数字逻辑与处理器基础实验
第三次作业

无 81 马啸阳 2018011054

2020 年 5 月 6 日

1 实验目的

- 掌握静态时序分析 STA
- 掌握略复杂的组合和时序电路设计
  - 设计频率计，对一个未知频率的周期信号进行频率测量

2 实验原理

频率计测量未知频率的周期信号，在 1s 内对信号周期进行计数，即为此周期信号的频率，框图如图 1所示。

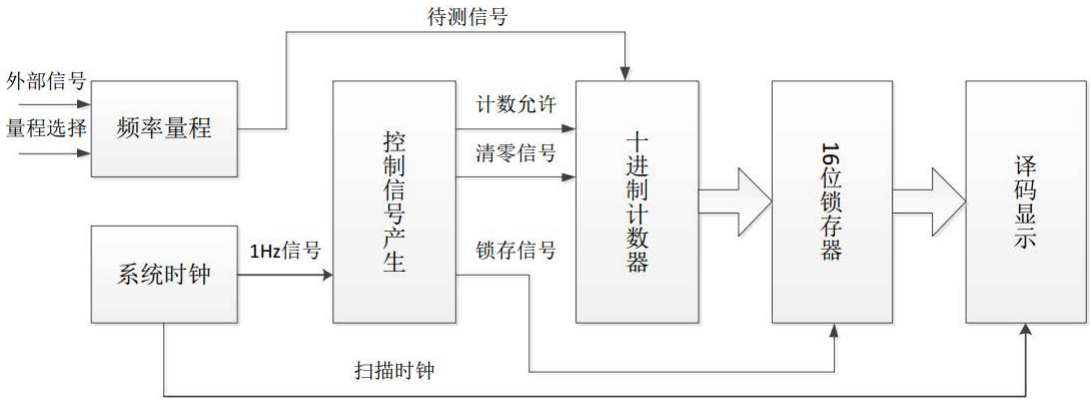

图 1: 频率计框图

频率计内部包括频率量程处理模块（10 分频）、时钟频率产生模块、控制信号产生模块、十进制计数器模块、锁存器模块、译码显示模块等。频率量程处理模块根据是否选择高量程，输出原信号或其十分频的待测信号。时钟频率产生模块根据系统时钟进行分频，生成 1Hz 的控制信号与 1kHz 的扫描时钟。控制信号通过 1Hz 时钟信号产生计数允许、清零信号以及锁存信号。十进制计数器对待测信号在 1s 内进行计数，由计数允许与清零信号控制。16 位锁存器由锁存信号控制，存储计数器的计数值。而译码显示单元将 16bit，4 位 BCD 码循环译码，由扫描时钟控制显示在七段数码管上。

2.1 频率量程处理

使用开关（SW7）控制量程，低量程直接输出原信号，高量程输出十分频。

分频的方法为：设置一个状态用以计数周期，每遇到一个上升沿计数 2，计数达到 10 时将输出信号反向，则输出信号是原信号的十分频（为节省线路规模，也可每遇到一个上升沿计数 1，达到 5 时反向）。

2.2 系统时钟模块

开发板上 W5 端口提供 100MHz 时钟，进行分频得到 1Hz 控制信号与 1kHz 扫描时钟。其中 1Hz 信号可以使用 1kHz 信号再进行分频，避免计数规模过大。

2.3 控制信号产生模块

控制信号产生模块输入 1Hz 时钟，产生计数允许（高电平有效）、清零信号（低电平有效）以及锁存信号，以 3s 为一周期，每周期三个状态如表 1所示，第一秒清零（锁存器锁定），第二秒计数（锁存器锁定），第三秒将结果存至锁存器（不计数）。

	计数	清零	锁存
0∼1s	0	0	1
1∼2s	1	1	1
2∼3s	0	1	0

表 1: 控制信号周期

2.4 4 位十进制计数器

计数器以 BCD 码计数 4 位十进制数，共 16bit，在输入信号上升沿计数，受清零信号（低电平有效）与计数允许信号（高电平有效）控制。需要注意的是此计数器的进位，由于是 BCD

