GVI KXZ7C01 Zynq Connectivity Kit

10GE MAC Design Example

QuickStart

December 2017

Order Hardware:

https://detail.tmall.com/item.htm?id=562769965498

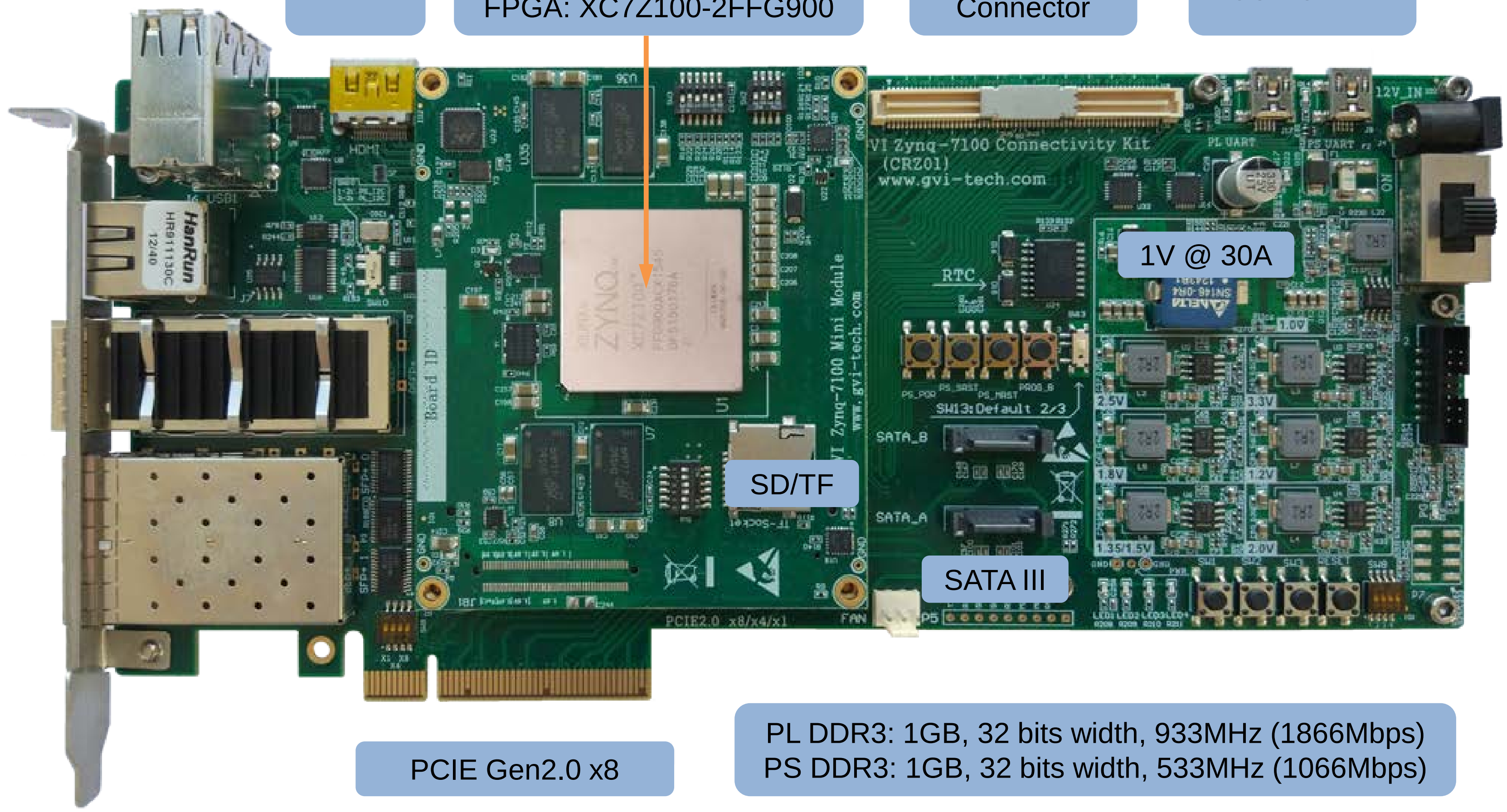
http://www.gvi-tech.com/products-fpga-kits

杭州言曼科技有限公司

Overview

- GVI KXZ7C01 Zynq Connectivity Kit
- Software Requirements
- Design Architecture
- Generate Ten Gigabit Ethernet MAC Core
- Generate Ten Gigabit Ethernet PCS/PMA Core
- System Setup
- 10GE Packet Transfer Demo
- References

GVI Zynq Connectivity Kit – KXZ7C01

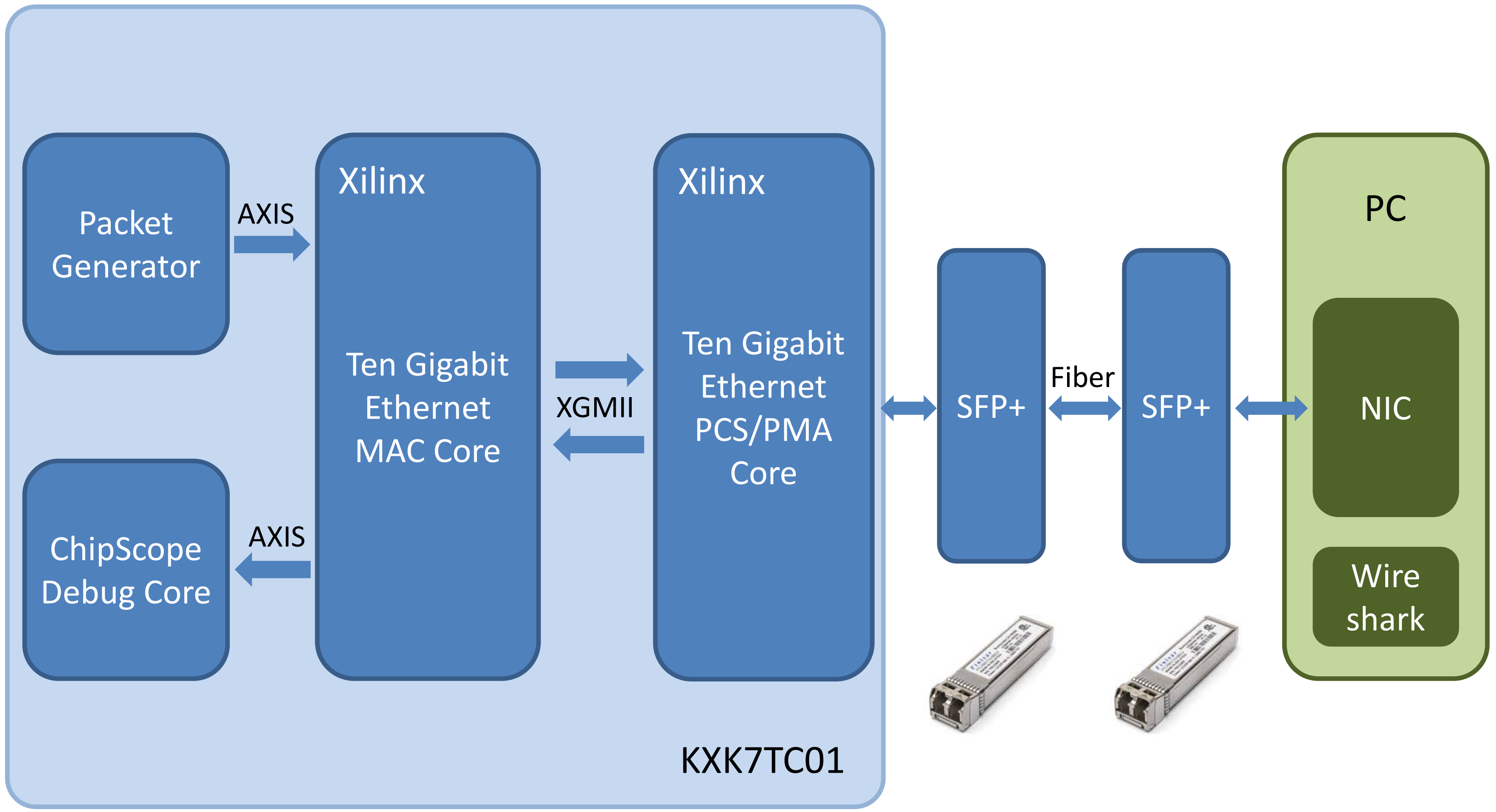


Xilinx Software Requirement

- Xilinx Vivado Design Suite (version 2016.4 is used for development).

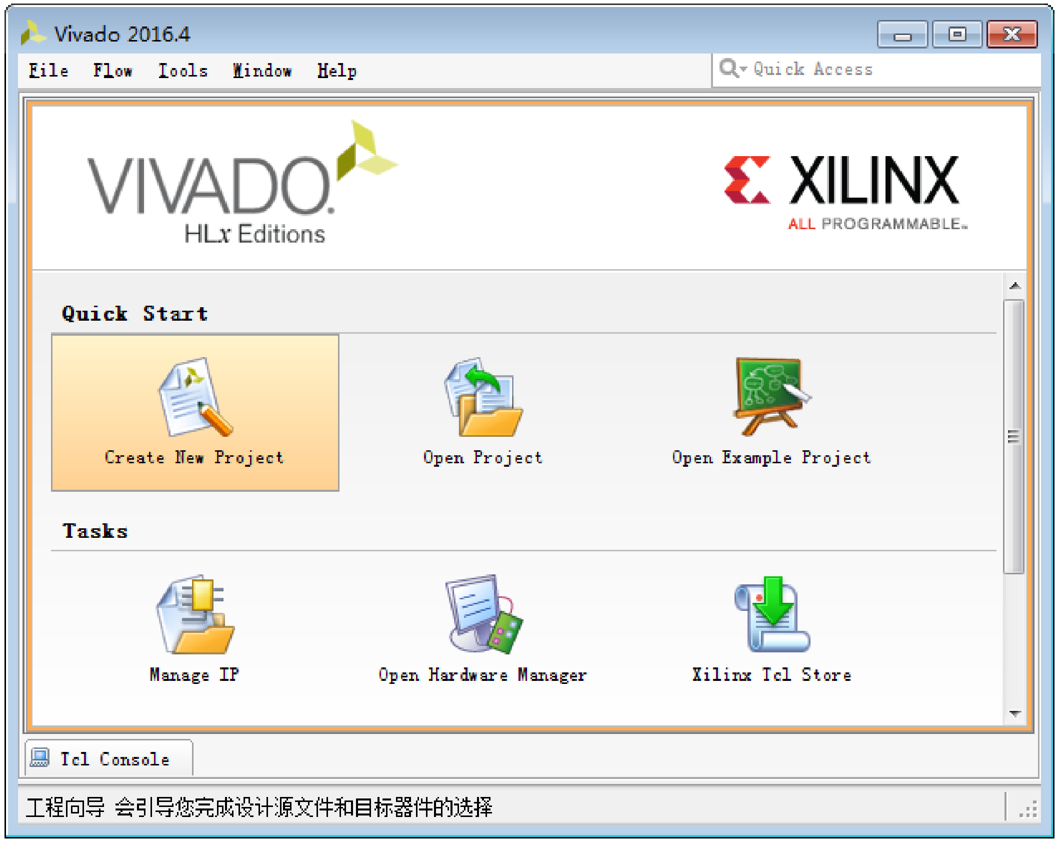
- Wireshark: www.wireshark.org

Architecture



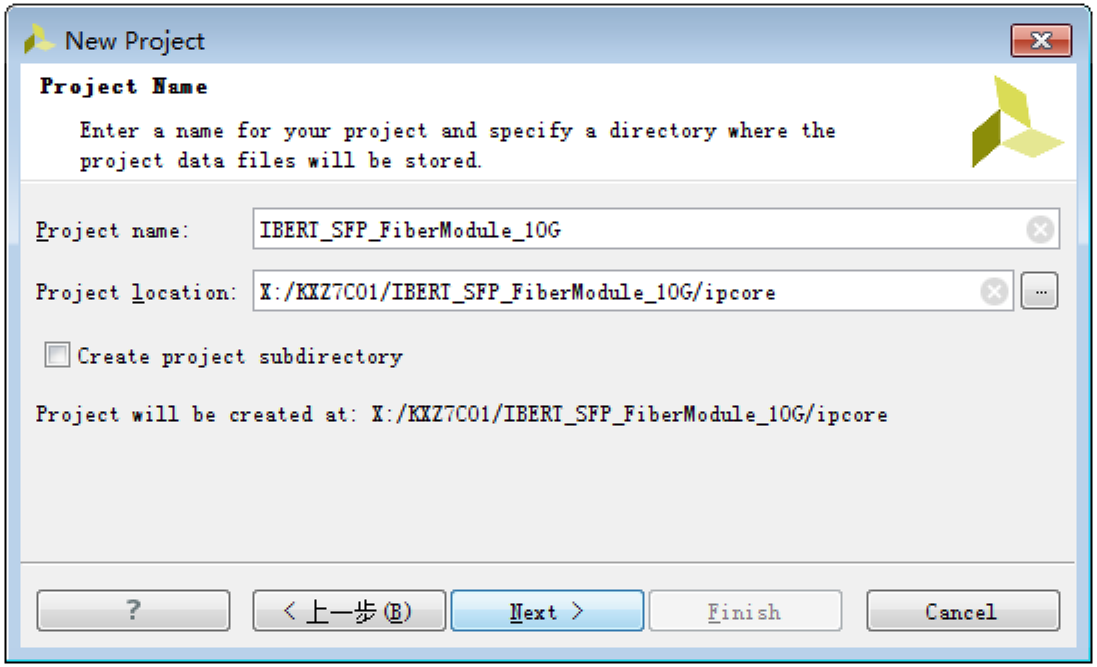
Create Vivado Project

- Open Vivado
  - Start → All Programs → Xilinx Design Tools → Vivado 2016.4 → Vivado
- Select Create New Project



Create IBERT Design

- Set the Project name and location; check Create project subdirectory


New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: IBERT_SFP_FiberModule_10G

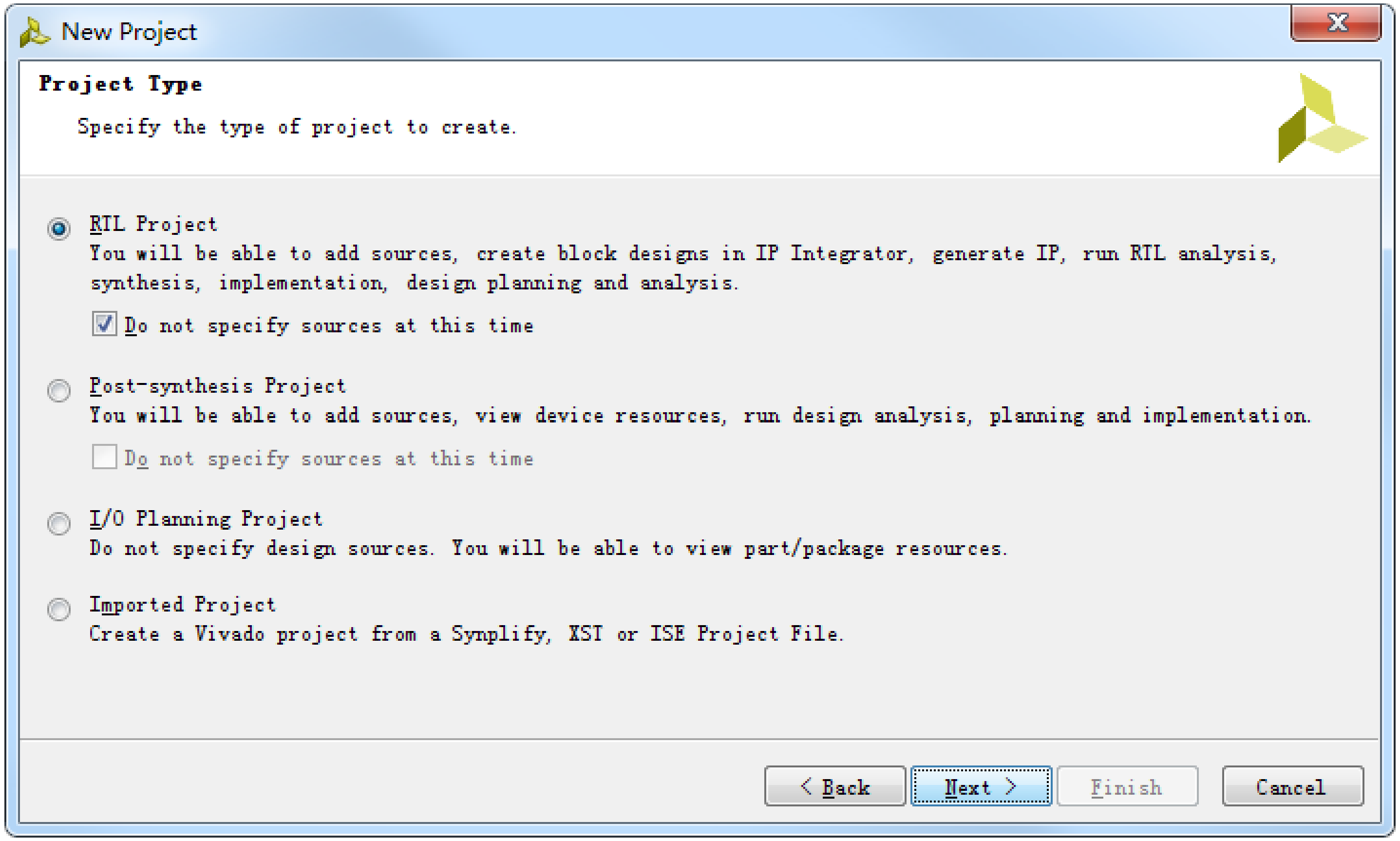
Project location: X:/KXZ7C01/IBERT_SFP_FiberModule_10G/ipcore

☐ Create project subdirectory

Project will be created at: X:/KXZ7C01/IBERT_SFP_FiberModule_10G/ipcore

Create Vivado Project

- Select RTL Project
  - Select Do not specify sources at this time



Project Type

Specify the type of project to create.

