
ADA106

实验教程

Rev. 1.0



版本记录

版本	时间	描述
Rev. 1.0	2019-03-18	第一版发布

版权声明:

XME/IME 系列核心模块、评估板、配套子卡及其相关知识产权由微相科技有限公司所有。

本文档由微相科技有限公司版权所有，并保留一切权利。在未经微相公司书面许可的情况下，不得以任何方式或形式来修改、分发或复制本文档的任何部分。

免责声明

产品中所提供的程序源代码、软件、资料文档等，微相科技不提供任何类型的担保；不论是明确的，还是隐含的，包括但不限于合适特定用途的保证，全部的风险，由使用者来承担。

目录

版本记录 1

版权声明： 2

免责声明 2

目录 3

1. 制作 coe 文件 4

2. 新建工程,工程名为 ADDA_7020..... 5

3. 添加 ROM IP 5

4. 添加 PLL IP..... 7

5. 编写 ADDA_7020 的顶层 verilog 程序 8

6. 添加 I/O 约束 11

7. 设置 Debug..... 12

8. 调试 15

本实验中, 程序要输出 8 位的正弦波数据作为 ADA106 模块的 DA 数字输入, 产生一个正弦波信号, 再将这个正弦波信号通过 SMA 线输入 AD 的模拟口, FPGA 板卡采集 AD 的数据, 通过内部 ILA 显示采集的数据

1. 制作 coe 文件

在 matlab 中

```
X=linspace(0,2*pi,512);
```

```
Y=(255/2)*(sin(X)+1);
```

```
Y=uint8(Y);
```

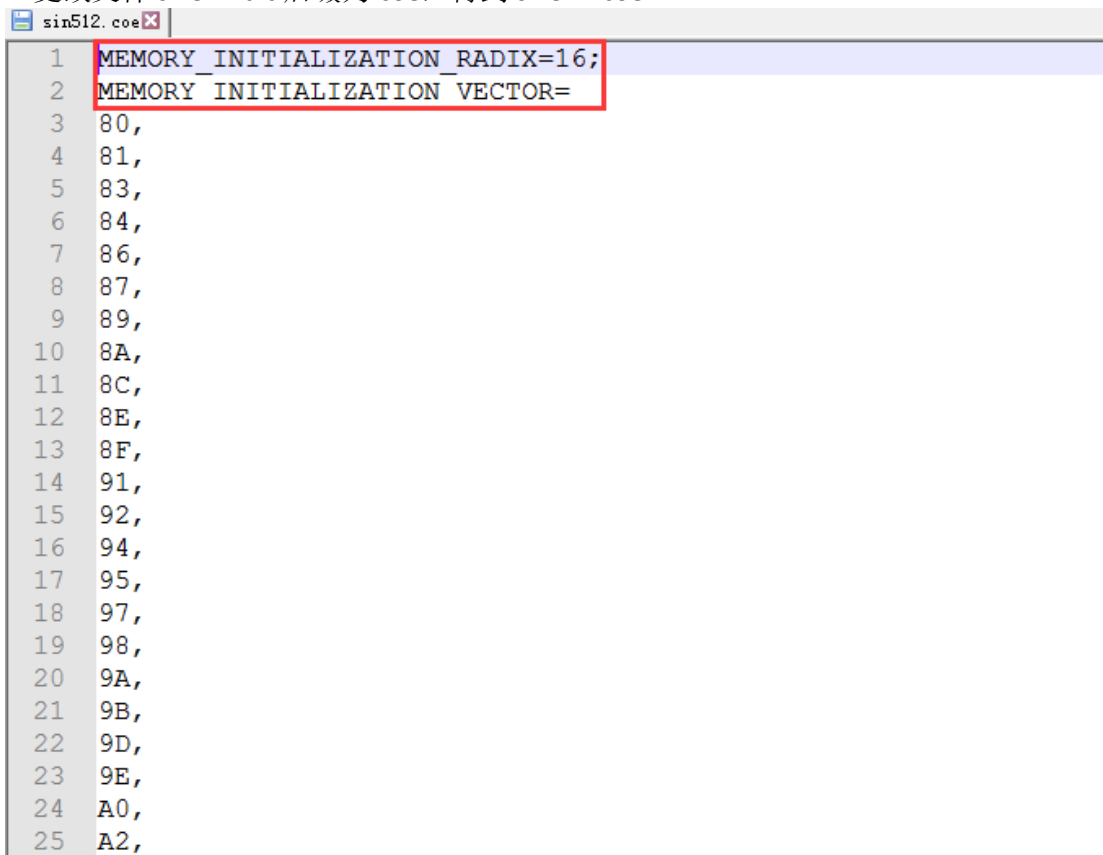
```
plot(X,Y);
```

```
dlmwrite('sin512.txt',Y);//保存数据为文本文件, 数据之间以逗号结尾
```

