

公司名称: \_\_\_\_\_ ID: \_\_\_\_\_  
 \_\_\_\_\_ 公司名称: \_\_\_\_\_  
 \_\_\_\_\_ ID: \_\_\_\_\_

开始日期: 2021年8月18日, 星期三  
 截止日期: 2021年8月20日, 星期五


## 与Zybo的软件和硬件联合设计, 2021年夏季高峰

### 实验室#8: 为一个从MIO读取两个TMP101温度传感器, 为另一个Zybo选择IO

这是一个小组实验室。每组两名学生应该在Zybo板上演示你的I2CTMP101阅读实现, 并提交一份pdf报告副本。延迟的实验室提交将被接受, 每天延迟的评分将降低10%。

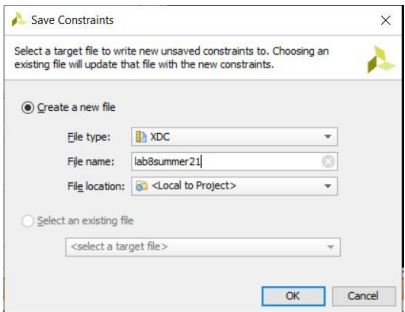
#### I. 所实现的目标

- (1) 构建一个Zynq7010电路, 在I2C控制器在MIO引脚14和15上实现I2C1: I2C0: I2C1的两个I2C接口, 在SelectIO引脚JB3和JB4上实现I2C1接口。您的块设计应该只有一个针对Zynq的I2C模块1的I2C接口连接。此接口需要连接到Zybo上的两个选择I/O引脚。以下示例将SCL引脚连接到JB3 (引脚V20), 并将SDA引脚连接到PmodJB连接器的JB4 (引脚W20)。如果您想自由使用, 您可以通过I/O端口菜单手动使用其他Pmod连接器插脚。将需要创建一个XDC文件来保存此分配。
- 您应该验证您的I2C模块0是否连接到MIO引脚14和15, 它们可以从Zybo的PmodJF访问。



The diagram shows a ZYNQ7 Processing System block with three output ports: DDR, FIXED\_IO, and IIC\_1. These are connected to a corresponding block on the right, also labeled with these three ports.


Name	Direction	Site	I/O Std	Bo
<b>All ports (132)</b>				
DDR_35186 (71)	INOUT		(Multiple)*	
FIXED_IO_35186 ...	INOUT		(Multiple)*	
IIC_1_35186 (2)	INOUT		LVC MOS33*	
<b>Scalar ports (2)</b>				
iic_1_sd_io	INOUT	V20	LVC MOS33*	
iic_1_sda_io	INOUT	W20	LVC MOS33*	
<b>Scalar ports (0)</b>				



The 'Save Constraints' dialog box is shown with the following settings:

- File type: XDC
- File name: lab8summer21
- File location: <Local to Project>

<input type="checkbox"/> UART 0	
<input checked="" type="checkbox"/> UART 1	MIO 48 .. 49
<input checked="" type="checkbox"/> I2C 0	MIO 14 .. 15
<input checked="" type="checkbox"/> I2C 1	EMIO
<input type="checkbox"/> SPI 0	



The 'Peripherals' block design shows the following configuration:

- UART 0: Not selected
- UART 1: Selected, connected to MIO 48 .. 49
- I2C 0: Selected, connected to MIO 14 .. 15
- I2C 1: Selected, connected to EMIO

