**3.1 体温检测模块的设计（撰写人：）**

**3.1.1 IP设计**

（1）IP实现的功能

本模块为体温检测模块，用到的器件型号为Max30205。用户可将手放在体温检测传感器上，经过一段时间稳定后即可测得人体温度。如果体温检测传感器暴露在空气中测得的是室温。

（2）工作原理

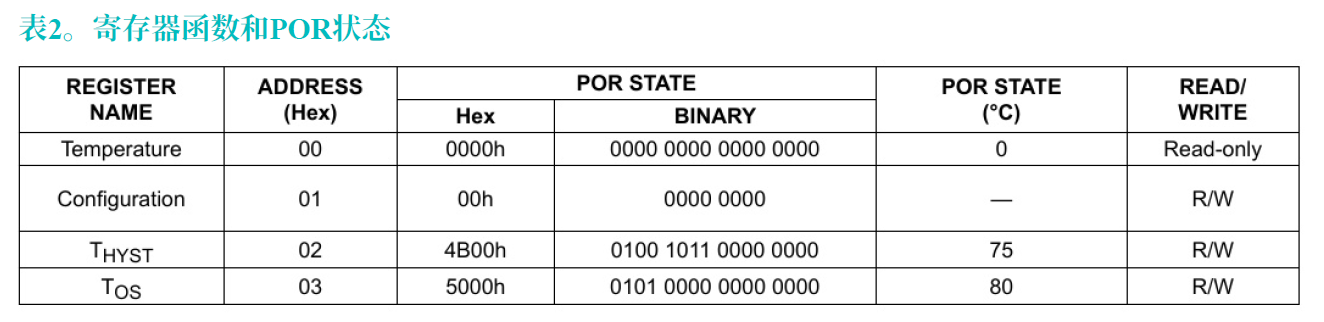
Max30205通过I2C兼容2线串行接口进行通信。

MAX30205具有三条地址选择线，提供总共32个可用地址。本模块将A0,A1,A2三个接口与GND连接，地址为90h（下图截取部分从地址选择的结果）。

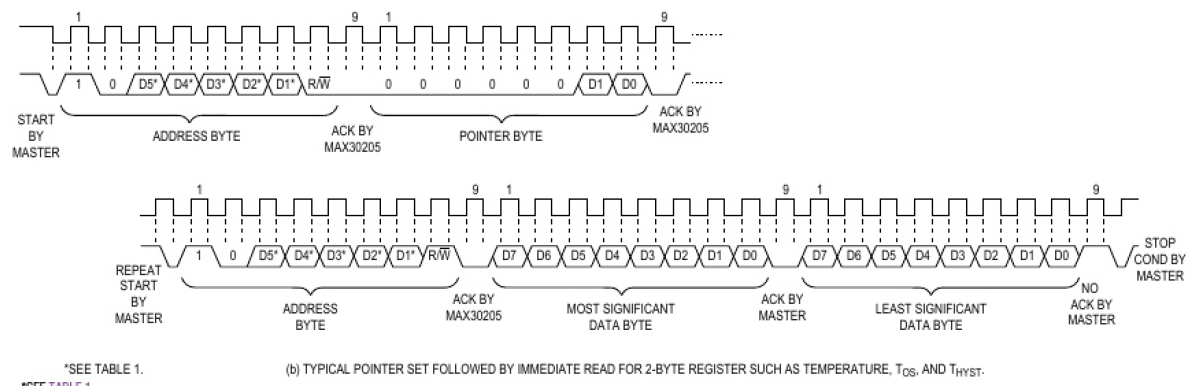


图：Max30205从地址选择（部分）

元器件内部有四个寄存器，在读取温度时只需要用到第一个Temperature寄存器，故内部寄存器选择的8位指针最低两位为00（如下图）。



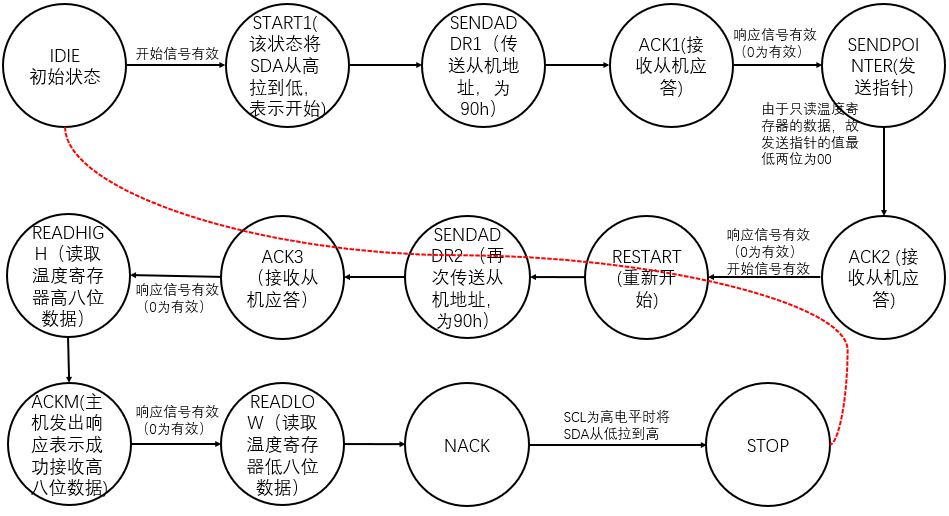
图：Max30205内部的寄存器，其中第一个为温度寄存器，存储温度值

读取Temperature温度寄存器数据的时序图如下：

图：读取温度寄存器数据的时序图

由上图可知读取温度寄存器中数据的整个过程如下：

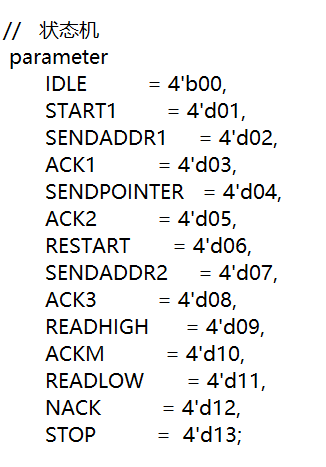