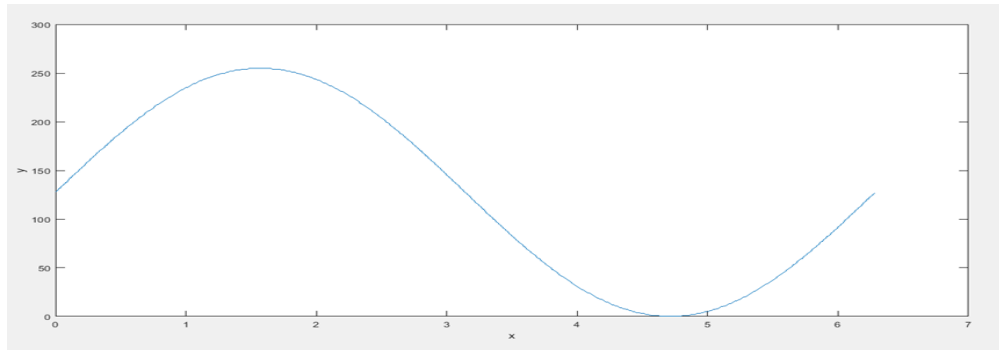
```
MEMORY_INITIALIZATION_RADIX=16;
```

```
MEMORY_INITIALIZATION_VECTOR=
```

更改文件 sin512.txt 后缀为.coe, 得到 sin512.coe



```
sin512.coe
1 MEMORY_INITIALIZATION_RADIX=16;
2 MEMORY_INITIALIZATION_VECTOR=
3 80,
4 81,
5 83,
6 84,
7 86,
8 87,
9 89,
10 8A,
11 8C,
12 8E,
13 8F,
14 91,
15 92,
16 94,
17 95,
18 97,
19 98,
20 9A,
21 9B,
22 9D,
23 9E,
24 A0,
25 A2,
```

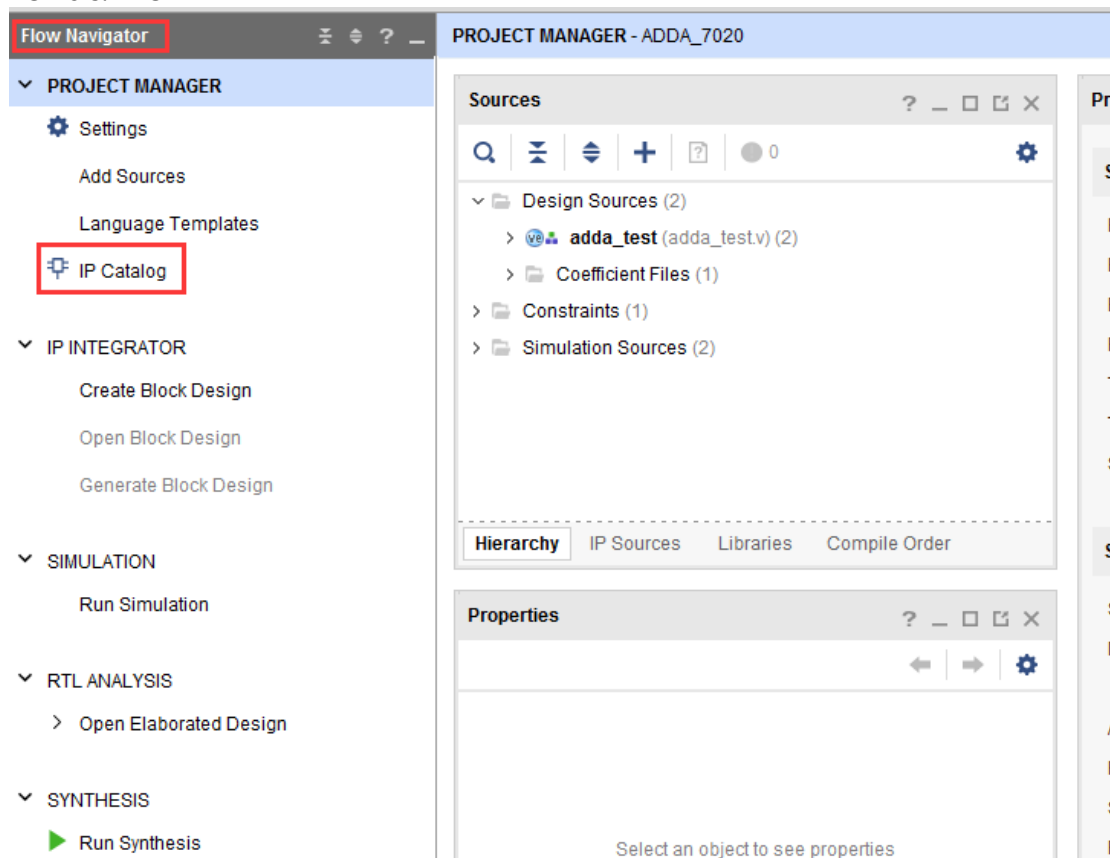


2. 新建工程,工程名为 ADDA_7020

点击 File -> New Project

3. 添加 ROM IP

点击 Flow Navigator -> PROJECT MANAGER -> IP Catalog -> Memories & Storage Elements -> RAMs & ROMs & BROAM



