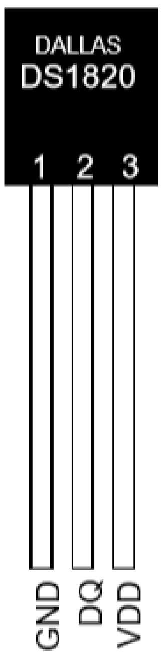
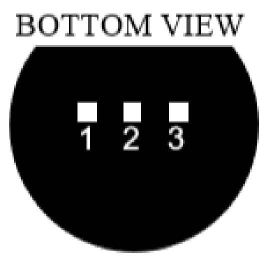
单总线温度传感器驱动

王安然
STEP FPGA

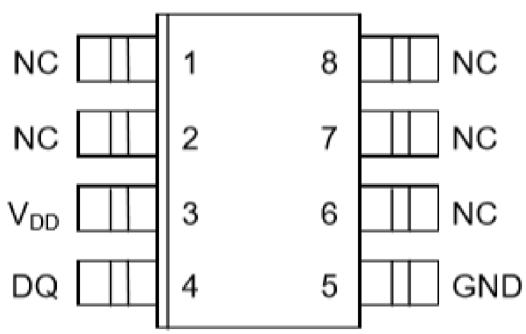
DS18B20Z

DS18B20是我们日常设计中常用的一款温度传感器芯片，只需要一根总线就可以实现通信，非常的方便，接下来一起学习DS18B20的驱动





DS18B20 To-92
Package

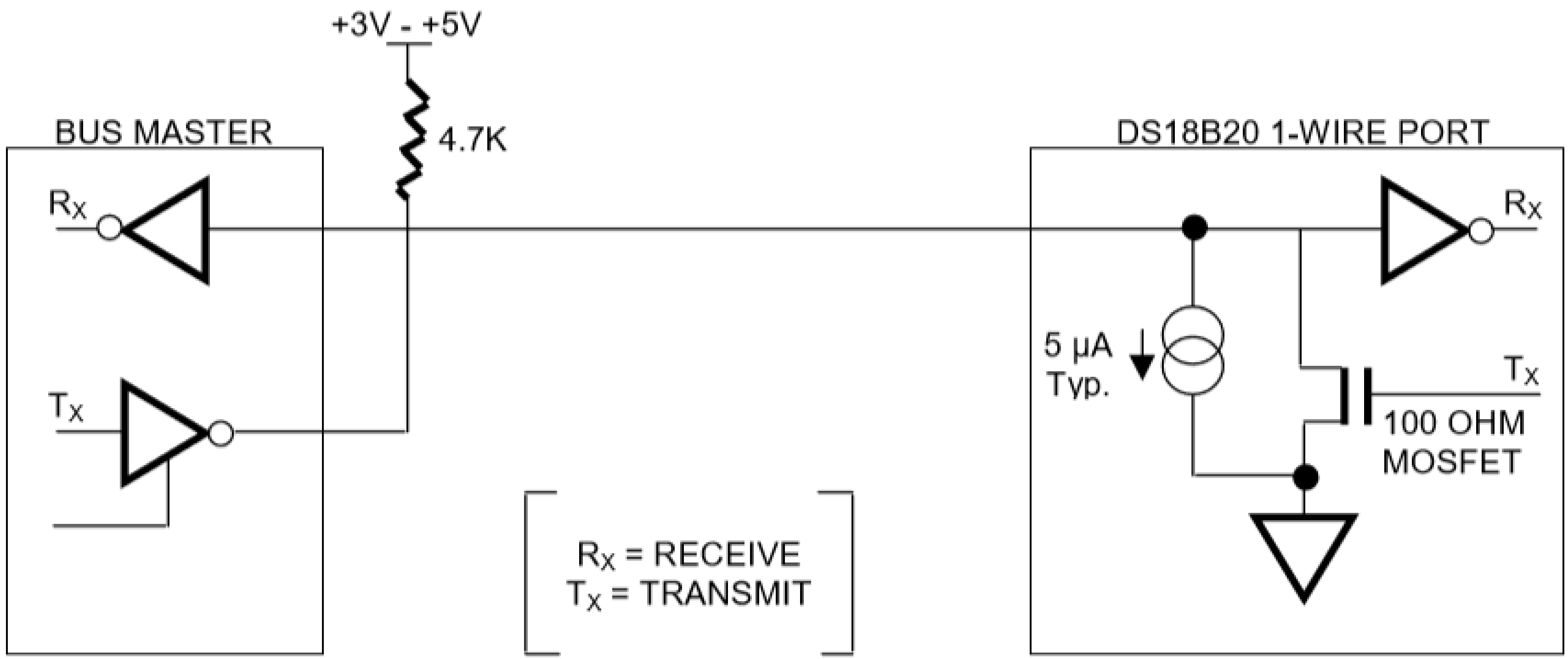


DS18B20Z
8-Pin SOIC (150 mil)