码，需逐位考虑其 BCD 码是否为 9 以及更低位是否进位。

2.5 锁存器模块

Lock 信号有效时（高电平）输出锁定，否则输出透明显示计数器值。

2.6 译码显示模块

4 个 7 段数码管分别显示十进制数的 4 位。由扫描信号控制每次使能一个数码管循环显示，7 段译码器与使能信号同步。

2.7 测试方法

测试方法框图如图 2所示，使用一个待测信号输入模块 siginput，其输入 2bit 测试模式 testmode，对应 4 种不同频率，对系统时钟 sysclk 分频。随后将待测信号接入频率计，频率计输入待测信号、系统时钟、量程选择 modecontrol（高电平对应高量程），输出 7 段数码管 LED 信号 cathodes[7:0]、使能信号 AN[3:0] 以及 LED 灯表明量程的 highfreq。

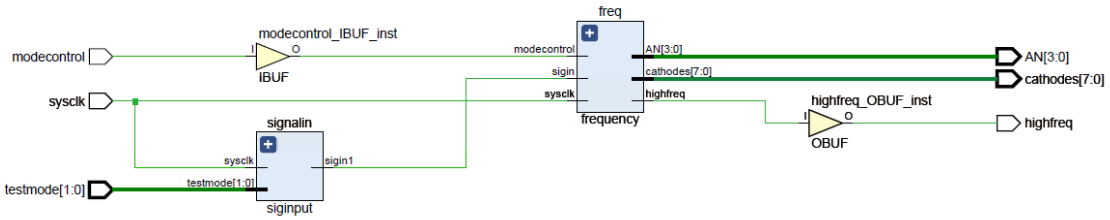

图 2: 测试框图

3 实验代码

3.1 文件清单

```
.
├── BCD7.v ..... 七段译码管模块
├── clock.v ..... 系统时钟模块
├── control.v ..... 控制信号产生模块
├── counter.v ..... 4 位十进制计数器
└── decoder.v ..... 译码显示模块
```

- frequency.v ..... 频率计模块
- latch.v ..... 锁存器模块
- range.v ..... 频率量程处理
- siginput.v ..... 待测信号输入模块
- siginput.xdc ..... 待测信号输入模块管脚约束
- test_tb.v ..... testbench
- test.v ..... 顶层测试模块
- test.xdc ..... 测试管脚约束

3.2 频率量程处理

```
module range (range, sigin, sigout);
input range, sigin;
output sigout;
reg sig10;
reg [4:0] state;

assign sigout = range ? sig10 : sigin;

initial begin
    state <= 0;
    sig10 <= 0;
end

always @(posedge sigin) begin
    if (state == 0)
        sig10 = ~sig10;
    state = state + 4'd2;
    if (state == 4'd10)
        state = 4'b0;
end

endmodule
```

3.3 系统时钟模块

对 100MHz 输入信号分频，先 100000 分频得到 1kHz 扫描时钟，再对扫描时钟 1000 分频得到 1Hz 控制信号。

```
module clock (sysclk, scan_clk, control_clk);
input sysclk;           // 100MHz
output scan_clk;        // 1kHz
output control_clk;     // 1Hz
reg scan_clk, control_clk;
reg [16:0] scan_state;
reg [9:0] control_state;

initial begin
        scan_state <= 0;
        control_state <= 0;
        scan_clk <= 0;
        control_clk <= 0;
end

always @(posedge sysclk) begin
        if (scan_state == 0)
                scan_clk = ~scan_clk;
                scan_state = scan_state + 17'd2;
        if (scan_state == 17'd100000)
                scan_state = 17'b0;
end

always @(posedge scan_clk) begin
        if (control_state == 0)
                control_clk = ~control_clk;
        control_state = control_state + 10'd2;
        if (control_state == 10'd1000)
                control_state = 10'b0;
end

endmodule
```