Project Summary x IP Catalog x

Cores | Interfaces

Name	AXI4	Status	License	VLNV
> Basic Elements				
> Communication & Networking				
> Debug & Verification				
> Digital Signal Processing				
> Embedded Processing				
> FPGA Features and Design				
> Kernels				
> Math Functions				
▼ Memories & Storage Elements				
ECC		Production	Included	xilinx.com
> FIFOs				
> Memory Interface Generators				
> RAMs & ROMs				
▼ RAMs & ROMs & BRAM				
Block Memory Generator	AXI4	Production	Included	xilinx.com
> Partial Reconfiguration				

Details

Memory Type 选择 Single Port ROM

Component Name blk_mem_gen_0

Basic | Port A Options | Other Options | Summary

Interface Type Native ☐ Generate address interface with 32 bits

Memory Type Single Port ROM ☐ Common Clock

ECC Options

ECC Type No ECC

☐ Error Injection Pins Single Bit Error Injection

Write Enable

☐ Byte Write Enable

Byte Size (bits) 9

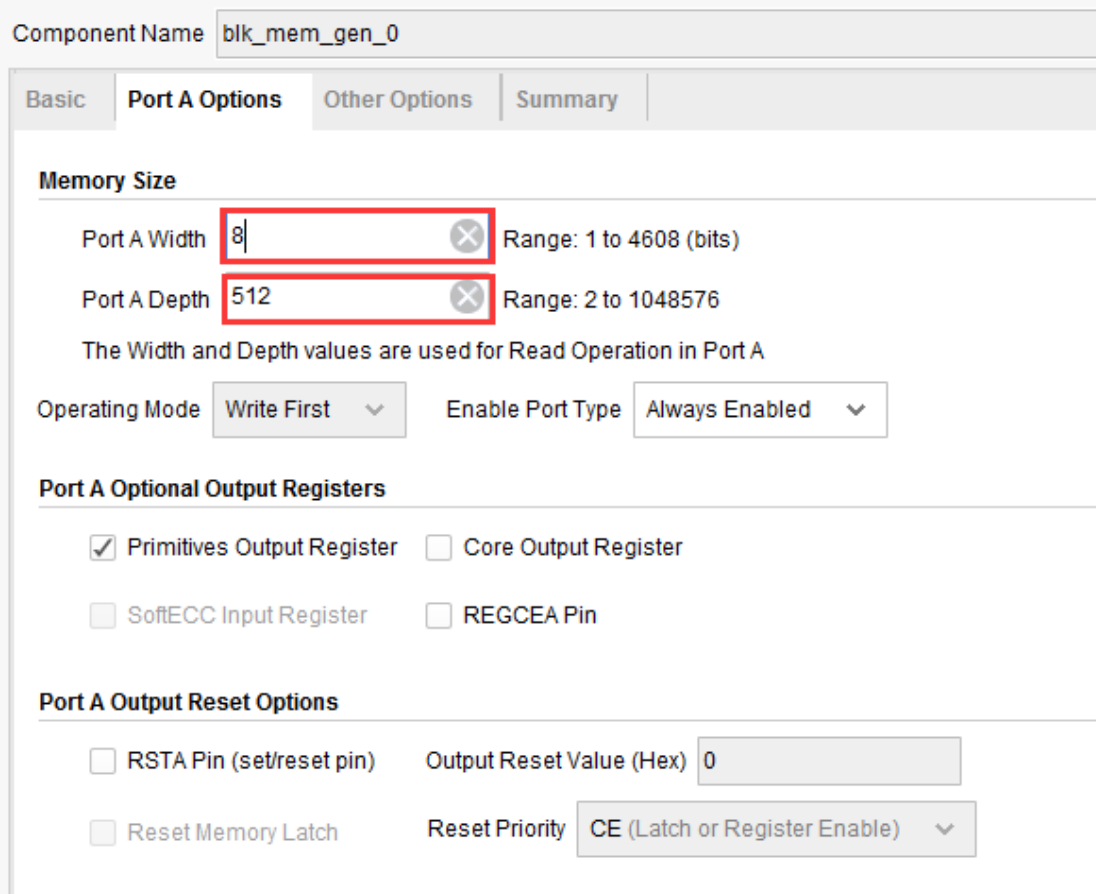
Algorithm Options

Defines the algorithm used to concatenate the block RAM primitives. Refer datasheet for more information.

