模拟量接口实验

1. 概述

本实验将练习如何使用Nexys4 DDR开发板上的ADXL362加速计和ADT7420温度传感器，从而掌握模块量接口技术。

1.1 ADXL362加速计规范

Nexys4 DDR开发板包括一个模拟器件ADXL362加速计。有关该器件的完整信息，请参看ADXL362数据手册。

ADXL362是一款3轴MEMS加速计，在100Hz输出数据速率下的功耗不到2μA，在运动触发唤醒模式下的功耗为270nA。 该加速计拥有12位输出分辨率，但同时也提供8位格式化数据，以便在低分辨率即可满足要求时采用更高效的单字节传输。 测量范围为±2g、±4g和±8g，其中±2g范围内的分辨率为1mg/LSB。当ADXL362处于测量模式时，将连续测量加速度数据并将其存储在X数据、Y数据和Z数据寄存器中。

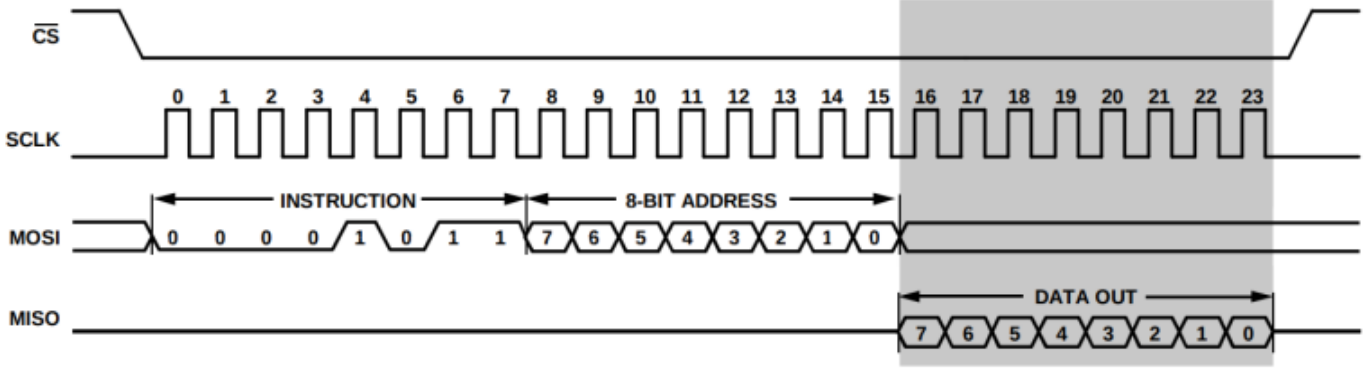
ADXL362加速计包括多个寄存器（如下表所示），用户可以使用这些寄存器对其进行配置以及读取加速度数据。 写入控制寄存器可以配置加速计，读取器件寄存器可以获取加速计数据。与该器件通信时，必须指定一个寄存器地址以及一个用于指示通信是读操作还是写操作的标志。 当寄存器地址与通信标志发送到器件后，即可开始传输数据。

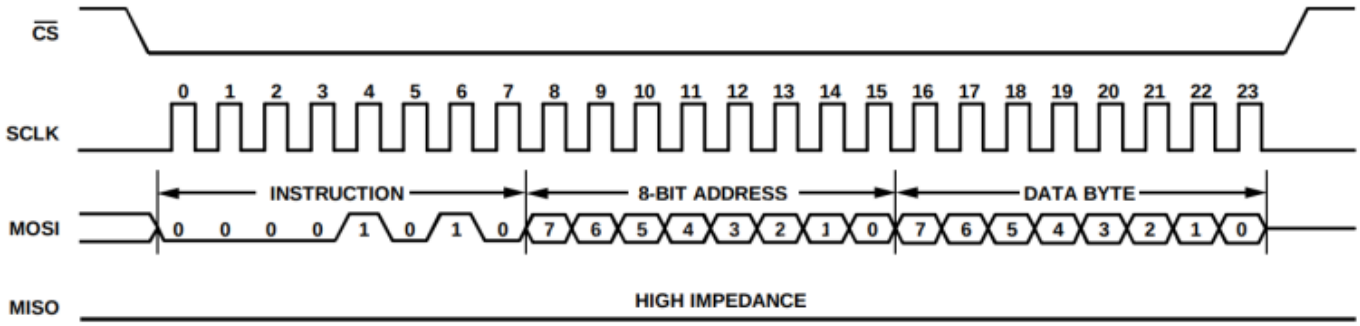
Reg	Name	Bits	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset	RW
0x00	DEVID_AD	[7:0]	DEVID_AD[7:0]								0xAD	R
0x01	DEVID_MST	[7:0]	DEVID_MST[7:0]								0x1D	R
0x02	PARTID	[7:0]	PARTID[7:0]								0xF2	R
0x03	REVID	[7:0]	REVID[7:0]								0x01	R
0x08	XDATA	[7:0]	XDATA[7:0]								0x00	R
0x09	YDATA	[7:0]	YDATA[7:0]								0x00	R
0x0A	ZDATA	[7:0]	ZDATA[7:0]								0x00	R
0x0B	STATUS	[7:0]	ERR_USER_REGS	AWAKE	INACT	ACT	FIFO_OVER-RUN	FIFO_WATER-MARK	FIFO_READY	DATA_READY	0x40	R
0x0C	FIFO_ENTRIES_L	[7:0]	FIFO_ENTRIES_L[7:0]								0x00	R
0x0D	FIFO_ENTRIES_H	[7:0]	UNUSED						FIFO_ENTRIES_H[1:0]		0x00	R
0x0E	XDATA_L	[7:0]	XDATA_L[7:0]								0x00	R
0x0F	XDATA_H	[7:0]	SX				XDATA_H[3:0]				0x00	R
0x10	YDATA_L	[7:0]	YDATA_L[7:0]								0x00	R
0x11	YDATA_H	[7:0]	SX				YDATA_H[3:0]				0x00	R
0x12	ZDATA_L	[7:0]	ZDATA_L[7:0]								0x00	R
0x13	ZDATA_H	[7:0]	SX				ZDATA_H[3:0]				0x00	R
0x14	TEMP_L	[7:0]	TEMP_L[7:0]								0x00	R
0x15	TEMP_H	[7:0]	SX				TEMP_H[3:0]				0x00	R
0x16	Reserved	[7:0]	Reserved[7:0]								0x00	R
0x17	Reserved	[7:0]	Reserved[7:0]								0x00	R
0x1F	SOFT_RESET	[7:0]	SOFT_RESET[7:0]								0x00	W
0x20	THRESH_ACT_L	[7:0]	THRESH_ACT_L[7:0]								0x00	RW
0x21	THRESH_ACT_H	[7:0]	UNUSED					THRESH_ACT_H[2:0]			0x00	RW
0x22	TIME_ACT	[7:0]	TIME_ACT[7:0]								0x00	RW
0x23	THRESH_INACT_L	[7:0]	THRESH_INACT_L[7:0]								0x00	RW
0x24	THRESH_INACT_H	[7:0]	UNUSED					THRESH_INACT_H[2:0]			0x00	RW
0x25	TIME_INACT_L	[7:0]	TIME_INACT_L[7:0]								0x00	RW
0x26	TIME_INACT_H	[7:0]	TIME_INACT_H[7:0]								0x00	RW
0x27	ACT_INACT_CTL	[7:0]	RES		LINKLOOP		INACT_REF	INACT_EN	ACT_REF	ACT_EN	0x00	RW
0x28	FIFO_CONTROL	[7:0]	UNUSED				AH	FIFO_TEMP	FIFO_MODE		0x00	RW
0x29	FIFO_SAMPLES	[7:0]	FIFO_SAMPLES[7:0]								0x80	RW
0x2A	INTMAP1	[7:0]	INT_LOW	AWAKE	INACT	ACT	FIFO_OVER-RUN	FIFO_WATER-MARK	FIFO_READY	DATA_READY	0x00	RW
0x2B	INTMAP2	[7:0]	INT_LOW	AWAKE	INACT	ACT	FIFO_OVER-RUN	FIFO_WATER-MARK	FIFO_READY	DATA_READY	0x00	RW
0x2C	FILTER_CTL	[7:0]	RANGE		RES	HALF_BW	EXT_SAMPLE	ODR			0x13	RW
0x2D	POWER_CTL	[7:0]	RES	EXT_CLK	LOW_NOISE		WAKEUP	AUTOSLEEP	MEASURE		0x00	RW
0x2E	SELF_TEST	[7:0]	UNUSED							ST	0x00	RW

