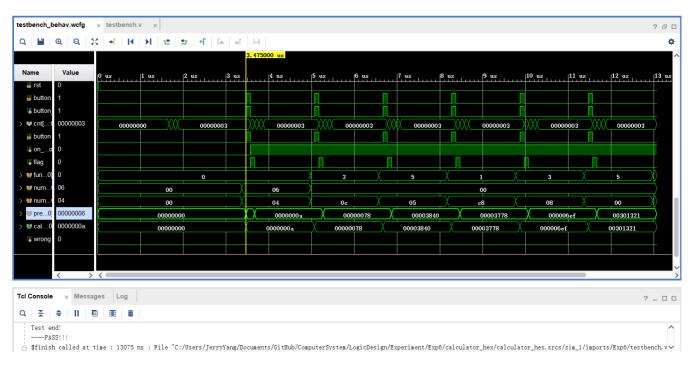
实验6 十六进制计算器设计

calculator_hex

仿真波形



波形分析

模块定义

calculator_hex实验是实现十六进制计算器的加、减、乘、除、模和平方的功能,默认进行连续运算,模块定义如下:

- calculator_top.v 电路总控制模块,调用各模块功能最后实现运算并显示在数码管上。
- clk_div 时钟IP核,用于时钟分频,分频时钟clk_g的频率为10MHz
- **key_filter.v** 消抖模块,用于在上板操作时,避免了按键时不稳定脉冲导致的波形抖动,也避免持续按键导致的非期望的连续计算
- calculator_hex.v 实现计算器核心功能的模块,用于完成加减乘除模平方的运算,默认进行连续运算,按下rst 后重置寄存器,每按下一次button进行一次运算
- calculator_display.v 实现将计算器得出的32位2进制数cal_result转为16进制数在8位数码管上显示

信号定义

只对模块中自定义信号和部分给定信号进行说明

calculator_top.v

```
wire clk_g ; // 分频时钟
wire locked ; // 时钟锁定信号,当分频时钟输出稳定后为高电平
wire button_f ; // button消除抖动后的信号,作为其他模块的button输入信号
wire on_button ; // 开始计算后处于工作状态的标记信号
wire [31:0] cal_result; // 传递32位二进制计算结果的信号
wire [7:0] led_w ; // _w代表wire类型,数码管段选信号
```

```
// parameter CNT_MAX = 32'd1000; // 计数器最大数值,控制两次按键时间间隔 parameter CNT_MAX = 32'd3; // simulation reg [31:0] cnt = 0 ; // 两次按键最小时间间隔计数,用于消除毛刺信号 wire rst_n = ~rst; // rst_n下降沿复位
```

calculator hex.v

```
wire rst_n = ~rst; // rst_n下降沿复位
reg on_button = 0 ; // 开始计算后处于工作状态的标记信号
reg flag = 0 ; // 用于标记按下button后的第一个时钟周期
reg [31:0] prev_result = 0 ; // 连续运算时,用于保存中间计算结果
```

calculator_display.v

```
// parameter SCAN_CNT_MAX = 20'd1_0000; // 扫描时间间隔计数最大时钟周期数 parameter SCAN_CNT_MAX = 20'd5; // simulation wire rst_n = ~rst; // rst_n下降沿复位 reg on_button = 0 ; // 开始计算后,处于工作状态的标记信号 reg [7:0] cur_code = 8'hff; // 当前结果位置数值显示到数码管上对应的编码 reg [20:0] scan_cnt = 0 ; // 扫描计数器 reg [2:0] scan_pos = 0 ; // 当前扫描的位置 wire [3:0] result [7:0] ; // 重新分割32位2进制计算结果的位,分割成8个4位数值,只是为了人直接的阅读方面
```

波形时序分析

• 对于第一次计算,相较于接下来的计算稍微特殊,在时钟分频之后,分频时钟的频率为10MHz,时钟周期为100ns

| clk(ns) | rst | button | on_button | flag | func | num1 | num2 | prev_result | cal_result |
|-----------------|------|--------|-----------|------|--------|---------|------------|-------------|------------|
| 0-20(+) | 1=>0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20(+)-3375(+) | 0 | 0 | 0 | 0 | 0 | 0=>'h06 | 0=>'h04 | 0 | 0 |
| 3375(+)-3475(+) | 0 | 0=>1 | 0 | 0 | 0 | 'h06 | 'h04 | 0=>'h6 | 0=>'ha |
| 3475(+)-3575(+) | 0 | 1=>0 | 0=>1 | 0=>1 | 0 | 'h06 | 'h04 | 'h6 | 'ha |
| 3575(+)-3675(+) | 0 | 0 | 1 | 1=>0 | 0 | 'h06 | 'h04 | 'h6=>'ha | 'ha |
| 3675(+)-4975(+) | 0 | 0 | 1 | 0 | 0=>'h2 | 'h06=>0 | 'h04=>'h0c | 'ha | 'ha |

由上表可见,在开始第一次计算之前(3375ns之前,之后只更新了num1和num2信号的值),对应各数值均为 0,rst之后的状态就是此时的状态

- 当第一次按下button时(3475ns),在当前时钟周期由时序逻辑控制的prev_result将num1数值存入,组合逻辑控制的cal_result将prev_result和num2进行运算,得到结果'ha
- o 在第一次按下button后的第一个时钟上沿到来的时候(3575ns),运算结果稳定一周期,该周期由上一周期on_button尚未修改确定(即标记了第一次按下button前的时间),对应到代码如下:

```
always @(posedge clk, negedge rst_n) begin
   if (~rst_n)
       prev_result <= 0;
   else if (!on_button)
       prev_result <= num1; // new requirement given during class
   else if (on_button && flag)
       prev_result <= cal_result;
   else
       prev_result <= prev_result;
end</pre>
```

- o 在第一次按下button后的第二个时钟上沿到来的时候(3675ns), prev_result更新为先前计算结果, flag 信号上一周期被更改的1确定。
- 对于此后的连续运算过程,与第一次计算无异,但prev_result的更新只由flag高电平的1个时钟周期控制 波形给出的计算过程如下:

| cnt | func | prev_result | num1 | num2 | cal_result |
|-----|------|-------------|------|------|------------|
| 0 | 0 | 'h00000006 | 'h06 | 'h04 | 'h0000000a |
| 1 | 'h2 | 'h0000000a | 0 | 'h0c | 'h00000078 |
| 2 | 'h5 | 'h00000078 | 0 | 'h05 | 'h00003840 |
| 3 | 'h1 | 'h00003840 | 0 | 'hc8 | 'h00003778 |
| 4 | 'h3 | 'h00003778 | 0 | 'h08 | 'h000006ef |
| 5 | 'h5 | 'h000006ef | 0 | 'h00 | 'h00301321 |

对应于十进制计算过程如下:

| cnt | func | prev_result | num1 | num2 | cal_result |
|-----|------|-------------|------|-----------|------------|
| 0 | 加法 | 6 | 6 | 4 | 10 |
| 1 | 乘法 | 10 | 0 | 12 | 120 |
| 2 | 平方 | 120 | 0 | 5 (与结果无关) | 14400 |
| 3 | 减法 | 14400 | 0 | 200 | 14200 |
| 4 | 除法 | 14200 | 0 | 8 | 1775 |
| 5 | 平方 | 1775 | 0 | 0 | 3150625 |

综上所述,本次实验完整的实现了十六进制计算器的基本功能,同时使用按键消抖模块避免了毛刺信号对按键的干扰,也避免了长按时对按键的错误判断。仿真通过,上板验证良好,说明本实验完成良好。