Algorithm Minimum Area

Primitive 8kx2

Port A Width 选择 8, Port A Depth 选择 512



Component Name blk_mem_gen_0

Basic Port A Options Other Options Summary

Memory Size

Port A Width 8 Range: 1 to 4608 (bits)

Port A Depth 512 Range: 2 to 1048576

The Width and Depth values are used for Read Operation in Port A

Operating Mode Write First Enable Port Type Always Enabled

Port A Optional Output Registers

☒ Primitives Output Register ☐ Core Output Register

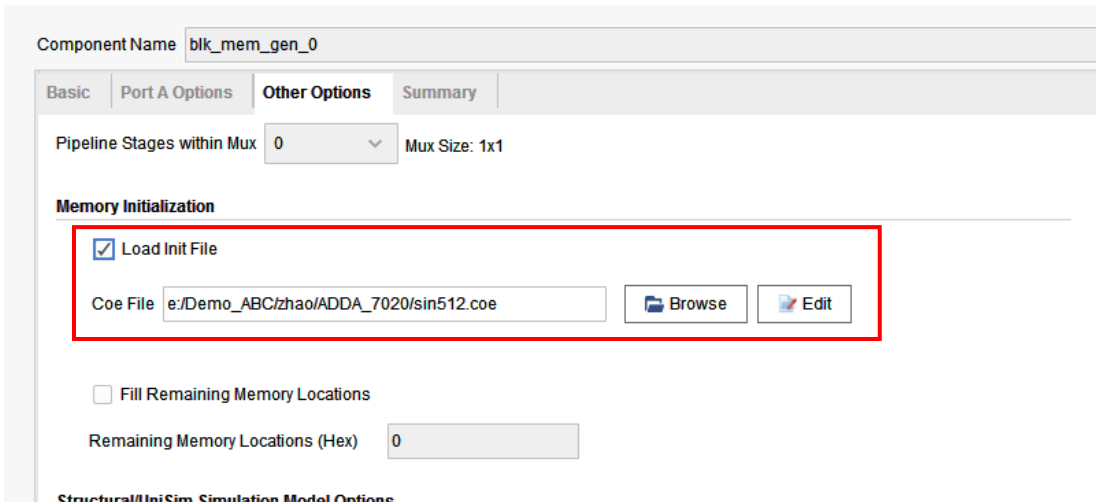
☐ SoftECC Input Register ☐ REGCEA Pin

Port A Output Reset Options

☒ RSTA Pin (set/reset pin) Output Reset Value (Hex) 0

☐ Reset Memory Latch Reset Priority CE (Latch or Register Enable)

Coe File 选择 步骤 1 制作的 coe 文件



Component Name blk_mem_gen_0

Basic Port A Options Other Options Summary

Pipeline Stages within Mux 0 Mux Size: 1x1

Memory Initialization

☒ Load Init File

Coe File e:/Demo_ABC/zhao/ADDA_7020/sin512.coe Browse Edit

☐ Fill Remaining Memory Locations

Remaining Memory Locations (Hex) 0

Structural/Ini Sim Simulation Model Options

4. 添加 PLL IP

点击 Flow Navigator -> PROJECT MANAGER -> IP Catalog -> FPGA Features and Design -> Clocking -> Clocking Wizard, 将出现图窗口, 并作如下设置
Primitive 选择 PLL

Component Name

Clocking Options | Output Clocks | Port Renaming | PLLE2 Settings | Summary

Clock Monitor

☐ Enable Clock Monitoring

Primitive

☐ MMCM ☒ **PLL**

Output Clock 选择 clk_out1 50MHZ clk_out2 25MHZ, Reset Type 选择 Active Low

Component Name

Clocking Options | **Output Clocks** | Port Renaming | PLLE2 Settings | Summary

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives
		Requested	Actual	Requested	Actual	Requested	Actual	
<input checked="" type="checkbox"/> clk_out1	clk_out1	50	50.000	0.000	0.000	50.000	50.0	BUFG
<input checked="" type="checkbox"/> clk_out2	clk_out2	25	25.000	0.000	0.000	50.000	50.0	BUFG
<input type="checkbox"/> clk_out3	clk_out3	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG

☐ USE CLOCK SEQUENCING

Clocking Feedback

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1

Source

☒ Automatic Control On-Chip
☐ Automatic Control Off-Chip
☐ User-Controlled On-Chip
☐ User-Controlled Off-Chip

