# 浙江大学实验报告

课程名称：数字系统　　任课老师：李宇波

实验名称：FSM design – Display a character string　　实验日期：2023/5/12

## 1 实验目的和要求

### 1.1 实验目的

To learn how to implement displaying a character string circularly by using FSM.

了解如何用有限状态机循环呈现字符串。

### 1.2 实验要求

1） Synthesize a FSM circuit (i.e., draw the complete FSM circuit) to display the character string in one 7-segment LED circularly at a rate of 0.5 Hz for each digital. Blank the display (i.e., turn off all the segments of the LED) for the spaces in the string. Assume that your main clock frequency is 100MHz. Write the Verilog code for your design.

综合一个有限状态机电路，在七段数码管中以 0.5Hz 呈现字符串。字符串的空格以全灭显示。假定主要时钟周期是 100MHz，为你的设计编写 Verilog 代码。

2） Implement the above circuit in your Basys3 board.

在 Basys3 板上完成电路。

3） Display the above character string in four 7-segment LED circularly at a rate of 0.5 Hz in your Basys3 board.

呈现字符串。

## 2 状态设计和描述

1） 设计状态图和状态编码如下：

| 状态 | 数码管显示(“.”代表该数码管全灭) |
|---|---|
| 0（4’b0000） | …2 |
| 1（4’b0001） | ..20 |
| 2（4’b0010） | .202 |
| 3（4’b0011） | 2023 |

| | |
|---|---|
| 4（4’b0100） | 0230 |
| 5（4’b0101） | 2305 |
| 6（4’b0110） | 3050 |
| 7（4’b0111） | 0505 |
| 8（4’b1000） | 505. |
| 9（4’b1001） | 05.. |
| 10（4’b1010） | 5… |
| 11（4’b1011） | …. |

从状态 1 到状态 11，可以完成日期滚动的循环。

2） 数码管显示设计

Fpga 设计数码管一次只能点亮一个，利用视觉暂留效应可以在人眼上呈现四个数码管同时点亮的效果。视觉暂留的间隔大概在 50ms 到 100ms 左右，实验中选取 250ms。

## 3 行为级模型代码

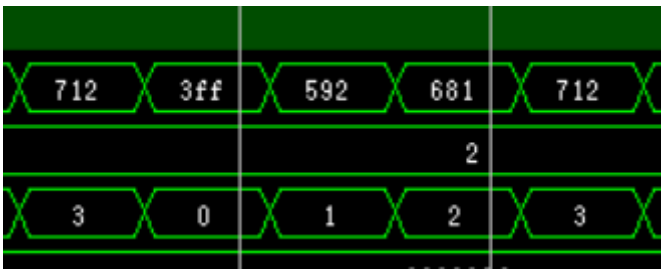
行为级模型代码设计分为顶层模块、状态机模块、译码处理模块和分频器模块。完整代码见附录 1 。

## 4 行为级仿真

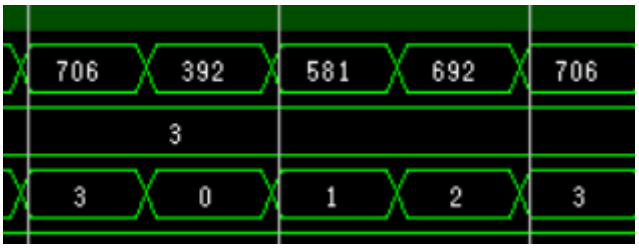
编写仿真代码（见附录 2），模拟实际的日期滚动。为了方便仿真测试，把参数 CNT_2500US 改成了 250，即 250 个时钟周期；CNT_600mS 改成 60000，即 60000 个时钟周期。