PIN DESCRIPTION

GND	- Ground
DQ	- Data In/Out
V_{DD}	- Power Supply Voltage
NC	- No Connect

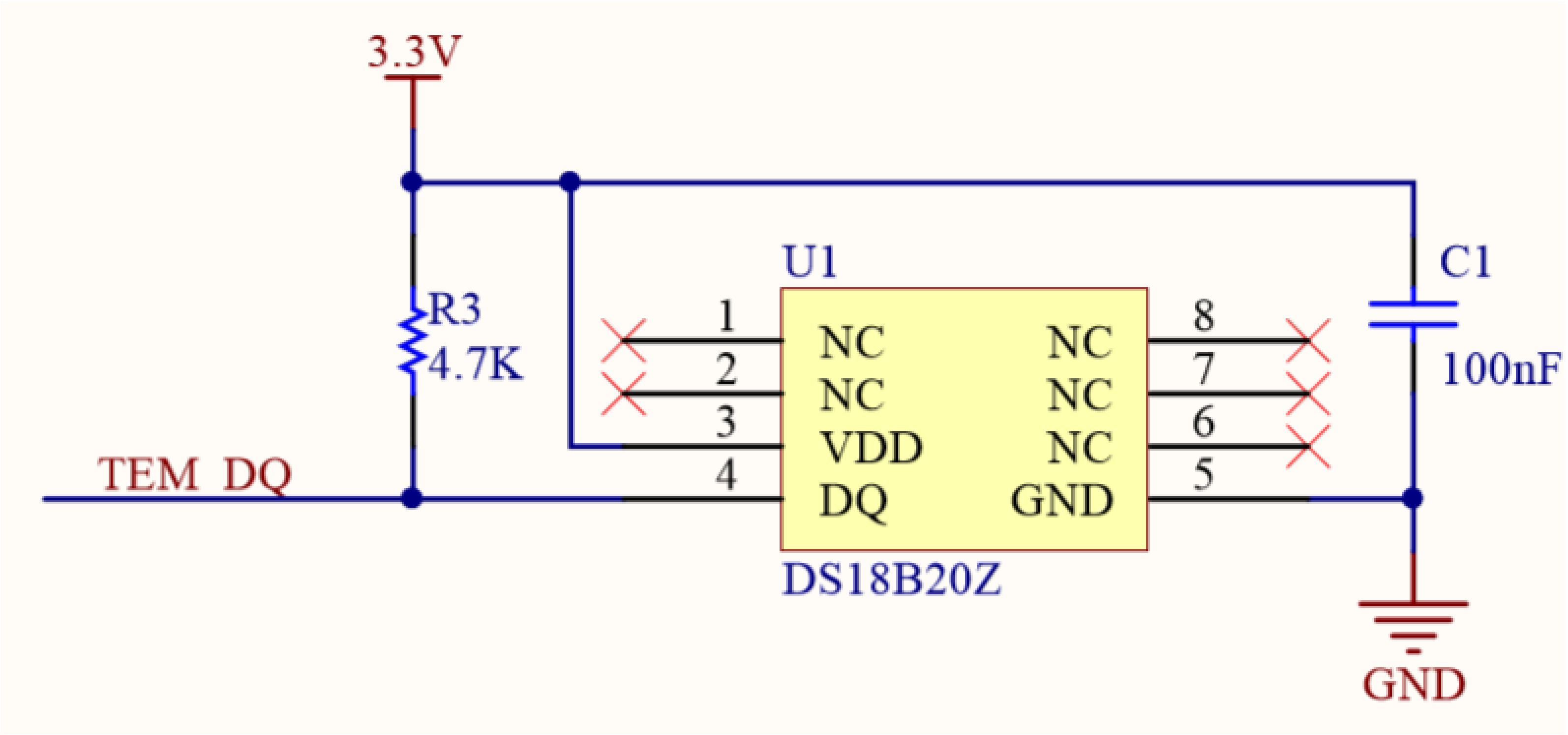
DS18B20Z配置



+3V - +5V
4.7K
BUS MASTER
Rx
Tx
DS18B20 1-WIRE PORT
Rx
5 µA
Typ.
Tx
100 OHM
MOSFET
Rx = RECEIVE
Tx = TRANSMIT

DS18B20Z连接

Dot Matrix板子上的温度传感器硬件连接如下:



DS18B20Z指令

INSTRUCTION	DESCRIPTION	PROTOCOL	1-WIRE BUS AFTER ISSUING PROTOCOL	NOTES
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	<read temperature busy status>	1
MEMORY COMMANDS				
Read Scratchpad	Reads bytes from scratchpad and reads CRC byte.	BEh	<read data up to 9 bytes>	
Write Scratchpad	Writes bytes into scratchpad at addresses 2 through 4 (TH and TL temperature triggers and config).	4Eh	<write data into 3 bytes at addr. 2 through. 4>	3
