实验四　时序逻辑电路的设计

实验报告

姓名：赵文亮

学号：2016011452

班级：自 64

桌号：22

日期：2017 年 12 月 7 日

目录

1 实验目的

2 预习任务

1. 电路设计。

(1) 查阅实现电路设计所需芯片的数据手册。所需的芯片如下：

- 74HC74——双边沿触发 D 触发器（含异步置位端）。
- 74HC00——4× 二输入与非门。
- 74HC11——3× 三输入与门。
- 74HC86——4× 二输入异或门。

查阅数据手册，引脚图如图 1～4所示。

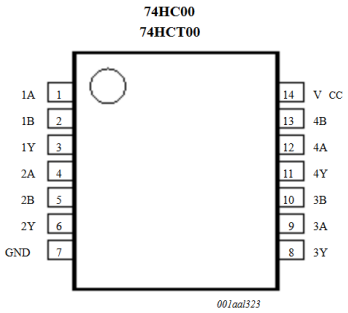

图 1: 74HC00

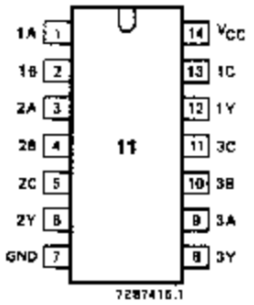

图 2: 74HC11

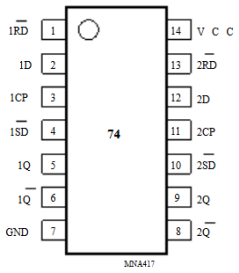

图 3: 74HC74

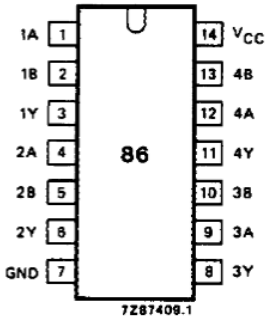

图 4: 74HC86

(2) 根据同步时序逻辑电路的设计方法，写出设计“星期显示”电路的具体步骤，如状态转换图、状态化简、方程组等。

状态转换图 分析可知，星期显示只需使用 3 个触发器。设 $Q_2Q_1Q_0$ 表示电路所处的状态，也是输出结果的后三位；Y 表示输出结果的高位。状态转换图如图 5 所示。

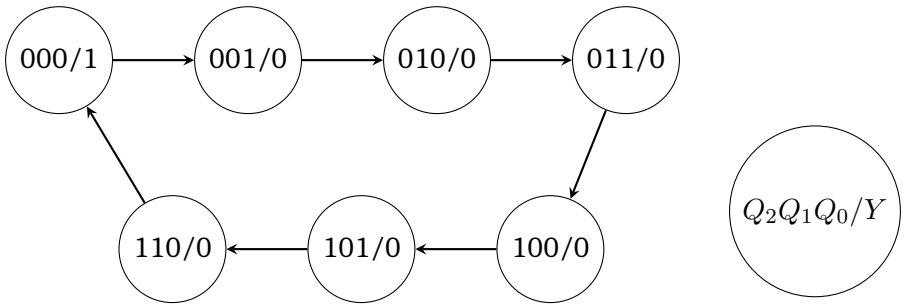

图 5: 星期显示状态转换图

表 1: 状态转换表

Q_2	Q_1	Q_0	Q_2^*	Q_1^*	Q_0^*	Y
0	0	0	0	0	1	1
0	0	1	0	1	0	0
0	1	0	0	1	1	0
0	1	1	1	0	0	0
1	0	0	1	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	0

状态转换表　根据状态转换图可得状态转换表如表 1 所示。化简得逻辑式

$$\begin{cases} Q_2^* = Q_1Q_0 + Q_2Q_1' = ((Q_1Q_0)'(Q_1'Q_2)')' \\ Q_1^* = Q_2Q_0 + Q_2'(Q_1 \oplus Q_0) = ((Q_2Q_0)'(Q_2'(Q_1 \oplus Q_0))')' \\ Q_0^* = Q_1'Q_0' + Q_2'Q_0' = Q_0'(Q_1Q_2)' \end{cases} \tag{1}$$

