LCD1602液晶模块编程

例：LCD1602显示

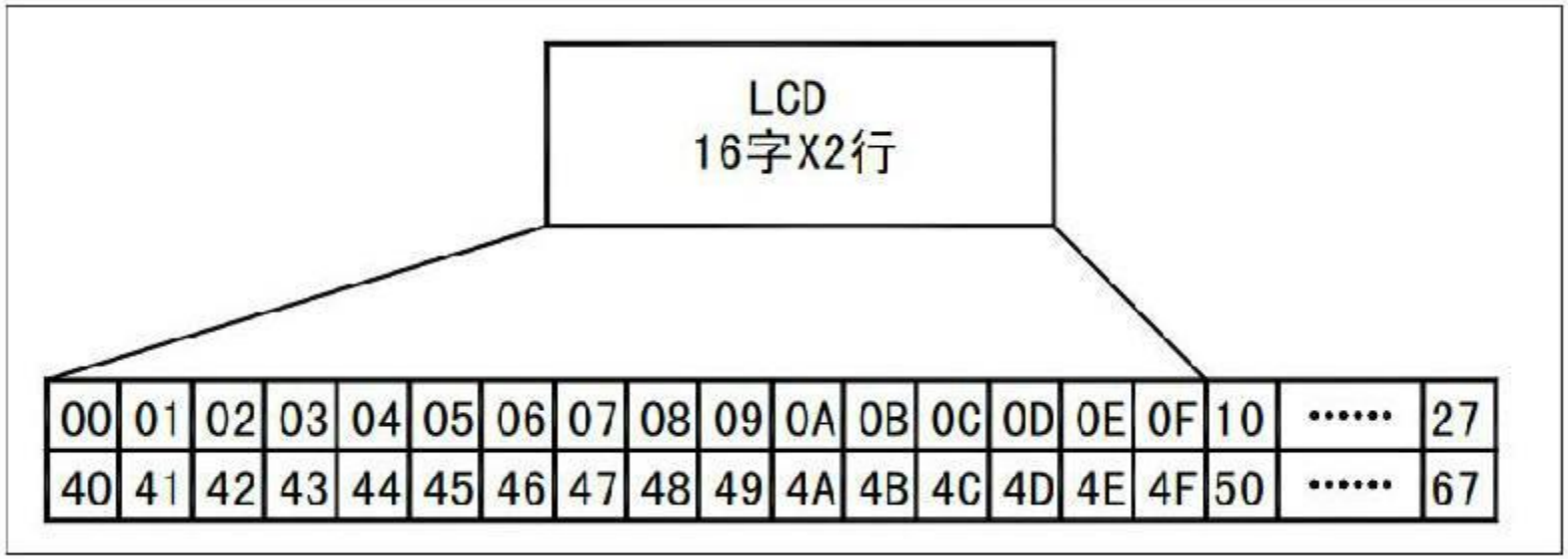
- 功能

LCD模块接口信号

编号	符号	引脚说明	编号	符号	引脚说明
1	VSS	电源地	9	D2	Data I/O
2	VDD	电源正极	10	D3	Data I/O
3	VL	液晶显示偏压信号	11	D4	Data I/O
4	RS	数据/命令选择端（H/L）	12	D5	Data I/O
5	R/W	读/写选择端（H/L）	13	D6	Data I/O
6	E	使能信号	14	D7	Data I/O
7	D0	Data I/O	15	BLA	背光源正极
8	D1	Data I/O	16	BLK	背光源负极

序号	指令	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	清显示	0	0	0	0	0	0	0	0	0	1
2	光标返回	0	0	0	0	0	0	0	0	1	*
3	置输入模式	0	0	0	0	0	0	0	1	I/D	S
4	显示开/关控制	0	0	0	0	0	0	1	D	C	B
5	光标或字符移位	0	0	0	0	0	1	S/C	R/L	*	*
6	置功能	0	0	0	0	1	DL	N	F	*	*
7	置字符发生存贮器地址	0	0	0	1	字符发生存贮器地址					
8	置数据存贮器地址	0	0	1	显示数据存贮器地址						
9	读忙标志或地址	0	1	BF	计数器地址						
10	写数到CGRAM或DDRAM）	1	0	要写的数据内容							
11	从CGRAM或DDRAM读数	1	1	读出的数据内容							

字符位置与数据地址

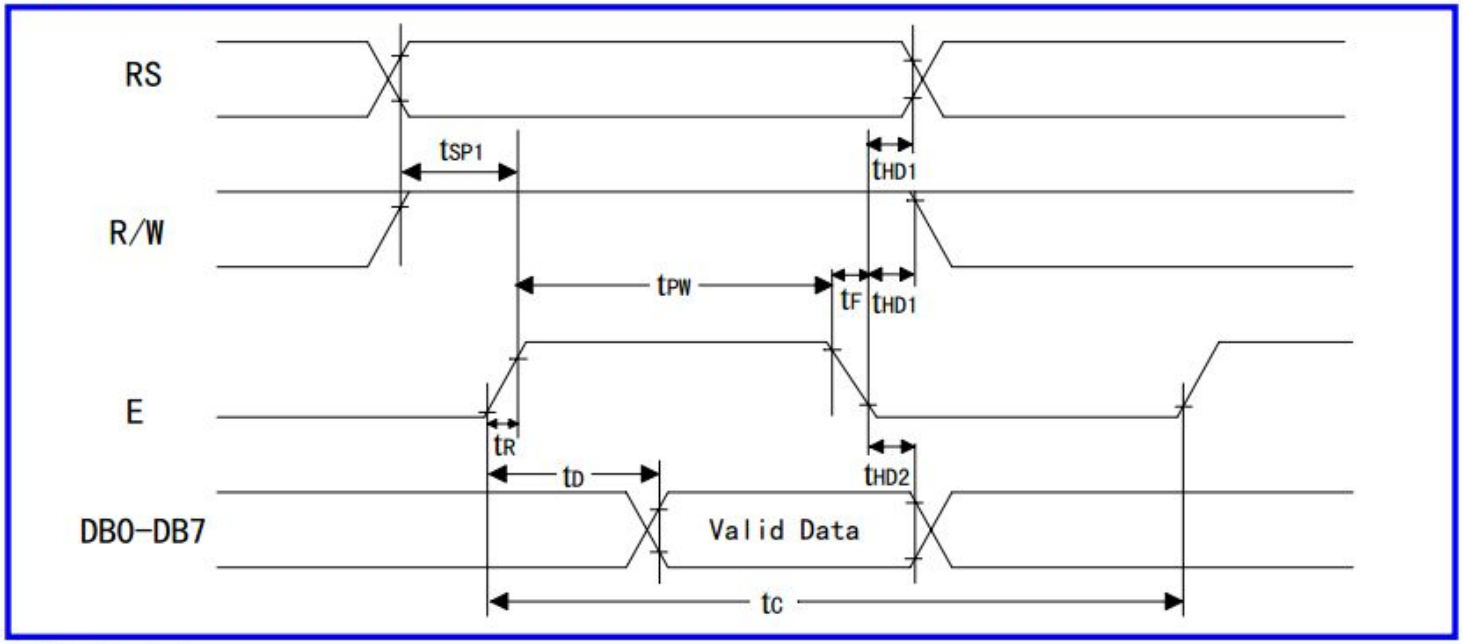

LCD
16字X2行
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 …… 27
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 …… 67