(●) RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.

[✓] Do not specify sources at this time

( ) Post-synthesis Project
You will be able to add sources, view device resources, run design analysis, planning and implementation.

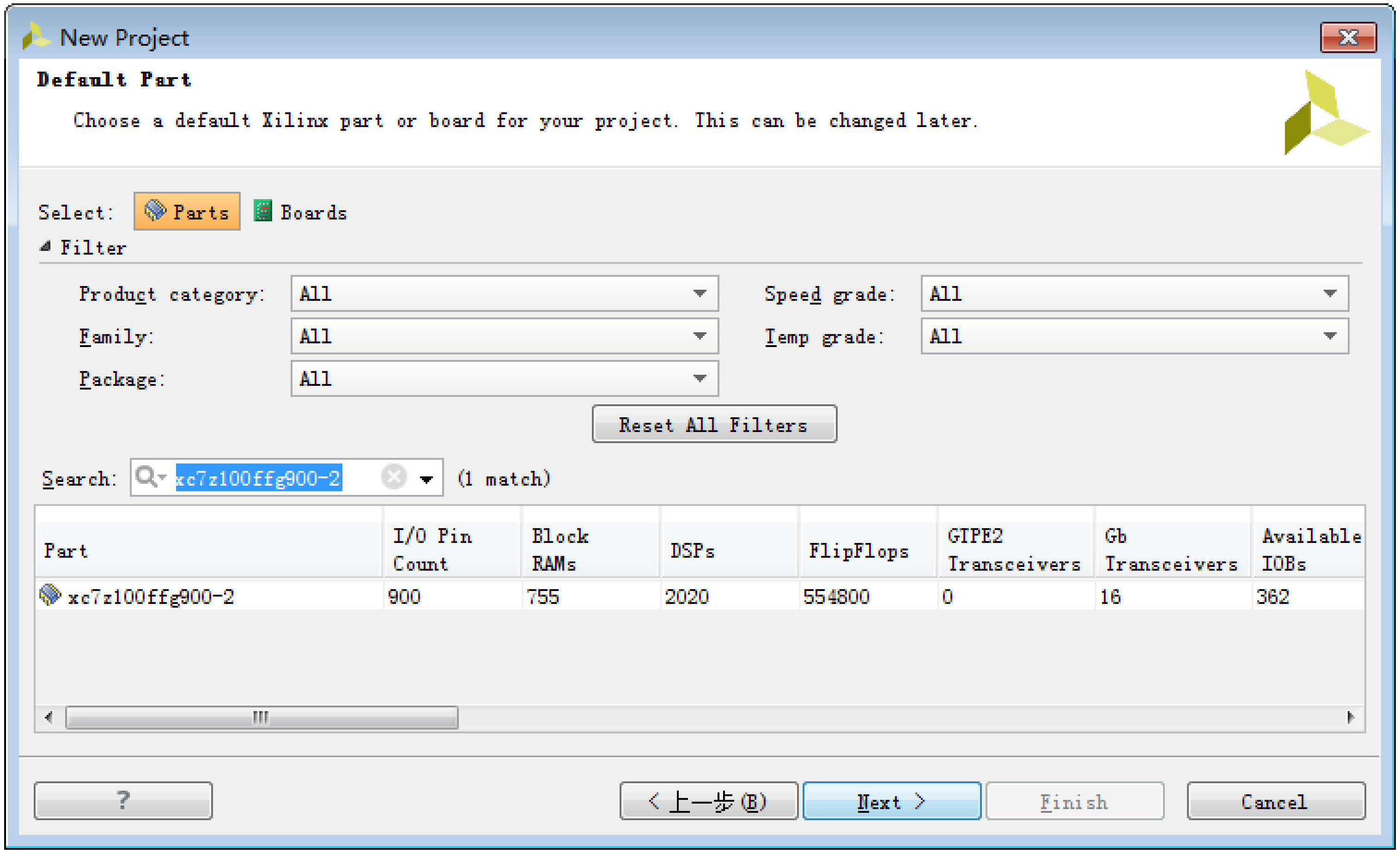
[ ] Do not specify sources at this time

( ) I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

( ) Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

Create Vivado Project

- Select FPGA Part: xc7z100ffg900-2. In the next page, click "Finish".



New Project

Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

Select: Parts Boards

Filter

Product category: All

Family: All

Package: All

Speed grade: All

Temp grade: All

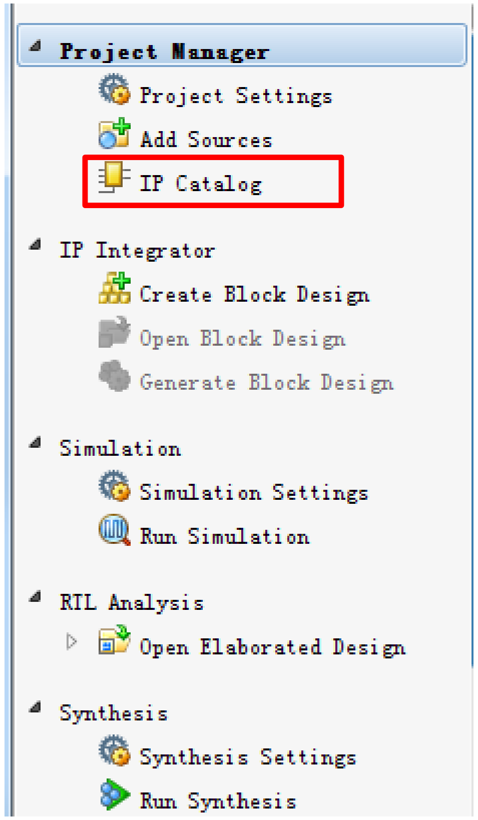
Reset All Filters

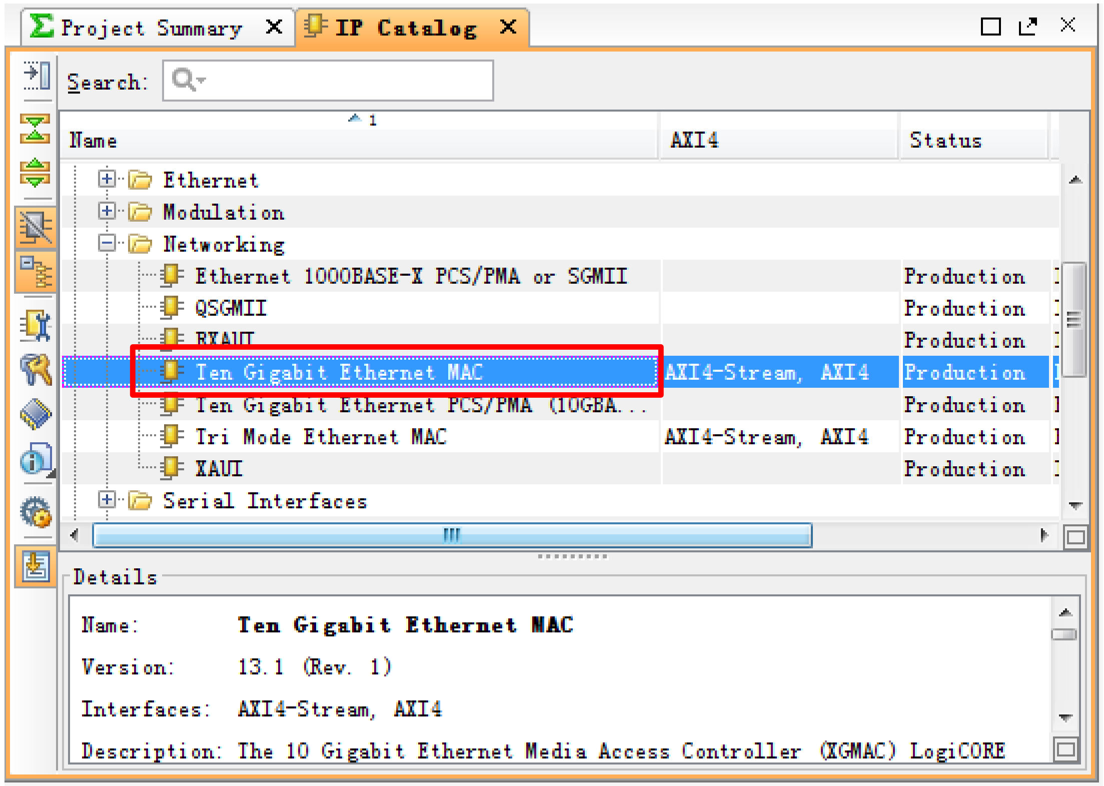
Search: xc7z100ffg900-2 (1 match)

Part	I/O Pin Count	Block RAMs	DSPs	FlipFlops	GTPE2 Transceivers	Gb Transceivers	Available IOBs
xc7z100ffg900-2	900	755	2020	554800	0	16	362

Generate Ten Gigabit Ethernet MAC Core

- Select IP Catalog and double click on the "Ten Gigabit Ethernet MAC Core" (you need a valid license in order to generate the IP Core).





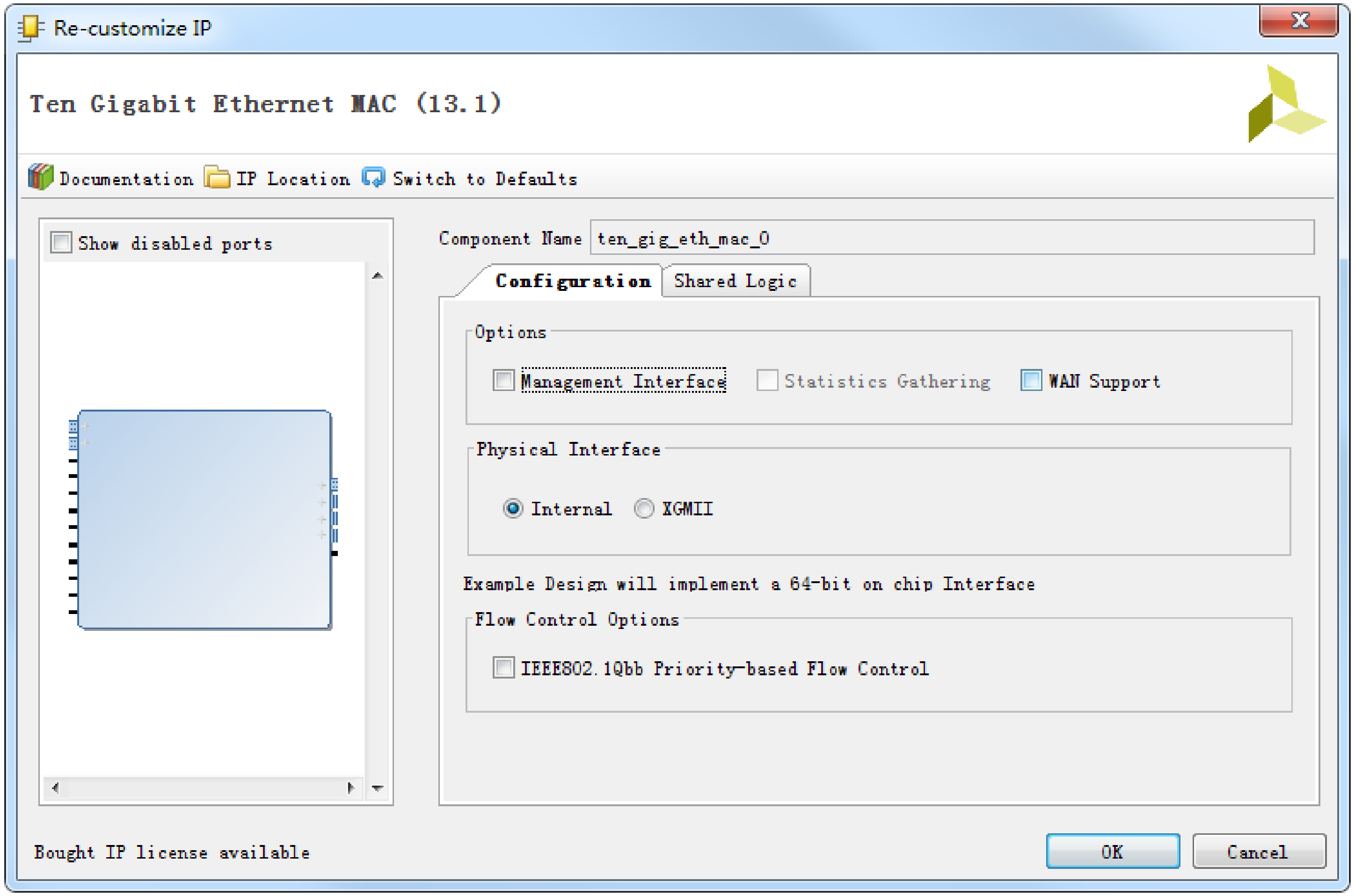
Name	AXI4	Status
Ethernet		
Modulation		
Networking		
Ethernet 1000BASE-X PCS/PMA or SGMII		Production
QSGMII		Production
RXAUI		Production
Ten Gigabit Ethernet MAC	AXI4-Stream, AXI4	Production
Ten Gigabit Ethernet PCS/PMA (10GBA...		Production
Tri Mode Ethernet MAC	AXI4-Stream, AXI4	Production
XAUI		Production
Serial Interfaces		

Details

Name:	**Ten Gigabit Ethernet MAC**
Version:	13.1 (Rev. 1)
Interfaces:	AXI4-Stream, AXI4
Description:	The 10 Gigabit Ethernet Media Access Controller (XGMAC) LogiCORE

Generate Ten Gigabit Ethernet MAC Core

- In the “Configuration” tab, use the configuration as shown in the following:


Re-customize IP

Ten Gigabit Ethernet MAC (13.1)

Show disabled ports [ ]

Component Name: ten_gig_eth_mac_0

Options

[ ] Management Interface [ ] Statistics Gathering [ ] WAN Support

Physical Interface

[✓ Internal] XGMII

Example Design will implement a 64-bit on chip Interface

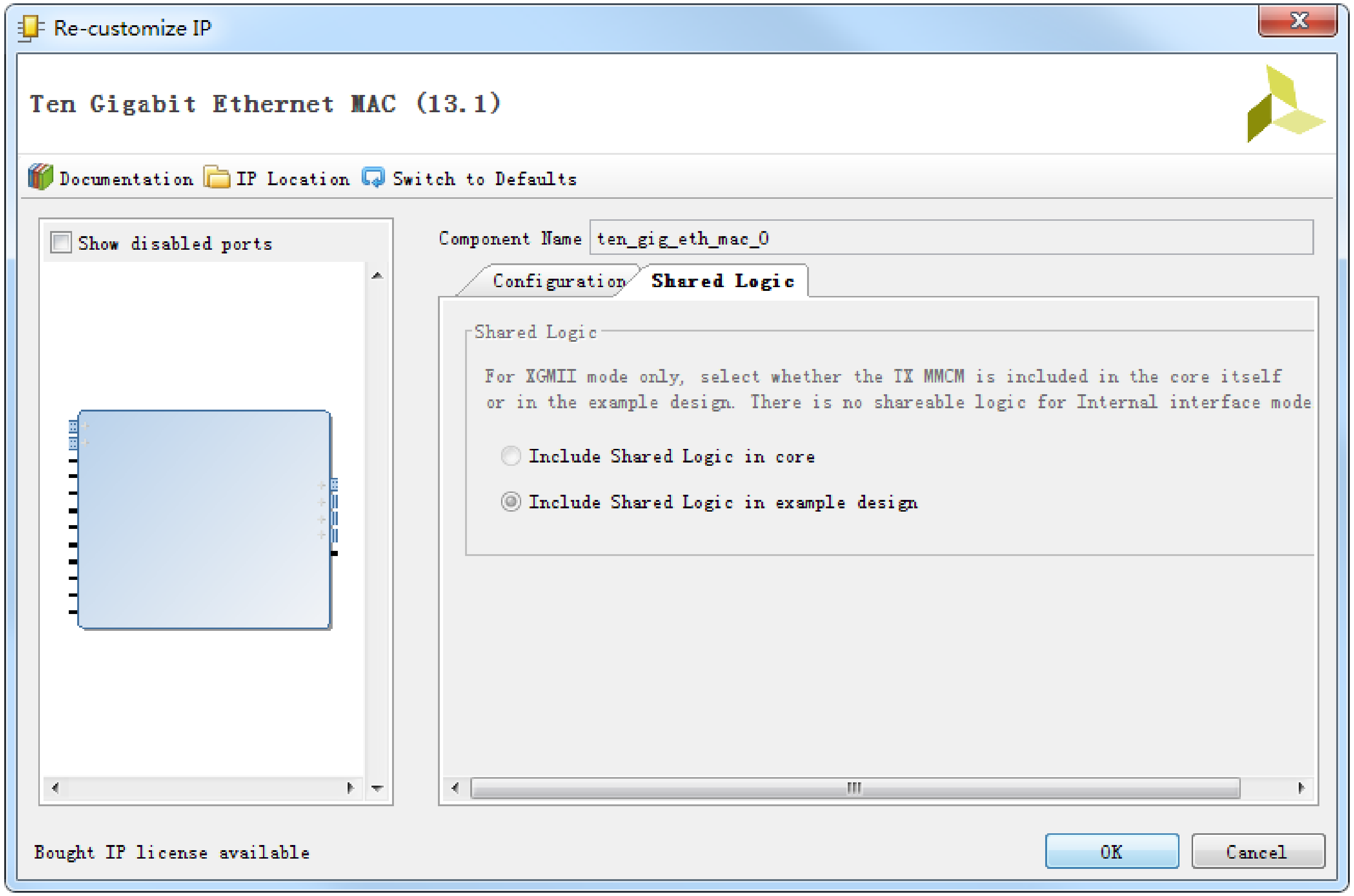
Flow Control Options

[ ] IEEE802.1Qbb Priority-based Flow Control

Bought IP license available

Generate Ten Gigabit Ethernet MAC Core

- In the “Shared Logic” tab, use the configuration as shown in the following, and then click “OK”.