- (1)在Zybo上连接两个TMP101断开板，以便它们可以被Zynq7010及其温度读取。您可以自由地设置他们的地址。
- (2)为ARMI2C模块开发一个C程序，以配置TMP101温度传感器，一个为10位分辨率，另一个为12位分辨率，并读取TMP101温度，并以正确的分辨率在SDK系列终端上以摄氏度和华氏度显示其温度。您将需要配置TMP101传感器，以传输12位的两个字节的温度值，以便温度将在小数点之后有四位数。
- (3)使用printf()进行浮点显示，因为xil\_printf()不支持浮点。浮点打印格式为8.4f，其中8为数字总数，4为小数点后的位数。请参见本讲义附录中来自Xilinx的I2C示例程序。此程序可用于测试您的硬件。
- (4)提交一份备忘录，说明您已经如何实施和验证了您的设计和实施。通过屏幕截图，显示哪个温度传感器具有10位分辨率，哪一个具有12位分辨率。在备忘录中包含相关的屏幕截图和源代码。



```

Problems Tasks Console Properties SDK Terminal Terminal 1
Serial: (COM11, 115200, 8, 1, None, None - CLOSED) - Encoding: (ISO-8859-1)
Read TMP101 Device on PL successfully
TMP101B on SelectIO pMod B= 30.2500 Celsius 86.4500 Fahrenheit

Read TMP101 Device on PS successfully
TMP101PS on MIO Pmod F = 29.1875 Celsius 84.5375 Fahrenheit

Read TMP101 Device on PL successfully
TMP101B on SelectIO pMod B= 30.1250 Celsius 86.2250 Fahrenheit

Read TMP101 Device on PS successfully
TMP101PS on MIO Pmod F = 29.2500 Celsius 84.6500 Fahrenheit

Read TMP101 Device on PL successfully
TMP101B on SelectIO pMod B= 30.1250 Celsius 86.2250 Fahrenheit

Read TMP101 Device on PS successfully
TMP101PS on MIO Pmod F = 29.0625 Celsius 84.3125 Fahrenheit

```

图1。在端口F和端口B上带有两个TMP101温度传感器的Zybo板。Zynq上的UART1必须选择在MIO48和49上。

端口F中的TMP101断开板通过MIO针脚连接到PS，并连接到端口F如下：JF9（M14）上的SDK和JF10（M15）上的SDA。端口B上的TMP101通过SelectIO引脚连接到PS，并连接到JB3（V20）为SCL，JB4（W20）为SDA。

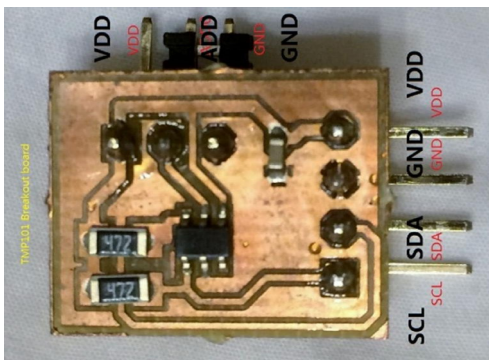


Figure 16. Pmod diagram.

Pmod JA (XADC)	Pmod JB (Hi-Speed)	Pmod JC (Hi-Speed)	Pmod JD (Hi-Speed)	Pmod JE (Std.)	Pmod JF (MIO)
JA1: N15	JB1: T20	JC1: V15	JD1: T14	JE1: V12	JF1: MIO-13
JA2: L14	JB2: U20	JC2: W15	JD2: T15	JE2: W16	JF2: MIO-10
JA3: K16	JB3: V20	JC3: T11	JD3: P14	JE3: J15	JF3: MIO-11
JA4: K14	JB4: W20	JC4: T10	JD4: R14	JE4: H15	JF4: MIO-12
JA7: N16	JB7: Y18	JC7: W14	JD7: U14	JE7: V13	JF7: MIO-0
JA8: L15	JB8: Y19	JC8: Y14	JD8: U15	JE8: U17	JF8: MIO-9
JA9: J16	JB9: W18	JC9: T12	JD9: V17	JE9: T17	JF9: MIO-14
JA10: J14	JB10: W19	JC10: U12	JD10: V18	JE10: Y17	JF10: MIO-15

分配针脚V20到I2C1时钟针脚和针脚W20到I2C1数据针脚。

Name	Direction	Site	Fixed	Bank	I/O Std	Vcco	Pull Type	Slew Type	Vref	Drive Strength	Off-Chip T
DDR_54127 (71)	INOUT		✓		(Multiple)*	1.500	NONE	(Multiple)*	(Multiple)		FP_VTT_50
FIXED_IO_54127 (59)	INOUT		✓		(Multiple)*	(Multiple)	(Multiple)	(Multiple)*	(Multiple)	(Multiple)*	(Multiple)
IIC_1_54127 (2)	INOUT		✓	34	LVCMOS33*	3.300	PULLUP	SLOW		8*	NONE
IIC_1_scl_io	INOUT	V20	✓	34	LVCMOS33*	3.300	PULLUP	SLOW		8*	NONE
IIC_1_sda_io	INOUT	W20	✓	34	LVCMOS33*	3.300	PULLUP	SLOW		8*	NONE