由于本实验中使用 D 触发器，故驱动方程为

$$\begin{cases} D_2 = Q_1Q_0 + Q_2Q_1' = ((Q_1Q_0)'(Q_1'Q_2)')' \\ D_1 = Q_2Q_0 + Q_2'(Q_1 \oplus Q_0) = ((Q_2Q_0)'(Q_2'(Q_1 \oplus Q_0))')' \\ D_0 = Q_1'Q_0' + Q_2'Q_0' = Q_0'(Q_1Q_2)' \end{cases} \tag{2}$$

将状态转换图中未出现的状态（111）代入可知，该电路可以自启动。

(3) 在面包板上搭接“星期显示”电路。

在搭接之前，我利用 Multisim 进行了仿真。电路如图 6 所示。由于我使用的 Multisim 中的的元件默认的 V_{DD} 为 6V，该电路中所有的高电平标准均为 6V。而实际搭接电路时，要以学习机上的 5V 为标准。图中左下方是为了模拟时钟信号，便于仿真。从仿真结果可以看出，该电路能实现所需的功能。

(4) 对“时间显示”电路进行模块划分，并说明各模块电路功能。

时间显示电路分为以下几个模块：

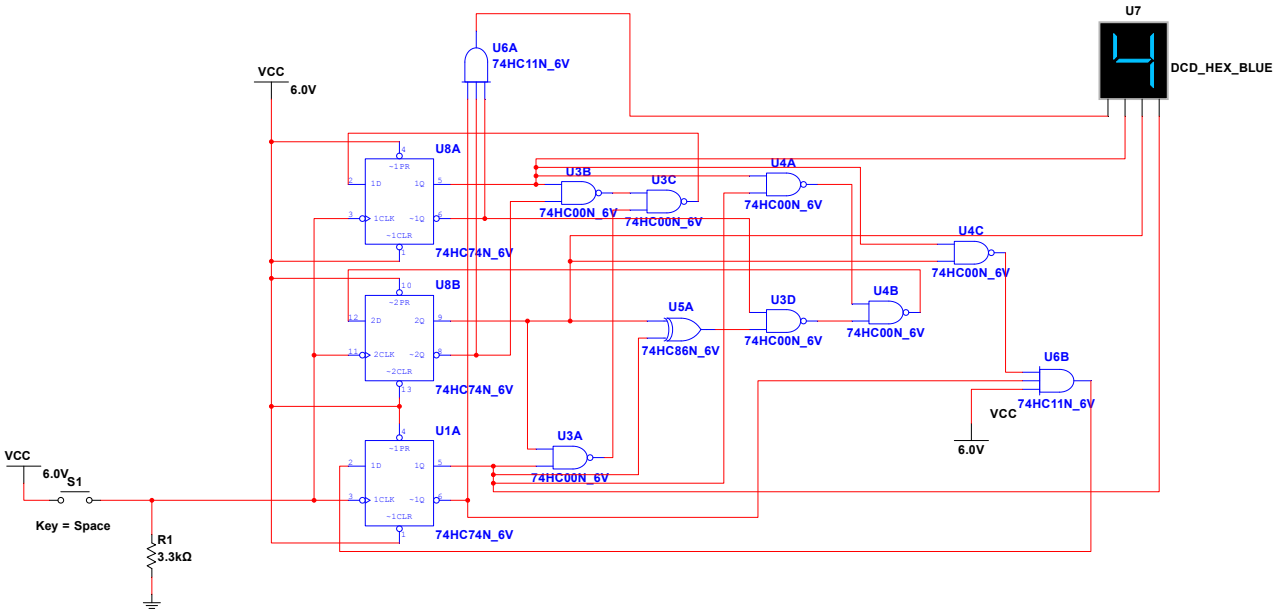

图 6: M2 电路仿真

- 60 进制计数器。用于分钟显示。输入时钟信号，输出分钟的十位和个位的二进制值，以及进位信号。此外还包括置零、报时使能、报时输出等附加端。
- 24 进制计数器。用于小时显示。输出时钟信号（来自分钟的进位），输出小时的十位和个位的二进制值，以及进位信号。
- 10 分频模块。降低时钟频率至输入的 1/10。
- 扫描地址产生模块。通过 10 分频模块将晶振的时钟分频，进而产生频率合适的扫描地址。在一个周期内遍历 00,01,10,11 四种取值，用于数码管的位选输入。
- 16 选 4 数据选择器。从四个四位二进制数（小时十位、小时个位、分钟十位、分钟个位）根据位选地址得到对应的数值。同时实现了将位选地址译码得到位选信号。
- BCD 七段显示译码器。将四位二进制转换为数码管的段选信号（直接使用 7448 实现）。
- 按键防抖模块。输入按键信号（可能会有抖动），输出经过处理后无抖动的信号。
- 2 选 1 数据选择器。用于控制分钟的时钟信号来源（按键输入或时钟脉冲）。