Signaling

☒ Single-ended
☐ Differential

Enable Optional Inputs / Outputs for MMCM/PLL

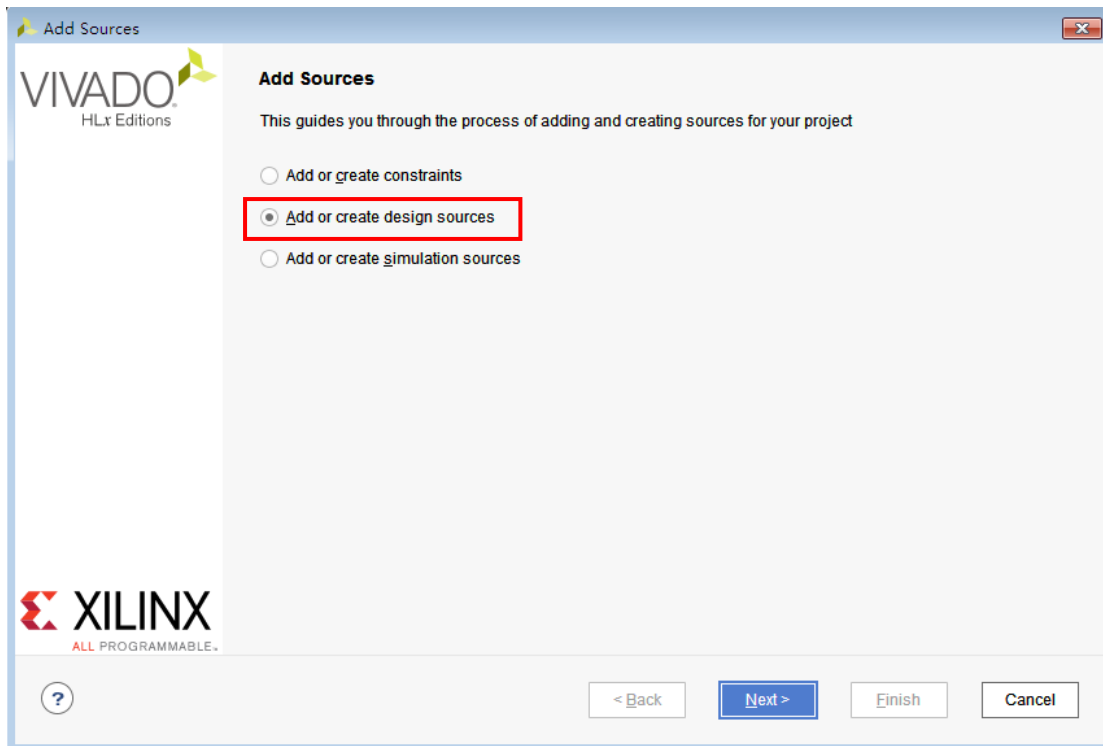
☒ reset ☐ power_down
☒ locked

Reset Type

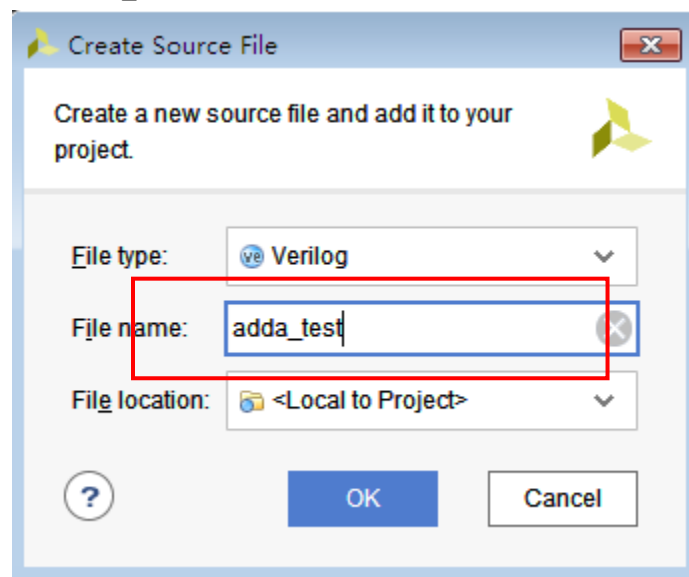
☐ Active High ☒ Active Low

5. 编写 ADDA_7020 的顶层 verilog 程序

左击 SOURCES-> Design Source ->Add Source



点击 Create File,文件名填 adda_test,



adda_test.v 程序如下:

```
`timescale 1ns / 1ps
module adda_test(
```

```
input clk,          //fpga clock

output daclk,
(* mark_debug = "TRUE"*)
output [7:0] dadata,    //DA data

output adclk,
(* mark_debug = "TRUE"*)
input [7:0] addata      //AD data

);
reg [8:0] rom_addr;

(*DONT_TOUCH="{TRUE|FALSE}"*)reg [7:0] ad_data;

wire [7:0] rom_data;
wire clk_25;
wire clk_50;

assign dadata=rom_data;
assign daclk=clk_50;
assign adclk=clk_25;

//DA output sin waveform
always @(negedge clk_50)
begin
    rom_addr <= rom_addr + 1'b1 ;
end

always @(posedge clk_25)
begin
    ad_data <= addata ;
end

blk_mem_gen_0 blk_mem_gen_0 (
    .clka(clk_50), // input clka
    .addra(rom_addr), // input [8 : 0] addra
    .douta(rom_data) // output [7 : 0] douta
);

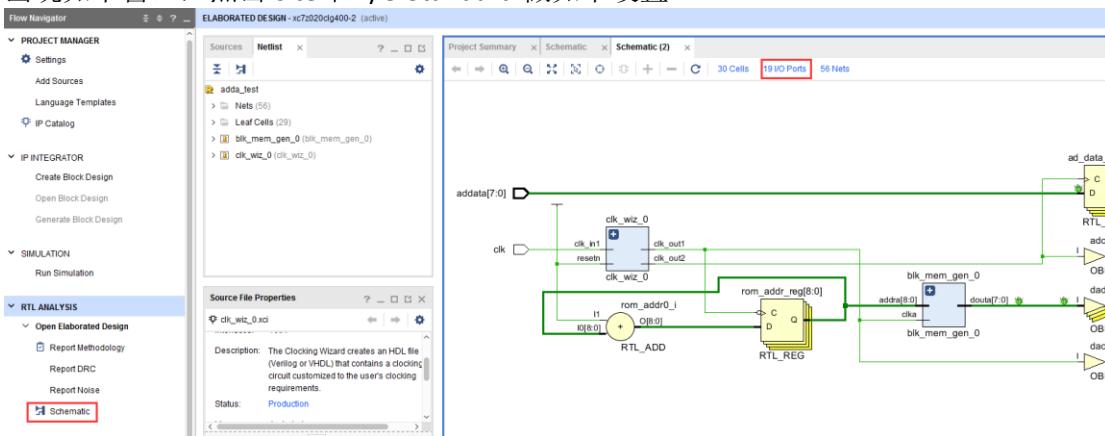
clk_wiz_0 clk_wiz_0
    // Clock in ports
    .clk_in1(clk), //IN
    // Clock out ports
    .clk_out1(clk_50), //OUT
```

```
.clk_out2(clk_25), // OUT
// Status and control signals
.resetn(1'b1),// IN
.locked(); // OUT
```

```
Endmodule
```

6. 添加 I/O 约束