启动元器件时处于IDIE状态，当主机发送start信号后，进入START1状态：在SCL为高电平时，将SDA信号从高拉到低，表示IIC开始通信。接下来进入SENDADDR1状态，发送Max30205元器件的地址（如上面所说为90h，即7’b1001\_000，最后一位由于此时相当于发送写的地址，故值为0），然后进入ACK1状态，SDA线由从机Max30205控制，当从机发送值为0的ACK响应信号后，进入SENDPOINTER状态，主机接下来发送元器件内寄存器选择地址（如上面所说最低两位为00，即整个POINTER BYTE 值为8’b0000\_0000），接下来进入ACK2状态，SDA线由从机Max30205控制，当从机发送值为0的ACK响应信号后，主机重新发送开始信号，此时为RESTART状态，再次将SDA从高拉到低，接下来进入SENADDR2状态，再次发送从机地址（即90h：7’b1001\_000，最后一位由于此时相当于发送读的地址，故值为1），然后进入ACK3状态，SDA线由从机Max30205控制，当从机发送值为0的ACK响应信号后，进入READHIGH状态，从机开始发送温度寄存器高八位的值，发送完毕后进入ACKM状态，主机发送值为0的响应信号，表示成功接收到从机发来的高八位数据，然后进入READLOW状态，从机发送温度寄存器低八位的数据，发送完毕后进入NACK状态，主机希望结束数据传输，于是向从机发送非应答（NACK）信号，此时从机已把温度寄存器所有数据传输完成，进入STOP状态，在SCL为高电平时把SDA从低拉到高，整个数据读取完毕，准备重复上述循环。