(5) 用 EDA 软件完成“时间显示”电路的设计输入、仿真和下载。

(6) 画出实现全部电路功能的纸版逻辑图（手绘或打印均可）。

本次实验的 M1 电路中，我使用 Verilog 编写所有模块，顶层文件使用原理图输入。顶层电路如图 7 所示。各个子模块的代码如下：

2. 由信号发生器为“时间显示”电路提供时钟脉冲，请写出该信号的电压取值范围。

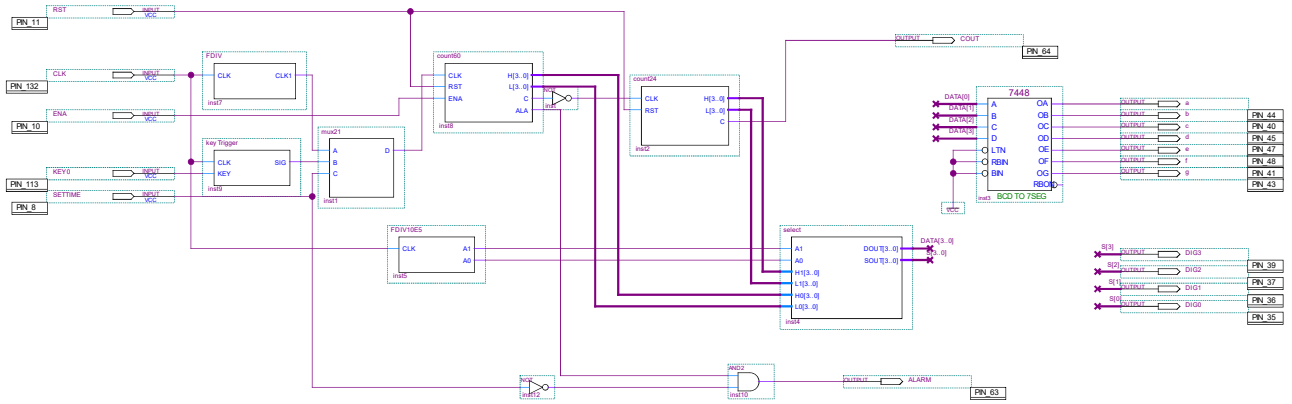

图 7: 顶层电路图

通过 Pin Planner 可以查看到引脚的 I/O 标准默认为 3.3VLVTTL。此标准下要求：

$$\begin{cases} V_{\text{cc}} = 3.3V \\ V_{\text{OH}} \geq 2.4V \\ V_{\text{OL}} \leq 0.4V \\ V_{\text{IH}} \geq 2V \\ V_{\text{IL}} \leq 0.8V \end{cases}$$

所以信号的电压应该满足：$0 \leq V_{\text{L}} \leq 0.8\text{V}, 2\text{V} \leq V_{\text{H}} \leq 3.3\text{V}$。例如可以用信号发生器产生 0∼3.3V 的方波作为时钟脉冲。

3. 写出调试电路的方法和步骤、注意事项等。

调试方法和步骤：

(1) FPGA 板和面包板用杜邦线连接，面包板上电路的输出连接到学习机上的数码管。打开学习机电源。

(2) FPGA 上电，用信号发生器产生合适的时钟脉冲，连接到 FPGA，观察输出是否符合预期。

(3) 如果出错，由于 FPGA 事先已经调试成功，需要调试面包板。利用万用表测量每个 D 触发器的输出端，获取当前状态，利用选做任务中设计的的调时功能手动给出一个进位脉冲，再测量每个 D 触发器的输出端，找到问题所在。

