电子发烧友
www.elecfans.com

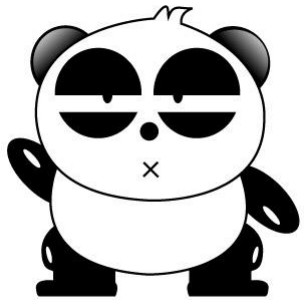
FPGA开发基础公益培训（第3讲）

使用FPGA的开发工具

个人新浪微博：lucky_mao

主要内容

- 编码工具(UltraEdit)
- 仿真工具(Modelsim/Questa Sim)
- 综合实现工具(QuartusII/ISE)
- 在线调试工具(SignalTapII/ChipScope)
- 提高设计可靠性的一些常见原则
- 温度传感器的数据采集电路设计讲解



目标：基本掌握FPGA的开发流程并建立良好的开发习惯

UltraEdit

1：通过模板提高编码的规范性与编码效率

Modelsim/Questa Sim

1：快速地编写testbench验证设计

2：含有器件IP的设计仿真过程

Quartus/ISE

1：快速地开发FPGA的数字系统

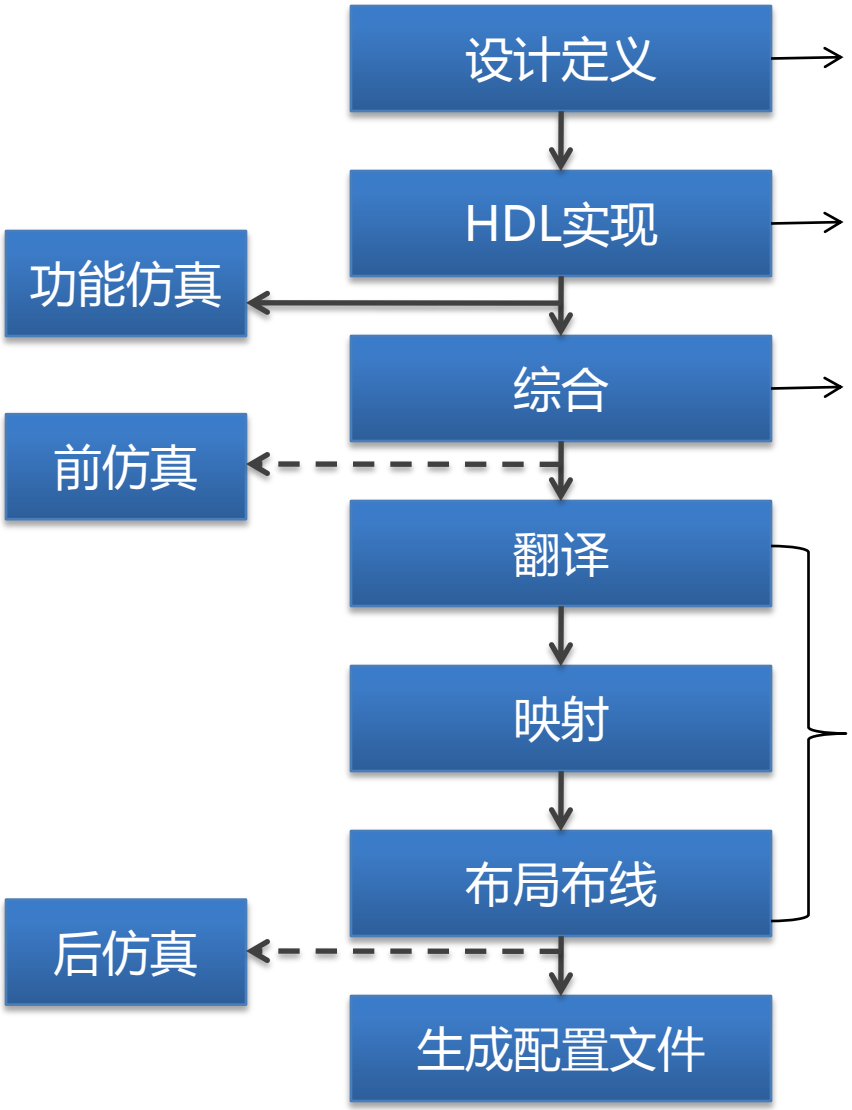
2：采用verilog进行顶层设计开发

SignalTapII/ChipScope

1：掌握电路调试的基本方法

合理的使用工具，使FPGA工程的开发、管理、团队协作变得更加规范和高效。

典型的EDA设计流程



◆进行模块划分，形成接口时序设计文档

◆编码仿真自底向上，集成验证自顶向下

◆关注告警信息，寻找设计隐藏的问题

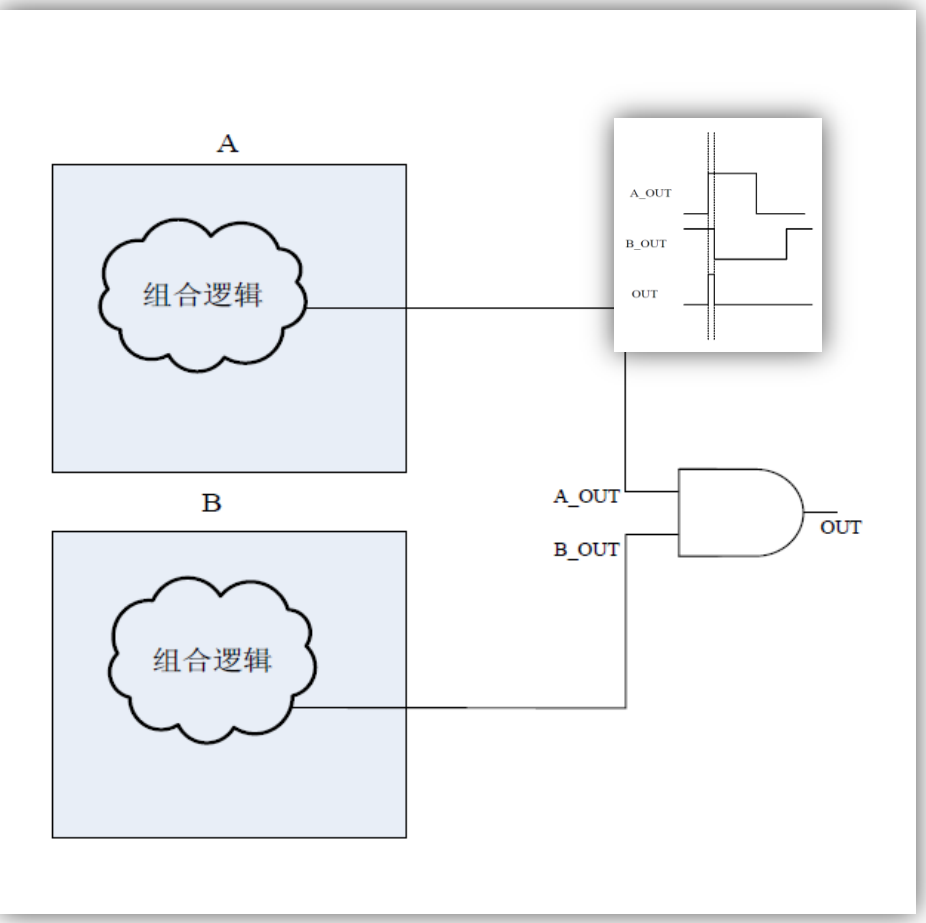
综合过程具有不确定性

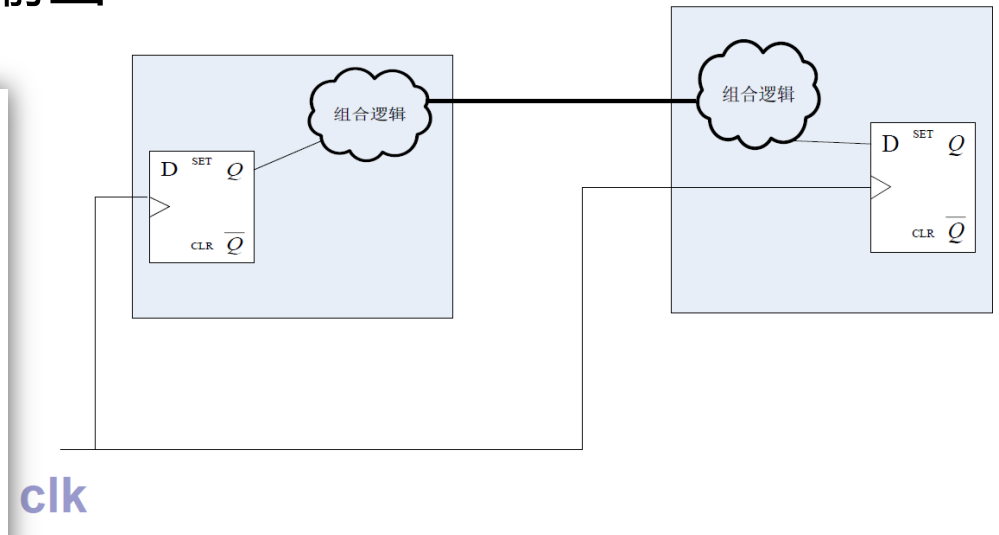
◆关注编译报告，检查并解决时序问题

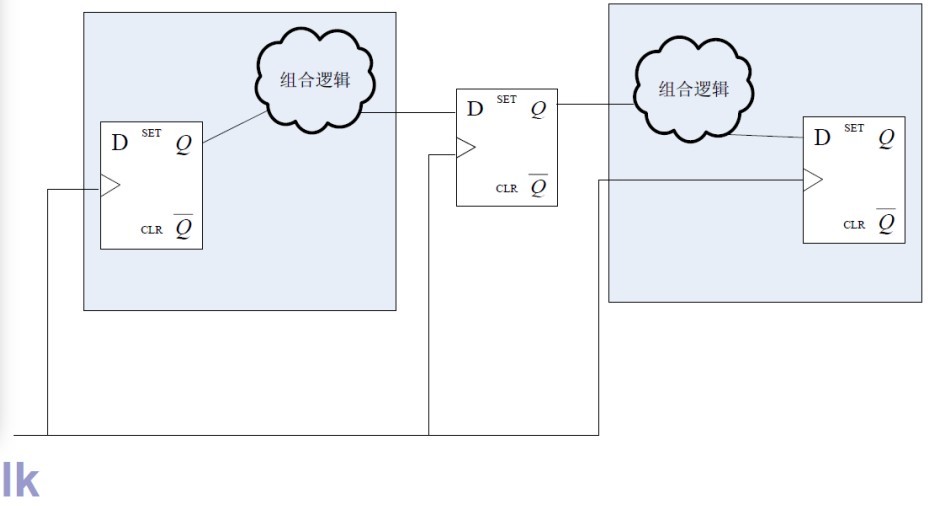
◆下载配置文件，进行在线调试与验证

