实验步骤和操作方法

实验一 两数之差绝对值电路的设计

1. 编写一位全加器的 Verilog HDL 代码，并用 ModelSim 软件进行功能仿真。

具体实现过程如下：

（1）在编辑器中编写 Verilog HDL 源代码与 testbench 测试文件，保存在正确路径的 src 文件夹中，确保文件管理的规范性；

（2）在 ModelSim 中建立工程，保存在正确路径的 sim 文件夹中，并添加（1）中 src 文件夹里的 .v 文件，编译；

（3）在 Library 标签下的 work 库中，选中待测试的 full_adder_tb 文件进行装载，后添加波形执行仿真。

步骤 2.3.4 具体操作方法与 1 类似，不再赘述

2. 编写 n 位二选一数据选择器的 Verilog HDL 代码及其测试代码，并用 ModelSim 软件进行功能仿真。
3. 编写 n 位比较器的 Verilog HDL 代码，并用 ModelSim 软件进行功能仿真。
4. 对两数之差的绝对值电路进行功能仿真。
5. 建立 Vivado 工程，对工程进行综合、引脚约束、实现，并下载到开发实验板中对设计进行验证.

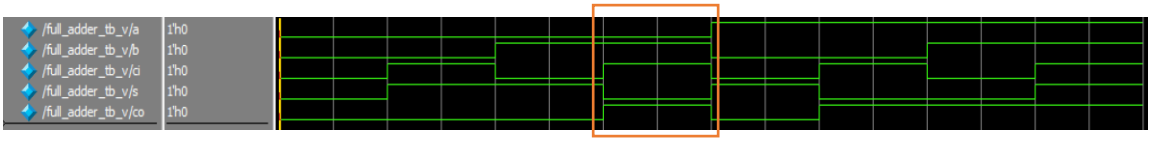
实验二 模式比较器

编写模式比较器顶层模块的 Verilog HDL 代码与 testbench 文件，并调用实验一中已经成功实现的各子模块，通过 ModelSim 软件进行功能仿真。**仿真具体步骤与实验一类似，不再赘述。**

六、实验结果记录及分析

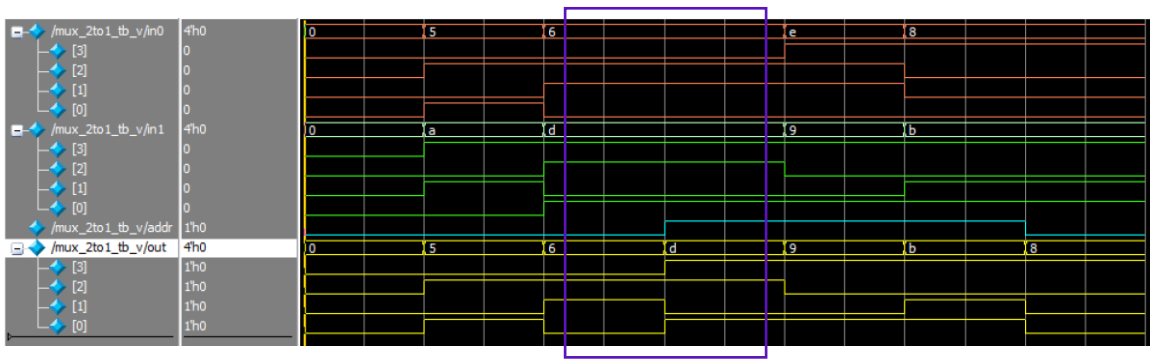
实验一 两数之差绝对值电路的设计

1. 全加器仿真结果



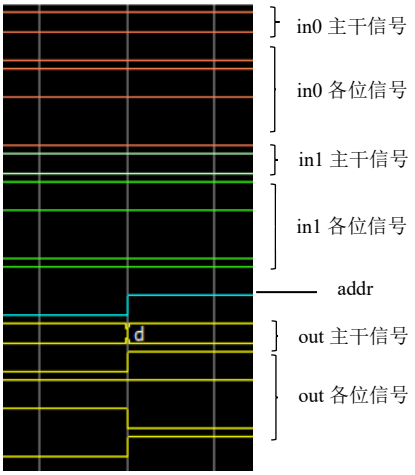
全加器仿真结果如上图所示，其中五个信号波形从上到下依次对应输入信号 a、输入信号 b、输入进位 ci、输出本位 s 和输出进位 co；同一信号仿真波形高位代表 1，低位代表 0。以矩形框选的信号为例，输入为 0、1，进位为 1，输出为 10，可以看到符合设计要求。其它时钟周期的分析方法类似，可以验证该全加器的设计是正确的。

