

MYIR-ZYNQ7000系列-zturn教程(11)：i2c对24c32进行读写

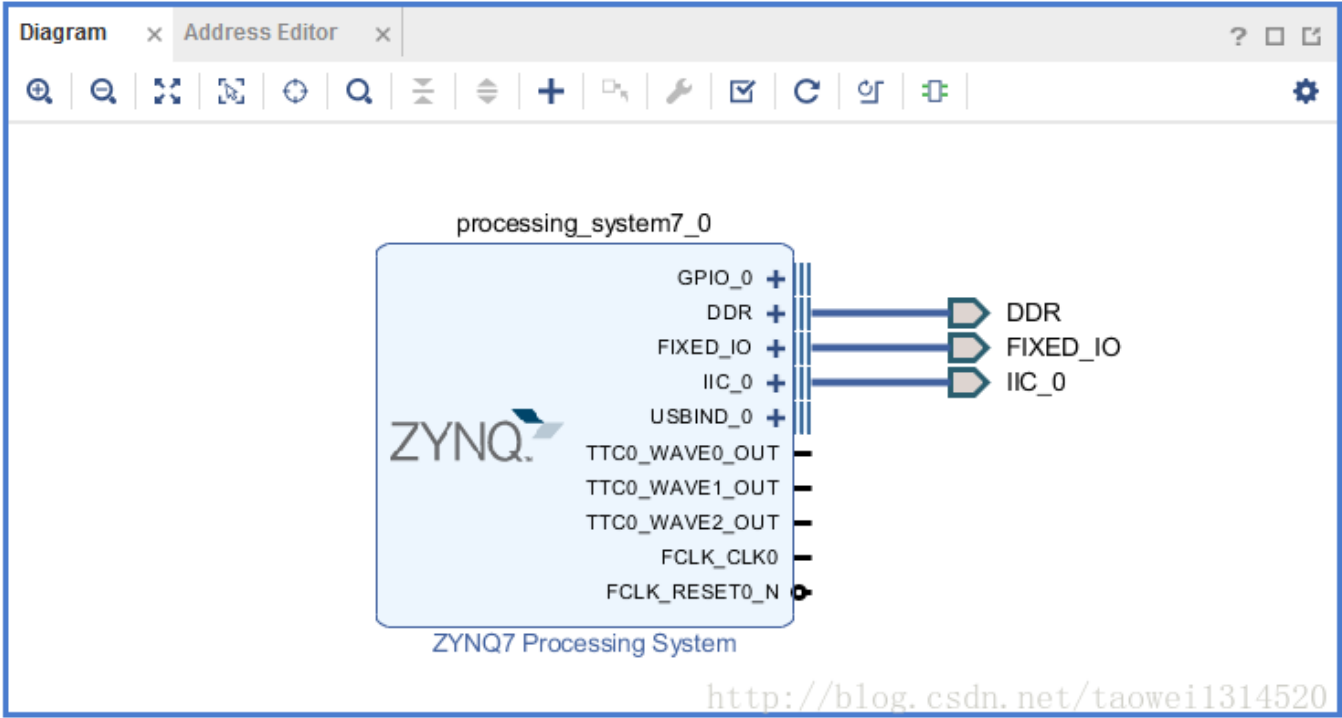
原创 虚无缥缈vs威武 最后发布于2018-02-23 20:04:55 阅读数 2380 ☆ 收藏

开发板环境：vivado 2017.1，开发板型号xc7z020clg400-1，这个工程主要是用i2c对24c32进行读写

链接：https://pan.baidu.com/s/1EjVY9kjbUKg3oegKkx2BA

提取码：dgcm

Step1 新建工程然后按照下面截图中进行配置（主要配置了DDR、i2c）



配置完成后进行综合、生成顶层文件，生成的顶层文件如下图所示

```
1 //Copyright 1986-2017 Xilinx, Inc. All Rights Reserved.
2 //-----
3 //Tool Version: Vivado v.2017.1 (win64) Build 1846317 Fri Apr 14 18:55:03 MDT 2017
4 //Date      : Sun Feb 18 22:06:05 2018
5 //Host      : MS-20180107KQQK running 64-bit Service Pack 1 (build 7601)
6 //Command   : generate_target design_1_wrapper.bd
7 //Design    : design_1_wrapper
8 //Purpose   : IP block netlist
9 //-----
10 `timescale 1 ps / 1 ps
11
12 module design_1_wrapper
13     (DDR_addr,
14      DDR_ba,
15      DDR_cas_n,
16      DDR_ck_n,
17      DDR_ck_p,
18      DDR_cke,
19      DDR_cs_n,
20      DDR_dm,
21      DDR_dq,
22      DDR_dqs_n,
23      DDR_dqs_p,
24      DDR_odt,
25      DDR_ras_n,
26      DDR_reset_n,
27      DDR_we_n,
28      FIXED_IO_dds_vrn,
29      FIXED_IO_dds_vrp,
30      FIXED_IO_mio,
31      FIXED_IO_ps_clk,
32      FIXED_IO_ps_porb,
33      FIXED_IO_ps_srstb,
34      iic_0_scl_io,
35      iic_0_sda_io);
36     inout [14:0]DDR_addr;
```

👍
2

🔗
展开

💬
8

☆


📱


<


>


赏


```
37     inout [2:0]DDR_ba;
38     inout DDR_cas_n;
39     inout DDR_ck_n;
40     inout DDR_ck_p;
41     inout DDR_cke;
42     inout DDR_cs_n;
43     inout [3:0]DDR_dm;
44     inout [31:0]DDR_dq;
45     inout [3:0]DDR_dqs_n;
46     inout [3:0]DDR_dqs_p;
47     inout DDR_odt;
48     inout DDR_ras_n;
49     inout DDR_reset_n;
50     inout DDR_we_n;
51     inout FIXED_IO_dds_vrn;
52     inout FIXED_IO_dds_vrp;
53     inout [53:0]FIXED_IO_mio;
54     inout FIXED_IO_ps_clk;
55     inout FIXED_IO_ps_porb;
56     inout FIXED_IO_ps_srstb;
57     inout iic_0_scl_io;
58     inout iic_0_sda_io;
59
60     wire [14:0]DDR_addr;
61     wire [2:0]DDR_ba;
62     wire DDR_cas_n;
63     wire DDR_ck_n;
64     wire DDR_ck_p;
65     wire DDR_cke;
66     wire DDR_cs_n;
67     wire [3:0]DDR_dm;
68     wire [31:0]DDR_dq;
69     wire [3:0]DDR_dqs_n;
70     wire [3:0]DDR_dqs_p;
71     wire DDR_odt;
72     wire DDR_ras_n;
73     wire DDR_reset_n;
74     wire DDR_we_n;
75     wire FIXED_IO_dds_vrn;
76     wire FIXED_IO_dds_vrp;
77     wire [53:0]FIXED_IO_mio;
78     wire FIXED_IO_ps_clk;
79     wire FIXED_IO_ps_porb;
80     wire FIXED_IO_ps_srstb;
81     wire iic_0_scl_i;
82     wire iic_0_scl_io;
83     wire iic_0_scl_o;
84     wire iic_0_scl_t;
85     wire iic_0_sda_i;
86     wire iic_0_sda_io;
87     wire iic_0_sda_o;
88     wire iic_0_sda_t;
89
90     design_1 design_1_i
91         (.DDR_addr(DDR_addr),
92          .DDR_ba(DDR_ba),
93          .DDR_cas_n(DDR_cas_n),
94          .DDR_ck_n(DDR_ck_n),
95          .DDR_ck_p(DDR_ck_p),
96          .DDR_cke(DDR_cke),
97          .DDR_cs_n(DDR_cs_n),
98          .DDR_dm(DDR_dm),
99          .DDR_dq(DDR_dq),
100         .DDR_dqs_n(DDR_dqs_n),
101         .DDR_dqs_p(DDR_dqs_p),
102         .DDR_odt(DDR_odt),
103         .DDR_ras_n(DDR_ras_n),
104         .DDR_reset_n(DDR_reset_n),
105         .DDR_we_n(DDR_we_n),
106         .FIXED_IO_dds_vrn(FIXED_IO_dds_vrn),
107         .FIXED_IO_dds_vrp(FIXED_IO_dds_vrp),
108         .FIXED_IO_mio(FIXED_IO_mio),
109         .FIXED_IO_ps_clk(FIXED_IO_ps_clk),
110         .FIXED_IO_ps_porb(FIXED_IO_ps_porb),
111         .FIXED_IO_ps_srstb(FIXED_IO_ps_srstb),
112         .IIC_0_scl_i(iic_0_scl_i),
113         .IIC_0_scl_o(iic_0_scl_o),
114         .IIC_0_scl_t(iic_0_scl_t),
115         .IIC_0_sda_i(iic_0_sda_i),
```