点击左侧边栏 RTL ANALYSIS 下的 Open elaborated design
 在右侧的 Schematic 窗口下点击 I/O Ports
 出现如下窗口，点击 site 和 I/O Standard 做如下设置



Name	Direction	Net	Diff Pair	Package Pin	Faced	Bank	I/O Std	Vcco	Vref	Drive Strength	Slow Type	Pull Type	On-Chip
BRAM_PORTA_63243 (9)	OUT				✓	35	LVCNMOS3*	3.300		12	FAST	NONE	FP_1
data (8)	OUT				✓	35	LVCNMOS3*	3.300		12	FAST	NONE	FP_1
data[4]	OUT			K14	✓	35	LVCNMOS3*	3.300		12	FAST	NONE	FP_1
data[3]	OUT			G20	✓	35	LVCNMOS3*	3.300		12	FAST	NONE	FP_1
data[2]	OUT			G19	✓	35	LVCNMOS3*	3.300		12	FAST	NONE	FP_1
data[5]	OUT			H15	✓	35	LVCNMOS3*	3.300		12	FAST	NONE	FP_1
data[1]	OUT			F20	✓	35	LVCNMOS3*	3.300		12	FAST	NONE	FP_1
data[0]	OUT			F19	✓	35	LVCNMOS3*	3.300		12	FAST	NONE	FP_1
data[7]	OUT			K18	✓	35	LVCNMOS3*	3.300		12	FAST	NONE	FP_1
data[5]	OUT			J14	✓	35	LVCNMOS3*	3.300		12	FAST	NONE	FP_1
dack	OUT			H18	✓	35	LVCNMOS3*	3.300		12	FAST	NONE	FP_1
addata (8)	IN				✓	34	LVCNMOS3*	3.300				NONE	NON
addata[5]	IN			W19	✓	34	LVCNMOS3*	3.300				NONE	NON
addata[2]	IN			V16	✓	34	LVCNMOS3*	3.300				NONE	NON
addata[3]	IN			W16	✓	34	LVCNMOS3*	3.300				NONE	NON
addata[0]	IN			R18	✓	34	LVCNMOS3*	3.300				NONE	NON
addata[7]	IN			V18	✓	34	LVCNMOS3*	3.300				NONE	NON
addata[5]	IN			V17	✓	34	LVCNMOS3*	3.300				NONE	NON
addata[4]	IN			W18	✓	34	LVCNMOS3*	3.300				NONE	NON
addata[1]	IN			R17	✓	34	LVCNMOS3*	3.300				NONE	NON
adck	OUT			P15	✓	34	LVCNMOS3*	3.300		12	FAST	NONE	FP_1
clk	IN			K17	✓	35	LVCNMOS3*	3.300				NONE	NON