3.4 控制信号产生模块

控制信号产生是 Moore 有限状态机，使用一段式，初值要赋予输出，同时对未定义的状态以 default 实现自启动。

```
module control (clk, reset, enable, latch);
input clk;
output reset, enable, latch;

reg [1:0] state;
reg reset, enable, latch;

initial begin
    state <= 2'b00;
    reset <= 0;
    enable <= 0;
    latch <= 1;
end

always @(posedge clk) begin
    case (state)
        2'b00: begin
            state <= 2'b01;
            reset <= 1;
            enable <= 1;
            latch <= 1;
        end
        2'b01: begin
            state <= 2'b10;
            reset <= 1;
            enable <= 0;
            latch <= 0;
        end
        2'b10: begin
            state <= 2'b00;
            reset <= 0;
            enable <= 0;
            latch <= 1;
        end
        default: begin
```

```
            state <= 2'b00;
            reset <= 0;
            enable <= 1;
            latch <= 1;
        end
    endcase
end

endmodule
```

3.5 4 位十进制计数器

不进位时，个位加 1。若个位为 9 发生进位，将其置零，并检查十位是否为 9，依此类推。

```
module counter (reset, enable, sig, out);
input reset, enable, sig;
output [15:0] out;

reg [15:0] out;

initial begin
    out <= 16'b0000_0000_0000_0000;
end

always @(negedge reset or posedge sig) begin
    if (~reset)
        out <= 0;
    else if (enable) begin
        if (out[3:0] == 4'b1001) begin
            out[3:0] <= 0;
            if (out[7:4] == 4'b1001) begin
                out[7:4] <= 0;
                if (out[11:8] == 4'b1001) begin
                    out[11:8] <= 0;
                    if (out[15:12] == 4'b1001) begin
                        out[15:12] <= 0;
                    end
                    else begin
                        out[15:12] <= out[15:12] + 1;
```

```
                    end
                end
                else begin
                    out[11:8] <= out[11:8] + 1;
                end
            end
            else begin
                out[7:4] <= out[7:4] + 1;
            end
        end
        else begin
            out[3:0] <= out[3:0] + 1;
        end
    end
end

endmodule
```

3.6 锁存器模块

```
module latch(lock, in, out);
input lock;
input [15:0] in;
output [15:0] out;
reg [15:0] out;

always @(*) begin
    if (~lock)
        out <= in;
end

endmodule
```

3.7 译码显示模块

每次在扫描信号上升沿改变使能位，同时将对应 BCD 码输入 BCD 译码器，输出至 7 段数码管。

```
module decoder (clk, count, leds, EN);
input clk;
input [15:0] count;
output [7:0] leds;
output [3:0] EN;

wire [15:0] count;
wire [7:0] leds;
reg [3:0] EN;
reg [3:0] cur_count;

BCD7 bcd7 (cur_count, leds);

initial begin
    EN <= 4'b0111;
end

always @(posedge clk) begin
    case (EN)
        4'b0111: begin
            EN <= 4'b1011;
            cur_count <= count[11:8];
        end
        4'b1011: begin
            EN <= 4'b1101;
            cur_count <= count[7:4];
        end
        4'b1101: begin
            EN <= 4'b1110;
            cur_count <= count[3:0];
        end
        4'b1110: begin
            EN <= 4'b0111;
            cur_count <= count[15:12];
        end
        default: begin
            EN <= 4'b1111;
            cur_count <= 4'b0000;
        end
```

```
    endcase
end

endmodule
```

3.8 频率计

频率计单元只需将各单元按逻辑连接即可。

```
module frequency (sigin, sysclk, modecontrol, highfreq, cathodes, AN);
input sigin, sysclk, modecontrol;
output highfreq;
output [7:0] cathodes;
output [3:0] AN;

wire [15:0] count, latch_out;
wire scan_clk, control_clk, reset, enable, latch, sig;

assign highfreq = modecontrol;
clock clk(.sysclk(sysclk), .scan_clk(scan_clk), .control_clk(control_clk));
counter c(.reset(reset), .enable(enable), .sig(sig), .out(count));
control ctrl(.clk(control_clk), .reset(reset), .enable(enable), .latch(latch));
range r(.range(modecontrol), .sigin(sigin), .sigout(sig));
latch l(.lock(latch), .in(count), .out(latch_out));
decoder d(.clk(scan_clk), .count(latch_out), .leds(cathodes), .EN(AN));

endmodule
```