2





8













举报

```
116         .IIC_0_sda_o(iic_0_sda_o),
117         .IIC_0_sda_t(iic_0_sda_t));
118     IOBUF iic_0_scl_iobuf
119         (.I(iic_0_scl_o),
120         .IO(iic_0_scl_io),
121         .O(iic_0_scl_i),
122         .T(iic_0_scl_t));
123     IOBUF iic_0_sda_iobuf
124         (.I(iic_0_sda_o),
125         .IO(iic_0_sda_io),
126         .O(iic_0_sda_i),
127         .T(iic_0_sda_t));
128 endmodule
```

这个i2c工程的管脚是用emio引出的，所以要加这两个IOBUF,如果是mio引出的可以不加这两个IOBUF

```
IOBUF iic_0_scl_iobuf
```

```
IOBUF iic_0_sda_iobuf
```

Step2 新建一个xdc文件

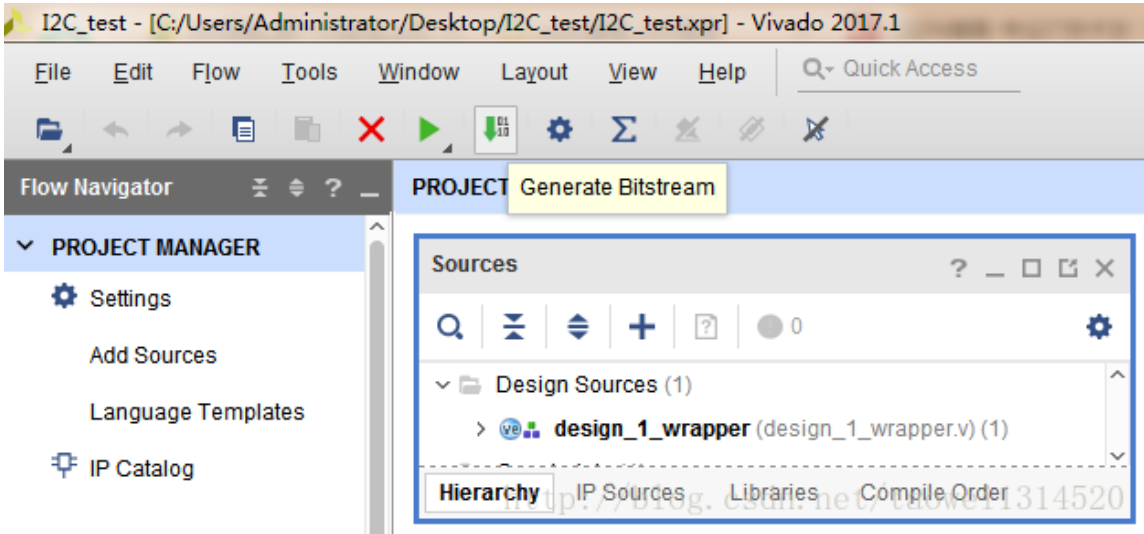
```
1 set_property PACKAGE_PIN B19 [get_ports iic_0_sda_io]
2 set_property PACKAGE_PIN A20 [get_ports iic_0_scl_io]
3 set_property IOSTANDARD LVCMOS33 [get_ports iic_0_scl_io]
4 set_property IOSTANDARD LVCMOS33 [get_ports iic_0_sda_io]
5
6 set_property PULLUP true [get_ports iic_0_scl_io]
7 set_property PULLUP true [get_ports iic_0_sda_io]
```

新建的工程因为是emio引出的所以要分配管脚，如果mio就不用分配引脚。

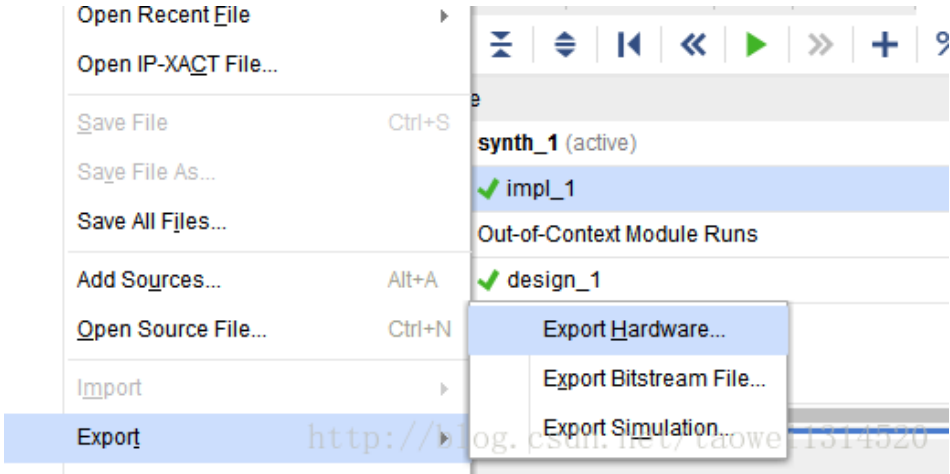
这个xdc文件主要注意下面这两行，这两行主要是加内部上拉电阻

```
1 set_property PULLUP true [get_ports iic_0_scl_io]
2 set_property PULLUP true [get_ports iic_0_sda_io]
```

