哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2021 秋季

课程名称：数字逻辑设计（实验）

实验名称：十六进制计算器设计

实验性质：综合设计型

实验学时：6 地点：T2506

学生班级：计算机学院 6 班

学生学号：200110617

学生姓名：蔡嘉豪

评阅教师：

报告成绩：

实验与创新实践教育中心制

2021 年 12 月

注：本设计报告中各个部分如果页数不够，请大家自行扩页，原则是一定要把报告写详细，能说明设计的成果和特色。报告中应该叙述设计中的每个模块。设计报告将是评定每个人成绩的重要组成部分（**设计内容及报告写作**都作为评分依据）。

设计的功能描述

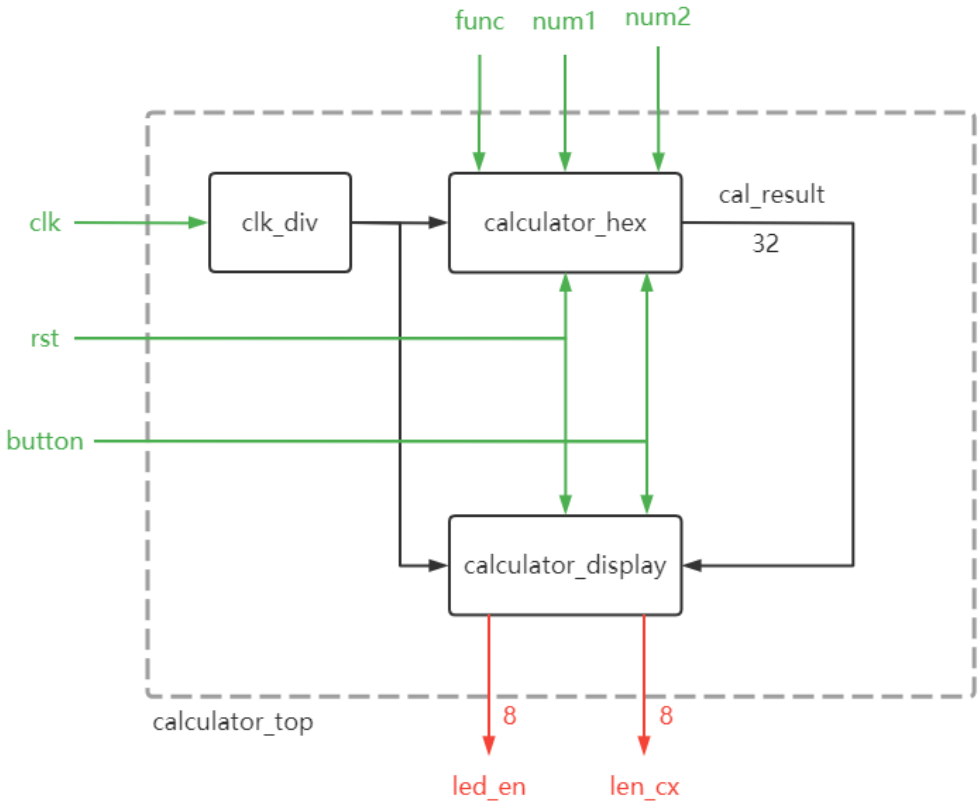
概述基本功能、详细描述自行扩展的功能

本次设计内容为一个十六进制计算器，输入两个 8 位二进制数，支持加、减、乘、除、取余、平方运算以及连续运算，输出结果以 8 位 16 进制数在数码管上显示。

本次实验完全按照实验要求设计和完成，未自行扩展额外功能。

系统功能详细设计

用硬件框图描述系统主要功能及各模块之间的相互关系



各模块描述

包括模块功能，输入、输出端口、变量含义及主要设计代码

共有三个模块：clk_div, calculator_hex, calculator_display

---

clk_div 模块

功能：clk_div 只负责对传入的 100MHz 时钟进行分频，生成一个 10MHz 的时钟（因为 100MHz 的时钟周期太短，加减乘除无法在一个时钟周期内完成）

输入：clk, locked

输出：clk_g

---

calculator_hex 模块

功能：calculator_hex 负责接受 2 个操作数以及读取操作符，在检测到 button 按下时，判断运算类型及是否连续运算，执行计算并将结果存到 cal_result；在检测到 rst 被按下时清零 cal_result

输入端口描述：

```
input   wire        clk,      //传入分频后的 10MHz 时钟
input   wire        rst,      //复位键
input   wire        button,   //启动(计算)键
input   wire [2:0]   func,    //运算类型
input   wire [7:0]   num1,    //操作数 1
input   wire [7:0]   num2,    //操作数 2
```