## II. 软件开发

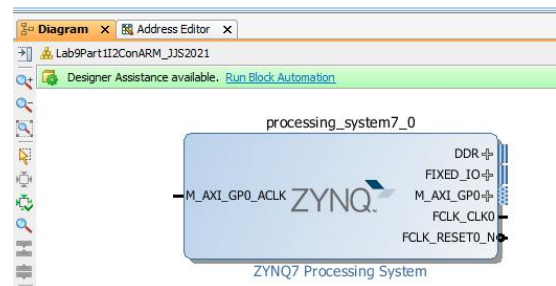
使用附录中的I2C示例程序作为起点，可以读取两个具有不同地址的I2CTMP101传感器。一个传感器被配置为12位分辨率，另一个为10位分辨率。在SDK的系列终端上同时显示摄氏度和华氏度的温度。

### I. 良好的块设计规范

项目文件名不应超过256个字符，也不应包含空格、破折号等特殊字符。

#### I.1 立即避免运行数据块自动化

在向块设计中添加zynq块后，请始终单击“运行块自动化”，以确保UART1已正确配置。不要在外设I/O销配置菜单中手动更改UART1配置。

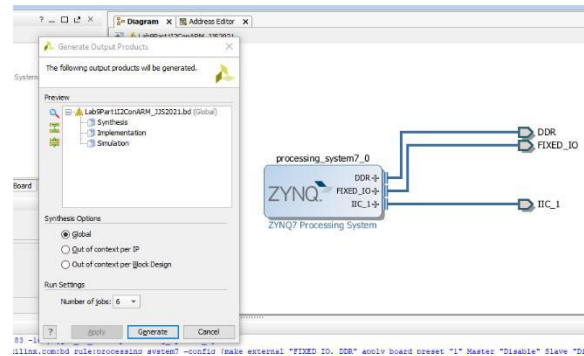


## I.2 块设计变更后运行生成输出产品

在每次块设计更改后和创建HDL包装器之前，请始终运行“生成输出产品”。有时，最好在运行生成输出产品之前先删除HDL包装器。

这是为了确保所有的软件驱动程序都与硬件IP组件正确地关联起来。

## II. 可能出现的错误



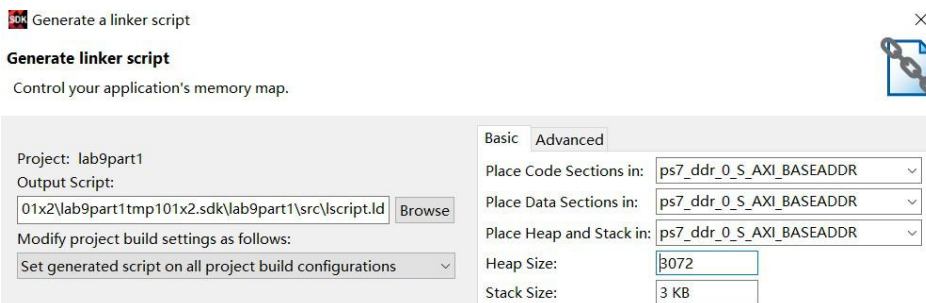


## II.1 printf() 和xil\_printf() 不应该一起使用

printf() 和xil\_printf() 相互独立。如果它们一起使用，即使首先调用printf()，xil\_printf() 也可能比printf() 显示得更快。因此，显示的文本可能似乎不正常。

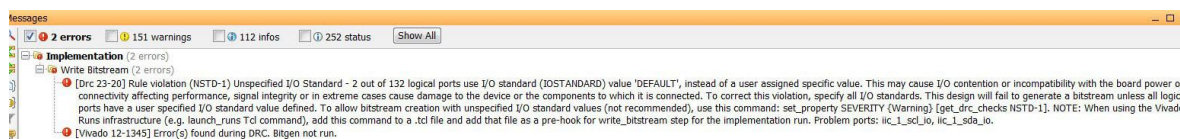
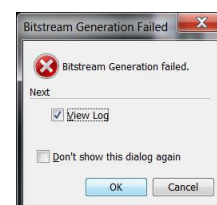
## II.2 打印文件() 错误