(4) 如果逻辑没有问题，检查芯片是否损坏。

注意事项：

(1) 一定要确保信号发生器产生正确的时钟脉冲之后再接入 FPGA。

(2) CMOS 不用的引脚要接地或接高。

3 最终电路

实验中使用的电路与设计中相差不大。M1 的顶层电路中，我加入了小时调时端 KEY1，方便改动小时产生进位信号，便于调试。顶层电路如图 8 所示。

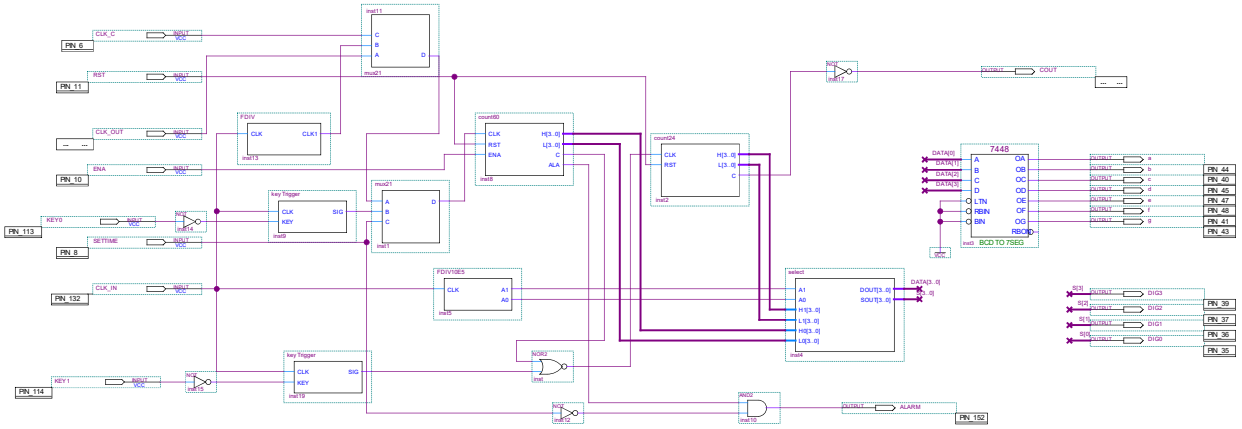

图 8: M1 最终顶层电路图

各个模块功能在预习任务中已经描述清楚。主要模块的工作原理如下：

- 60 进制计数器。内部用变量 count 记录时钟脉冲的个数。输出分钟的高位和低位直接通过 count/10 和 count% 10 得到。此外，还增加了整点报时的控制端和异步置零控制端，只需要进行简单的判断语句，不再赘述。
- 24 进制计数器。原理和 60 进制计数器相似。
- 10 分频模块。内部使用一个变量 count 记录脉冲到达的次数，当变量等于 9 时给出进位信号。
- 扫描地址产生模块。通过 10 分频模块将晶振的时钟分频得到 CLK1，再通过分别将 CLK1 进行一次和两次分频得到地址 A_0 和 A_1。
- 16 选 4 数据选择器。直接根据输入地址 A_1, A_0 使用 case 语句进行判断，将对应的数据输出。此外还根据地址的不同输出了数码管的位选控制信号（相当于顺便实现了 2-4 译码器）。
- BCD 七段显示译码器。直接使用 7448 实现。
- 按键防抖模块。使用晶振时钟采样，内部通过一个 8 位 test 数组存放采样测试数据。每次时钟上升沿到来时读取按键的状态，并移位寄存到 test 中。当 test 中的 8 位数据完全相同时，认为已经按键稳定。事实证明这样的实现非常可靠。