Step3 生成bit文件



Step4点击菜单栏上的 File->Export->Export Hardware 导出硬件配置文件



👍
2

🔗

💬
8

☆

📱

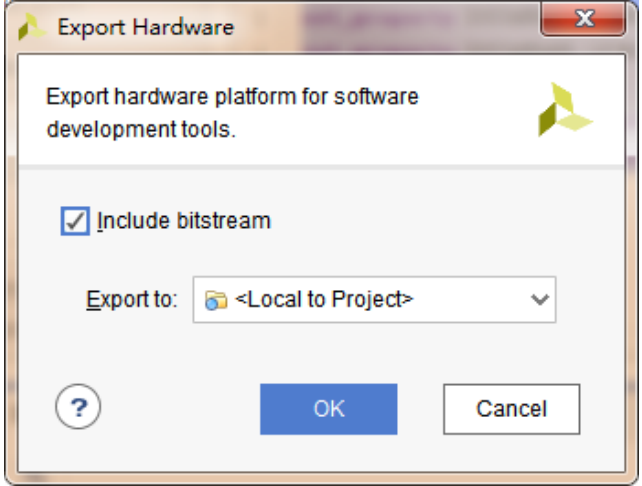
<

>

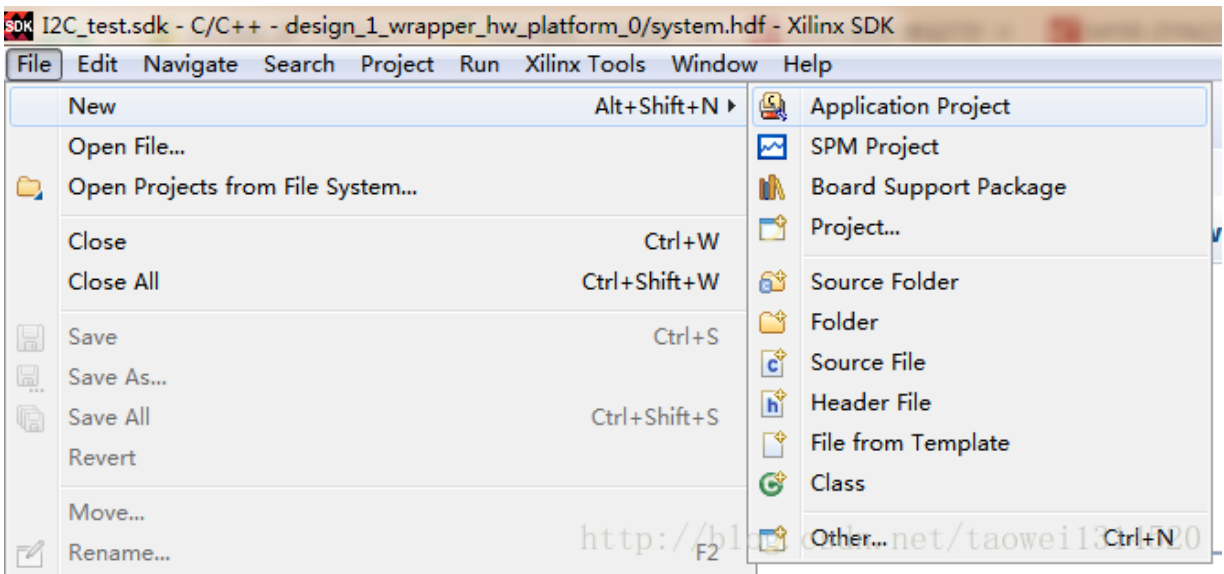
赏

🔊

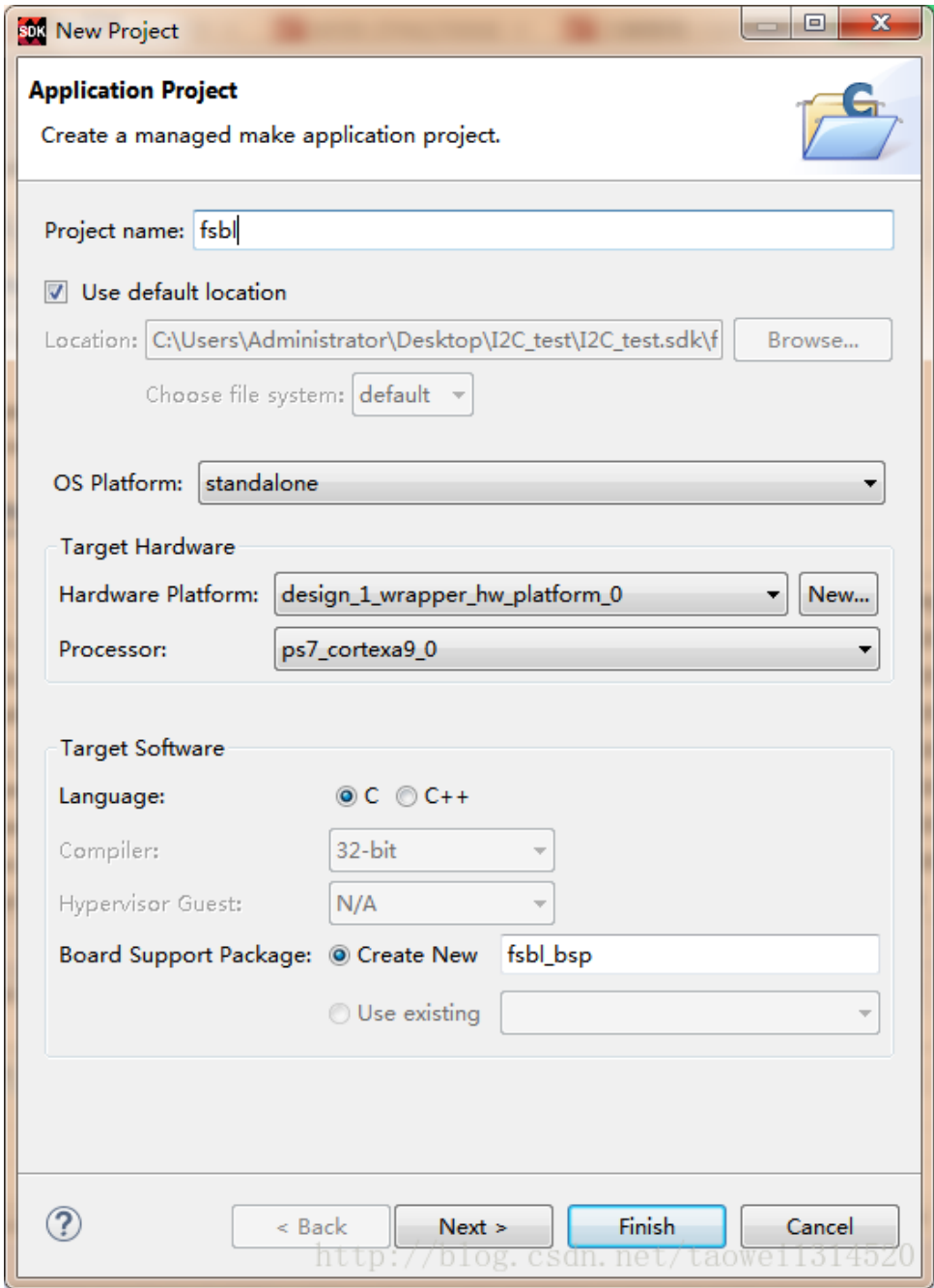