字符编码

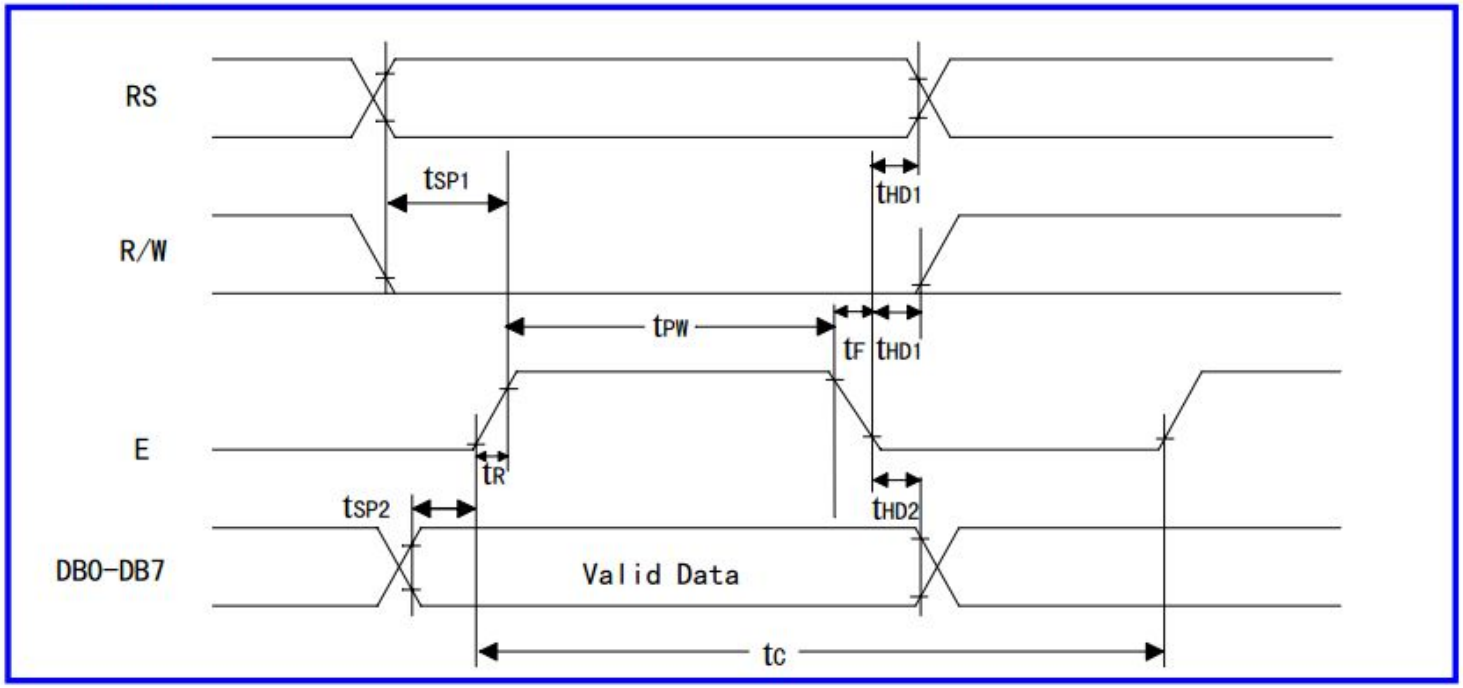
- 基于ASCII
- 无控制码
- 扩充日文和希腊字符

基本时序

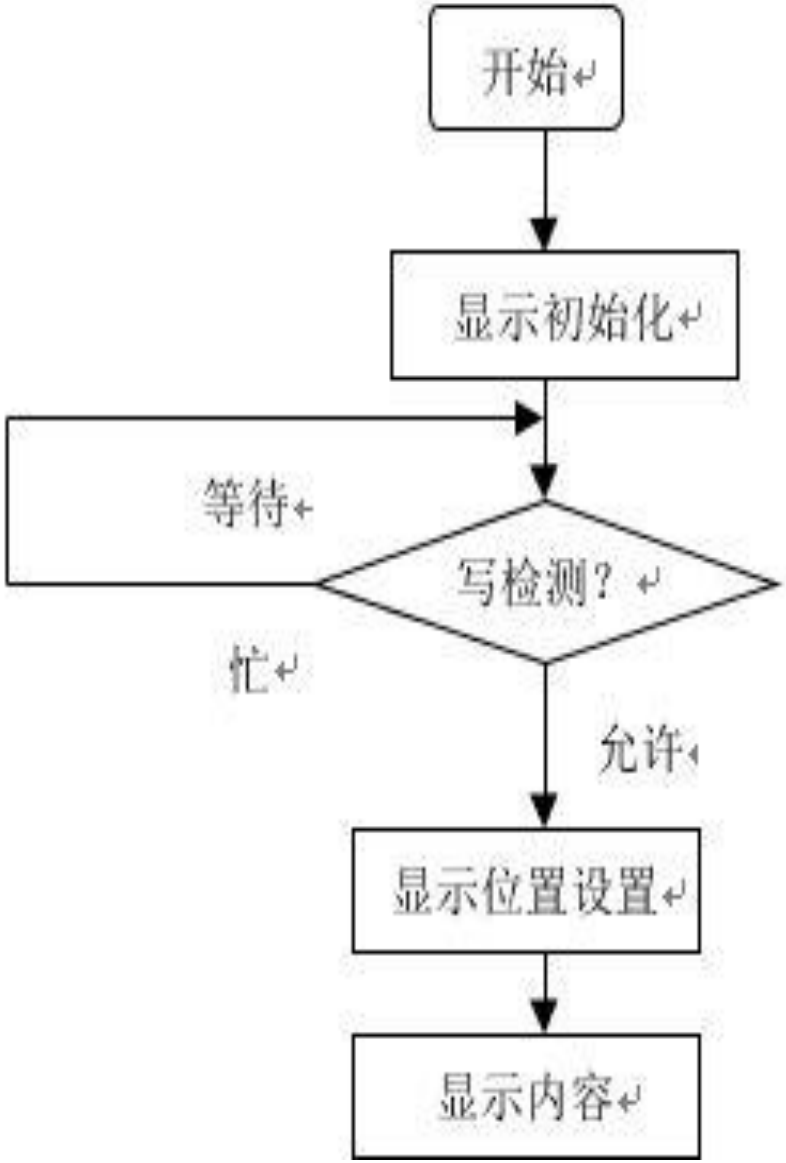
1. 读操作时序



