

# *Lab7: Memory Test*

## ——SRAM 存储单元测试实验

基于 Nexys 4 FPGA 平台

## Lab 7: Memory Test

### 实验简介

本实验旨在指导读者使用 Xilinx 的 XPS 和 SDK 工具，实现一个 memory test 的实验。本实验中最重要的一部分就是参考 Nexys 4 的数据手册，在 XPS 中，对系统 ip 核的关键数据进行配置。

### 实验目标

在完成本实验后，您将学会：

- 提供的系统 IP 核中的参数与数据手册的 IP 核的相关参数的对应关系
- 在 SDK 中运行内存测试文件

### 实验过程

本实验旨在指导读者使用 Xilinx 的 XPS 工具，调用串口的 IP 核、以及 EMCip 核，修改相关的参数，并将导入到 SDK，调用这个 IP 核，在串口上显示 SRAM 测试的结果，然后在 Nexys 4 平台上进行测试验证。

实验由以下步骤组成：

1. 在 XPS 中建立工程
2. 添加 IP 核并调整相关设置
3. 进行端口的互连
4. 将工程导入到 SDK
5. 在 SDK 中添加 c 语言源程序
6. 在 Nexys 4 上进行测试验证

### 实验环境

#### ◆ 硬件环境

1. PC 机
2. Nexys 4 FPGA 平台

#### ◆ 软件环境

Xilinx ISE Design Suite 14.3 (FPGA 开发工具)

## 第一步 创建工程

- 1-1. 运行 **Xilinx Platform Studio**, 创建一个空的新工程, 基于 **xc6slx45csg484-3** 芯片和 **VHDL** 语言.
- 1-1-1. 选择 开始菜单 > 所有程序 > **Xilinx Design Tools > ISE Design Suite 14.3 > EDK > Xilinx Platform Studio**. 点击运行 **Xilinx Platform Studio(XPS)** (Xilinx Platform Studio 是 ISE 嵌入式版本 Design Suite 的关键组件, 可帮助硬件设计人员方便地构建、连接和配置嵌入式处理器系统, 能充分满足从简单状态机到成熟的 32 位 RISC 微处理器系统的需求。).
- 1-1-2. 点击 **Create New Project Using Base System Builder** 来打开新工程建立向导。会出现一个 **Create New XPS Project Using BSB Wizard** 对话框, 如图 3.



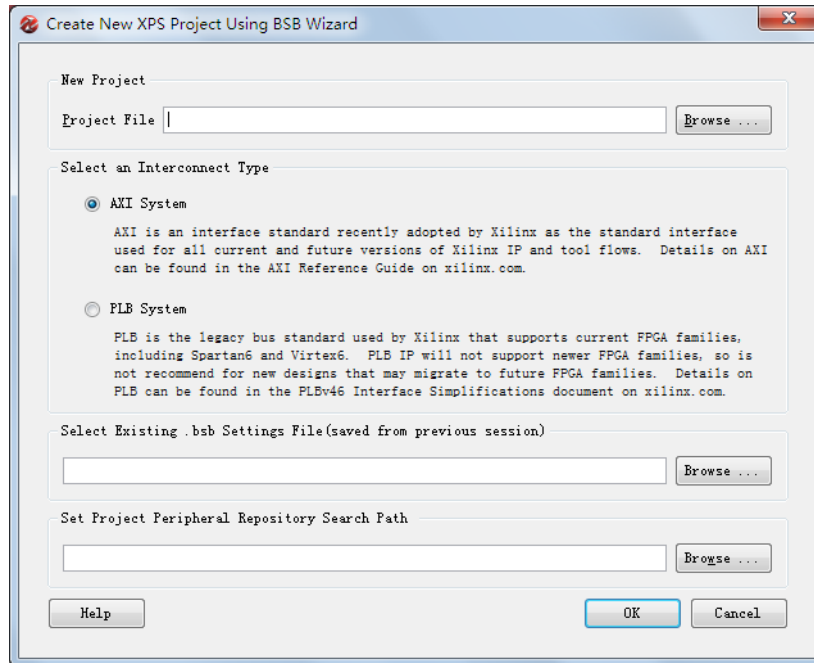


图 3：新工程建立向导

- 1-1-3. 如图 3，在新工程建立向导对话框的 **Project File** 栏选择工程建立后存放的路径，这里可以选择 **c:\Nexys4\_lab\**，可以将 **system.xmp** 改成所建立工程的名字，这里取 **lab7**(名字中不要有中文和空格)，于是 **Project File** 栏中的路径变为 **c:\Nexys4\_lab\lab7\lab7**。点击 **OK**。
- 1-1-4. 新出现的是关于工程的一些参数设置的对话框，设置如下的参数后，点击 **Next**，如图 4。
- architecture: artix 7**
  - Device: XC7a1007**
  - Package: CSG324**
  - Speed: -3**

