

# *Lab6: 用户自定义 IP 核完成 AD/DA 转换*

*基于 Nexys 4 FPGA 平台*

## Lab 6: 用户自定义 IP 完成 AD/DA 转换

### 实验简介

本实验旨在使读者学会 Xilinx 的 XPS 和 SDK 工具的使用，学会建立并添加用户自定义 IP 核，完成 AD/DA 的输入输出转换

### 实验目标

在完成本实验后，您将学会：

- 建立并添加用户自定义 IP 核的具体流程
- AD/DA 模块的使用

### 实验过程

本实验旨在指导读者使用 Xilinx 的 XPS 工具，建立并添加用户自定义 IP 核，模数转换 IP 核和数模转换 IP 核，并将其导入到 SDK，调用这个两个 IP 核，完成数字信号转换成模拟信号，模拟信号再转换成数字信号的过程，然后在 Nexys 4 平台上进行测试验证。

实验由以下步骤组成：

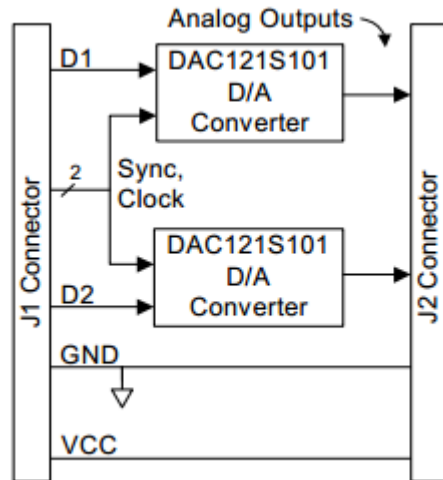
1. 在 XPS 中建立工程
2. 添加串口 IP 核
3. 建立用户自定义 IP 核
4. 添加用户自定义 IP 核
5. 进行端口的互连
6. 将工程导入到 SDK
7. 在 SDK 中添加 c 语言源程序
8. 在 Nexys 4 上进行测试验证

### 实验环境

#### ◆ 硬件环境

1. PC 机
2. Nexys 4 FPGA 平台

### 3. 数模转换器：Pmod-DA2—两路 12 位 D/A 输出接口

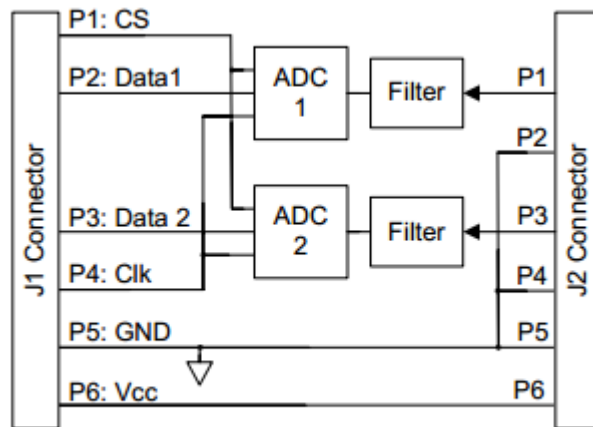


两个 National Semiconductor DAC121S101，12 位 D/A 转换器

6 针接口和 6 针插口

两个同步 D/A 转换通道

### 4. 模数转换器：Pmod-AD1—两路 12 位 A/D 输入接口



**AD1 Circuit Diagram**



两个两极 Sallen-Key 抗干扰滤波器

两个同步 A/D 转换通道，每通道可高达一 MSa

#### ◆ 软件环境

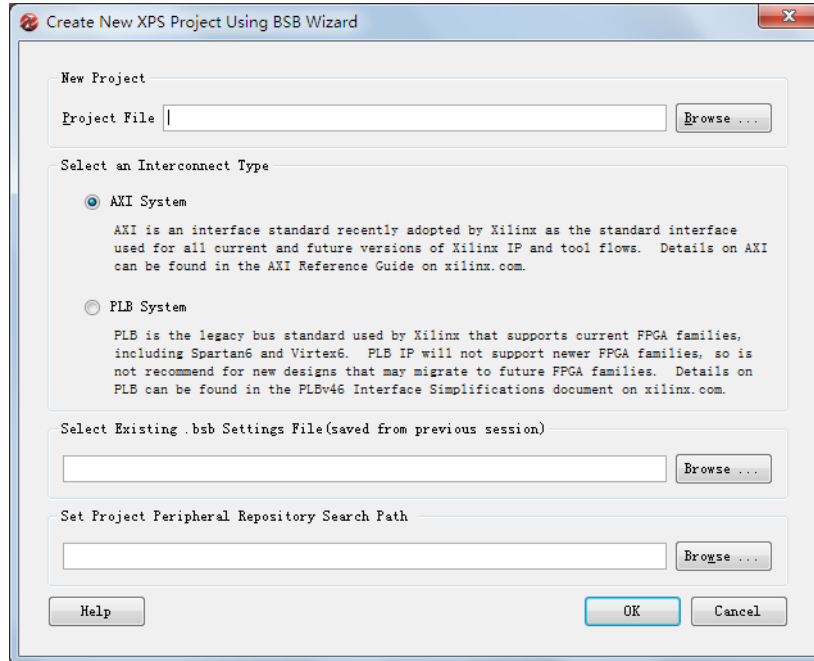
Xilinx ISE Design Suite 14.3 (FPGA 开发工具)

注：所有源代码可在 \*\*/lab6/Sources/ 目下找到

## 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



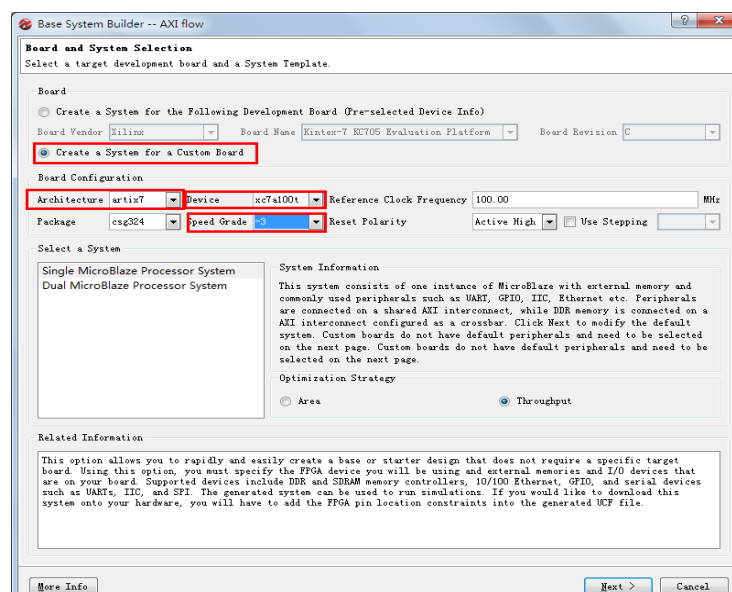


图：新工程建立向导

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 \*\*\*\*\*\Nexys4\_lab\lab6， 可以将 **system.xmp** 改成所建立工程的名字，这里取 **lab6**(名字中不要有中文和空格)，于是 **Project File** 栏中的路径变为\*\*\*\*\*\Nexys4\_lab\lab6\lab6。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

**architecture: artix 7**  
**Device: XC7a1007**  
**Package: CSG324**  
**Speed: -3**



图：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 IP 核，并设置 IP 核的参数：

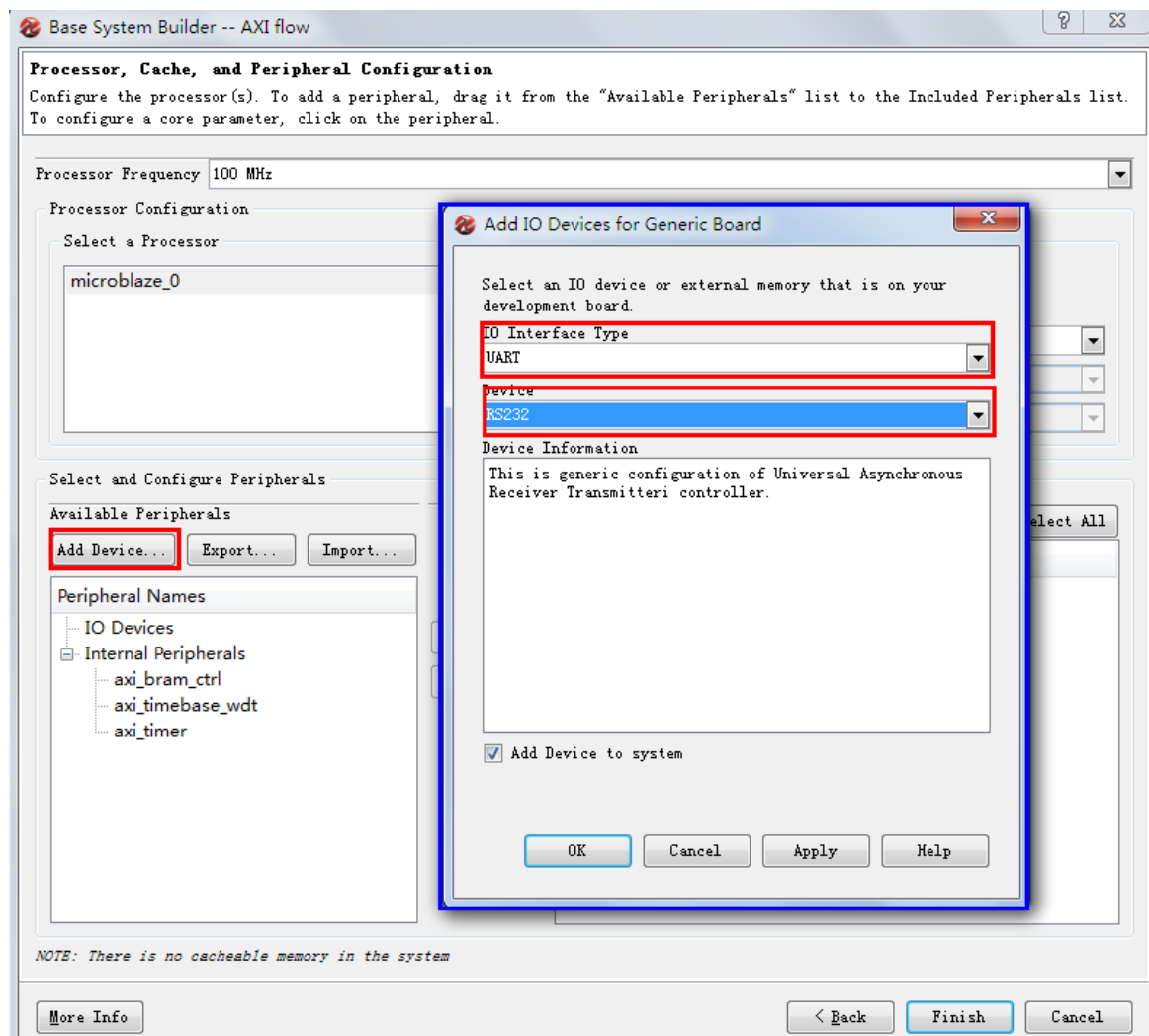
单击 **Select and configure Peripherals** 下的 **Add Device...**