举报




Step5 打开SDK，然后新建一个fsbl





点击Next





点击Finish


2





8





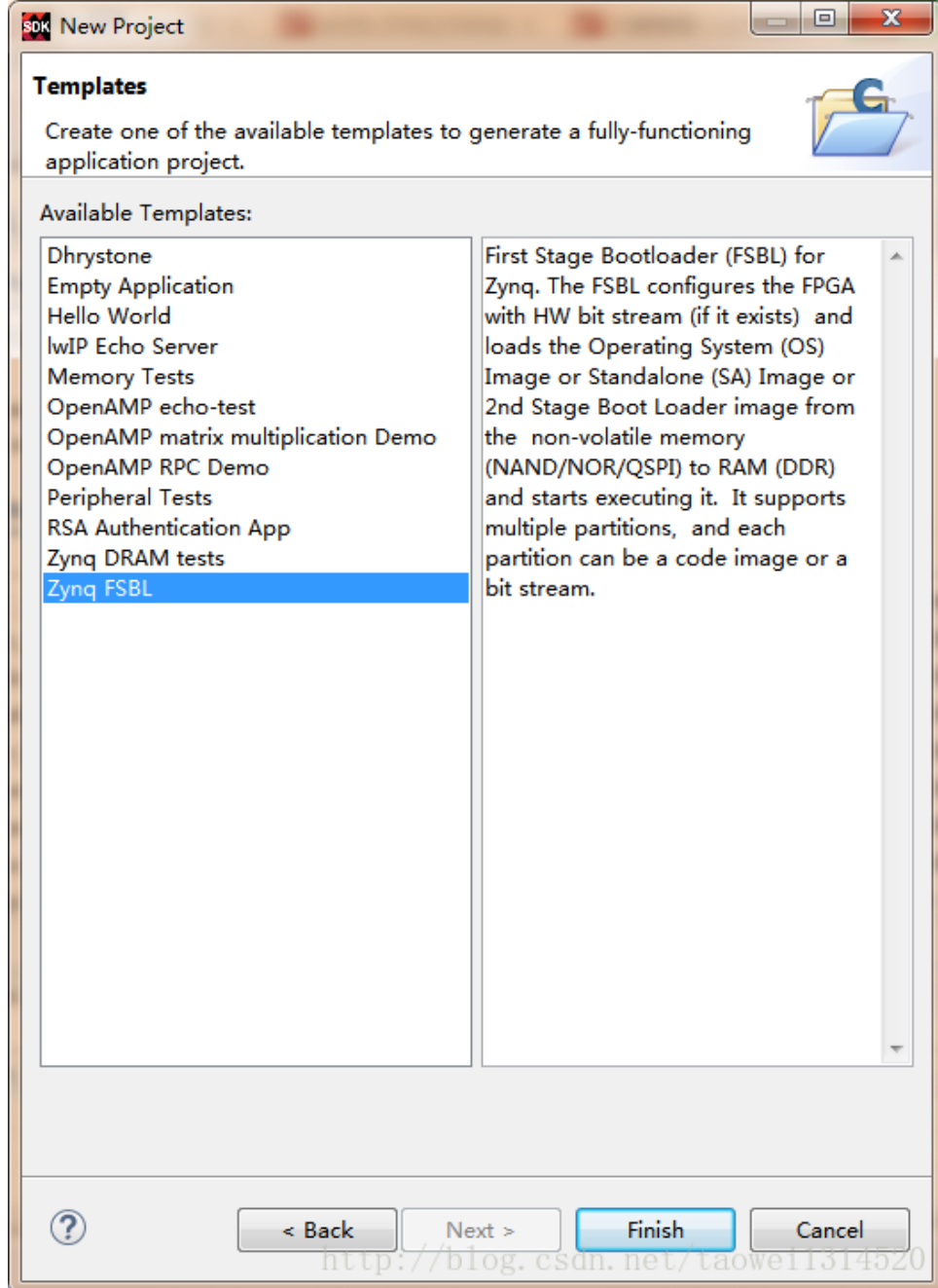




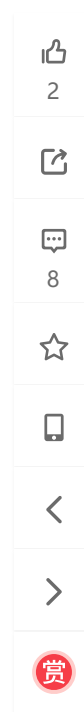
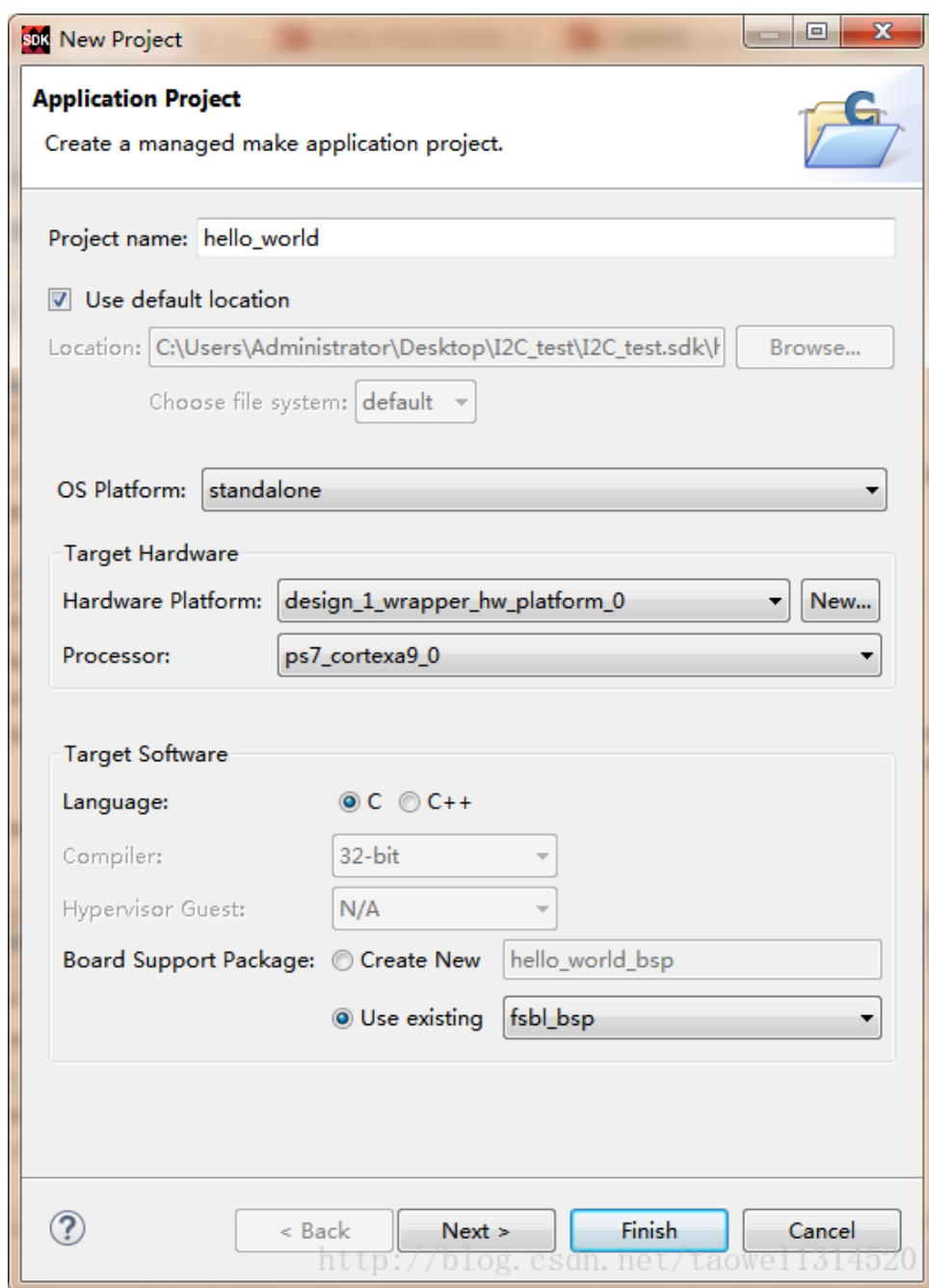