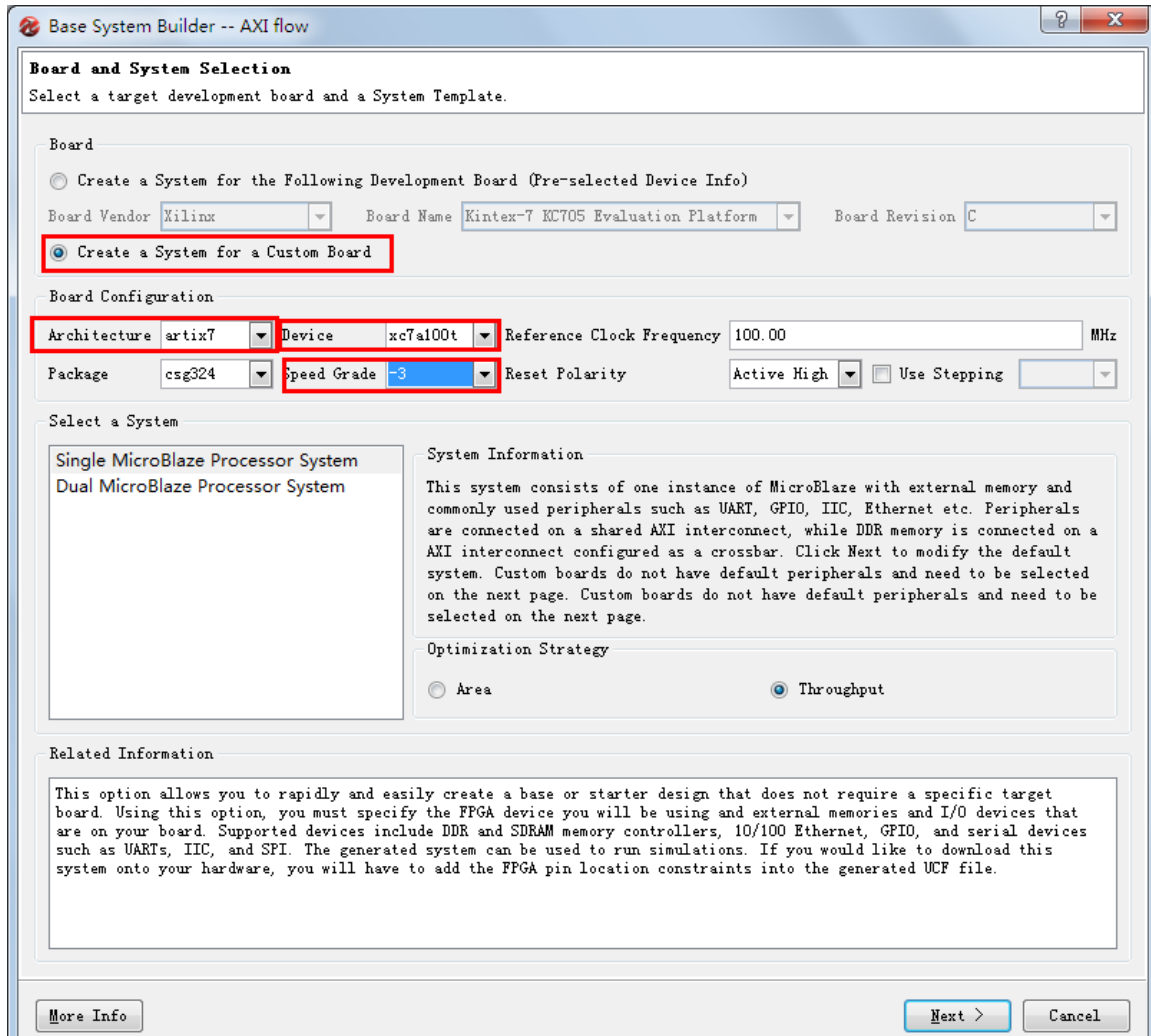


图 4：新工程参数设置

1-1-5. 在接下来出现的页面中选择要添加的串口 IP 核，并设置 IP 核的参数：

单击 **Select and configure Peripherals** 下的 **Add Device...**

出现图 5 中的蓝色对话框。

在 **IO Interface Type** 中的下拉菜单中选择 **UART**。

在 **Device** 的下拉菜单中选择 **RS232**。

单击 **OK**

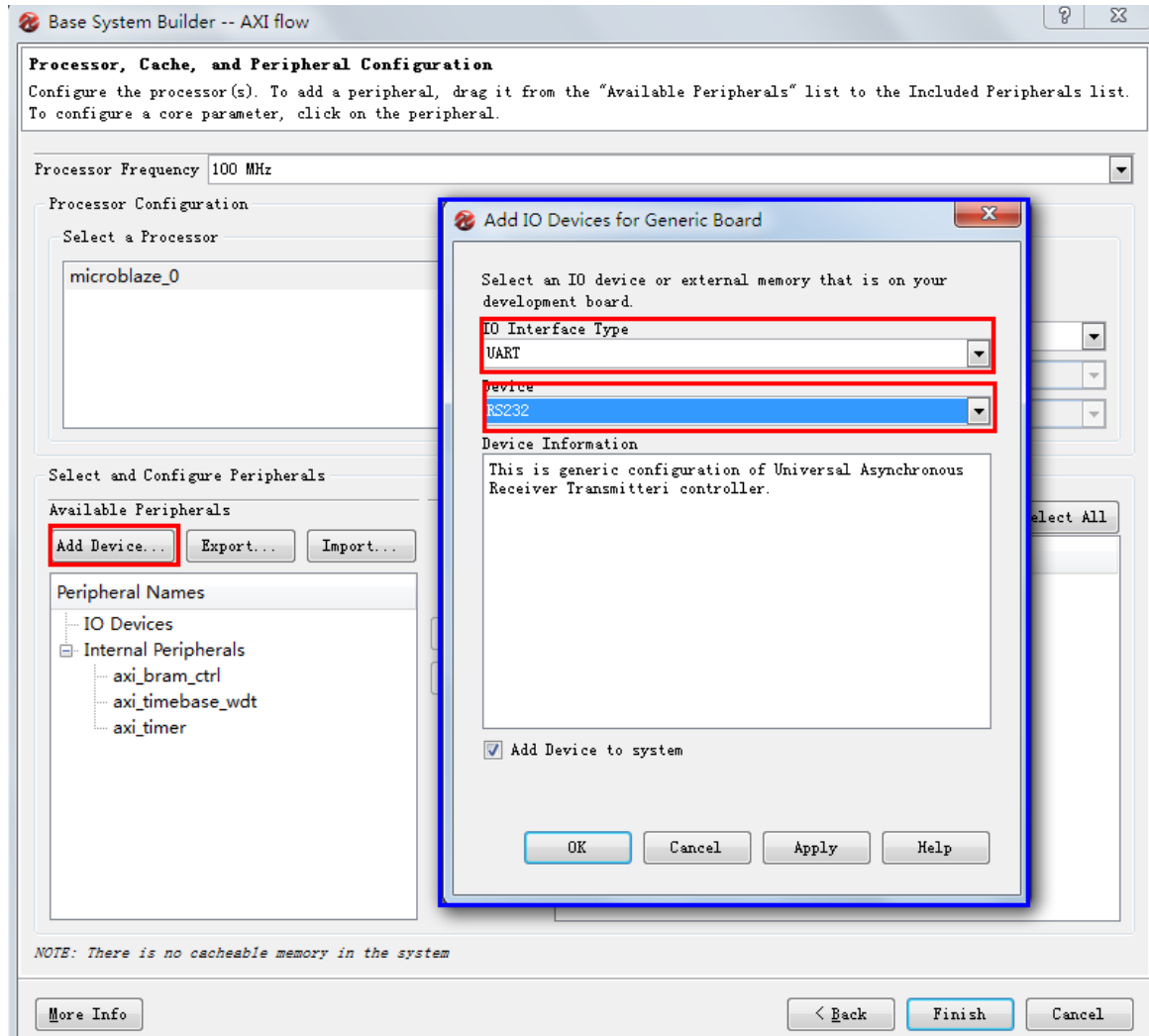


图 5. 添加串口的 IP

**1-1-6.** 注意,串口的默认波特率设置为9600。在Lab1中,不做修改。以后的设计中根据需要进行调整。但是为了确保串口的正常通讯, SDK工程中的Terminal的波特率以及串口的其他设置必须与之保持一致。

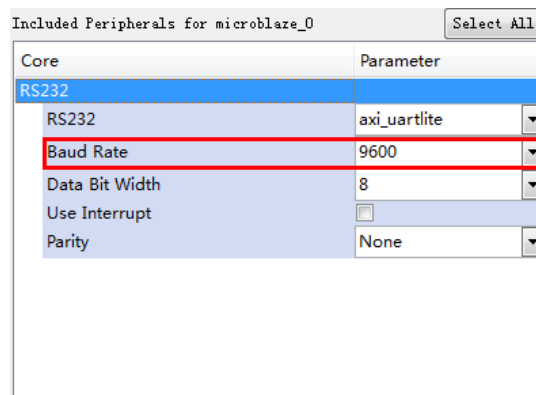


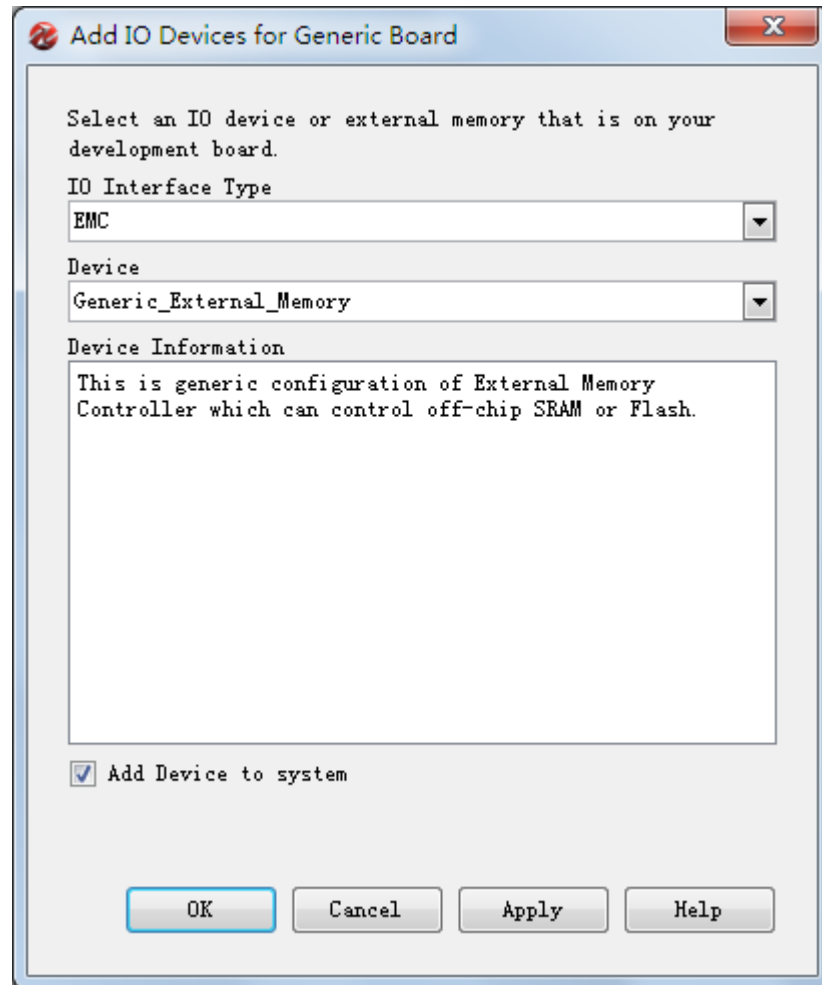
图 6. 串口的设置

### 1-1-7. 添加EMC-IP(external memory control)

在 **IO Interface Type** 中的下拉菜单中选择 **EMC**。

在 **Device** 的下拉菜单中选择 **Generic External Memory**。

单击 **OK**



---

## 第二步 进行端口的互连

---

### 2-1. 在 **PORT** 选项卡中修改时钟的相关设置

**2-1-1.** Port 选项卡（展开 **External Port**），如图 7.

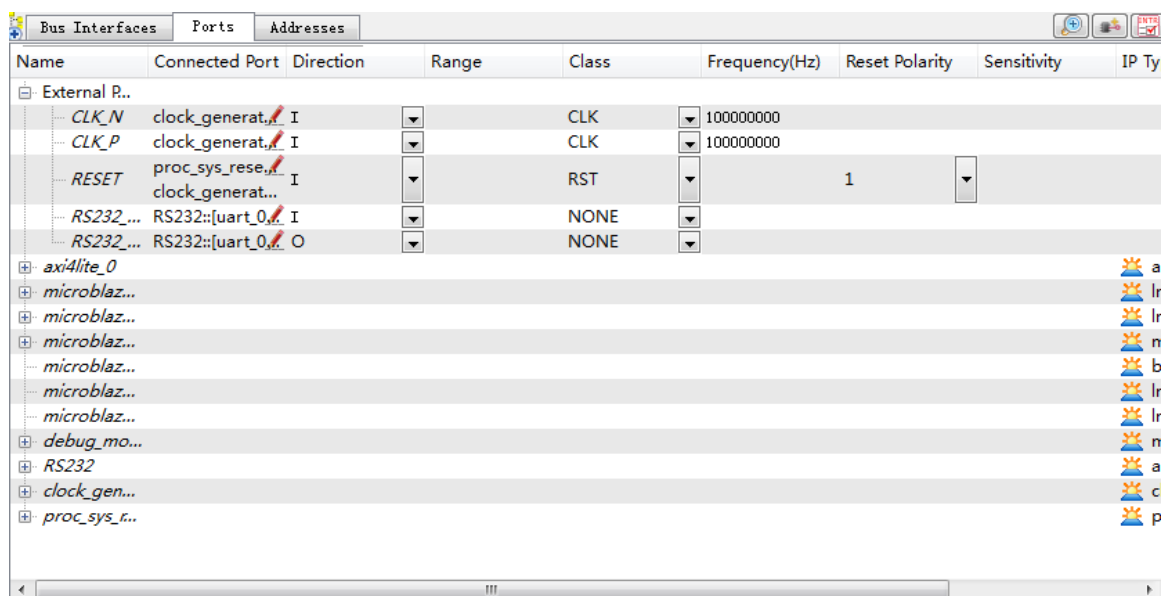


图 7: PORT 选项卡的初始状况

## 2-1-2. 将 External Port 中的 CLK\_N 和 CLK\_P 都去掉。

右键选中该端口，然后点击 **Delete External Port**，如图 8。

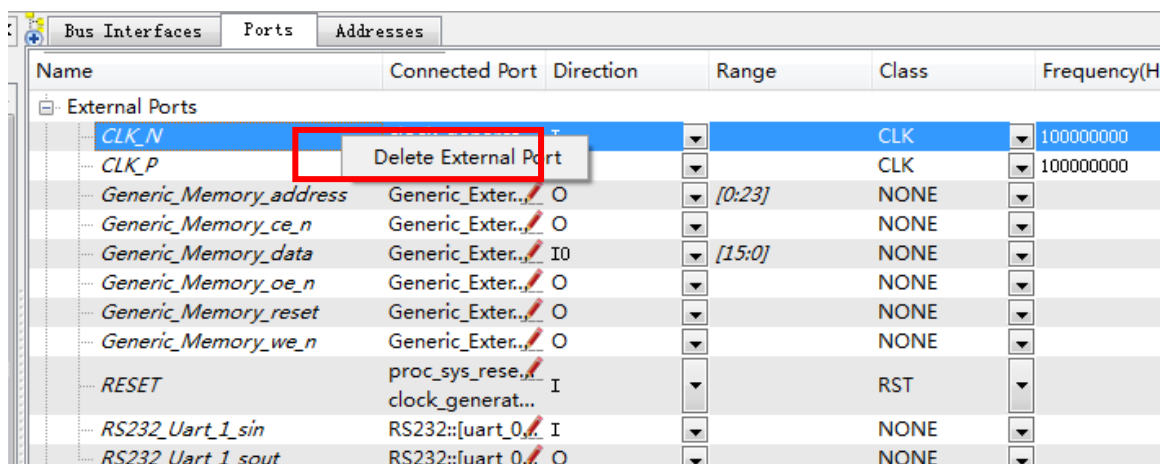


图 8: 源文件添加后的 ISE 用户界面



2-1-3.将 **Clock\_generator\_0** 作为新的时钟，加入外部端口。

找到 **Clock\_generator\_0** 中的 **CLKIN**，右键选中，在菜单中点击 **Make external**，如图 9

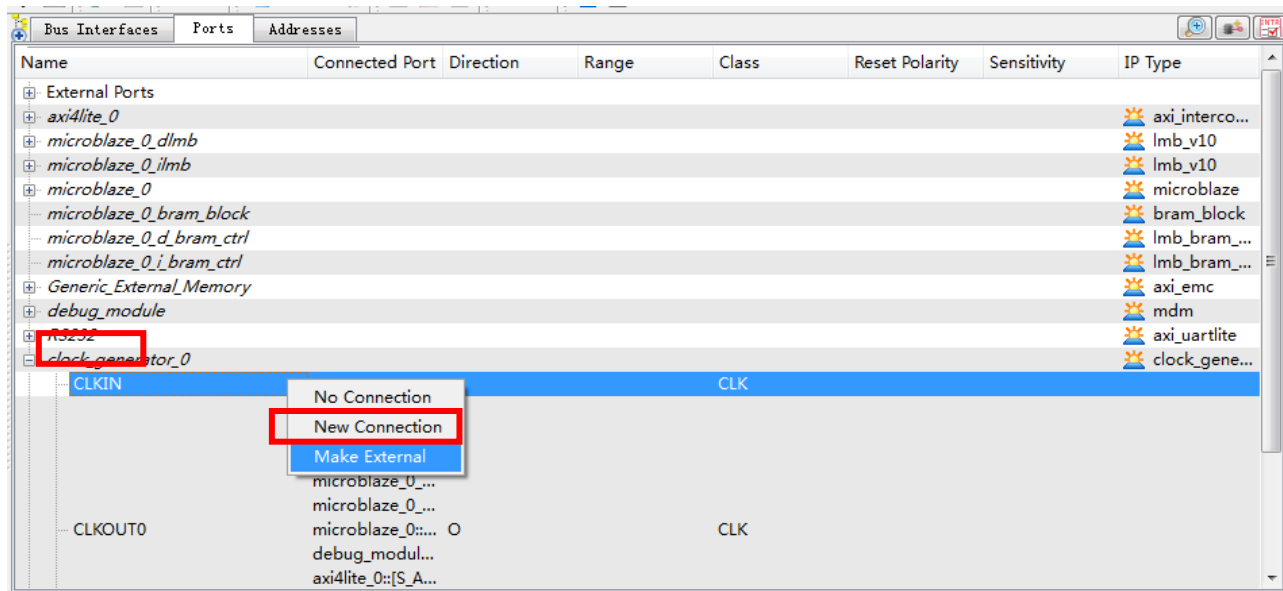


图 9: **Clock\_generator\_0** 中的 **CLKIN**

2-1-4.对 **Generic External Memory** 端口的调整，如图 10:

在 **Generic External Memory---(IO\_IF)emc\_0** 下找到 **Mem\_BEN** 右键，选择 **make external**。

这个端口在 8 bit memory test 的时候将决定是测试高八位还是低八位。如果没有将这个端口添加为外部端口，将导致 8bit memory test 失败。

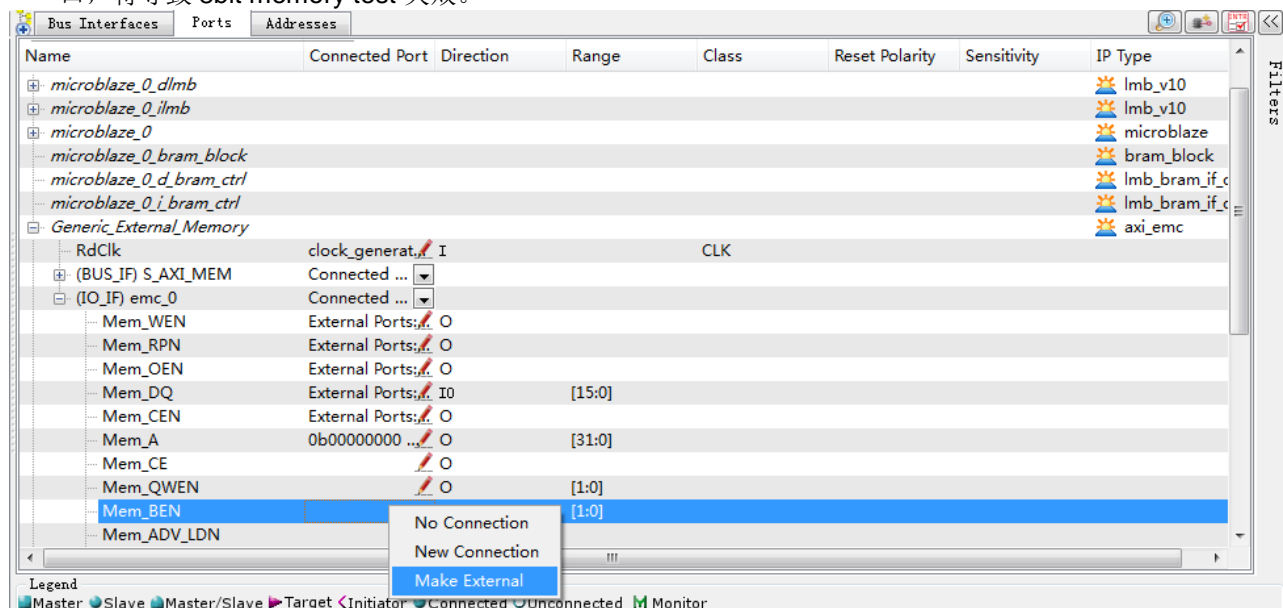


图 10

## 2-1-5. 对地址的位数进行修改：

由于原理图中的地址总线只有 23 位(0~22)。所以我们将 address 的范围修改为 0~22 （原本是 0~23），如图 11：

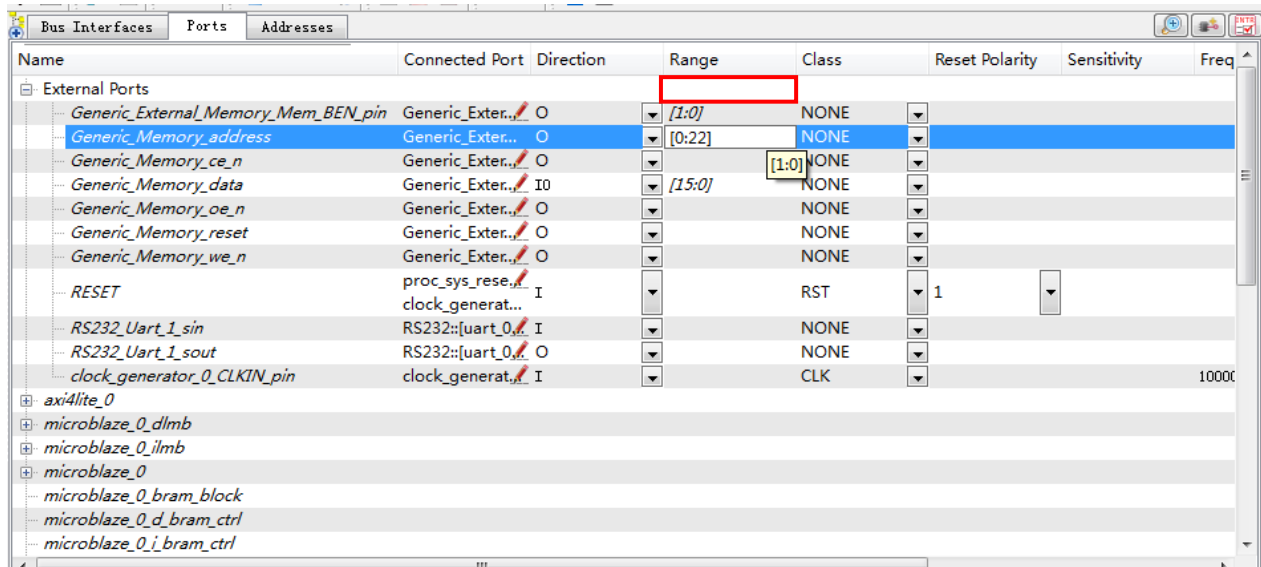


图 11

注意 External 中的 Name 一项，这是我们添加用户约束文件（UCF）的依据。

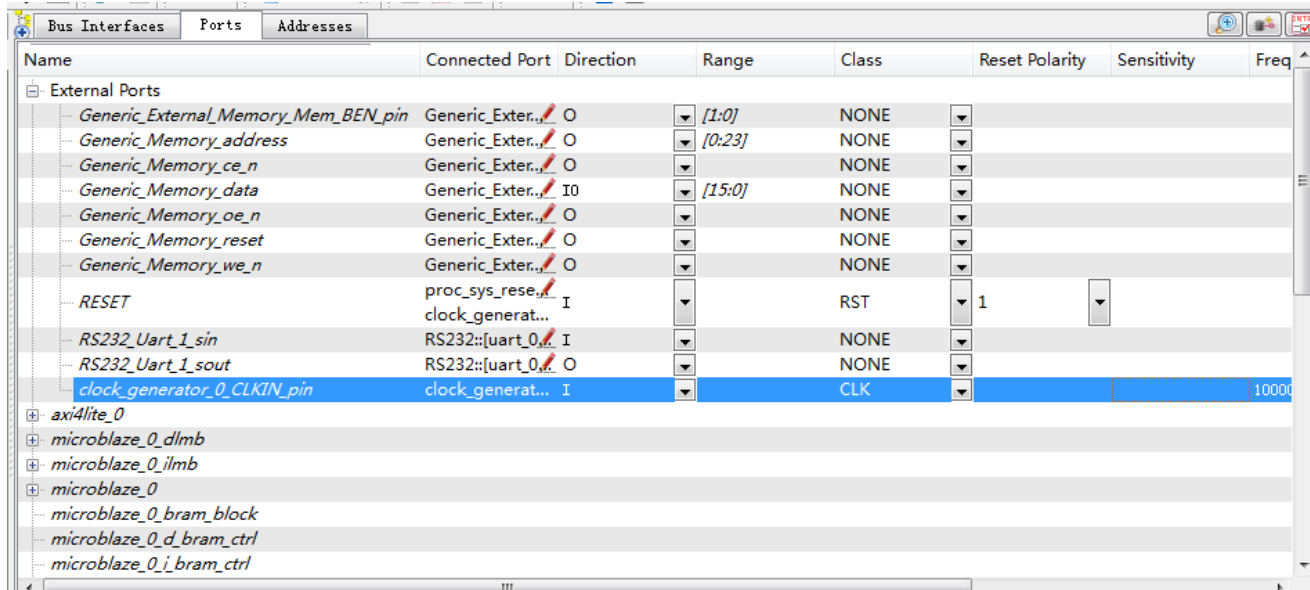


图 12: 修改后的 External PORT

## 第三步 修改 EMC 设置

### 3-1. 双击 external memory

Name	Connected Port	Direction
External Ports		
axi4lite_0		
microblaze_0_dlmb		
microblaze_0_ilmb		
microblaze_0		
microblaze_0_bram_block		
microblaze_0_d_bram_ctrl		
microblaze_0_i_bram_ctrl		
Generic_External_Memory		
debug_module		
RS232		
clock_generator_0		
proc_sys_reset_0		