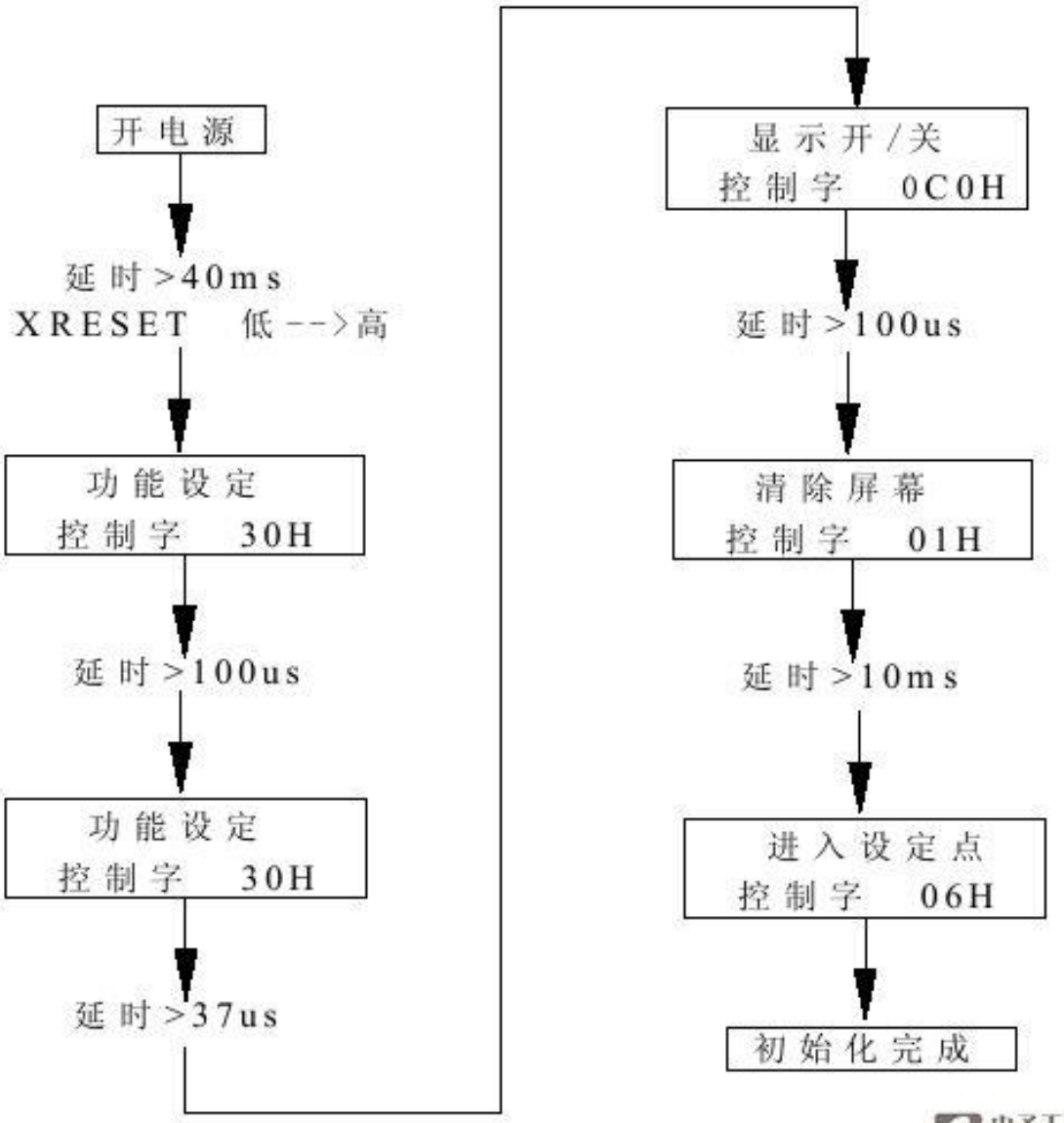
2. 写操作时序



基本工作流程



5. 初始化流程



并行执行与串行操作

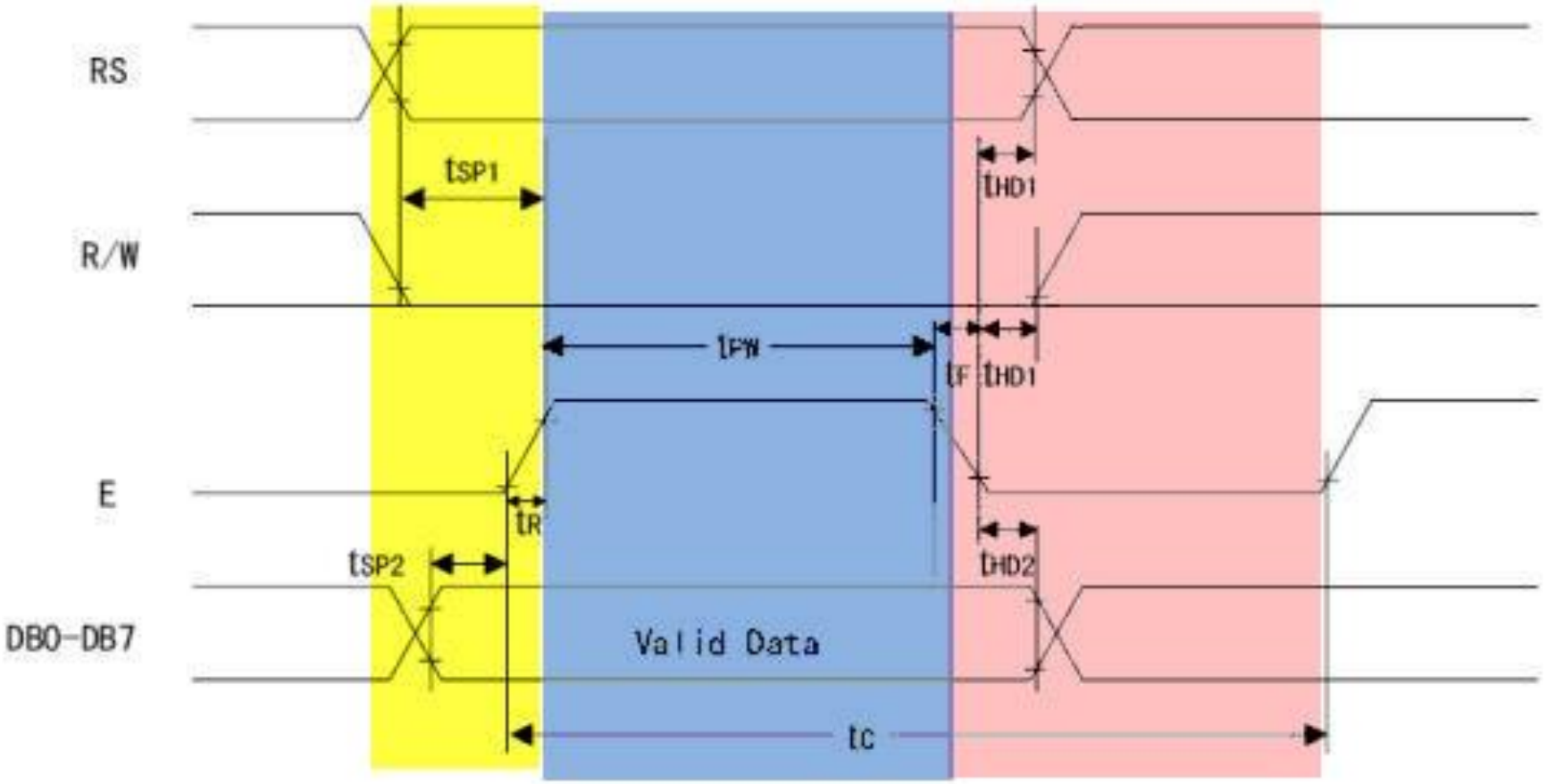
- Verilog中的行为语句之间是并行执行的关系
- LCD模块的操作命令是顺序执行的

