



**Nuclei**

**FPGA Evaluation Boards**

**and Debugger Kit**

**Introduction**

## Copyright Notice

Copyright © 2018–2020 Nuclei System Technology. All rights reserved.

Nuclei™ are trademarks owned by Nuclei System Technology. All other trademarks used herein are the property of their respective owners.

The product described herein is subject to continuous development and improvement; information herein is given by Nuclei in good faith but without warranties.

This document is intended only to assist the reader in the use of the product. Nuclei System Technology shall not be liable for any loss or damage arising from the use of any information in this document, or any incorrect use of the product.

## Contact Information

Should you have any problems with the information contained herein or any suggestions, please contact Nuclei System Technology by email [support@nucleisys.com](mailto:support@nucleisys.com), or visit “Nuclei User Center” website <http://user.nucleisys.com> for supports or online discussion.

## Revision History

Rev	Revision Date	Revised Section	Revised Content
1.0.0	2020/1/20	N/A	1. First version as the full English
2.0.0	2020/12/1	4. & 5.	1. Add MCU200T Evaluation FPGA Board 2. Add DDR200T Evaluation FPGA Board

# Table of Contents

<b>COPYRIGHT NOTICE</b> .....	0
<b>CONTACT INFORMATION</b> .....	0
<b>REVISION HISTORY</b> .....	1
<b>TABLE OF CONTENTS</b> .....	2
<b>LIST OF FIGURES</b> .....	4
<b>1. INTRODUCTION</b> .....	5
<b>2. HUMMINGBIRD DEBUGGER KIT</b> .....	8
2.1. OVERVIEW OF DEBUGGER KIT.....	8
2.2. FEATURES OF DEBUGGER KIT.....	8
2.3. INSTALL THE DRIVER FOR DEBUGGER KIT.....	9
2.3.1. <i>Install the Driver in Linux PC</i> .....	9
2.3.2. <i>Install the Driver in Windows PC</i> .....	10
<b>3. HUMMINGBIRD EVALUATION FPGA BOARD</b> .....	12
3.1. OVERVIEW OF FPGA BOARD.....	12
3.2. BOARD HARDWARE RESOURCES .....	13
3.2.1. <i>Overall Introduction of Hardware Resources</i> .....	13
3.2.2. <i>As the SoC prototype board directly</i> .....	15
3.2.3. <i>As regular FPGA board resource</i> .....	17
3.3. GENERATE THE BITSTREAM FILE (MCS FORMAT).....	19
3.4. PROGRAM THE BITSTREAM (MCS FORMAT) INTO FPGA .....	19
<b>4. MCU200T EVALUATION FPGA BOARD</b> .....	24
4.1. OVERVIEW OF FPGA BOARD.....	24
4.2. BOARD HARDWARE RESOURCES .....	25
4.2.1. <i>Overall Introduction of Hardware Resources</i> .....	25

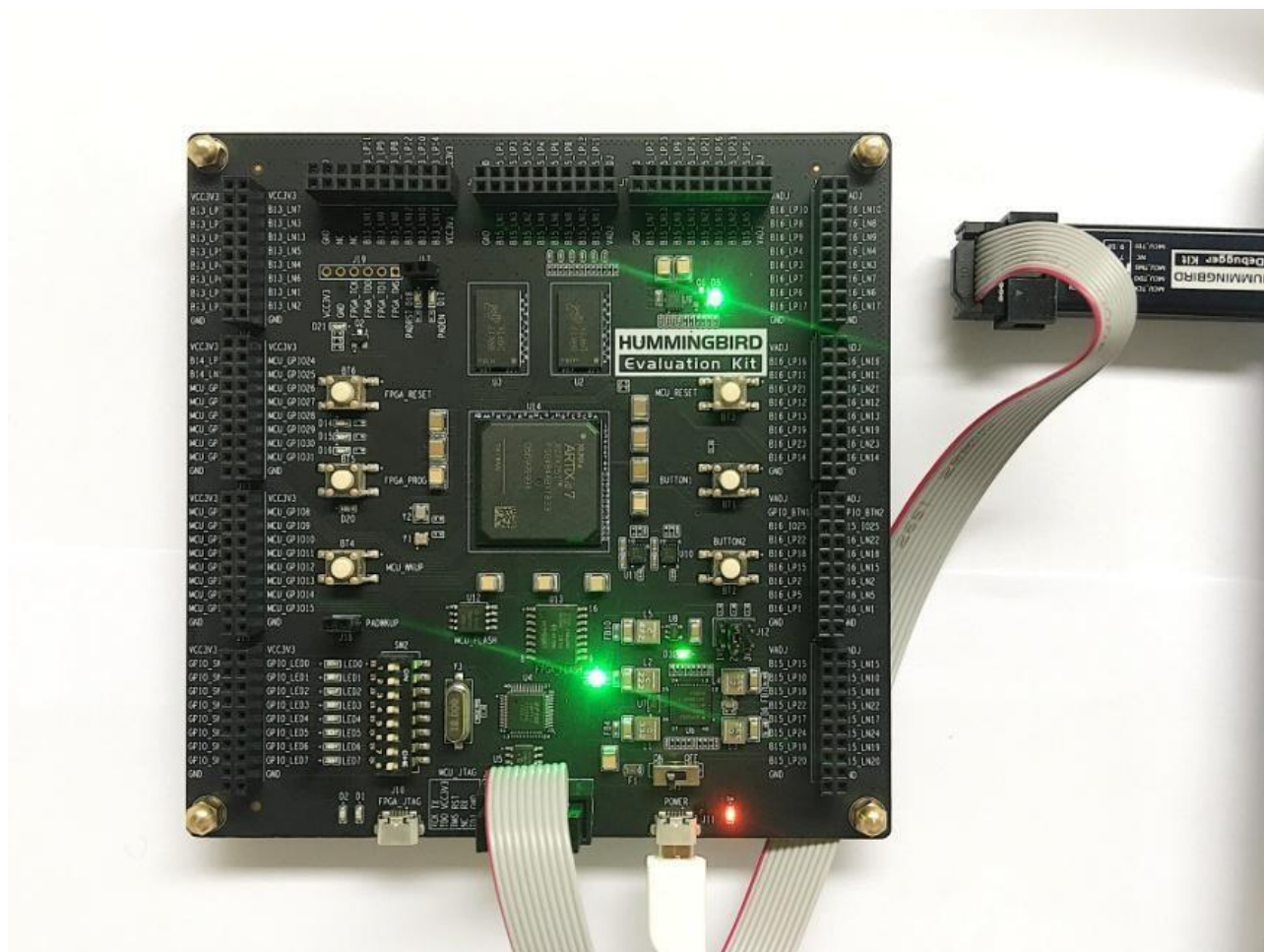
4.2.2.	<i>As the SoC prototype board directly</i> .....	28
4.2.3.	<i>As regular FPGA board resource</i> .....	29
4.3.	GENERATE THE BITSTREAM FILE (MCS FORMAT).....	30
4.4.	PROGRAM THE BITSTREAM (MCS FORMAT) INTO FPGA .....	30
<b>5.</b>	<b>DDR200T EVALUATION FPGA BOARD</b> .....	<b>35</b>
5.1.	OVERVIEW OF FPGA <sub>BOARD</sub> .....	35
5.2.	BOARD HARDWARE RESOURCES .....	36
5.2.1.	<i>Overall Introduction of Hardware Resources</i> .....	36
5.2.2.	<i>As the SoC prototype board directly</i> .....	39
5.2.3.	<i>As regular FPGA board resource</i> .....	41
5.3.	GENERATE THE BITSTREAM FILE (MCS FORMAT).....	41
5.4.	PROGRAM THE BITSTREAM (MCS FORMAT) INTO FPGA .....	42