出现图中的蓝色对话框。

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

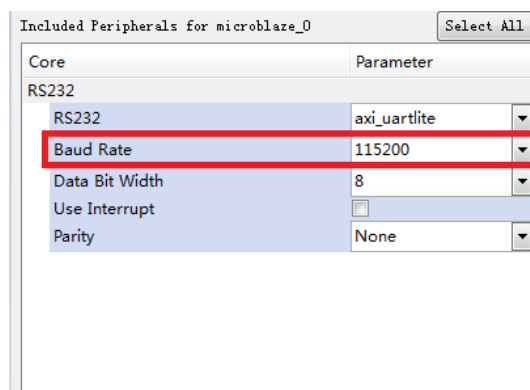
在 **Device** 的下拉菜单中选择 **RS232**。

单击 **OK**



图：添加串口的 IP

- 1-1-6.** 注意,串口的默认波特率设置为 115200。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。

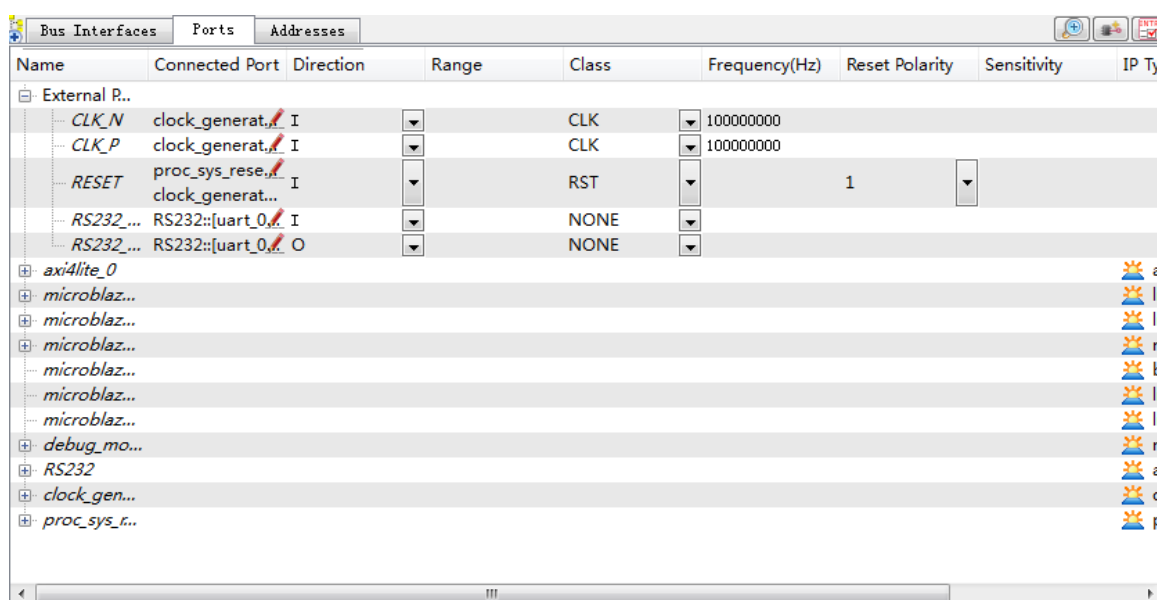


图：串口的设置

## 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

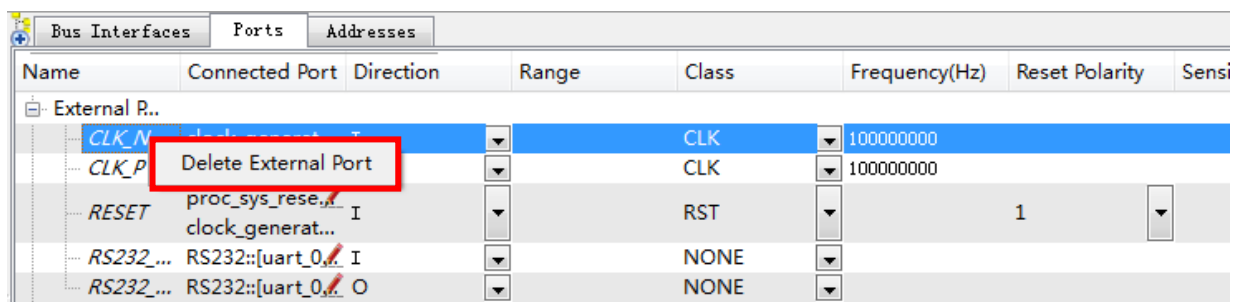
#### 2-1-1. Port 选项卡（展开 External Port），如图



图：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。

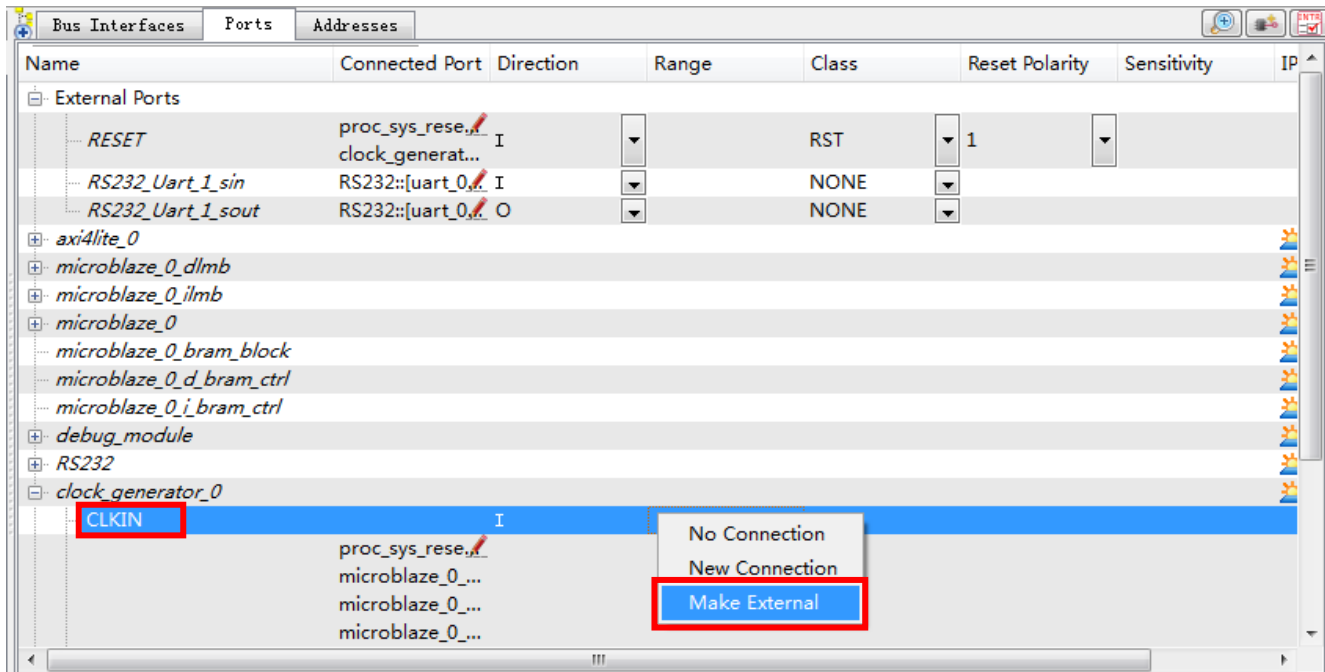


图：源文件添加后的 ISE 用户界面



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**



图：Clock\_generator\_0 中的 CLKIN

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

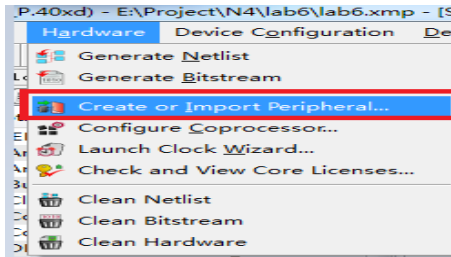
Name	Connected Port	Direction	Range	Class	Reset Polarity	Se
<b>External Ports</b>						
RESET	proc_sys_res...	I		RST	1	
RS232_Uart_1_sin	RS232::[uart_0...	I		NONE		
RS232_Uart_1_sout	RS232::[uart_0...	O		NONE		
clock_generator_0_CLKIN_pin	clock_generat...	I		CLK		

图：修改后的 External PORT

## 第三步 创建并添加用户自定义 IP 核（AD 模块）

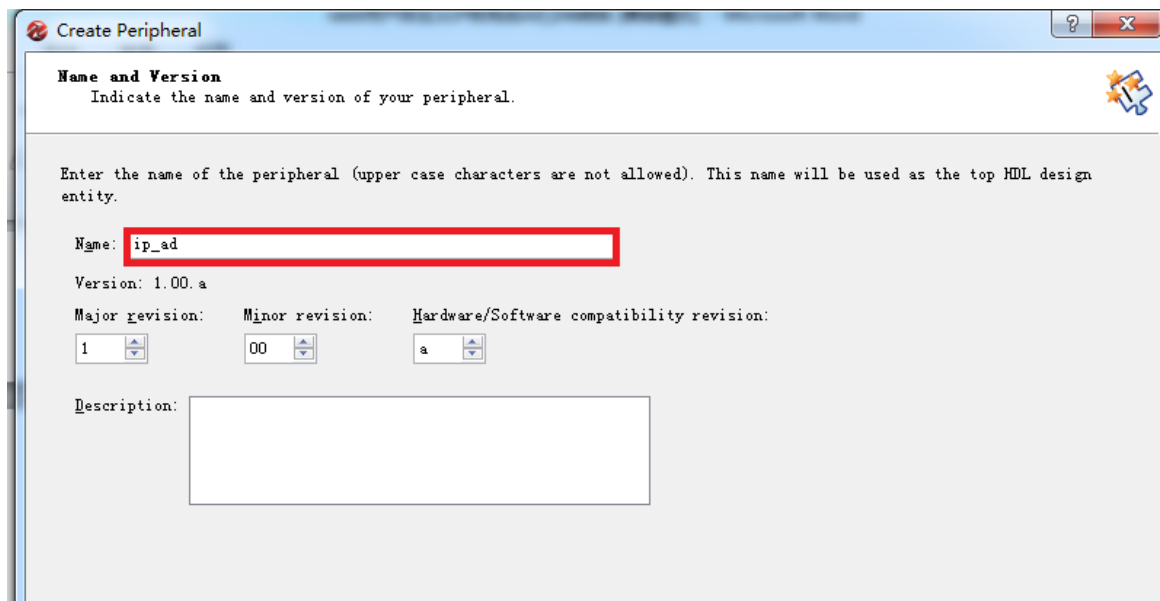
### 3-1. 创建用户自定义 IP 核

#### 3-1-1. 选择 Hardware>Create or Import Peripheral



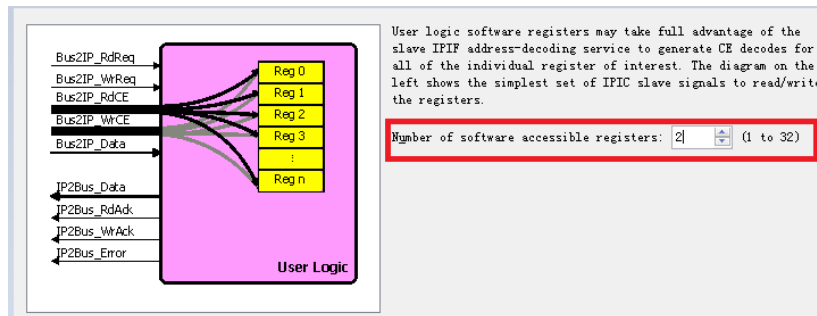
#### 3-1-2. 在 Peripheral Flow/Repository or Project 选项栏中全部保持默认，直接 next