Copy Scratchpad	Copies scratchpad into nonvolatile memory (addresses 2 through 4 only).	48h	<read copy status>	2
Recall E^2	Recalls values stored in nonvolatile memory into scratchpad (temperature triggers).	B8h	<read temperature busy status>	
Read Power Supply	Signals the mode of DS18B20 power supply to the master.	B4h	<read supply status>	

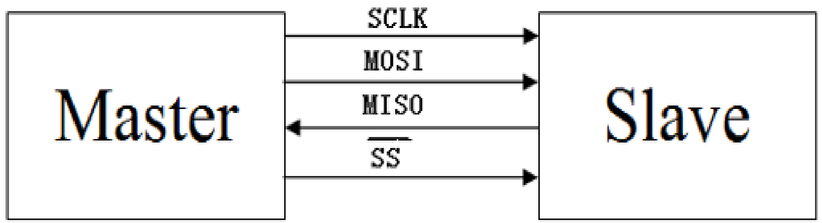
DS18B20Z驱动流程

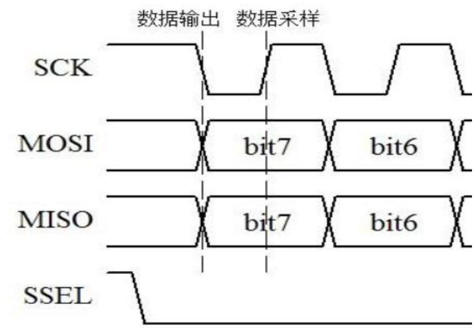
接下来简要介绍如何驱动（更加详细的信息需要大家参考数据手册），不同的功能需求对应不同寄存器配置，本设计执行的操作案例如下。

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	44h	Master issues Convert T command.
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	BEh	Master issues Read Scratchpad command.
Rx	2 data Byte	Master reads temperature data of scratchpad.