状态 1 四个数码管：第一行为十六进制值，第三行为数码管编号（从左到右）。

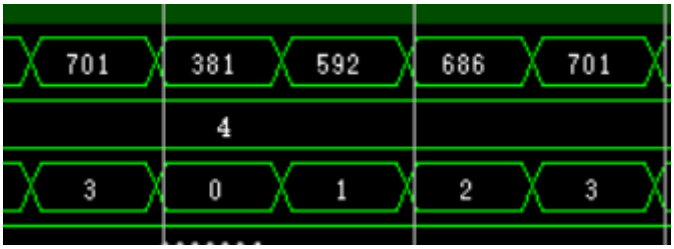

状态 2 四个数码管：第一行为十六进制值，第三行为数码管编号（从左到右）。



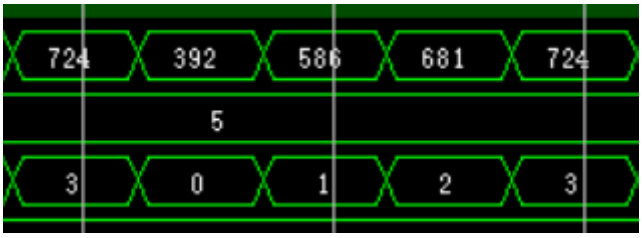

状态 3 四个数码管：第一行为十六进制值，第三行为数码管编号（从左到右）。



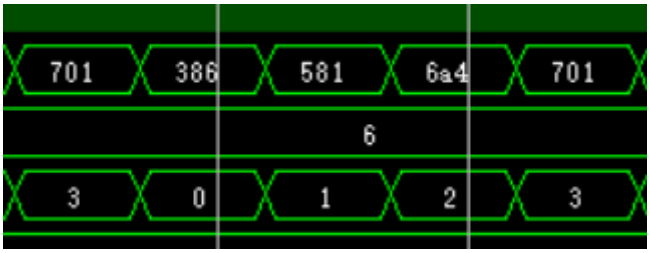

状态 4 四个数码管：第一行为十六进制值，第三行为数码管编号（从左到右）。



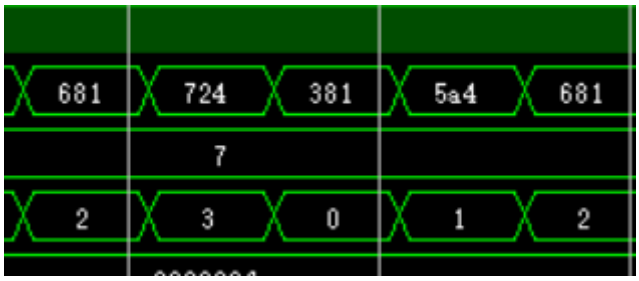

状态 5 四个数码管：第一行为十六进制值，第三行为数码管编号（从左到右）。



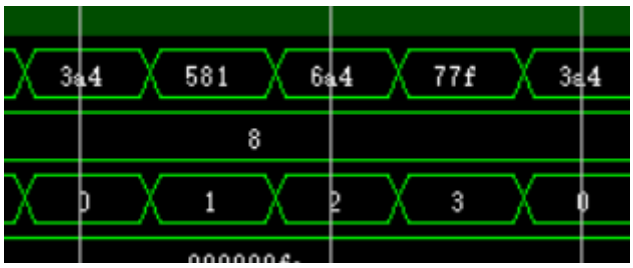

状态 6 四个数码管：第一行为十六进制值，第三行为数码管编号（从左到右）。




状态 7 四个数码管：第一行为十六进制值，第三行为数码管编号（从左到右）。

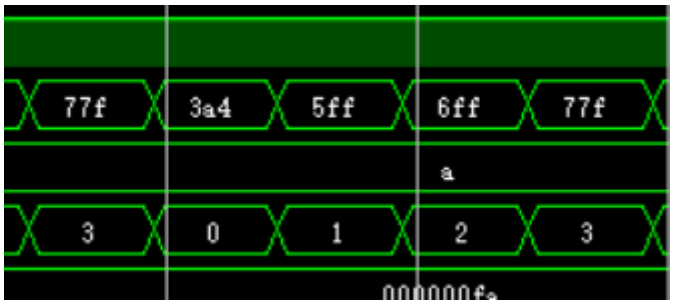



状态 8 四个数码管：第一行为十六进制值，第三行为数码管编号（从左到右）。

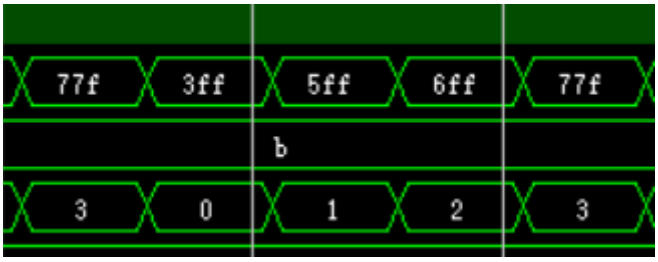



状态 9 四个数码管：第一行为十六进制值，第三行为数码管编号（从左到右）。


状态 10 四个数码管：第一行为十六进制值，第三行为数码管编号（从左到右）。




状态 11 四个数码管：第一行为十六进制值，第三行为数码管编号（从左到右）。




综合以上仿真结果，认为设计无误。

# 附录 1

1. 顶层模块

```
module Rolling(
    input clk,
    input rst_n,
    output [10:0] LED
);
parameter CNT_2500US = 250000;
parameter CNT_600mS = 60000000;
//parameter CNT_2500US = 250; // test
//parameter CNT_600mS = 60000;// test
wire pulse400Hz, pulse4Hz;
wire [3:0] state;

divider #(.CNT_N(CNT_2500US))
dc1(.clk(clk), .divided_clock(pulse400Hz));
divider #(.CNT_N(CNT_600mS)) dc2(.clk(clk), .divided_clock(pulse4Hz));
FSM fsm1(.clk(pulse4Hz), .en(1), .rst_n(rst_n), .state(state));
led_decode
led1(.clk(pulse400Hz), .rst_n(rst_n), .state(state), .led_out(LED));

endmodule
```