#### 3-1-3. 在 Name and Version 选项栏中输入自定义 IP 核的名字，这里取名 ip\_ad。点击 next



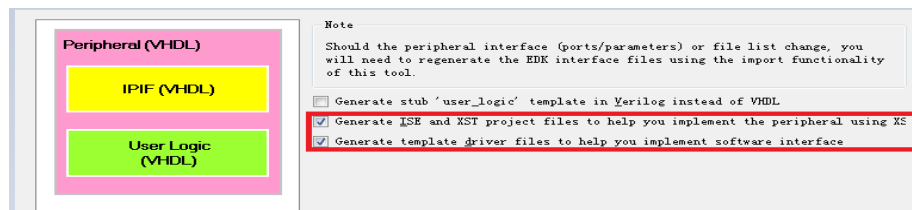
#### 3-1-4. Bus Interface/IPIF Services 选项栏保持默认选项。

### 3-1-5.User S/W Register 选项栏中，选择寄存器数量。修改为 2

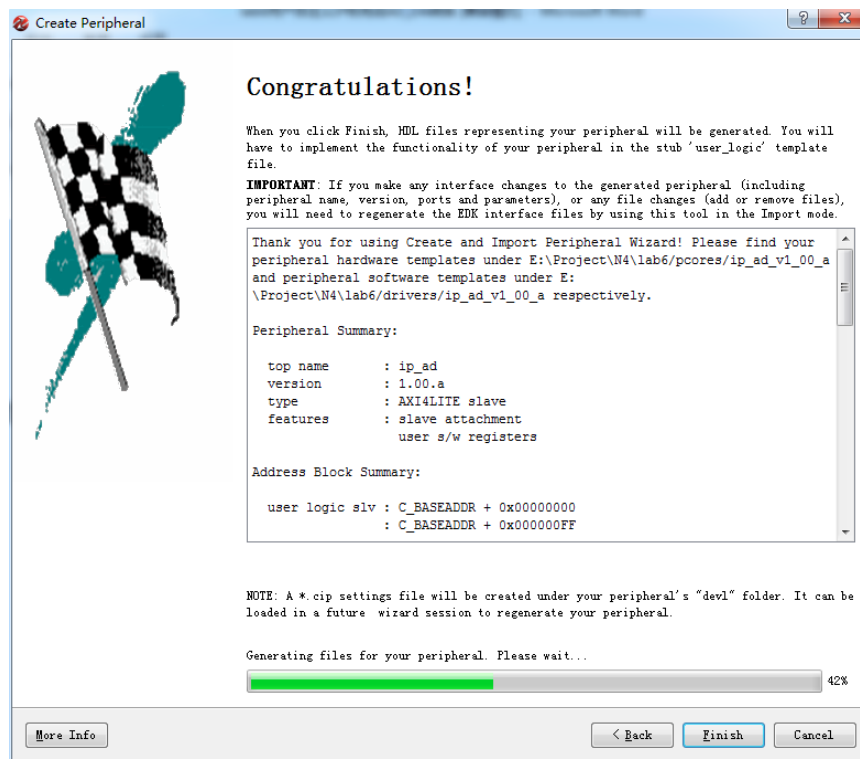


### 3-1-6.IP Interconnect/Peripheral Simulation Support 选项栏保持默认

### 3-1-7.Peripheral Implementation Support 选项栏勾选最后两栏，点击 next

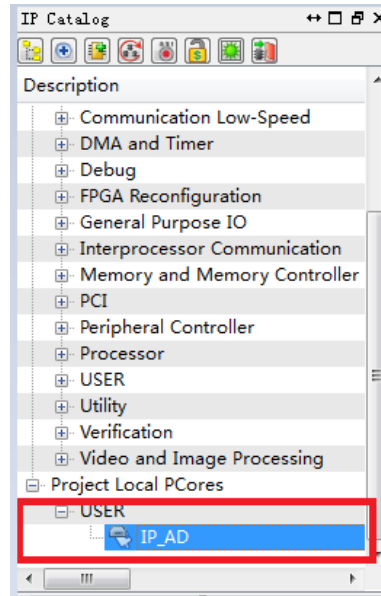


### 3-1-8.用户自定义 IP 核定义完毕，点击 finish。在这个面板里，可以看到一些自定义 IP 核的基本信息

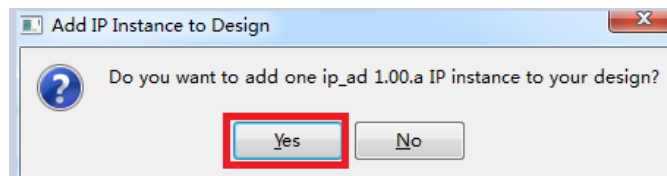


## 3-2. 添加自定义 IP 核并修改其内部结构

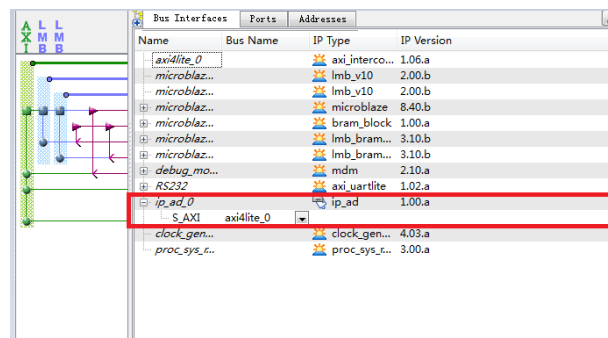
### 3-2-1. 在 IP Catalog 栏中 双击 Project Local PCores-User-IP\_AD



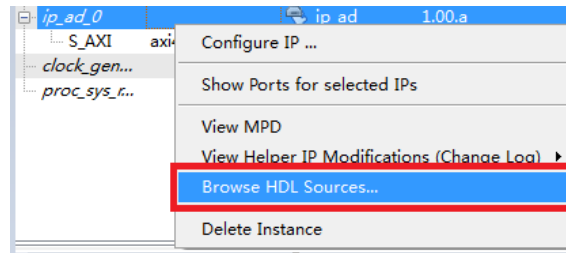
### 3-2-2. 选择 yes，在之后的两个对话框中保持默认



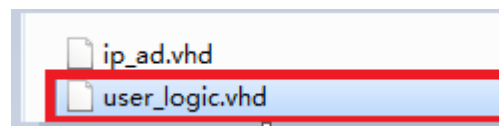
### 3-2-3. 可以看到添加进的 IP 核和总线之间的互联



### 3-2-4.右键 IP 核选择 Browse HDL Sources



### 3-2-5.选择 user\_logic.vhd



### 3-2-6.在 user\_logic 实体中添加如下端口

```
);  
port  
(  
    -- ADD USER PORTS BELOW THIS LINE -----  
    --USER ports added here  
    --General usage  
    RST      : in std_logic;  
  
    --Fmod interface signals  
    SDATA1   : in std_logic;  
    SDATA2   : in std_logic;  
    SCLK     : out std_logic;  
    nCS      : out std_logic;  
  
    --User interface signals  
    START    : in std_logic;  
    DONE     : out std_logic;  
    -- ADD USER PORTS ABOVE THIS LINE -----  
  
    -- DO NOT EDIT BELOW THIS LINE -----  
    -- Bus protocol ports, do not add to or delete  
    Bus2IP_Clk      : in std_logic;  
    Bus2IP_Resetn   : in std_logic;  
    Bus2IP_Data     : in std_logic_vector(C_SLV_DWIDTH-1 downto 0);  
    Bus2IP_BE       : in std_logic_vector(C_SLV_DWIDTH/8-1 downto 0);  
    Bus2IP_RdCE     : in std_logic_vector(C_NUM_REG-1 downto 0);  
);
```

### 3-2-7.在 user\_logic 的 architecture 中添加如下声明

```
architecture IMP of user_logic is
    --USER signal declarations added here as needed for user logic
    type states is (Idle,
                    ShiftIn,
                    SyncData);
    signal current_state : states;
    signal next_state    : states;

    signal temp1          : std_logic_vector(15 downto 0);
    signal temp2          : std_logic_vector(15 downto 0);
    signal clk_div         : std_logic;
    signal clk_counter    : std_logic_vector(2 downto 0);
    signal shiftCounter   : std_logic_vector(3 downto 0) := x"0";
    signal enShiftCounter : std_logic;
    signal enParallelLoad : std_logic;

    -----
    -- Signals for user logic slave model s/w accessible register example
    -----
    signal slv_reg0          : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
    signal slv_reg1          : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
```

### 3-2-8.在 user logic implementation 中添加

```
    signal slv_read_ack      : std_logic;
    signal slv_write_ack     : std_logic;

begin

    --USER logic implementation added here

    clock_divide : process(rst, Bus2IP_Clk)
    begin
        if rst = '1' then
            clk_counter <= "000";
        elsif (Bus2IP_Clk = '1' and Bus2IP_Clk'event) then
            clk_counter <= clk_counter + '1';
        end if;
    end process;

    clk_div <= clk_counter(2);
    SCLK <= not clk_counter(2);
```

```
counter : process (clk_div, enParallelLoad, enShiftCounter)
begin
    if (clk_div = '1' and clk_div'event) then

        if (enShiftCounter = '1') then
            temp1 <= temp1(14 downto 0) & SDATA1;
            temp2 <= temp2(14 downto 0) & SDATA2;
            shiftCounter <= shiftCounter + '1';
        elsif (enParallelLoad = '1') then
            shiftCounter <= "0000";
            slv_reg0(11 downto 0) <= temp1(11 downto 0);
            slv_reg1(11 downto 0) <= temp2(11 downto 0);
        end if;
    end if;
end process;

SYNC_PROC: process (clk_div, rst)
begin
    if (clk_div'event and clk_div = '1') then
        if (rst = '1') then
            current_state <= Idle;
        else
            current_state <= next_state;
        end if;
    end if;
end process;

OUTPUT_DECODE: process (current_state)
begin
    if current_state = Idle then
        enShiftCounter <= '0';
        DONE <= '1';
        nCS <= '1';
        enParallelLoad <= '0';
    elsif current_state = ShiftIn then
        enShiftCounter <= '1';
        DONE <= '0';
        nCS <= '0';
        enParallelLoad <= '0';
    else --if current_state = SyncData then
        enShiftCounter <= '0';
        DONE <= '0';
        nCS <= '1';
        enParallelLoad <= '1';
    end if;
end process;
```

```
NEXT_STATE_DECODE: process (current_state, START, shiftCounter)
begin
    next_state <= current_state; -- default is to stay in current state
    case (current_state) is
        when Idle =>
            if START = '1' then
                next_state <= ShiftIn;
            end if;
        when ShiftIn =>
            if shiftCounter = x"F" then
                next_state <= SyncData;
            end if;
        when SyncData =>
            if START = '0' then
                next_state <= Idle;
            end if;
        when others =>
            next_state <= Idle;
    end case;
end process;
```