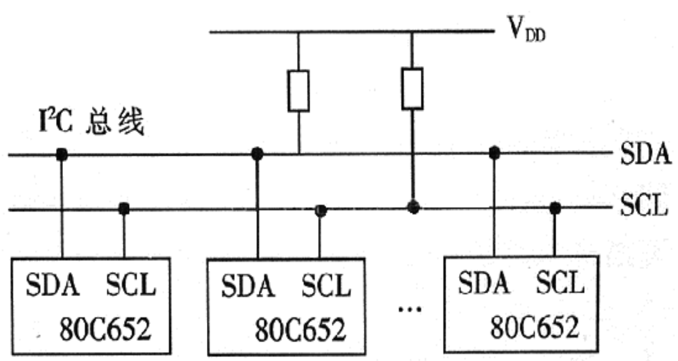
通信总线对比

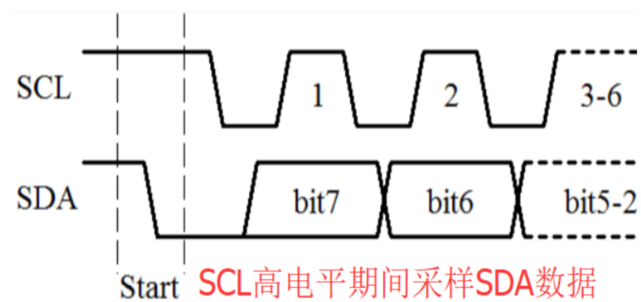
SPI总线





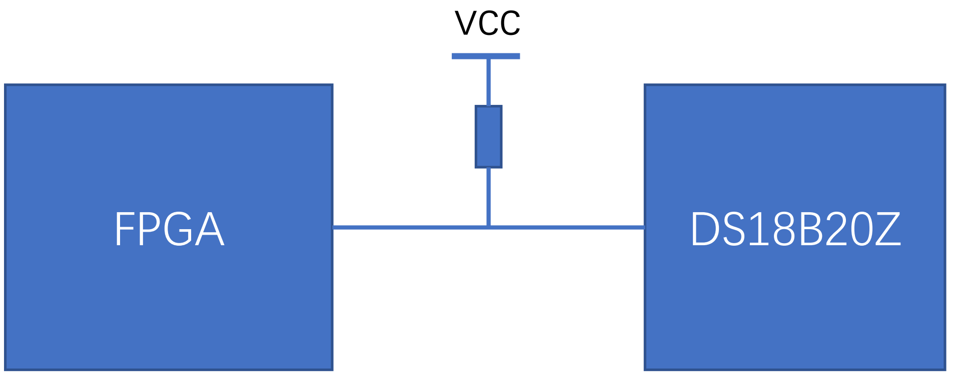
I2C总线





单总线通信

单总线连接方式



单根总线通信需要考虑多个因素：

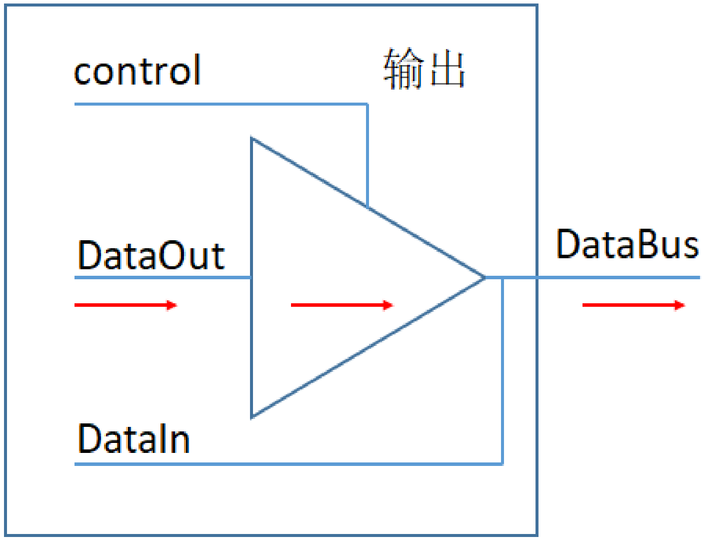
双向通信，分时收发，两个设备收发控制（不能同时发数据）

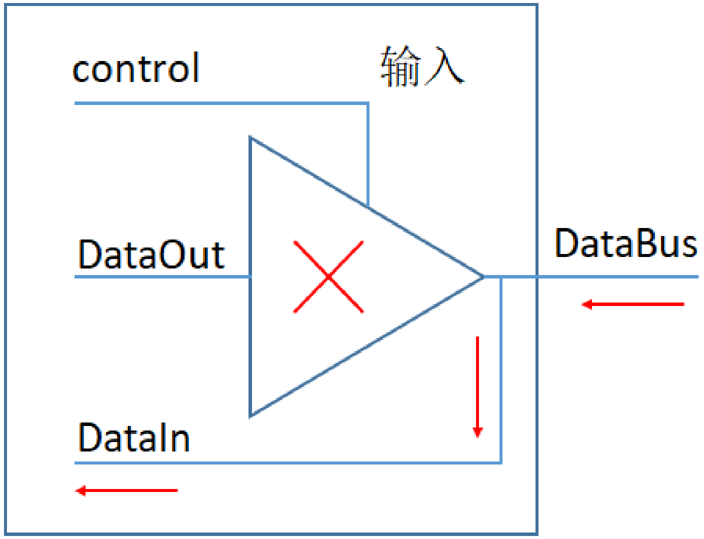
通信速率，收发设备约定好数据传输的速率，是的双方吞吐率一致

通信同步处理

双向端口设计

FPGA管脚模型如下：
当端口输出时，control端控制三态门导通，DataBus状态受DataOut控制
当端口输入时，control端控制三态门高阻，FPGA不能驱动DataBus，DataBus状态受外设电路控制，并通过DataIn端输入