如何提高设计的可靠性

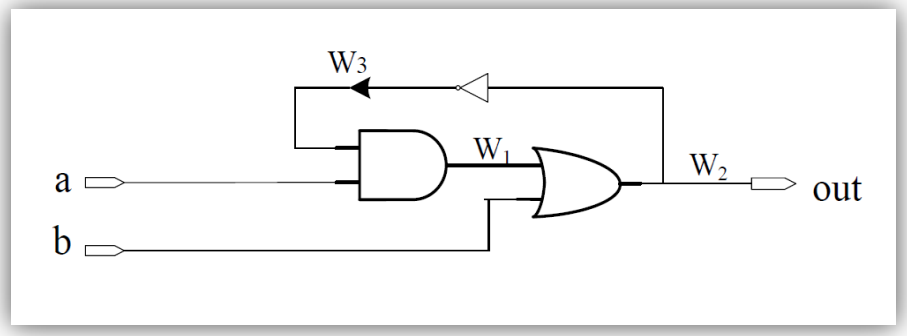
1、模块的边界信号采用寄存器输出



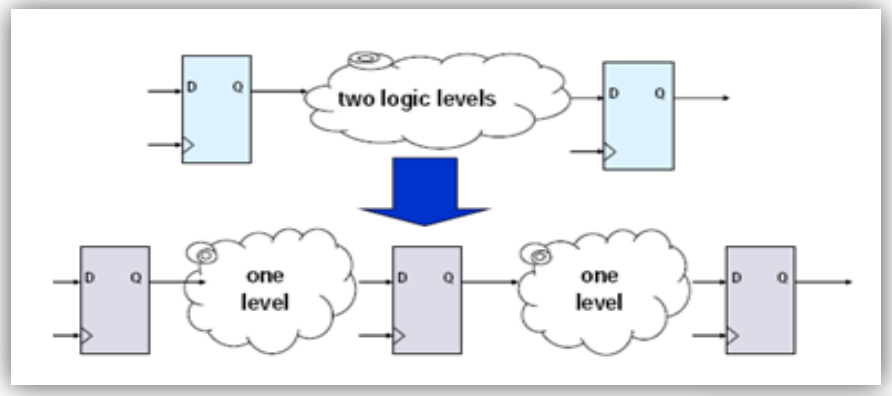




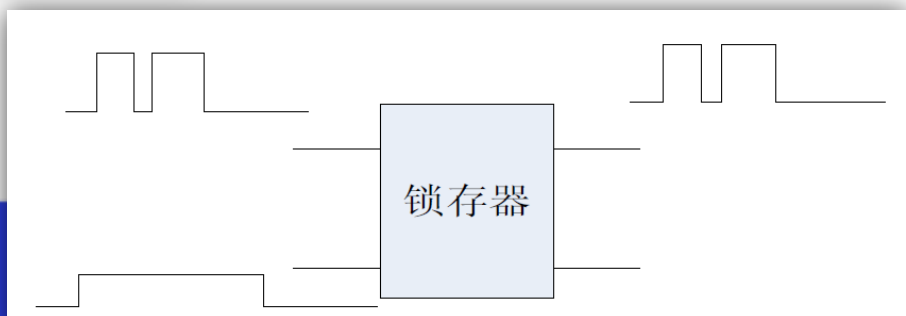
2、组合逻辑模块不要引入反馈



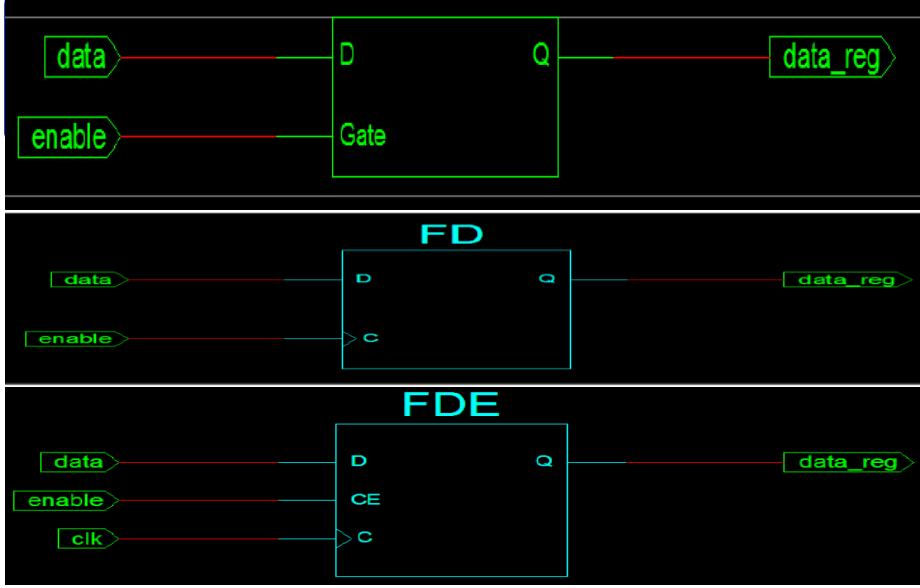
3、用流水线技术提高同步电路的速度



4、设计中要尽量避免锁存器



```
module test(enable,data,data_reg );

input  data,enable;
output data_reg;

assign data_reg= enable? {data} : {data_reg};

endmodule;
```