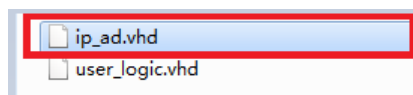
### 3-2-9. 将如下代码注释并保存

```
-- implement slave model software accessible register(s)
SLAVE_REG_WRITE_PROC : process( Bus2IP_Clk ) is
begin

    if Bus2IP_Clk'event and Bus2IP_Clk = '1' then
        if Bus2IP_Resetn = '0' then
            slv_reg0 <= (others => '0');
            slv_reg1 <= (others => '0');
        else
            case slv_reg_write_sel is
                when "10" =>
                    for byte_index in 0 to (C_SLV_DWIDTH/8)-1 loop
                        if ( Bus2IP_BE(byte_index) = '1' ) then
                            slv_reg0(byte_index*8+7 downto byte_index*8) <= Bus2IP_Data(byte_index*8+7 downto byte_index*8);
                        end if;
                    end loop;
                when "01" =>
                    for byte_index in 0 to (C_SLV_DWIDTH/8)-1 loop
                        if ( Bus2IP_BE(byte_index) = '1' ) then
                            slv_reg1(byte_index*8+7 downto byte_index*8) <= Bus2IP_Data(byte_index*8+7 downto byte_index*8);
                        end if;
                    end loop;
                when others => null;
            end case;
        end if;
    end if;
```

注：如上所有代码通过状态机将 AD 转换器采集的结果，一个一个接收并存储到软件寄存器 (slv\_reg0,slv\_reg1) 上。

### 3-2-10.User\_logic 修改完毕，重新选择 browse HDL Sources，选择 ip\_ad.vhd





### 3-2-11.在 ip\_ad 的实体中添加如下端口

```
C_SLV_AWIDTH      : integer      := 32;
C_SLV_DWIDTH      : integer      := 32;
-- DO NOT EDIT ABOVE THIS LINE -----
);
port
(
  -- ADD USER PORTS BELOW THIS LINE -----
  --USER ports added here
  RST      : in std_logic;

  --Pmod interface signals
  SDATA1   : in std_logic;
  SDATA2   : in std_logic;
  SCLK     : out std_logic;
  nCS      : out std_logic;

  --User interface signals
  START    : in std_logic;
  DONE     : out std_logic;
  -- ADD USER PORTS ABOVE THIS LINE -----

  -- DO NOT EDIT BELOW THIS LINE -----
  -- Bus protocol ports, do not add to or delete
  S_AXI_ACLK : in std_logic;
  S_AXI_ARESETN : in std_logic;
```

### 3-2-12.在 ip\_ad 的 architecture 中 user\_logic 的例化中添加如下代码，并保存

```
USER_LOGIC_I : entity ip_ad_v1_00_a.user_logic
generic map
(
  -- MAP USER GENERICS BELOW THIS LINE -----
  --USER generics mapped here
  -- MAP USER GENERICS ABOVE THIS LINE -----

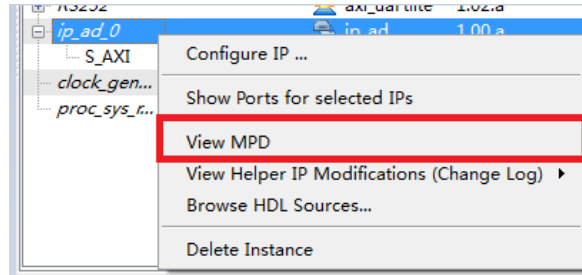
  C_NUM_REG      => USER_NUM_REG,
  C_SLV_DWIDTH   => USER_SLV_DWIDTH
)
port map
(
  -- MAP USER PORTS BELOW THIS LINE -----
  --USER ports mapped here
  RST => RST,
  SDATA1 => SDATA1,
  SDATA2 => SDATA2,
  SCLK => SCLK,
  nCS => nCS,

  --User interface signals
  START => START,
  DONE => DONE,
  -- MAP USER PORTS ABOVE THIS LINE -----

  Bus2IP_Clk      => ipif_Bus2IP_Clk,
  Bus2IP_Resetn   => ipif_Bus2IP_Resetn,
```

### 3-3. 修改 IP 核的 MPD 文件

#### 3-3-1. 右键 IP 核选择 View MPD

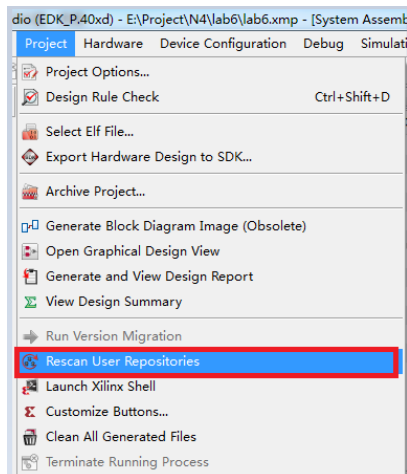


#### 3-3-2. 在 PORT 下添加如下代码并保存

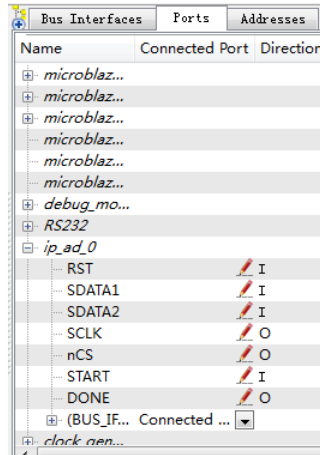
```
## Ports
PORT RST="", DIR=I
PORT SDATA1="", DIR=I
PORT SDATA2="", DIR=I
PORT SCLK="", DIR=O
PORT nCS="", DIR=O
PORT START="", DIR=I
PORT DONE="", DIR=O

PORT S_AXI_ACLK = "", DIR = I, SIGIS = CLK, BUS = S_AXI
PORT S_AXI_ARESETN = ARESETN, DIR = I, SIGIS = RST, BUS = S_AXI
```

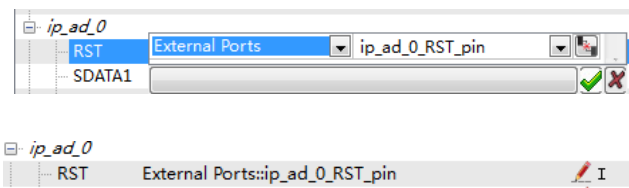
#### 3-3-3. 选择菜单栏中 Project-Rescan User Repositories



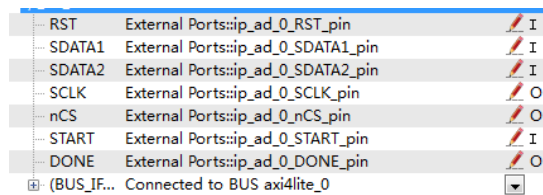
3-3-4.工程会重新导入，导入完毕后，可以在 **port** 选项卡中看到 **ip\_ad\_0** 内添加了如下端口



3-3-5.单击 **ip\_ad\_0-rst** 的 **Connect Port** 栏，选择 **External Ports**



3-3-6.将 **SDATA1**，**SDATA2**，**SCLK**，**nCS**，**START**，**DONE** 按 3-3-5 同样操作

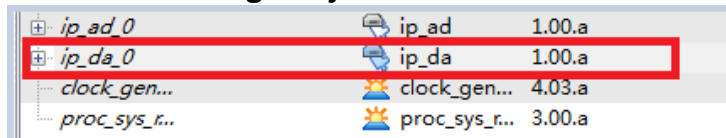


## 第四步 创建并添加用户自定义 IP 核（DA 模块）

4-1. 创建 IP 核步骤同 3-1，名字改为 ip\_da

4-2. 添加自定义 IP 核并修改其内部结构

4-1-1. 双击 IP Catalog-Project Local PCores-USER-IP-DA，对话框全保持默认



4-1-2. 右键 Ip\_da\_0 单击 browse hdl sources，选择 usr\_logic.vhd

4-1-3. 在 use\_logic 的实体中加如下代码

```
C_NUM_REG          : integer          := 2;
C_SLV_DWIDTH       : integer          := 32;
-- DO NOT EDIT ABOVE THIS LINE -----
);
port
(
  -- ADD USER PORTS BELOW THIS LINE -----
  RST      : in std_logic;

  --Fmod interface signals
  D1       : out std_logic;
  D2       : out std_logic;
  CLK_OUT  : out std_logic;
  nSYNC    : out std_logic;

  --User interface signals
  START    : in std_logic;
  DONE     : out std_logic;

  -- ADD USER PORTS ABOVE THIS LINE -----

  -- DO NOT EDIT BELOW THIS LINE -----
  -- Bus protocol ports, do not add to or delete
  Bus2IP Clk : in std_logic;
```

#### 4-1-4.在 user\_logic 的 architecture 中添加如下声明

```
architecture IMP of user_logic is
    --USER signal declarations added here, as needed for user logic
    constant control      : std_logic_vector(3 downto 0) := "0000";
    type states is (Idle,
                    ShiftOut,
                    SyncData);
    signal current_state : states;
    signal next_state    : states;

    signal temp1          : std_logic_vector(15 downto 0);
    signal temp2          : std_logic_vector(15 downto 0);
    signal clk_div         : std_logic;
    signal clk_counter    : std_logic_vector(27 downto 0);
    signal shiftCounter   : std_logic_vector(3 downto 0);
    signal enShiftCounter : std_logic;
    signal enParallelLoad : std_logic;

    -- Signals for user logic slave model s/w accessible register example
    -----
    signal slv_reg0          : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
    signal slv_reg1          : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
    -- Bus protocol ports, do not add to or delete
    Bus2IP_Clk              : in std_logic;
end architecture
```

