

（深圳）

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 |