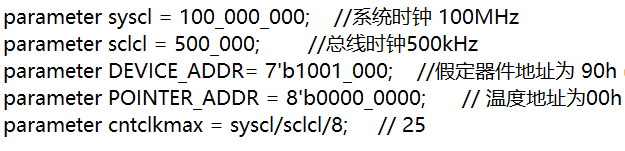
状态转移图如下：

图：读取温度寄存器数据的状态转移图

（3）电路设计

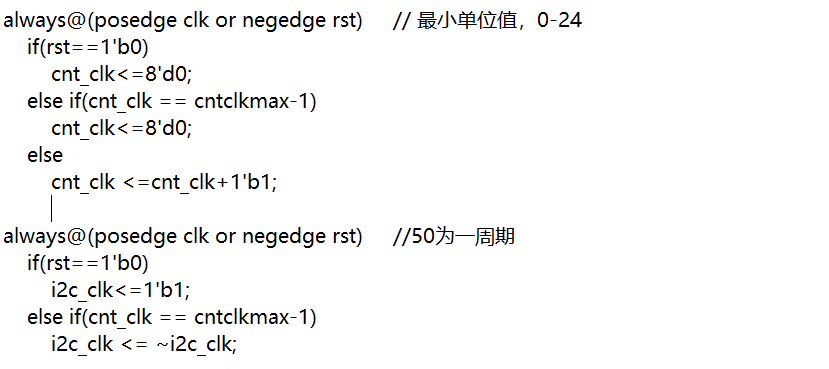
状态转移图如上，下图为Verilog代码中对状态的定义，与转移图中一致。



图：Verilog代码中状态定义

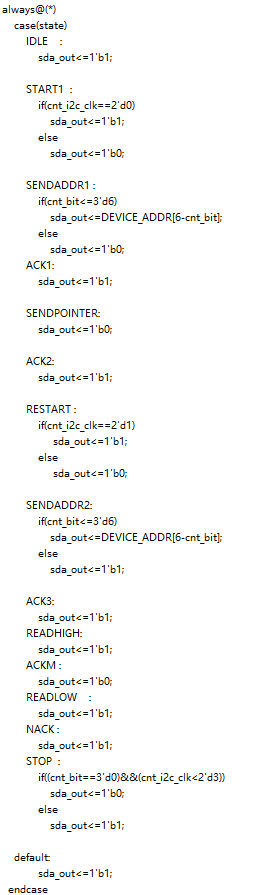
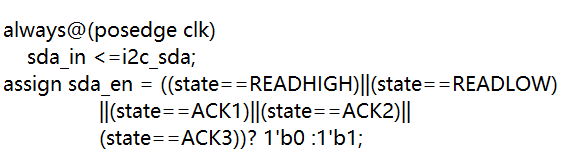
图：系统变量定义

上图为用到的模块系统变量。其中设置系统变量为100MHz，总线时钟设置为500kHz，器件地址为90h，温度寄存器地址为00h。由于系统周期过小，故不直接使用系统周期，而是额外定义i2c工作周期。其中的cntclkmax用来记录i2c的clk周期，设置为50个系统clk为一个i2c周期。如下图：

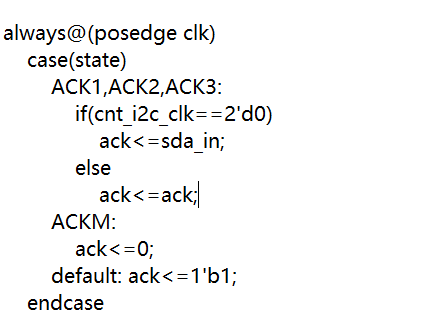


图：系统周期和i2c工作周期设置

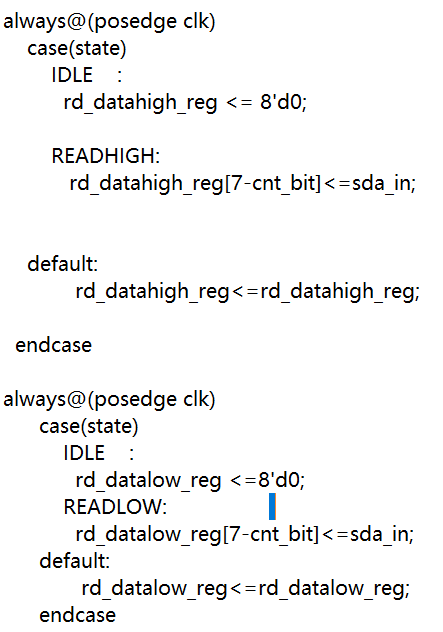
I2C有SDA数据线和SCL时钟线，其中的SDA为双向inout接口，所以在代码中引入reg 型 sda\_in 和 sda\_out ，其值设定如下：



图：sda\_in信号和sda\_en信号赋值



图：ack信号赋值



图：sda\_out数据赋值 图：在读取从机数据阶段把sda线上数据存储起来

assign i2c\_sda = (sda\_en == 1’b1)? sda\_out:1’bz;