输出端口描述：

```
output  reg  [31:0]  cal_result   //运算结果
```

变量含义：

```
parameter PLUS      = 3'b000;
parameter SUBTRACT  = 3'b001;
parameter MULTIPLY  = 3'b010;
parameter DIVIDE    = 3'b011;
parameter MOD       = 3'b100;
parameter SQUARE    = 3'b101;
wire [2:0] cur_operator   //当前运算类型,值为上述六种之一
reg  [1:0] status   //模块当前状态, 1'b0 为 OFF, 1'b1 为 ENABLED
reg  [31:0] cal_tmp  //缓存上一次计算结果，用于连续运算
```

主要设计代码：

见下一页

```
always @(posedge clk or negedge rst_n) begin
    if (~rst_n) begin
        status = OFF;
        cal_tmp = NO_TMP;
    end
    else if (status == OFF) begin
        if (button) begin
            status = ENABLED;
        end
    end
    // 此处两个条件同时成立才允许进入计算，简单防抖处理
    else if (status == ENABLED && button == 1'b0) begin
        case (cur_operator)
            PLUS:
                // 通过计算缓存是否为空判断是连续计算还是新计算
                if (cal_tmp == NO_TMP) cal_tmp = num1 + num2;
                else cal_tmp = cal_tmp + num2;
            SUBTRACT:
                if (cal_tmp == NO_TMP) cal_tmp = num1 - num2;
                else cal_tmp = cal_tmp - num2;
            MULTIPLY:
                if (cal_tmp == NO_TMP) cal_tmp = num1 * num2;
                else cal_tmp = cal_tmp * num2;
            DIVIDE:
                if (cal_tmp == NO_TMP) cal_tmp = num1 / num2;
                else cal_tmp = cal_tmp / num2;
            MOD:
                if (cal_tmp == NO_TMP) cal_tmp = num1 % num2;
                else cal_tmp = cal_tmp % num2;
            SQUARE:
                if (cal_tmp == NO_TMP) cal_tmp = num1 * num1;
                else cal_tmp = cal_tmp * cal_tmp;
        endcase
        cal_result = cal_tmp;
        status = OFF;
    end
end
```

calculator_display 模块

模块功能： calculator_display 负责控制数码管显示计算结果的功能（包括将 cal_result 由 32 位二进制数转换成 8 位十六进制，以及控制数码管显示这个十六进制数），当 button 第一次被按下之后进入工作状态，此后时刻读取 cal_result 的数据并在数码管上显示它们转换成十六进制后的表示形式

输入：

```
input  wire       clk   ,         // 10MHz 时钟
input  wire       rst   ,         // 复位键
input  wire       button,          // 计算键
input  wire [31:0] cal_result,   // 计算结果
```

输出：

```
output reg [7:0]   led_en,
output reg         led_ca,
output reg         led_cb,
output reg         led_cc,
output reg         led_cd,
output reg         led_ce,
output reg         led_cf,
output reg         led_cg,
output reg         led_dp
```

变量含义：

/*基本状态变量*/

```
reg status;    // 模块当前工作状态,0 为 OFF,1 为 ENABLED
reg [7:0] tube_en_status;   // 存储数码管使能信号
parameter ALL_TUBES_OFF = 8'b1111_1111;  //数码管初始状态
```

/*各数字与字母在数码管各位置的表示常量(不包括 led_dp)*/

```
parameter NUMBER_0 = 7'b0000001;
parameter NUMBER_1 = 7'b1001111;
parameter NUMBER_2 = 7'b0010010;
//...
parameter NUMBER_9 = 7'b0001100;
parameter CHAR_A = 7'b0001000;
//...
parameter CHAR_F = 7'b0111000;
```

/*数码管显示计数器的触发点（每 2ms 一个常量）*/

```
parameter SLICE_PERIOD = 21'd160000;
parameter DIG_0_TRIG = 21'd160000;
parameter DIG_1_TRIG = 21'd20000;
//...
parameter DIG_7_TRIG = 21'd140000;
reg [20:0] slice_cnt; //2ms 计数器
```

```
/*4 位二进制到 1 位十六进制的映射表，用于转换 cal_result，复位时赋值*/
reg [6:0] binary2hex [15:0];
binary2hex[4'b0000] = NUMBER_0;
binary2hex[4'b0001] = NUMBER_1;
binary2hex[4'b0010] = NUMBER_2;
binary2hex[4'b0011] = NUMBER_3;
binary2hex[4'b0100] = NUMBER_4;
binary2hex[4'b0101] = NUMBER_5;
binary2hex[4'b0110] = NUMBER_6;
binary2hex[4'b0111] = NUMBER_7;
binary2hex[4'b1000] = NUMBER_8;
binary2hex[4'b1001] = NUMBER_9;
binary2hex[4'b1010] = CHAR_A;
binary2hex[4'b1011] = CHAR_B;
binary2hex[4'b1100] = CHAR_C;
binary2hex[4'b1101] = CHAR_D;
binary2hex[4'b1110] = CHAR_E;
binary2hex[4'b1111] = CHAR_F;
```