# List of Figures

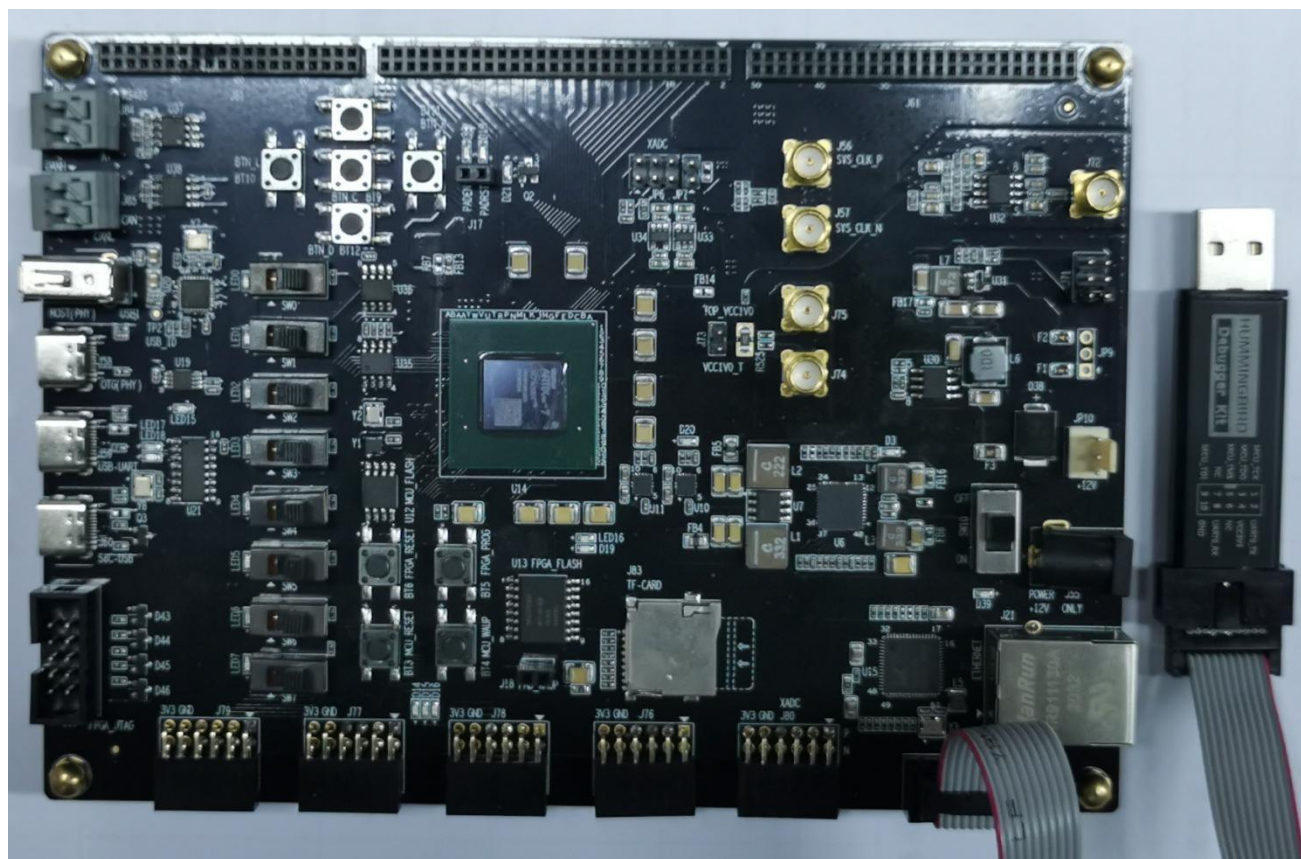
FIGURE 1-1 HUMMINGBIRD EVALUATION KIT AND HUMMINGBIRD DEBUGGER KIT .....	5	FIGURE 1-2 MCU200T EVALUATION KIT AND HUMMINGBIRD DEBUGGER KIT.....	6
FIGURE 1-3 DDR200T EVALUATION KIT AND HUMMINGBIRD DEBUGGER KIT .....	7	FIGURE 2-1 HUMMINGBIRD DEBUGGER KIT .....	8
FIGURE 2-2 THE UBUNTU OS RECOGNIZED USB .....	10	FIGURE 2-3 DOWNLOAD THE HUMMINGBIRD DEBUGGER KIT'S DRIVER FOR WINDOWS .....	11
FIGURE 3-1 BOARD HARDWARE RESOURCES.....	13	FIGURE 3-2 CONNECTING HUMMINGBIRD FPGA BOARD AND DEBUGGER KIT WITH PC .....	17
FIGURE 3-3 EXAMPLE OF USING OTHER BOARD RESOURCE TO CONTROL SOC PINS .....	18	FIGURE 3-4 THE FPGA_JTAG INTERFACE AND THE 5V POWER INTERFACE.....	20
FIGURE 3-5 THE HARDWARE MANAGER OF VIVADO .....	21	FIGURE 3-6 USE VIVADO HARDWARE MANAGER TO CONNECT FPGA.....	21
FIGURE 3-7 ADD CONFIGURATION MEMORY DEVICE .....	22	FIGURE 3-8 SELECT THE FLASH TYPES .....	22
FIGURE 3-9 ADD THE CONFIGURATION FILES .....	23	FIGURE 4-1 BOARD HARDWARE RESOURCES .....	25
FIGURE 4-2 CONNECTING MCU200T FPGA BOARD AND DEBUGGER KIT WITH PC.....	29	FIGURE 4-3 THE FPGA_JTAG CONNECTOR AND THE 12V POWER INTERFACE .....	31
FIGURE 4-4 THE HARDWARE MANAGER OF VIVADO .....	32	FIGURE 4-5 USE VIVADO HARDWARE MANAGER TO CONNECT FPGA.....	32
FIGURE 4-6 ADD CONFIGURATION MEMORY DEVICE.....	33	FIGURE 4-7 SELECT THE FLASH TYPES.....	33
FIGURE 4-8 ADD THE CONFIGURATION FILES.....	34	FIGURE 5-1 BOARD HARDWARE RESOURCES .....	36
FIGURE 5-2 CONNECTING DDR200T FPGA BOARD AND DEBUGGER KIT WITH PC .....	40	FIGURE 5-3 THE FPGA_JTAG INTERFACE AND THE 12V POWER INTERFACE .....	43
FIGURE 5-4 THE HARDWARE MANAGER OF VIVADO .....	44	FIGURE 5-5 USE VIVADO HARDWARE MANAGER TO CONNECT FPGA .....	44
FIGURE 5-6 ADD CONFIGURATION MEMORY DEVICE.....	45	FIGURE 5-7 SELECT THE FLASH TYPES .....	45
FIGURE 5-8 ADD THE CONFIGURATION FILES .....	46		

## 1. Introduction

This document introduce the Nuclei made FPGA evaluation boards, which can be used to prototype the Nuclei evaluation SoC (included RISC-V Core) in FPGA boards; and the Nuclei made JTAG Debugger kit (called **Hummingbird Debugger Kit**), which can be used to debug the RISC-V Core in FPGA prototype or in real chip.