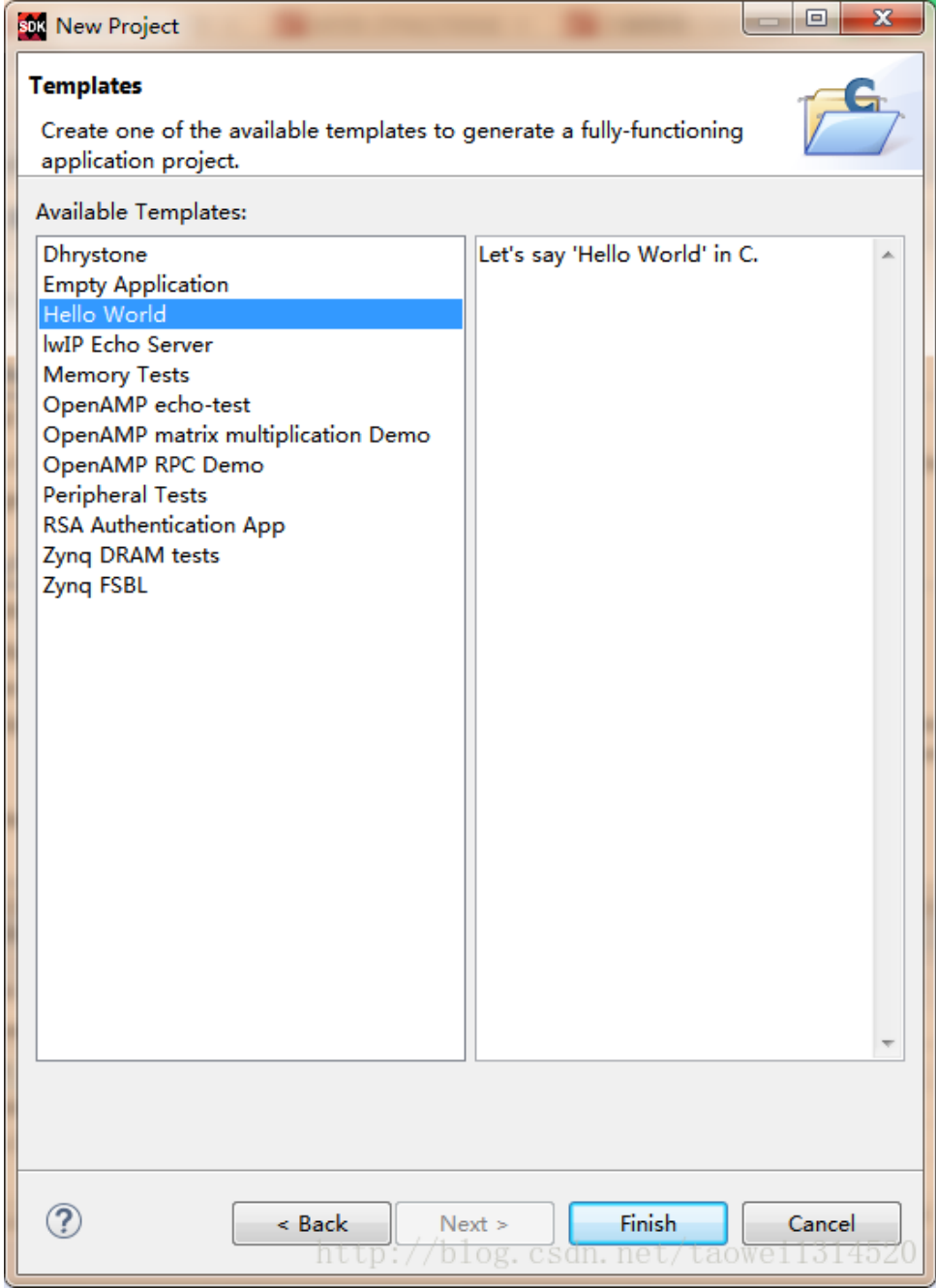


举报



Step 6 新建一个hello_world模板工程

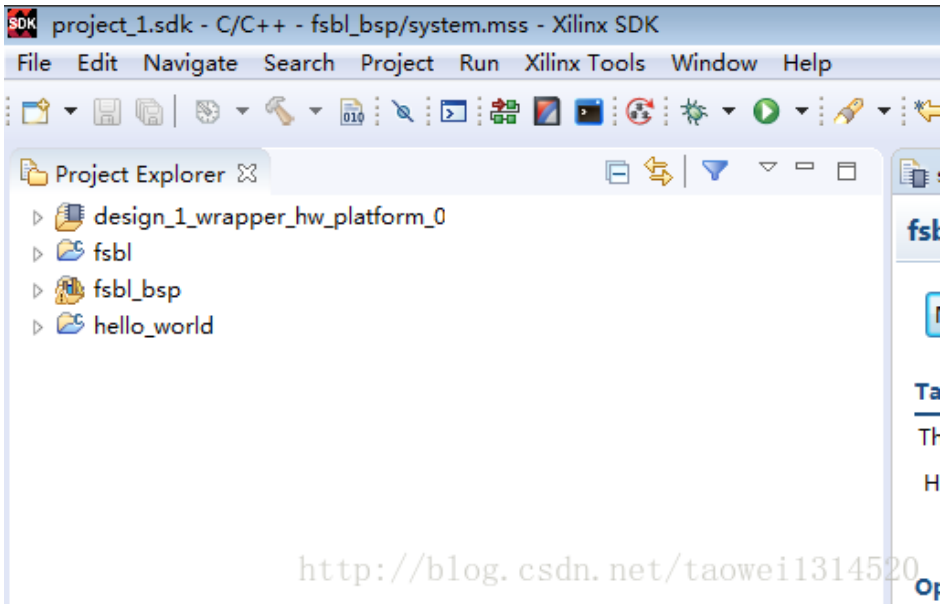




2

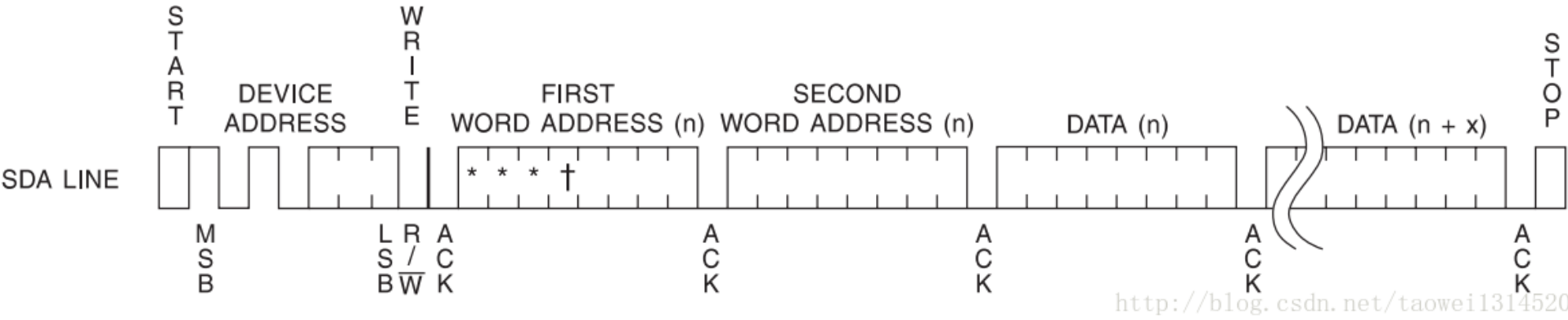
8

新建完成后如下图所示



下面是hello_world的主程序这里是写入16位起始地址0x00进行连续写和读

Figure 3. Page Write



```
1  /*****
2  *
3  * Copyright (C) 2009 - 2014 Xilinx, Inc. All rights reserved.
4  *
5  * Permission is hereby granted, free of charge, to any person obtaining a copy
6  * of this software and associated documentation files (the "Software"), to deal
7  * in the Software without restriction, including without limitation the rights
8  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
9  * copies of the Software, and to permit persons to whom the Software is
```

举报

```
10 * furnished to do so, subject to the following conditions: 11 | *
12 * The above copyright notice and this permission notice shall be included in
13 * all copies or substantial portions of the Software.
14 *
15 * Use of the Software is limited solely to applications:
16 * (a) running on a Xilinx device, or
17 * (b) that interact with a Xilinx device through a bus or interconnect.
18 *
19 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
22 * XILINX BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
23 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF
24 * OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
25 * SOFTWARE.
26 *
27 * Except as contained in this notice, the name of the Xilinx shall not be used
28 * in advertising or otherwise to promote the sale, use or other dealings in
29 * this Software without prior written authorization from Xilinx.
30 *
31 *****/
32
33 /*
34  * helloworld.c: simple test application
35  *
36  * This application configures UART 16550 to baud rate 9600.
37  * PS7 UART (Zynq) is not initialized by this application, since
38  * bootrom/bsp configures it to baud rate 115200
39  *
40  * -----
41  * | UART TYPE   BAUD RATE                                |
42  * -----
43  *  uartns550    9600
44  *  uartlite     Configurable only in HW design
45  *  ps7_uart     115200 (configured by bootrom/bsp)
46  */
47
48 #include <stdio.h>
49 #include "platform.h"
50 #include "xil_printf.h"
51
52
53 #include "sleep.h"
54 #include "xiicps.h"
55
56 XIicPs IicInstance;    /* The instance of the IIC device. */
57
58 #define IIC_DEVICE_ID   XPAR_XIICPS_0_DEVICE_ID
59
60 u8 WriteBuffer[2 + 1];
61
62 u8 ReadBuffer[1];    /* Read buffer for reading a page. */
63
64 struct sensor_register {
65     u8 value;
66 };
67
68
69 static const struct sensor_register i2c_data[] = {
70
71     { 0x00},
72     { 0x00},
73     { 0x04},
74     { 0x01},
75     { 0x11},
76     { 0x02},
77     { 0x3a},
78     { 0x70},
79     { 0x17},
80     { 0x98},
81     { 0x08},
82     { 0x65},
83     { 0x04},
84     { 0x70},
85     { 0x01},
86
87     { 0xff}, /* over */
88 };
```