Re-customize IP

Ten Gigabit Ethernet MAC (13.1)

[ ] Show disabled ports

Component Name: ten_gig_eth_mac_0

Shared Logic

For XGMII mode only, select whether the TX MMCM is included in the core itself or in the example design. There is no shareable logic for Internal interface mode

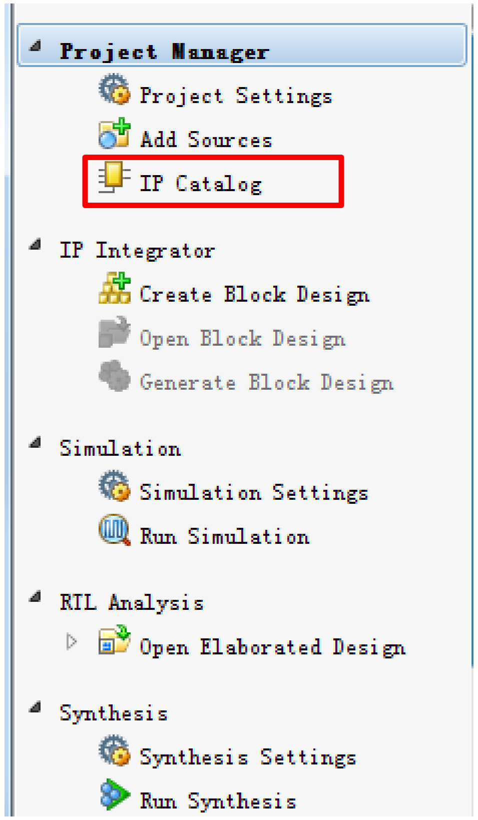
○ Include Shared Logic in core

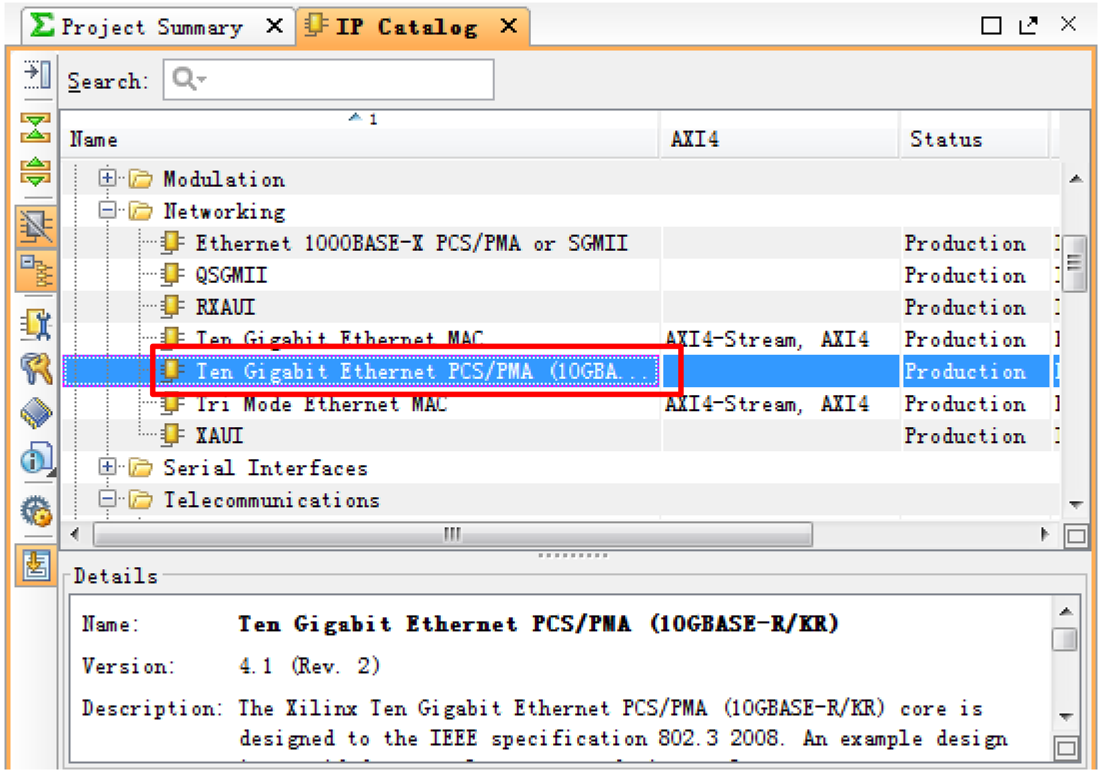
◉ Include Shared Logic in example design

Bought IP license available

Generate Ten Gigabit Ethernet PCS/PMA Core

- Select IP Catalog and double click on the “Ten Gigabit Ethernet PCS/PMA Core” (you need a valid license in order to generate the IP Core).





Name	AXI4	Status
Modulation		
Networking		
Ethernet 1000BASE-X PCS/PMA or SGMII		Production
QSGMII		Production
RXAUI		Production
Ten Gigabit Ethernet MAC	AXI4-Stream, AXI4	Production
Ten Gigabit Ethernet PCS/PMA (10GBA...		Production
Tri Mode Ethernet MAC	AXI4-Stream, AXI4	Production
XAUI		Production
Serial Interfaces		
Telecommunications		

Details

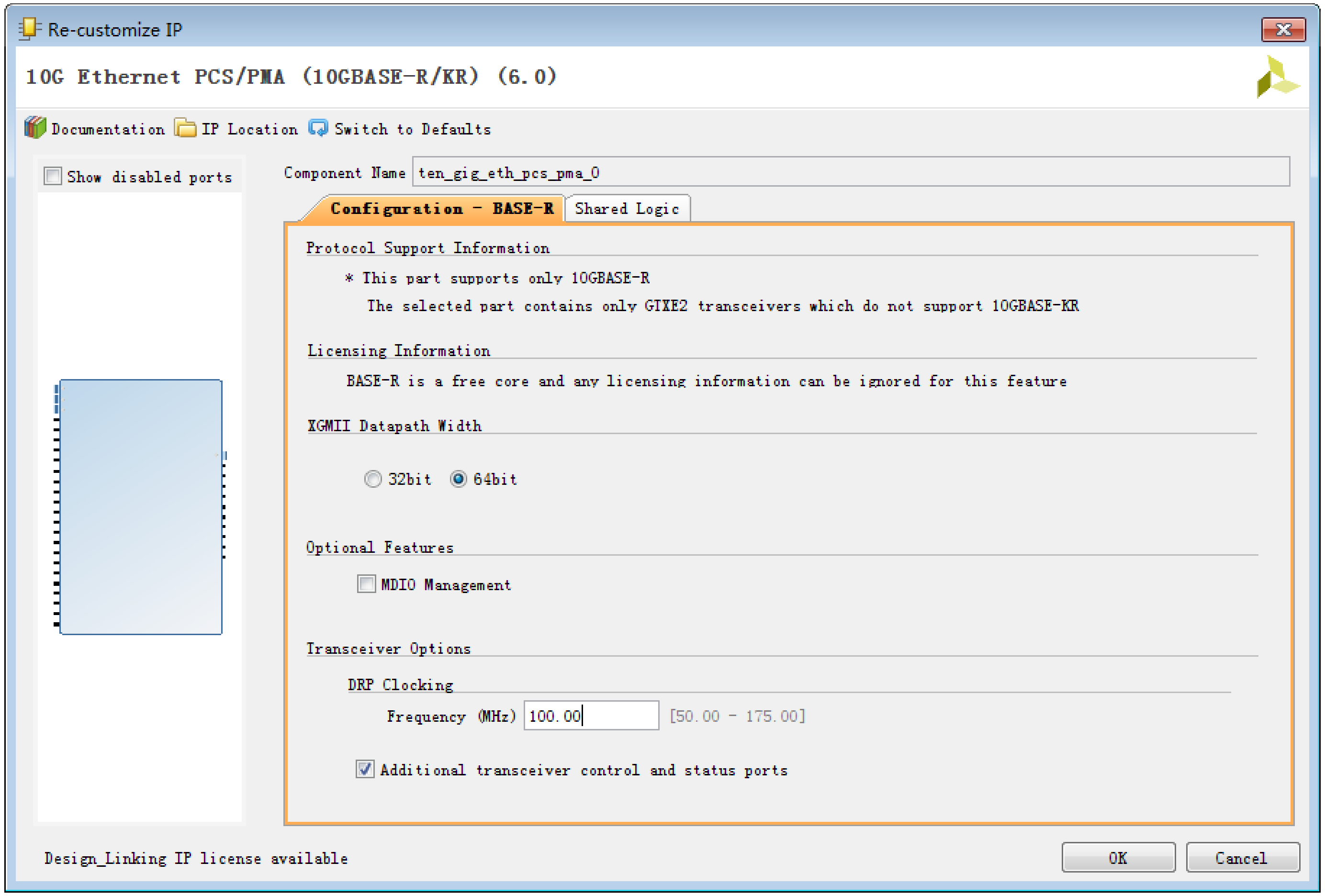
Name: **Ten Gigabit Ethernet PCS/PMA (10GBASE-R/KR)**

Version: 4.1 (Rev. 2)

Description: The Xilinx Ten Gigabit Ethernet PCS/PMA (10GBASE-R/KR) core is designed to the IEEE specification 802.3 2008. An example design

Generate Ten Gigabit Ethernet PCS/PMA Core

- In the “Configuration – BASE-R” tab, use the configuration as shown in the following:



Re-customize IP

10G Ethernet PCS/PMA (10GBASE-R/KR) (6.0)

Show disabled ports [ ]

Component Name: ten_gig_eth_pcs_pma_0

Protocol Support Information

* This part supports only 10GBASE-R
The selected part contains only GTXE2 transceivers which do not support 10GBASE-KR

Licensing Information

BASE-R is a free core and any licensing information can be ignored for this feature

XGMII Datapath Width

32bit [✓ 64bit]

Optional Features

MDIO Management [ ]

Transceiver Options

DRP Clocking

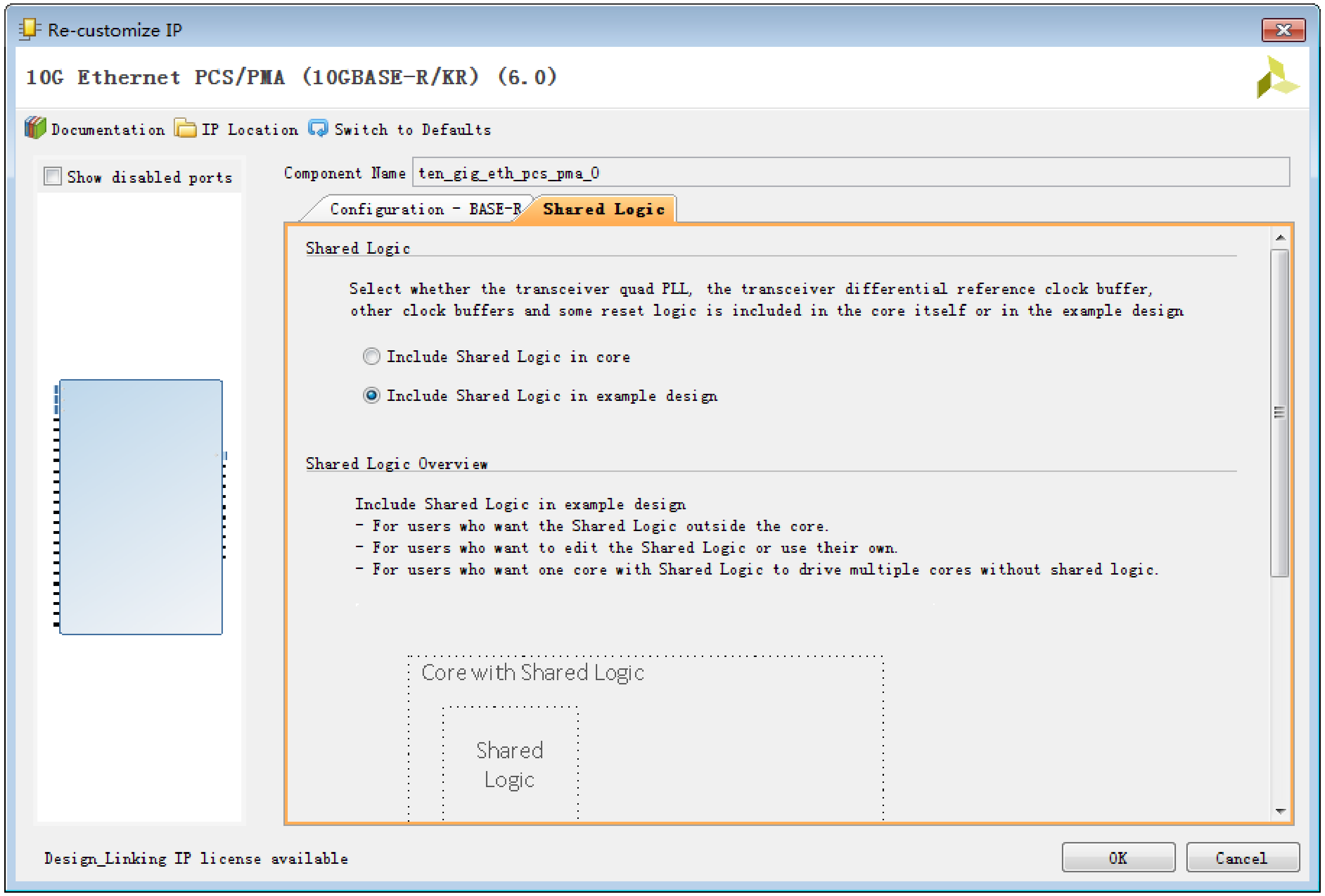
Frequency (MHz) 100.00 [50.00 - 175.00]

[✓] Additional transceiver control and status ports

Design_Linking IP license available

Generate Ten Gigabit Ethernet PCS/PMA Core

- In the “Shared Logic” tab, use the configuration as shown in the following, and then click “OK”.