**Figure 1-1** Hummingbird Evaluation Kit and Hummingbird Debugger Kit



**Figure 1-2** MCU200T Evaluation Kit and Hummingbird Debugger Kit



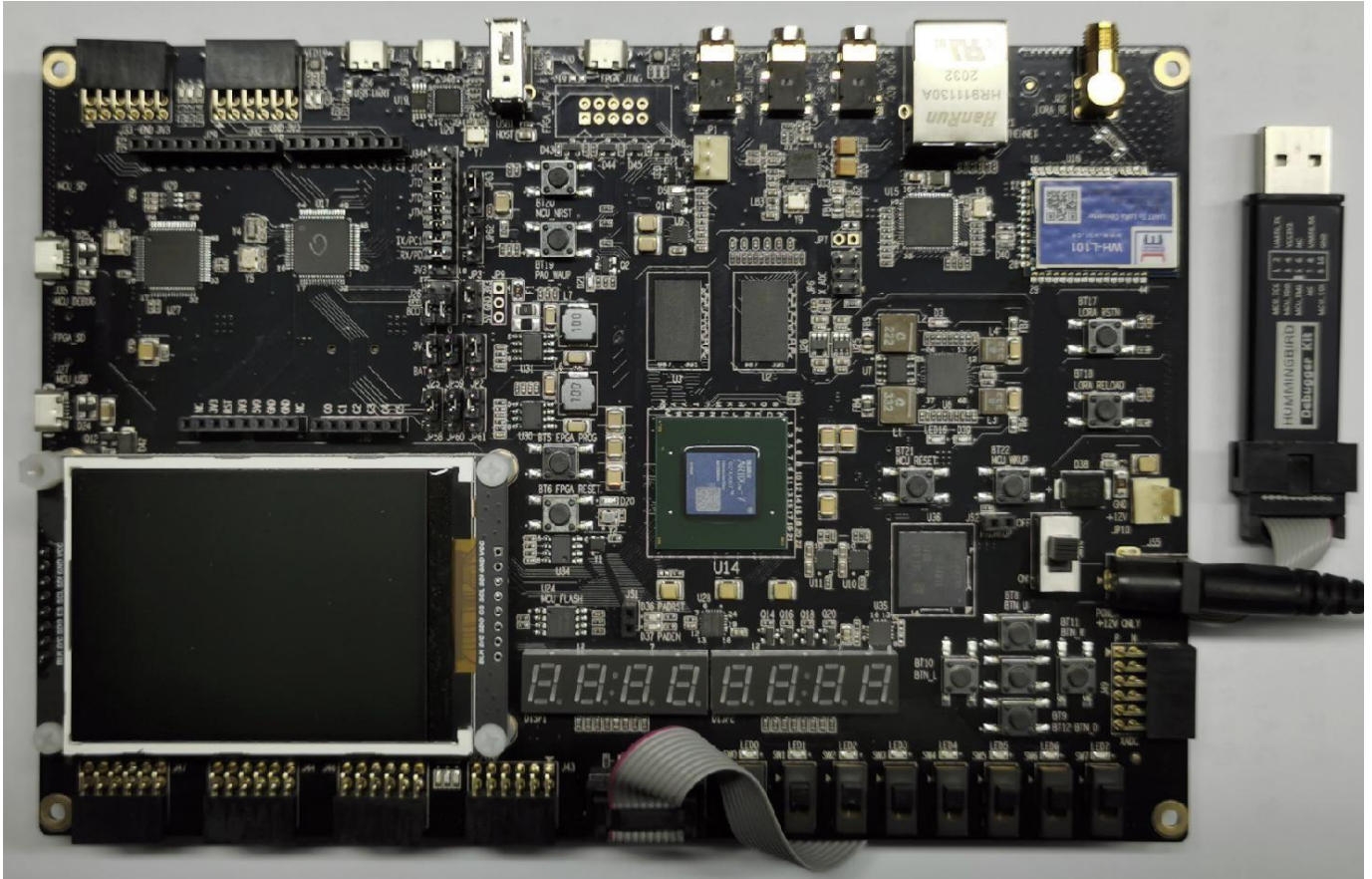


Figure 1-3 DDR200T Evaluation Kit and Hummingbird Debugger Kit

## 2. Hummingbird Debugger Kit

### 2.1. Overview of Debugger Kit

Nuclei have customized a Debugger hardware (called Hummingbird Debugger Kit), which can be used to debug the RISC-V core in FPGA prototype or in real chip.



**Figure 2-1 Hummingbird Debugger Kit**

## 2.2. Features of Debugger Kit

The Hummingbird Debugger Kit, as depicted in Figure 2-1, which have the features as below:

- One end of the Debugger Kit is the USB interface, which can be connected to PC.
- The other end of Debugger Kit is the interface socket, 4 wires of which are for JTAG, and 2 wires are for UART.
- The Debugger Kit is using the FTDI 2232 chip inside it, when it is connected to PC, it will be recognized as two USB devices:
  - One device is to bridge the USB to JTAG, such that the debugger software running on PC (e.g., GDB + OpenOCD) can debug the RISC-V Core through this bridge.
  - Another device is to bridge the UART to USB, such that the SoC's UART's information can also be redirected to PC.

## 2.3. Install the Driver for Debugger Kit

Since the Hummingbird Debugger Kit is connecting PC with USB interface, the PC need to be installed with the drivers to make sure it recognize USB devices.

### 2.3.1. Install the Driver in Linux PC

For the Linux computer, the steps of installing driver for Hummingbird Debugger Kit are as below:

```
// Step 1: Prepare the PC, it is recommended to have the following configurations.  
Use the VMware WorkStation to have Linux virtual machine installed, or just use the native  
Linux OS on your computer. It is recommended to use Ubuntu 16.04 or above version.
```

```
// Step 2: Connect the PC with the "Debugger Kit", make sure the USB is really be recognized  
by the Linux. For example, the USB symbol is highlighted in Ubuntu, as depicted in Figure 2-2.
```

```
// Step 3: Use the command to check the USB status:
```

```
lsusb // The example information displayed as below ...  
Bus 001 Device 010: ID 0403:6010 Future Technology Devices International, Ltd FT2232xxxx
```

```
// Step 4: Use the following command to set udev rules, to make this USB can be accessed by  
plugdev group sudo vi /etc/udev/rules.d/99-openocd.rules
```

```
// Use vi command to edit the file, and add the following lines  
SUBSYSTEM=="usb", ATTR{idVendor}=="0403",  
ATTR{idProduct}=="6010", MODE="664", GROUP="plugdev"  
SUBSYSTEM=="tty", ATTR{idVendor}=="0403",  
ATTR{idProduct}=="6010", MODE="664", GROUP="plugdev"
```

```
// Step 5: Use the following command to check if this USB is belong to plugdev group:
```

```
ls /dev/ttyUSB* // The example information showed as below after this command  
/dev/ttyUSB0 /dev/ttyUSB1
```

```
ls -l /dev/ttyUSB1 // The example information showed as below after this command crw-rw-r--  
1 root plugdev 188, 1 Nov 28 12:53 /dev/ttyUSB1
```

```
// Step 6: Add your user name into the plugdev group
whoami
// Use above command to check your user name, assuming it is your_user_name
// Use below command to add your_user_name into plugdev group sudo
usermod -a -G plugdev your_user_name

// Step 7: Double check if your user name is really belong to plugdev group
groups // The example information showed as below after this command
... plugdev ...
// As long as you can see plugdev in groups, then means it is really belong to.
```