#### 4-1-5.添加 USER logic implementation, 保存

```
signal slv_reg0          : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal slv_reg1          : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal slv_reg_write_sel : std_logic_vector(1 downto 0);
signal slv_reg_read_sel  : std_logic_vector(1 downto 0);
signal slv_ip2bus_data   : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal slv_read_ack      : std_logic;
signal slv_write_ack     : std_logic;

begin

    --USER logic implementation added here
    clock_divide : process(rst, Bus2IP_Clk)
    begin
        if rst = '1' then
            clk_counter <= "00000000000000000000000000000000";
        elsif (Bus2IP_Clk = '1' and Bus2IP_Clk'event) then
            clk_counter <= clk_counter + '1';
        end if;
    end process;

    clk_div <= clk_counter(1);
    clk_out <= clk_counter(1);

    counter : process(clk_div, enParallelLoad, enShiftCounter)
    begin
        if (clk_div = '1' and clk_div'event) then
            if enParallelLoad = '1' then
                shiftCounter <= "0000";
                temp1 <= control & slv_reg0(11 downto 0);
                temp2 <= control & slv_reg1(11 downto 0);
            elsif (enShiftCounter = '1') then
                temp1 <= temp1(14 downto 0) & temp1(15);
                temp2 <= temp2(14 downto 0) & temp2(15);
                shiftCounter <= shiftCounter + '1';
            end if;
        end if;
    end process;

    D1 <= temp1(15);
    D2 <= temp2(15);
end
```

```
SYNC_PROC: process (clk_div, rst)
begin
    if (clk_div'event and clk_div = '1') then
        if (rst = '1') then
            current_state <= Idle;
        else
            current_state <= next_state;
        end if;
    end if;
end process;

OUTPUT_DECODE: process (current_state)
begin
    if current_state = Idle then
        enShiftCounter <='0';
        DONE <='1';
        nSYNC <='1';
        enParallelLoad <= '1';
    elsif current_state = ShiftOut then
        enShiftCounter <='1';
        DONE <='0';
        nSYNC <='0';
        enParallelLoad <= '0';
    else --if current_state = SyncData then
        enShiftCounter <='0';
        DONE <='0';
        nSYNC <='1';
        enParallelLoad <= '0';
    end if;
end process;

NEXT_STATE_DECODE: process (current_state, START, shiftCounter)
begin
    next_state <= current_state; --default is to stay in current state
    case (current_state) is
        when Idle =>
            if START = '1' then
                next_state <= ShiftOut;
            end if;
        when ShiftOut =>
            if shiftCounter = x"F" then
                next_state <= SyncData;
            end if;
        when SyncData =>
            if START = '0' then
                next_state <= Idle;
            end if;
        when others =>
            next_state <= Idle;
    end case;
end process;
```

注：如上所有代码通过状态机将数字信号一个一个推送到 DA 转换器上，DA 转换器会自动接收这些信号并转换成模拟信号

#### 4-1-6.选择 ip\_da\_0 的 ip\_da.vhd

#### 4-1-7.在 ip\_da 实体中添加

```
);  
port  
(  
    -- ADD USER PORTS BELOW THIS LINE -----  
    --USER ports added here  
    RST      : in std_logic;  
  
    --Pmod interface signals  
    D1       : out std_logic;  
    D2       : out std_logic;  
    CLK_OUT  : out std_logic;  
    nSYNC    : out std_logic;  
  
    --User interface signals  
    START    : in std_logic;  
    DONE     : out std_logic;  
  
    -- ADD USER PORTS ABOVE THIS LINE -----  
  
    -- DO NOT EDIT BELOW THIS LINE -----  
    -- Bus protocol ports, do not add to or delete  
    S_AXI_ACLK      : in std_logic;  
    S_AXI_ARESETN   : in std_logic;  
    S_AXI_AWADDR    : in std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);  
    S_AXI_AWVALID   : in std_logic;
```

#### 4-1-8.在例化 user\_logic 中添加端口映射，保存

```
USER_LOGIC_I : entity ip_da_v1_00_a.user_logic  
    generic map  
    (  
        -- MAP USER GENERICS BELOW THIS LINE -----  
        --USER generics mapped here  
        -- MAP USER GENERICS ABOVE THIS LINE -----  
  
        C_NUM_REG      => USER_NUM_REG,  
        C_SLV_DWIDTH   => USER_SLV_DWIDTH  
    )  
    port map  
    (  
        -- MAP USER PORTS BELOW THIS LINE -----  
        --USER ports mapped here  
        RST      => RST,  
  
        --Pmod interface signals  
        D1       => D1,  
        D2       => D2,  
        CLK_OUT  => CLK_OUT,  
        nSYNC    => nSYNC,  
  
        --User interface signals  
        START    => START,  
        DONE     => DONE,  
  
        -- MAP USER PORTS ABOVE THIS LINE -----  
  
        Bus2IP Clk      => ipif Bus2IP Clk,
```

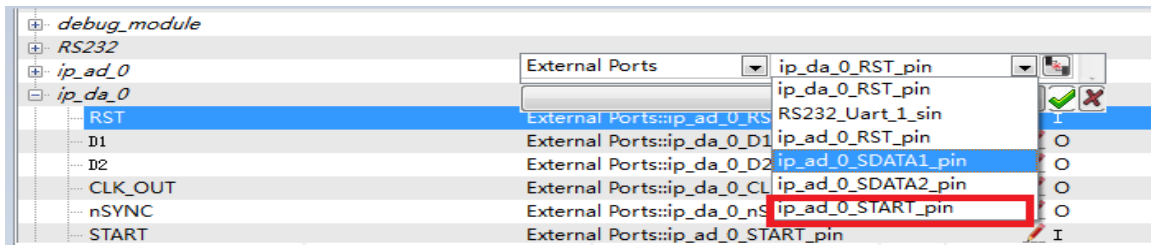
#### 4-1-9.修改 ip\_da 的 mpd 文件

```
## Ports
PORT RST = "", DIR=I
PORT D1="", DIR=O
PORT D2="", DIR=O
PORT CLK_OUT="", DIR=O
PORT nSYNC="", DIR=O
PORT START="", DIR=I
PORT DONE = "", DIR=O

PORT S_AXI_ACLK = "", DIR = I, SIGIS = CLK, BUS = S_AXI
PORT S_AXI_ARESETN = ARESETN, DIR = I, SIGIS = RST, BUS = S_AXI
PORT S_AXI_AWADDR = AWADDR, DIR = I, VEC = [(C S_AXI_ADDR_WIDTH-1):0], ENDIAN = LITTLE, BUS = S_AXI
```

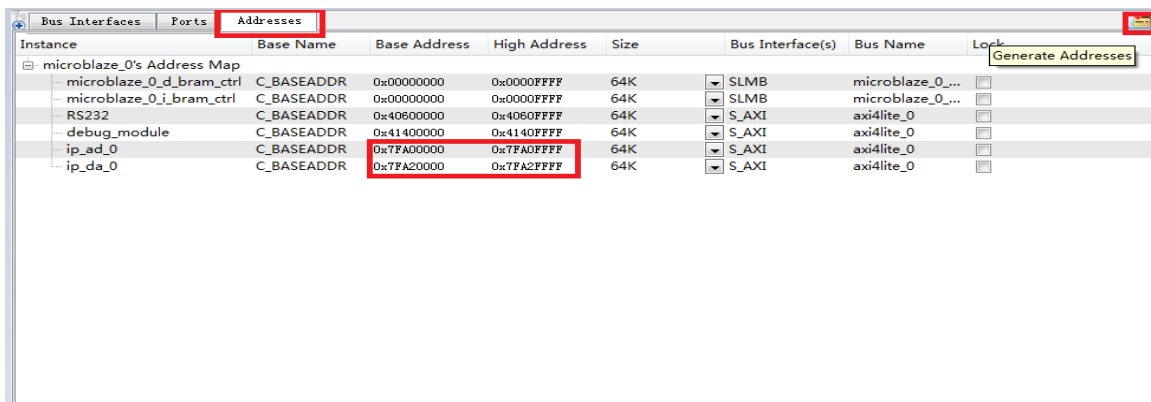
#### 4-1-10.选择菜单栏中 Project-Rescan User Repositories

#### 4-1-11.在 Ports 为 ip\_da\_0 的端口添加 External Ports, RST 和 START 选择 ad 的 RST, STARTPORTS



ip_da_0	RST	External Ports::ip_ad_0_RST_pin	I
	D1	External Ports::ip_da_0_D1_pin	O
	D2	External Ports::ip_da_0_D2_pin	O
	CLK_OUT	External Ports::ip_da_0_CLK_OUT_pin	O
	nSYNC	External Ports::ip_da_0_nSYNC_pin	O
	START	External Ports::ip_ad_0_START_pin	I
	DONE	External Ports::ip_da_0_DONE_pin	O

#### 4-1-12.查看 Address 项目栏如果 XPS 未自动给添加的自定义 IP 核分配地址, 则需要手动分配, 点击 Generate Address



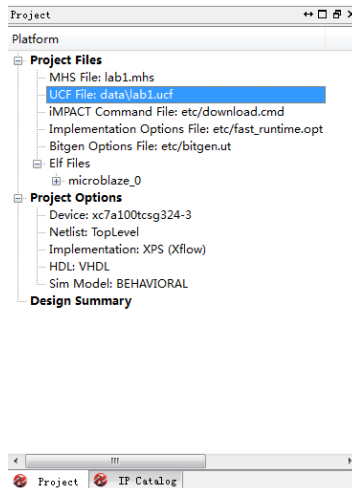
Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)	Bus Name	Lock
microblaze_0's Address Map							
microblaze_0_d_bram_ctrl	C_BASEADDR	0x00000000	0x0000FFFF	64K	SLMB	microblaze_0...	
microblaze_0_i_bram_ctrl	C_BASEADDR	0x00000000	0x0000FFFF	64K	SLMB	microblaze_0...	
RS232	C_BASEADDR	0x40600000	0x4060FFFF	64K	S_AXI	axi4lite_0	
debug_module	C_BASEADDR	0x41400000	0x4140FFFF	64K	S_AXI	axi4lite_0	
ip_ad_0	C_BASEADDR	0x7FA00000	0x7FA0FFFF	64K	S_AXI	axi4lite_0	
ip_da_0	C_BASEADDR	0x7FA20000	0x7FA2FFFF	64K	S_AXI	axi4lite_0	