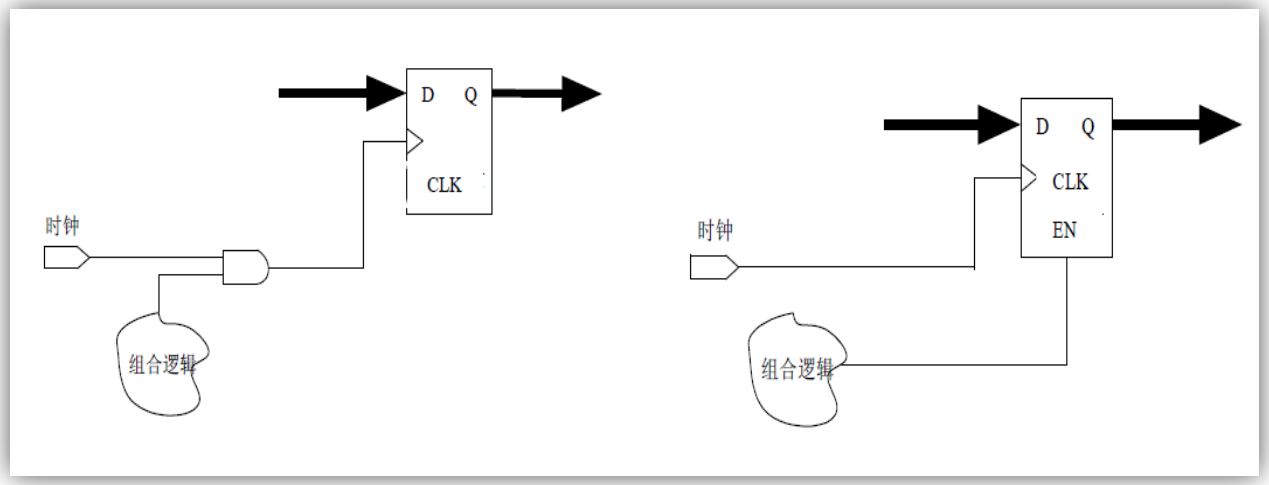

```
module test(enable,data,data_reg );

input  data,enable;
output data_reg;
reg data_reg;

always @(posedge enable)

    if(enable)
      data_reg<=data;
    else
      data_reg<=data_reg;

endmodule;
```

```
module mSDC_IDL08(clk,enable,data,data_reg);

input  data,enable,clk;
output data_reg;
reg data_reg;

always @(posedge clk)

    if(enable)
      data_reg<=data;
    else
      data_reg<=data_reg;

endmodule
```