Re-customize IP

10G Ethernet PCS/PMA (10GBASE-R/KR) (6.0)

Documentation IP Location Switch to Defaults

Show disabled ports

Component Name: ten_gig_eth_pcs_pma_0

Configuration - BASE-R | **Shared Logic**

Shared Logic

Select whether the transceiver quad PLL, the transceiver differential reference clock buffer, other clock buffers and some reset logic is included in the core itself or in the example design

( ) Include Shared Logic in core

(●) Include Shared Logic in example design

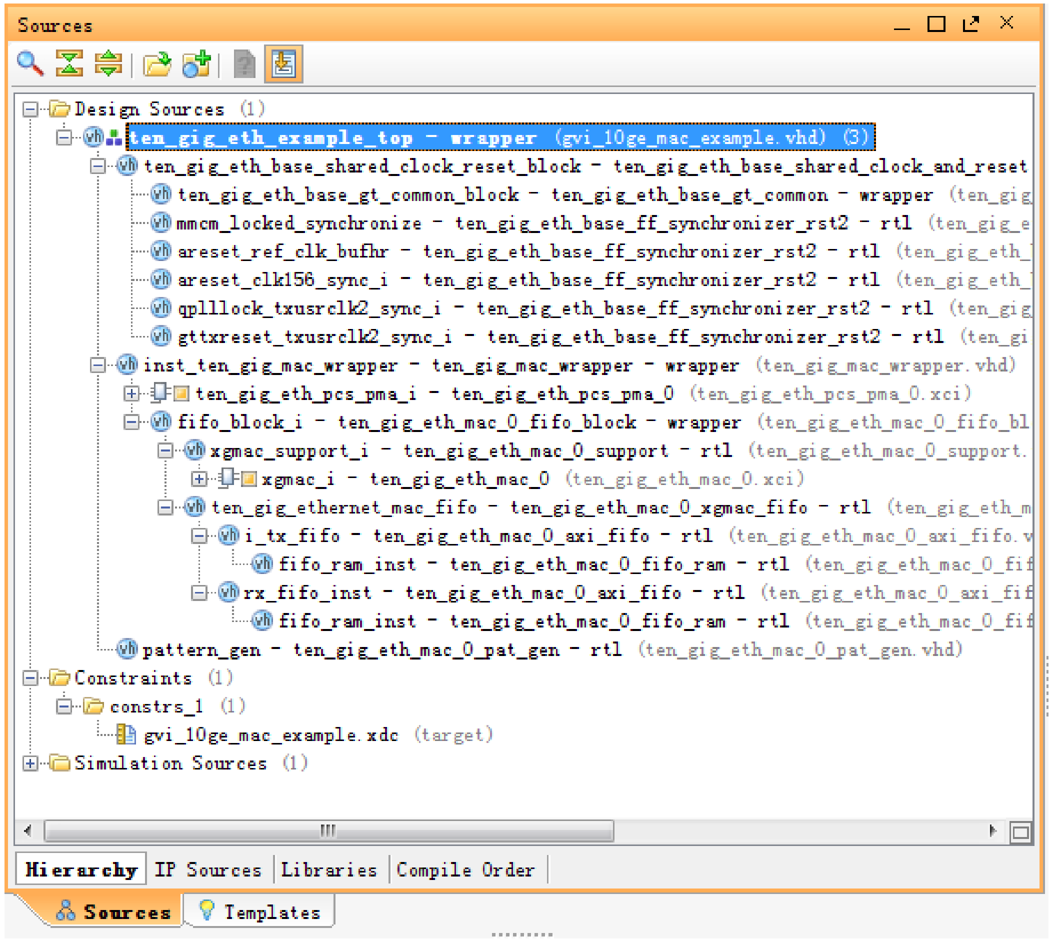
Shared Logic Overview

Include Shared Logic in example design
- For users who want the Shared Logic outside the core.
- For users who want to edit the Shared Logic or use their own.
- For users who want one core with Shared Logic to drive multiple cores without shared logic.

Design_Linking IP license available

Add Files to Vivado Project

- Add source files and constrain file into the Vivado project (the source files are available in GVI product CD)

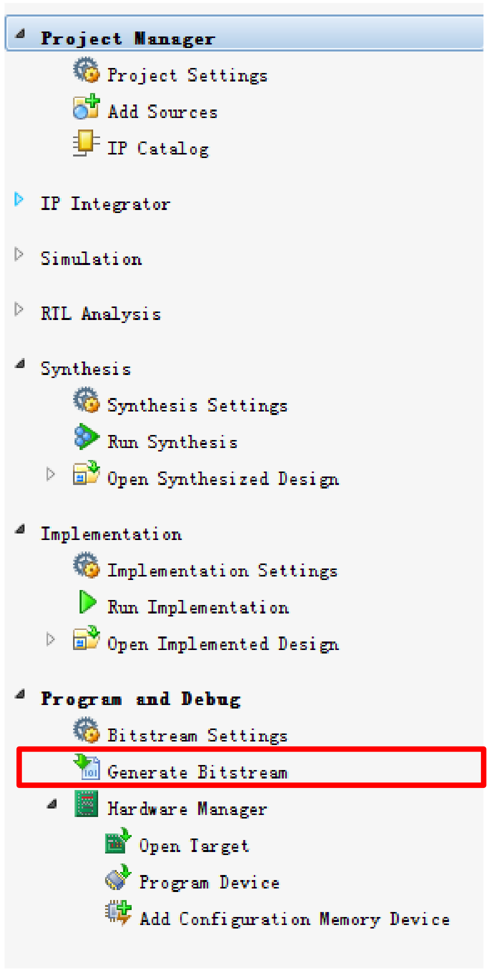


Sources

```
Design Sources (1)
  ten_gig_eth_example_top - wrapper (gvi_10ge_mac_example.vhd) (3)
    ten_gig_eth_base_shared_clock_reset_block - ten_gig_eth_base_shared_clock_and_reset
      ten_gig_eth_base_gt_common_block - ten_gig_eth_base_gt_common - wrapper (ten_gig
      mmcm_locked_synchronize - ten_gig_eth_base_ff_synchronizer_rst2 - rtl (ten_gig_e
      areset_ref_clk_bufhr - ten_gig_eth_base_ff_synchronizer_rst2 - rtl (ten_gig_eth_
      areset_clk156_sync_i - ten_gig_eth_base_ff_synchronizer_rst2 - rtl (ten_gig_eth_
      qplllock_txusrclk2_sync_i - ten_gig_eth_base_ff_synchronizer_rst2 - rtl (ten_gig
      gttxreset_txusrclk2_sync_i - ten_gig_eth_base_ff_synchronizer_rst2 - rtl (ten_gi
    inst_ten_gig_mac_wrapper - ten_gig_mac_wrapper - wrapper (ten_gig_mac_wrapper.vhd)
      ten_gig_eth_pcs_pma_i - ten_gig_eth_pcs_pma_0 (ten_gig_eth_pcs_pma_0.xci)
      fifo_block_i - ten_gig_eth_mac_0_fifo_block - wrapper (ten_gig_eth_mac_0_fifo_bl
        xgmac_support_i - ten_gig_eth_mac_0_support - rtl (ten_gig_eth_mac_0_support.
          xgmac_i - ten_gig_eth_mac_0 (ten_gig_eth_mac_0.xci)
        ten_gig_ethernet_mac_fifo - ten_gig_eth_mac_0_xgmac_fifo - rtl (ten_gig_eth_m
          i_tx_fifo - ten_gig_eth_mac_0_axi_fifo - rtl (ten_gig_eth_mac_0_axi_fifo.v
            fifo_ram_inst - ten_gig_eth_mac_0_fifo_ram - rtl (ten_gig_eth_mac_0_fif
          rx_fifo_inst - ten_gig_eth_mac_0_axi_fifo - rtl (ten_gig_eth_mac_0_axi_fif
            fifo_ram_inst - ten_gig_eth_mac_0_fifo_ram - rtl (ten_gig_eth_mac_0_fif
    pattern_gen - ten_gig_eth_mac_0_pat_gen - rtl (ten_gig_eth_mac_0_pat_gen.vhd)
Constraints (1)
  constrs_1 (1)
    gvi_10ge_mac_example.xdc (target)
Simulation Sources (1)
```

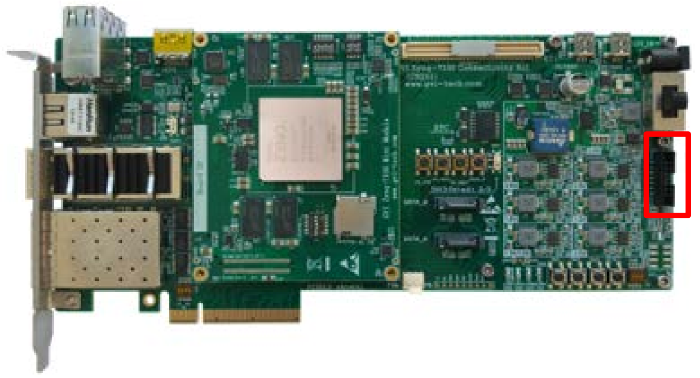
Compile the Demo Project

- Set “gvi_10ge_mac_example.vhd” to be the top-level design.
- Click “Generate Bitstream” to generate the FPGA configuration file.

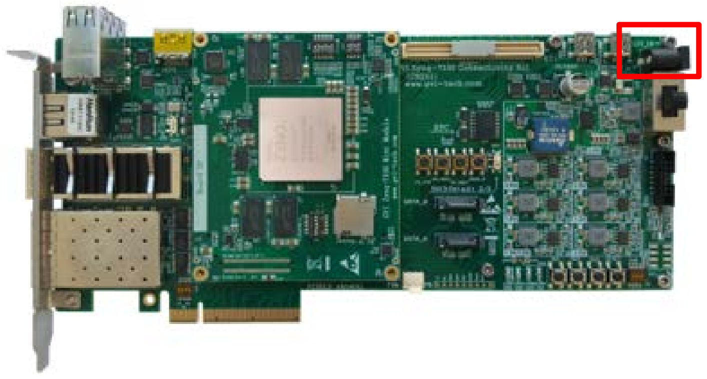


Hardware Setup

- Connect the JTAG cable to the base board.



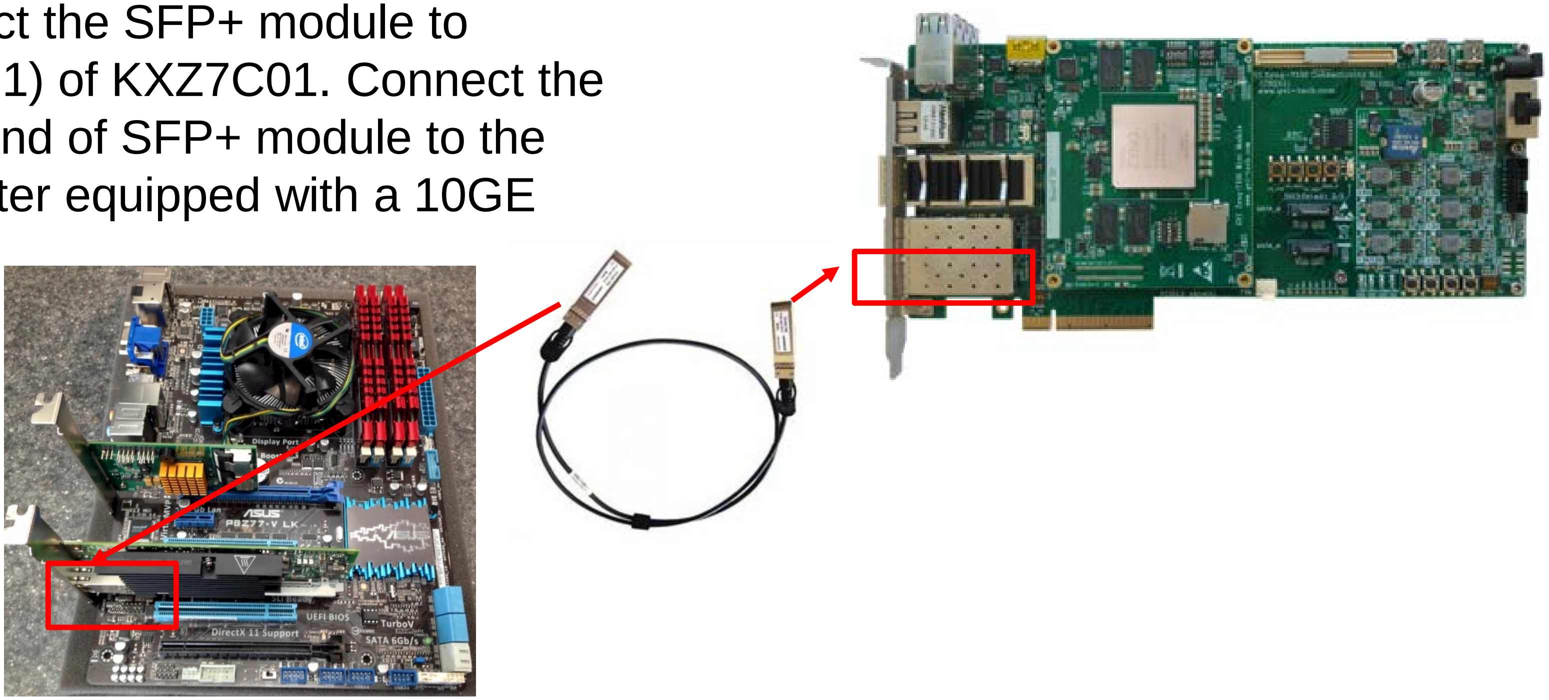
- Connect the power supplier.



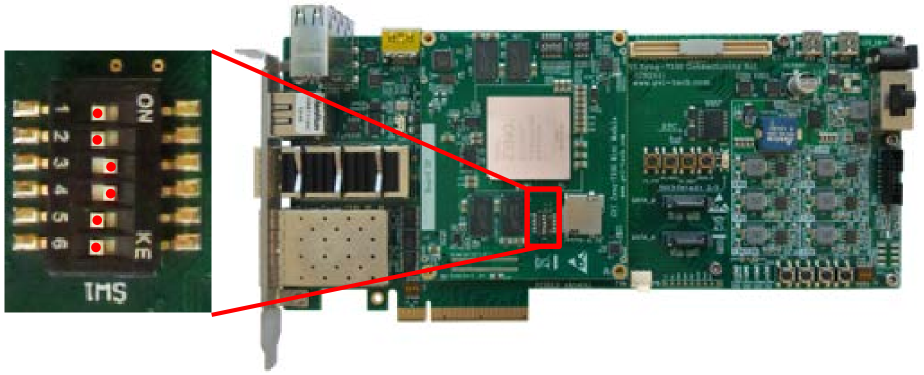
Order Hardware: https://detail.tmall.com/item.htm?id=562769965498

System Setup

- Connect the SFP+ module to (SFP+ 1) of KXZ7C01. Connect the other end of SFP+ module to the computer equipped with a 10GE NIC.

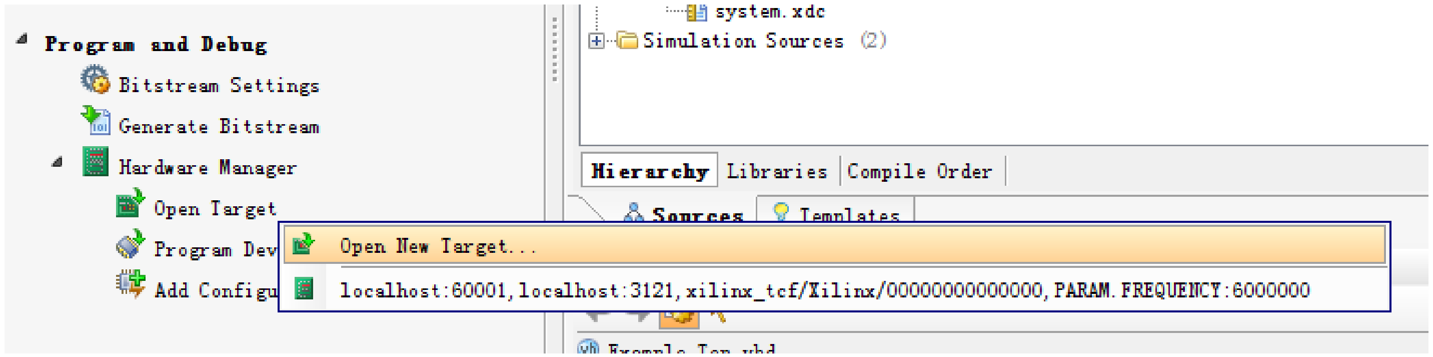


- Set SW1 to OFF-OFF-ON-ON-OFF-OFF. This sets the frequency to 156.25MHz.