图 13

### 3-2. 修改存储类型

在 memory type for bank 0 下拉菜单中选择 PSRAM，如图 14

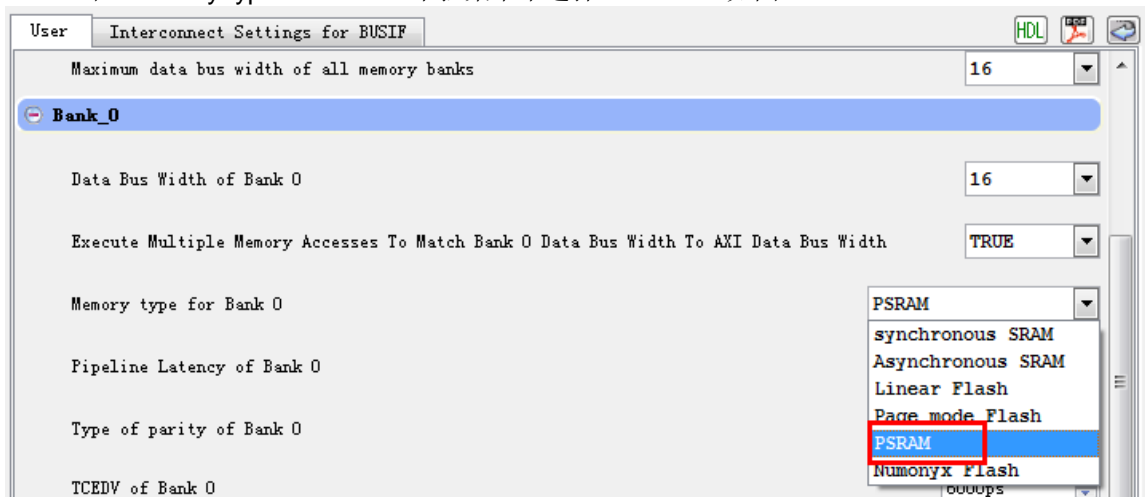


图 14

### 3.3. 勾选 axi 寄存器接口使能

打开 common 下拉菜单

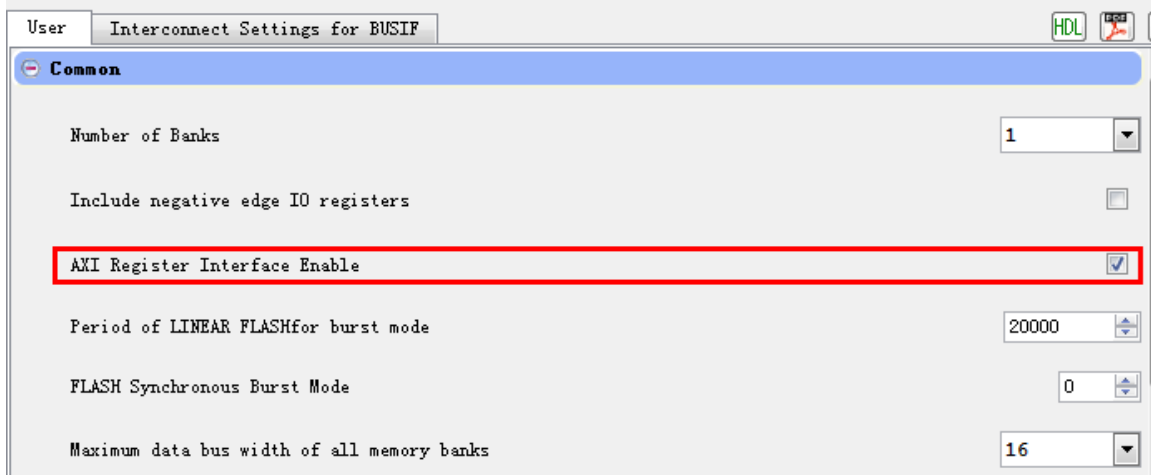


图 15

### 3.4. 修改相关参数:

点开 Bank0, 以下就是我们要修改的参数。

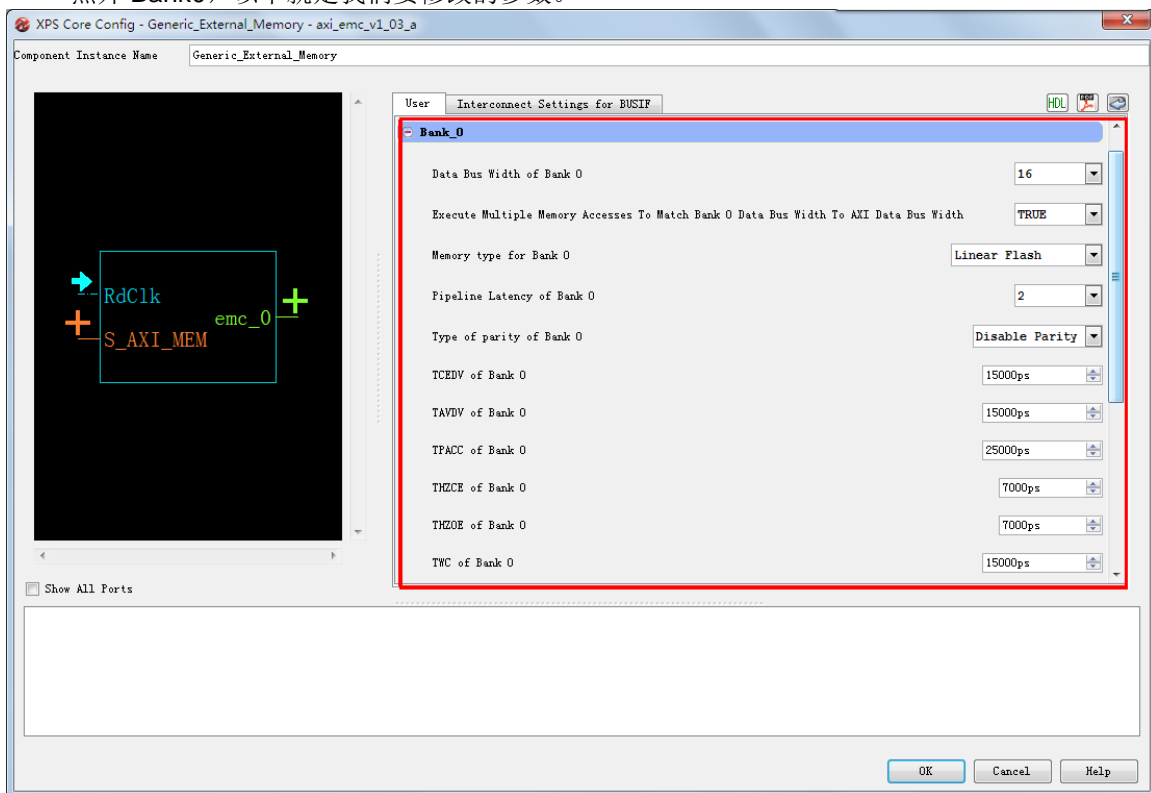


图 16

应该依据 psdram 的数据手册进行修改。

Nexys 4 的 psdram 的型号从原理图 ( Nexys 4\_sch.pdf ) 的第 9 页找到(MT45W8MW16BGX), 然后可以到 Micron 官网去找 pdf 版本的数据手册, 连接如下:

<http://www.micron.com/parts/psram/cellularram/mt45w8mw16bgx-701-it?source=ps>

我们主要通过数据手册给出的 Table 14、Table 15、Table16、Table 17 对号入座。（数据手册 37-40 页）  
以下给出我们的修改表：

Name in XPS	Meaning	Default value(ps)	Name in datasheet	Requirements (ns)	Modification(ps)
TCEDV of Bank0	Chip enable to data valid time	15000	tCO	Max: 70	→60000
TAVDV of Bank0	Address valid to data valid time	15000	tAADV	Max: 70	→6000
TPACC of Bank0	Page mode read time	25000	tPC	Min: 20	→25000
THZCE of Bank0	CE disable to data bus High Z time	7000	tHZ	Max:8	Remain unchanged
THZOE of Bank0	OE disable to data bus High Z time	7000	tOHZ	Max:8	Remain unchanged
TWC of Bank0	Write Cycle time	15000	tWC	Min: 70	→80000
TWP of Bank0	Min pulse width of write enable	12000	tWP	Min: 45	→45000
TWPH of Bank0	Min pulse high width of write enable	12000	tWPH	Min: 10	→10000
TLZWE of Bank0	WE disable to data bus low Z	0	tOW	Min: 5	→6000
Write recovery of Bank0	Write recovery	100000	WRITE recovery time	Min: 0	→10000

修改后，图 17，然后点击 **OK**：

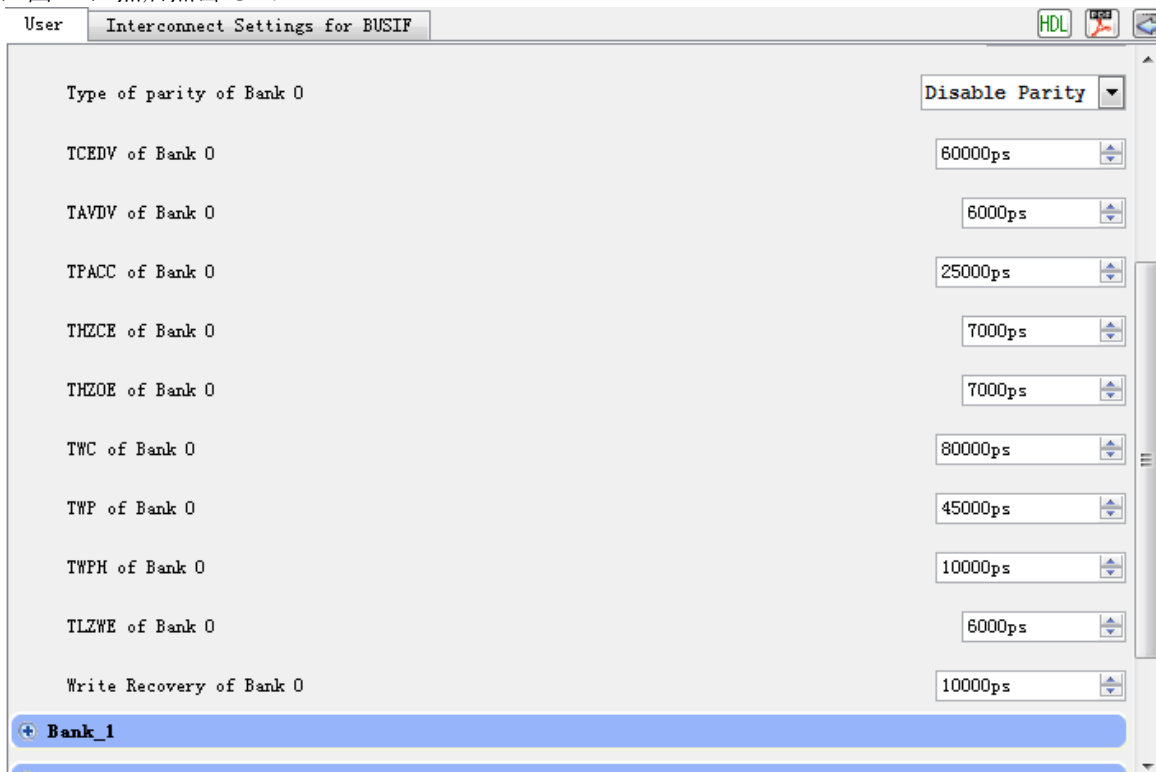


图 17

## 第四步 添加用户约束文件

### 4-1. 打开初始 UCF 文件，根据需求进行修改

- 4-1-1. 在页面偏左找到 IP catalogue / Project 选项卡，双击 UCF File: DATA\lab1.ucf，ucf 文件在右侧打开，如图 18

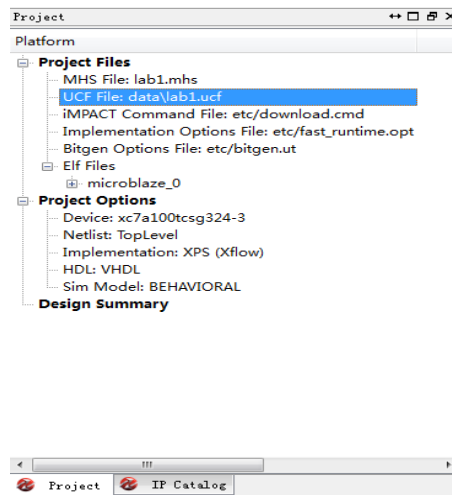


图 18: UCF 文件的位置

- 4-1-2. 这里我们手动输入 LOC（引脚位置）约束代码，如图 19。点击保存。

在编写完下图用户约束文件引脚约束代码之后，XPS 设计中的那些外部端口就连接到了 FPGA 芯片的相关引脚上，从而与 Nexys 4 板上的外设联系起来。具体应该如何链接 UCF 文件可以参考原理图。例如管脚 Generic\_Memory\_ce\_n，我们在原理图的第 7 页，看到他 L18 相连