该加速计是采用SPI通信方案的外设。推荐的SPI时钟频率范围为1-5MHz。SPI以SPI模式0（CPOL=0、且CPHA=0）工作。SPI端口采用多字节结构，其中第一个字节指示通信是执行寄存器读操作（0x0B）还是寄存器写操作（0x0A）。SPI数据格式如下图所示。

```
<CS down> <Write/Read (0x0A/0x0B)> <address byte> <data byte> <CS up>
```

下图两张图举例说明了SPI控制器（控制器）与加速计（外设）之间的通信，图一给出了寄存器读操作，图二给出了寄存器写操作。





1.2 ADT7420温度传感器

Nexys4 DDR开发板还包括一个模拟器件ADT7420温度传感器。有关该器件的完整信息，请参看ADT7420数据手册。

ADT7420是一款4mm×4mm LFCSP封装高精度数字温度传感器，可在较宽的工业温度范围内提供突破性的性能。它内置一个带隙温度基准源、一个温度传感器和一个16位ADC，用来监控温度并进行数字转换，分辨率为0.0078℃。默认ADC分辨率设置为13位(0.0625℃)。ADC分辨率为用 户可编程模式，可通过串行接口更改。

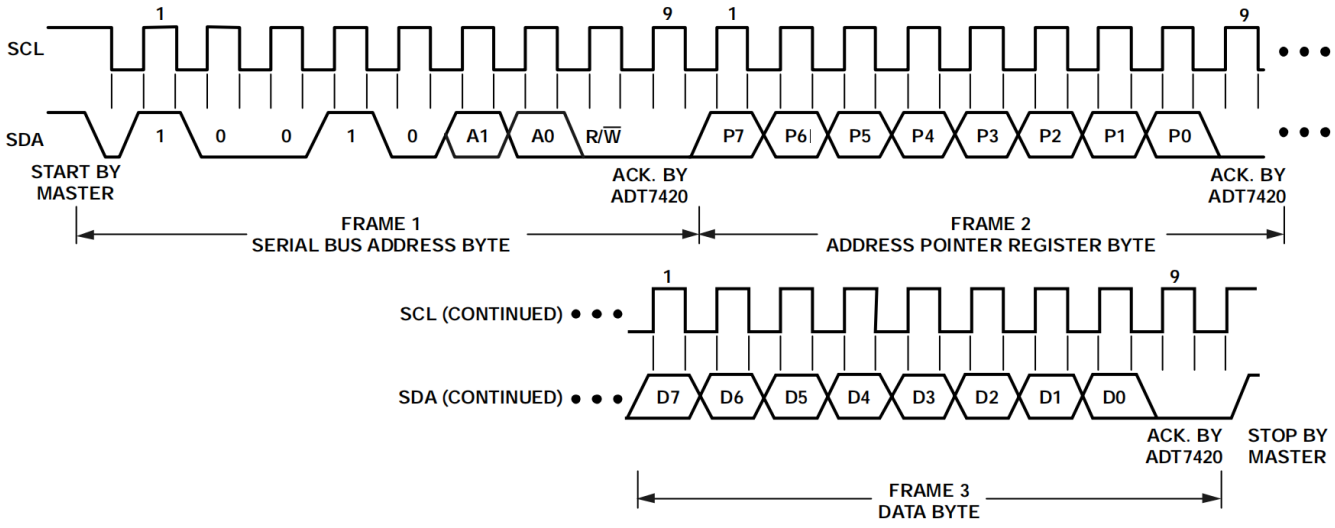
ADT7420的引脚A0和A1用于地址选择，可为ADT7420提供四个I2C地址。CT引脚属于开漏输出，当温度超过临界温度限值(可编程)时，该引脚变为有效。INT引脚也属于开漏输出，当温度超过限值(可编程)时，该引脚变为有效。INT引脚和CT引脚可在比较器模式和中断事件模式下工作。

ADT7420内置14个寄存器（如下图所示）：

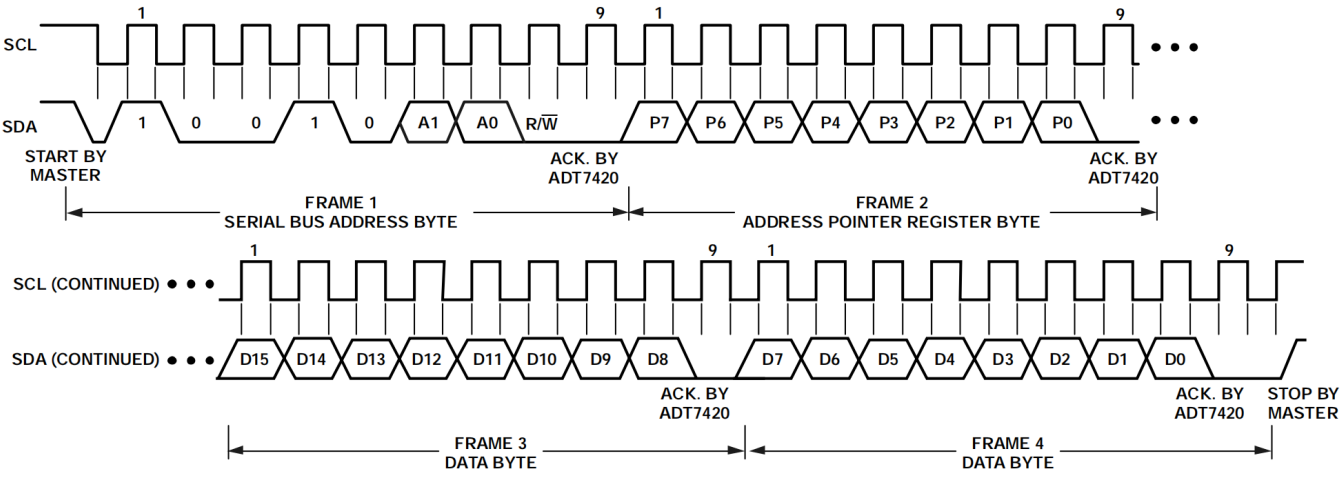
- 9个温度寄存器
- 1个状态寄存器
- 1个ID寄存器
- 1个配置寄存器
- 1个地址指针寄存器
- 1个软件复位