Printf() 使用堆栈进行操作。如果堆栈大小太小，printf() 可能会工作得很奇怪。请确保堆栈大小至少为3KB。



## II.3 输入输出标准上出现错误。

如果您看到以下错误，请将pinV20和W20的I/O标准更改为LVCMOS33。



默认的I/O标准是LVCMOS18 (1.8V)，但所需的I2C标准应该是3.3V。

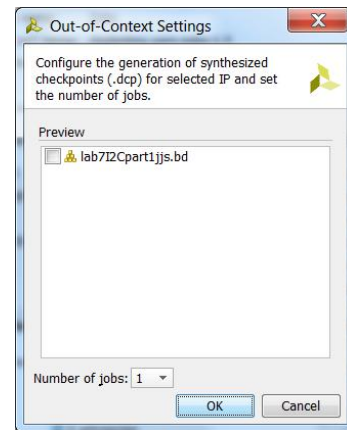
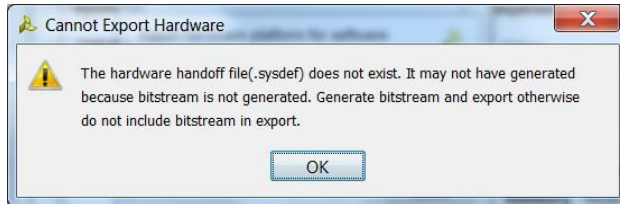
Name	Direction	Site	Fixed	Bank	I/O Std	Vcco	...	Pull Type	Slew Type	Vref	Drive Strength	Off-Chip
All ports (132)												
DDR_54127 (71)	INOUT		✓		(Multiple)*	1.500	NONE	(Multiple)*	(Multiple)*	(Multiple)	(Multiple)*	FP_VTT_5
FXED_IO_54127 (59)	INOUT		✓		(Multiple)*	(Multiple)	(Multiple)	(Multiple)*	(Multiple)	(Multiple)*	(Multiple)*	(Multiple)
IIC_1_54127 (2)	INOUT		✓	34	default (LVCMOS18)	1.800	NONE	SLOW		12		FP_VTT_5
Scalar ports (2)												
iic_1_scl_io	INOUT	V20	✓	34	default (LVCMOS18)	1.800	NONE	SLOW		12		FP_VTT_5
iic_1_sda_io	INOUT	W20	✓	34	default (LVCMOS18)	1.800	NONE	SLOW		12		FP_VTT_5
Scalar ports (0)												

以下是正确的输入/输出标准：LVCMOS33

Name	Direction	Site	Fixed	Bank	I/O Std	Vcco	...	Pull Type	Slew Type	Vref	Drive Strength
All ports (132)											
DDR_54127 (71)	INOUT		✓		(Multiple)*	1.500	NONE	(Multiple)*	(Multiple)*	(Multiple)	(Multiple)*
FXED_IO_54127 (59)	INOUT		✓		(Multiple)*	(Multiple)	(Multiple)	(Multiple)*	(Multiple)*	(Multiple)	(Multiple)*
IIC_1_54127 (2)	INOUT		✓	34	LVCMOS33*	3.300	NONE	SLOW		12	
Scalar ports (2)											
iic_1_scl_io	INOUT	V20	✓	34	LVCMOS33*	3.300	NONE	SLOW		12	
iic_1_sda_io	INOUT	W20	✓	34	LVCMOS33*	3.300	NONE	SLOW		12	
Scalar ports (0)											