3.9 测试模块

测试模块连接待测信号产生模块及频率计。

```
module test(
            input [1:0] testmode,
            input sysclk,
            input modecontrol,
            output highfreq,
            output [7:0]cathodes,
            output[3:0] AN
```

```
        );
wire sigin;
siginput signalin(testmode, sysclk, sigin);
frequency freq(sigin, sysclk, modecontrol, highfreq, cathodes, AN);
endmodule
```

3.10 仿真代码

仿真时由于速度过慢，因此所有信号频率调大 1000 倍，这样显示的数据仍然保持一致（计数也只计 1ms，相当于仿真中 1ms 为现实中 1s），只需将所有分频调整为原先 1/1000（对扫描信号分频得到控制信号除外）。

依次测试低量程的 testmode 为 00、01、10、11 的情形，以及高量程的 testmode 为 00、01、10、11 的情形。

```
`timescale 1ns/1ps
`define PERIOD 10

module test_tb;
reg [1:0] testmode;
reg sysclk;
reg modecontrol;
wire highfreq;
wire [7:0] cathodes;
wire [3:0] AN;

test u_test (
                .testmode               ( testmode      [1:0] ),
                .sysclk                 ( sysclk              ),
                .modecontrol            ( modecontrol         ),

                .highfreq               ( highfreq            ),
                .cathodes               ( cathodes      [7:0] ),
                .AN                     ( AN            [3:0] )
         );

initial begin
        testmode <= 2'b00;
        sysclk <= 0;
        modecontrol <= 0;
```

```
26 end
27
28 initial fork
29         forever
30                 #(`PERIOD/2) sysclk <= ~sysclk;
31         #3000000 testmode <= 2'b01;
32         #6000000 testmode <= 2'b10;
33         #9000000 testmode <= 2'b11;
34         #12000000 modecontrol <= 1;
35         #12000000 testmode <= 2'b00;
36         #15000000 testmode <= 2'b01;
37         #18000000 testmode <= 2'b10;
38         #21000000 testmode <= 2'b11;
39         #24000000 $finish;
40 join
41
42 endmodule
```

4　仿真结果与分析

仿真结果如图 3所示（仿真结果显示较小，但放大也能清晰显示）。count 为计数器的输出结点，可见每个 3s 周期内，计数、保持、清零依次进行。相应地，latch_out 为锁存器输出，一直保持计数器的计数值直至有新的计数值。前 12ms 仿真为低量程，依次显示 3125、6250、0050、2500，在 12500Hz 时产生溢出，其余三个计数结果正确。而高量程进行了十分频，统计结果为实际频率的十分之一，依次显示 0313、0625、0005、1250，结果正确。

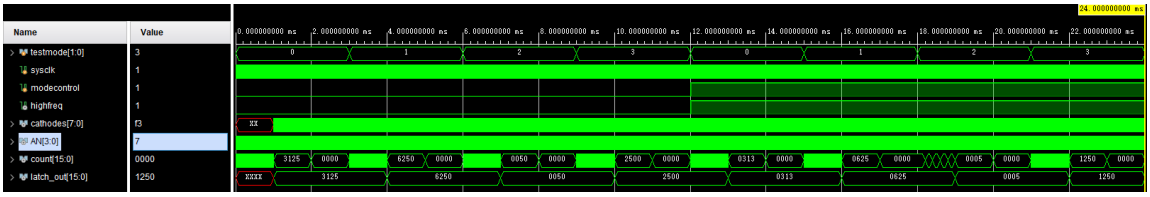

图 3: 仿真结果

对于七段数码管部分，仿真放大最后一部分 1250 为例。如图 4所示，使能信号依次为十六进制 7、b、d、e（二进制 0111、1011、1101、11110），即依次使能千位、百位、十位、个位，而对应的七段数码管输入十六进制 f3、49、25、81，为包括小数点的 1、2、5、0，确实正