寄存器地址	描述	上电默认值
0x00	温度值最高有效字节	0x00
0x01	温度值最低有效字节	0x00
0x02	状态	0x00
0x03	配置	0x00
0x04	T_{HIGH}设定点最高有效字节	0x20 (64°C)
0x05	T_{HIGH}设定点最低有效字节	0x00 (64°C)
0x06	T_{LOW}设定点最高有效字节	0x05 (10°C)
0x07	T_{LOW}设定点最低有效字节	0x00 (10°C)
0x08	T_{CRIT}设定点最高有效字节	0x49 (147°C)
0x09	T_{CRIT}设定点最低有效字节	0x80 (147°C)
0x0A	T_{HYST}设定点	0x05 (5°C)
0x0B	ID	0xCB
0x2F	软件复位	0xXX

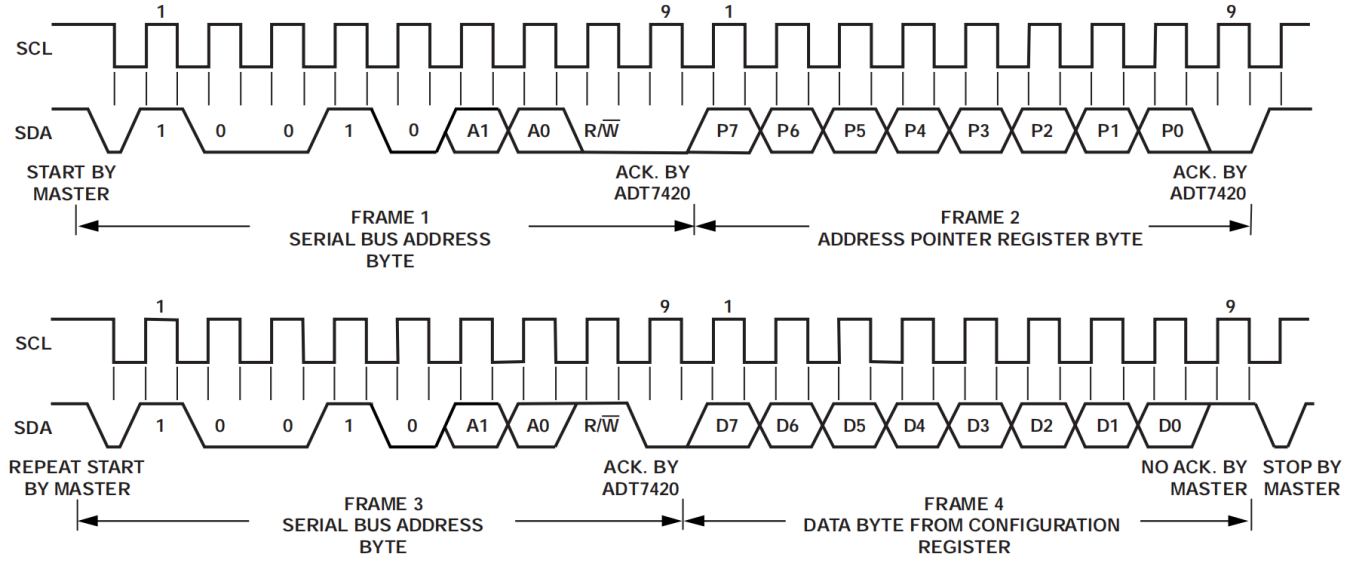
该温度传感器是采用I2C通信方案的外设。其写入一个寄存器、后跟单字节数据的I2C格式如下图所示。