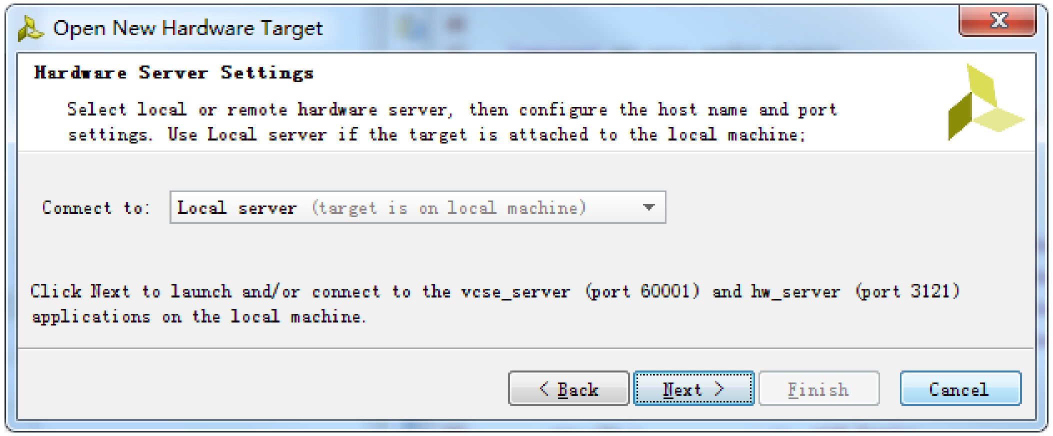
Configure FPGA

- Click “Open Target”, and then click “Open New Target…”



localhost:60001,localhost:3121,xilinx_tcf/Xilinx/00000000000000,PARAM.FREQUENCY:6000000

- In this dialog, choose “Local server”, and then click “Next”.



Hardware Server Settings

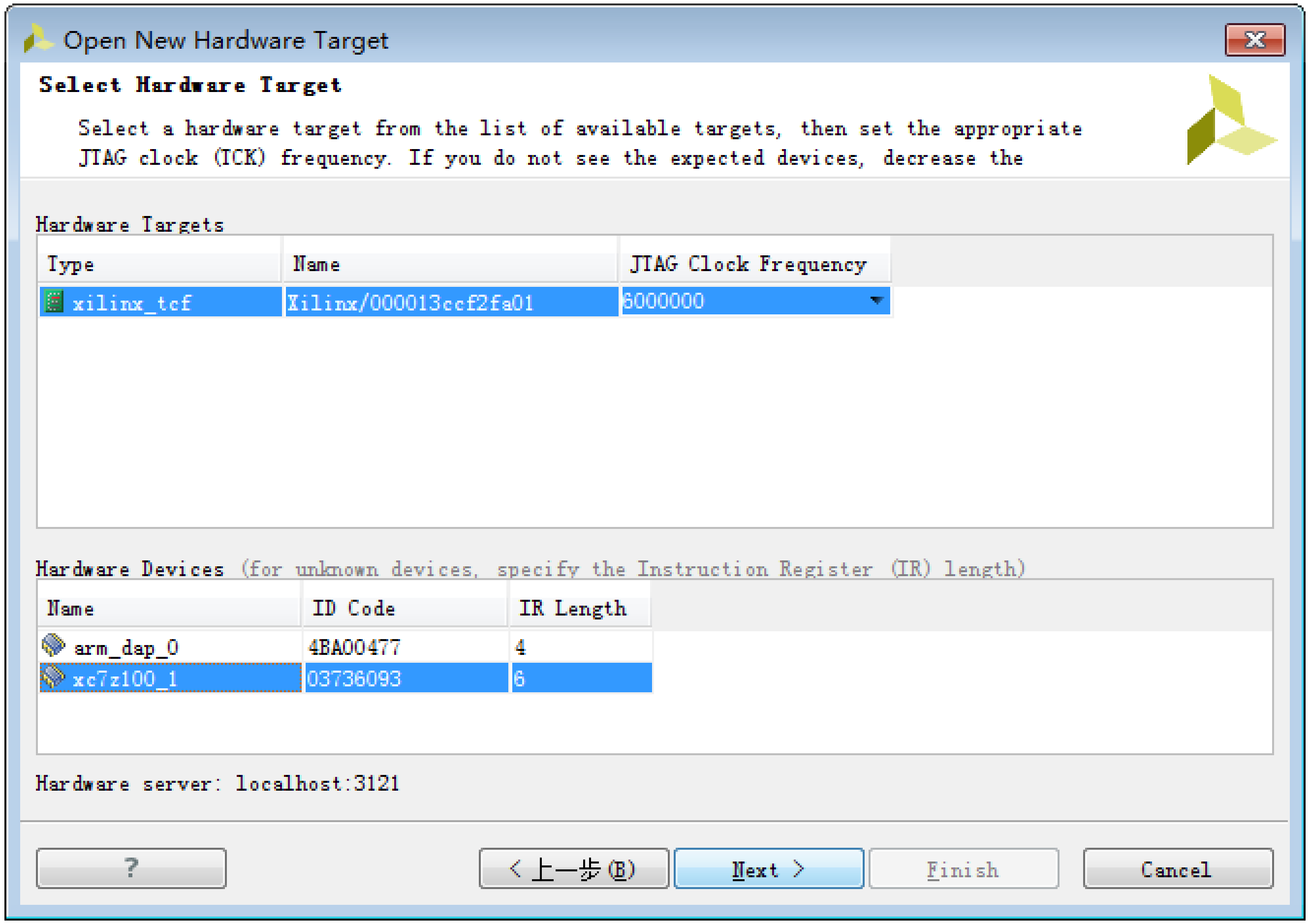
Select local or remote hardware server, then configure the host name and port settings. Use Local server if the target is attached to the local machine:

Connect to: Local server (target is on local machine)

Click Next to launch and/or connect to the vcse_server (port 60001) and hw_server (port 3121) applications on the local machine.

Configure FPGA

- Choose the target FPGA, and then click “Next”.



Open New Hardware Target

Select Hardware Target

Select a hardware target from the list of available targets, then set the appropriate JTAG clock (TCK) frequency. If you do not see the expected devices, decrease the

Hardware Targets

Type	Name	JTAG Clock Frequency
xilinx_tcf	Xilinx/000013ccf2fa01	6000000

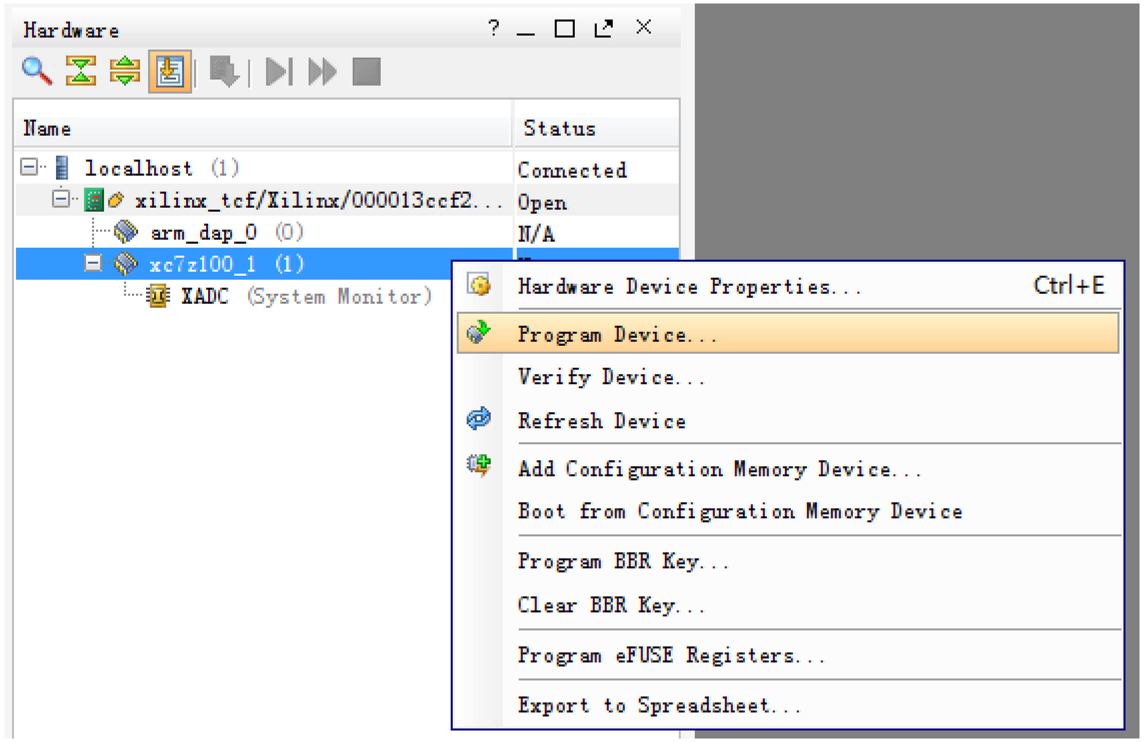
Hardware Devices (for unknown devices, specify the Instruction Register (IR) length)

Name	ID Code	IR Length
arm_dap_0	4BA00477	4
xc7z100_1	03736093	6

Hardware server: localhost:3121

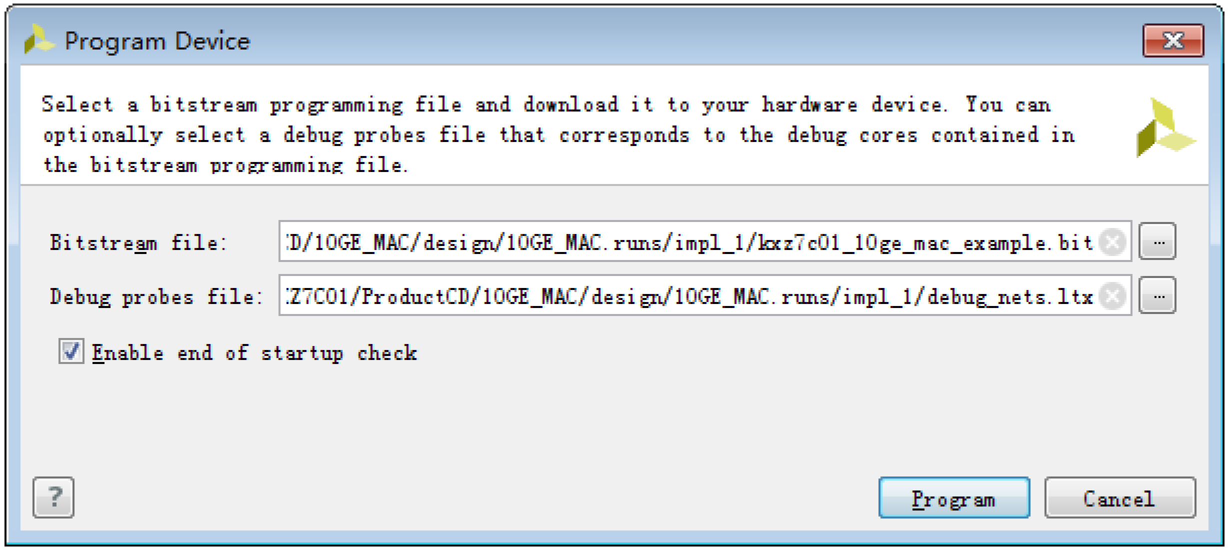
Configure FPGA

- Go to the “Hardware Manager” window.
- Right click on the FPGA, and then click “Program Device…”



Configure FPGA

- In this dialog, choose the right bit file, and continue with “Program”.



Program Device

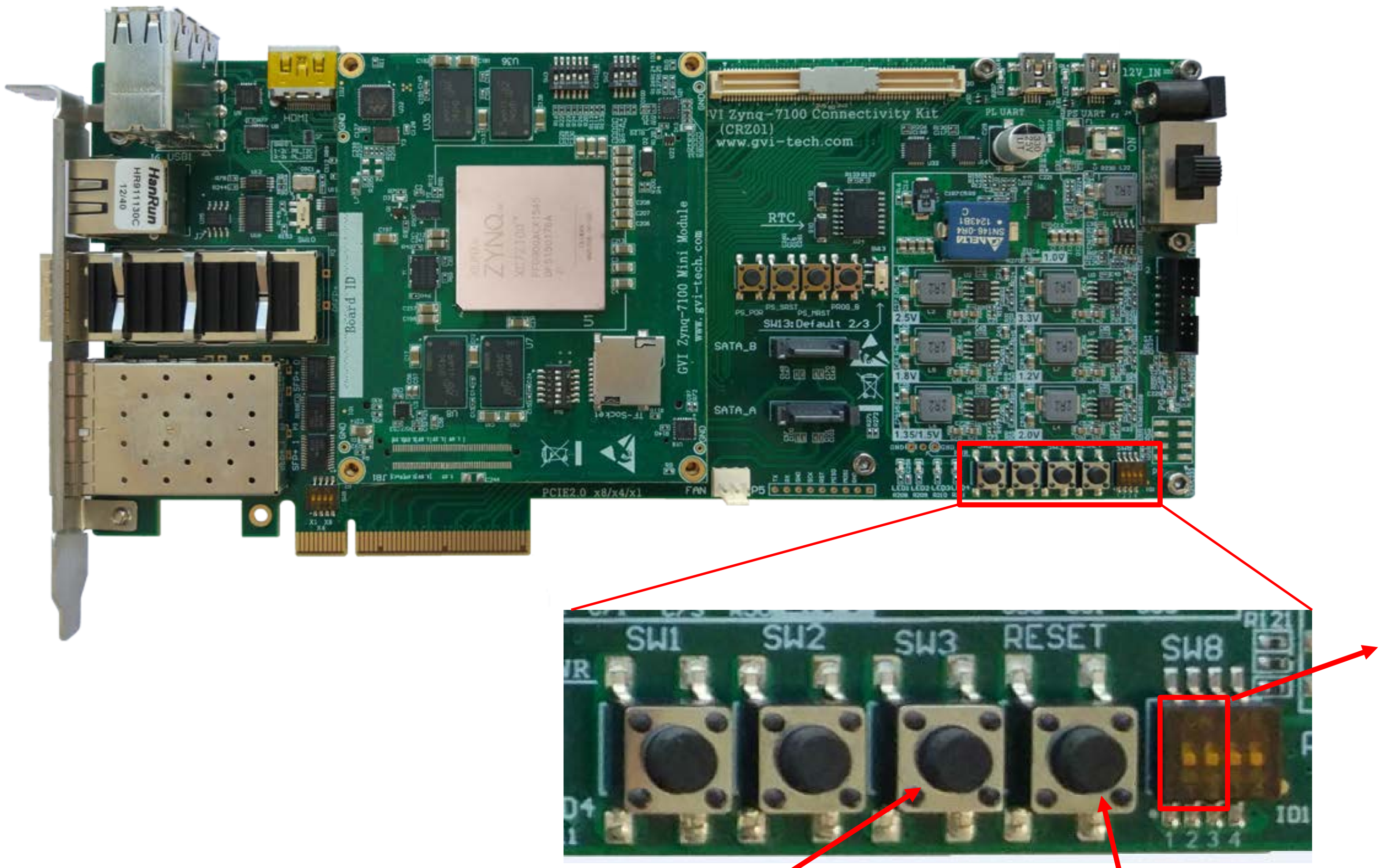
Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: D/10GE_MAC/design/10GE_MAC.runs/impl_1/kxz7c01_10ge_mac_example.bit

Debug probes file: Z7C01/ProductCD/10GE_MAC/design/10GE_MAC.runs/impl_1/debug_nets.ltx

☑ Enable end of startup check

How Does the Demo Work



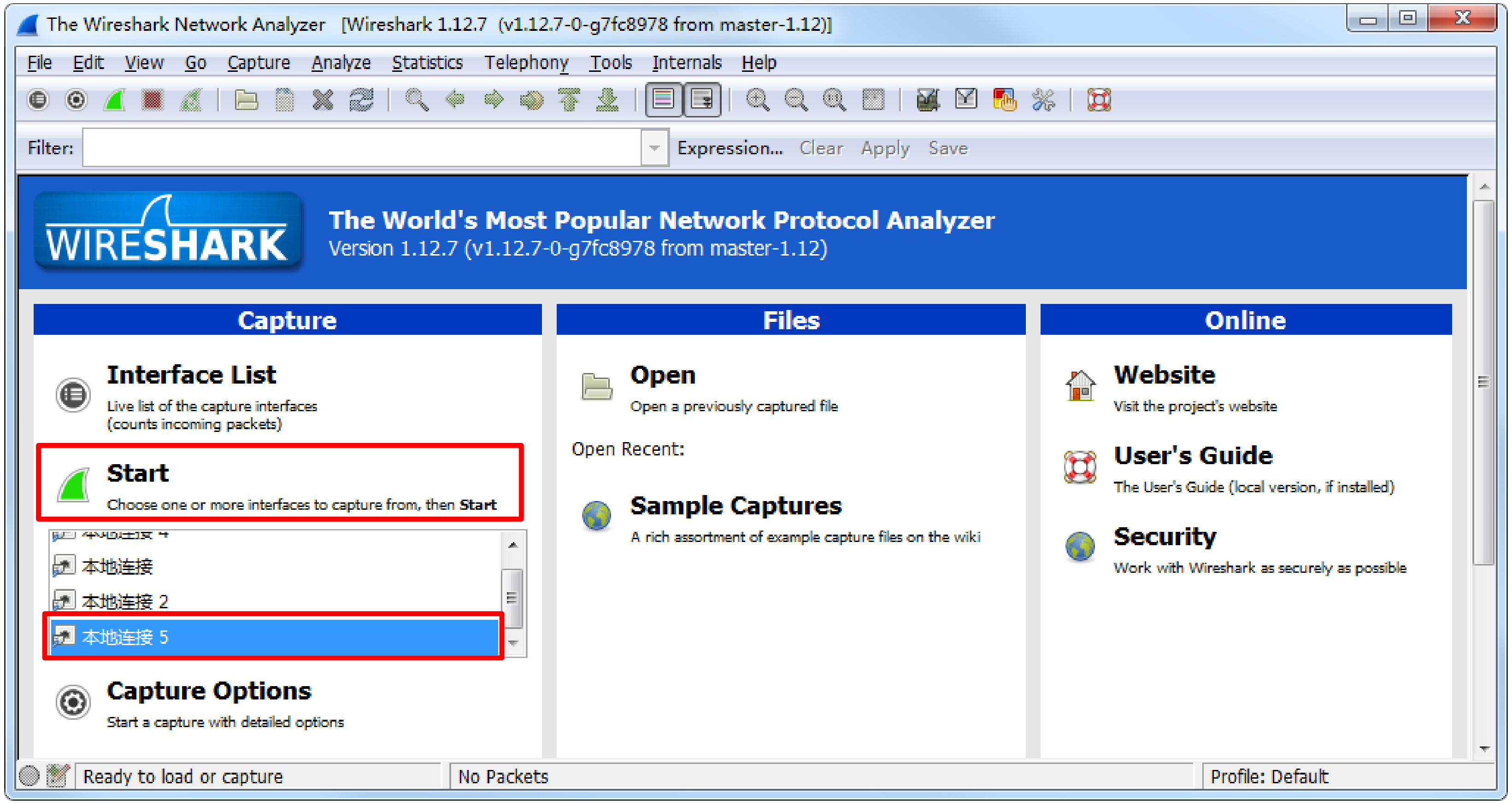
SW3: Press this button to send 100 packets to PC.