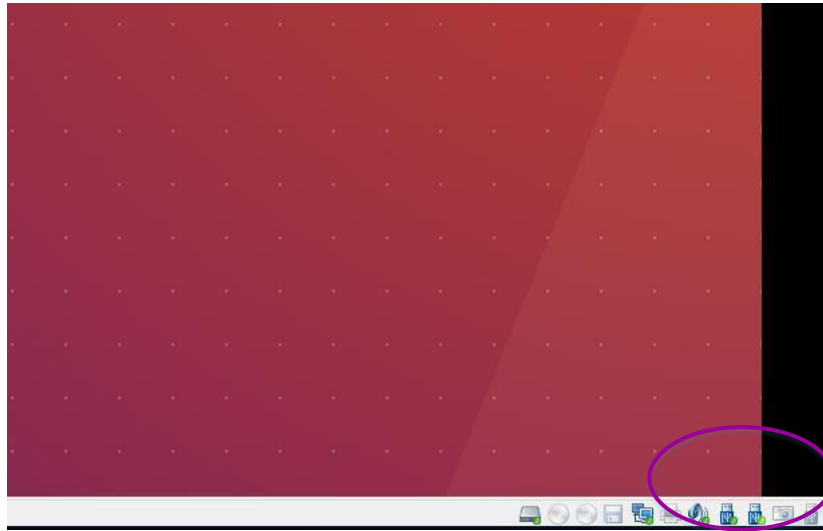


Figure 2-2 The Ubuntu OS recognized USB

### 2.3.2. Install the Driver in Windows PC

For the Windows computer, the steps of installing driver for Hummingbird Debugger Kit are as below:

```
// Step 1: Connect the PC with the "Debugger Kit", make sure the USB is really be recognized by the Windows.
// Step 2: Download the Hummingbird Debugger Kit's Driver for Windows, from the website:
http://www.nucleisys.com/developboard.php, as depicted in Figure 2-3.

// Step 3: After downloading the package, double-click the HBird-Driver.exe from it.
```

```
// Step 4: Since the Hummingbird Debugger Kit have the functionality that "convert the UART to USB", so if you have connected the "Hummingbird Debugger Kit" with PC and installed the driver successfully, then you will be able to see a USB Serial Port (e.g., COM8) show up in your Windows Device Manager.
```



Figure 2-3 Download the Hummingbird Debugger Kit's Driver for Windows

## 3. Hummingbird Evaluation FPGA Board

### 3.1. Overview of FPGA board

Nuclei have customized a FPGA evaluation board (called Hummingbird Evaluation Kit), which is to make "One board serves two purposes":

- As the SoC prototype board directly:
  - If the FPGA have been pre-burned (programmed) with "Nuclei evaluation SoC", this board can be worked as a SoC prototype directly. Since the board has been designed with buttons and extended ports names in line with the SoC GPIO pin name, the

embedded software engineers can directly use this board without knowing any FPGA hardware knowledge.

- To easy the writing, the “Nuclei evaluation SoC” will be shorted as “the SoC” in this document thereby.
- As the regular FPGA board.
  - For those hardware engineers who know FPGA hardware knowledge, this board can be used as regular FPGA board, which can be used to burn (program) any Verilog HDL design.

## **3.2. Board Hardware Resources**

### **3.2.1. Overall Introduction of Hardware Resources**



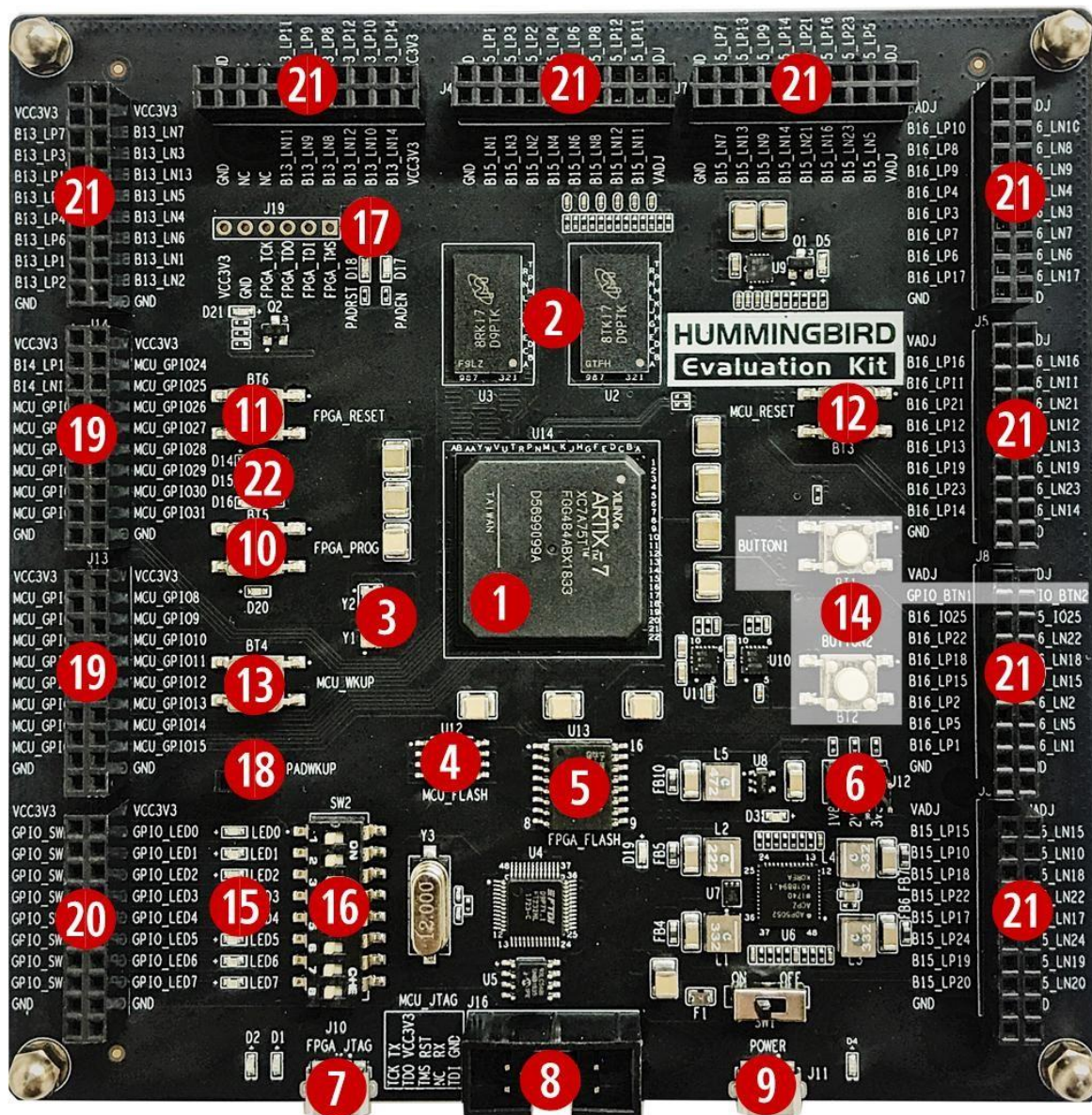


Figure 3-1 Board Hardware Resources

Hummingbird Evaluation Kit is an entry-level FPGA board based on Xilinx FPGA, as depicted in Figure 3-1, the hardware resource (marked as with different numbers in above picture) of this board is described as follows.

The basic hardware resources for FPGA:

- Xilinx FPGA Chip (XC7A75T-2FGG484I).
- 4Gb DDR3L SDRAM with 32-bit bus (MT41K128M16JT-125K DDR III).