## 第五步 添加用户约束文件

### 5-1. 打开初始 UCF 文件，根据需求进行修改

#### 5-1-1. 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



图：UCF 文件的位置

#### 5-1-2. 这里我们手动输入 LOC（引脚位置）约束代码，如图。点击保存。

```
## 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%;

NET "ip_ad_0_DONE_pin" LOC = "R2" | IOSTANDARD = "LVCMOS33";
NET "ip_da_0_DONE_pin" LOC = "P2" | IOSTANDARD = "LVCMOS33";

## Buttons
NET "ip_ad_0_RST_pin" LOC = "T16" | IOSTANDARD = "LVCMOS33";
NET "ip_ad_0_START_pin" LOC = "R10" | IOSTANDARD = "LVCMOS33";
NET "RESET" LOC = "E16" | IOSTANDARD = "LVCMOS33";

## Pmod Header JA
NET "ip_ad_0_nCS_pin" LOC = "B13" | IOSTANDARD = "LVCMOS33";
NET "ip_ad_0_SDATA1_pin" LOC = "F14" | IOSTANDARD = "LVCMOS33";
NET "ip_ad_0_SDATA2_pin" LOC = "D17" | IOSTANDARD = "LVCMOS33";
NET "ip_ad_0_SCLK_pin" LOC = "E17" | IOSTANDARD = "LVCMOS33";

## Pmod Header JB
NET "ip_da_0_nSYNC_pin" LOC = "G14" | IOSTANDARD = "LVCMOS33";
NET "ip_da_0_D1_pin" LOC = "P15" | IOSTANDARD = "LVCMOS33";
NET "ip_da_0_D2_pin" LOC = "V11" | IOSTANDARD = "LVCMOS33";
NET "ip_da_0_CLK_OUT_pin" LOC = "V15" | IOSTANDARD = "LVCMOS33";

## USB-RS232 Interface
NET "RS232_Uart_1_sin" LOC = "C4" | IOSTANDARD = "LVCMOS33";
NET "RS232_Uart_1_sout" LOC = "D4" | IOSTANDARD = "LVCMOS33";
```

图：UCF 文件

#### 5-1-3. 修改 address 选项卡中 microblaze 的 Size 为 64K

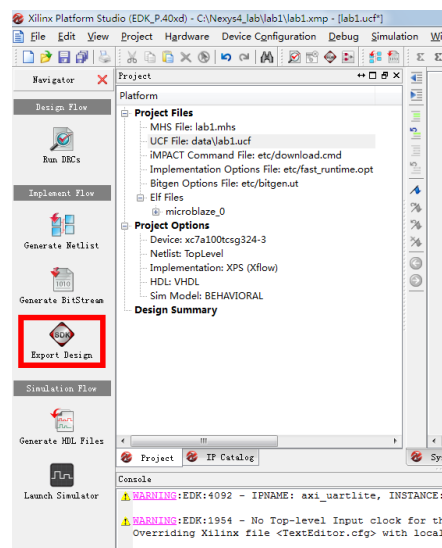
microblaze_0's Address Map					
microblaze_0_d_bram_ctrl	C_BASEADDR	0x00000000	0x00001FFF	8K	SLMB

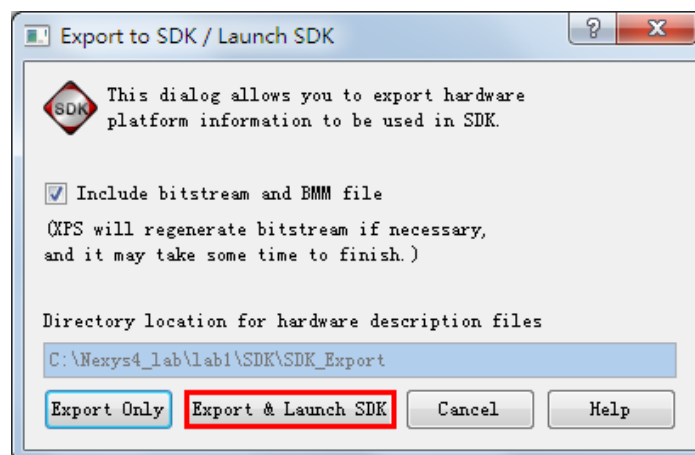
microblaze_0's Address Map					
microblaze_0_d_bram_ctrl	C_BASEADDR	0x00000000	0x0000FFFF	64K	SLMB
microblaze_0_i_bram_ctrl	C_BASEADDR	0x00000000	0x0000FFFF	64K	SLMB

#### 5-1-4. 保存之后将工程导入到 SDK

在页面左边，点击 **Export Design**，如图。



图：export design

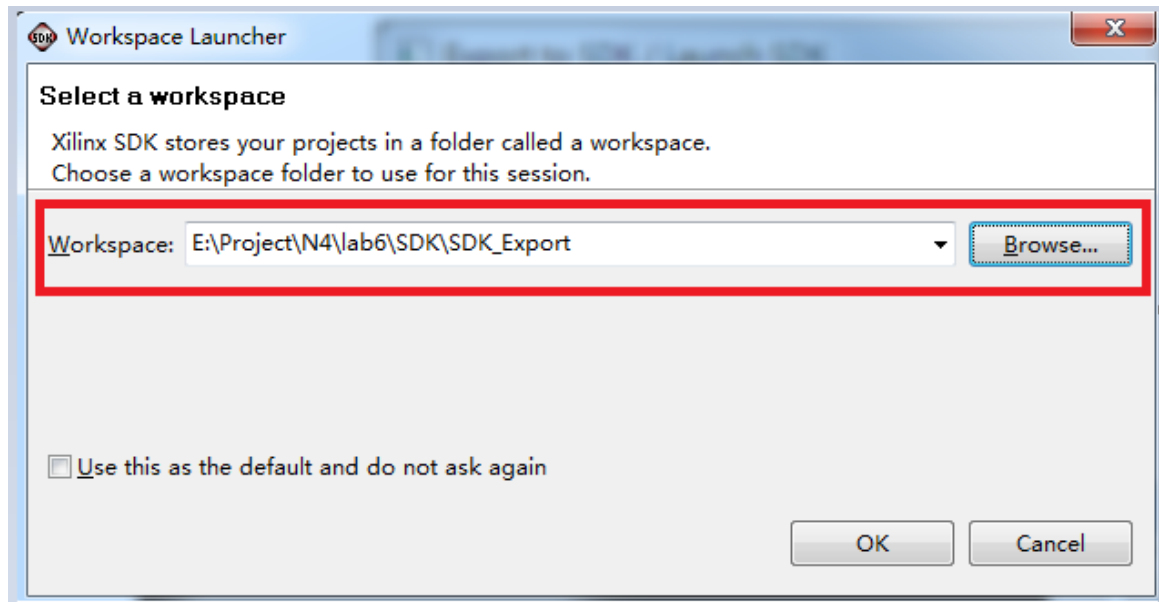


图：在弹出的对话框中选择 **Export & launch sdk**

### 3-1-4. 选择 SDK 导入路径

选择工程路径，注意要具体到..\sdk\sdk\_export，如图

点击 **ok**



图：导入 sdk 的工作空间选择

## 第六步 添加 app

### 6-1. 添加软件应用。

#### 6-1-1. 找到工程路径下\*\*\*\*\*\lab6\drivers\ip\_ad\_v1\_00\_a\src\ip\_ad\_selftest.c 修改

```
#include "ip_ad.h"  
#include "stdio.h"  
#include "xio.h"  
#include "xparameters.h"
```

```
#include "ip_ad.h"  
#include "stdio.h"  
#include "xil_io.h"  
#include "xparameters.h"
```

添加一个新的宏

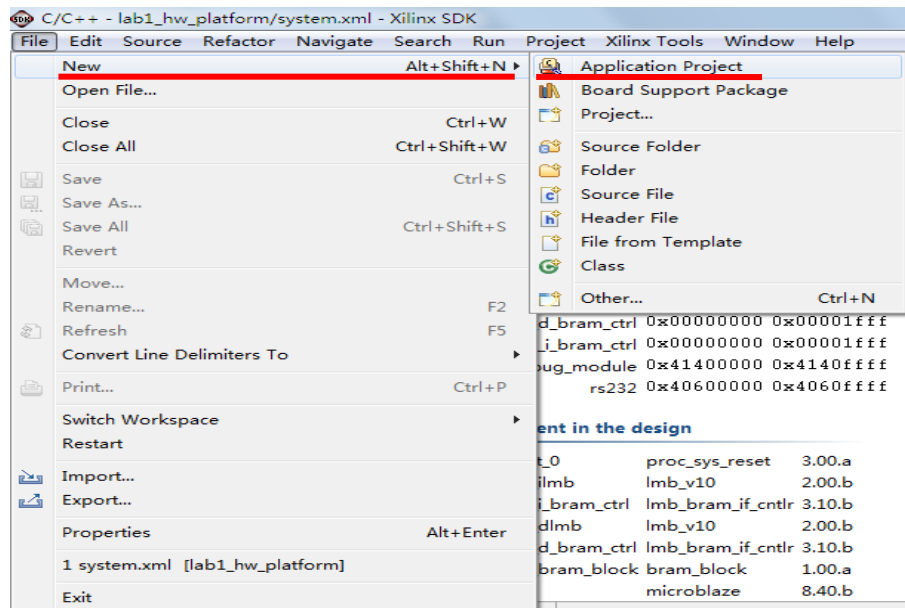
```
#define READ_WRITE_MUL_FACTOR 0x10  
#define IP_AD_USER_NUM_REG 2
```

#### 6-1-2. 找到工程路径下\*\*\*\*\*\lab6\drivers\ip\_da\_v1\_00\_a\src\ip\_da\_selftest.c 修改

```
#include "ip_da.h"  
#include "stdio.h"  
#include "xil_io.h"  
#include "xparameters.h"
```

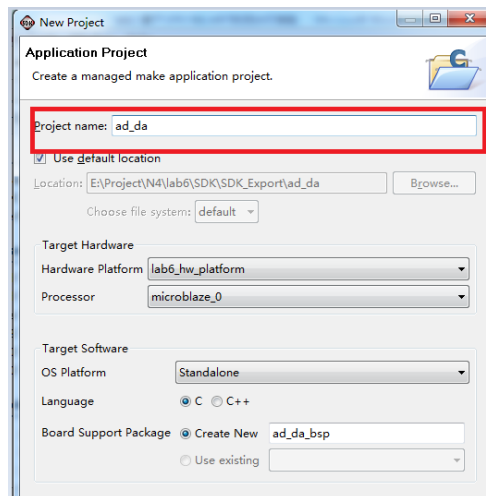
```
#define READ_WRITE_MUL_FACTOR 0x10  
#define IP_DA_USER_NUM_REG 2
```