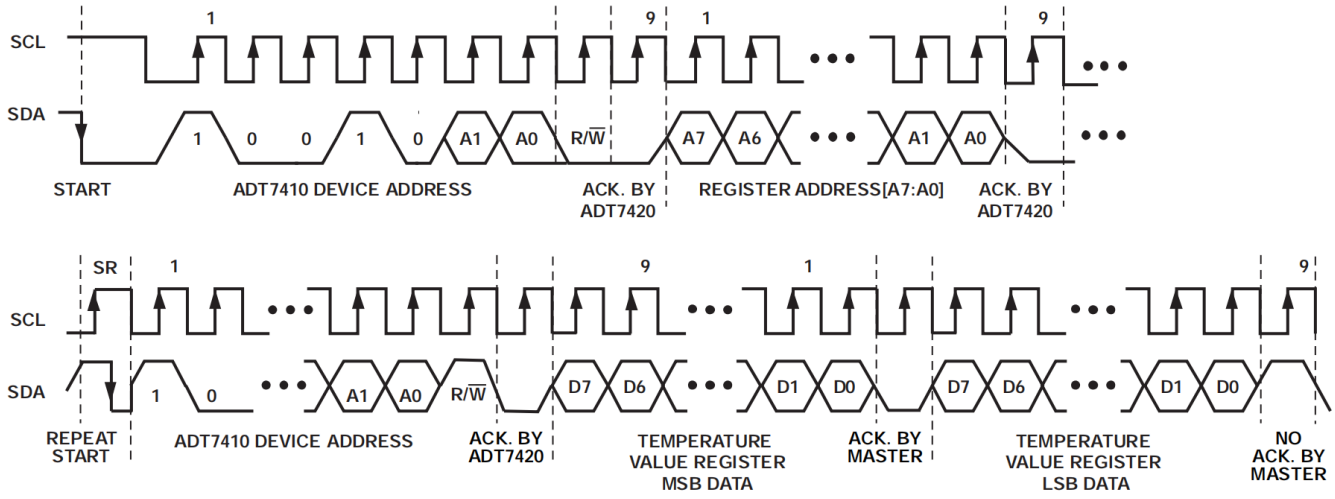
写入一个寄存器、后跟两字节数据的I2C格式如下图所示。



从配置寄存器读回数据的I2C格式如下图所示。



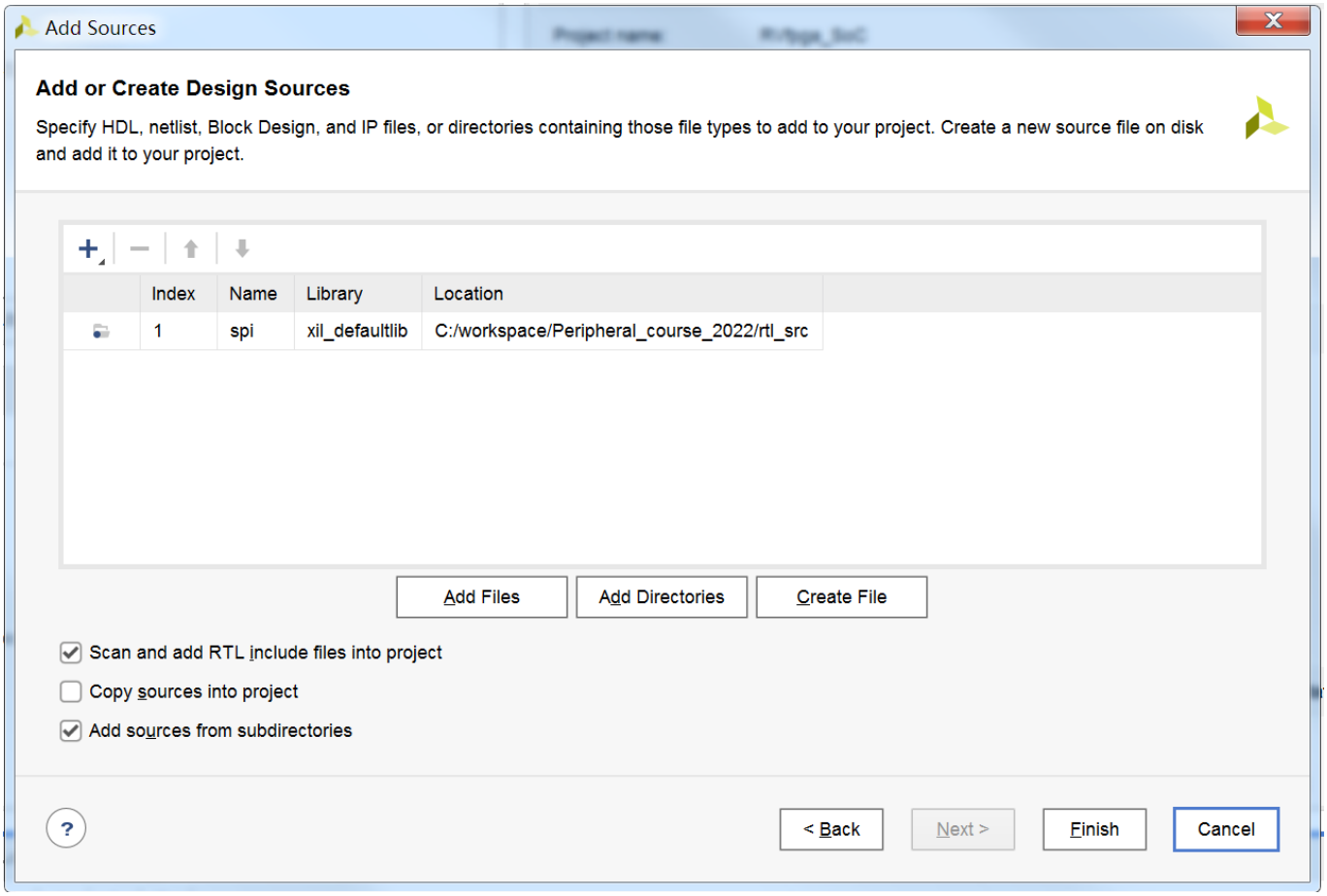
从温度值寄存器读回数据的I2C格式如下图所示。



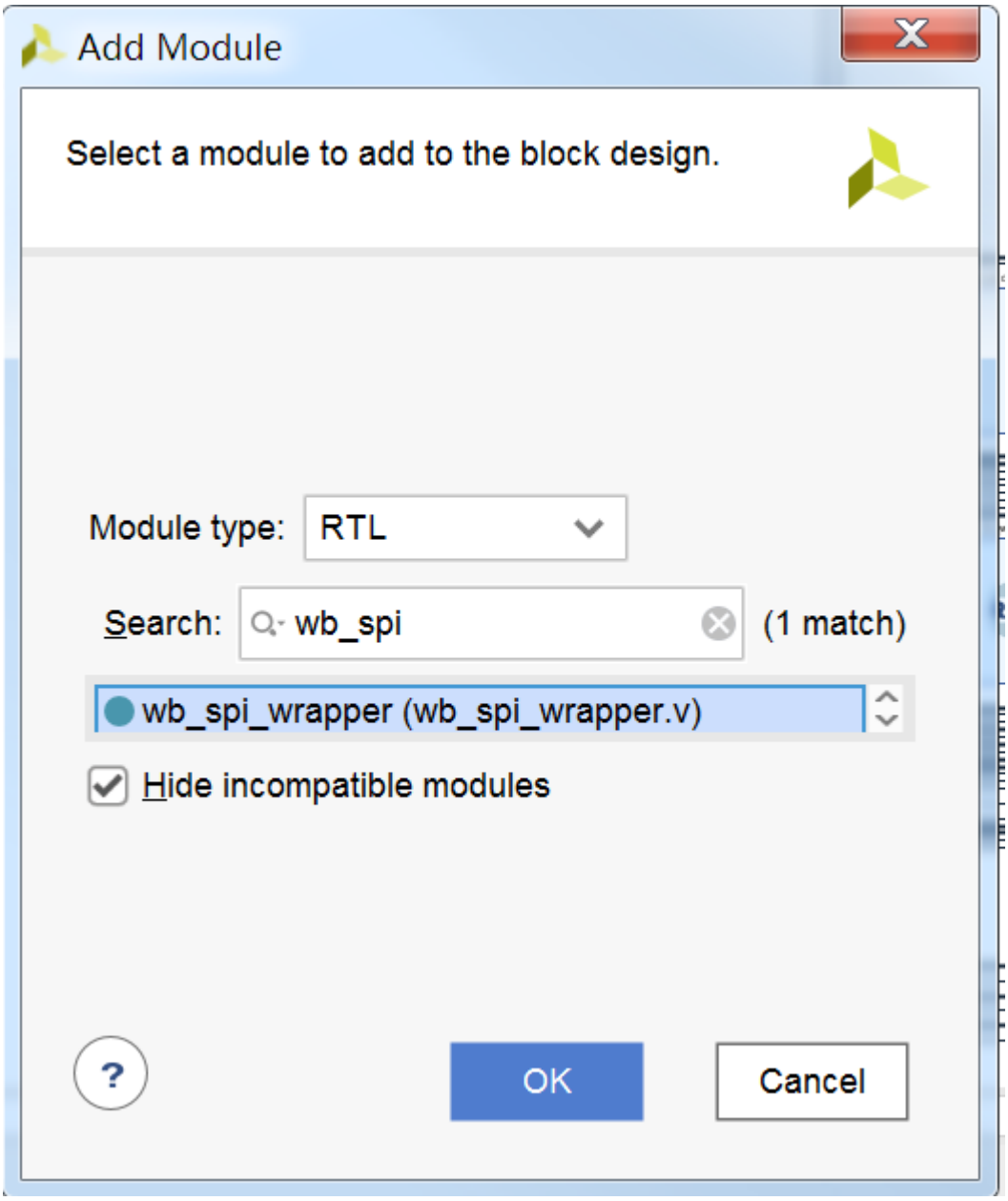
2. RVfpga_SoC模拟量接口实验

2.1 修改RVfpga_SoC硬件

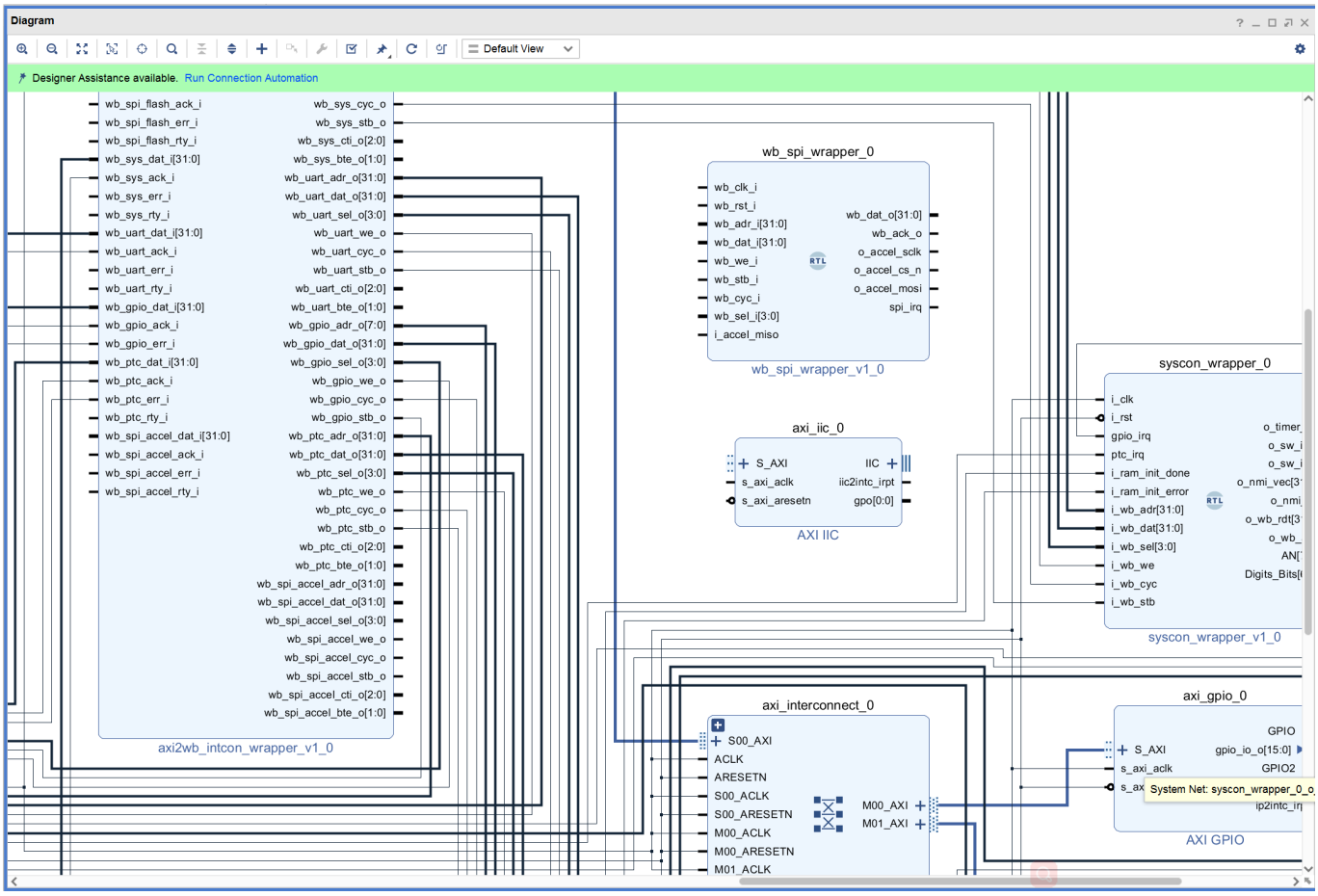