实验任务中使用的 M2 电路仍为图 6 所示。在实现思考题的按键防抖时，我将电路修改为图 9 。M2 电路的工作原理在预习任务中已经进行了详细的推导。

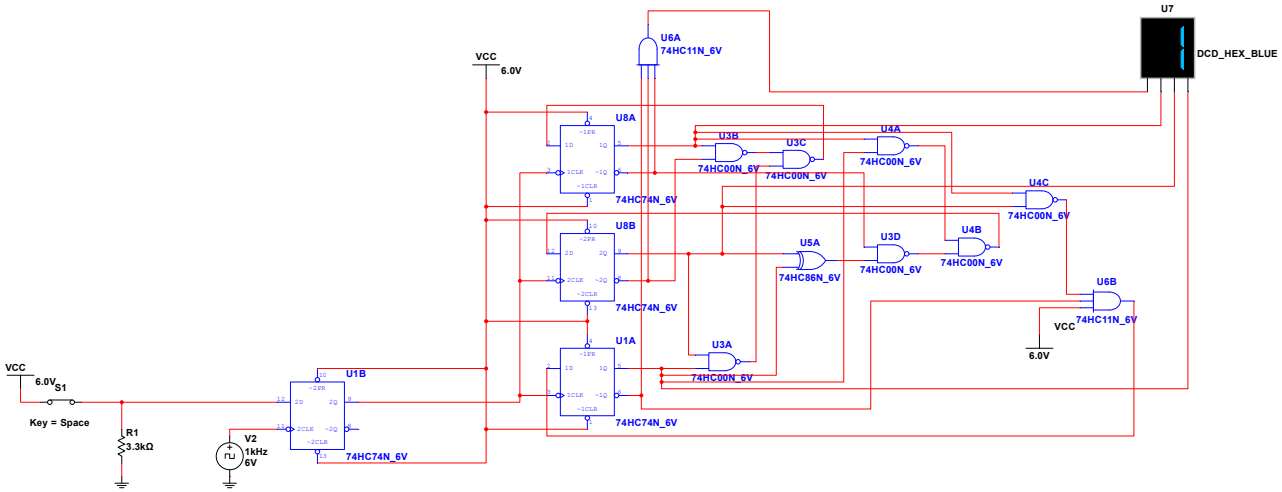

图 9: M2 最终电路图

4 实验总结

4.1 时序逻辑电路的设计和调试步骤

4.1.1 时序逻辑电路的设计步骤

时序逻辑电路的设计通常可以分为以下几步：

1. 分析问题中的状态个数，对每个状态进行编码，画出状态转换图。
2. 化简状态转换图，列出状态转换表。
3. 根据状态转换表写出状态方程和输出方程，并化简。化简时要根据已有的芯片资源适当将逻辑式转化。
4. 选择使用触发器的种类，根据触发器的特性方程，得出驱动方程。
5. 根据驱动方程和输出方程搭接电路。

4.1.2 时序电路的调试步骤

1. 实验前应该先进行仿真，确保方程推导无误。
2. 检查电路功能是否正确。搭接好电路，并提供一定频率的时钟，观察结果是否符合预期。
3. 结果不符合预期时，先检查一下是否存在导线松动、芯片损坏等问题。如果没有则应该逐步排查逻辑方面的问题。
4. 找到不符合预期的状态，用电压表测量每个触发器的输出。为电路提供一个单独的脉冲使其跳到下一个状态，再次测量每个触发器的输出。找到输出错误的触发器，检查该触发器的输入端连线是否符合推导出的驱动方程。

5. 如果状态转换仍然出错，并且每次状态转换比较随机，可能需要考虑时钟信号是否存在毛刺。

4.2 调试过程中的问题和解决方法

虽然这次实验准备的十分充分，但是实验中出现了很多之前没有想到的问题。

1. VCC 和 GND 弄混。实验开始时我先测试了 M1 电路，使用信号发生器生成脉冲信号给 FPGA 板。这就需要信号发生器和 FPGA 共地。一开始我以为扩展端口的最下面的两个是 GND，但是为了进一步确认，我查阅了引脚图，发现最上面的两个才是 GND，而最下面的两个是 VCC。幸好提前发现，否则可能对 FPGA 板造成损坏。后来我也了解到，本次实验好多同学都因为将 VCC 和 GND 接反烧坏了实验板。

2. 两个电路没有共地。虽然之前已经提前想到了这一点，但是实验一开始还没共地的时候我就期待得到预期的结果，显然没有成功。不过我马上就注意到了这一点，及时进行了修改。

3. 电路连接没有问题之后，显示还是不符合预期。虽然我之前已经进行了充分的准备，已经很认真地在面包板上搭接好了电路，这时我又不禁开始怀疑。这时我先将面包板和学习机的连线断开，准备检查上面的连线，发现没有问题。当我再次将面包板连接到学习机上，使用信号发生器产生时钟脉冲给面包板的 M2 电路时，我发现数码管显示完全符合预期。这说明可能是连接面包板和学习机时导线有松动或没插紧。之前的几次实验，我也发现导线的直径比起学习机上的插口较小，这就导致了连接可能不稳定。以后遇到类似的情况要首先考虑一下这个问题。