- 问题：在HDL中如何实现LCD操作？

解决方法：状态机

- 状态机的状态之间转移是顺序的
- 利用状态机可以实现特定时序

LCD1602的基本写操作分为3阶段：



1. 发命令，准备数据，1ms
2. 写数据，8ms
3. 等待完成，180ms（足够完成，避免查询忙状态）

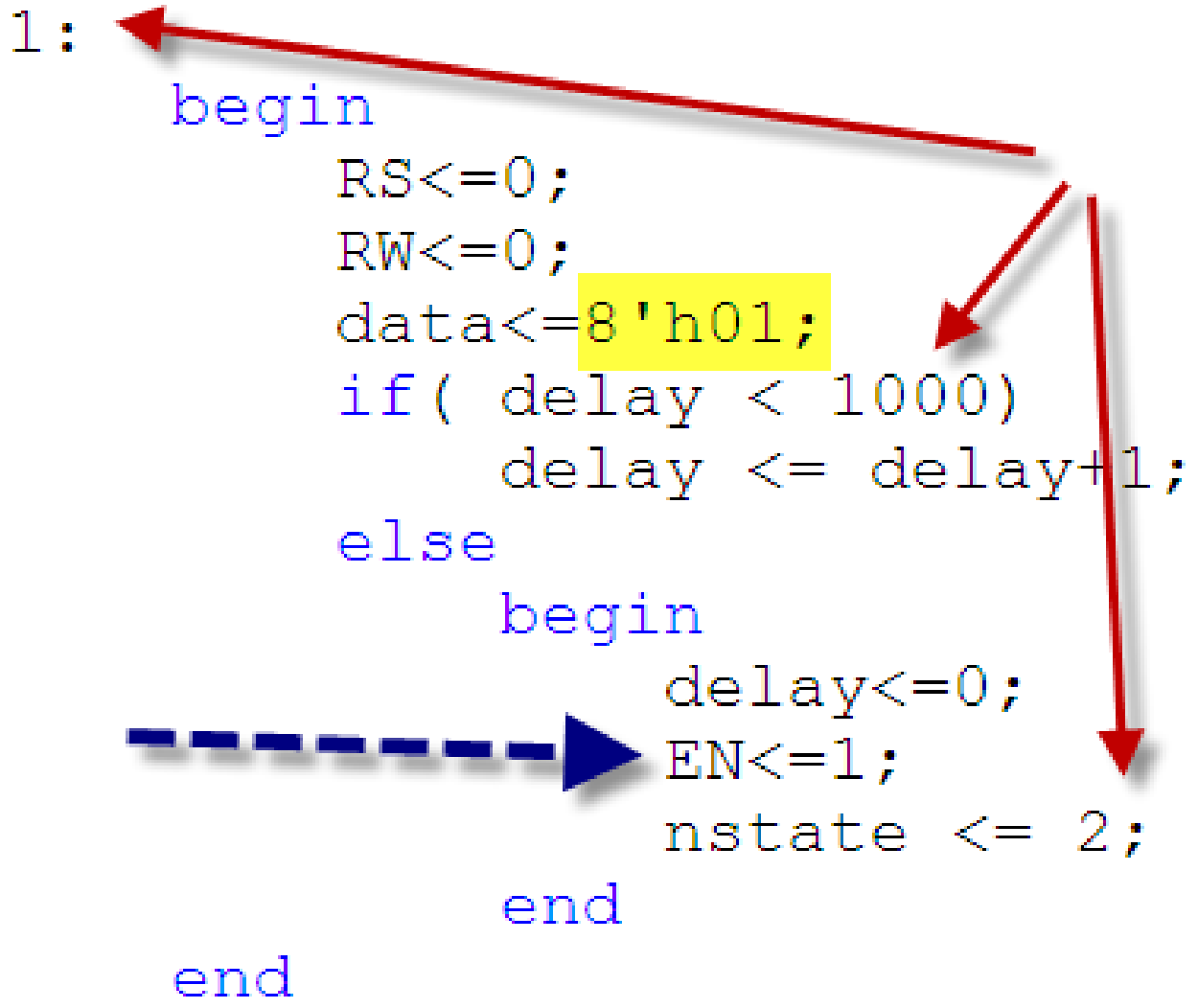
如何确定持续时间（定时）？

- 对已知时钟脉冲计数

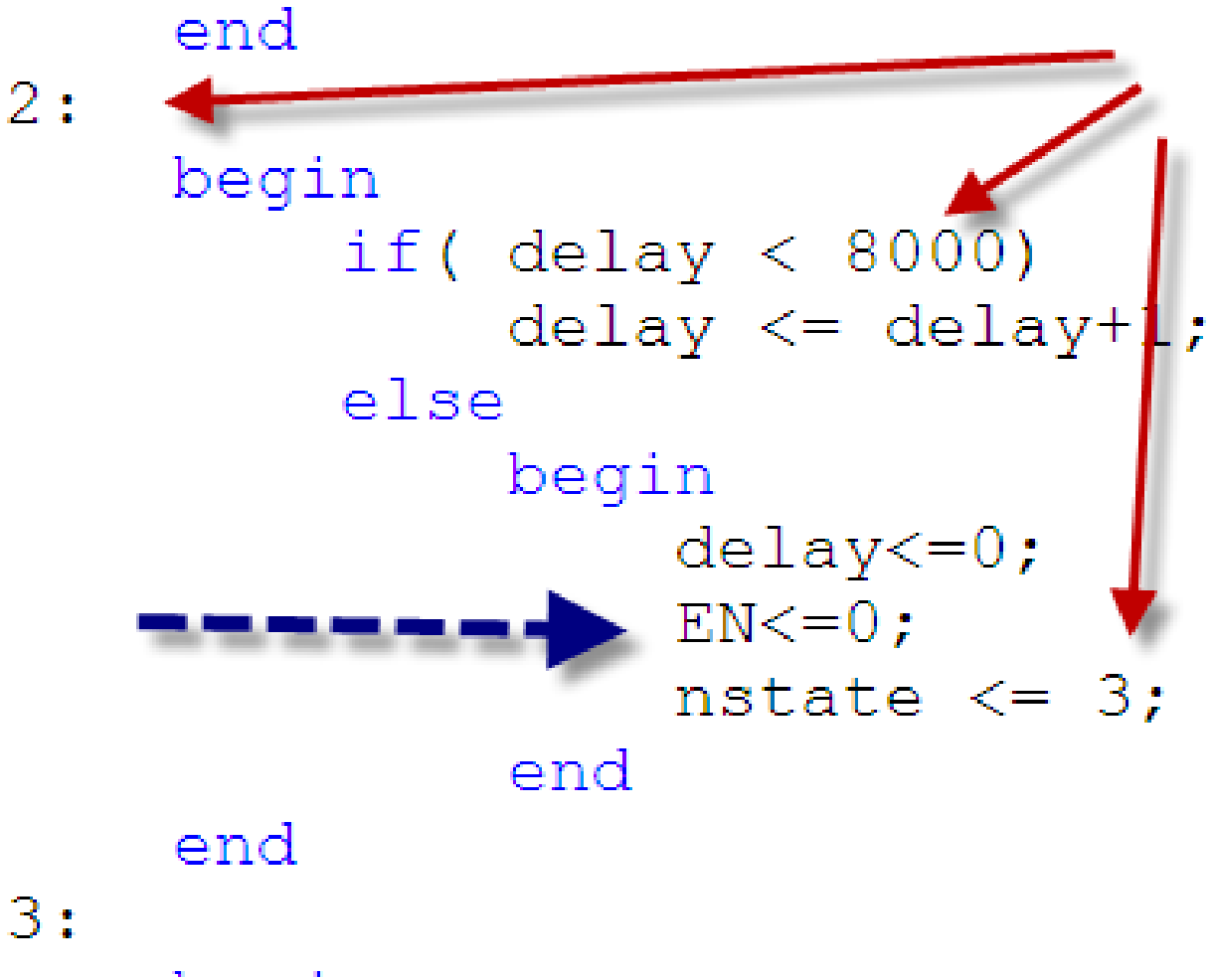
```
                begin
                    delay <=0;
                    nstate <= 1;
                end
        end
1:
        begin
            RS<=0;
            RW<=0;
            data<=8'h01;
            if( delay < 1000)
                delay <= delay+1;
            else
                begin
                    delay<=0;
                    EN<=1;
                    nstate <= 2;
                end
        end
```