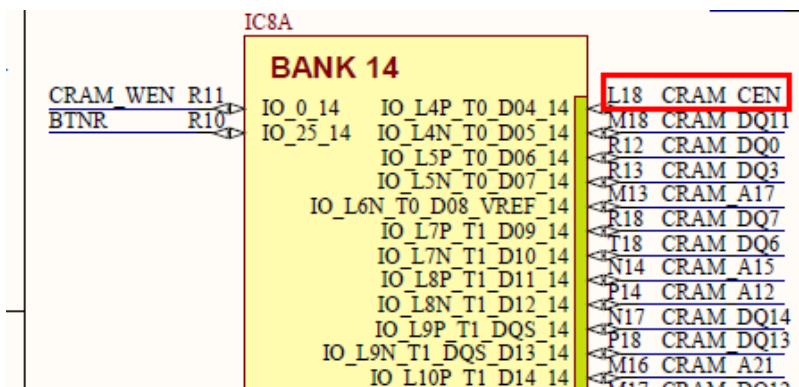


图 19

这里要注意：不能参考原理图第 9 页，而将 Generic\_Memory\_ce\_n 与 B5 管脚相连。因为第九页的图描述的是芯片 MT45W8MW16BGX 的管脚，而非 Nexys 4 的。而 bank 描述的则是 Nexys 4 上管脚封装的情况。如图 20。

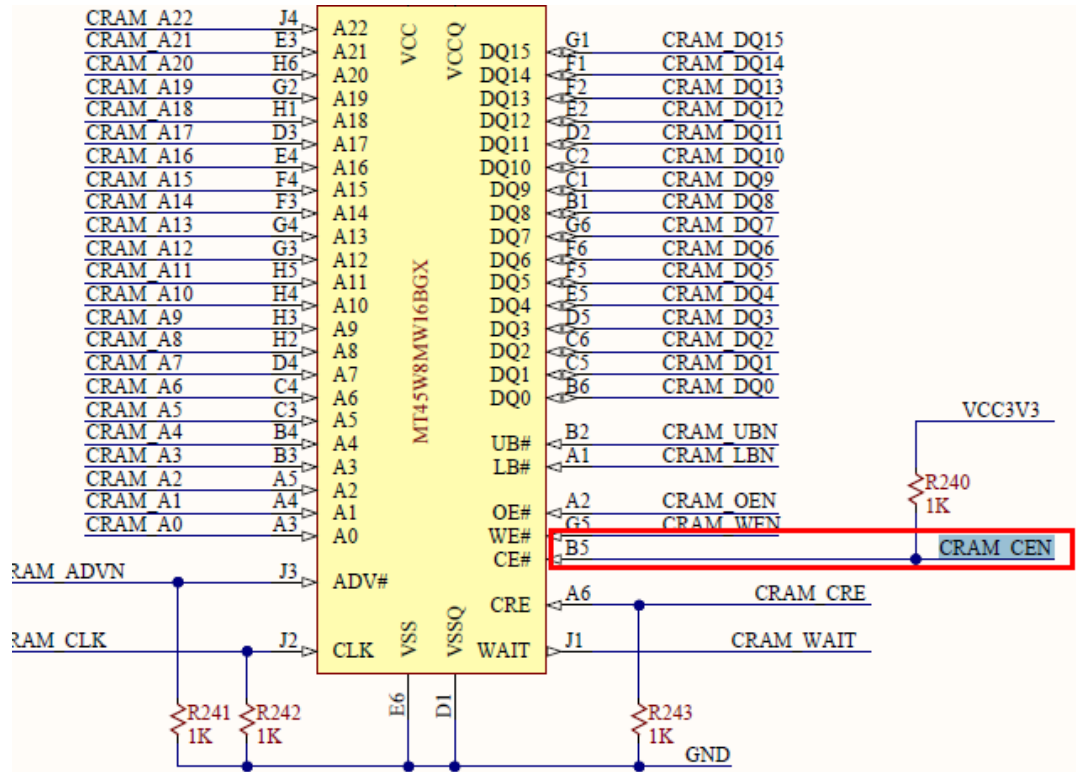


图 20

#### 4-1-3. 关于 BEN 管脚的问题：

如果在原理图中直接搜索 BEN 管脚，不会找到任何结果。因为 Ben (Byte Enable) 是一个两位的信号，被分别传给了 UB (upper Byte) 和 LB (lower Byte)。因此可以搜索 UBN(n 代表低电平有效) 和 LBN 来找到。

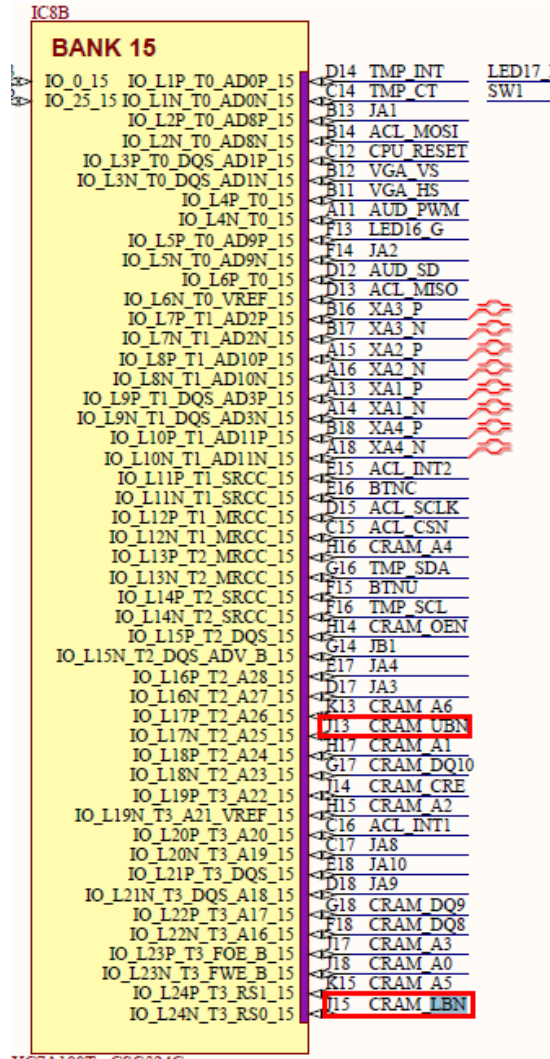


图 21

至于 Ben 的两位中，到底是哪位给了 UBN，哪位给了 LBN 还要通过查看另一个 PDF 文件得到。

在 port 端口选项卡中右键 **Generic External Memory**，然后选择 **view pdf datasheet**。如图 21



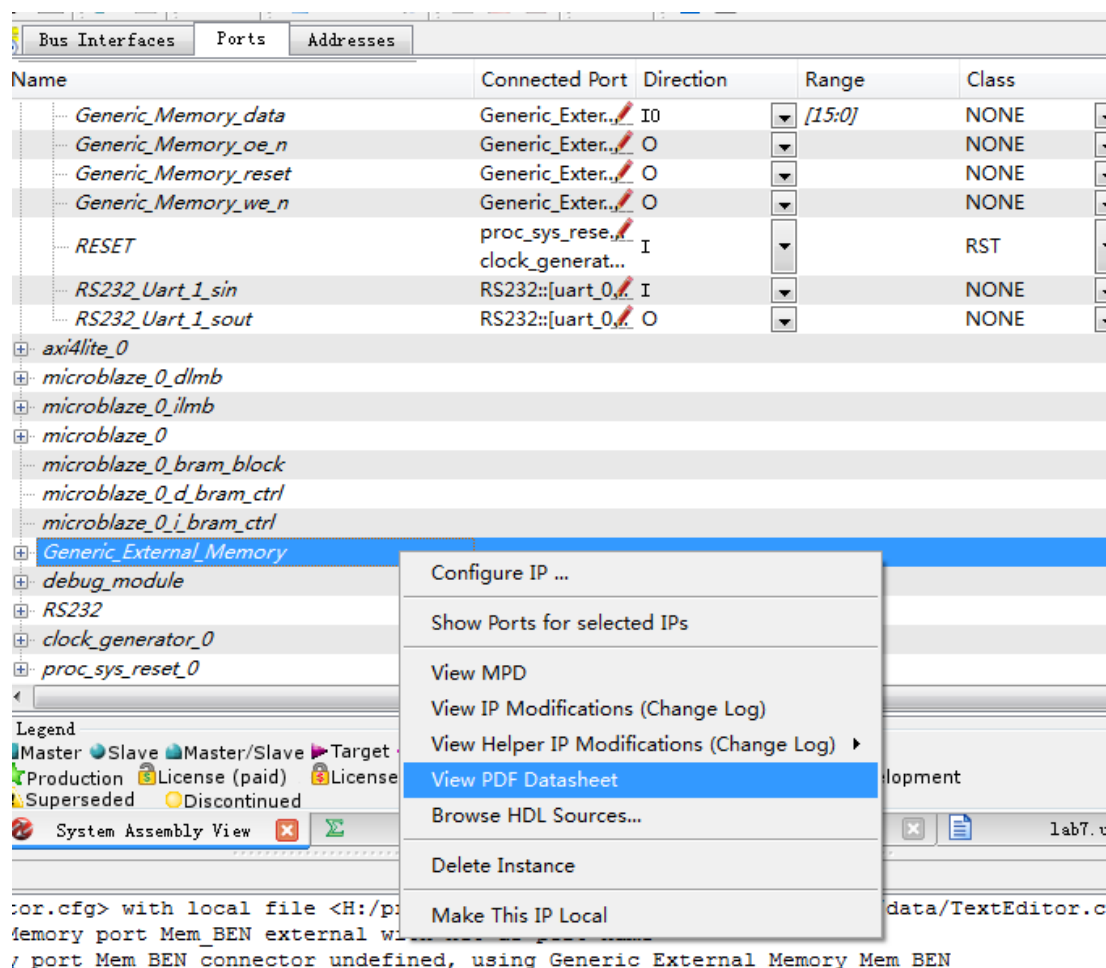


图 22

在后在自动打开的 pdf 页面中点击 click here，即可在线打开最新版的数据手册。数据手册的第 28 页的 table 21 中，可以看到：如图 23

Table 21: Connection to 16-bit Memory Using PSRAM Parts

DN	Description	AXI EMC Signals (MSB:LSB)	PageModeFlash Signals (MSB:LSB)
0	Data bus	MEM_DQ(15 : 0)	DQ(15 : 0)
	Address bus	MEM_A(22 : 1)	A(21 : 0)
	Chip enable (active Low)	MEM_CEN(0)	CE
	Output enable (active Low)	MEM_OEN	OE#
	Write enable (active Low)	MEM_QWEN(0)	WE#
	Reset/Power down (active Low)	MEM_RPN	RP#
	Byte Enable (active Low)	Mem BEN(1:0)	UB#, LB#
	Control Register Enable (active High)	Mem_CRE	CRE