2



8



举报


```
89 | 90 |
91
92 int iic_master_init(void)
93 {
94     int Status;
95     XIicPs_Config *ConfigPtr;    /* Pointer to configuration data */
96
97     ConfigPtr = XIicPs_LookupConfig(IIC_DEVICE_ID);
98     if (ConfigPtr == NULL) {
99         return XST_FAILURE;
100     }
101
102     Status = XIicPs_CfgInitialize(&IicInstance, ConfigPtr,
103                                   ConfigPtr->BaseAddress);
104     if (Status != XST_SUCCESS) {
105         return XST_FAILURE;
106     }
107
108     XIicPs_SetSCLk(&IicInstance, 400000);
109
110     return XST_SUCCESS;
111 }
112
113
114
115 int iic_write_read_8(u8 Device_Address,u8 First_Word_Address,u8 Second_Word_address,u8 data)
116 {
117     int Status;
118
119     WriteBuffer[0] = First_Word_Address;
120     WriteBuffer[1] = Second_Word_address;
121     WriteBuffer[2] = data;
122
123     Status = XIicPs_MasterSendPolled(&IicInstance, WriteBuffer,
124                                       3, Device_Address>>1);
125     if (Status != XST_SUCCESS) {
126         return XST_FAILURE;
127     }
128
129     while (XIicPs_BusIsBusy(&IicInstance));
130
131     usleep(2500);
132     if (Status != XST_SUCCESS) {
133         return XST_FAILURE;
134     }
135
136     WriteBuffer[0] = First_Word_Address;
137     WriteBuffer[1] = Second_Word_address;
138
139     Status = XIicPs_MasterSendPolled(&IicInstance, WriteBuffer,
140                                       2, Device_Address>>1);
141     if (Status != XST_SUCCESS) {
142         return XST_FAILURE;
143     }
144
145     while (XIicPs_BusIsBusy(&IicInstance));
146
147     usleep(2500);
148
149     Status = XIicPs_MasterRecvPolled(&IicInstance, ReadBuffer,
150                                       1, Device_Address>>1);
151     if (Status != XST_SUCCESS) {
152         return XST_FAILURE;
153     }
154     while (XIicPs_BusIsBusy(&IicInstance));
155     xil_printf("0x%02x\r\n",ReadBuffer[0]);
156     return 0;
157 }
158
159
160 int main(void)
161 {
162     int i;
163     u8 Device_Address;
164     u8 First_Word_Address;
165     u8 Second_Word_address;
166
167
```



2





8





举报



```
168     Device_Address      = 0xAE;169     First_Word_Address   = 0x00;
170     Second_Word_address = 0x00;
171     i = 0;
172
173
174
175 iic_master_init();
176
177
178 while(1)
179 {
180     if(i2c_data[i].value==0xff)
181         break;
182     iic_write_read_8(Device_Address,First_Word_Address,Second_Word_address,i2c_data[i].value);
183     i++;
184 }
185
186
187     return 0;
188 }
```


2





8











这两个i2c的读写程序都比较简单，这里只是简单的介绍下要注意的地方

这个主要是对master控制器进行初始化

```
iic_master_init();
```

初始化中主要主要注意：

```
ConfigPtr = XIicPs_LookupConfig(IIC_DEVICE_ID);
```

其中这个i2c的设备ID主要是看你用zynq哪个设备，一般都是用的i2c0、i2c1

XPAR_XIICPS_0_DEVICE_ID //对应i2c0

XPAR_XIICPS_1_DEVICE_ID //对应i2c1

下面的这段代码主要设置i2c的工作频率，我这里用的是400k

```
XIicPs_SetSCLK(&IicInstance, 400000);
```

初始化完成后主要就是进行读写了，在读写之前要将进行的读写的数据进行缓冲，所以会用到Buffer

这个24c32的地址是16位，数据是8位，所以一共要用到3个Buffer进行缓冲

```
1 WriteBuffer[0] = First_Word_Address;
2 WriteBuffer[1] = Second_Word_address;
3 WriteBuffer[2] = data;
```

上面的是写的3个缓冲Buffer，这个是读的缓冲Buffer只有一个

```
ReadBuffer[0]
```

缓冲Buffer做完后就开始进行写，这里进行写的函数只要设置缓冲Buffer的个数以及所接的24C32的设备地址就可以了

```
1 Status = XIicPs_MasterSendPolled(&IicInstance, WriteBuffer,
```



举报

```
2 | 3, Device_Address>>1);

当数据写完成后开始进行读，读分为两步：第一步写入你所要读的地址、第二步读这个地址里的数据

1 | 1.   Status = XIicPs_MasterSendPolled(&IicInstance, WriteBuffer,
2 |     2, Device_Address>>1);

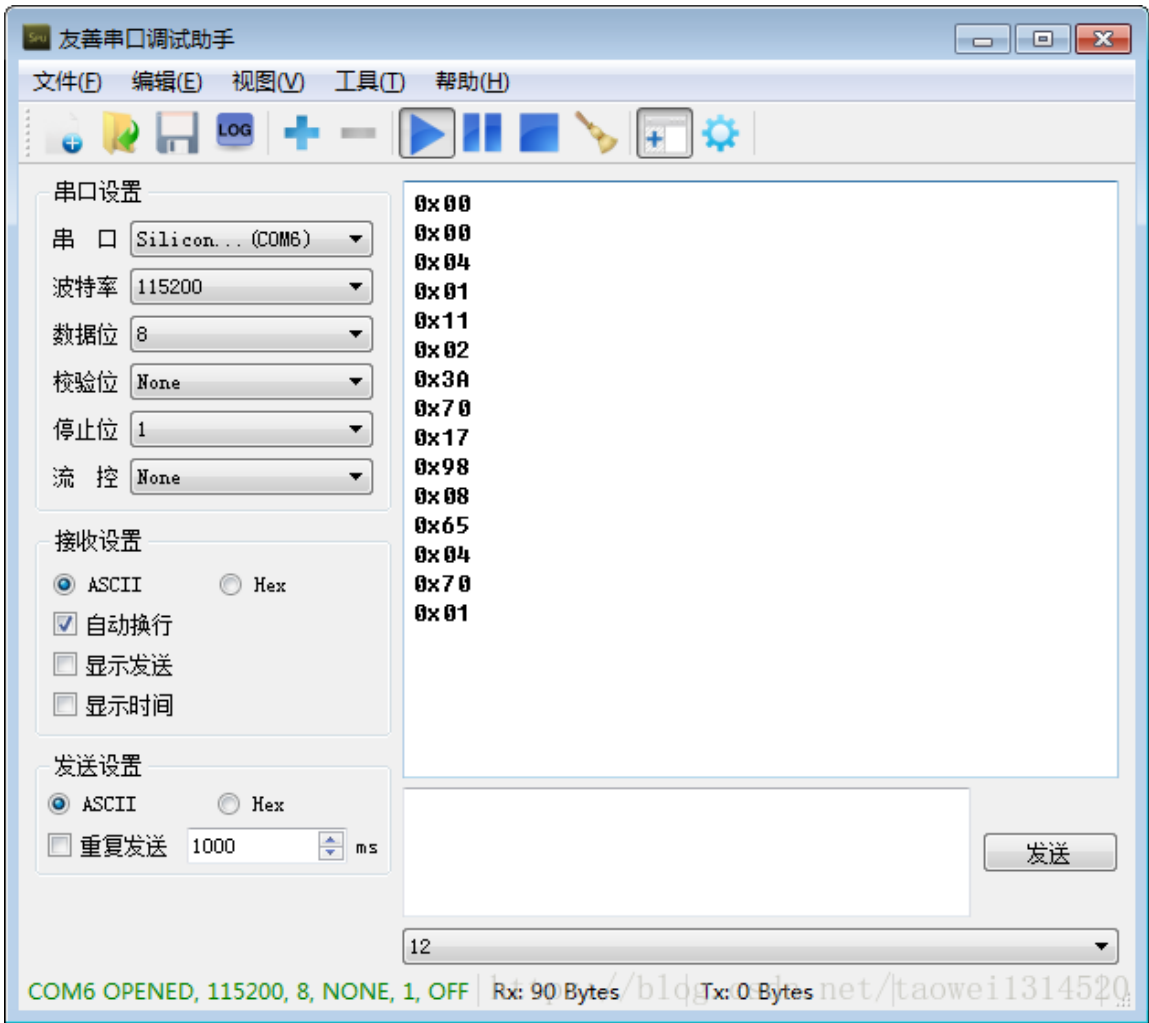
1 | 2.   Status = XIicPs_MasterRecvPolled(&IicInstance, ReadBuffer,
2 |     1, Device_Address>>1);
```

