

实验六 EDA 作业二

自 35 夏弘宇 2023011004

一、实验目的

1. 熟练掌握面向 FPGA 的简单数字系统的设计流程。
2. 学习编写测试文件对设计电路进行仿真验证。
3. 熟悉实验装置——实验板，掌握板上外设的工作原理。

二、预习任务

1. 阅读网络学堂中的“FPGA 实验板说明书”了解实验板上的外设资源，并掌握其工作原理。

(1) 拨码开关：当拨码开关设置为“1”时，开关断开，输出高电平；反之，当拨码开关设置为“0”时，输出低电平。

(2) 发光二极管 LED：高电平有效（LED 亮）

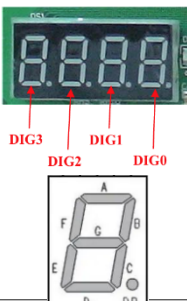
(3) 4 位扫描显示数码管：共 12 根数据线，分为 8 根段选线、4 根位选线，由段选线上信号选择哪一根线段（管）亮，由位选线选择哪一位亮。

当段选信号固定时，数码管上显示的字符内容是固定的。给定特定的位选信号后，对应的数码管会显示出相应的字符。如果按时间轮流控制各个数码管的位选端，数码管将会依次显示字符。在轮流显示的过程中，每个数码管的点亮时间一般为 1 到 2 毫秒。由于人眼的视觉暂留现象和发光二极管的余辉效应，尽管各个数码管并不同时点亮，只要扫描速度足够快，肉眼就能看到一组稳定的显示数据。

(4) 晶振：提供频率为 50MHz 的高低电平信号。

器件名称	编号	引脚号	备注	实物图
拨码开关	DIP1	PIN_12	L ↑ ↓ H	
	DIP2	PIN_11		
	DIP3	PIN_10		
	DIP4	PIN_8		
	DIP5	PIN_6		
	DIP6	PIN_5		
	DIP7	PIN_4		
	DIP8	PIN_3		

器件名称	编号	引脚号	备注	实物图
LED	D7	PIN_56	高电平有效	
	D6	PIN_57		
	D5	PIN_58		
	D4	PIN_59		
	D3	PIN_60		
	D2	PIN_61		
	D1	PIN_63		
	D0	PIN_64		

数码管	DIG3	PIN_39	高电平有效		
	DIG2	PIN_37			
	DIG1	PIN_36			
	DIG0	PIN_35			
	DP	PIN_46	高电平有效		
	G	PIN_43			
	F	PIN_41			
	E	PIN_48			
	D	PIN_47			
	C	PIN_45			
	B	PIN_40			
	A	PIN_44			

晶振	XTAL_1	PIN_132	50MHz	
	XTAL_2	PIN_131	50MHz	

2. 根据实验任务中的步骤提示，写出要用到的电路模块及其功能。

- (1) 二选一数据选择器：
- (2) 四二选一数据选择器：
- (3) 4 位二进制补码运算器：输入负数时，将输入的原码转为补码，将输出的补码转位原码。
- (4) 1 位二进制全加器：能处理进位的 1 位二进制全加器。
- (5) 4 位二进制全加器：能处理进位的 4 位二进制全加器。
- (6) 位置选择器（位选器）：选择哪一位显示
- (7) 显示译码器（段选器）：将三位 2 进制信号转变为显示的数值
- (8) 分频器：根据需要进行选择倍数，考虑 5、10、16 倍，并最终集成为 200k。

3. 进入实验室之前，可预先完成电路的设计输入。

三、实验任务（含选做）

在可编程逻辑器件上设计一个运算电路，可以实现 $S=M+N$ 。M 和 N 为 3 位二进制数，其中 1 位是符号位，2 位是有效数字。要求用原理图的输入方式完成。

用实验板上的拨码开关模拟运算数（原码输入），用发光二极管表示运算数的正负标志；用数码管显示运算数、运算结果（原码）及运算结果的正负标志。

具体内容及步骤如下：

1. 用门电路设计一个 1 位二进制全加器，将其封装成 1 位全加器模块。
2. 以 1 中已封装的 1 位全加器模块为基础实现一个 4 位二进制全加器，并仿真检查功能正确与否（仿真工具不限）。
3. 以 2 中的 4 位全加器模块为基础实现一个二进制运算器，可以完成运算 $S=M+N$ 。编写测试文件，使用 Modelsim 仿真验证运算器的功能。
4. 设计一个数码管的动态扫描显示电路，可以在 4 位数码管上同时显示 4 个数字。电路的输入方式不限（即可用原理图或硬件描述语言方式）。