2. 分频器模块

```
module divider(
    input clk,
    output wire divided_clock
);
parameter CNT_N = 2;
reg [26:0] count = 0;
reg flag = 0;

always @(posedge clk) begin
    if(count == CNT_N - 1)begin
        flag = ~flag;
        count = 0;
    end
    count <= count + 1;
end
```

```
assign divided_clock = flag;

endmodule
```

3. 状态机模块

```
module FSM (
    input en,
    input clk,
    input rst_n,
    output reg [3:0] state
);
reg [3:0] state_next = 0;

reg [26:0] count = 0;

always @(posedge clk) begin
    if(!rst_n)
        state_next = 0;
    else if(en) begin
        if(state_next == 4'b1011)
            state_next = 0;
        else
            state_next = state_next + 1;
    end
end

always @(posedge clk) begin
        state = state_next;
end

endmodule
```

4. 译码模块

```
module led_decode(
    input clk,
    input rst_n,
    input signed [3:0] state,
    output reg [10:0] led_out
);
reg [2:0] count = 0;
//count for displaying each digit
```

```
always @(posedge clk) begin
    if(count == 3)
        count = 0;
    else count = count + 1;
end
reg [3:0] led [0:3]; reg [6:0] num [0:4];
initial begin
    led[0] <= 4'b0111;
    led[1] <= 4'b1011;
    led[2] <= 4'b1101;
    led[3] <= 4'b1110;

    num[0] <= 7'b0000001; //0
    num[1] <= 7'b0010010; //2
    num[2] <= 7'b0000110; //3
    num[3] <= 7'b0100100; //5
    num[4] <= 7'b1111111; //none
end

always @(posedge clk)
    if(!rst_n)
        led_out = 11'b1111_1111111;
    else
    case(state)
        4'b0000: led_out <= {led[count], num[count==3?1:4]}; //...2
        4'b0001: led_out <= {led[count],
num[count==3?0:(count==2?1:4)]}; //..20
        4'b0010: led_out <= {led[count],
num[count==3?1:(count==2?0:(count==1?1:4))]}; //.202
        4'b0011: led_out <= {led[count],
num[count==3?2:(count==2?1:(count==1?0:1))]}; //2023
        4'b0100: led_out <= {led[count],
num[count==3?0:(count==2?2:(count==1?1:0))]}; //0230
        4'b0101: led_out <= {led[count],
num[count==3?3:(count==2?0:(count==1?2:1))]}; //2305
        4'b0110: led_out <= {led[count],
num[count==3?0:(count==2?3:(count==1?0:2))]}; //3050
        4'b0111: led_out <= {led[count],
num[count==3?3:(count==2?0:(count==1?3:0))]}; //0505
        4'b1000: led_out <= {led[count],
num[count==0?3:(count==1?0:(count==2?3:4))]}; //505.
        4'b1001: led_out <= {led[count],
num[count==0?0:(count==1?3:4)]}; //05..
        4'b1010: led_out <= {led[count], num[count==0?3:4]}; //5...
```

```
        4'b1011: led_out <= {led[count], num[4]}; //....
    endcase

endmodule
```

# 附录 2

仿真代码（没有输入参数的改变）

```
//~ `New testbench
`timescale  1ns / 1ps

module tb_Rolling;

// Rolling Parameters
parameter PERIOD      = 10   ;
parameter CNT_2500US  = 250  ;
parameter CNT_500mS   = 50000;


// Rolling Inputs
reg   clk                                  = 0 ;
reg   rst_n                                = 0 ;

// Rolling Outputs
wire  [10:0]  LED                          ;


initial
begin
    forever #(PERIOD/2)  clk=~clk;
end

initial
begin
    #(PERIOD*2) rst_n  =  1;
end

Rolling #(
    .CNT_2500US ( CNT_2500US ),
    .CNT_500mS  ( CNT_500mS  ))
 u_Rolling (
    .clk                    ( clk          ),
    .rst_n                  ( rst_n        ),

    .LED                    ( LED    [10:0] )
);

initial
```

```
begin
    //start automatically
    $finish;
end

endmodule
```