图 23

显然 Ben 的高位与 UB 相连，低位与 LB 相连。

最终版 UCF 文件：（如图 24）

```
## Clock signal
NET "clock_generator_0_CLKIN_pin" LOC = "E3" | IOSTANDARD = "LVCMOS33";
NET "clock_generator_0_CLKIN_pin" TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100 MHz HIGH 50%;
## Switches
NET "RESET" LOC = "U9" | IOSTANDARD = "LVCMOS33"; #Bank = 34, Pin
## USB-RS232 Interface
NET "RS232_Uart_1_sin" LOC = "C4" | IOSTANDARD = "LVCMOS33"; #
NET "RS232_Uart_1_sout" LOC = "D4" | IOSTANDARD = "LVCMOS33"; #
## Cellular RAM
NET "Generic_Memory_ce_n" LOC = "L18" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_oe_n" LOC = "H14" | IOSTANDARD = "LVCMOS33"; #Bank = 15, Pin
NET "Generic_Memory_we_n" LOC = "R11" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_External_Memory_Mem_BEN_pin<0>" LOC = "J15" | IOSTANDARD = "LVCMOS33";
NET "Generic_External_Memory_Mem_BEN_pin<1>" LOC = "J13" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_data<0>" LOC = "R12" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_data<1>" LOC = "T11" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_data<2>" LOC = "U12" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_data<3>" LOC = "R13" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_data<4>" LOC = "U18" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_data<5>" LOC = "R17" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_data<6>" LOC = "T18" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_data<7>" LOC = "R18" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_data<8>" LOC = "F18" | IOSTANDARD = "LVCMOS33"; #Bank = 15, Pin
NET "Generic_Memory_data<9>" LOC = "G18" | IOSTANDARD = "LVCMOS33"; #Bank = 15, Pin
NET "Generic_Memory_data<10>" LOC = "G17" | IOSTANDARD = "LVCMOS33"; #Bank = 15, Pin
NET "Generic_Memory_data<11>" LOC = "M18" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_data<12>" LOC = "M17" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_data<13>" LOC = "P18" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_data<14>" LOC = "N17" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_data<15>" LOC = "P17" | IOSTANDARD = "LVCMOS33"; #Bank = 14, Pin
NET "Generic_Memory_address<0>" LOC = "J18" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<1>" LOC = "H17" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<2>" LOC = "H15" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<3>" LOC = "J17" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<4>" LOC = "H16" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<5>" LOC = "K15" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<6>" LOC = "K13" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<7>" LOC = "N15" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<8>" LOC = "V16" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<9>" LOC = "U14" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<10>" LOC = "V14" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<11>" LOC = "V12" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<12>" LOC = "P14" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<13>" LOC = "U16" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<14>" LOC = "R15" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<15>" LOC = "N14" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<16>" LOC = "N16" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<17>" LOC = "M13" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<18>" LOC = "V17" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<19>" LOC = "U17" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<20>" LOC = "T10" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<21>" LOC = "M16" | IOSTANDARD = "LVCMOS33";
NET "Generic_Memory_address<22>" LOC = "U13" | IOSTANDARD = "LVCMOS33";
```