具体内容及步骤：

- (1) 设计一个分频器，建议将系统时钟由 50MHz 分频至 250Hz。
- (2) 设计一个电路，使运算器的两个运算数和运算结果同时显示在 4 位数码管上。

DIP1、DIP2	数码管 3 (DIG3)	数码管 2 (DIG2)	数码管 1 (DIG1)	数码管 0 (DIG0)
00	M	不亮	不亮	不亮
01	不亮	N	不亮	不亮
10	不亮	不亮	S（正负标志）	不亮
11	不亮	不亮	不亮	S（运算结果）

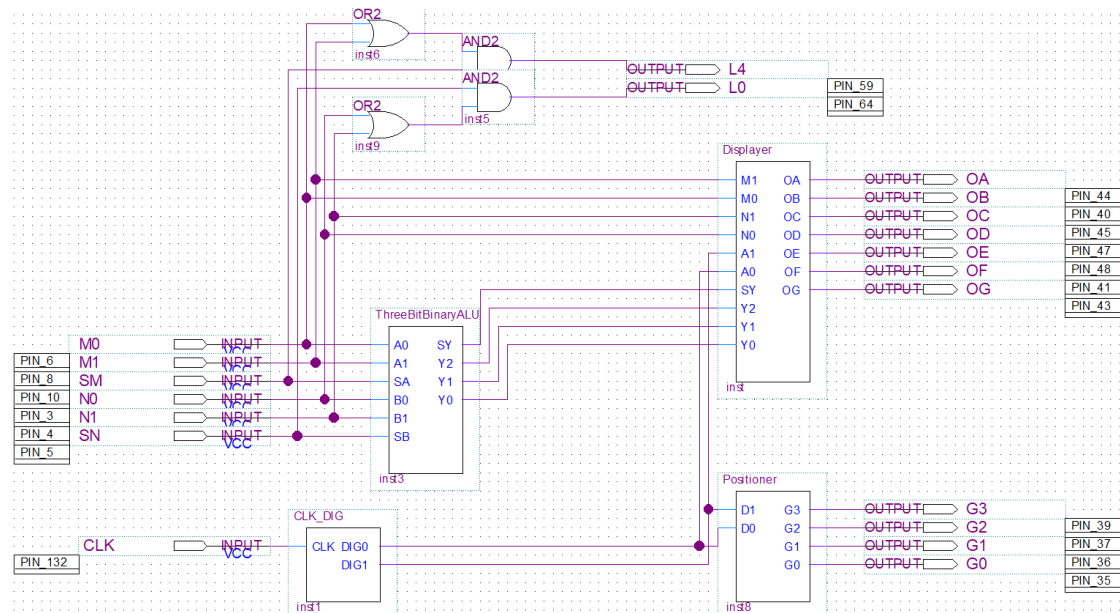
5. 下载到实验板上验证功能。

六、实验报告-实验总结

1. 阐述设计思路。

本实验涉及相对复杂逻辑功能的实现，我在设计过程采用模块化的思路。首先来看需要哪些模块：本次实验的核心是运算和显示，对于运算，需要一个**三位二进制运算器**，由于涉及符号，运算器除了需要一个**四位全加器**，还需要**补码运算器**，四位全加器由四个**一位全加器**组成，一位全加器属于简单逻辑，容易实现；对于显示，二极管发光的控制本身就很简单，不必再使用模块，而数码管的控制需要位选与段选两部分，**段选模块**有现成的**7448 模块**来负责，但具体输出什么数字与位选信号相关，需要**数据选择器**，由于数字范围 0-6，考虑**4 位二选一数据选择器**，由 4 个**一位二选一数据选择器**构成，而位选模块本质上是一个 2 线-4 线译码器，直接实现即可；还有就是位选信号的产生，涉及时钟信号的分频。