6-1-3. 在 SDK 的用户界面中，选择 file—new—application project，如图。



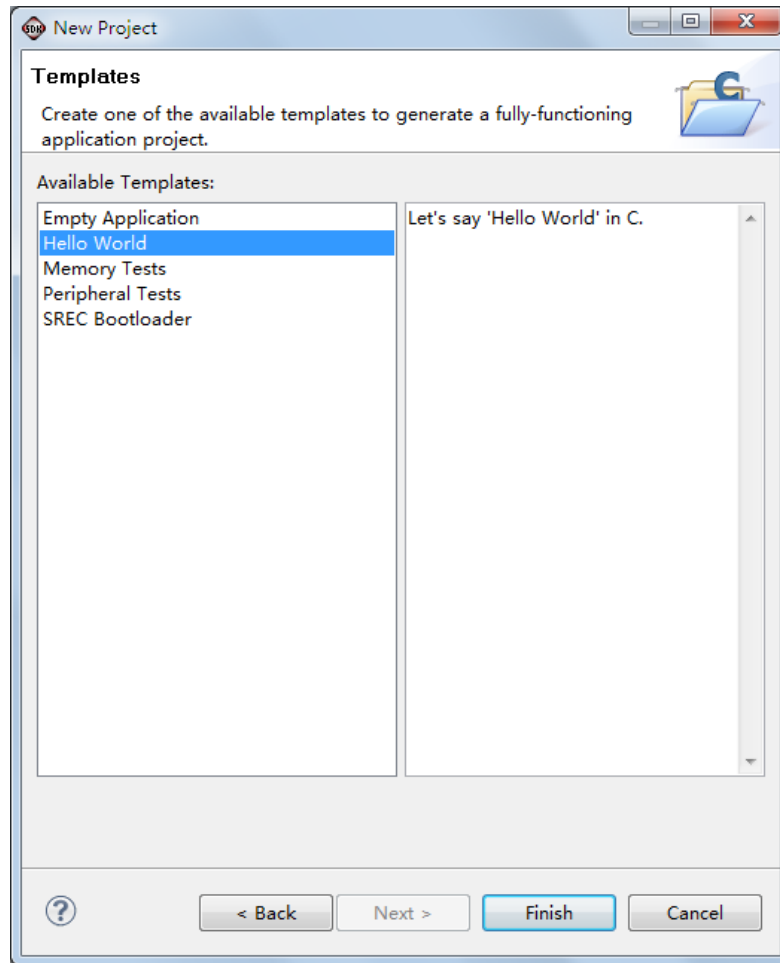
图：新建软件应用

6-1-4. 输入工程的名称，这里使用 ad\_da, 同样不要包含空格和中文，点击 next



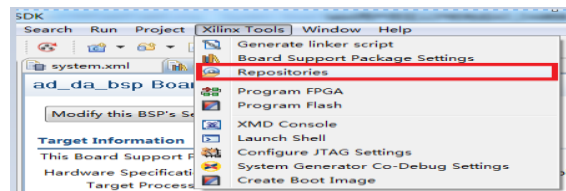
图：新建工程---命名

6-1-5. 在下一步弹出的对话框中选择 **hello world**，然后点击 **finish**

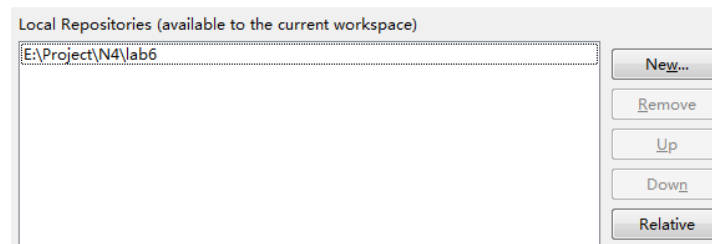
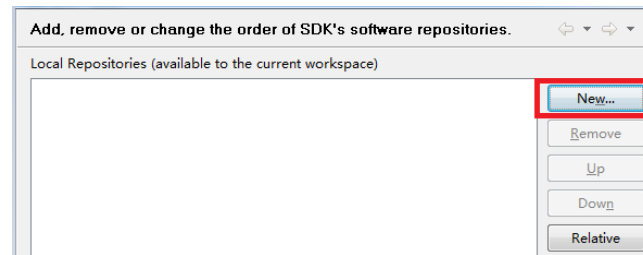


图：选择 Helloworld 示例代码

6-1-6. 选择菜单栏 **Xilinx Tools-Respositories**

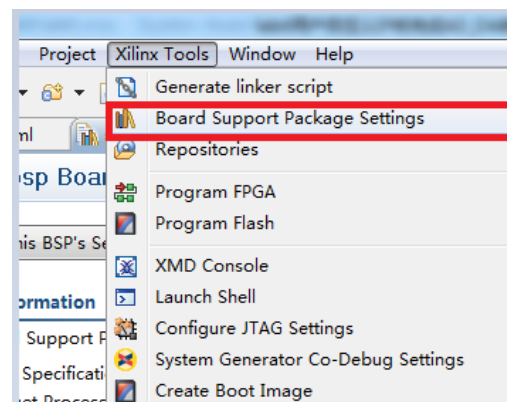


### 6-1-7. 在 local Repositories 内添加工程路径

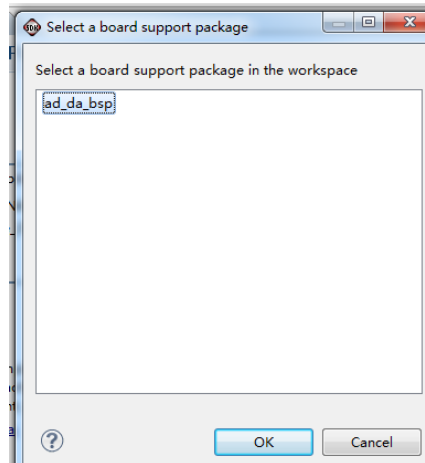


点击 OK.

### 6-1-8. 选择菜单栏 Xilinx Tools-Board Support Package Settings



## 选择 ad\_da\_bsp



## 选择 drivers



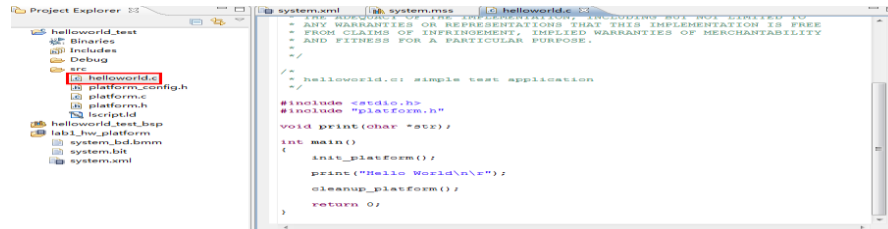
在 ip\_ad\_0 和 ip\_da\_0 的 Driver 一栏，将 generic 改成 ip\_ad 和 ip\_da,点击 OK

Component	Component Type	Driver	Dri...	
microblaze_0	microblaze	cpu	1.1...	
debug_module	mdm	uartlite	2.0...	
ip_ad_0	ip_ad	ip_ad	1.0...	
ip_da_0	ip_da	ip_da	1.0...	
microblaze_0_d_bram_ctrl	lmb_bram_if_ctrlr	bram	3.0...	
microblaze_0_i_bram_ctrl	lmb_bram_if_ctrlr	bram	3.0...	
rs232	axi_uartlite	uartlite	2.0...	



6-1-9. 添加完毕以后左侧双击源文件，添加代码：

可以根据自己的需要进行一些修改，修改后保存



```
#include <stdio.h>
#include "platform.h"
#include "ip_da.h"
#include "ip_ad.h"
#include "xparameters.h"
#include "xil_io.h"
#include "xuartlite.h"
#include "xuartlite_1.h"

int main()
{
    init_platform();

    char a;
    while(1)
    {
        a=inbyte();
        Test_AD();
        Test_DA();
    }

    cleanup_platform();

    return 0;
}

void Test_AD()
{
    u32 R;
    int V;
    R=IP_AD_mReadReg(XPAR_IP_AD_0_BASEADDR,IP_AD_SLV_REG0_OFFSET);
    V=3.3*((double)R/(double)4095)*100;
    xil_printf("AD:%d\r\n",V);
}

void Test_DA()
{
    u32 R;
    double V=2.7; //此处修改数模转换的电压值 0-3.3
    R=(u32)((V/3.3)*4095);
    IP_DA_mWriteReg(XPAR_IP_DA_0_BASEADDR,IP_DA_SLV_REG0_OFFSET,R);
    xil_printf("DA:%d\r\n",R);
}
```

## 第七步 上板验证

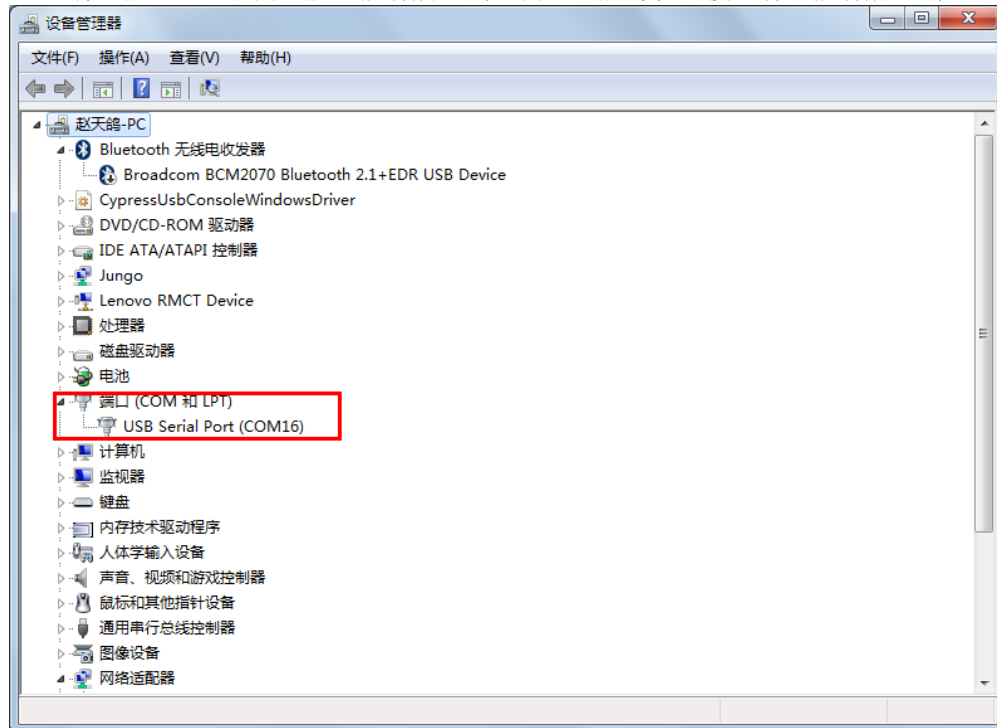
7-1. 链接 AD，DA 转换器到 N4 的 JA，JB Pmod 上，将 AD 的输出和 DA 输入链接

7-2. 将 Nexys4 与 Pc 的 USB 接口连接

7-3. 查看端口号：

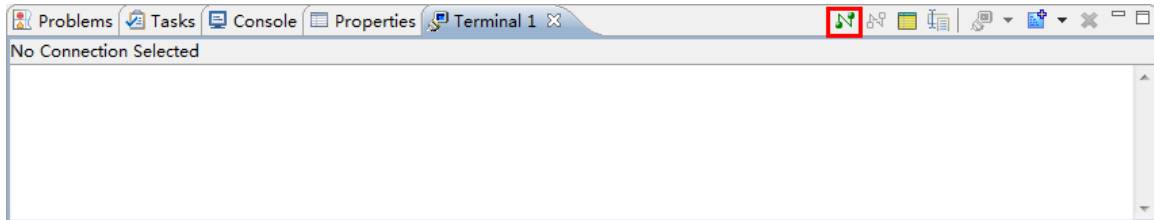
右键“我的电脑”——“属性”——在页面左侧选择“设备管理器”