确循环显示。

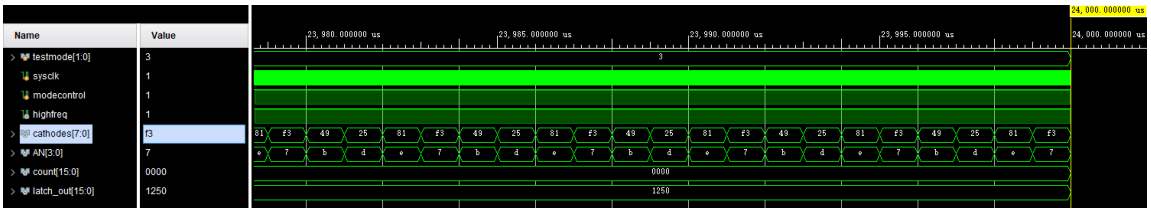

图 4: 七段数码管仿真放大

后仿结果如图 5与图 6所示，除边缘毛刺外同样正确运行。

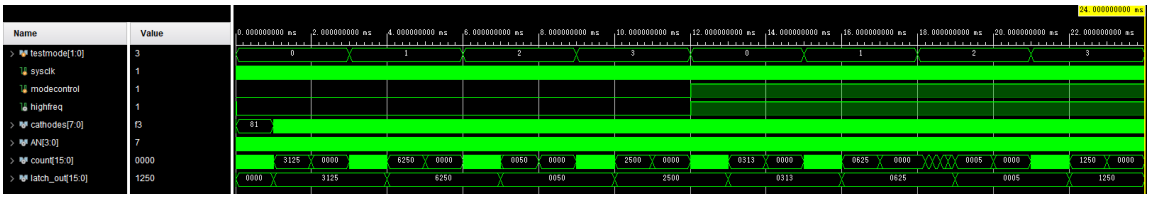

图 5: 后仿结果

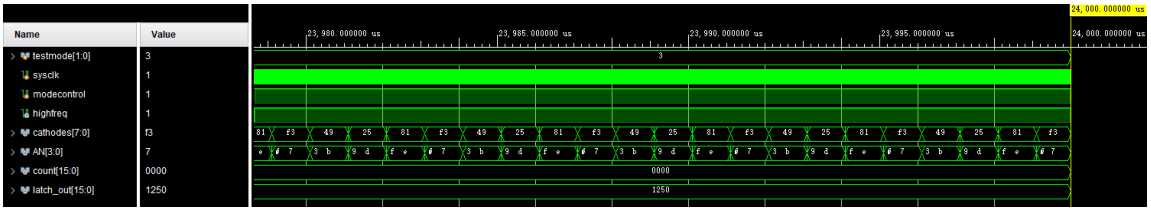

图 6: 七段数码管后仿放大

5　综合情况

面积占用情况如图 7所示，其中锁存器模块使用了 16 个寄存器作为锁存器使用，对应锁存器的 16 位。

Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	Bonded IOB (106)	BUFGCTRL (32)
test	72	98	45	72	17	1
freq (frequency)	55	77	33	55	0	0
signalin (siginput)	17	21	17	17	0	0

图 7: 面积占用

控制信号产生模块和译码显示模块分别生成了 3、4 个状态的状态机，综合产生了独热码。

State	New Encoding	Previous Encoding
iSTATE	001	00
iSTATE0	010	01
iSTATE1	100	10

INFO: [Synth 8-3354] encoded FSM with state register 'state_reg' using encoding 'one-hot' in module 'control'
WARNING: [Synth 8-327] inferring latch for variable 'out_reg' [D:/Verilog/Digital-Logic-and-Processor/hw3/frequency/latch.v:9]

State	New Encoding	Previous Encoding
iSTATE0	0001	0111
iSTATE1	0010	1011
iSTATE2	0100	1101
iSTATE3	1000	1110

INFO: [Synth 8-3354] encoded FSM with state register 'EN_reg' using encoding 'one-hot' in module 'decoder'

图 8: 有限状态机

时序性能如图 9所示，满足所有时序要求。

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.986 ns	Worst Hold Slack (WHS):	0.230 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	74	Total Number of Endpoints:	74	Total Number of Endpoints:	39

All user specified timing constraints are met.