## II.4 超出上下文设置错误：硬件切换(.sysdef)不存在

检查断义设置时会发生此错误。禁用“断章取义设置”将修复此错误。



### III. 附录xiicps\_polled\_master\_example.c

下面的例子是，xiicps\_polled\_master\_example.c来自C:  
\\Xilinx\\SDK\\2016.2\\data\\embeddedsd\\XilinxProcessorIPLib\\drivers\\iicps\_v3\_2\\exampl  
es.

```
/*版权所有(C) 2010 -2014Xilinx, Inc. 保留所有权利。*/
/**
 * @文件xiicps_polled_master_example.c
 * 将使用轮询模式下的设备作为主设备的一个设计示例。
 * 该示例使用的缓冲区大小为132。请设置的发送缓冲区
 * 金刚石设备是连续的64字节从0x00到0x3F。
 * <预>修改历史记录:
 * 具体日期      更改
 * -----
 * 1. 00ajz01月30日第一次发布
 */
#include "x参数.h" #包
#include "xiicps.h" #包括
#include "xil_printf.h"
/*
 * 下面的常量映射到在
 * x参数.h文件。它们在这里被定义为这样一个用户
 * 可以很容易地在一个地方更改所有所需的参数。
 */
#define IIC_DEVICE_ID                xpar_xiicps_0_device_id

/*要发送到和接收到的从属地址。*/ #定义了
IIC_SLAVE_ADDR                      0x55
#define IIC_SCLK_RATE                100000

/*以下常数值控制要发送的缓冲区的长度
 * 并收到了IIC。*/ #定义了
TEST_BUFFER_SIZE132

/*****函数原型*****/

例如(u16DeviceId);
/*****变量定义*****/

XilPsIic;          /**IICDevice*/的<实例

/*在本示例中，将使用以下缓冲区来发送和
 * 使用IIC接收数据。 */
u8发送缓冲区[TEST_BUFFER_SIZE]; /**传输数据的<缓冲区*/u8接收缓
冲区[TEST_BUFFER_SIZE]; /**<接收数据*/的缓冲区

/*****
 * 调用轮询的主例的主函数。
 * @param      没有。
 * @return      成功为XST_SUCCESS，不成功为XST_FAILURE。
 * @注意事项      没有。
 *****/
/主机（空）
{
```

```
xil_printf(“IIC主服务器轮询示例测试\r\n” );
```



```

/*在主模式下运行Iic轮询的示例，指定设备
* 在xparameters.h. 中指定的ID */
    状态=IicPs主投票示例(IIC_DEVICE_ID); 如果
    (Status!=XST_SUCCESS) {
        xil_printf(“IIC主服务器轮询示例测试失败
        \r\n”); 返回XST_FAILURE;
    }

xil_printf(“已成功运行IIC主轮询示例测试\r\n”); 返回
XST_SUCCESS;
}

/*****
* 此函数发送数据，并期望接收来自
* 从属机作为模块化的64。
* 此功能使用设备的中断驱动模式。
* @param      DeviceId是iicp设备的设备ID，是
*              来自x参数的XPAR_<IICPS_instance>_DEVICE_ID值
* @return      如果成功，否则XST_FAILURE。
* @注意事项      没有。
*****/
/的主投票示例(u16设备)
{
    int状态;
    XIicPs_Config*配
    置; int索引;

/*初始化IIC驱动程序，以便其可以使用
* 在配置表中查找配置，然后初始化它
    */Config=XIicPs_LookupConfig(DeviceId);
    如果(空的==配置){
        返回XST_FAILURE;
    }
    Status=XIicPs_CfgInitialize(&Iic、配置、配置->基本地址); 如果(Status!=XST_SUCCESS){
        返回XST_FAILURE;
    }

/*执行一个自检，以确保硬件
* 已构建正确。 */
    Status=XIicPs_SelfTest(&Iic、
        c); 如果(Status!=XST_SUCCESS){
        返回XST_FAILURE;
    }

/*设置IIC串行时钟速率。
    */XIicPs_SetSclk(&Iic,
    IIC_SCLK_RATE);

/*使用要发送的模式初始化发送缓冲区字节
* 接收缓冲区字节为零，以允许接收数据是
* 已验证。 */
    用于(索引=0; 索引<TEST_BUFFER_SIZE; 索引++){发送缓冲区
    区[索引]=(索引%TEST_BUFFER_SIZE); Recv缓冲区

```

```
        [索引]=0;
    }
/*使用IIC发送缓冲区，并忽略已发送的字节数
* 作为返回值，因为我们在中断模式下使用它。
*/Status=XIicPs_MasterSendPolled(&Iic, 发送缓
    冲区,
```

```
        test_buffer_size, iic_slave_addr);
    如果 (Status!
        =XST_SUCCESS) {返回
        XST_FAILURE;
    }
    /*等待总线空闲后启动另一个传输。*/同时
    (XIicPs_BusIsBusy(&Iic)) {
        /*没有*/
    }
    状态=XIicPs_MasterRecvPolled(&Iic, 恢复缓冲区,
        test_buffer_size, iic_slave_addr);
    如果 (Status!
        =XST_SUCCESS) {返回
        XST_FAILURE;
    }
    /*请验证接收到的数据是否正确。*/
    对于 (索引=0; 索引<TEST_BUFFER_SIZE; 索引++) {

    /*银块作为从属的输出只能设置64个字节, 如果
        (还原缓冲区[索引] !=索引%64) {
            返回XST_FAILURE;
        }
    }
```