- Two Crystal Oscillators, Y1 is for 100MHz Clock, Y2 is for 32.768K RTC Clock. (Mark:[3])
- FPGA\_FLASH, independent SPI Flash chip for FPGA to store its Bitstream (MCS format). (Mark:[5])
  - The Xilinx Vivado support the Bitstream to be burned into external SPI Flash with MCS format, the FPGA hardware will automatically re-load the Bitstream from external Flash into the FPGA each time after reset. By this way, FPGA can make sure its Bitstream will not be lost after power-down (since Flash will retain its Bitstream). Please refer to Section 3.4 for the steps how to burn MCS format Bitstream into external Flash.
- The jumpers, which can be used to configure the voltage level of FPGA\_GPIO, as 1.8V, 2.5V, or 3.3V. (Mark:[6])
- The FPGA\_JTAG interface, which is the on-board USB JTAG programmer for programming the Bitstream of the FPGA. (Mark:[7])
- Separated DC 5V power supply and a power switch. (Mark:[9])
- The FPGA\_PROG button to force the FPGA to re-load the Bitstream from the external Flash. (Mark:[10])

The pre-defined hardware resources for the RISC-V Core and the SoC are as below. Please refer to Section 3.2.2 for more details of how these resource work.

- The MCU\_FLASH, independent Flash chip for RISC-V core (burned inside FPGA) to store its instruction programs or read-only data. (Mark:[4])
- The MCU\_JTAG socket, which is the on-board JTAG interface socket for RISC-V Core debugging. (Mark:[8])
- The MCU\_GPIO connector, for 32 GPIO pins of the SoC. (Mark:[19])
- The FPGA\_RESET button for PoR reset of the SoC. (Mark:[11])



- The MCU\_RESET button for System reset of the SoC. (Mark:[12])

Other hardware resources as regular FPGA board resource:

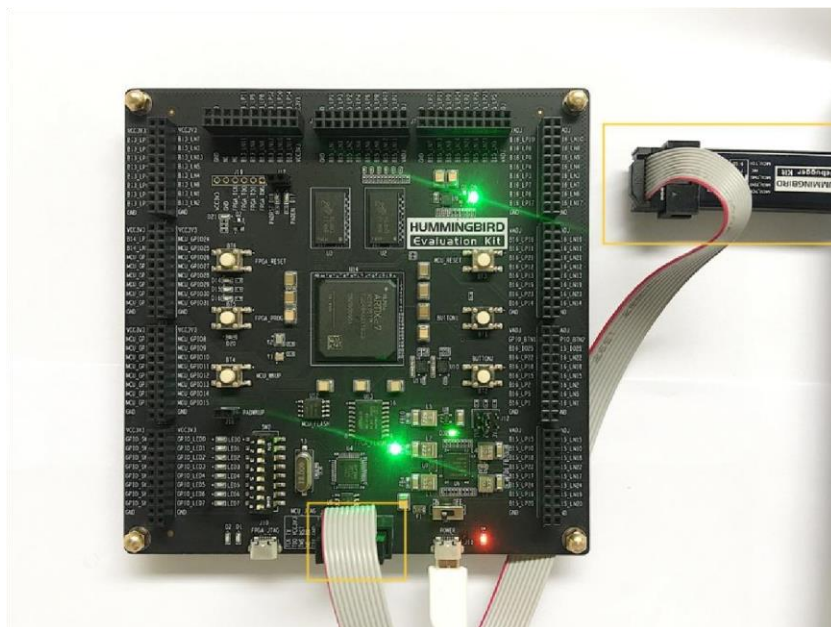
- The button 1 and button 2. (Mark:[14])
- The LEDs. (Mark:[15])
- The Switches. (Mark:[16])
- The LEDs and Switches exposed connectors. (Mark:[20])
- Up to 126 connectors for other FPGA IOs which can be used by user freely. (Mark:[21])
- The RGB LEDs. (Mark:[22])

### 3.2.2. As the SoC prototype board directly

As described in Section 3.2.1, one of the purposes of “Hummingbird Evaluation Kit” is to be as the SoC board directly. To achieve this goal, there are several key points here:

- The “Hummingbird Evaluation Kit” is supposed to be programmed with the Nuclei evaluation SoC (included RISC-V core). For the details of the SoC, please refer to document <Nuclei\_Eval\_SoC\_Intro.pdf>, this document can be easily acquired in Nuclei User Center <http://user.nucleisys.com>.
- There is a separated SPI Nor Flash on the board (as depicted in the Figure 3-1 mark [4]), this SPI Nor Flash will be connected to the SoC’s SPI interface, which support the XiP mode. The RISC-V Core in the FPGA will start to fetch the instructions (with XiP mode) from this Flash after reset.
- The SoC need two clock inputs, which are allocated at board as below:
  - The RTC clock is directly from the FPGA board 32.768KHz clock source (as depicted in the Figure 3-1 mark [3]).

- The high-speed clock is from the PLL output (in FPGA project). The original clock source of PLL is from the FPGA board 100MHz clock source (as depicted in the Figure 3-1 mark [3]).
- The SoC have the “System Reset” pin, and the FPGA project mapped the pin to the board MCU\_RESET button, as depicted in the Figure 3-1 mark [12].
- The SoC have the “PoR Reset” pin, and the FPGA project mapped the pin to the board FPGA\_RESET button, as depicted in the Figure 3-1 mark [11].
- The SoC have 32 GPIO pins, and the FPGA project mapped these GPIO pins to the board exposed connectors (with specific MCU\_GPIOx names as depicted in the Figure 3-1 mark [19]).
- There are 3 LED lights on the board, they are wired to 3 GPIO pins respectively, Red light to GPIO 19, Green light to GPIO 21, and Blue light to GPIO22.
- The SoC have JTAG pins and UART0 pins (pin-mux to GPIO\_16 and GPIO\_17 of the SoC), and the FPGA project mapped these pins to the board MCU\_JTAG interface socket, as depicted in the Figure 3-1 mark [8]. The “Hummingbird Debugger Kit” can be connected to this socket, as depicted in Figure 3-2, and then be used to debug the RISC-V core (burned inside FPGA).