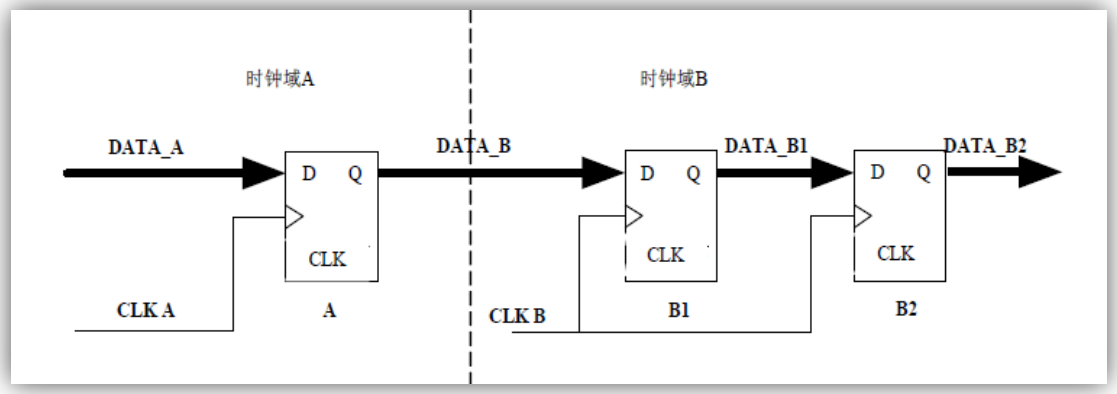
5、采用规范的建模方式

- 组合逻辑采用阻塞赋值
- 时序逻辑采用非阻塞赋值
- 采用分立的always块进行信号建模

6、避免使用门控时钟



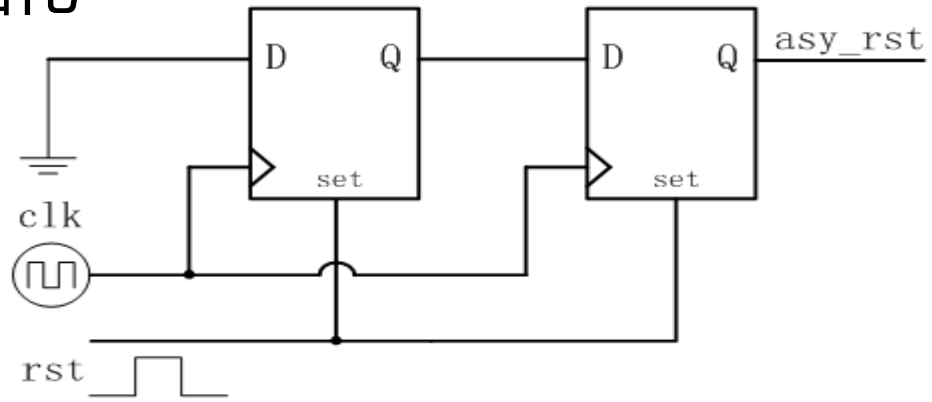
7、注意信号的跨时钟域传输处理



单bit信号的采延与延展

多bit信号采用ram

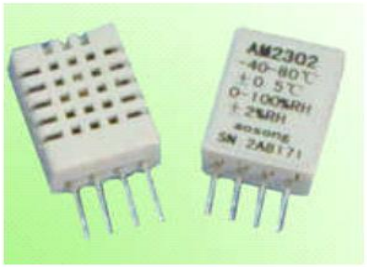
8、异步复位、同步释放，用复位信号对寄存器和输出信号进行初始化



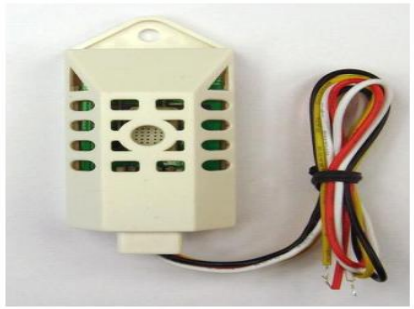
温度传感器的数据采集电路设计讲解