启动Vivado，打开实验7的工程。在“Project Manager”（项目管理）中选择“Add Sources”（添加源文件），在“Add Sources”（添加源文件）窗口中，单击“Add Directories”（添加目录），将spi目录添加到工程，如下图所示。



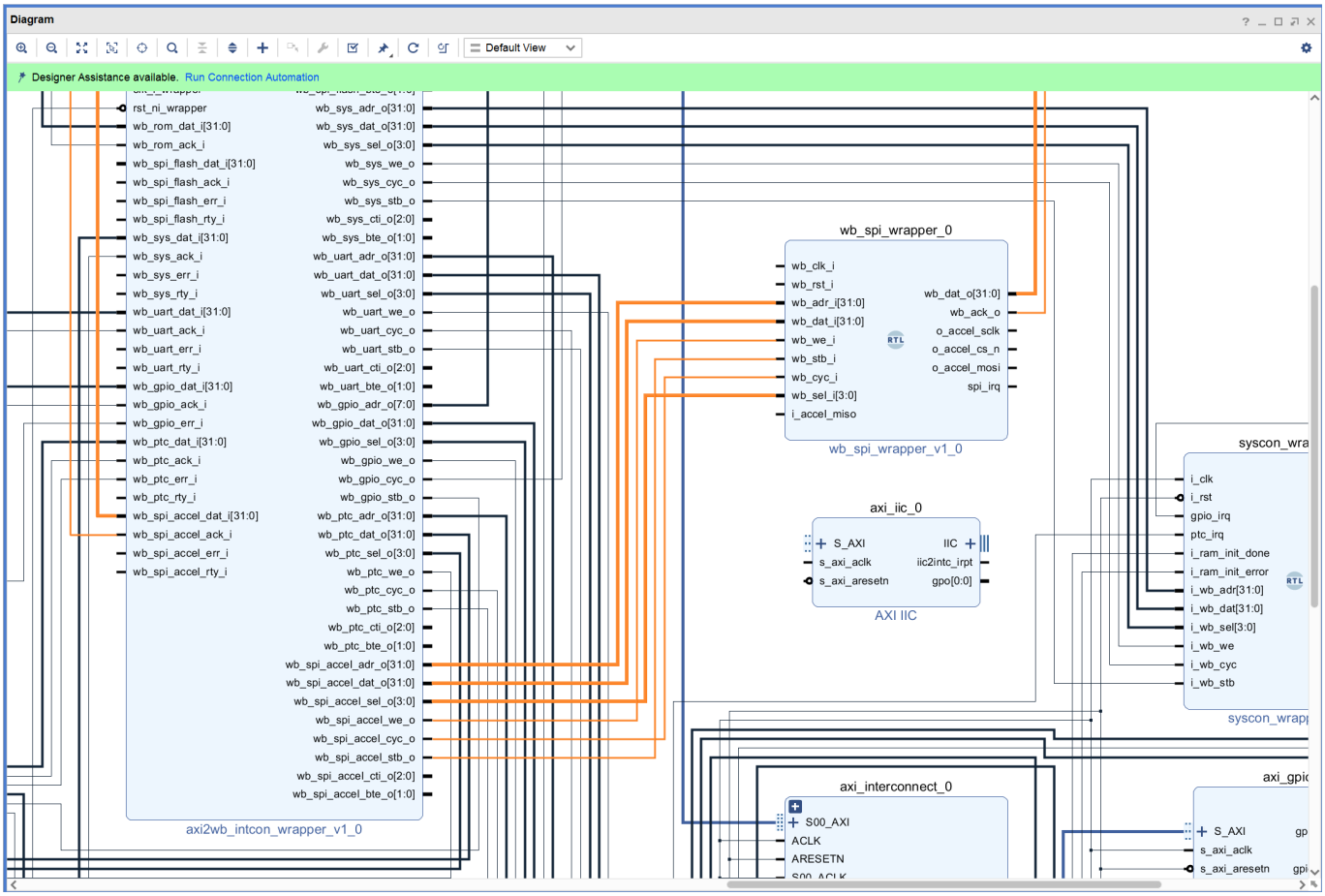
点击“Open Block Design”打开块设计，如下图所示，通过“Add Module”将wb_spi_wrapper模块添加到块设计。