一个写操作由3个状态完成

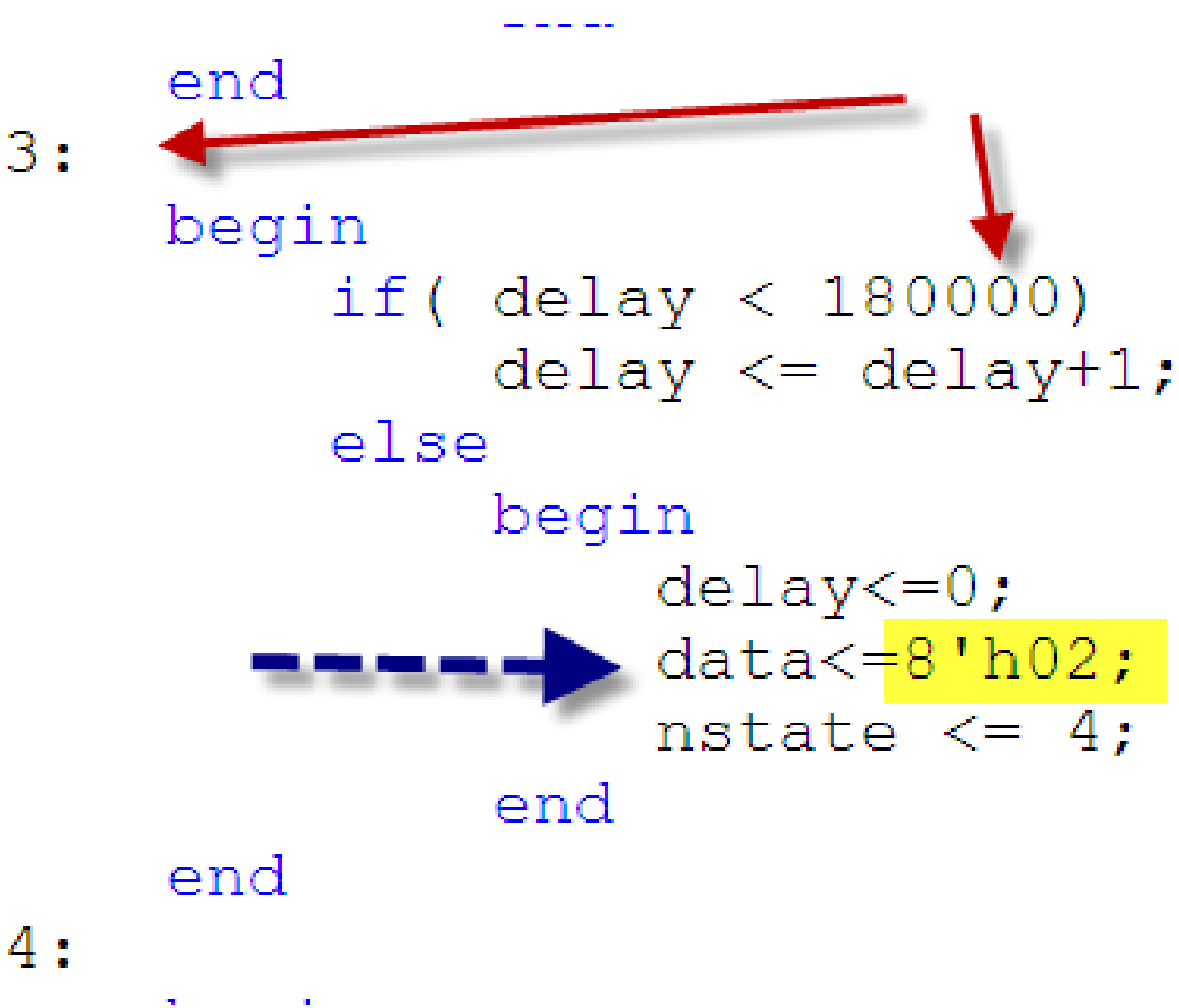
1. 发命令，准备数据，1ms



2. 写数据，
8ms


end
2:
begin
if(delay < 8000)
delay <= delay+1;
else
begin
delay<=0;
EN<=0;
nstate <= 3;
end
end
3:

3.等待完成
180ms


end
3:
begin
if(delay < 180000)
delay <= delay+1;
else
begin
delay<=0;
data<=8'h02;
nstate <= 4;
end
end
4:

写一个字符A的流程

复位等待—>发01h命令清屏幕

—>发02h命令

—>发06h命令

—>发0ch命令

—>发14h命令

—>发38h命令

—>发地址80h＋00

—>发数据”A”

—>自陷循环

顶层模块定义了LCD引脚

```
module lcd1602ak(
        input clk20m,
        input reset,
        output reg[7:0] data,
        output reg RS,
        output reg RW,
        output reg EN
                    );
```

由20MHz时钟分频得到1MHz时钟

```
reg[6:0] cnt20;
reg[23:0] delay;
reg clk1m;

always@(posedge clk20m)
      if( cnt20 < 10 )
            cnt20<=cnt20+1;
      else
            begin
                  cnt20<=0;
                  clk1m <= ~clk1m;
            end
```

由于需要定时，状态机由时钟驱动

```
reg[6:0] lcd_state, nstate;
always@(posedge clklm)
    if(!reset)
        lcd_state<= 0;
    else
        lcd_state <= nstate;

always@(posedge clklm)
    begin
        case( lcd_state )
            0:
                begin
                    if( delay
                        delay
```