这里是连续读写,所以开始写入16位的0x00的起始地址，然后对这个起始地址不断累加进行连续读

对特定地址进行读写和连续读写差不多，这里不再进行介绍

最后的这个是打印i2c读Buffer里的数据

```
xil_printf("0x%02x\r\n",ReadBuffer[0]);
```



👍
2

🔗

💬
8

☆


📱

<

>

赏

👍 点赞 2 ☆ 收藏 🔗 分享 ...



虚无缥缈vs威武

发布了37 篇原创文章 · 获赞 58 · 访问量 15万+

私信

关注



CorelDRAW Graphics Suite 2020

為您帶來市面上最快速、流暢、精準的向量圖形設計軟體。

广告 coreldraw.com

 想对作者说点什么

举报

博主，这个工程有链接吗？不太知道里面tcl文件

<div><div></div><div><div>闪宝宝真可爱</div><div>1年前</div></div></div> <div>你好，ax7020板子上的eeprom也要加上拉电阻吗</div>	<div><div></div><div>2</div><div>回复(3)</div><div></div></div>
<div><div></div><div><div>Joyce_Chan</div><div>1年前</div></div></div> <div>博主，24c32你是外接的么？我在7020这个板子上好像没看到有这个东西啊。。</div>	<div><div></div><div>2</div><div>回复(1)</div><div></div></div>
<div><div><h3>【原创】zynq-7010下运用I2C总线完成对LSM303D传感器的数据读取</h3><div>这是本人第一次写博客，我的毕设在用FPGA去读取LSM303D传感器的中的三轴的磁场强度数据，这也是我第一次用... 博文 来自: weixin_40389136...</div><div><h4>zynq上IIC驱动</h4><div>i2c总线驱动编写: /*filename: I2C.caauthor: pingbo andescription: EEPROM I2C*/#include &lt;linux/module.h... 博文 来自: anpingbo的专栏</div><div><h4>Xilinx的IIC程序中的XlicPs_MasterSendPolled和XlicPs_MasterRecvPolled函数的使用，8位寄存器地址写...</h4><div>Xilinx FPGA的IIC程序中的XlicPs_MasterSendPolled和XlicPs_MasterRecvPolled函数的使用，8位寄存器地址写入... 博文 来自: weixin_43950612...</div></div></div></div><div><div><div>阅读数 5108</div><div></div></div><div><div></div><div></div><div></div><div></div></div></div></div>	

Zynq7020 MIO ps端的裸板编程

阅读数 192

之前有篇博客是介绍pl的gpio的使用，这里的MIO是直接连到逻辑的，并不是连到pl，所以在ps的编程如下：`#defin...` [博文](#) 来自：[smile_5me](#)的博客

i2c对24c32进行读写 阅读数 600
开发板环境: vivado 2017.1 , 开发板型号xc7z020clg400-1, 这个工程主要是用i2c对24c32进行读写Step1 新建工... [博文](#) 来自: [feifansong的博客](#)

zynq PS端I2C的使用

阅读数 2986

因为需要连接一款i2c接口的OLED，需要使用PS端的i2c接口。1、PL端勾选上i2c并通过EMIO分配引脚（PS可以通... [博文](#) 来自: [ma_cheng_yuan的...](#)

ZYNQ-I2C 调试 阅读数 2623

在zynq上成功移植I2C驱动，对eeprom进行读写访问。用示波器采样得到以下结论：MSB模式：1.先发送低地址，以... [博文](#) 来自： [微微一笑](#)

zcu102_17_PS通过I2C接口读写外设寄存器

阅读数 275

根据 ug1182, zcu102 板卡上 PS 连接 2 个 I2C 接口, 分别是 I2C0(MIO 14-15) 和 I2C1(MIO 16-17)。本试验使... [博文](#) 来自: [bt_的博客](#)

zynq学习笔记——EMIO方式模拟I2C时序对ADV7511进行读写

阅读数 3422

创建硬件工程，很简单，PS接出两个EMIO和一个74.25M时钟管脚约束# ADV7511 I2C_SCLset_property PACKAG... [博文](#) 来自: [luotong86的专栏](#)

BeagleBoneBlack学习 (3) ——U-Boot中的I2C驱动分析


阅读数 2660


BeagleBoneBlack等TI开发板上都有一块eeprom芯片, u-boot根据eeprom芯片不同的内容对开发板进项配置, 芯... [博文](#) 来自: [hkchenhao的博客](#)


Uboot关于i2c和EEPROM的命令


阅读数 3844

在uboot命令行下输入i2c并回车，会打印出i2c所有命令的使用方法：比如i2c dev会打印出目前挂载的i2c设备：i2c ... [博文](#) 来自： [hahachenchen78...](#)


Tyc_小胖
 1篇文章
[关注](#) 排名:千里之外


apple^?
 35篇文章
[关注](#) 排名:千里之外


请answer1996
 13篇文章
[关注](#) 排名:千里之外


smile_5me
 96篇文章
[关注](#) 排名:千里之外

Verilog实现IIC主机对从机的写操作(zybo z7板运行代码)

Zynq7000的MIO和EMIO之区别

阅读数 3500

Zynq7000系列芯片有5

博文 来自: [linyangspring的专栏](#)

Zynq, 使用ps与pl数据交互问题 05-12

需要在pl端设置多个串口, 但要求每个串口FIFO要大于128byte, 求问, 用什么方式和ps通信? 如何设定使用这个FIFO? ... [问答](#)

Zynq-Linux移植学习笔记之13-i2c驱动配置

阅读数 5173

1、背景介绍板子上通过I2C总线与zynq相连的是三片1848如上图所示，zynq通过I2C总线与3片CPS-1848交换芯片... [博文](#) 来自: [无知的我](#)

Zynq7020 16位i2c地址的读写调试方法

阅读数 555

以前我有篇博客写的是i2ctool的使用方法，博客链接：[i2ctool的使用方法](#)里面介绍的i2cset，i2cget等的指令，都是... 博文 来自：[smile_5me的博客](#)

MYIR-ZYNQ7000系列-zturn教程(12): 用i2c接口读取温度传感器STLM75 阅读量 2148

开发板环境: vivado 2017.1, 开发板型号xc7z020c1g400-1, 这个工程主要用I2C接口读取STLM75的温度, 同时也... [博文](#) 来自: [taowei1314520的...](#)



举报

Zynq-Linux移植学习笔记之43-linux下应用读写I2C设备 阅读数 160

在读写I2C设备时，由于I2C地址分为7bit和10bit，读写时序存在区别，因此应用程序也存在区别。首先来看读写时... 博文 来自： 无知的我

ZYNQ_IIC读写M24M01记录板子状态 阅读数 323

ZYNQ_IIC读写M24M01记录板子状态1 M24M01特点1.1 特征1，兼容IIC的模式：1MHz；400kHz；100kHz；2， ... 博文 来自： 想做一条贪吃蛇

ZYNQ使用PL部分IIC收发数测试 阅读数 96

1、背景介绍ZYNQ在PS部分有两路I2C，但有时候存在不够用的情况，这时就需要使用PL部分的I2C IP核（以下简称... 博文 来自： 无知的我

uboot i2c 驱动 阅读数 81

Preloader and U-Boot Customization - v13.1U-Boot: Adding a New Driver in U-Boothttps://rocketboards.... 博文 来自： 爱她就要努力

Xilinx FPGA Microblaze AXI_IIC使用方法及心得 阅读数 1324

Xilinx FPGA Microblaze AXI_IIC使用方法及心得前言最近公司要将主控程序从Cortex M系列的ARM上移植到Xilinx ... 博文 来自： 一路前行的博客

linux配置静态ip 阅读数 821

linux设置静态ip在使用虚拟机时，当重启虚拟机，发现每次ip都会改变，这样在连接vm时都需要修改ip，比较麻烦... 博文 来自： qingfengsongyue...