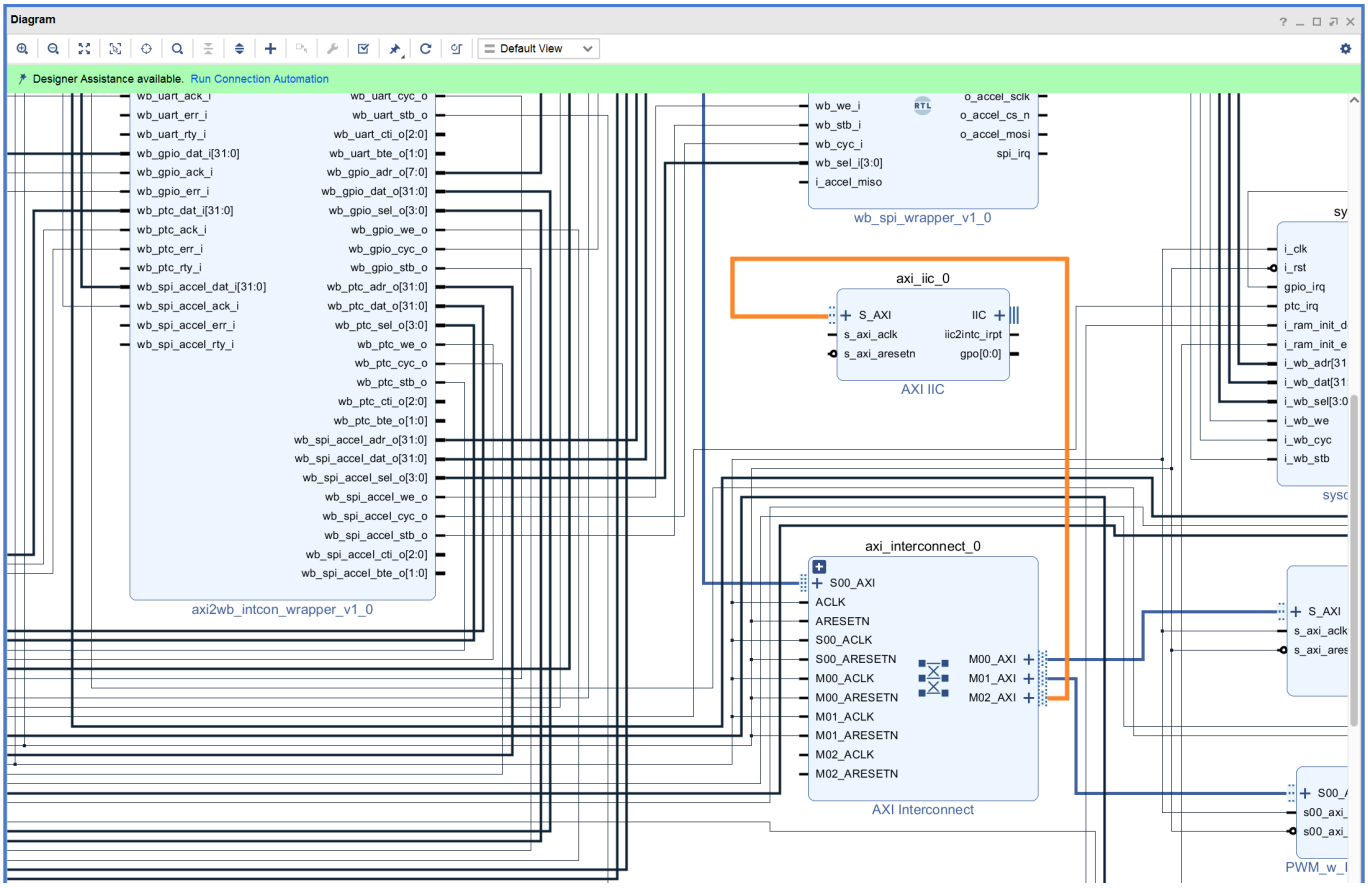
点击“Add IP”添加一个AXI IIC模块，添加后的块设计如下图所示。



将wb_spi_warpper_0模块连接到块设计，如下图所示。

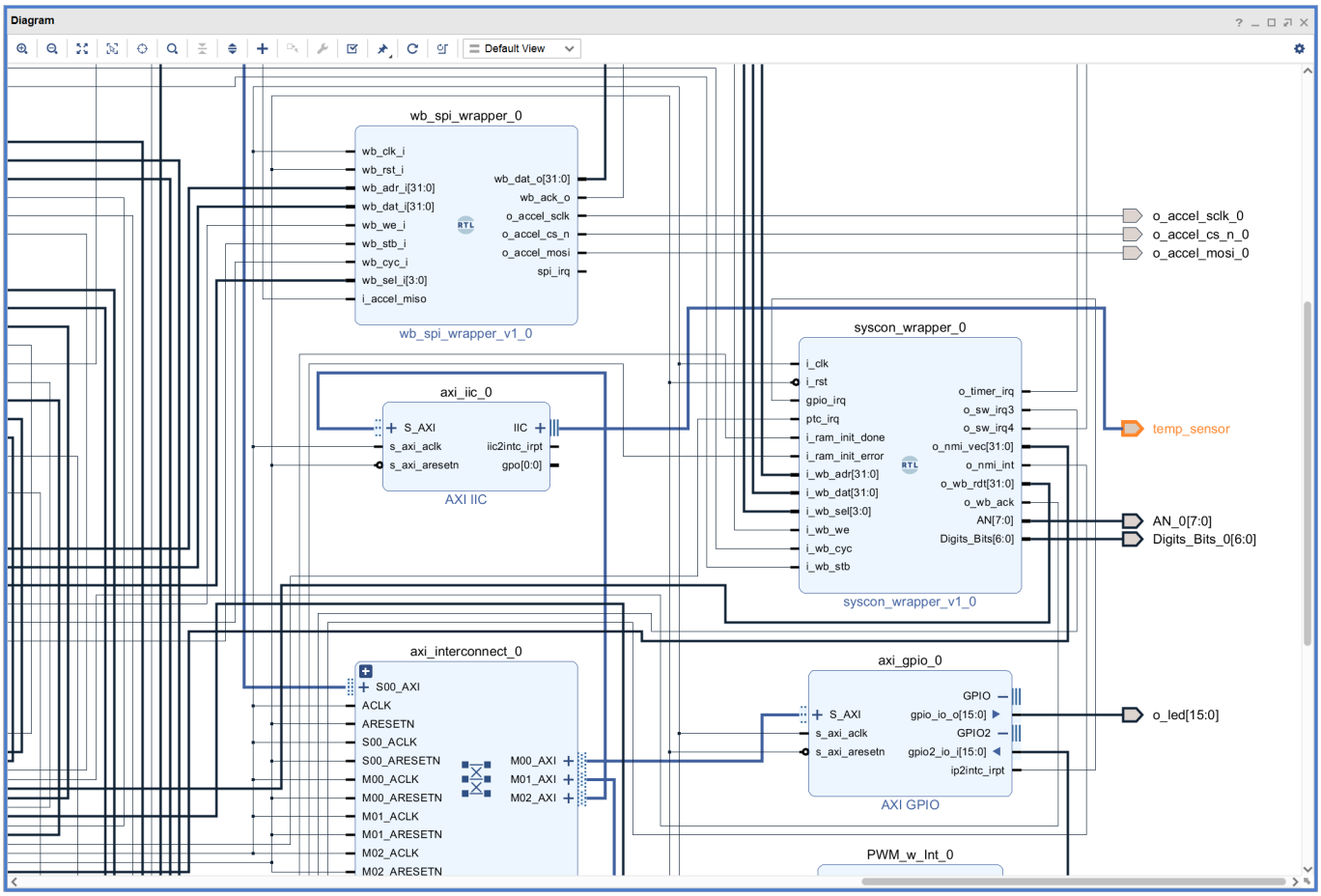


双击axi_interconnect_0模块，在互连模块上增加一个AXI的主端口，然后将axi_iic_0模块连接到新增加的AXI4端口，如下图所示。