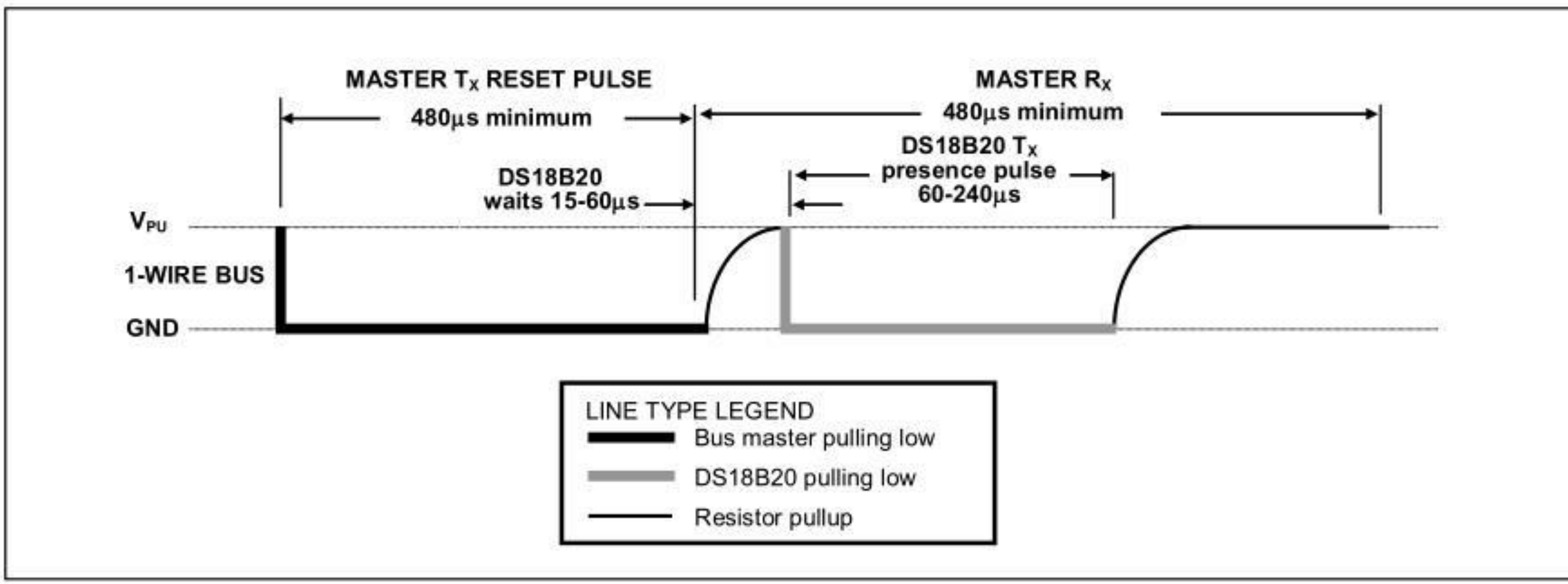




软件复位时序控制

软件复位，需要主机发送低电平，持续至少480us时间，
然后释放单总线，
经过至少60us时间，主机采样单总线状态，判断温度传感器的响应状态。

Figure 13. Initialization Timing

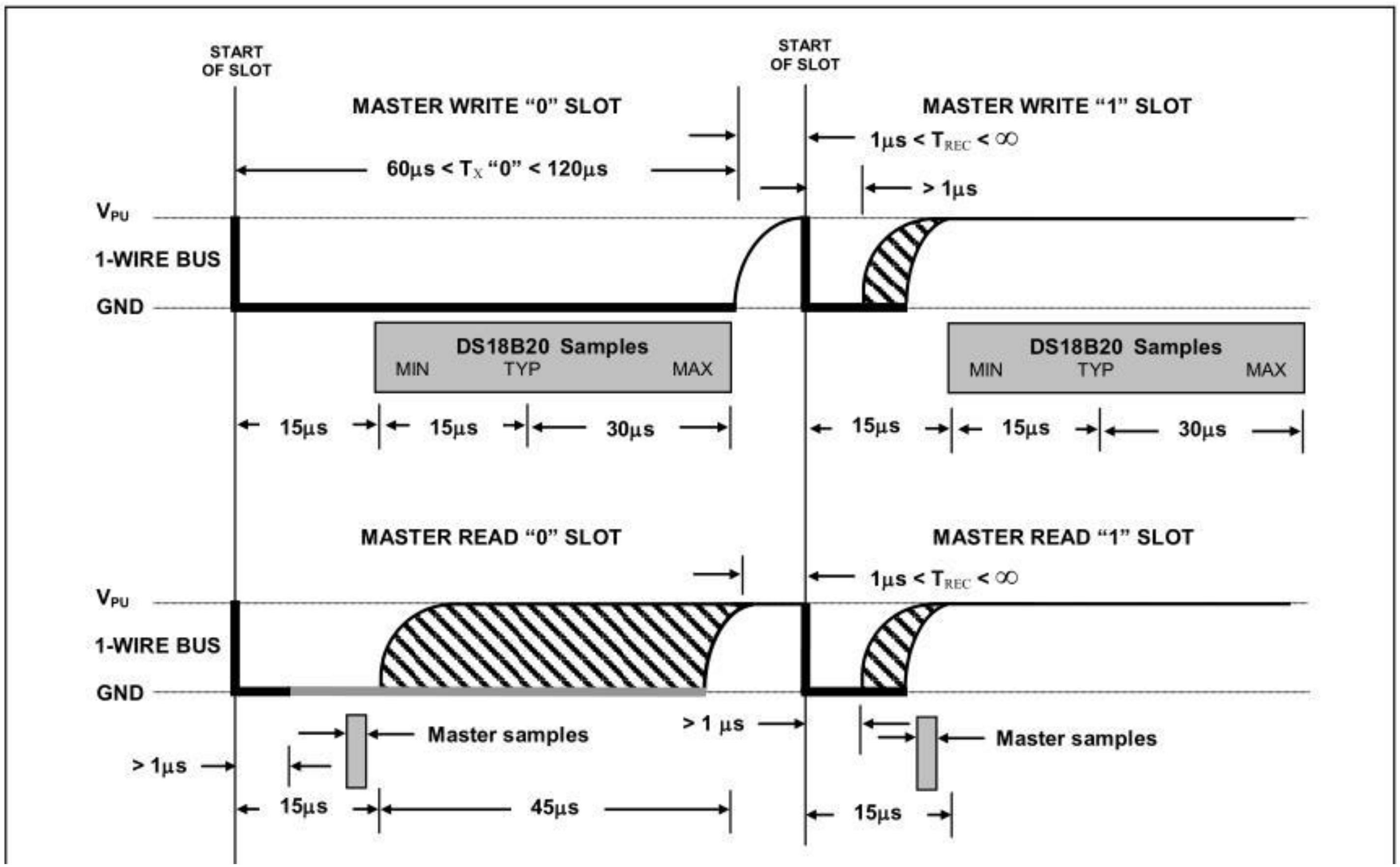


软件复位实现

```
INIT:begin
  if(cnt_init >= 3'd6) cnt_init <= 1'b0;
  else cnt_init <= cnt_init + 1'b1;
  case(cnt_init)
    3'd0: begin one_wire_buffer <= 1'b0; end
    3'd1: begin num_delay <= 20'd500;state <= DELAY;state_back <= INIT; end
    3'd2: begin one_wire_buffer <= 1'bz; end
    3'd3: begin num_delay <= 20'd100;state <= DELAY;state_back <= INIT; end
    3'd4: begin if(one_wire) state <= IDLE; else state <= INIT; end
    3'd5: begin num_delay <= 20'd400;state <= DELAY;state_back <= INIT; end
    3'd6: begin state <= MAIN; end default: state <= IDLE;
  endcase
end
```

收发时序控制

Figure 14. Read/Write Time Slot Timing Diagram



主机发送数据:

控制拉低至少1us

输出数据至少60us

释放总线至少1us

主机接收数据:

控制拉低至少1us

主机释放总线

自开始起15us内完成数据采样

发送数据实现

发送数据时序

```
6'd1: begin one_wire_buffer <= 1'b0; end
6'd2: begin num_delay <= 20'd2;state <= DELAY;state_back <= WRITE; end