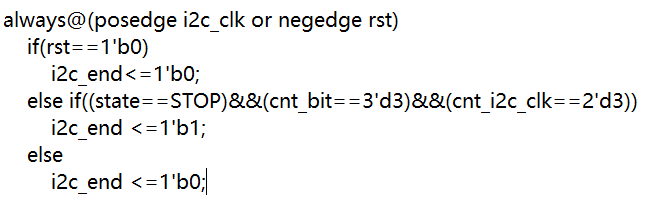
对于sda\_en，当主机不掌控SDA线时应为低，主机掌控时应为高，故给其赋值按上图所示，在READHIGH,READLOW,ACK1,ACK2,ACK3状态给sda\_en赋值为0，其他状态为1。

对于sda\_out：由上式可知，在主机掌控SDA线时，i2c\_sda的数据就是sda\_out中的数据。sda\_out在主机有效态IDLE时，由于此时还未开始I2C传输，故SDA和SCL均保持高电平；在START1和RESTART时，I2C开始传输，此时要让SCL在高电平时将SDA从高电平拉到低电平，故使sda\_out在开始的第一个clk时为高，后面拉低；在SENDADDR1和SENDADDR2时，都是主机通过SDA线发送从机地址（7位的90h），由于状态为0时应该是先发高位（第7位，数组下标为6），故要把顺序反转一下，即在前七个clk时sda\_out<=DEVICE\_ADDR[6-cnt\_bit]; 在第八个clk时根据时序图，在SENDADDR1时应该是发送写信号（这时实际不进行写入，只是为了确认从机地址），即0，故else 时 sda\_out<=1'b0; 而SENDADDR2第二次发送从机地址时应该是说明要进行读操作，故发送的信号为1，故代码中 else 时sda\_out<=1'b1; 在SENDPOINTER状态也应该进行类似的操作，这里由于前六位全为0，且最后两位指向温度寄存器的位置，即00，故八位全为0，可以直接写为sda\_out<=1'b0; 在ACKM状态时由时序图，主机要发送值为0的响应信号表示已经成功读取到了从机传送过来的高八位数据，故 sda\_out值始终为0；在NACK状态时由时序图可知此时主机要发送值为1的NACK信号用于表示已经完成了数据传输，故这个状态sda\_out值为1；在STOP状态要结束I2C的通信，即要在SCL为高电平时把SDA从低拉到高，故在开始时sda\_out为0，接下来变成1，结束了I2C通信。

对于sda\_in，其始终与i2c\_sda上的数据保持一致，但是只有在从机Max30205有效时的状态才会用到这个数据。其使用见下：

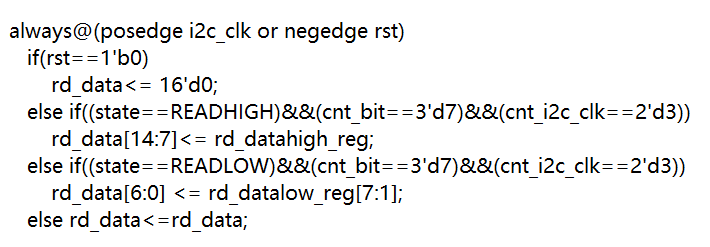
特别设置中间变量ack用来表示主机从机发送的响应信号，在状态转换时使用，如从ACK1状态转变为下一个SENDPOINTER状态所需的条件之一即ack为0。显然该变量在从机发送响应信号时才有效（即在这些状态才会使用到ack信号）。故让ack信号在ACK1,ACK2,ACK3状态(即从机发送响应信号的状态)时等于sda\_in(间接等于i2c\_sda，此时从机在掌握SDA线，这个数据不在协议代码中体现，而是实际硬件发出的实际值)。

同时在读取数据的状态，即READHIGH和READLOW状态，此时从机在掌握SDA线，会把实际寄存器中的值发送过来，故此时的sda\_in(等于i2c\_sda)即为所需的数据，在这两个状态要把读取到的数据存储住。即如上图所写，在READHIGH状态，把sda上的数据一位一位地送到rd\_datahigh\_reg。注意由于第一个时钟从机发送的是最高位，故要把写入顺序颠倒一下。READLOW状态做同样处理。