其后就依次发命令

- 每个命令有3个阶段，对应3个状态
- 状态按顺序转移

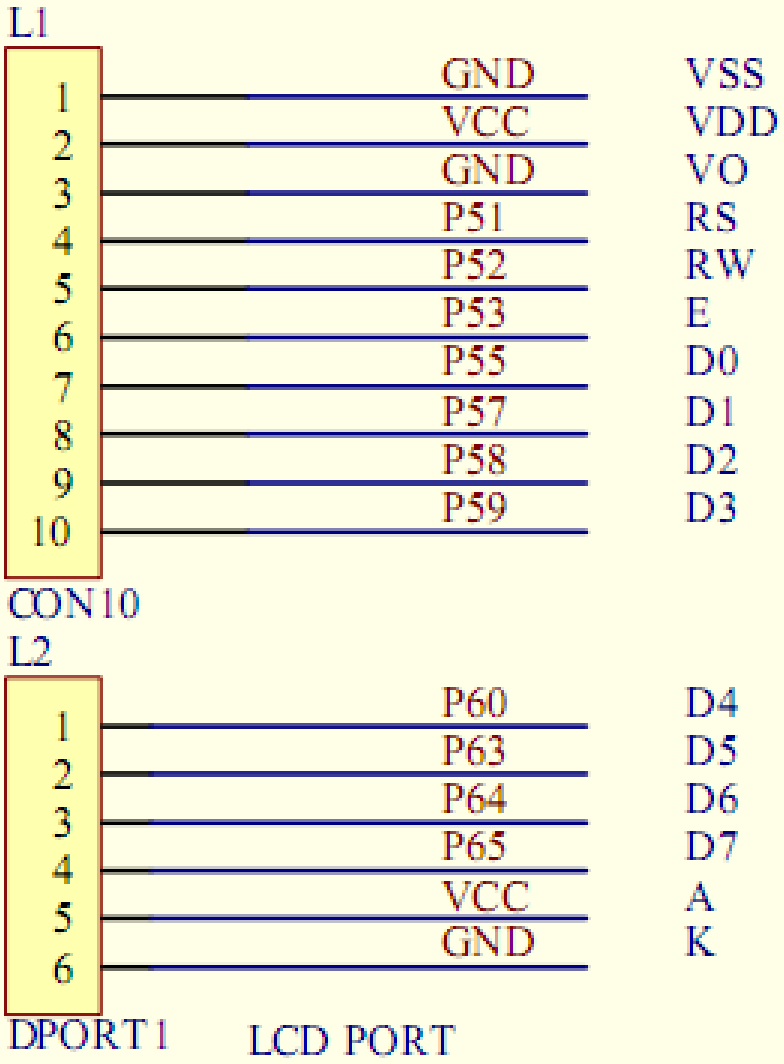
- 注意发数据时RS=1，地址要加80或c0

康新实验板引脚定义

20M时钟：Pin_17

自定义复位开关：

Pin_114



按照以上步骤，在LCD1602上实现显示单个/多个字符

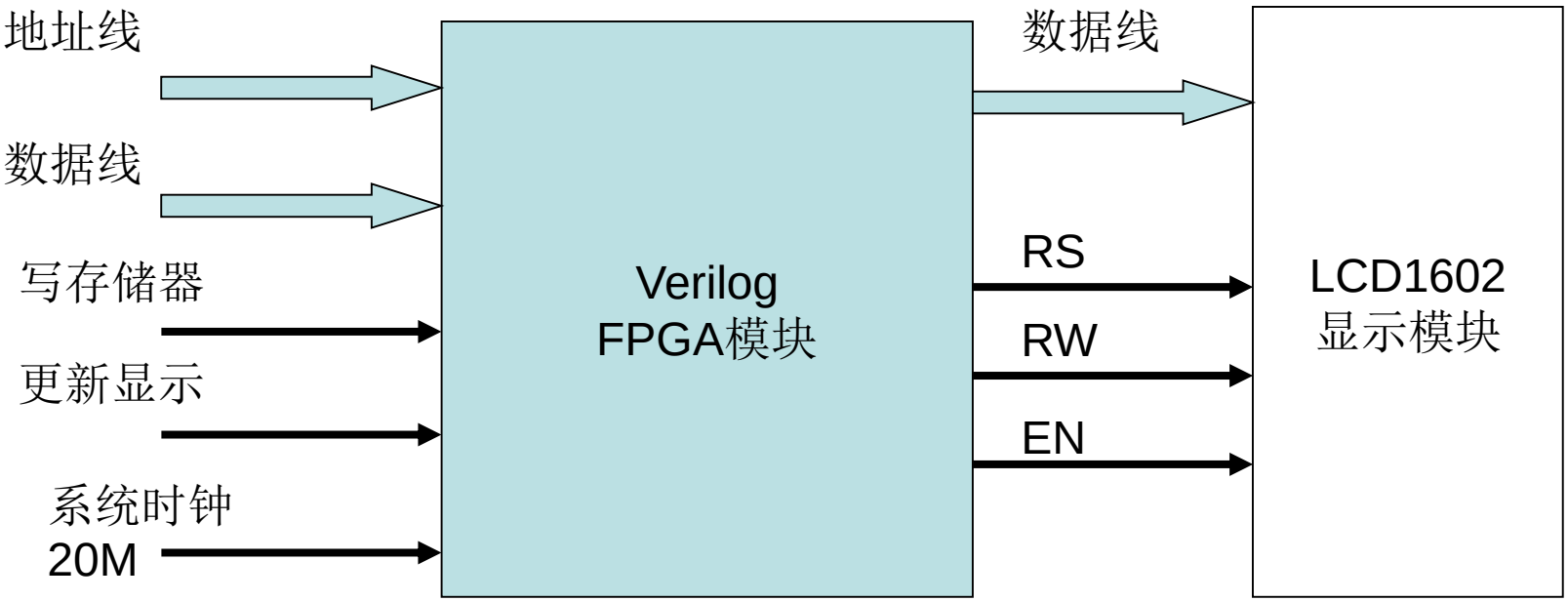
Good Luck

进一步设计
模块基本要求

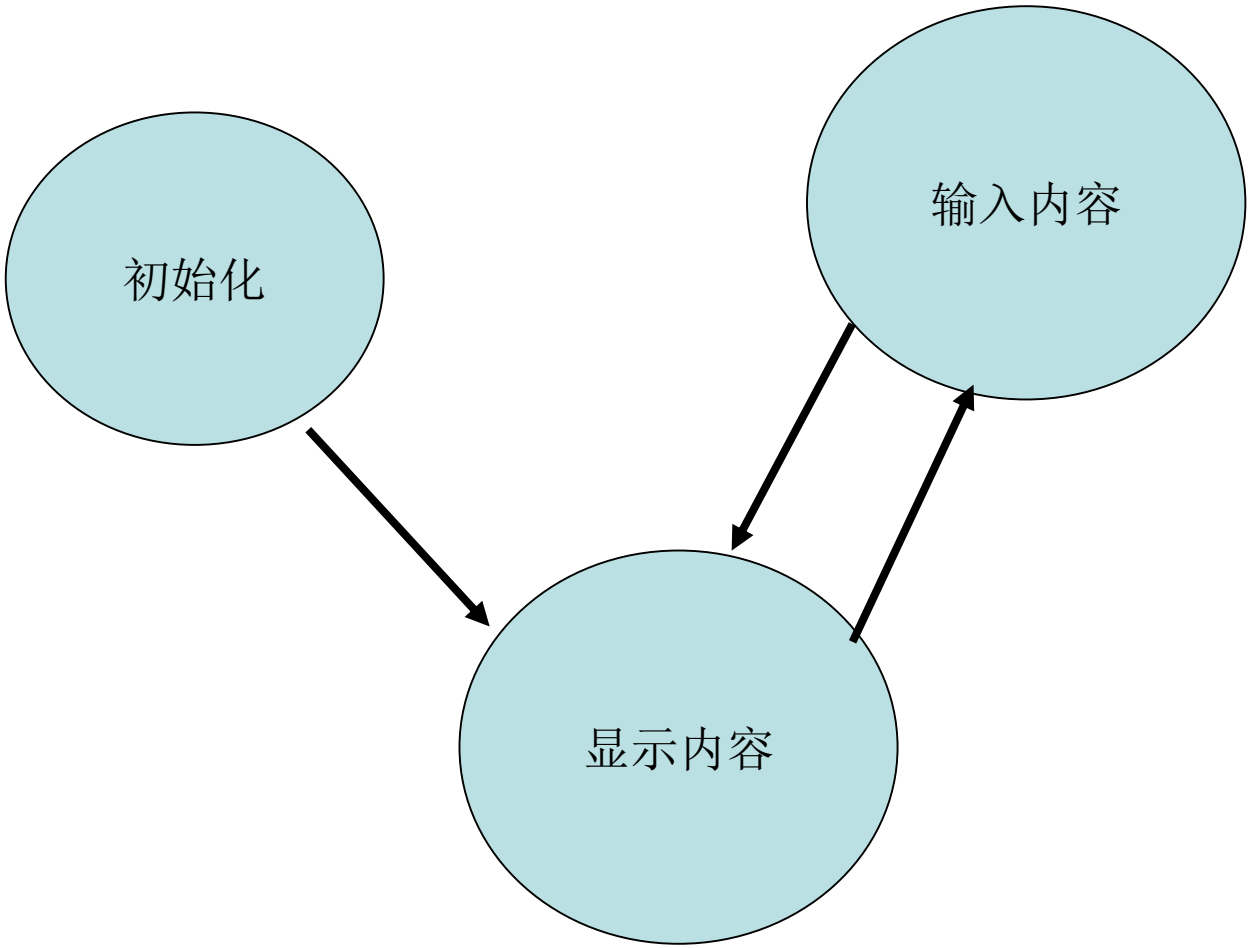
- 采用双端口存储器作为显示缓冲区
- LCD模块负责显示内容
- 在LCD_fresh命令下，读取并显示缓冲区

模块基本设计

- 采用双端口存储器作为显示缓冲区
- LCD模块负责显示内容
- 在LCD_fresh命令下，读取并显示缓冲区



基本状态转移图设计



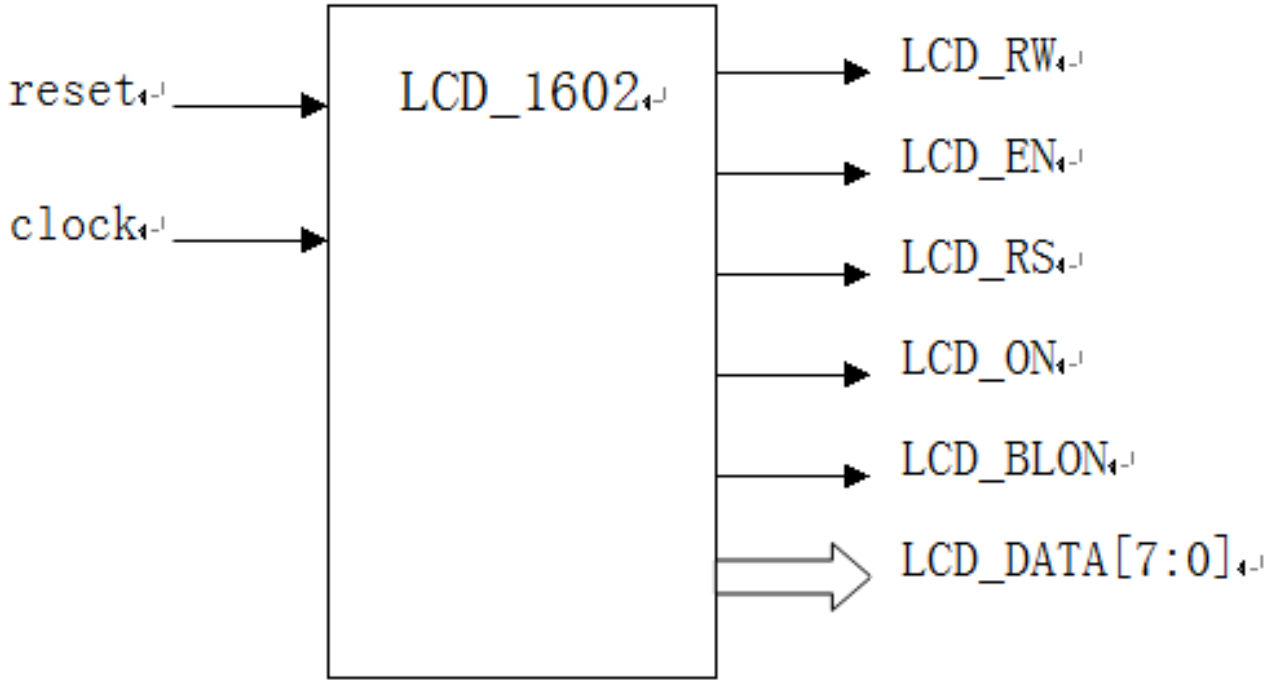
另一种解决方案

- **一、功能描述**
- 本设计实现LCD_1602的接口，具体功能定义如下：
- 1、异步复位信号；
- 2、按下复位键后在LCD_1602液晶屏显示内部设置好的字符,每间隔0.1秒显示一个字符。

来源于网络

```
//==========================================================
// Company:  Jackin
// Engineer: Jackin
//
// Create Date:   2012-2-26
// Design Name:   LCD_1602
// Module Name:   LCD_1602
// Project Name:  LCD_1602
// Target Device: EP2C5t144
// Tool versions: Modelsim SE PLUS 6.2b  &  Quartus II 9.0
// Description:   CFAH1602B-TMC-JP port
//
// Dependencies:  DE2 BOARD
//
// Revision:
// Additional Comments:
//==========================================================
```

二、输入输出信号描述



信号名	输入/输出	目标/源	功能描述
reset	Input	Pin	复位。
clock	Input	Pin	时钟，频率 50MHz ，占空比 1：1。
LCD_RW	Output	Pin	读、写操作选择，0：写，1：读。
LCD_EN	Output	Pin	片选使能，高电平有效。
LCD_RS	Output	Pin	寄存器选择，0：指令，1：数据。
LCD_ON	Output	Pin	液晶驱动电源。
LCD_BLON	Output	Pin	背景灯开关，0 为关，1 为开。
LCD_DATA[7:0]	Output	Pin	显示的 8 位数据代码