2. 顶层电路图，并说明其中各模块电路的功能。



1 三位二进制带符号加法器 ThreeBitBinaryALU

输入：SA 是 A 的符号，A1 和 A0 分别为 A 的第 1 位和第 0 位；B 同理。

输出：计算 $A+B$ 得到 Y，SY 为符号，正为 0 负为 1；Y2，Y1 和 Y0 分别为 Y 的第 2, 1, 0 位。

1-1 无输入进位符号的四位全加器 FourBitFullAdder

输入：四位无符号二进制数 $A=A_3A_2A_1A_0$ ， $B=B_3B_2B_1B_0$ 。

输出：四位无符号二进制数 $S=S_3S_2S_1S_0$ 以及进位信号 C0。

1-1-1 一位全加器

输入：一位无符号二进制数 A, B，进位信号 C1

输出：一位无符号二进制数 S，进位信号 C0

1-2 补码运算器 Complementor

输入：四位有符号二进制数，S 为符号 1 为负 0 为正，D=D2D1D0 为绝对值。

输出：S 为 0 时，直接输出 Y3=S, Y2Y1Y0=D2D1D0; S 为 1 时，输出补码 Y3Y2Y1Y0。

2 显示模块（段选）Displayer

输入：M1M0 为|M|，N1N0 为|N|，SY 为 Y 的符号正为 0 负为 1，Y2Y1Y0 为|Y|，A1A0 为“位选”信号（实际上应该叫数据选择信号）（00 选|M|，01 选|N|，10 选 SY，11 选|Y|）。

输出：七位数码管所需的段选信号

2-1 4 位二选一数据选择器 Mux4_21

输入：A3A2A1A0，B3B2B1B0，选择信号 SEL

输出：SEL 为 1 时 Y3Y2Y1Y0 输出 A3A2A1A0，SEL 为 0 时输出 B3B2B1B0。

2-1-1 1 位二选一数据选择器 Mux4_21

输入：A，B，选择信号 SEL

输出：SEL 为 1 时 Y 输出 A，SEL 为 0 时输出 B。

3 位选模块 Positioner：控制哪个数码管显示：00 数码管 3，01 数码管 2，10 数码管 1，11 数码管 0。

本质上是 2 线-4 线译码器。D1D0 输入 00，G3G2G1G0 输出 0001；D1D0 输入 01，G3G2G1G0 输出 0010；D1D0 输入 10，G3G2G1G0 输出 0100；D1D0 输入 11，G3G2G1G0 输出 1000。

4 分频器 CLK_DIV：输入 50MHz 信号，输出两个 62.5Hz 的时钟信号，每个周期依次输出 00, 01, 10, 11。

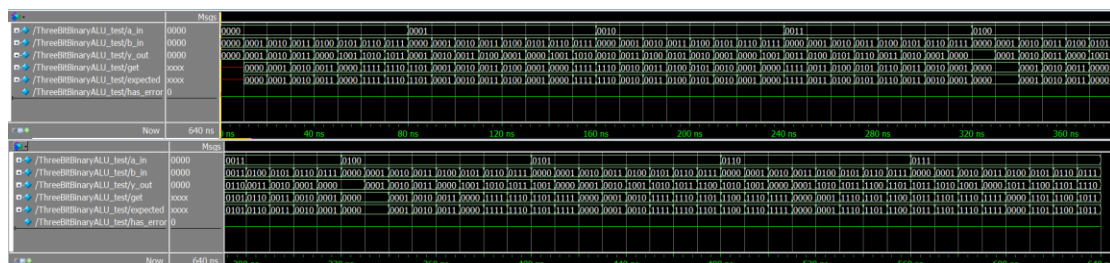
4-1 CLK_div16: 16 倍分频器，将输入信号的频率降为 1/16

4-2 CLK_div10: 10 倍分频器，将输入信号的频率降为 1/10

4-3 CLK_div5: 5 倍分频器，将输入信号的频率降为 1/5

4-4 CLK_div200k: 200k 倍分频器，将输入信号的频率降为 1/200k

3. 仿真波形图及其分析说明。



```

22 initial begin
23     has_error = 1'b0;
24     for (a_in = 4'b0000; a_in != 8; a_in = a_in + 1)
25     for (b_in = 4'b0000; b_in != 8; b_in = b_in + 1) begin
26         #1;
27         get = {y_out[2], y_out[1], y_out[0]};
28         get = get*(y_out[3] == 1?-1:1);
29         expected = a_in[1:0]*(a_in[2] == 1?-1:1) + b_in[1:0]*(b_in[2] == 1?-1:1);
30         if (get != expected) begin
31             $display("a_in = %d, b_in = %d, expected %d, get %d",
32                 a_in, b_in, expected, get);
33             has_error = 1'b1;
34         end
35     end
36     if (has_error == 1'b0) begin
37         $display("ALL TESTS PASSED!");
38     end
39 end