2. 数据选择器仿真结果



数据选择器仿真结果如上图所示，因为信号较多，因此设置不同信号对应不同颜色便于区别分析。红线、绿线对应两个输入信号；蓝线 addr 代表地址线，用于数据选择；黄线代表输出信号 out。若 addr=1，out=in1；addr=0，out=in0。

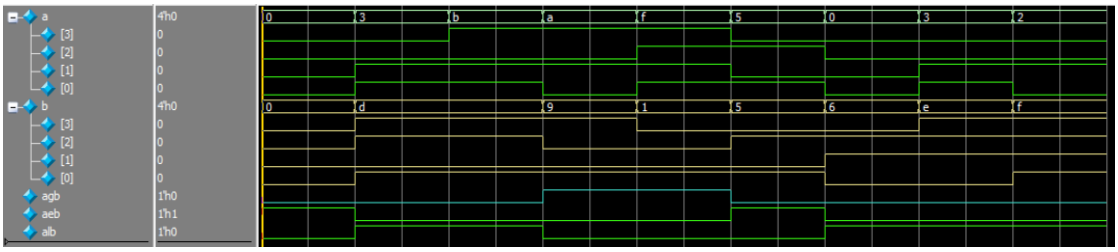
以矩形框选的信号为例，下图为矩形框选部分放大图：



前半部分 addr=0（**蓝线**），因此输出信号 out（**黄线**）波形与 in0（**红线**）一致；后半部分 addr=1，因此输

出信号 out 波形与 in1（**绿线**）一致。同理，可验证其它时钟周期仿真结果也符合预期，数据选择器设计正确。

3. 比较器仿真结果



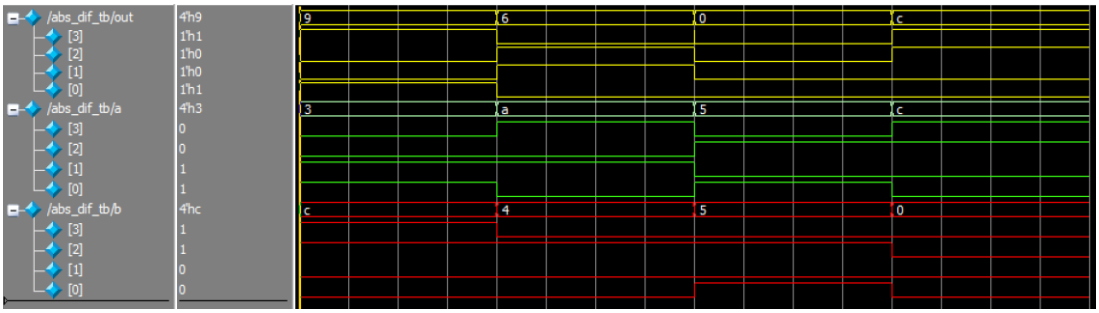
比较器仿真结果如上图所示。最上方的绿线代表输入信号 a，黄线代表输入信号 b，蓝线代表输出信号 agb；当 a>b 时 agb 为高位，aeb、alb 为低位；当 a<b 时 alb 为高位，agb、aeb 为低位；当 a=b 时 aeb 为高位，alb、agb 为低位。

根据仿真结果可知：

a	0	3	b	a	f	5	0	3	2
b	0	d	d	9	1	5	6	e	f
agb	0	0	0	1	1	0	0	0	0
aeb	1	0	0	0	0	1	0	0	0
alb	0	1	1	0	0	0	1	1	1