图：i2c\_end信号赋值

一次读取结束的信号i2c\_end赋值。（仿真用）

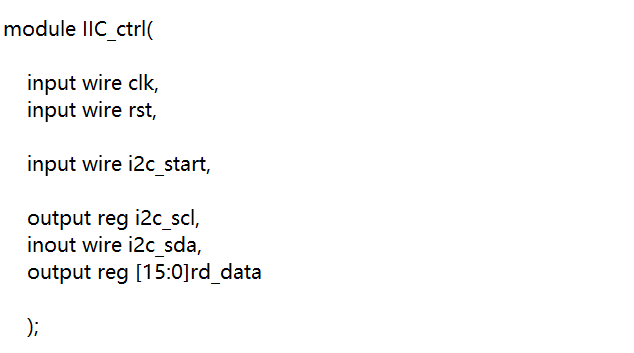


. 图：将中间值合并为16位输出

上图为将两个8位的中间变量rd\_datahigh\_reg和rd\_datalow\_reg经过移位后赋值给十六位输出，作为从从机读取到的温度寄存器的值，后续对该值进行读取及处理。

（4）IP封装

将上面的IIC控制代码封装为IP，这里需要两个寄存器，第一个寄存器用于后面软件控制使用IIC开始传输。第二个寄存器就用来存放从硬件Max30205温度寄存器中读取的16位数据。如下图：

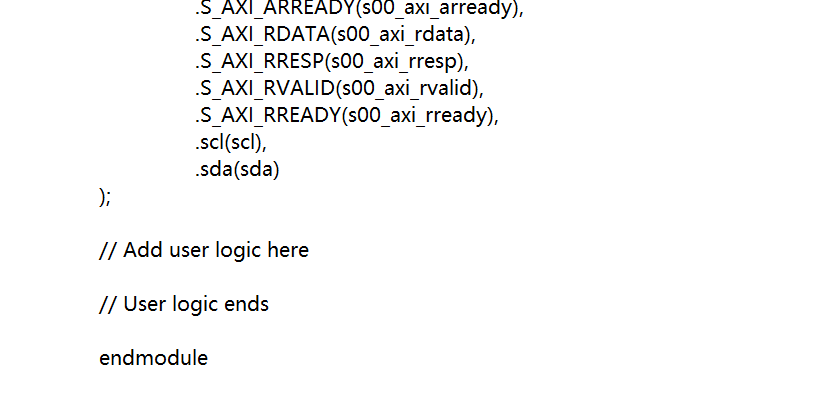


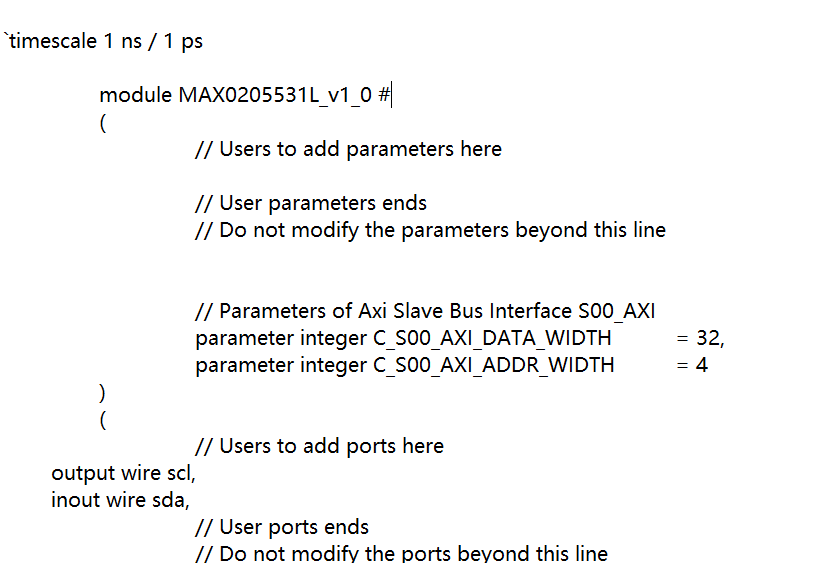
图：IIC控制模块的输入输出

由上图，i2c\_scl和i2c\_sda需要作为IIC固定的线引出，故分配引脚，clk接到板子的时钟引脚，剩下的控制IIC模块读取开始的i2c\_start分配寄存器进行控制，而rd\_data即为需要读取的数据，需要分配寄存器放进数据。