4. 当我将 FPGA 的进位输出作为 M2 电路的时钟信号时，又没有得到预期的结果。这时陈老师告诉我可以利用学习机上的脉冲按键产生信号来调试。于是我使用脉冲按键调试，但是结果还是不太对。后来助教告诉我可以用学习机上的时钟来接入调试。我尝试了 1Hz 的时钟，这时，显示结果非常完美。其实，我觉得用学习机上的脉冲按键和用 1Hz 的时钟调试并无本质上的区别，但是结果却不一样的原因还是导线连接的松动问题。所以我又检查并紧固了导线的连接。

5. 这时我已经可以确认面包板的电路没有问题了。但是当我连接 FPGA 的进位端和 M2 电路的时候，结果又不符合预期。我用万用表测量了 FPGA 的输出，发现确实会在正确的时间产生进位信号，但是由于 FPGA 的工作电压是 3.3V，万用表测出的电压也不过 3.2V 左右。这时我想到，可能是因为 M2 电路的供电电压是 5V，理论上由于 CMOS 的噪声容限，3.2V 足以被认为是高电平。但是如果芯片有损坏，可能就会出现无法将 3.2V 识别为高电平的现象。于是我更换了 D 触发器 74HC74，发现 FPGA 的进位输出已经可以让 M2 电路的数码管数字变动了。

6. 虽然 M2 电路的数码管数字可以变动，但是我发现每次变动的值不定。有时是加一，有时是加二。这明显是因为输出电压有毛刺。我开始试图修改电路图，但是无济于事。这时，我想到了上次实验中关于消除竞争冒险得到选做任务。我在 FPGA 进位输出端并联了一个滤波电容，再作为 M2 电路的时钟信号，圆满地解决了问题。后来我与另一位同学探讨这个问题时，他说他也遇到了毛刺的问题，老师说这可能是面包板中的噪声造成的。从实验二中可以知道，74HC 系列的噪声容限并不是完美的 $\frac{1}{2}V_{\mathrm{DD}}$，例如 74HC00 的噪声容限为 $V_{\mathrm{NH}} = 1.678\mathrm{V}, V_{\mathrm{NL}} = 1.874\mathrm{V}$。再加上 M1 电路给出的进位电压是大约 3.3V，已经很接近噪声容限允许的 $V_{\mathrm{IH(min)}}$，故电路中的噪声很容易影响电路的功能。

4.3 实验收获

本次实验中我遇到了各种各样的问题，虽然解决这些问题的过程很浪费时间，但是调试的过程也是一种收获。回想上次实验一遍就成功，收获的只有电路的设计和搭接技巧。而本次实验，我对电路中可能出现的问题有了更多了了解，在调试的过程中总结了很多经验。

当然，我们还是希望问题越少越好。当经验积累到一定程度后，我们还是希望能够尽可能地避免出现问题；反之一旦出现问题，最好能做到根据经验快速找到解决的办法。

5 思考题

1. **基于“时间显示”电路的设计和实现，请说明采用同步或异步时序电路的设计方式分别会产生哪些问题？如何解决？**

   答：同步电路在分钟到小时的进位处理上比较困难。可以在小时计数模块添加使能端，并将分钟计数模块的进位输出连接到小时计数模块的使能端上，即可处理进位。

   异步电路逻辑上较为符合直观感受，但是可能会出现竞争——冒险现象。例如可能由于设计不当，分钟的进位输出中存在毛刺，会直接导致小时显示不正确。可以通过修改逻辑式的方法避免竞争冒险现象，或采用一个滤波模块（例如已经设计的按键防抖模块）或外接滤波电容来解决。

2. **若选用学习机上的“START”按键作为 M2 电路的手动时钟输入，即按动 1 次星期显示数字加一，按动 1 次按键也会显示跳动多个数字的现象。请设计电路予以解决，并说明电路参数的选择依据（可以通过实验验证）。**

   设计电路如图 9 左下角所示。只需通过一个 D 触发器和采样时钟即可实现消抖。采样时钟频率的选择不能过大，否则会导致消抖信号不好；也不能过小，否则会导致反应不灵敏。选择的标准应该是，时钟的周期要大于按键前后抖动的时间，并小于人的按动时长。按键抖动的时间一般为 5∼10 ms。本实验中采用了 10Hz 的时钟进行采样。经验证效果很好。后来与老师交流时老师说，这是一种很成熟的做法。