```

• 既然都写 testbench 了，不妨遍历所有输入情况，由于加法器不是时序逻辑电路，还可以方便地利用程序自动判断正误。采用两重 for 循环产生输入，每个输入都是 3 位，第 0 和 1 位是数值，第 2 位是符号，在 0-7 遍历就可以产生所有输入情况；

• 仿真时，窗口输出 ALL TESTS PASSED! 说明电路逻辑没问题（未判断-0 问题），下面简要挑两种情况说明：①输入 0100 和 0100，即输入-0 和-0，相加为 0，实际输出 0000，实现了“无-0”，符合预期；②输入 0111 和 0001，即输入-3 和 1，相加为-2，实际输出 1010，符合预期。

4. 设计和调试中遇到的问题及解决方法。

• 四位二进制全加器有八个输入端，仿真测试时，设置 8 个引脚的输入比较麻烦：采用右键全选，或者双击全选，再设置周期的方式会方便很多。

• 产生如下报错：MUX21 模块的实例化失败。据查是语言设置问题，但本地并没有存在这样的问题，切换仿真工具为自带仿真器问题就解决了。

```

# ** Fatal: (vsim-3039) EDA2.vt(174): Instantiation of 'MUX21' failed.
# Time: 0 ps Iteration: 0 Instance: /MUX21_vlg_vec_tst File: EDA2.vt
# FATAL ERROR while loading design
# Error loading design

```

• 调整某个模块后，需要在使用该模块的文件内右击点击更新模块，才能生效。

• 报错：同一个输入不可以有两个值：发现是两根相交的线莫名其妙多了一个连接点，去除就行

• 报错：与非门命名重复，重新命名即可

```

Quartus II 64-Bit Analysis & Synthesis was unsuccessful. 2 errors, 1 warning
293001 Quartus II Full Compilation was unsuccessful. 4 errors, 1 warning
275062 Logic function of type NAND2 and instance "inst" is already defined as a signal name or another logic function
12153 Can't elaborate top-level user hierarchy

```

• 测试代码报错：Verilog 不能使用*=，还是得把左侧的变量再抄一遍。

```

10170 Verilog HDL syntax error at ThreeBitBinaryALU_test.v(28) near text "*"; expecting "<=", or "="
10170 Verilog HDL syntax error at ThreeBitBinaryALU_test.v(28) near text "==" ; expecting "<=", or "="
10112 Ignored design unit "ThreeBitBinaryALU test" at ThreeBitBinaryALU test.v(2) due to previous errors

```

- 测试时报错:

```
ModelSim-Altera Error: # ** Error: (vsim-3053) D:/altera/project/EDA2/ThreeBitBinaryALU_test.v(19): Illegal output or inout port connection for "port 'Y0'".
ModelSim-Altera Info: #
ModelSim-Altera Info: #       Region: /ThreeBitBinaryALU_test/i1
ModelSim-Altera Error: # ** Error: (vsim-3053) D:/altera/project/EDA2/ThreeBitBinaryALU_test.v(19): Illegal output or inout port connection for "port 'Y1'".
ModelSim-Altera Info: #
ModelSim-Altera Info: #       Region: /ThreeBitBinaryALU_test/i1
ModelSim-Altera Error: # ** Error: (vsim-3053) D:/altera/project/EDA2/ThreeBitBinaryALU_test.v(19): Illegal output or inout port connection for "port 'Y2'".
ModelSim-Altera Info: #
ModelSim-Altera Info: #       Region: /ThreeBitBinaryALU_test/i1
ModelSim-Altera Error: # ** Error: (vsim-3053) D:/altera/project/EDA2/ThreeBitBinaryALU_test.v(19): Illegal output or inout port connection for "port 'SY'".
```

- 常见易错点: \leq 的含义跟 C++ 不同, 因此 for 循环中的判断用 \neq 更靠谱。
- 仿真波形中, 输出产生了高阻态, 经查发现在 testbench 中, 输出多开了一位, 改成 4 位后就没有上述现象了: 仿真中不像平时编程随便开数组大小, 最好做到用多少开多少, 方便后续调试。