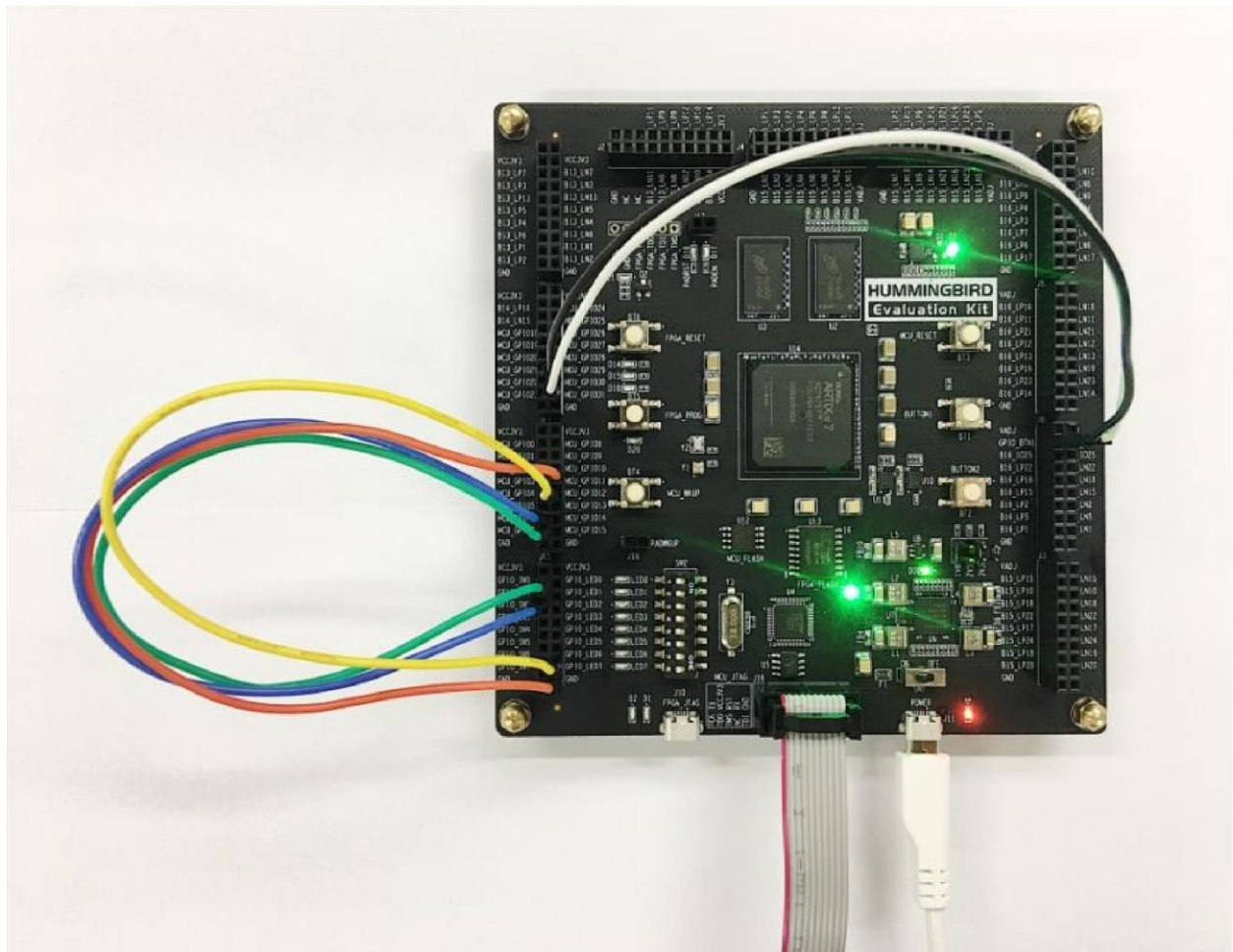
**Figure 3-2 Connecting Hummingbird FPGA board and Debugger Kit with PC**

### 3.2.3. As regular FPGA board resource

As described in Section 3.2.1, one of the purposes of “Hummingbird Evaluation Kit” is to be as the regular FPGA board. To achieve this goal, there are independent switches and LED lights, as depicted in the Figure 3-1 mark [14] [15] [16]. User can utilize these on-board resource to control the SoC, for examples as below:

- The switches (as depicted in the Figure 3-1 mark [16]) are independent there, without connecting to any others. User can use the jumper wire to connect it to MCU\_GPIOx pins, then to use the switch to control the MCU\_GPIO input value.
  - As in the example of Figure 3-3, SW0 and SW1 are connected to MCU\_GPIO6 and MCU\_GPIO7 respectively.
- The LED lights (as depicted in the Figure 3-1 mark [15]) are independent there, without connecting to any others. User can use the jumper wire to connect it to MCU\_GPIOx pins, then to use the MCU\_GPIO output value to control the LED lights.

- As in the example of Figure 3-3, MCU\_GPIO6 and MCU\_GPIO7 are connected to LED6 and LED7 respectively.
- The buttons (as depicted in the Figure 3-1 mark [14]) are independent there, without connecting to any others. User can use the jumper wire to connect it to MCU\_GPIOx pins, then to use the button to control the MCU\_GPIO input value.
- As in the example of Figure 3-3, Button 1 and Button 2 are connected to MCU\_GPIO30 and MCU\_GPIO31 respectively.



**Figure 3-3 Example of using other board resource to control SoC pins**

### **3.3. Generate the Bitstream file (MCS format)**

As depicted in the Figure 3-1 mark [5], the FPGA board has the SPI Flash to store FPGA's Bitstream (MCS format). If the users want to re-program the FPGA with the Bitstream file, Nuclei have provided the completed FPGA project (for each IP product), and then use Xilinx Vivado to generate the Bitstream file (MCS format).

For the detailed steps how to generate the Bitstream, please refer to each IP product's FPGA Prototype QuickStart document, which can be easily acquired in Nuclei User Center <http://user.nucleisys.com>.

### **3.4. Program the Bitstream (MCS format) into FPGA**

After generating the Bitstream file (MCS format), then we can program it into the FPGA's SPI Flash (as depicted in the Figure 3-1 mark [5]).

Here are the steps how to program it with Vivado:

// Step 1: Use the USB cable to connect the PC and the FPGA board's FPGA\_JTAG interface, the FPGA\_JTAG interface is the left yellow color highlighted box as depicted in the Figure 3-4.

// Step 2: Use the USB cable to connect the Power source and the FPGA board's 5V Power interface, the Power interface is the right yellow color highlighted box as depicted in the Figure 3-4. And then manually turn on the power by switching it to ON status, the LED light along with it will be on after it.

// Step 3: Open Xilinx Vivado, and open its Hardware Manager (as depicted in Figure 3-5), it will automatically recognize the board (via USB interface), as depicted in Figure 3-6.

// Step 4: Right click the FPGA Device name, and choose the "Add Configuration Memory Device", as depicted in Figure 3-7.

// Step 5: Select the Flash parameters, as depicted in Figure 3-8.

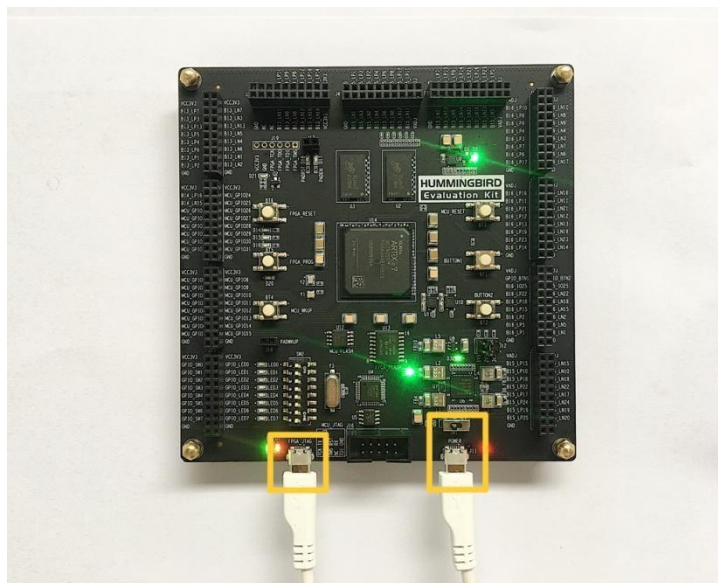
Part n25q128-3.3v  
Manufacturer Micron  
Family n25q  
Type spi  
Density 128  
Width x1 x2 x4

// Step 6: There will be a box pop out: "Do you want to program the configuration memory device now?" choose "OK".