图 24: UCF 文件

#### 4-1-4. 保存之后将工程导入到 SDK

在页面左边，点击 **Export Design**，如图 25。

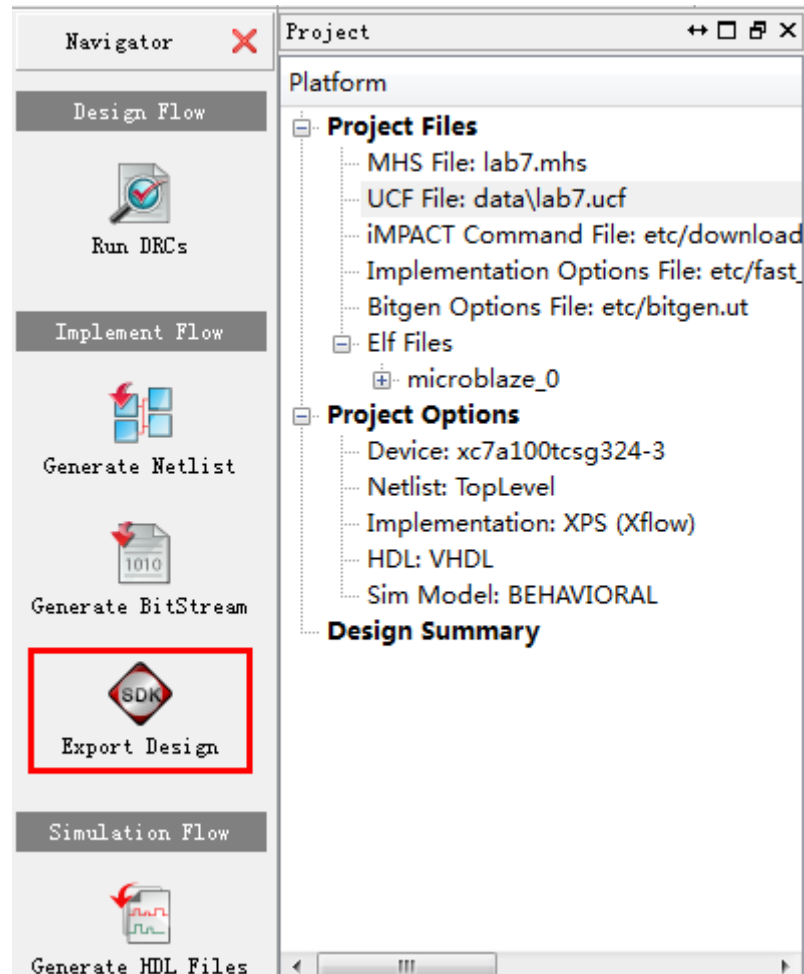


图 25: export design

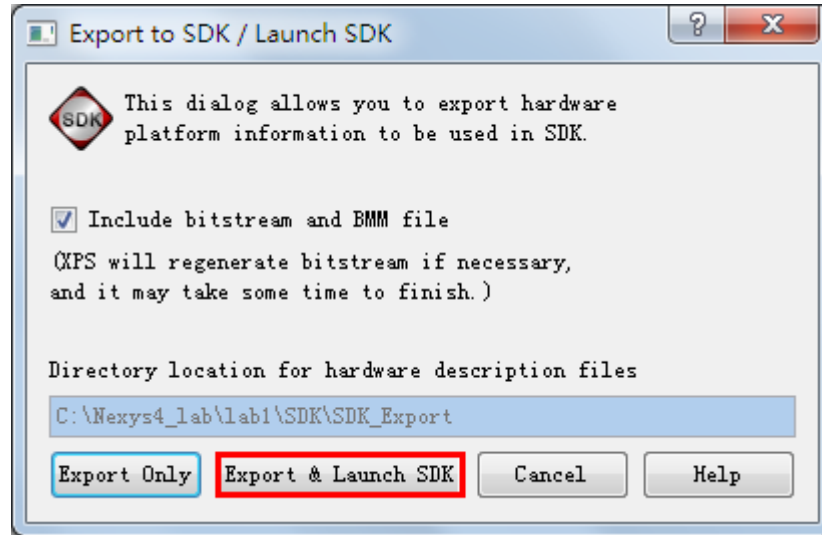


图 26：在弹出的对话框中选择 **Export & launch sdk**

#### 3-1-4. 选择 SDK 导入路径

注意要具体到..\sdk\sdk\_export，如图 27

点击 **ok**

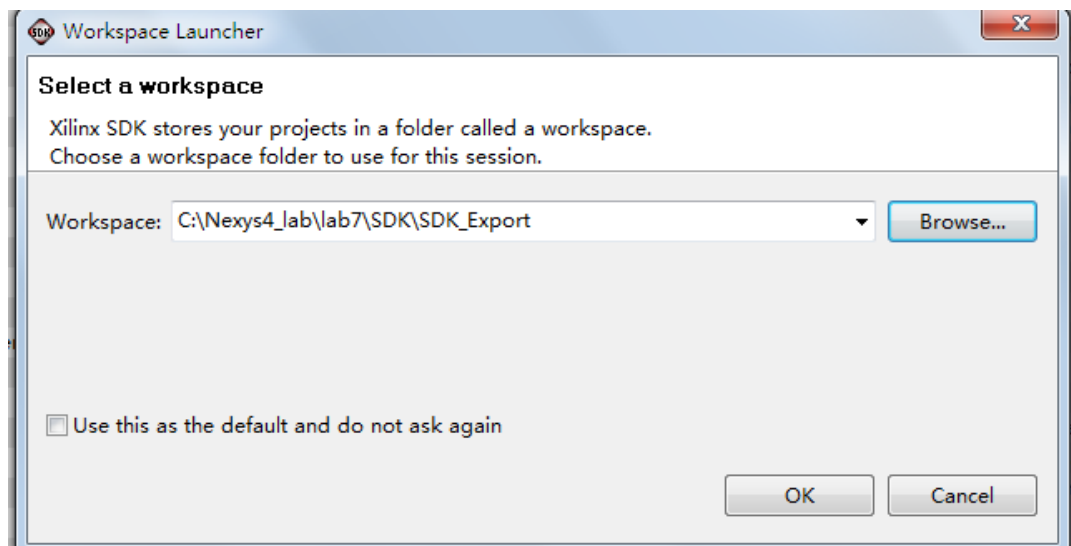


图 27：导入 **sdk** 的工作空间选择

## 第四步 添加 app

### 5-1. 添加软件应用。

5-1-1. 在 SDK 的用户界面中，选择 **file—new—application project**，如图 28。

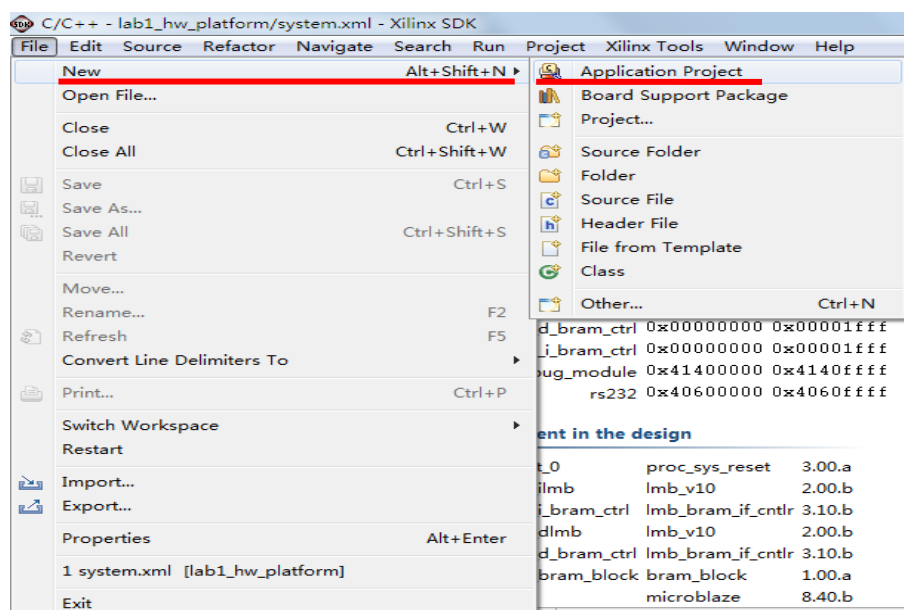


图 28: 新建软件应用

5-1-2. 输入工程的名称，这里使用 **mem\_test**，同样不要包含空格和中文，点击 **next**

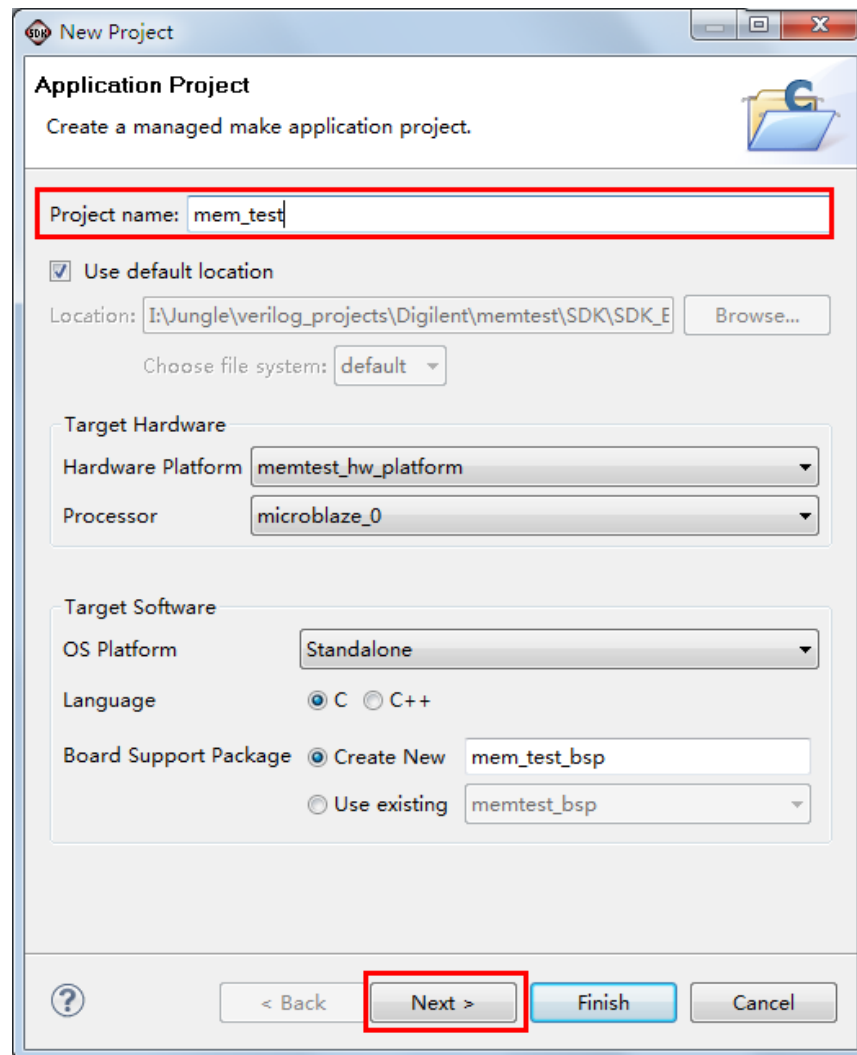


图 29：新建工程---命名

5-1-3. 在下一步弹出的对话框中选择 **memory tests**，然后点击 **finish**

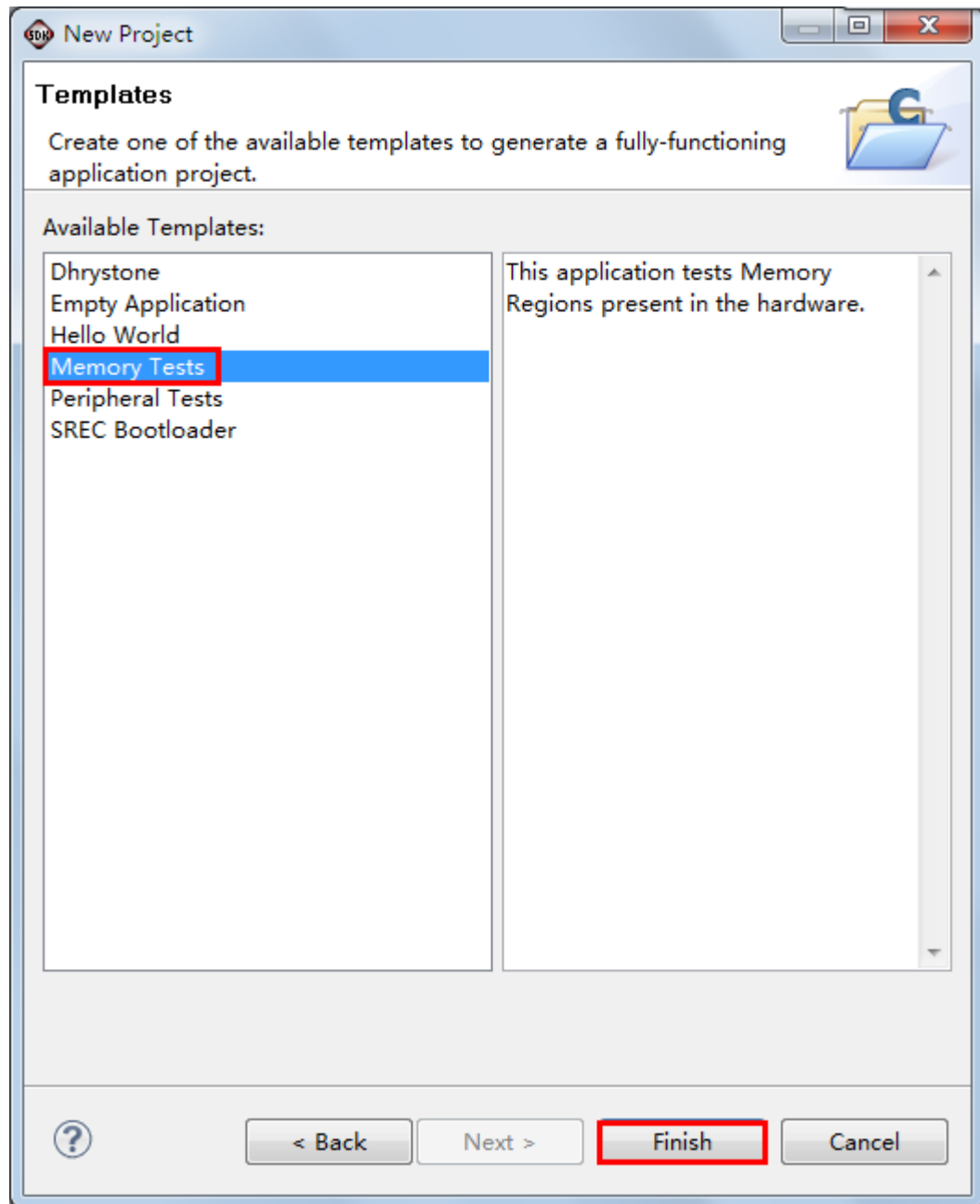
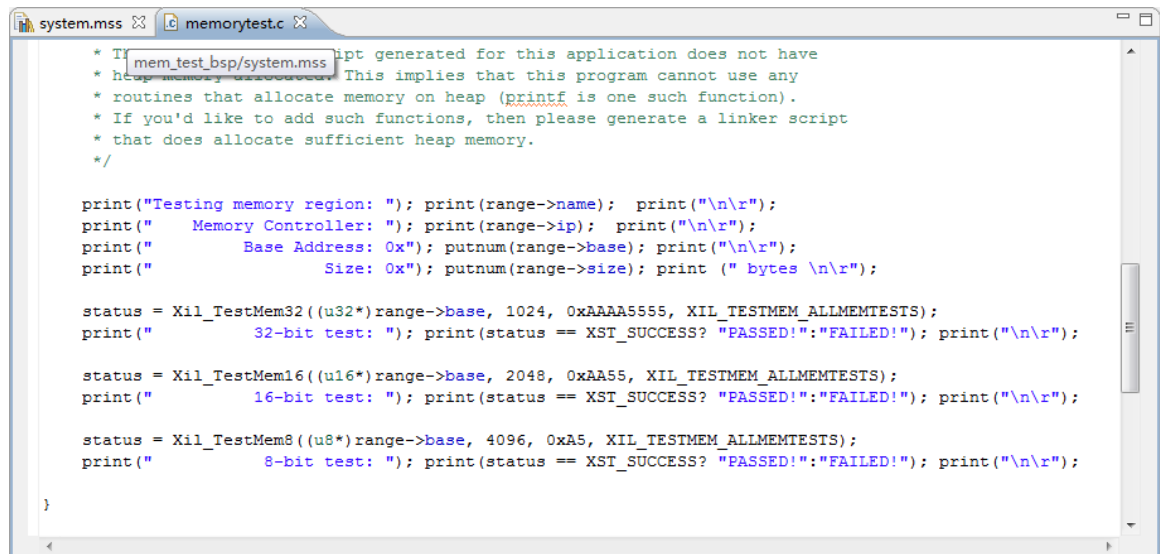


图 29: 选择 **memory test** 示例代码

5-1-4. 添加完毕以后可以在左侧双击源文件，查看这段代码：

可以根据自己的需要进行一些修改，修改后保存。

原文件代码总共针对 32bit, 16bit 和 8bit 做了 3 种不同的 test。就是想一段特定存储单元中, 写入一个内容, 然后在读出。如果写入内容和读出内容一样, 则说明测试成功。



```

* This file was generated by the Digilent IDE. The linker script generated for this application does not have
* heap memory allocated. This implies that this program cannot use any
* routines that allocate memory on heap (printf is one such function).
* If you'd like to add such functions, then please generate a linker script
* that does allocate sufficient heap memory.
*/

print("Testing memory region: "); print(range->name); print("\n\r");
print("    Memory Controller: "); print(range->ip); print("\n\r");
print("    Base Address: 0x"); putnum(range->base); print("\n\r");
print("    Size: 0x"); putnum(range->size); print (" bytes \n\r");

status = Xil_TestMem32((u32*)range->base, 1024, 0xAAAA5555, XIL_TESTMEM_ALLMEMTESTS);
print("    32-bit test: "); print(status == XST_SUCCESS? "PASSED!":"FAILED!"); print("\n\r");

status = Xil_TestMem16((u16*)range->base, 2048, 0xAA55, XIL_TESTMEM_ALLMEMTESTS);
print("    16-bit test: "); print(status == XST_SUCCESS? "PASSED!":"FAILED!"); print("\n\r");

status = Xil_TestMem8((u8*)range->base, 4096, 0xA5, XIL_TESTMEM_ALLMEMTESTS);
print("    8-bit test: "); print(status == XST_SUCCESS? "PASSED!":"FAILED!"); print("\n\r");
    
```

图 30: memory test 源代码

## 第五步 上板验证

### 6-1. 将 Nexys4 与 Pc 的 USB 接口连接

### 6-2. 查看端口号:

右键“我的电脑”——“属性”——在页面左侧选择“设备管理器”

发现与 com16 端口相连: (不同电脑可能有所区别, 同一电脑每次连接也有可能有所区别)