图 9: 时序性能

前 20 条关键路径都来自待测信号产生模块（第一条时序裕量 3.986ns），第 21 条来自系

统时钟模块的分频（时间裕量 4.602ns），分别如图 10、11所示。总体而言，分频所经过的路径较长（计数状态需要经过多个位的寄存器）。

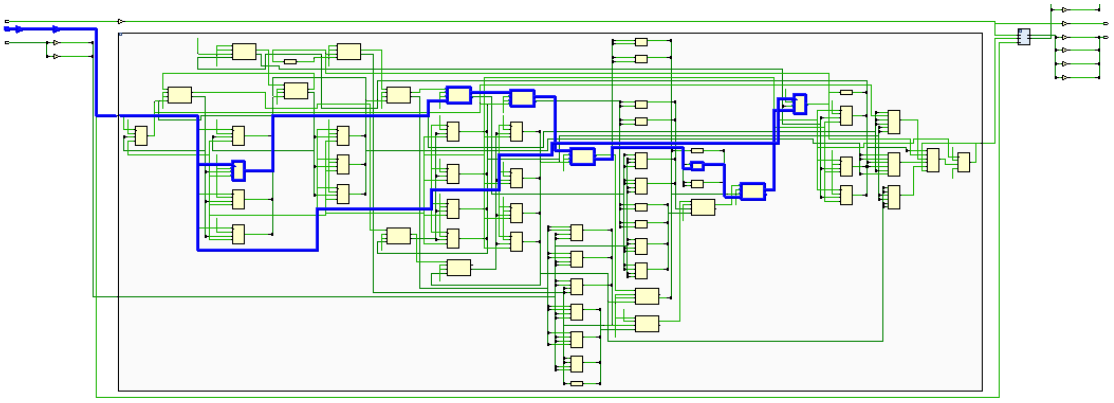

图 10: 关键路径 1

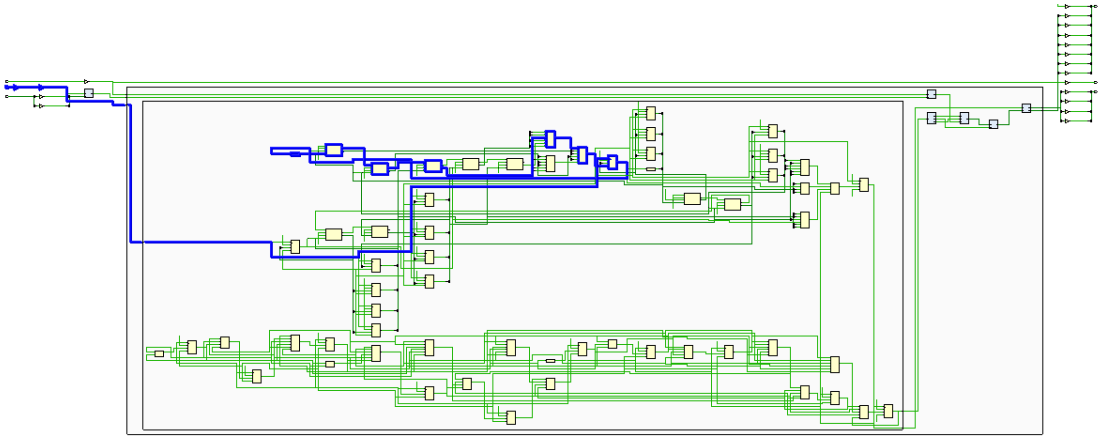

图 11: 关键路径 21