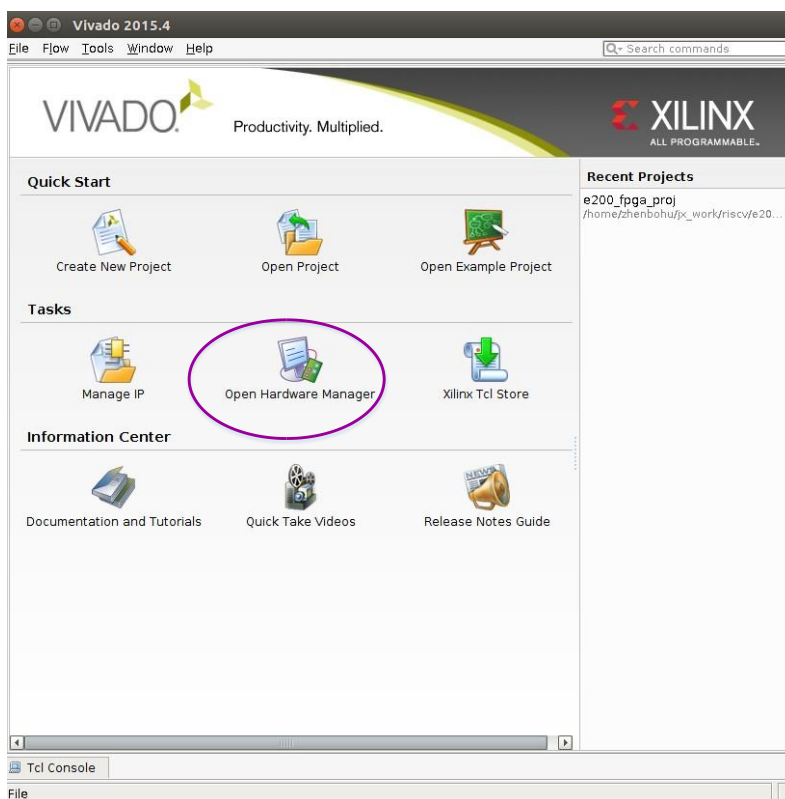
// Step 7: Choose the "Configuration file" as depicted in Figure 3-9, and add your MCS files (e.g., n205\_rls\_pkg/n205\_cct/fpga/hbirdkit/system.mcs), and then select OK, it will start to program FPGA's SPI Flash, it will take a few seconds to wait.

// Step 8: If the above step succeeded, user can push FPGA board's "FPGA\_PROG" button (as depicted in the Figure 3-1 mark [10]), it will trigger the FPGA to load the Bitstream from the SPI Flash into the FPGA chip and make it start to work.

// Step 9: If the above steps succeeded, then user can pull out the USB cable from "FPGA JTAG" interface, unless user want to re-program FPGA again.