三、设计思想

1、首先把50MHz的时钟信号转化为10Hz的信号，实现每0.1秒显示一个字符，设计一个分频器。

2、LCD_RW设置为0，因为只有写信号，没有读信号。

3、指令参数设置，LCD_RS设置为0，在每个10Hz时钟的上升沿，对LCD_DATA输入一个参数，实现内部参数的设置，依次为清零、归位、光标右移、画面不动、显示开、光标不显示、光标闪烁关、光标右移一个字符位、设置八位数据接口、两行显示、5*8点阵字符。

4、显示数据的输入，LCD_RS设置为1，在每个10Hz时钟的上升沿，对LCD_DATA输入一个8位字符代码并在液晶屏显示。

顶层模块定义

```
`define LINE_1 12    //the number of line 1
`define LINE_2 19    //the number of line 1 and line 2
module lcd1602bk(reset,
                 clock,
                 LCD_DATA,
                 LCD_RW,
                 LCD_EN,
                 LCD_RS //,
                 //LCD_ON,
                 //LCD_BLON
                 );
```

接口信号定义

```
input reset;  //reset
input clock;  //20MHz Clock

output [7:0] LCD_DATA; //8-bit LCD DATA
output LCD_RW;   //LCD Read/Write Select,0=Write,1=Read
output LCD_EN;   //LCD Enable
output LCD_RS;   //LCD Command/Data select,0=Command,1=Data
//output LCD_ON;   //LCD Power ON/OFF,0=OFF,1=ON
//output LCD_BLON; //LCD Back Light ON/OFF,0=OFF,1=ON

reg LCD_RS;
reg [7:0] LCD_DATA;

//Fixed signal
//assign LCD_ON = 1'b1;  //Power On
//assign LCD_BLON = 1'b1;//Back Light On
assign LCD_RW = 1'b0;  //Because of no write,so LCD_RW signal
                       //is always low level
```

10Hz 时钟的产生

```
//----------------------------------------
//Produce 10Hz clock signal
//----------------------------------------

reg LCD_CLOCK;     //10Hz Clock
reg [21:0] count; //counter

always@(posedge clock or negedge reset)
begin
    if(!reset)  //reset
        begin
        count <= 22'd0;
        LCD_CLOCK <= 1'b0;
        end
    else if(count == 999999)  //20Hz turn
        begin
        count <= 22'd0;
        LCD_CLOCK <= ~LCD_CLOCK;
        end
    else
        count <= count + 1'b1;  //count
end
```

- 10Hz时钟也负责产生操作信号EN，100ms

```
//enable negative edge
assign LCD_EN = LCD_CLOCK;
```

参数定义和变量定义

```
//------------------------------------------------------------
//LCD internal parameter Settings
//------------------------------------------------------------

//Set parameters
parameter     IDLE = 10'b00_0000_0000; //initial state
parameter    CLEAR = 10'b00_0000_0001; //clear
parameter   RETURN = 10'b00_0000_0010; //return home
parameter     MODE = 10'b00_0000_0100; //entry mode set
parameter  DISPLAY = 10'b00_0000_1000;//display ON/OFF control
parameter    SHIFT = 10'b00_0001_0000;//cursor or display shift
parameter FUNCTION = 10'b00_0010_0000; //function set
parameter    CGRAM = 10'b00_0100_0000; //set CGRAM address
parameter    DDRAM = 10'b00_1000_0000; //set DDRAM address
parameter    WRITE = 10'b01_0000_0000; //write data to RAM
parameter     STOP = 10'b10_0000_0000; //release control
```

```
reg [9:0] state; //state machine code
reg [5:0] char_count;  //char counter
reg [7:0] data_display;//display data
```

由状态确定是写命令还是写数据
RS信号

```
//If read,LCD_RS is high level,else is low level
always@(posedge LCD_CLOCK or negedge reset)
begin
    if(!reset)
        LCD_RS <= 1'b0;
    else if(state == WRITE)
        LCD_RS <= 1'b1;
    else
        LCD_RS <= 1'b0;
end
```

```
//State machine
always@(posedge LCD_CLOCK or negedge reset)
begin
    if(!reset)
    begin
        state <= IDLE;
        LCD_DATA <= 8'bzzzz_zzzz;
        char_count <= 6'd0;
    end
    else
    begin
        case(state)
        //start
        IDLE:begin
                state <= CLEAR;
                LCD_DATA <= 8'bzzzz_zzzz;
             end
        //clear
```

```
//clear
CLEAR:begin
        state <= RETURN;
        LCD_DATA <= 8'b0000_0001;
       end
//home
RETURN:begin
        state <= MODE;
        LCD_DATA <= 8'b0000_0010;
        end
//cursor move to the right
//display don't move
MODE:begin
        state <= DISPLAY;
        LCD_DATA <= 8'b0000_0110;
      end
//display on
//cursor and blinking of cursor off
DISPLAY:begin
        state <= SHIFT;
        LCD_DATA <= 8'b0000_1100;
         end
//cursor moving
```

```
//cursor moving
//move to the right
SHIFT:begin
        state <= FUNCTION;
        LCD_DATA <= 8'b0001_0100;
      end
//Set interface data length(8-bit)
//numbers of display line(2-line)
//display font type(5*8 dots)
FUNCTION:begin
        state <= DDRAM;
        LCD_DATA <= 8'b0011_1000;
          end
//Set DDRAM address in address counter
DDRAM:begin
        state <= WRITE;
        if(char_count <= `LINE_1)
           LCD_DATA <= 8'b1000_0000;//line 1
        else
           LCD_DATA <= 8'b1100_0000;//line 2
      end
//Write data into internal RAM
```

```
        //Write data into internal RAM
        WRITE:begin
                if(char_count == `LINE_1)
                    state <= DDRAM;
                else
                    state <= WRITE;
                if(char_count == `LINE_2)
                    state <= STOP;
                char_count <= char_count + 1'b1;
                LCD_DATA <= data_display;
              end
        //Finish
        STOP:state <= STOP;
        //Other state
        default:state <= IDLE;
        endcase
    end
end
```

```
//-------------------------------
//the data of display
//-------------------------------

always@(char_count)
begin
    case(char_count)
            6'd0: data_display = "H";
            6'd1: data_display = "e";
            6'd2: data_display = "l";
            6'd3: data_display = "l";
            6'd4: data_display = "o";
            6'd5: data_display = ",";
            6'd6: data_display = "J";
            6'd7: data_display = "a";
            6'd8: data_display = "c";
            6'd9: data_display = "k";
            6'd10: data_display = "i";
            6'd11: data_display = "n";
```

```
            6'd10: data_display = "!";
            6'd11: data_display = "n";
            6'd12: data_display = "!";
            6'd13: data_display = "W";
            6'd14: data_display = "e";
            6'd15: data_display = "l";
            6'd16: data_display = "c";
            6'd17: data_display = "o";
            6'd18: data_display = "m";
            6'd19: data_display = "e";
            default:data_display = 8'd32;
    endcase
end
endmodule
```

- 按照实验板锁定引脚
- 下载程序以后，LCD显示内容

修改程序实现显示内容可变

- 使用实验板按钮开关，每按一次改变LCD显示内容
- 改变单个显示字符
- 改变多个显示字符

- 例如：按S3(引脚P115)依次显示ABCDEF等单个字符
- 按S4(P114)依次显示“1234”，“5678”，“9012”等

- 按钮开关去抖动程序可以参考教材P219

end