数字温湿度传感器

AM2302



小体积 AM2302



大体积 AM2302

数据格式： 40bit数据=16bit湿度数据+16bit温度数据+8bit校验和

例子： 接收40bit数据如下：

0000 0010 1000 1100　0000 0001 0101 1111　1110 1110

湿度数据　温度数据　校验和

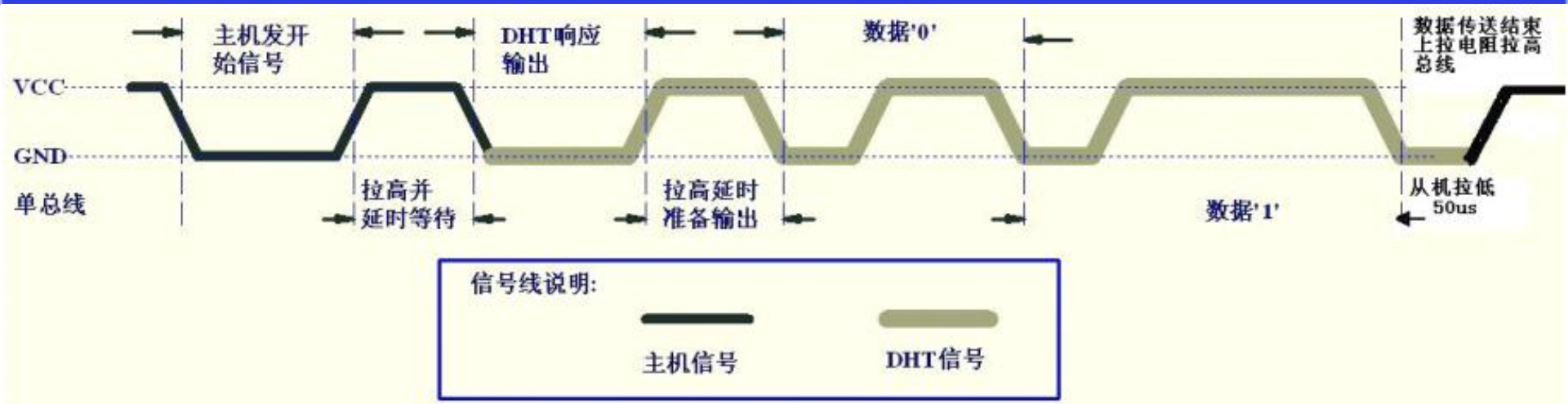
湿度高8位+湿度低8位+温度高8位+温度低8位=的末8位=校验和

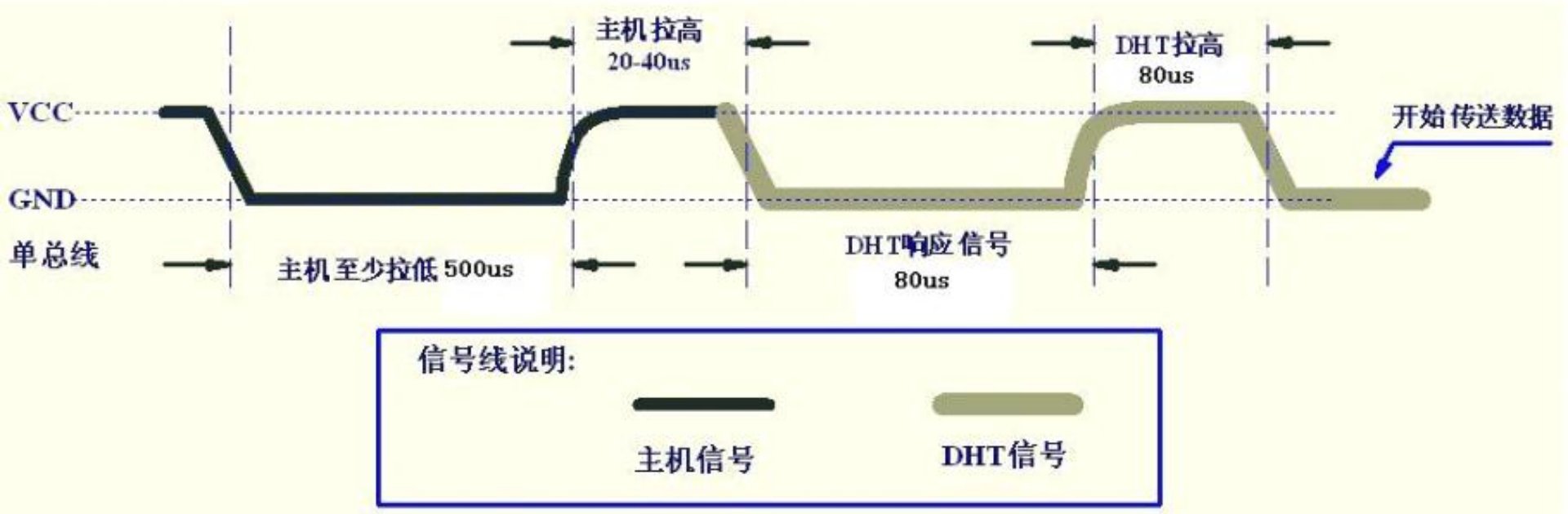
例如：**0000 0010+1000 1100+0000 0001+0101 1111=1110 1110**