A 子模块代码

- 60 进制计数器:

```
module count60(CLK, H, L, C, RST, ENA, ALA);
        input CLK;
        input RST;
        input ENA;
        output reg C, ALA;
        output reg [3:0] H;
        output reg [3:0] L;
        reg [5:0] count;
        always @ (posedge CLK or posedge RST)
        begin
                if (RST)
                        count <= 5'b0;
                else if (count < 59)
                        count <= count + 5'b1;
                else
                   count <= 5'b0;
        end
        always @ (count or ENA)
        begin
                 H = count / 10;
                 L = count % 10;
                if (count == 59)
                 C  = 1'b1;
                else
            C = 1'b0;
                if (ENA && count >= 54)
                        ALA = 1;
                else
                        ALA = 0;
        end
endmodule
```

- 24 进制计数器:

```
module count24(CLK, H, L, C, RST);
        input CLK;
```

```
3         input RST;
4         output C;
5         reg C;
6         output reg [3:0] H;
7         output reg [3:0] L;
8         reg [4:0] count;
9         always @ (posedge CLK or posedge RST)
10        begin
11                if (RST)
12                        count <= 5'b0;
13                else if (count < 23)
14                        count <= count + 5'b1;
15                else
16                   count <= 5'b0;
17        end
18        always @ (count)
19        begin
20                 H <= count / 10;
21                 L <= count % 10;
22                if (count == 23)
23                 C <= 1'b1;
24                else
25            C <= 1'b0;
26        end
27  endmodule
```

- 16 选 4 数据选择器:

```
1   module select(A1, A0, H1, L1, H0, L0, DOUT, SOUT);
2         input A1, A0;
3         input [3:0] H1, H0;
4         input [3:0] L1, L0;
5         output [3:0] DOUT;
6         output [3:0] SOUT;
7         reg [3:0] DOUT;
8         reg [3:0] SOUT;
9         always @ (A1 or A0 or H1 or L1 or H0 or L0)
10        begin
11                case ({A1, A0})
```

```
12                          2'b00 :
13                          begin
14                                  DOUT <= L0;
15                                  SOUT <= 4'b0001;
16                          end
17                          2'b01 :
18                          begin
19                                  DOUT <= H0;
20                                  SOUT <= 4'b0010;
21                          end
22                          2'b10 :
23                          begin
24                                  DOUT <= L1;
25                                  SOUT <= 4'b0100;
26                          end
27                          2'b11 :
28                          begin
29                                  DOUT <= H1;
30                                  SOUT <= 4'b1000;
31                          end
32                          default:
33                          begin
34                                  DOUT = 4'b1111;
35                                  SOUT = 4'b0000;
36                          end
37                  endcase
38          end
39  endmodule
```

- 2 选 1 数据选择器:

```
1  module mux21(A, B, C, D);
2          input A, B, C;
3          output D;
4          assign D = C ? B :A;
5  endmodule
```

- 按键防抖模块:

```
1  module keyTrigger(CLK, KEY, SIG);
```

```
2         input KEY;
3         input CLK;
4         output reg SIG;
5         reg [7:0] test;
6         always @ (posedge CLK)
7         begin
8         if (CLK)
9         begin
10                test[6:0] <= test[7:1];
11                test[7] <= KEY;
12                if (test == 8'b11111111)
13                        SIG <= 1;
14                else if (test == 8'b00000000)
15                        SIG <= 0;
16        end
17        end
18 endmodule
```

- 10 分频模块:

```
1  module FDIV10 (CLK, CLK1);
2         input CLK;
3         output CLK1;
4         reg [3:0] count;
5         reg FULL;
6         always @ (posedge CLK)
7         begin
8         if (count == 4'b1001)
9         begin
10                count <= 4'b000;
11                FULL <= 1;
12        end
13        else
14                FULL <= 0;
15                count <= count + 4'b1;
16        end
17        assign CLK1 = FULL;
18 endmodule
```

- 扫描地址产生模块:

```
module FDIV10E5(CLK, A1, A0);
        input CLK;
        output A1, A0;
        reg A1;
        reg A0;
        wire CLK1;
        wire [2:0] temp;
        FDIV10 div_1(CLK, temp[0]);
        FDIV10 div_2(temp[0], temp[1]);
        FDIV10 div_3(temp[1], temp[2]);
        FDIV10 div_4(temp[2], CLK1);
        always @ (posedge CLK1)
        begin
                A0 <= ~A0;
        end
        always @ (posedge A0)
        begin
                A1 <= ~A1;
        end
endmodule
```