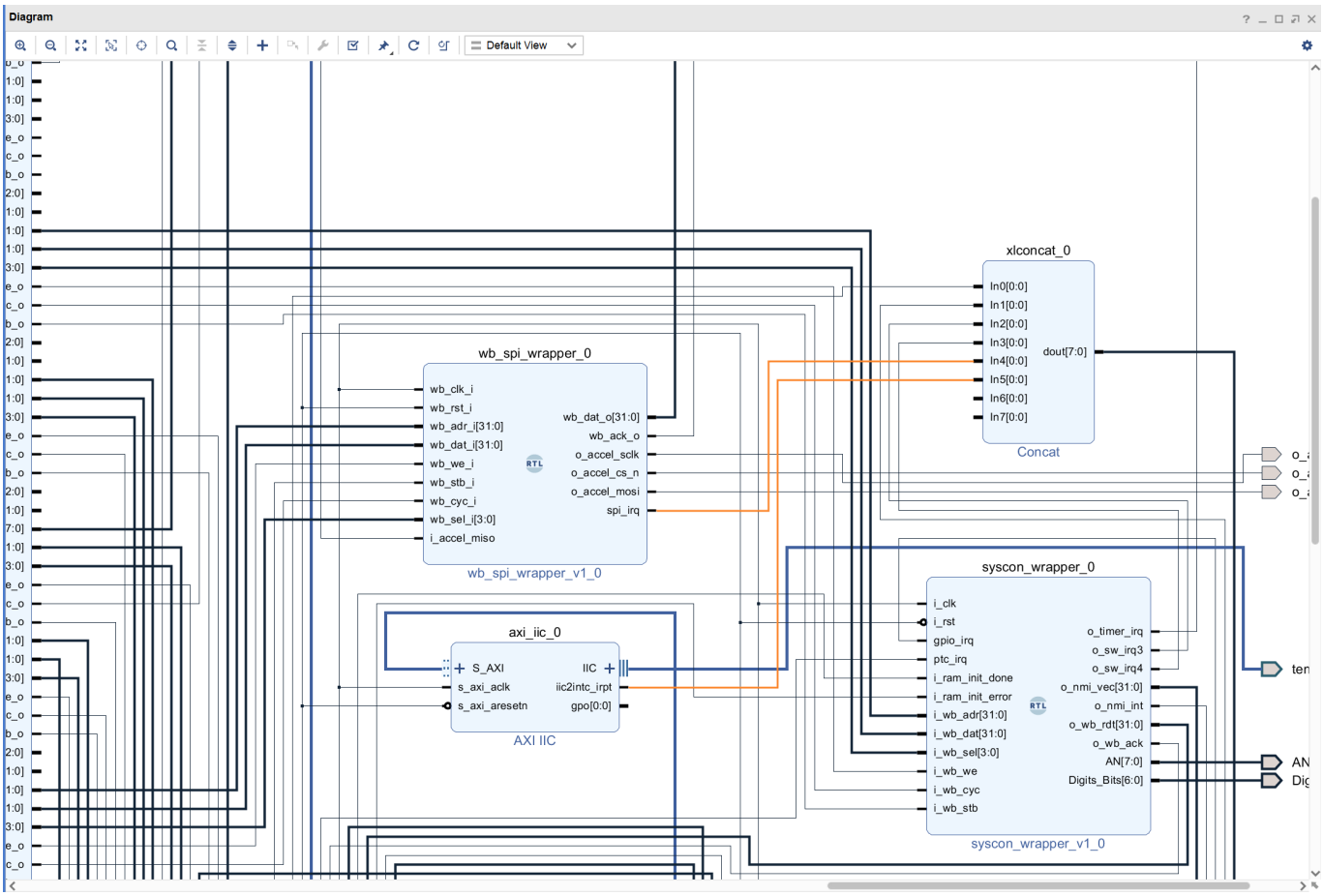


连接wb_spi_warpper_0模块和axi_iic_0模块的时钟和复位信号。

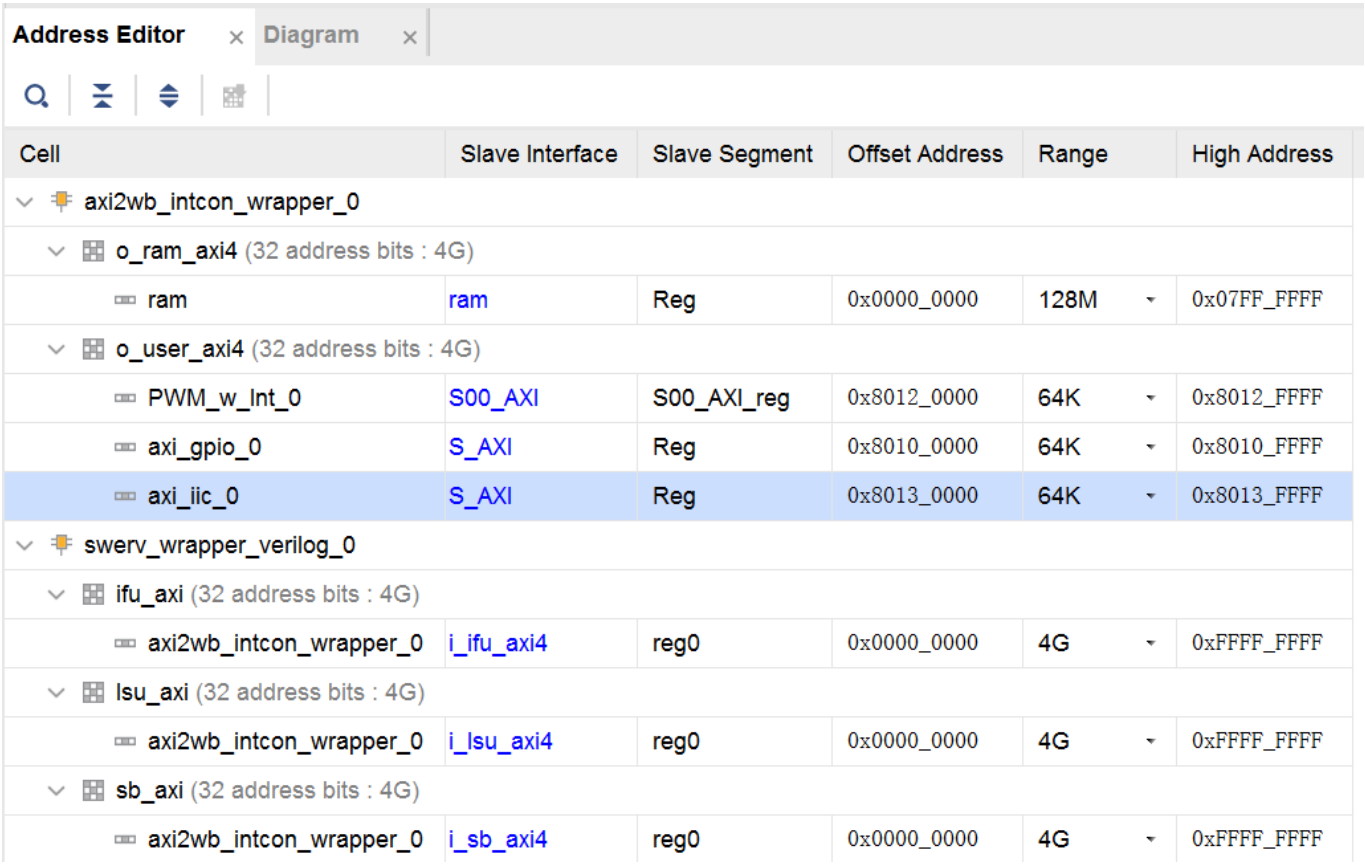
将wb_spi_warpper_0模块的“o_accel_sclk”、“o_accel_cs_n”、“o_accel_mosi”和“i_accel_miso”引脚设置为外部引脚。再将axi_iic_0模块的IIC端口设置为外部引脚，并更名为“temp_sensor”，如下图所示。