该仿真结果符合预期，比较器的设计是正确的。

4. 顶层模块实验结果

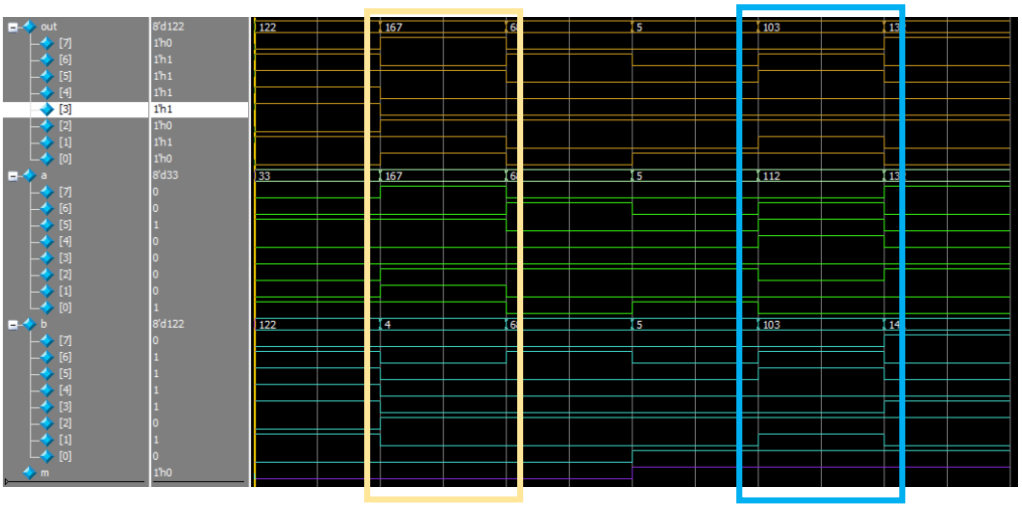


两数之差绝对值电路的顶层模块仿真波形如上图所示。输出信号 out 的波形为黄色，输入信号 a、b 的波形分别为绿色与红色。testbench 设置了四组数据进行验证，如第一组输入 3、c，out＝9＝12 - 3 符合预期；其他几组数据同样可验证是正确的。下图为 Transcript 窗口的仿真输出结果，该结果与波形、理论预期一致。

```
VSIM 2> run -all
# |  3 - 12 |=  9
# | 10 -  4 |=  6
# |  5 -  5 |=  0
# | 12 -  0 |= 12
# ** Note: $stop    : D:/StudyRepository/SystemDesignLab/CombinatinalCircuit/src/top1/abs_dif_tb.v(18)
#    Time: 80 ns  Iteration: 0  Instance: /abs_dif_tb
```

该实验有开发实验板上的具体操作验证部分，**此部分内容在“演示视频”中详细展示，实验报告里不再赘述。**

实验二 模式比较器



模式比较器电路的顶层模块仿真波形如上图所示。输出信号 out 的波形为黄色，输入信号 a、b 的波形分别为绿色与蓝色，模式控制信号 m 的波形为紫色。以矩形框选的信号为例。左侧矩形框内，输入信号分别为 167（10100111）与 4（00000100），模式控制信号 m=0，因此输出两数最大值，out=167；右侧矩形框内，输入信号分别为 103（01100111）与 112（01110000），模式控制信号 m=1，因此输出两数最小值，out=103；其他几组数据同样可验证是正确的。下图为 Transcript 窗口的仿真输出结果，该结果与波形、理论预期一致，至此 “模式比较器电路”设计完成。

```
VSIM 4> 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
# ** Note: $stop    : D:/StudyRepository/SystemDesignLab/CombinatinalCircuit/src/top2/ModeComparator_tb.v(27)
#    Time: 120 ps  Iteration: 0  Instance: /ModeComparator_tb
```

七、思考题

思考题 1

假设输入为两个 4 位有符号数的补码，其中最高位为符号位，则分以下 3 类情况讨论：

1. 输入两个数均为正数，则该情况与实验一电路输入情况相同。

   如 0010(2)与 0111(7)，经过实验一设计电路处理后输出 0101(5)，即为最终正确结果。

2. 输入两个数均为负数，则该情况与实验一电路输入情况相同。

   如 1010(-6)与 1100(-4),经过实验一设计电路处理后输出 0010(2)，即为最终正确结果。

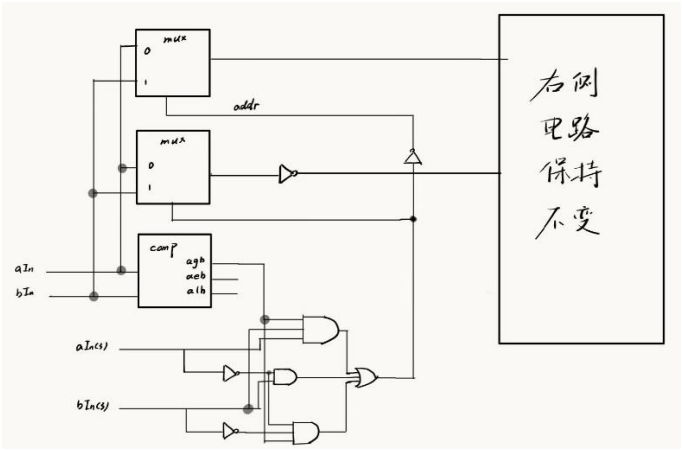
3. 设输入正数 a 与负数 b，则 out=a-b=a+(-b)；将负数补码取反加一即可得到(-b)，因此将 a 与~b 作为 4bit 加法器

的输入，同时第一位全加器输入 ci 端置 1。如 1010(-6)与 0111(7)，加法器输入端取 0101(5)、0111，可得 1101。

根据讨论结果可对电路进行改进：改进后的电路在情况 1、2 下对最小输入取反，在情况 3 下对最大位取反，则 addr’= 1 对应 3 种情况：

- aIn(3) = bIn(3) = 0，agb = 1
- aIn(3) = bIn(3) = 1，agb = 1
- aIn(3) = 0，bIn(3) = 1