RESET: Press this button to reset the demo.

Switch Position	Packet Length
OFF OFF	64 Bytes
OFF ON	256 Bytes
ON OFF	1024 Bytes
ON ON	4096 Bytes

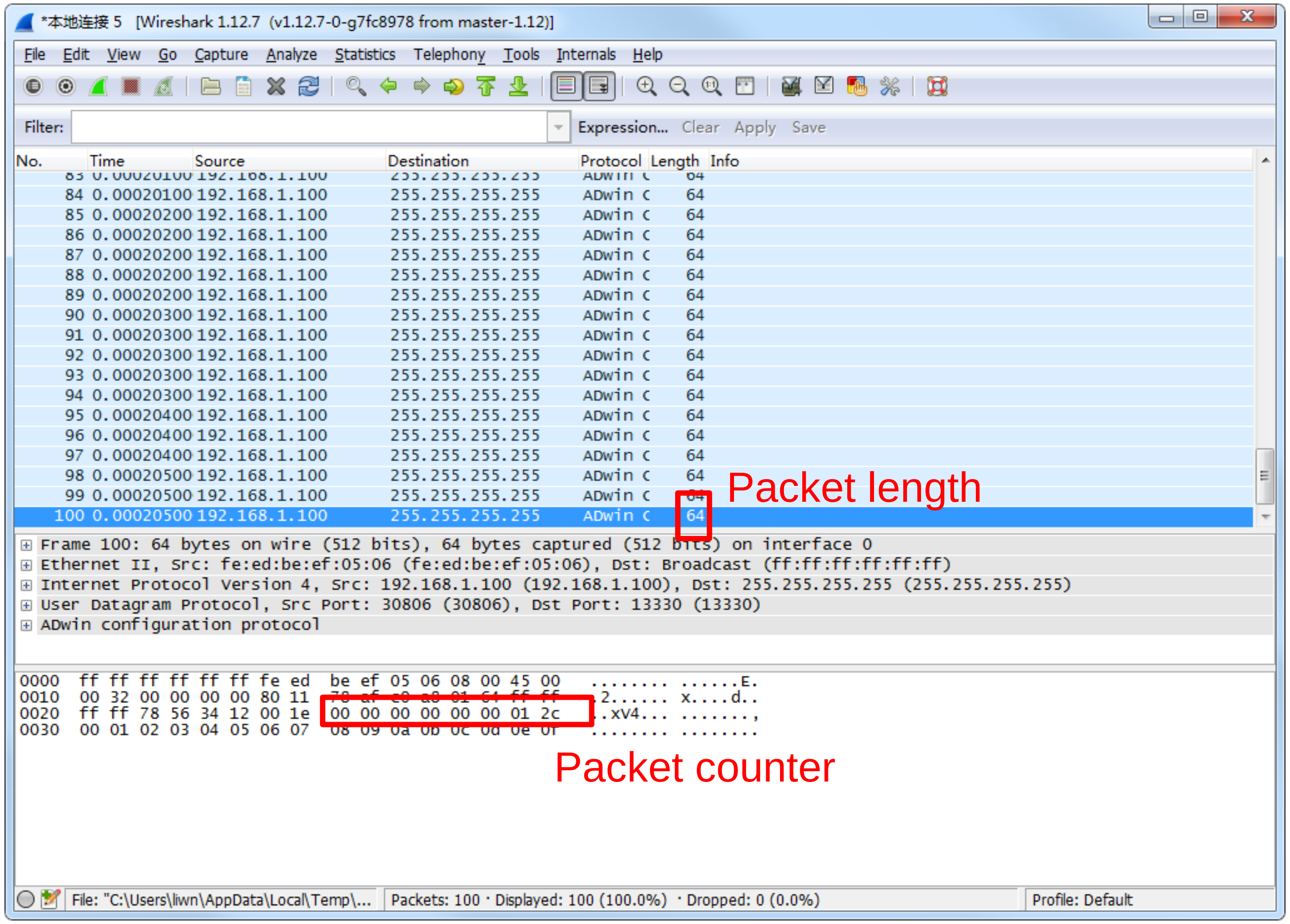
Running the 10GE MAC Demo

- Open Wireshark. Select the interface which is connected the KXZ7C01, and then click “Start”.


The Wireshark Network Analyzer [Wireshark 1.12.7 (v1.12.7-0-g7fc8978 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
WIRESHARK
The World's Most Popular Network Protocol Analyzer
Version 1.12.7 (v1.12.7-0-g7fc8978 from master-1.12)
Capture
Files
Online
Interface List
Live list of the capture interfaces (counts incoming packets)
Start
Choose one or more interfaces to capture from, then Start
本地连接
本地连接 2
本地连接 5
Capture Options
Start a capture with detailed options
Open
Open a previously captured file
Open Recent:
Sample Captures
A rich assortment of example capture files on the wiki
Website
Visit the project's website
User's Guide
The User's Guide (local version, if installed)
Security
Work with Wireshark as securely as possible
Ready to load or capture
No Packets
Profile: Default

Sending Packet From KXZ7C01 to PC

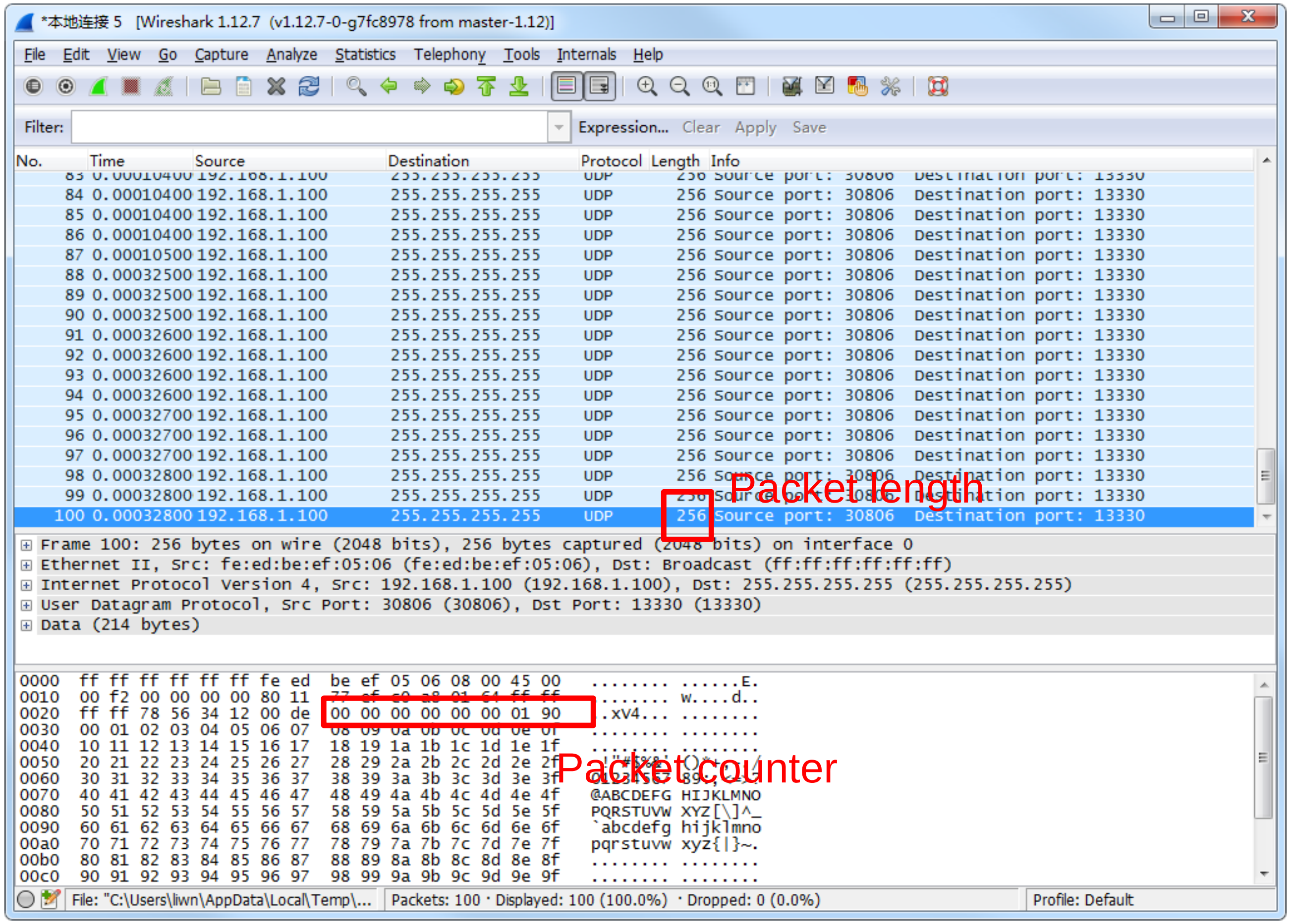
- Swtich SW8 to “OFF OFF”, and then press SW3. In Wireshark, check the packets sent from KXZ7C01 to PC.


*本地连接 5 [Wireshark 1.12.7 (v1.12.7-0-g7fc8978 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
Packet length
Packet Content
Packet counter

```
No.    Time        Source          Destination      Protocol Length Info
 83 0.00020100 192.168.1.100   255.255.255.255  ADwin C   64
 84 0.00020100 192.168.1.100   255.255.255.255  ADwin C   64
 85 0.00020200 192.168.1.100   255.255.255.255  ADwin C   64
 86 0.00020200 192.168.1.100   255.255.255.255  ADwin C   64
 87 0.00020200 192.168.1.100   255.255.255.255  ADwin C   64
 88 0.00020200 192.168.1.100   255.255.255.255  ADwin C   64
 89 0.00020200 192.168.1.100   255.255.255.255  ADwin C   64
 90 0.00020300 192.168.1.100   255.255.255.255  ADwin C   64
 91 0.00020300 192.168.1.100   255.255.255.255  ADwin C   64
 92 0.00020300 192.168.1.100   255.255.255.255  ADwin C   64
 93 0.00020300 192.168.1.100   255.255.255.255  ADwin C   64
 94 0.00020300 192.168.1.100   255.255.255.255  ADwin C   64
 95 0.00020400 192.168.1.100   255.255.255.255  ADwin C   64
 96 0.00020400 192.168.1.100   255.255.255.255  ADwin C   64
 97 0.00020400 192.168.1.100   255.255.255.255  ADwin C   64
 98 0.00020500 192.168.1.100   255.255.255.255  ADwin C   64
 99 0.00020500 192.168.1.100   255.255.255.255  ADwin C   64
100 0.00020500 192.168.1.100   255.255.255.255  ADwin C   64

⊞ Frame 100: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
⊞ Ethernet II, Src: fe:ed:be:ef:05:06 (fe:ed:be:ef:05:06), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Internet Protocol Version 4, Src: 192.168.1.100 (192.168.1.100), Dst: 255.255.255.255 (255.255.255.255)
⊞ User Datagram Protocol, Src Port: 30806 (30806), Dst Port: 13330 (13330)
⊞ ADwin configuration protocol

0000  ff ff ff ff ff ff fe ed  be ef 05 06 08 00 45 00   ........ ......E.
0010  00 32 00 00 00 00 80 11  78 af c0 a8 01 64 ff ff   .2...... x....d..
0020  ff ff 78 56 34 12 00 1e  00 00 00 00 00 00 01 2c   ..xV4... .......,
0030  00 01 02 03 04 05 06 07  08 09 0a 0b 0c 0d 0e 0f   ........ ........

File: "C:\Users\liwn\AppData\Local\Temp\...  Packets: 100 · Displayed: 100 (100.0%) · Dropped: 0 (0.0%)  Profile: Default
```