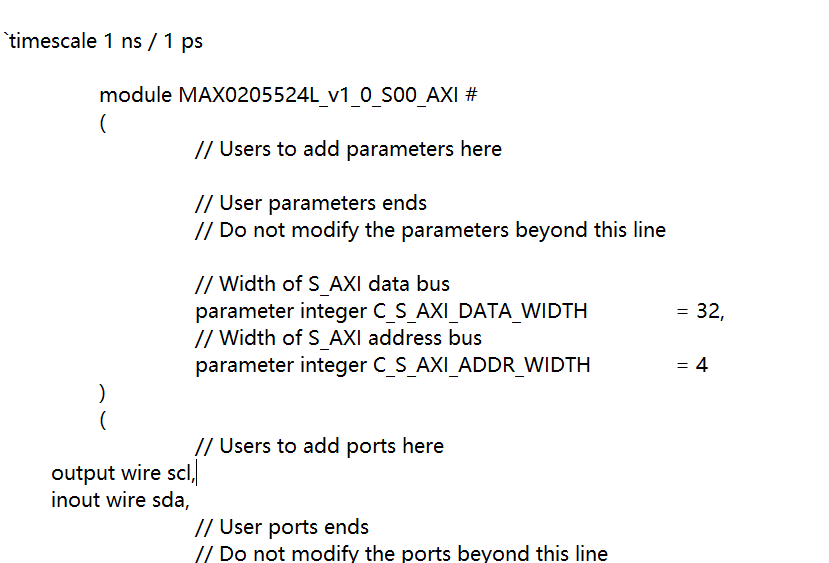


图：封装后的IP(名称为MAX30205531)