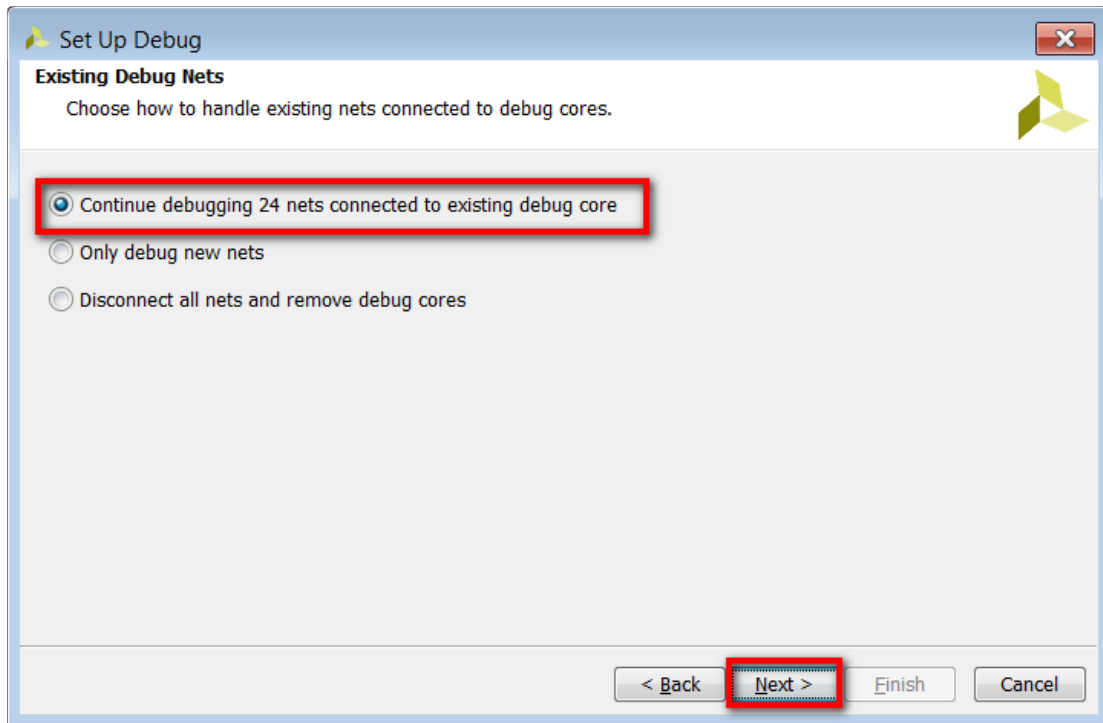
添加完成后保存，添加 XDC 文件命名为 ADDA_7020。

7. 设置 Debug

在左侧栏 synthesis 下 点击 Run Synthesis,
Synthesis Completed 弹窗口 点击 Cancel,
左侧边栏展开 Open Synthesized Design Set Up Debug,
如下图, 点击 Next



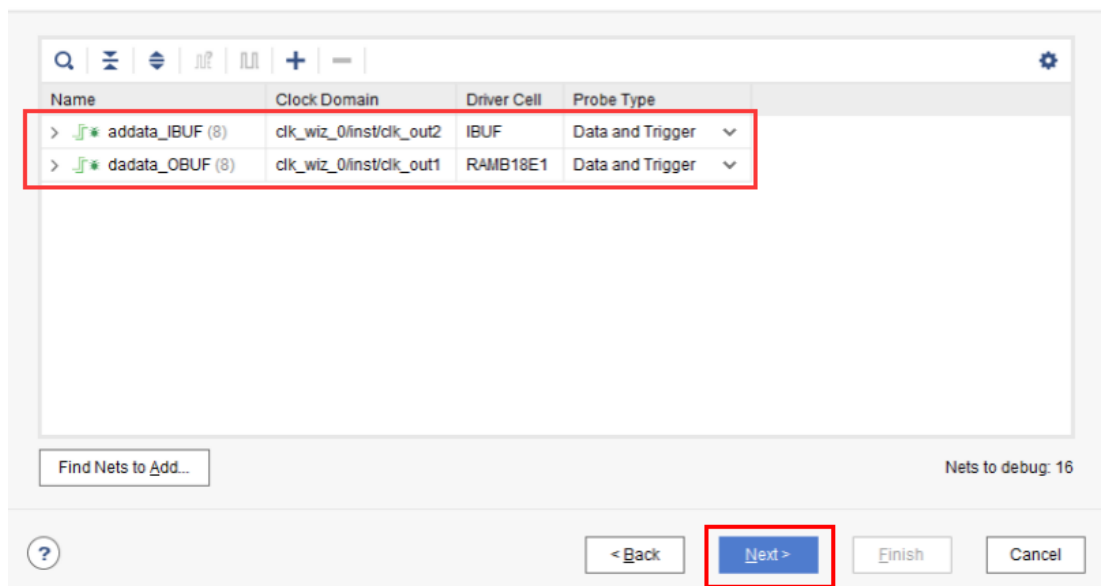
如下选择 Continue debugging, 点击 Next,



如下勾选，点击 Next，

Nets to Debug

The nets below will be debugged with ILA cores. To add nets click "Find Nets to Add". You can also select nets in the Netlist or other windows, then drag them to the list or click "Add Selected Nets".



可以看到，我们在源代码中 Mark 的信号，都出现在了这里。点击 next

ILA Core Options

Choose features for the ILA debug cores.