6'd3: begin one_wire_buffer <= data_wr_buffer[0]; end
6'd4: begin num_delay <= 20'd80;state <= DELAY;state_back <= WRITE; end
6'd5: begin one_wire_buffer <= 1'bz; end
6'd6: begin num_delay <= 20'd2;state <= DELAY;state_back <= WRITE; end
```

接收数据时序

```
6'd0: begin one_wire_buffer <= 1'b0; end
6'd1: begin num_delay <= 20'd2;state <= DELAY;state_back <= READ; end
6'd2: begin one_wire_buffer <= 1'bz; end
6'd3: begin num_delay <= 20'd10;state <= DELAY;state_back <= READ; end
6'd4: begin temperature_buffer[0] <= one_wire; end
6'd5: begin num_delay <= 20'd55;state <= DELAY;state_back <= READ; end
```

传感器分辨率配置

CONFIGURATION REGISTER

Byte 4 of the scratchpad memory contains the configuration register, which is organized as illustrated in Figure 8. The user can set the conversion resolution of the DS18B20 using the R0 and R1 bits in this register as shown in Table 2. The power-up default of these bits is R0 = 1 and R1 = 1 (12-bit resolution). Note that there is a direct tradeoff between resolution and conversion time. Bit 7 and bits 0 to 4 in the configuration register are reserved for internal use by the device and cannot be overwritten.