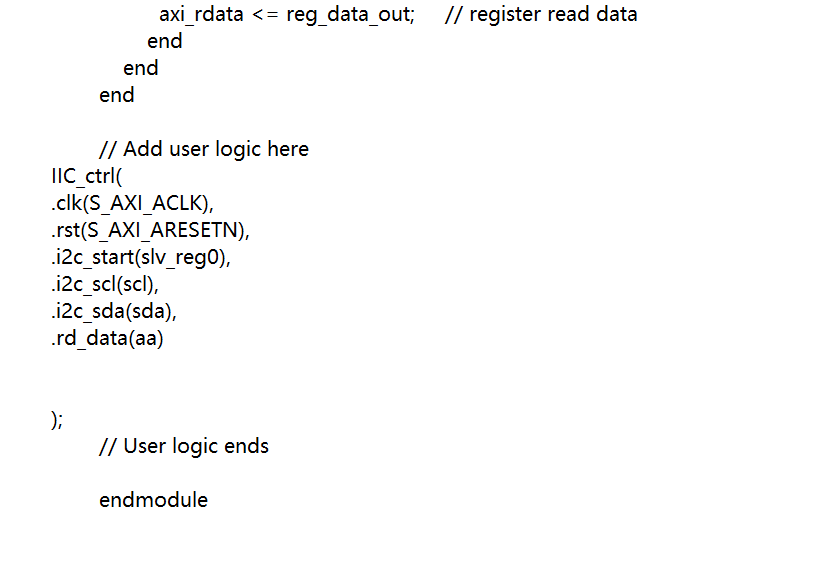




图：封装IP时的模块文件需要添加scl，sda两个信号

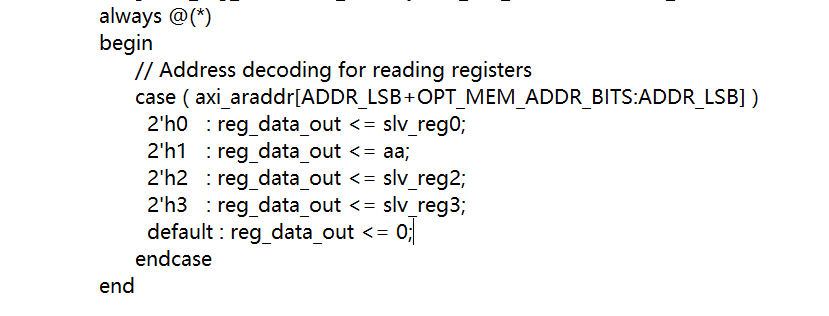


图：实例化封装模块的文件时添加scl，sda信号



图：调用IIC\_ctrl模块时对输入输出信号实例化

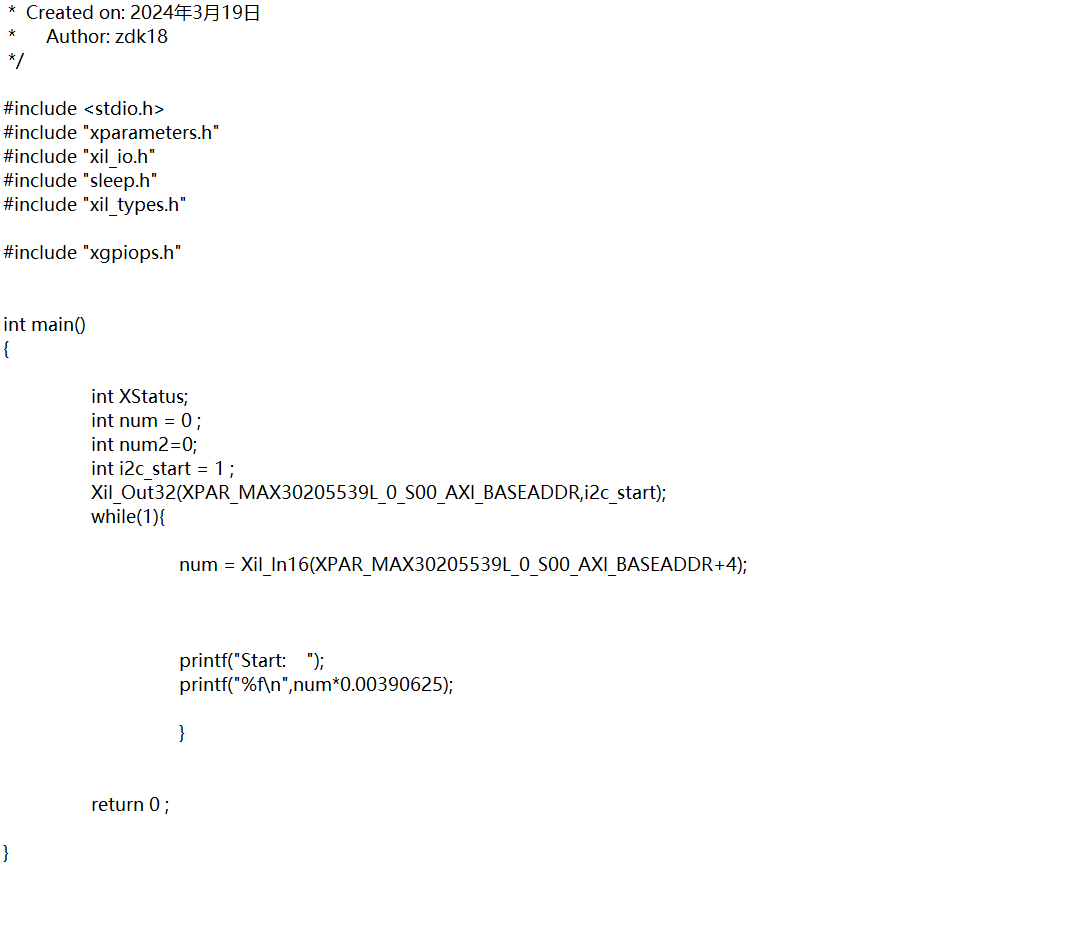
如上图：在调用IIC\_ctrl模块时需要对输入输出信号进行实例化：将第一个寄存器中的值作为i2c\_start，即IIC模块的起始信号。将输出的值先赋值给中间wire型变量aa，再把aa的值赋值给第二个寄存器。如下图：