LINUX的IIC从这开始（一） 阅读数 3384

首先介绍一下所分析LINUX的版本：linux-3.0.8 博文 来自： xie0812的专栏

IIC (I2C)总线 FPGA Verilog HDL 阅读数 436

IIC (I2C)总线 FPGA Verilog HDL配置文件：根据具体的IIC设备改一下时钟频率就可以产生正确的时钟波形`define ... 博文 来自： weixin_42965338...

模拟IIC 24cxx系列 读写eeprom 连续读写 页写 stm32 24c128 24c256 02-10 下载

24c系列的快速读写eeprom，同等情况下，使用页写的方式，比传统的一个一个字节的写快64倍，适用于需要快速读写的... 博文 来自： 想做一条贪吃蛇

MYIR-ZYNQ7000系列-zturn教程(22)：用axi_iic对24C32进行读写 阅读数 1513

开发板环境：vivado 2017.4，开发板型号xc7z020clg400-1，这个工程主要用axi_iic对24C32进行读写链接：https... 博文 来自： taowei1314520的...

Zynq-Linux移植学习笔记之28-PS端I2C从模式实现 阅读数 1720

1、背景介绍最近在调试集群处理平台，模块上使用了支持IPMI的BMC控制芯片。该芯片与ZYNQ通过I2C总线相连... 博文 来自： 无知的我

zynq-7000系列基于7015的linux下IIC->RTC的扩展使用（DS3232） 阅读数 2097

zynq-7000系列基于7015的linux下IIC->RTC的扩展使用（DS3232） ... 博文 来自： luhao806的专栏

米联客 ZYNQ/SOC 精品教程 S02-CH24 利用AXI VDMA 实现MT9V034摄像头采集 阅读数 36

软件版本：VIVADO2017.4 操作系统：WIN10 64bit 硬件平台：适用米联客 ZYNQ系列开发板 米联客(MSXBO)论... 博文

ZYNQ--矿机通信控制板 阅读数 496

目录前言1、外设1.1电源1.2 程序下载口1.3 SOC内部1.4 FPGA点亮LED前言在网上淘了一块ZYNQ的板子，学习了... 博文 来自： 大咖之路，砥砺前行...

zynq i2c软核连接mcp79410RTC 阅读数 251

1、设备树更改IIC:i2c@0x41600000{ compatible = "xlnx,xps-iic-2.00.a"; status = "q... 博文 来自： ma_cheng_yuan的...

ZYNQ进阶之路5--PS端hello xilinx zynq设计 阅读数 754

在ZYNQ进阶之路1-4中我们大致了解了ZYNQ PL端的开发流程以及使用verilog硬件描述语言写了几个硬件模块，希... 博文 来自： WP_FD的博客

嵌入式开发之zynq---Zynq PS侧I2C驱动架构 阅读数 203

http://blog.chinaunix.net/uid-24148050-id-120532.htmlhttp://bbs.csdn.net/topics/390538368?page=1http... 博文 来自： weixin_34032621...

初学24CXX系列EEPROM使用详解&STM32库函数I2C总线 阅读数 3002

24CXX系列芯片属于EEPROM（Electrically Erasable Programmable read only memory）即电可擦可编程只... 博文 来自： zhb2004xp的博客

zedboard将ZYNQ的EMIO映射到PS端串口1使用收发 阅读数 429

做了两天ZYNQ的EMIO映射到PS端串口1使用的实验终于成功了，原因竟然是SDK收发程序不适用，先是做了逻辑证... 博文 来自： fighting2019的博客

MYIR-ZYNQ7000系列-zturn教程(13)：用SPI接口对eeprom M95512进行读写 阅读数 1931

开发板环境：vivado 2017.1，开发板型号xc7z020clg400-1，这个工程是用spi接口对eeprom进行读写Step1 新建... 博文 来自： taowei1314520的...

Zynq-Linux移植学习笔记之31-用户自定义I2C驱动 阅读数 1009

1、背景介绍板子上通过ZYNQ的I2C-0控制器连接了三片DBF芯片和一片Ti的226测功耗芯片，示意图如下：如上图... 博文 来自： 无知的我

Zynq7020 i2c ps端的裸板编程 阅读数 751

ps端的i2c我就以我们这里的一个sensor为例子，这是我写的一个裸板的例子，仅供参考。int licPsSelfTestExample(... 博文 来自： smile_5me的博客



2



8



举报

1.FPGA的JTAG信号通过隔离芯片引出，用ise能识别到FPGA的JTAG却无法识别ARM的JTAG;用vivado什么也识别不了; 2....

论坛

python json java mysql pycharm android linux json格式



虚无缥缈vs威武

TA的个人主页 >

原创

粉丝

获赞

评论

访问

37

195

58

164

15万+

等级:

博客 4

周排名: 3万+

积分:

1254

总排名: 6万+

勋章:





关注

私信

最新文章

- quartus II 12.1 使用教程（7） vga显示测试
- MYIR-ZYNQ7000系列-zturn教程(27): lwip测试
- quartus II 12.1 使用教程（6） ROM 测试
- quartus II 12.1 使用教程（5） eeprom 读写测试
- quartus II 12.1 使用教程（4） uart 测试

分类专栏

- 

VIVADO 安装教程

1篇
- 

quartus II

5篇
- 

三态门详解
- 

quartus II 12.1 使用...

1篇
- 

ZYNQ7000

27篇

归档

- 2019年12月1篇
- 2019年9月1篇
- 2019年8月5篇
- 2019年7月2篇
- 2019年4月1篇
- 2019年3月2篇
- 2019年1月1篇
- 2018年11月1篇



2



8



举报

展开

热门文章

VIVADO 安装教程

阅读数 84216

三态门详解

阅读数 15398

quartus II 12.1 使用教程（1） 怎样调用PLL 核

阅读数 7556

MYIR-ZYNQ7000系列-zturn教程(17)：用axi_uart发送数据

阅读数 4156

MYIR-ZYNQ7000系列-zturn教程(9)：将bit文件固化到QSPI_Flash

阅读数 4055

最新评论

VIVADO 安装教程

rq8866： 缺License的小伙伴 链接：https://pan.baidu.com/s/11mjcpyERdUH3q5C_TpfQxQ ...

FT232H如何使用jtag接口

taowei1314520： [reply]qq_42662835[/reply]我是直接对eeprom里写数据进去的，数据我已经 ...

FT232H如何使用jtag接口

taowei1314520： [reply]sssshhhhhhhh[/reply]这个vivado有这个usb驱动也需要安装一下，你 ...

FT232H如何使用jtag接口

sssshhhhhhhh： 你好，插上电脑以后显示 USB Serial Conventor （仅配置了USB和EEPROM这 ...

MYIR-ZYNQ7000系列-z...

kuyunge： SPI一次是通信一个字节码？



👤 QQ客服 ✉ kefu@csdn.net

🗣 客服论坛 ☎ 400-660-0108

工作时间 8:30-22:00

[关于我们](#) [招聘](#) [广告服务](#) [网站地图](#)

京ICP备19004658号 经营性网站备案信息

👮 公安备案号 11010502030143

©1999-2020 北京创新乐知网络技术有限公司
网络110报警服务

北京互联网违法和不良信息举报中心

中国互联网举报中心 家长监护 版权申诉



2



8



举报