Sending Packet From KXZ7C01 to PC

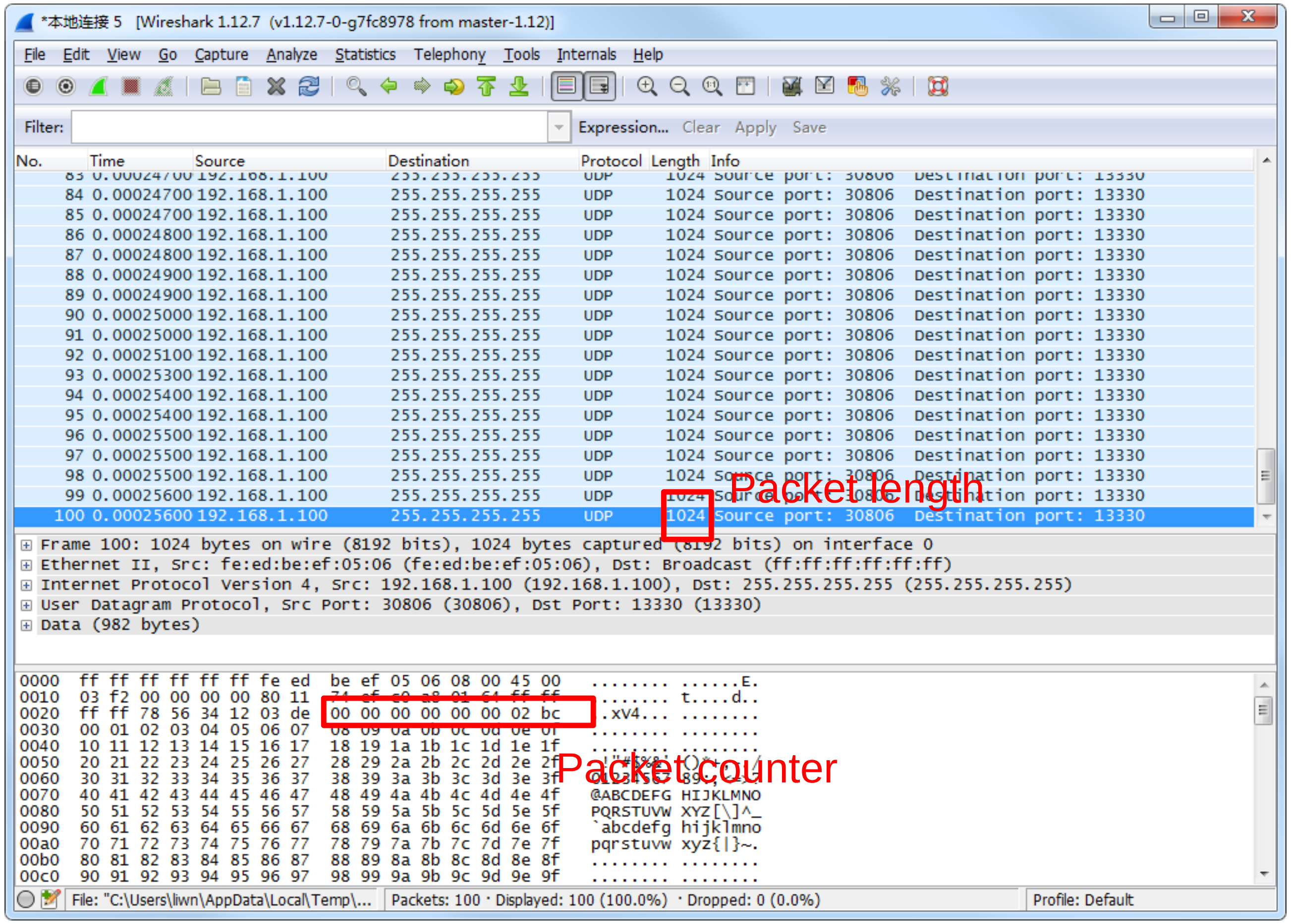
- Swtich SW8 to “OFF ON”, and then press SW3. In Wireshark, check the packets sent from KXZ7C01 to PC.



```
*本地连接 5   [Wireshark 1.12.7  (v1.12.7-0-g7fc8978 from master-1.12)]
File  Edit  View  Go  Capture  Analyze  Statistics  Telephony  Tools  Internals  Help
Filter:                                              Expression...  Clear  Apply  Save

No.   Time         Source          Destination       Protocol Length Info
   83 0.00010400   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   84 0.00010400   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   85 0.00010400   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   86 0.00010400   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   87 0.00010500   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   88 0.00032500   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   89 0.00032500   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   90 0.00032500   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   91 0.00032600   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   92 0.00032600   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   93 0.00032600   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   94 0.00032600   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   95 0.00032700   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   96 0.00032700   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   97 0.00032700   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   98 0.00032800   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
   99 0.00032800   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330
  100 0.00032800   192.168.1.100   255.255.255.255   UDP      256 Source port: 30806  Destination port: 13330

⊞ Frame 100: 256 bytes on wire (2048 bits), 256 bytes captured (2048 bits) on interface 0
⊞ Ethernet II, Src: fe:ed:be:ef:05:06 (fe:ed:be:ef:05:06), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Internet Protocol Version 4, Src: 192.168.1.100 (192.168.1.100), Dst: 255.255.255.255 (255.255.255.255)
⊞ User Datagram Protocol, Src Port: 30806 (30806), Dst Port: 13330 (13330)
⊞ Data (214 bytes)

0000  ff ff ff ff ff ff fe ed  be ef 05 06 08 00 45 00   ........ ......E.
0010  00 f2 00 00 00 00 80 11  77 ef c0 a8 01 64 ff ff   ........ w....d..
0020  ff ff 78 56 34 12 00 de  00 00 00 00 00 00 01 90   ..xV4... ........
0030  00 01 02 03 04 05 06 07  08 09 0a 0b 0c 0d 0e 0f   ........ ........
0040  10 11 12 13 14 15 16 17  18 19 1a 1b 1c 1d 1e 1f   ........ ........
0050  20 21 22 23 24 25 26 27  28 29 2a 2b 2c 2d 2e 2f    !"#$%&' ()*+,-./
0060  30 31 32 33 34 35 36 37  38 39 3a 3b 3c 3d 3e 3f   01234567 89:;<=>?
0070  40 41 42 43 44 45 46 47  48 49 4a 4b 4c 4d 4e 4f   @ABCDEFG HIJKLMNO
0080  50 51 52 53 54 55 56 57  58 59 5a 5b 5c 5d 5e 5f   PQRSTUVW XYZ[\]^_
0090  60 61 62 63 64 65 66 67  68 69 6a 6b 6c 6d 6e 6f   `abcdefg hijklmno
00a0  70 71 72 73 74 75 76 77  78 79 7a 7b 7c 7d 7e 7f   pqrstuvw xyz{|}~.
00b0  80 81 82 83 84 85 86 87  88 89 8a 8b 8c 8d 8e 8f   ........ ........
00c0  90 91 92 93 94 95 96 97  98 99 9a 9b 9c 9d 9e 9f   ........ ........

File: "C:\Users\liwn\AppData\Local\Temp\...   Packets: 100 · Displayed: 100 (100.0%) · Dropped: 0 (0.0%)   Profile: Default
```

Sending Packet From KXZ7C01 to PC

- Swtich SW8 to “ON OFF”, and then press SW3. In Wireshark, check the packets sent from KXZ7C01 to PC.


*本地连接 5 [Wireshark 1.12.7 (v1.12.7-0-g7fc8978 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
Packet length
Packet counter

```
No.  Time        Source          Destination       Protocol Length Info
  83 0.00024700 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  84 0.00024700 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  85 0.00024700 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  86 0.00024800 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  87 0.00024800 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  88 0.00024900 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  89 0.00024900 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  90 0.00025000 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  91 0.00025000 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  92 0.00025100 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  93 0.00025300 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  94 0.00025400 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  95 0.00025400 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  96 0.00025500 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  97 0.00025500 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  98 0.00025500 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
  99 0.00025600 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
 100 0.00025600 192.168.1.100   255.255.255.255   UDP      1024 Source port: 30806  Destination port: 13330
```

```
⊞ Frame 100: 1024 bytes on wire (8192 bits), 1024 bytes captured (8192 bits) on interface 0
⊞ Ethernet II, Src: fe:ed:be:ef:05:06 (fe:ed:be:ef:05:06), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Internet Protocol Version 4, Src: 192.168.1.100 (192.168.1.100), Dst: 255.255.255.255 (255.255.255.255)
⊞ User Datagram Protocol, Src Port: 30806 (30806), Dst Port: 13330 (13330)
⊞ Data (982 bytes)
```

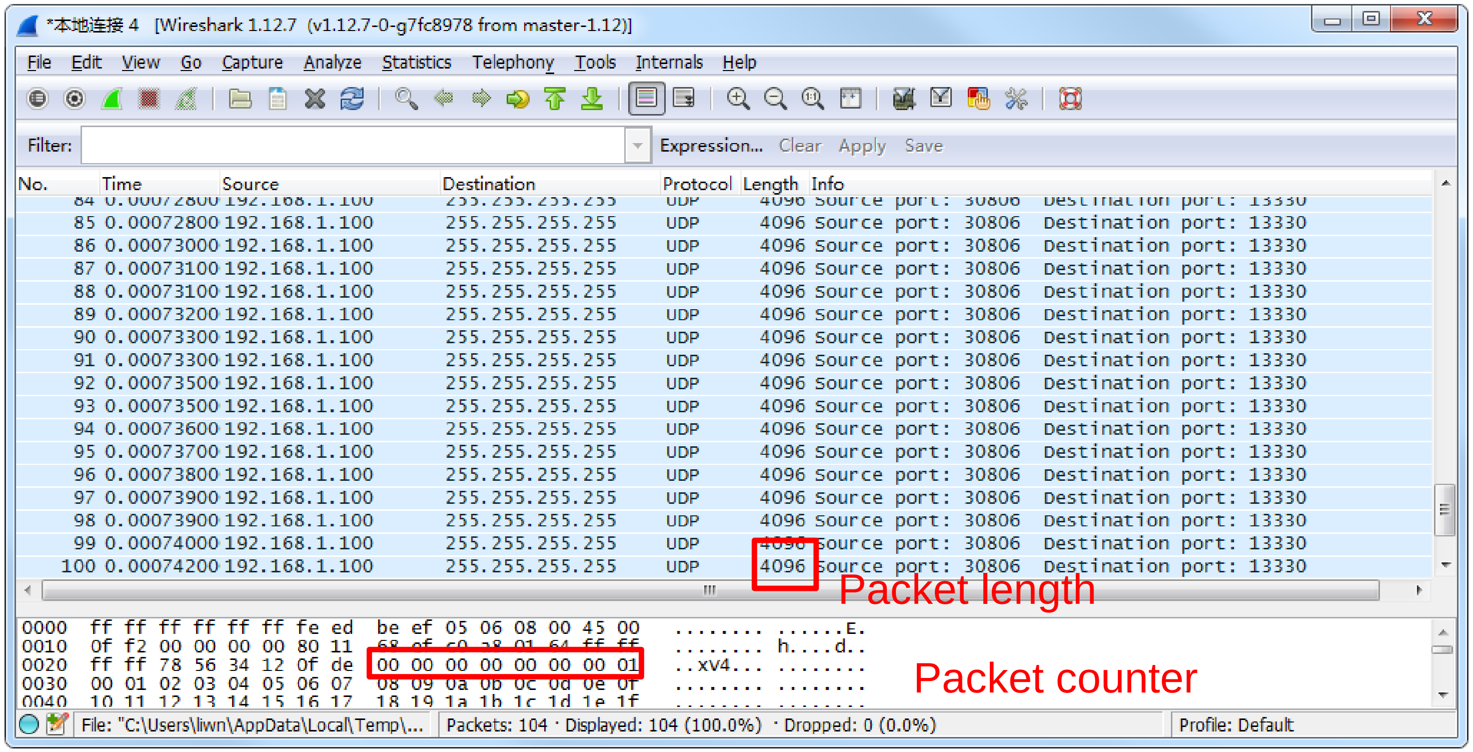
Packet Content

```
0000  ff ff ff ff ff ff fe ed  be ef 05 06 08 00 45 00   ........ ......E.
0010  03 f2 00 00 00 00 80 11  74 ef c0 a8 01 64 ff ff   ........ t....d..
0020  ff ff 78 56 34 12 03 de  00 00 00 00 00 00 02 bc   ..xV4... ........
0030  00 01 02 03 04 05 06 07  08 09 0a 0b 0c 0d 0e 0f   ........ ........
0040  10 11 12 13 14 15 16 17  18 19 1a 1b 1c 1d 1e 1f   ........ ........
0050  20 21 22 23 24 25 26 27  28 29 2a 2b 2c 2d 2e 2f    !"#$%&' ()*+,-./
0060  30 31 32 33 34 35 36 37  38 39 3a 3b 3c 3d 3e 3f   01234567 89:;<=>?
0070  40 41 42 43 44 45 46 47  48 49 4a 4b 4c 4d 4e 4f   @ABCDEFG HIJKLMNO
0080  50 51 52 53 54 55 56 57  58 59 5a 5b 5c 5d 5e 5f   PQRSTUVW XYZ[\]^_
0090  60 61 62 63 64 65 66 67  68 69 6a 6b 6c 6d 6e 6f   `abcdefg hijklmno
00a0  70 71 72 73 74 75 76 77  78 79 7a 7b 7c 7d 7e 7f   pqrstuvw xyz{|}~.
00b0  80 81 82 83 84 85 86 87  88 89 8a 8b 8c 8d 8e 8f   ........ ........
00c0  90 91 92 93 94 95 96 97  98 99 9a 9b 9c 9d 9e 9f   ........ ........
```

File: "C:\Users\liwn\AppData\Local\Temp\... Packets: 100 · Displayed: 100 (100.0%) · Dropped: 0 (0.0%) Profile: Default

Sending Packet From KXZ7C01 to PC

- Swtich SW8 to “ON ON”, and then press SW3. In Wireshark, check the packets sent from KXZ7C01 to PC.
- Note: in order to receive packet larger than 1514, make sure your 10GE NIC support Jumbo Frame.


*本地连接 4 [Wireshark 1.12.7 (v1.12.7-0-g7fc8978 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
Packet length
Packet counter
Packet
Content

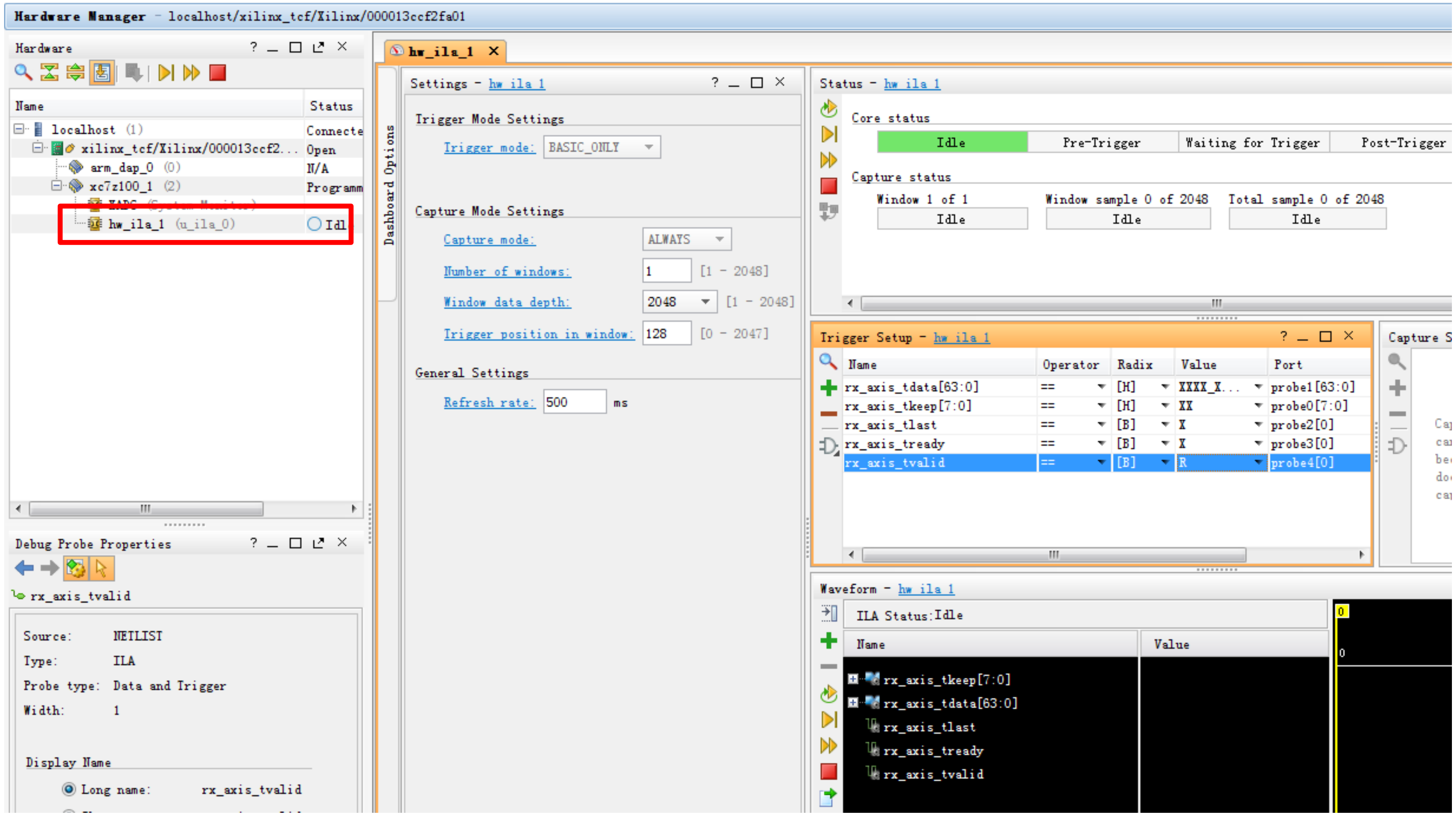
```
No.   Time        Source         Destination      Protocol Length Info
   84 0.00072800 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   85 0.00072800 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   86 0.00073000 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   87 0.00073100 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   88 0.00073100 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   89 0.00073200 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   90 0.00073300 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   91 0.00073300 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   92 0.00073500 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   93 0.00073500 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   94 0.00073600 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   95 0.00073700 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   96 0.00073800 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   97 0.00073900 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   98 0.00073900 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
   99 0.00074000 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
  100 0.00074200 192.168.1.100  255.255.255.255  UDP      4096 Source port: 30806  Destination port: 13330
```

```
0000  ff ff ff ff ff ff fe ed  be ef 05 06 08 00 45 00   ........ ......E.
0010  0f f2 00 00 00 00 80 11  68 ef c0 a8 01 64 ff ff   ........ h....d..
0020  ff ff 78 56 34 12 0f de  00 00 00 00 00 00 00 01   ..xV4... ........
0030  00 01 02 03 04 05 06 07  08 09 0a 0b 0c 0d 0e 0f   ........ ........
0040  10 11 12 13 14 15 16 17  18 19 1a 1b 1c 1d 1e 1f   ........ ........
```

File: "C:\Users\liwn\AppData\Local\Temp\... Packets: 104 · Displayed: 104 (100.0%) · Dropped: 0 (0.0%) Profile: Default

Sending Packet From PC to KXZ7C01

- In the “Hardware Manager”, double click the “hw_ila_1(ILA)” to open the Chipscope analyzer.