Figure 8. Configuration Register

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	R1	R0	1	1	1	1	1

Table 2. Thermometer Resolution Configuration

R1	R0	RESOLUTION (BITS)	MAX CONVERSION TIME	
0	0	9	93.75ms	($t_{CONV}/8$)
0	1	10	187.5ms	($t_{CONV}/4$)
1	0	11	375ms	($t_{CONV}/2$)
1	1	12	750ms	(t_{CONV})

上电默认采用12-bit解析，
温度转换时间为 750ms

温度编码运算

Table 1. Temperature/Data Relationship

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+125	0000 0111 1101 0000	07D0h
+85*	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

温度传感器数据结构

温度数据运算，如果温度为正，有效数据为自身，如果温度为负，有效数据为自身的补码。

温度运算为有效数据乘以0.0625

Figure 2. Temperature Register Format

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

BCD转码方法

最后将温度值显示在点阵上，数字系统为二进制数，不符合人的阅读习惯，需要将其转换成十进制的形式，在数字系统一般采用BCD码的形式。

将二进制数转换成BCD码的形式，采用左移加三的算法（以8’hff为例）：

① 左移要转换的二进制码1位
② 左移之后，BCD码分别置于百位、十位、个位
③ 如果移位后所在的BCD码列大于或等于5，则对该值加3
④ 继续左移的过程直至全部移位完成