图：把中间变量aa的值赋值给第二个寄存器

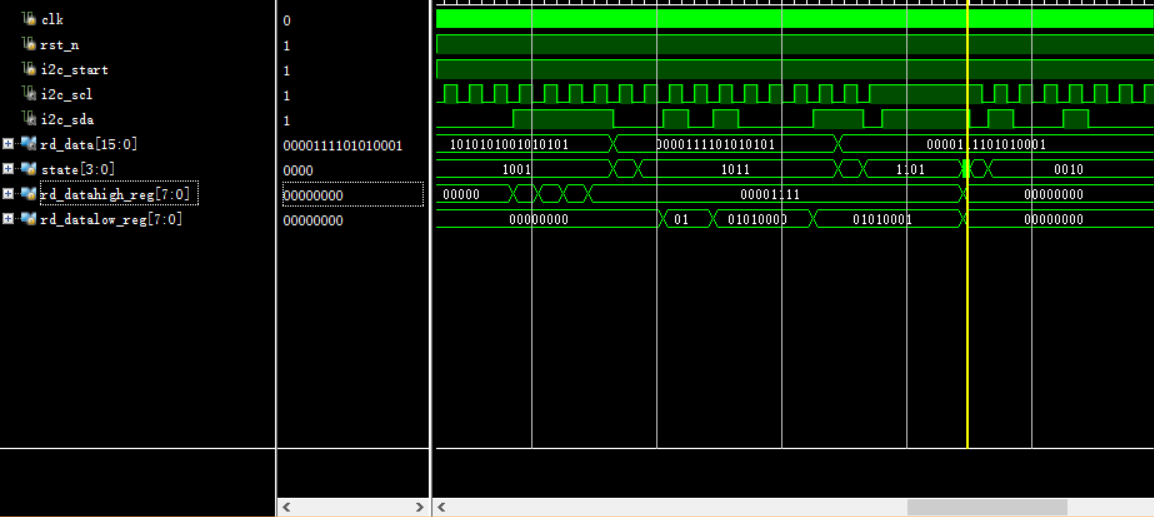
封装后的IP名称为MAX30205531。

**3.1.2 软件设计**

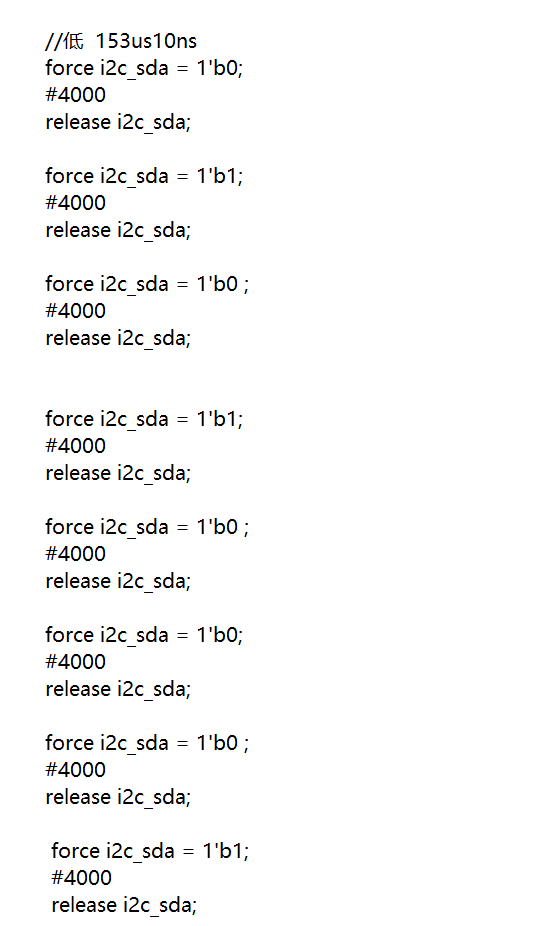


图：软件部分代码

软件部分代码较为简单，开始时给第一个寄存器赋值为1，即i2c\_start为1，故IIC模块可以通信，接下来循环从第二个寄存器中读值，这个值就是Max30205温度寄存器中的值，把这个值乘上0.00390625即为所需的真实的温度值，把它打印到屏幕上，可以看出大约是25到35摄氏度，符合真实情况。用num寄存住这个数据，便于与其他模块部分进行传输通信。

**3.1.3 模块测试**

图：IIC模块测试仿真图



图：IIC模块仿真代码

对于IIC模块进行仿真测试：在READHIGH阶段模拟从机发出八位数据8’b0000\_1111，在READLOW阶段模拟从机发出八位数据8’b0101\_0001，由仿真图可知数据成功赋值给了输出rd\_data，故IIC模块正确（即硬件正确）。

软件模块进行测试，将Max30205连接上显示屏看是否成功读取到数据：



图：连接上其他模块后检测到可以读取到数据

由上图可知当连接上显示屏后成功读取到了合理的数据，可认为软硬件设计都无问题。