Sample of data depth: 1024

Input pipe stages: 0

Trigger and Storage Settings

☒ Capture control

☒ Advanced trigger

? < Back Next > Finish Cancel

点击 Next

VIVADO
HLx Editions

Set up Debug Summary

2 debug cores will be removed: u_ila_0 and u_ila_1

2 debug cores will be created

Found 2 clocks

☒ Open in Debug layout

To apply the above changes, click Finish

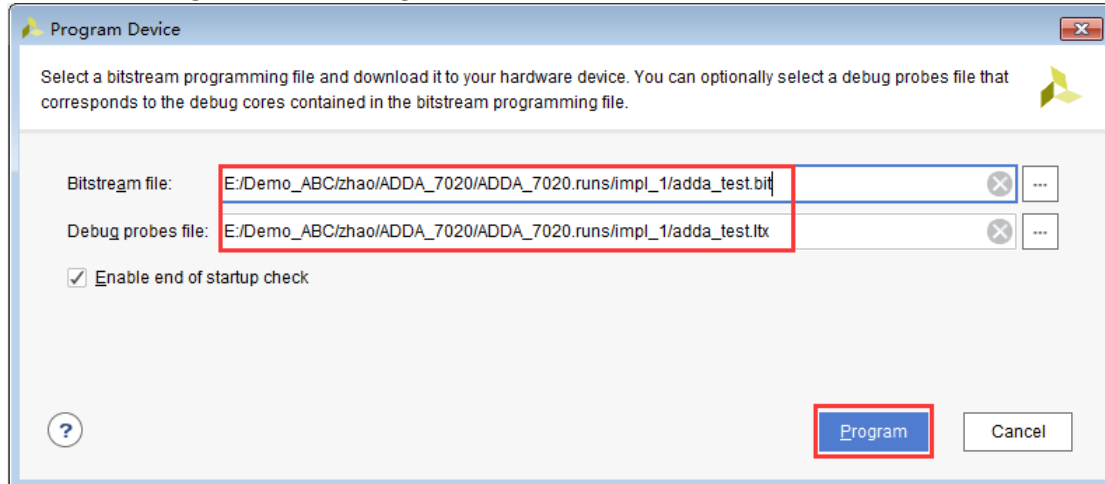
XILINX

? < Back Next > Finish Cancel

点击完成。在左侧边栏 Program and Debug 下点击 Generate Bitstream，Bitstream Generation Completed 弹窗点击 OK。

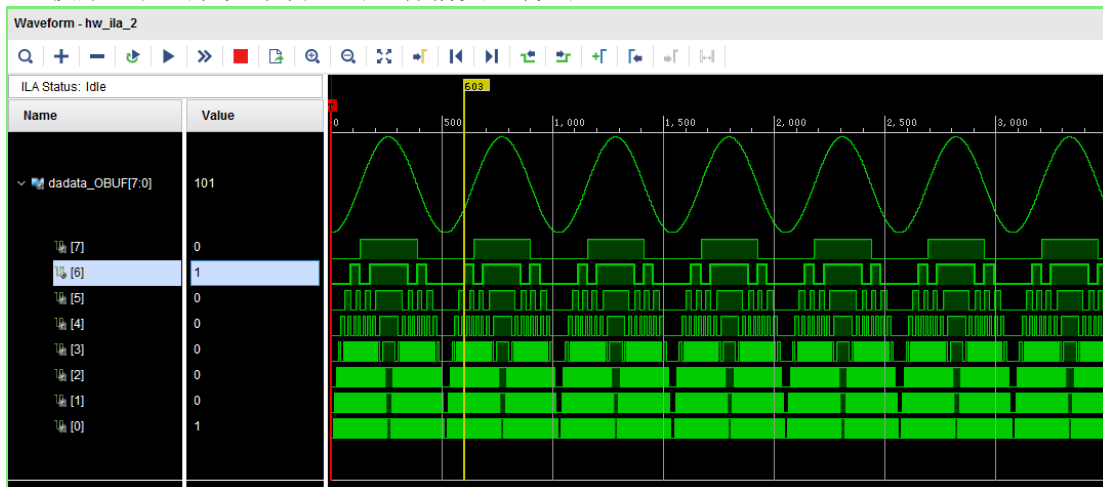
8. 调试

将 ADDA 子卡连接到开发板上 JM2 上，连接电脑 USB 到开发板 JTAG 口，确保开发板拨码开关 1, 2 都在 ON 的位置，给开发板上电，回到左侧边栏，点击 Open Hardware Manager，在 Hardware Manager 窗口点击 Auto Connect 图标，自动检测到 Device 后，右键单击 xc7z020 选择 Program Device，出现如下弹窗，可见除了 Bitstream 外，还有一个 Debug probes 文件，这就是我们插入的 Debug core，点击 Program。



点击 program 出现如下窗口

点击 hw_ila_2 选择 DA 波形，点击 hw_ila_1 选择 AD 波形。
DA 波形，在出现如下窗口点击运行箭头，将出



AD 波形