Hardware Manager - localhost/xilinx_tcf/Xilinx/000013ccf2fa01

Hardware

Name	Status
localhost (1)	Connecte
xilinx_tcf/Xilinx/000013ccf2...	Open
arm_dap_0 (0)	N/A
xc7z100_1 (2)	Programm
XADC (System Monitor)	
hw_ila_1 (u_ila_0)	Idl

Debug Probe Properties

rx_axis_tvalid

Source: NETLIST
Type: ILA
Probe type: Data and Trigger
Width: 1

Display Name

Long name: rx_axis_tvalid

hw_ila_1

Settings - hw_ila_1

Trigger Mode Settings

Trigger mode: BASIC_ONLY

Capture Mode Settings

Capture mode: ALWAYS
Number of windows: 1 [1 - 2048]
Window data depth: 2048 [1 - 2048]
Trigger position in window: 128 [0 - 2047]

General Settings

Refresh rate: 500 ms

Status - hw_ila_1

Core status

Idle	Pre-Trigger	Waiting for Trigger	Post-Trigger

Capture status

Window 1 of 1	Window sample 0 of 2048	Total sample 0 of 2048
Idle	Idle	Idle

Trigger Setup - hw_ila_1

Name	Operator	Radix	Value	Port
rx_axis_tdata[63:0]	==	[H]	XXXX_X...	probe1[63:0]
rx_axis_tkeep[7:0]	==	[H]	XX	probe0[7:0]
rx_axis_tlast	==	[B]	X	probe2[0]
rx_axis_tready	==	[B]	X	probe3[0]
rx_axis_tvalid	==	[B]	R	probe4[0]

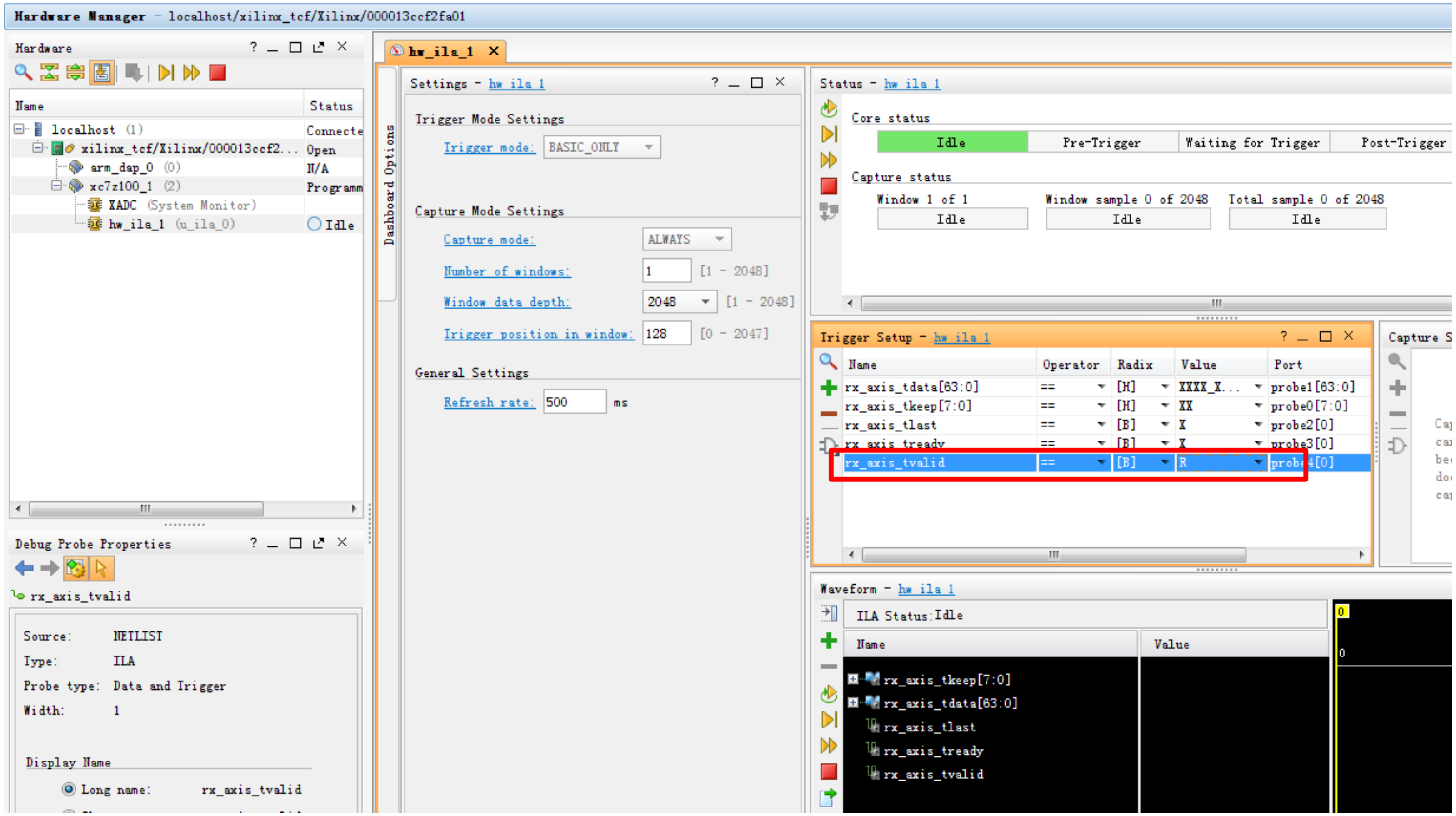
Waveform - hw_ila_1

ILA Status: Idle

Name	Value
rx_axis_tkeep[7:0]	
rx_axis_tdata[63:0]	
rx_axis_tlast	
rx_axis_tready	
rx_axis_tvalid	

Sending Packet From PC to KXZ7C01

- In the “Basic Trigger Setup”, change the value of rx_axis_tvalid to ‘R’.
- Set the “Trigger position in window” to 128



Hardware Manager - localhost/xilinx_tcf/Xilinx/000013ccf2fa01

Hardware

Name	Status
localhost (1)	Connecte
xilinx_tcf/Xilinx/000013ccf2...	Open
arm_dap_0 (0)	N/A
xc7z100_1 (2)	Programm
XADC (System Monitor)	
hw_ila_1 (u_ila_0)	Idle

Debug Probe Properties

rx_axis_tvalid

Source: NETLIST
Type: ILA
Probe type: Data and Trigger
Width: 1

Display Name

Long name: rx_axis_tvalid

hw_ila_1

Settings - hw_ila_1

Trigger Mode Settings

Trigger mode: BASIC_ONLY

Capture Mode Settings

Capture mode: ALWAYS
Number of windows: 1 [1 - 2048]
Window data depth: 2048 [1 - 2048]
Trigger position in window: 128 [0 - 2047]

General Settings

Refresh rate: 500 ms

Status - hw_ila_1

Core status

Idle	Pre-Trigger	Waiting for Trigger	Post-Trigger

Capture status

Window 1 of 1	Window sample 0 of 2048	Total sample 0 of 2048
Idle	Idle	Idle

Trigger Setup - hw_ila_1

Name	Operator	Radix	Value	Port
rx_axis_tdata[63:0]	==	[H]	XXXX_X...	probe1[63:0]
rx_axis_tkeep[7:0]	==	[H]	XX	probe0[7:0]
rx_axis_tlast	==	[B]	X	probe2[0]
rx_axis_tready	==	[B]	X	probe3[0]
rx_axis_tvalid	==	[B]	R	probe4[0]

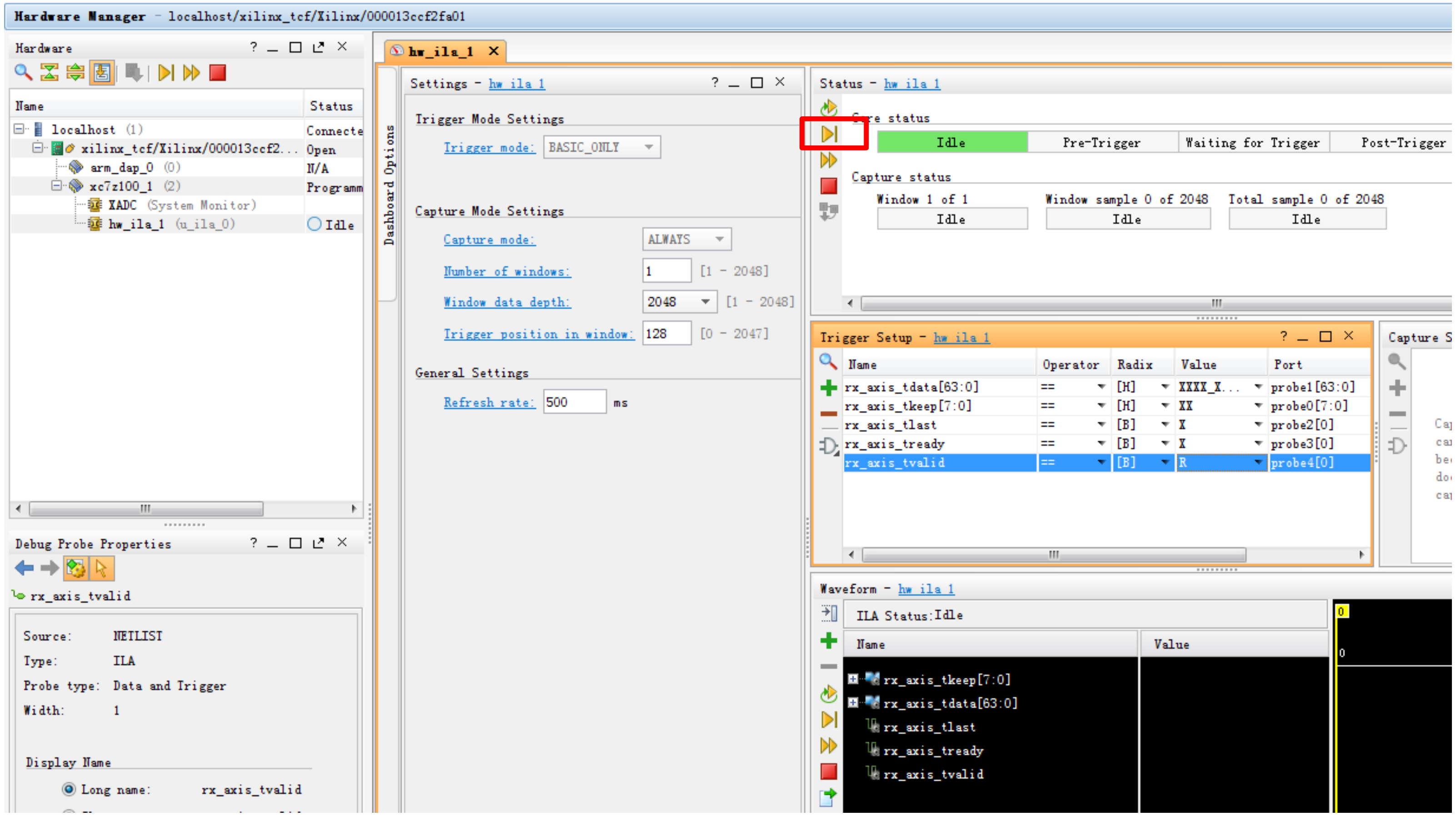
Waveform - hw_ila_1

ILA Status: Idle

Name	Value
rx_axis_tkeep[7:0]	
rx_axis_tdata[63:0]	
rx_axis_tlast	
rx_axis_tready	
rx_axis_tvalid	

Sending Packet From PC to KXK7TC01

- Click “Run trigger of this ILA core” to start capturing.



Hardware Manager - localhost/xilinx_tcf/Xilinx/000013ccf2fa01

Hardware

Name	Status
localhost (1)	Connecte
xilinx_tcf/Xilinx/000013ccf2...	Open
arm_dap_0 (0)	N/A
xc7z100_1 (2)	Programm
XADC (System Monitor)	
hw_ila_1 (u_ila_0)	Idle

Debug Probe Properties

rx_axis_tvalid

Source: NETLIST
Type: ILA
Probe type: Data and Trigger
Width: 1

Display Name
Long name: rx_axis_tvalid

hw_ila_1

Settings - hw_ila_1

Trigger Mode Settings
Trigger mode: BASIC_ONLY

Capture Mode Settings
Capture mode: ALWAYS
Number of windows: 1 [1 - 2048]
Window data depth: 2048 [1 - 2048]
Trigger position in window: 128 [0 - 2047]

General Settings
Refresh rate: 500 ms

Status - hw_ila_1

Core status
Idle | Pre-Trigger | Waiting for Trigger | Post-Trigger

Capture status
Window 1 of 1: Idle
Window sample 0 of 2048: Idle
Total sample 0 of 2048: Idle

Trigger Setup - hw_ila_1

Name	Operator	Radix	Value	Port
rx_axis_tdata[63:0]	==	[H]	XXXX_X...	probe1[63:0]
rx_axis_tkeep[7:0]	==	[H]	XX	probe0[7:0]
rx_axis_tlast	==	[B]	X	probe2[0]
rx_axis_tready	==	[B]	X	probe3[0]
rx_axis_tvalid	==	[B]	R	probe4[0]

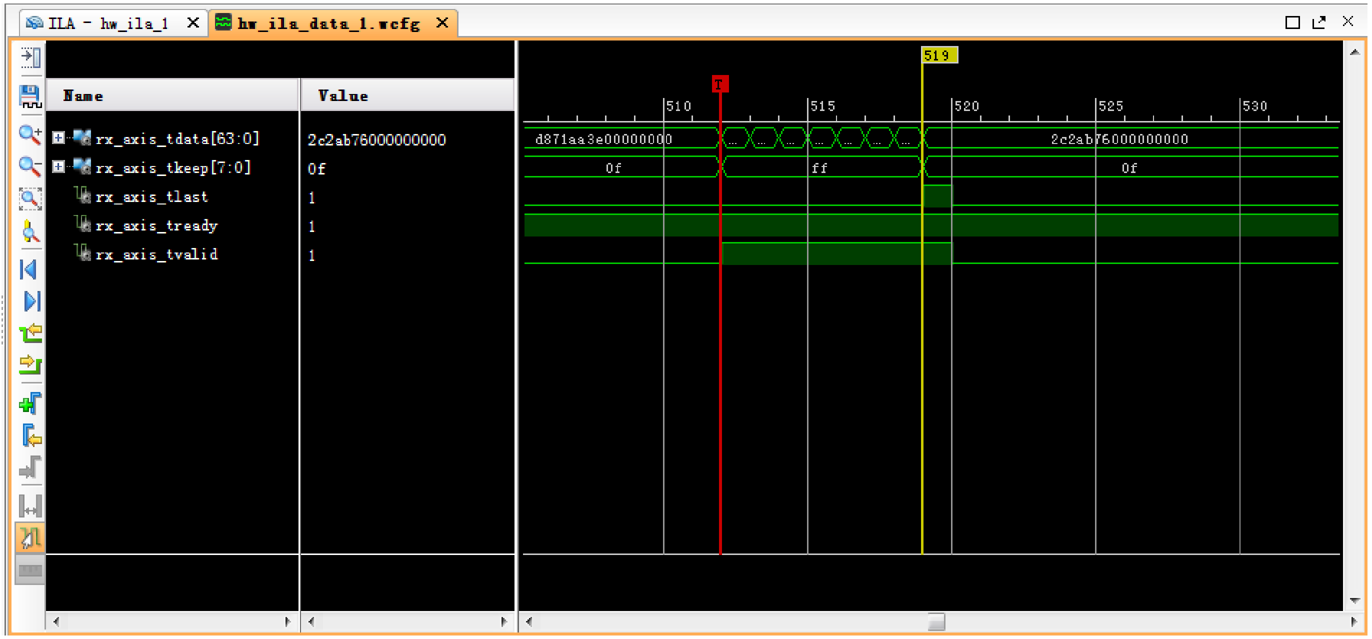
Waveform - hw_ila_1

ILA Status: Idle

Name	Value
rx_axis_tkeep[7:0]	
rx_axis_tdata[63:0]	
rx_axis_tlast	
rx_axis_tready	
rx_axis_tvalid	

Sending Packet From PC to KXK7TC01

- Wait until the core is triggered. Usually the PC sends some packets automatically from time to time. If your PC does not send packets automatically, you can also use some debugging tool to send a packet manually.
- When the ILA core is triggered, open the waveform viewer. Compare the value received in FPGA with the packet content displayed in Wireshark.



Name	Value
rx_axis_tdata[63:0]	2c2ab76000000000
rx_axis_tkeep[7:0]	0f
rx_axis_tlast	1
rx_axis_tready	1
rx_axis_tvalid	1

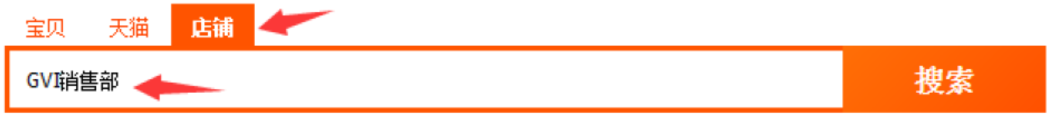
www.gvi-tech.com

Reference

- More design resources can be found :

  www.gvi-tech.com

- How to Buy：
  - https://detail.tmall.com/item.htm?id=562769965498


宝贝
天猫
店铺
GVI销售部
搜索