将wb_spi_warpper_0模块的“spi_irq”和axi_iic_0模块的“iic2intc_irpt”引脚分别连接到xlconcat_0模块的“In4”和“In5”引脚，如下图所示。



打开“Address Editor”，将axi_iic_0模块的地址设置为0x80130000，如下图所示。



Address Editor | Diagram

Cell	Slave Interface	Slave Segment	Offset Address	Range	High Address
axi2wb_intcon_wrapper_0					
o_ram_axi4 (32 address bits : 4G)					
ram	ram	Reg	0x0000_0000	128M	0x07FF_FFFF
o_user_axi4 (32 address bits : 4G)					
PWM_w_Int_0	S00_AXI	S00_AXI_reg	0x8012_0000	64K	0x8012_FFFF
axi_gpio_0	S_AXI	Reg	0x8010_0000	64K	0x8010_FFFF
axi_iic_0	S_AXI	Reg	0x8013_0000	64K	0x8013_FFFF
swerv_wrapper_verilog_0					
ifu_axi (32 address bits : 4G)					
axi2wb_intcon_wrapper_0	i_ifu_axi4	reg0	0x0000_0000	4G	0xFFFF_FFFF
lsu_axi (32 address bits : 4G)					
axi2wb_intcon_wrapper_0	i_lsu_axi4	reg0	0x0000_0000	4G	0xFFFF_FFFF
sb_axi (32 address bits : 4G)					
axi2wb_intcon_wrapper_0	i_sb_axi4	reg0	0x0000_0000	4G	0xFFFF_FFFF

点击Validate Design，对设计的正确性进行校验。校验过程中如果出现警告，点击OK忽略。

点击Generate Block Design，弹出对话框后选择Generate更新swerv_soc_wrapper文件。

根据更新后的swerv_soc_wrapper对rvfpga.sv文件进行修改，如下图所示，添加2个输入输出端口，同时增加swerv_soc_wrapper模块的端口引用。

```
26 module rvfpga(
27     input  wire             clk,
28     input  wire             rstn,
29     output wire [12:0]      ddram_a,
30     output wire [2:0]       ddram_ba,
31     output wire             ddram_ras_n,
32     output wire             ddram_cas_n,
33     output wire             ddram_we_n,
34     output wire             ddram_cs_n,
35     output wire [1:0]       ddram_dm,
36     inout  wire [15:0]      ddram_dq,
37     inout  wire [1:0]       ddram_dqs_p,
38     inout  wire [1:0]       ddram_dqs_n,
39     output wire             ddram_clk_p,
40     output wire             ddram_clk_n,
41     output wire             ddram_cke,
42     output wire             ddram_odt,
43     output wire             o_flash_cs_n,
44     output wire             o_flash_mosi,
45     input  wire             i_flash_miso,
46     input  wire             i_uart_rx,
47     output wire             o_uart_tx,
48     inout  wire [15:0]      i_sw,
49     output reg  [15:0]      o_led,
50     output wire [1:0]       PWMs,
51     output reg  [7:0]       AN,
52     output reg              CA, CB, CC, CD, CE, CF, CG,
53     inout                   temp_sensor_scl_io,
54     inout                   temp_sensor_sda_io,
55     output wire             o_accel_cs_n,
56     output wire             o_accel_mosi,
57     input  wire             i_accel_miso,
58     output wire             accel_sclk
59 );
```

```
260         .i_ram_init_done_0  (litedram_init_done),
261         .i_ram_init_error_0 (litedram_init_error),
262         .bidir              ({gpio_out, gpio_out2}),
263         .i_sw               (i_sw),
264         .o_led              (o_led),
265         .PWMs               (PWMs),
266         .AN_0               (AN),
267         .Digits_Bits_0      ({CA,CB,CC,CD,CE,CF,CG}),
268         .temp_sensor_scl_io (temp_sensor_scl_io),
269         .temp_sensor_sda_io (temp_sensor_sda_io),
270         .o_accel_sclk_0     (accel_sclk),
271         .o_accel_cs_n_0     (o_accel_cs_n),
272         .o_accel_mosi_0     (o_accel_mosi),
273         .i_accel_miso_0     (i_accel_miso)
274     );
275
276     //always @(posedge clk_core) begin
277     //  o_led[15:0] <= gpio_out2[15:0];
278     //end
279
280     //assign o_uart_tx = 1'b0 ? litedram_tx : cpu_tx;
281
282 endmodule
```

再根据修改后的rvfpga.sv文件对rvfpga.xdc约束文件进行修改，如下图所示，增加对温度传感器引脚的约束。

```
76 set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { AN[4] }];
   #IO_L8N_T1_D12_14 Sch=an[4]
77 set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { AN[5] }];
   #IO_L14P_T2_SRCC_14 Sch=an[5]
78 set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { AN[6] }];
   #IO_L23P_T3_35 Sch=an[6]
79 set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { AN[7] }];
   #IO_L23N_T3_A02_D18_14 Sch=an[7]
80
81 ##Accelerometer
82 set_property -dict { PACKAGE_PIN E15   IOSTANDARD LVCMOS33 } [get_ports { i_accel_miso }];
   #IO_L11P_T1_SRCC_15 Sch=acl_miso
83 set_property -dict { PACKAGE_PIN F14   IOSTANDARD LVCMOS33 } [get_ports { o_accel_mosi }];
   #IO_L5N_T0_AD9N_15 Sch=acl_mosi
84 set_property -dict { PACKAGE_PIN F15   IOSTANDARD LVCMOS33 } [get_ports { accel_sclk }];
   #IO_L14P_T2_SRCC_15 Sch=acl_sclk
85 set_property -dict { PACKAGE_PIN D15   IOSTANDARD LVCMOS33 } [get_ports { o_accel_cs_n }];
86
87 ##Temperature Sensor
88 set_property -dict { PACKAGE_PIN C14   IOSTANDARD LVCMOS33 } [get_ports {
   temp_sensor_scl_io }]; #IO_L1N_T0_AD0N_15 Sch=tmp_scl
89 set_property -dict { PACKAGE_PIN C15   IOSTANDARD LVCMOS33 } [get_ports {
   temp_sensor_sda_io }]; #IO_L12N_T1_MRCC_15 Sch=tmp_sda
90 #set_property -dict { PACKAGE_PIN D13   IOSTANDARD LVCMOS33 } [get_ports { TMP_INT }];
   #IO_L6N_T0_VREF_15 Sch=tmp_int
91 #set_property -dict { PACKAGE_PIN B14   IOSTANDARD LVCMOS33 } [get_ports { TMP_CT }];
   #IO_L2N_T0_AD8N_15 Sch=tmp_ct
```

最后，点击Generate Bitstream按键，生成bitstream文件。

2.2 ADXL362加速计应用

创建RVfpga工程，编写程序，读取ADT7420温度传感器测量的环境温度，并通过UART传送到主机显示。

3. 动手实验

创建RVfpga工程，编写程序，读取X轴、Y轴和Z轴加速度数据的高8位，然后在8位7段显示屏上显示这些值。