主要设计代码：

见下页

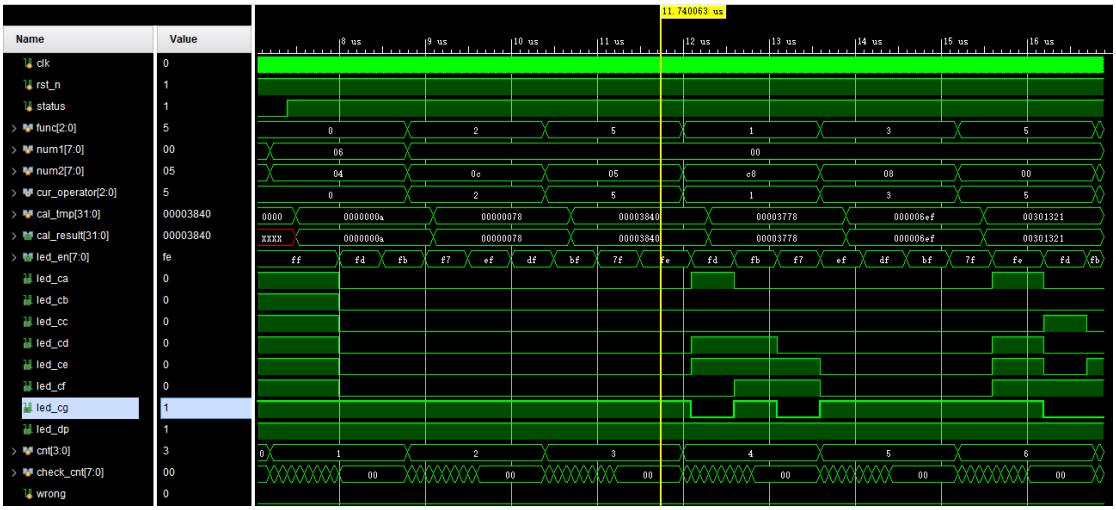
/*此处略去 2ms 计数器的控制代码*/

/*主要控制代码*/

```
always @(posedge clk or negedge rst_n) begin
	// 初始化
	if (~rst_n) begin
		led_dp = 1;
		status = OFF;
		led_en = ALL_TUBES_OFF;
		binary2hex[4'b0000] = NUMBER_0;
		binary2hex[4'b0001] = NUMBER_1;
		/* ...4 位二进制到 1 为十六进制的映射表，赋初值，此处略...*/
		binary2hex[4'b1111] = CHAR_F;
	end
	else if (status == OFF) begin
		{led_ca, led_cb, led_cc, led_cd, led_ce, led_cf, led_cg} = 7'b1111111;
		if (button) begin
			status = ENABLED;
			tube_en_status = 8'b1111_1110;
		end
	end
	else if (status == ENABLED) begin
		// 每 2ms，将数码管使能信号左移一位，同时给七个显示位赋值
		// 二进制转十六进制的规则是：每 4 位二进制数恰好对应 1 位十六进制数
		// 每 2ms，从 32 位的 cal_result 中取出 4 位，从映射表中取出这 4 位对应的十六进制数，把对应的显示常量赋给七个显示位
		case (slice_cnt)
			DIG_0_TRIG: begin
				{led_ca, led_cb, led_cc, led_cd, led_ce, led_cf, led_cg} = binary2hex[cal_result[3:0]];
				tube_en_status = {tube_en_status[6:0], tube_en_status[7]};
				led_en = tube_en_status;
			end
			DIG_1_TRIG: begin
				{led_ca, led_cb, led_cc, led_cd, led_ce, led_cf, led_cg} = binary2hex[cal_result[7:4]];
				tube_en_status = {tube_en_status[6:0], tube_en_status[7]};
				led_en = tube_en_status;
			end
			/*...每 2ms 执行一次，中间类似步骤省略...*/
			DIG_7_TRIG: begin
				{led_ca, led_cb, led_cc, led_cd, led_ce, led_cf, led_cg} = binary2hex[cal_result[31:28]];
				tube_en_status = {tube_en_status[6:0], tube_en_status[7]};
				led_en = tube_en_status;
			end
		endcase
	end
end
```