Operation	Hundreds	Tens	Units	Binary	
HEX				F	F
Start				1 1 1 1	1 1 1 1
Shift 1			1	1 1 1 1	1 1 1
Shift 2			1 1	1 1 1 1	1 1
Shift 3			1 1 1	1 1 1 1	1
Add 3			1 0 1 0	1 1 1 1	1
Shift 4		1	0 1 0 1	1 1 1 1	
Add 3		1	1 0 0 0	1 1 1 1	
Shift 5		1 1	0 0 0 1	1 1 1	
Shift 6		1 1 0	0 0 1 1	1 1	
Add 3		1 0 0 1	0 0 1 1	1 1	
Shift 7	1	0 0 1 0	0 1 1 1	1	
Add 3	1	0 0 1 0	1 0 1 0	1	
Shift 8	1 0	0 1 0 1	0 1 0 1		
BCD	2	5	5		

BCD转码实现

```
module bin_to_bcd # ( parameter B_SIZE = 21 )
(
input rst_n, // system reset, active low
input [B_SIZE-1:0] bin_code, // binary code
output reg [B_SIZE+3:0] bcd_code // bcd code
);

reg [2*B_SIZE+3:0] shift_reg;
always@(bin_code or rst_n)begin
  shift_reg= {25'h0,bin_code};
  if(!rst_n) bcd_code <= 0;
  else begin
    repeat(B_SIZE)//repeat B_SIZE times begin
      if (shift_reg[24:21] >= 5) shift_reg[24:21] = shift_reg[24:21] + 2'b11;
      if (shift_reg[28:25] >= 5) shift_reg[28:25] = shift_reg[28:25] + 2'b11;
      if (shift_reg[32:29] >= 5) shift_reg[32:29] = shift_reg[32:29] + 2'b11;
      if (shift_reg[36:33] >= 5) shift_reg[36:33] = shift_reg[36:33] + 2'b11;
      if (shift_reg[40:37] >= 5) shift_reg[40:37] = shift_reg[40:37] + 2'b11;
      if (shift_reg[44:41] >= 5) shift_reg[44:41] = shift_reg[44:41] + 2'b11;
      shift_reg = shift_reg << 1;
    end
    bcd_code<=shift_reg[45:21];
  end
end
```

组合逻辑实现
使用阻塞赋值语句

温度运算实现

温度运算为有效数据乘以0.0625，这里乘以625，转码后移动小数点实现除以10000

```
// translate temperature_code to real temperature
wire [20:0] bin_code = data_out[10:0] * 16'd625;

wire [24:0] bcd_code; //Translate binary code to bcd code
bin_to_bcd u2
(
.rst_n (rst_n ), // system reset, active low
.bin_code (bin_code ), // binary code
.bcd_code (bcd_code ) // bcd code
);

dot_array_driver u3
(
.clk (clk ),
.rst_n (rst_n ),
.data (bcd_code[23:12]), //保留一位小数
.row (row ),
.col (col )
);
```

点阵数据更新

点阵显示温度的格式XX.X，两次数据之间间隔一个空白页

```
reg [15:0] dot = {8'h40, 8'h00}; //小数点字库
reg [63:0] space = {8{8'h00}}; //空白页字库

reg [7:0] cnt2;
always @(posedge clk_800hz or negedge rst_n)
  if(!rst_n) cnt2 <= 1'b0;
  else if(cnt2 >= 8'd28) cnt2 <= 1'b0;
  else if(clk_5hz) cnt2 <= cnt2 + 1'b1;
  else cnt2 <= cnt2;

reg [223:0] mem_r;
always @(posedge clk_800hz or negedge rst_n)
  if(!rst_n) mem_r <= 1'b0;
  else if(cnt2 == 8'd28)
        mem_r <= {space,mem[data[11:8]],mem[data[7:4]],dot,mem[data[3:0]]};
  else if(clk_5hz) mem_r <= {mem_r[215:0],mem_r[223:216]};
  else mem_r <= mem_r;
```

Thanks

扫描二维码
关注小脚丫微信公众号
了解更多FPGA知识