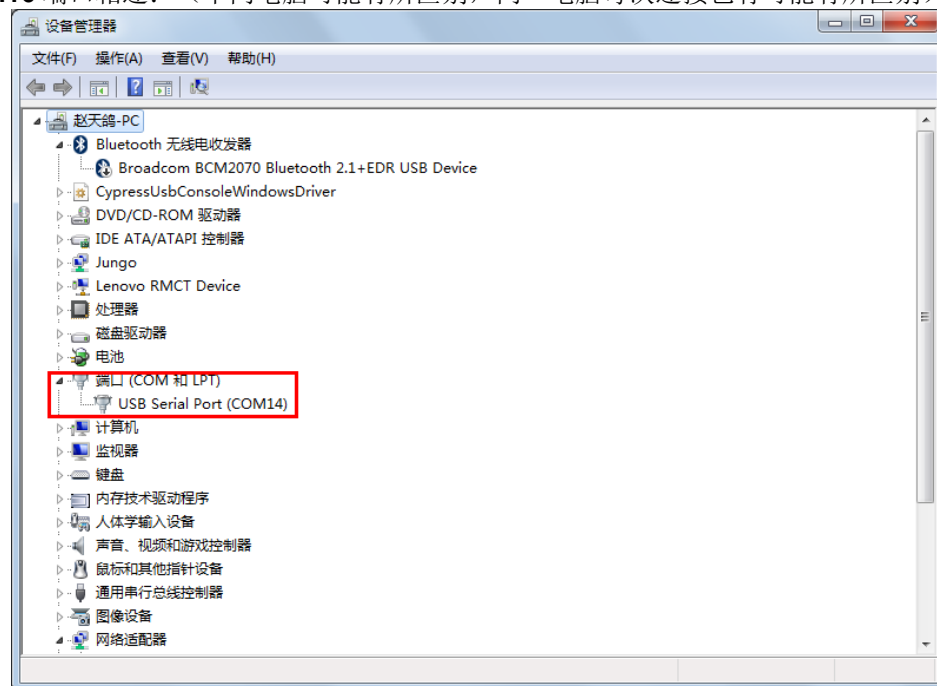


图 30



### 6-3. 在 SDK 中打开串口：

6-3-1. 在下面的在页面下方找到 **terminal** 选项卡，然后点击绿色的连接按钮。

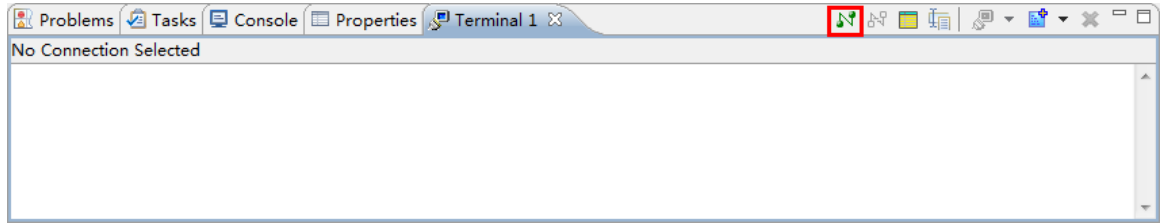


图 31

6-3-2. 按照端口号和 XPS 中的波特率（baud rate）进行如下设置：

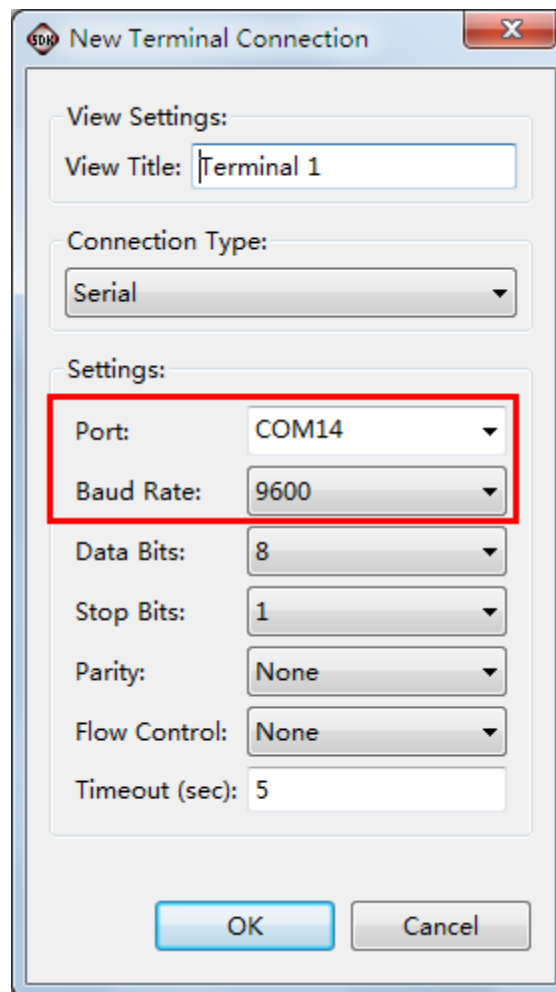


图 32

如果报了“no such port”的错误，可以通过新建串口，更改串口号：

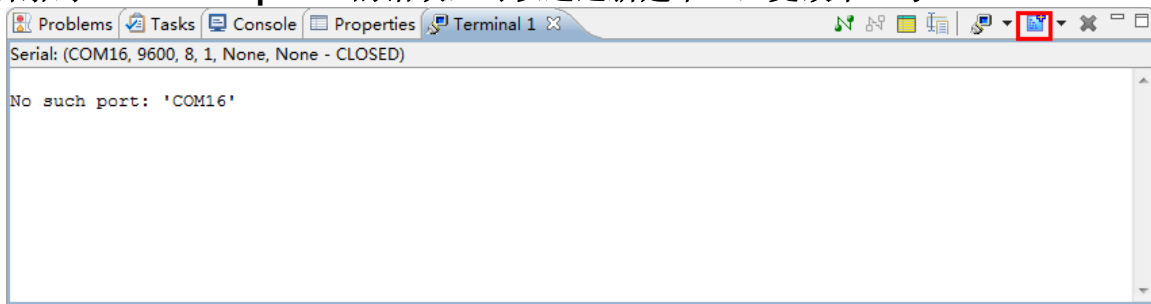


图 32

## 6-4. 将程序下载到板子上并运行

6-4-1.在页面上方，xilinx tools 下拉菜单中选择 program fpga

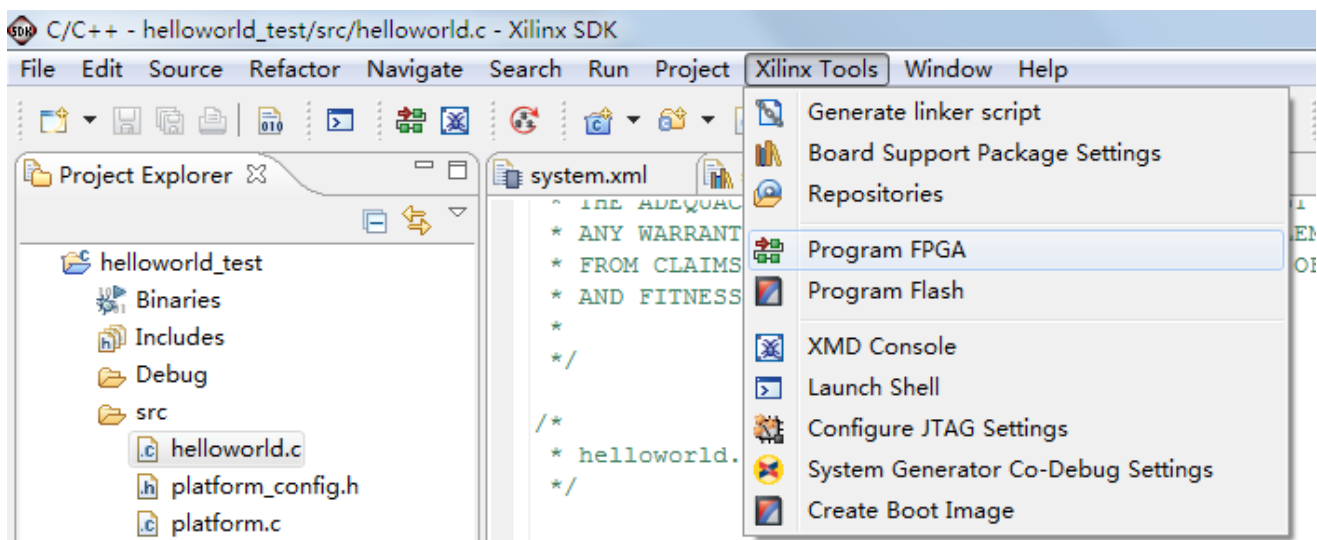


图 33

6-4-2.注意要选择正确的 elf 文件，然后点击 **program**：

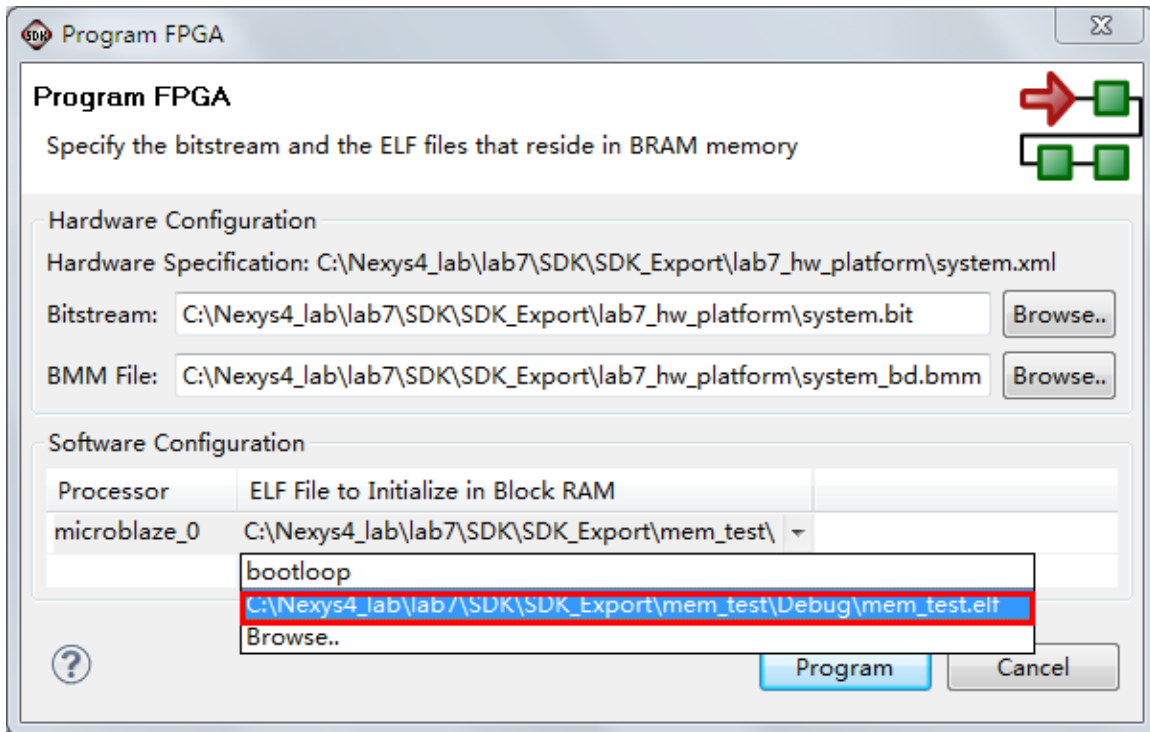
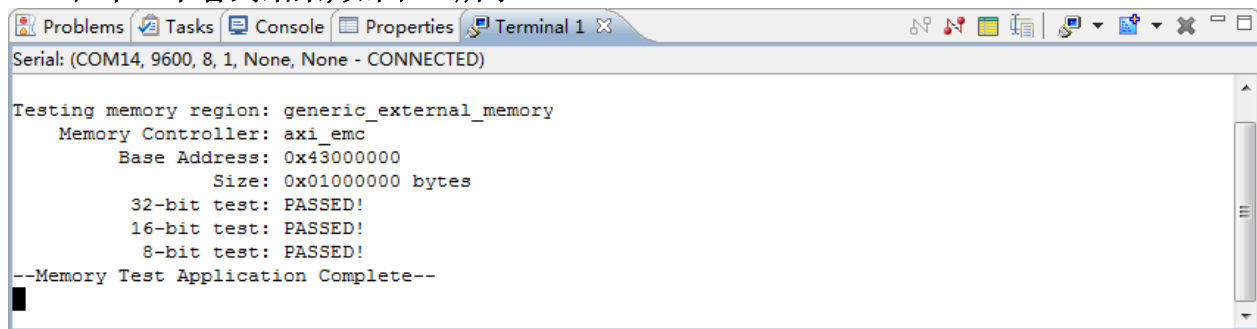


图 34

**6-5. 在串口中看到结果, 如图 35 所示:**



```
Problems Tasks Console Properties Terminal 1 X
Serial: (COM14, 9600, 8, 1, None, None - CONNECTED)

Testing memory region: generic_external_memory
  Memory Controller: axi_emc
    Base Address: 0x43000000
    Size: 0x01000000 bytes
    32-bit test: PASSED!
    16-bit test: PASSED!
    8-bit test: PASSED!
--Memory Test Application Complete--
```

图 35