调试报告

仿真波形截图及仿真分析



如上图所示，status 拉高代表进入工作状态

可以看到,第一次计算操作数为 6 与 4,cur_operator 显示当前运算类型为+,在 cal_tmp 和 cal_result 可以看到运算结果为 0x0a，即十进制的 10，运算正确；

后续几次运算都为连续运算，num1 为 0，不参与运算；

第二次运算为 10×12，结果为 120，十六进制表示为 0x0078，正确；

第三次运算为 120 的平方，结果为 14400，十六进制表示为 0x3840，正确；

第四次运算为 14400-200=14200，十六进制表示为 0x3778，正确；

与此同时，可以看到数码管 8 位使能信号序列 led_en 在不断前移，显示位信号 led_cx 同步改变；

特别地，图中黄色坐标线位置附近，可以看到数码管显示的数字是 00003840，与此时计算结果是匹配的。

综上所述，从波形可以看出，所做的设计已经具备了连续运算和数码管显示功能

设计过程中遇到的问题及解决方法

1. 32 位二进制数如何转成 8 位 16 进制数显示在数码管上？

结合编码理论知识可知，一个 4 位二进制数对应 1 位十六进制数，所以，一个二进制串想要转换成十六进制串，只需要逐 4 位地扫描二进制串，将每 4 位二进制串转换成十六进制数即可。

进一步分析，其实我们要实现的不是"**二进制串转十六进制串**"功能，而是"**读取二进制串并将其十六进制形式显示在数码管上**"的功能。

而在控制数码管显示时，我们采用的方法是每 2ms 让使能信号移动一位，同时改变 led_cx 显示当前位应该显示的数字，利用人眼视觉暂留效应使得数码管看起来像在同时显示 8 位。

结合这个原理分析，其实我们需要做的只有：每次切换数码管使能位时，从二进制串 cal_result 中读取当前位置对应的那个 4 位二进制数（如数码管第 1 位对应 cal_result[3:0]，第 2 位对应 cal_result[7:4]...），然后将其映射成十六进制数字或字母，赋值给 led_cx 即可

与此同时，为了方便 4 位二进制->1 位 16 进制的转换，使用了一个映射表

```
reg [6:0] binary2Hex [15:0]
binary2hex[4'b0000] = NUMBER_0;
binary2hex[4'b0001] = NUMBER_1;
binary2hex[4'b0010] = NUMBER_2;
/*...略...*/
```

每次给 led_cx 赋值时只需要取 binary2Hex[cal_result[x+3:x]]即可

2. button 按一次，计算的次数非常多

分析代码可知，按照原来的写法（检测到 button 为 1 则执行计算），人每手动按一次 button，都会执行非常多次计算，因为 button 会抖动，人按一次可能会经历很多次高电平，所以导致会计算次数特别多。

解决方法是：

(1)设置两个状态 OFF 和 ENABLED，让该模块在两个状态之间转换；
(2)处于 OFF 状态时，按下 button 则切换到 ENABLED 状态；
(3)执行完单次计算后，立即切换回 OFF 状态；
(4)当且仅当处于 ENABLED 状态且 button 为低电平时才执行一次计算

大致代码如下：

```
if (~rst_n) begin
    //复位相关操作
end
else if (status == OFF) begin
    if (button) status = ENABLED;
else if (status == ENABLED && button == 1'b0) begin
    //执行单次计算
end
```

课程设计总结

包括设计的总结和还需改进的内容以及收获

总结：

1．开始做前进行了模块的拆分设计，遵守单一模块单一职责的原则
2．模块的拆分帮助了代码和功能的解耦，calculator_display 模块只需要负责对传入的信号进行处理并将其显示到数码管上，calculator_hex 模块则只需要处理计算逻辑，然后将 cal_result 传给 display 模块
3．实现了十六进制计算器的基本计算、连续运算、显示功能
4．通过设置状态转换，大幅减少了 button 的抖动次数，现在按一次 button 只会连续运算 1~4 次

还需要改进的地方：

1．button 尚未能完全消除抖动
2．代码写得还不够简洁

收获：

1．模块化设计思想，自顶向下设计，先不用管实现细节，先将项目按照功能拆分为若干个低耦合的模块，再对每个模块分别进行讨论和细节设计，
2．二进制串转十六进制串的方法
3．对为什么要做时钟分频有了比之前更深的理解
4．通过状态转换来使代码结构更清晰，流程描述更易懂，容易改动和维护
5．对电子设计工程的复杂度和细节把控程度有了更多的了解，知道要完整做出一个电子产品需要考虑到各种反常理的细节，需要方方面面做好优化，做到极致