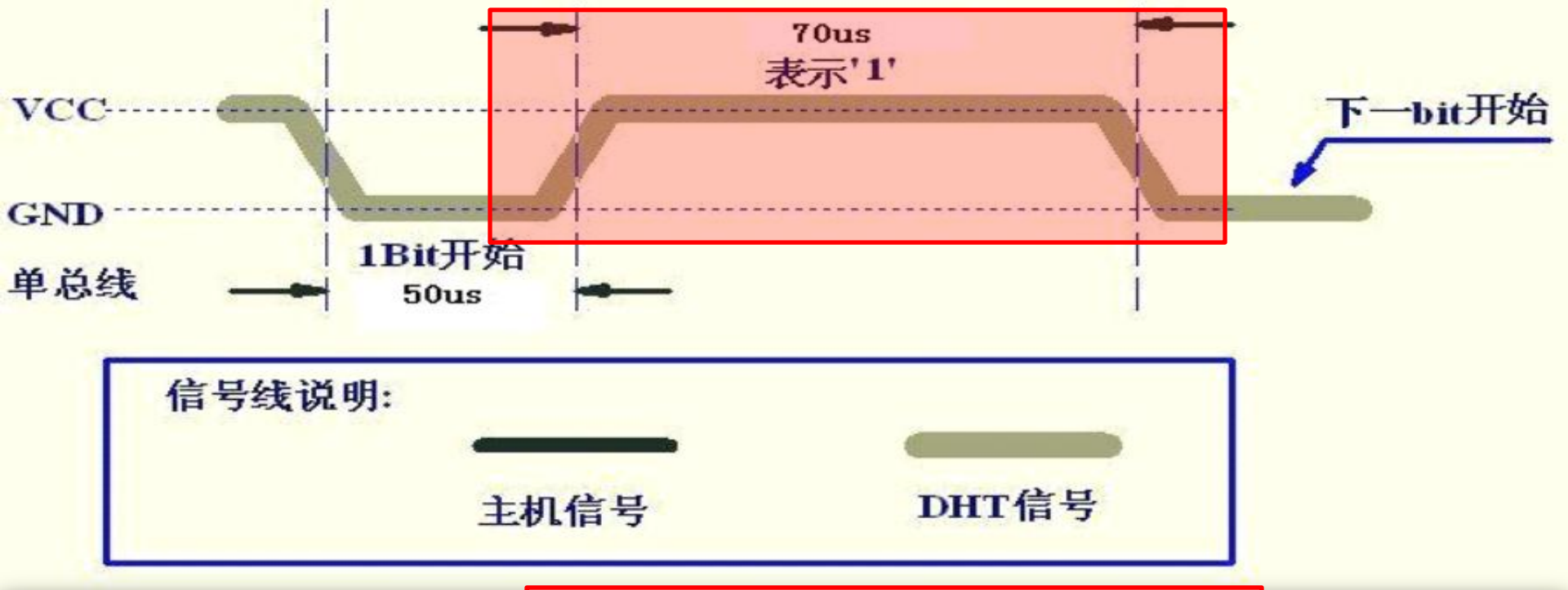
湿度=65.2%RH　温度=35.1℃

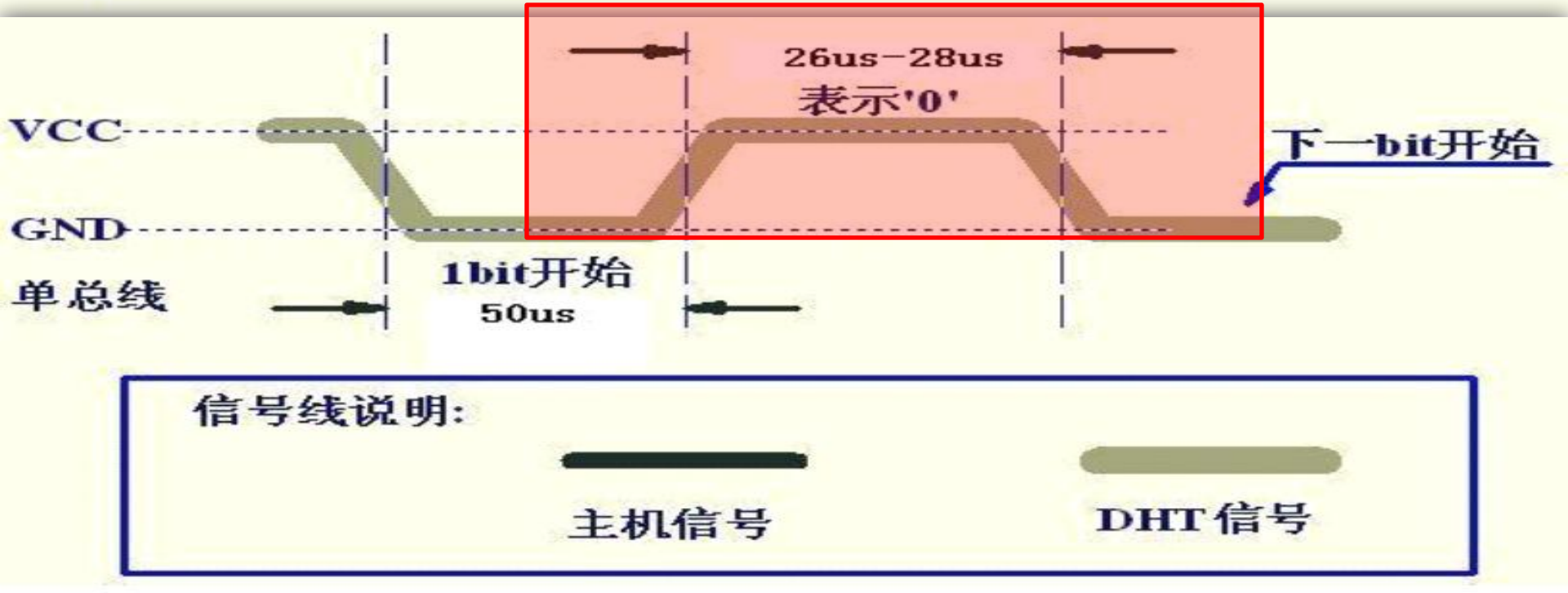
当温度低于0℃时温度数据的最高位置1。

例如：-10.1℃表示为1000 0000 0110 0101


主机发开
始信号
DHT响应
输出
数据'0'
数据传送结束
上拉电阻拉高
总线
VCC
GND
单总线
拉高并
延时等待
拉高延时
准备输出
数据'1'
从机拉低
50us
信号线说明:
主机信号
DHT信号


主机拉高
20-40us
DHT拉高
80us
VCC
开始传送数据
GND
单总线
主机至少拉低500us
DHT响应信号
80us
信号线说明:
主机信号
DHT信号