因此可对电路进行如下改进：

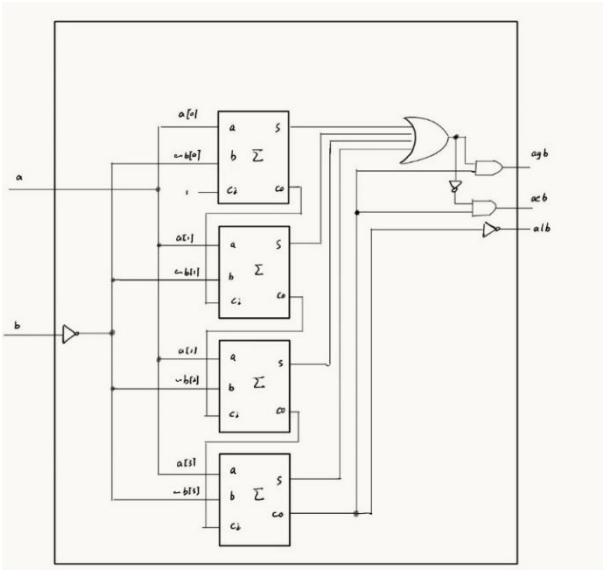


思考题 2

通过调用加法器模块计算两数之差，计算 a - b，即 a+(~b)+1，此方法产生的进位 co 及输出值 s 来比较两数大小。

- 若 a > b，则 co = 1，s 各位不全为 0；
- 若 a = b，则 co = 1，s 各位为 0；
- 若 a < b 则 co = 0。

因此电路可采用如下设计：



思考题 3

参数传递有两种格式：

方法 1：在例化过程中通过 #(A) 进行参数的传递，格式为

模块名 #(参数 1 值，参数 2 值，…) 例化模块名(端口列表)；

方法 2：使用在线显式重载参数方式，格式为

模块名 #(. 参数 1(参数 1 值)，. 参数 2(参数 2 值)，…) 例化模块名(端口列表)；

模块例化也有两种方式：

1. **信号名关联**：这种方法将需要例化的模块端口与外部信号按照其名字进行连接，端口顺序随意，可以与引用 module 的声明端口顺序不一致，只要保证端口名字与外部信号匹配即可，比如

   mod_a instance1 (.out(wc), .in1(wa), .in2(wb))；

   三个参数顺序可以调换。
2. **位置关联**：这种方法将需要例化的模块端口按照模块声明时端口的顺序与外部信号进行连接，位置要严格保持一致，如

   mod_a instance2 (wa, wb, wc)；

   三个参数顺序不可以调换。

八、实验心得与总结

Lab5 对于我来说是一个比较大的挑战。首先，我对于 Verilog HDL 语言的掌握非常有限，即使已经设计出原理框图，一开始在编写代码的过程中也常常不能正确的把它转化为对应的 Verilog 代码。比如在设计二选一数据选择器的过程中，输出 out 应该设置为 reg [n-1:0]，但一开始我因为对语法不熟悉，声明类型写为了 reg out，导致仿真波形一直有问题，后来在老师帮助下才解决。其次，我对于 ModelSim 和 Vivado 这两个软件并不熟悉，在建立工程、添加文件、执行综合等过程中要时不时翻阅之前的实验指南，导致消耗了很多时间；同时在使用这两个软件的过程中我还遇到了很多小问题，比如引脚约束没做好，文件管理不规范等……

遇到问题时，我学会了不断查阅资料和参考书籍，同时也在网络论坛和社区中寻求帮助。在这个过程中，我不仅学习了新的技能，更重要的是，我培养了独立解决问题的能力和坚持不懈的精神。随着实验进行，在代码编写方面我感觉逐渐轻松。实际上 verilog 语言和 c 语言很类似，而且逻辑非常清晰，在熟练掌握一些基本语法后上手很快，所以实验二的顶层模块的设计、仿真工作我做的比预期迅速很多。在实验后期我最大的困难就是两个思考题部分。因为对于二进制反码、补码的掌握还不是很输入，因此完成这两个思考题花费了许多时间。

总之，这次实验虽然充满挑战，但最终却成为了一段难忘的学习旅程。我了解了系统层次结构设计方法，学习了模块调用方法、参数定义和参数传递方法，同时学习了各种经典数字电路模块的设计过程——更重要的是我获得了自我探索和实践的珍贵体验，为我的未来学习和职业发展奠定了更坚实的基础。