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）

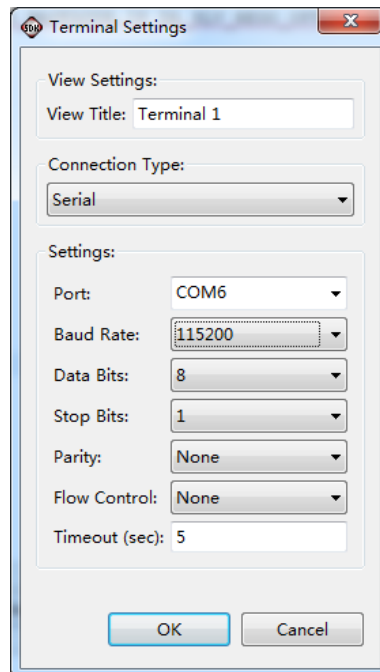


## 7-4. 在 SDK 中打开串口：

7-4-1. 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。



7-4-2. 按照端口号和 XPS 中的波特率（**baud rate**）进行如下设置：

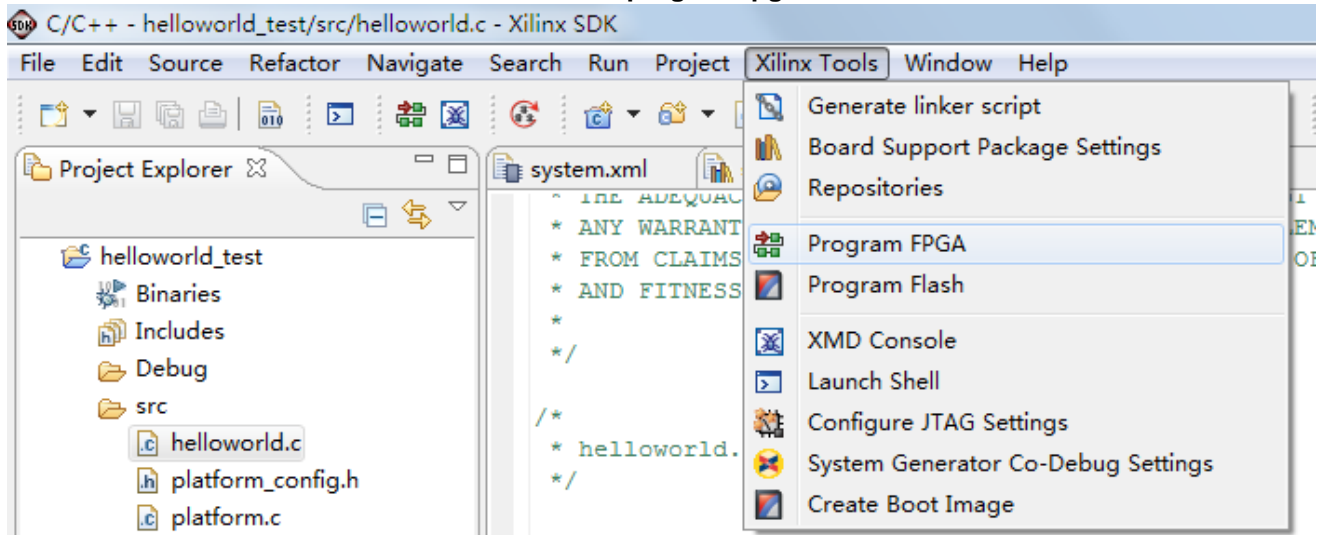


如果报了“no such port”的错误，可以通过新建串口，更改端口号。如下图所示，“**CONNECTED**”表示已经连接上了

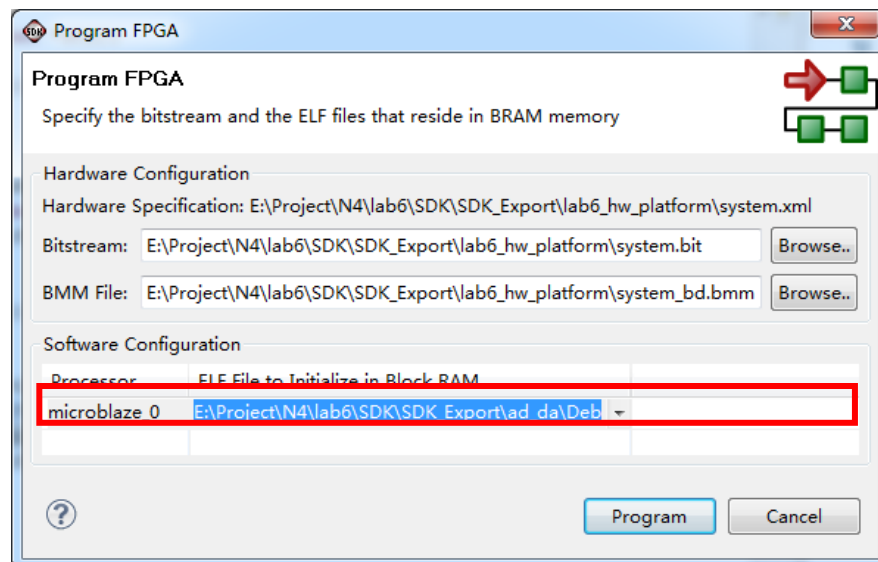
Serial: (COM6, 115200, 8, 1, None, None - CONNECTED)

## 7-5. 将程序下载到板子上并运行

### 7-5-1. 在页面上方，xilinx tools 下拉菜单中选择 program fpga

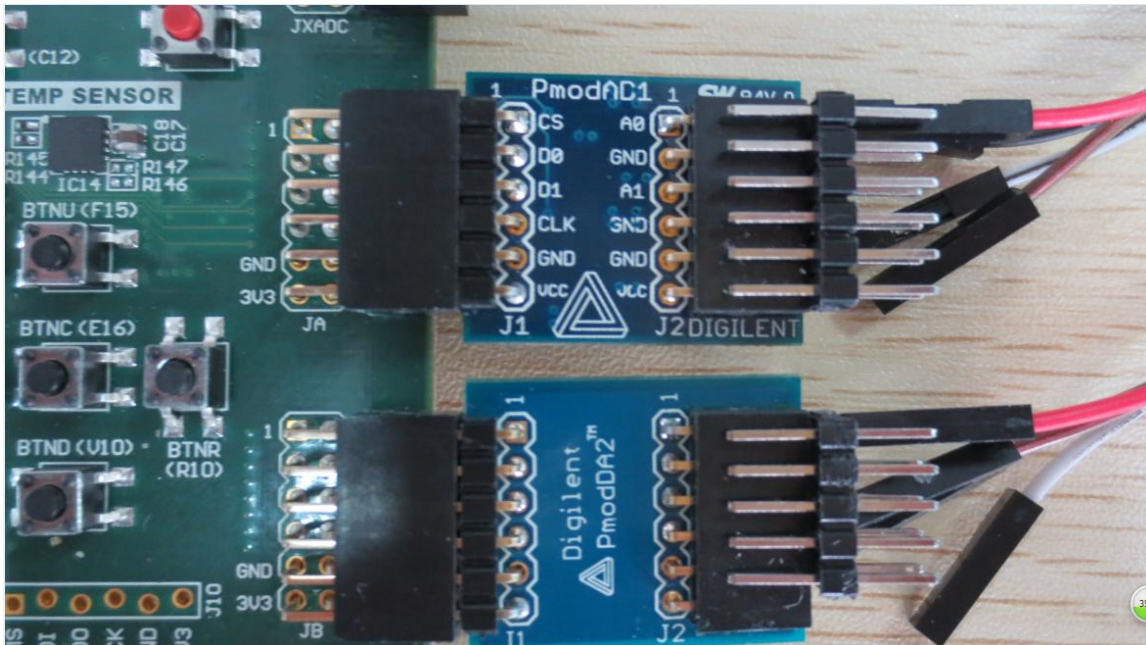


### 7-5-2. 注意要选择正确的 elf 文件:



点击 **program**

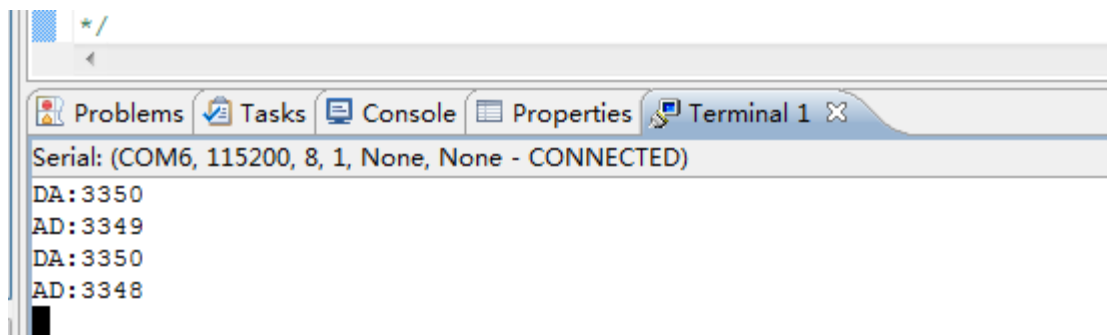
## 7-6. AD/DA 链接图



注意：AD 转换器的 A0 端口和 DA 转换器的输出端口（1 号输出端口，图示中位于 DA2 模块右上角的端口）用杜邦线链接。这样 DA 转换器的输出会通过杜邦线流向 AD 转换器的输入，完成数字信号转模拟信号，模拟信号再转换成数字信号的过程。

7-7. 点击 N4 板上 BTNR 键，然后按下键盘上任意按键

7-8. 重复 7-7 可看到模数，数模转换器 采集转换的结果。在串口中看到结果：



如上图所示：

DA:

1. 本实验所用 DA 转换器的数字信号转换精度为 12 位，模拟信号幅度 0-3.3v。
2. 3350 为 2.7V 转换成数字信号的数值。
3. 12 位二进制数字上限为 4095，待推送的数字信号数值应为（电压值 / 模拟信号上限）\* 数字信号上限。
4. 转换公式：数字信号数值 =  $(x/3.3) * 4095$ 。x 表示待转换的电压值。
5. DA 转换器接收到数字信号后，通过 DA 转换芯片，将数字信号转换成模拟信号后，由 1 号输出端口输出，通过杜邦线，接入 AD 转换器的模拟信号输入端口。

**AD:**

1. 本实验所用 AD 转换器的数字信号转换精度为 12 位，模拟信号幅度 0-3.3v。
2. 3348/3349 为 AD 转换器将模拟信号转换成数字信号的数值，理想误差应在±3 以内。
3. 12 位二进制数字上限为 4095，最终得到的电压值应为（采集到的数字信号 / 数字信号上限）\* 模拟信号幅度上限。
4. 转换公式：电压值 =  $(x/4095) * 3.3$ ，x 表示 AD 转换器将模拟信号转换成数字信号并存储到软件寄存器中的数值。
5. AD 转换器接收到模拟信号，通过 AD 转换芯片，将模拟信号转换成数字信号。Microblaze 读取软件寄存器中的数值，并通过公示将其转换成具体的电压值。