70us
表示'1'
VCC
GND
单总线
1Bit开始
50us
下一bit开始
信号线说明:
主机信号
DHT信号


26us-28us
表示'0'
VCC
GND
单总线
1bit开始
50us
下一bit开始
信号线说明:
主机信号
DHT信号

```
----------------------------------------------------------------------------*/
`timescale 1ns / 10ps

module top(
          input     wire                asy_rst             , //
          input     wire                clk                 , //
          inout     wire                io_data             , //
          output    wire      [7:0]     o_humih             , //
          output    wire      [7:0]     o_humil             , //
          output    wire      [7:0]     o_temph             , //
          output    wire      [7:0]     o_templ             , //
          output    wire                o_data_valid          //
);
/*------------------Internal registers and wires----------------------------*/
/* Register Input Signals */
 reg                 bus_data_d1;
 reg                 bus_data_d2;

/* Register Output Signals */
 reg                 data;
 reg       [39:0]    result;
 reg                 valid;


/* internal logic use signals */
reg       [5:0]     cnt;
reg       [5:0]     bit_cnt;
reg                 r_clk;
reg       [21:0]    timer_cnt;
reg                 input_flag;
wire                bus_posedge;
wire                bus_negedge;
reg                 r_bus_negedge;
reg       [21:0]    timer1;
reg       [21:0]    timer2;
reg                 bus_value;
reg                 data_valid;
```

```
assign io_data = input_flag?1'bz:data;

assign o_humih = result[39:32];
assign o_humil = result[31:24];
assign o_temph = result[23:16];
assign o_templ = result[15:8];


/*-------------------------------------------------------------
gen the cnt signal
-------------------------------------------------------------*/
always @ ( posedge clk or posedge asy_rst )
begin
    if ( asy_rst )
        cnt <= 6'd0;
    else if ( cnt == 6'd24 )
        cnt <= 6'd0;
    else
        cnt <= cnt + 6'd1;
end


/*-------------------------------------------------------------
gen the r_clk signal
-------------------------------------------------------------*/
always @ ( posedge clk or posedge asy_rst )
begin
    if ( asy_rst )
        r_clk <= 1'b0;
    else if(cnt == 6'd24)
        r_clk <= ~r_clk;
    else
        r_clk <= r_clk;
end
```

```
/*------------------------------------------------------------
gen the timer_cnt signal
------------------------------------------------------------*/
always @ ( posedge r_clk or posedge asy_rst )
begin
    if ( asy_rst )
        timer_cnt <= 22'd0;
    else
        timer_cnt <= timer_cnt +1'b1;
end


/*------------------------------------------------------------
gen the input_flag signal
------------------------------------------------------------*/
always @ ( posedge r_clk or posedge asy_rst )
begin
    if ( asy_rst )
        input_flag <= 1'b0;
    else if ( timer_cnt < 22'd580 )
        input_flag<= 1'b0;
    else
        input_flag <=1'b1;
end
/*------------------------------------------------------------
gen the data signal
------------------------------------------------------------*/
always @ ( posedge r_clk or posedge asy_rst )
begin
    if ( asy_rst )
        data <= 1'b1;
    else if ( timer_cnt < 22'd550 )
        data<= 1'b0;
    else
        data <=1'b1;
end
```

```
/*------------------------------------------------------------
gen the bus_data_d1,bus_data_d2 signal
------------------------------------------------------------*/
always @ ( posedge clk or posedge asy_rst )
begin
    if ( asy_rst )
    begin
        bus_data_d1 <= 1'b0;
        bus_data_d2 <= 1'b0;
    end
    else
    begin
        bus_data_d1 <= io_data;
        bus_data_d2 <= bus_data_d1;
    end
end

assign bus_posedge = input_flag ? (( {bus_data_d1,bus_data_d2} == 2'b10 ) ? 1'b1 :1'b0 ) : 1'b0;
assign bus_negedge = input_flag ? (( {bus_data_d1,bus_data_d2} == 2'b01 ) ? 1'b1 :1'b0 ) : 1'b0;
```

```
/*------------------------------------------------------------
gen the name signal
------------------------------------------------------------*/
always @ ( posedge r_clk or posedge asy_rst )
begin
    if ( asy_rst )
        r_bus_negedge <= 1'b0;
    else
        r_bus_negedge<= bus_negedge;
end


/*------------------------------------------------------------
gen the timer1 signal
------------------------------------------------------------*/
always @ ( posedge r_clk or posedge asy_rst )
begin
    if ( asy_rst )
        timer1 <= 22'd0;
    else if( bus_posedge )
        timer1 <= timer_cnt;
    else
        timer1 <= timer1;
end
```

```
/*------------------------------------------------------------
gen the bus_value signal
------------------------------------------------------------*/
always @ ( posedge r_clk or posedge asy_rst )
begin
    if ( asy_rst )
        bus_value <= 1'b0;
    else if( bus_negedge )
        bus_value <= input_flag ? ( ( timer_cnt-timer1 < 22'd45 )? 1'b0: 1'b1 ) : 1'b0;
    else
        bus_value <= bus_value;
end


/*------------------------------------------------------------
gen the result signal
------------------------------------------------------------*/
always @ ( posedge r_clk or posedge asy_rst )
begin
    if ( asy_rst )
        result <= 40'd0;
    else if( data_valid && r_bus_negedge )
        result <= { result[38:0], result };
    else
        result <= result;
end
```

```
/*-------------------------------------------------------------
gen the bit_cnt signal
-------------------------------------------------------------*/
always @ ( posedge r_clk or posedge asy_rst )
begin
    if ( asy_rst )
        bit_cnt <= 6'd0;
    else if( data_valid && r_bus_negedge )
        bit_cnt <= bit_cnt + 1'b1;
    else if( bit_cnt == 6'd40 )
        bit_cnt <= 6'd0;
    else
        bit_cnt <= bit_cnt;
end

/*-------------------------------------------------------------
gen the data_valid signal
-------------------------------------------------------------*/
always @ ( posedge r_clk or posedge asy_rst )
begin
    if ( asy_rst )
        data_valid <= 1'b0;
    else if( !input_flag )
        data_valid <= 1'b0;
    else if( bus_negedge )
        data_valid <= 1'b1;
    else
        data_valid <= data_valid;
end

assign o_data_valid = input_flag ? 1'b0 : 1'b1;

endmodule
```

后续交流课程的一些说明

【第四讲】

- SOPC的基本概念与解决方案
- 基于的MicroBlaze的SOPC系统搭建（XPS、SDK）
- 基于NiosII的SOPC系统搭建（SOPC Builder、Qsys、EDS）