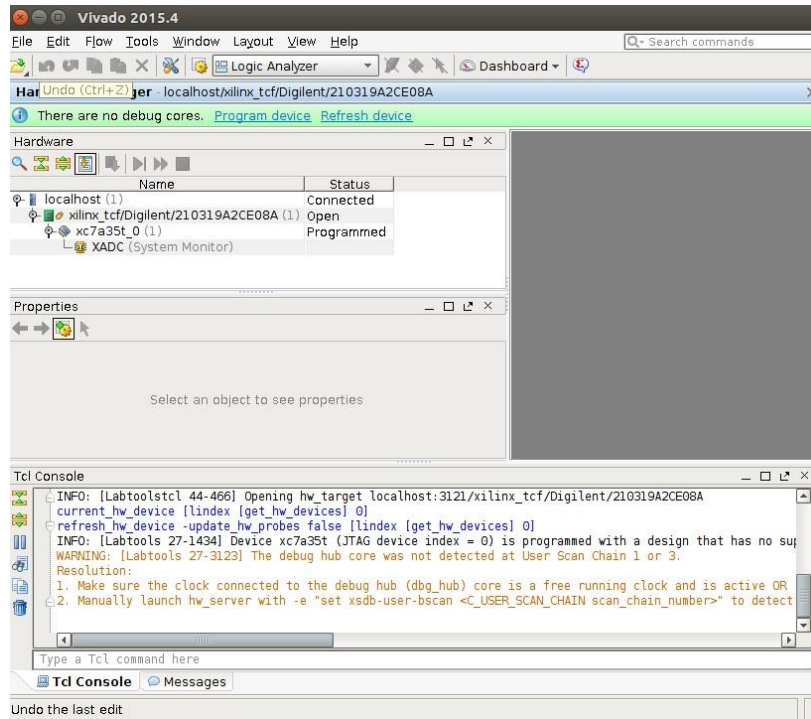


**Figure 3-4 The FPGA\_JTAG interface and the 5V Power interface**

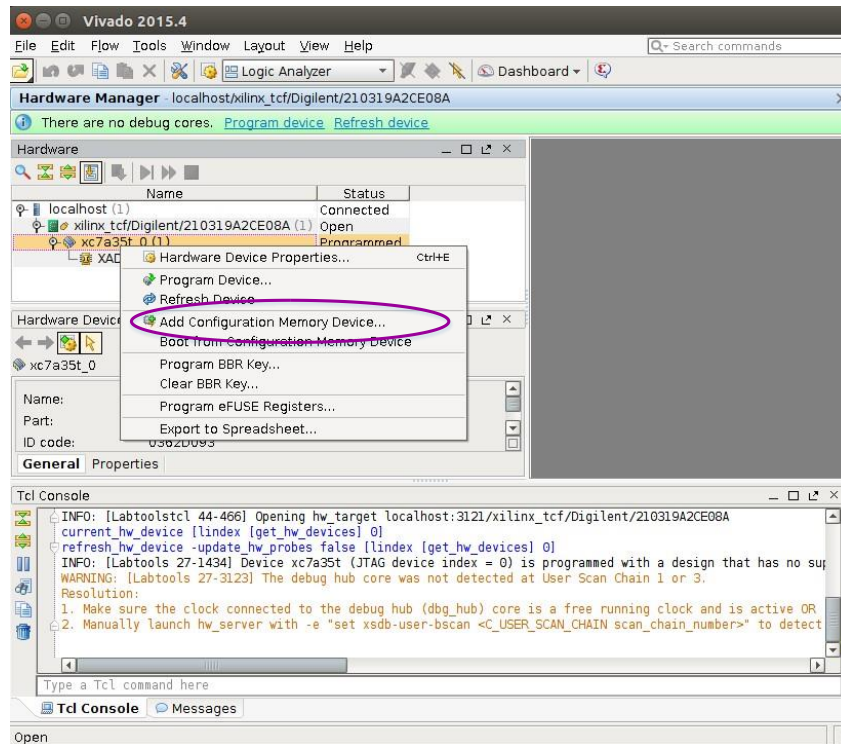


**Figure 3-5 The Hardware Manager of Vivado**





**Figure 3-6 Use Vivado Hardware Manager to connect FPGA**



**Figure 3-7 Add Configuration Memory Device**



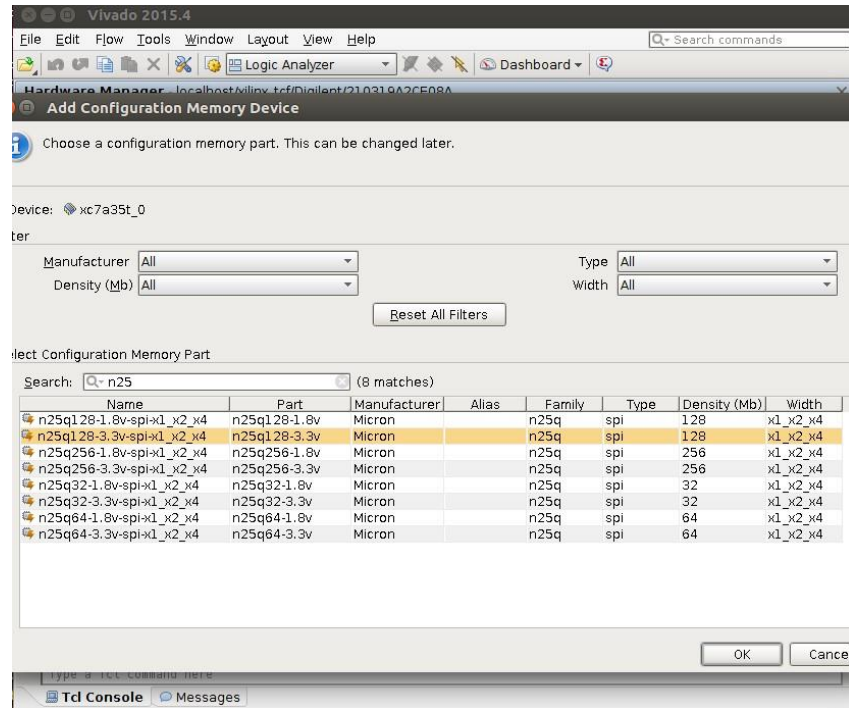


Figure 3-8 Select the Flash Types

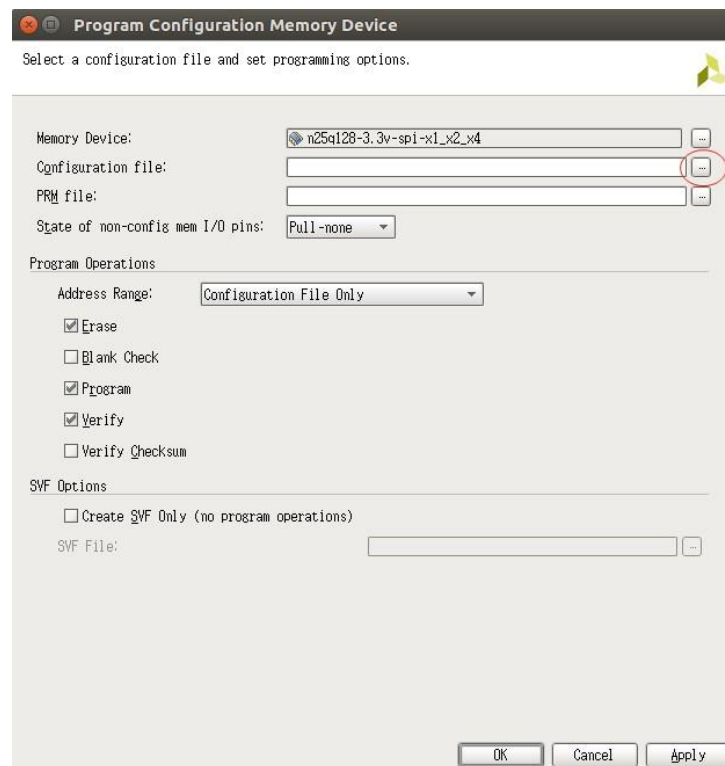


Figure 3-9 Add the Configuration Files

## 4. MCU200T Evaluation FPGA Board

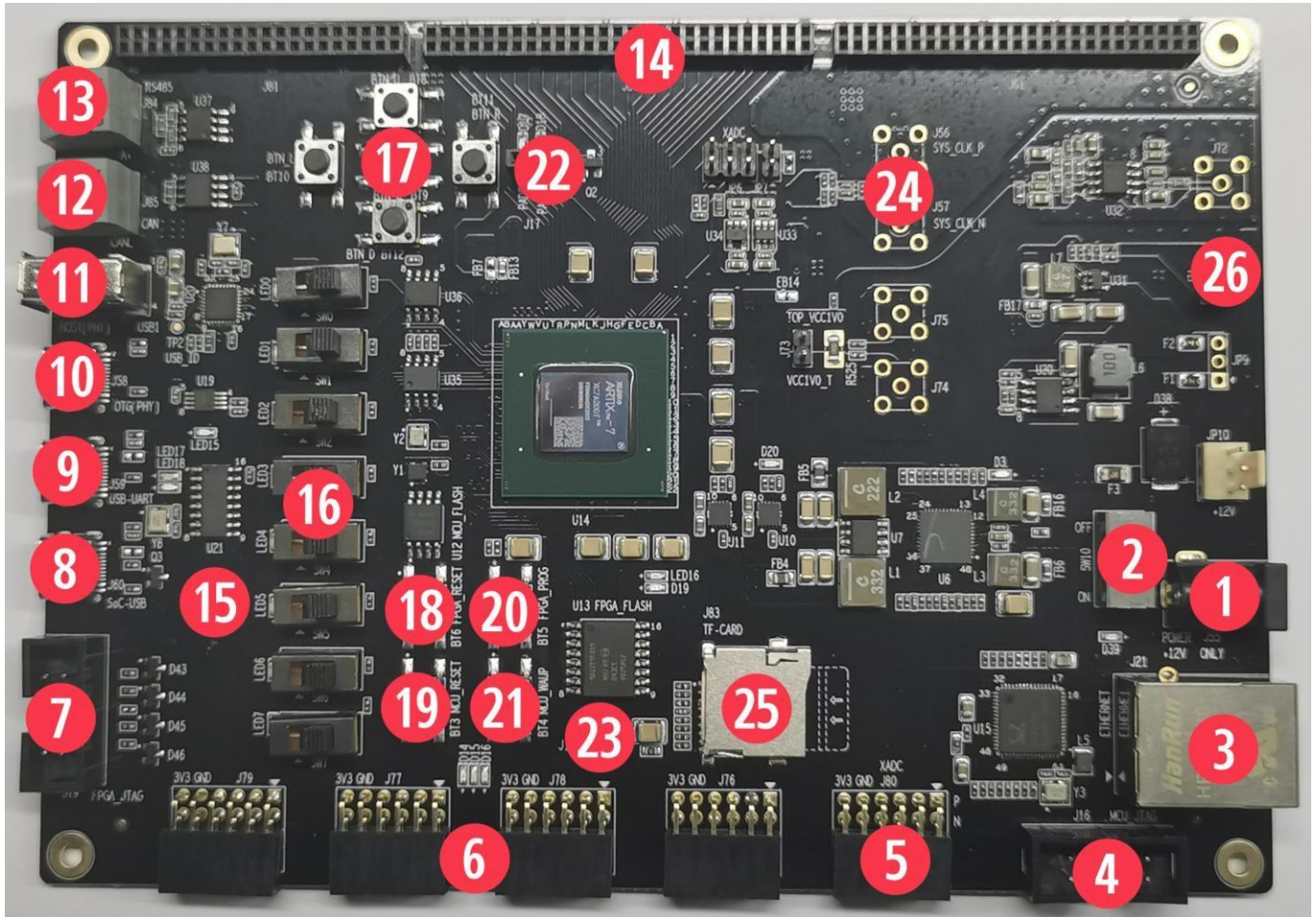
### 4.1. Overview of FPGA board

Nuclei have customized a FPGA evaluation board (called MCU200T Evaluation Kit), which is to make “One board serves two purposes”:

- As the SoC prototype board directly:
  - If the FPGA have been pre-burned (programmed) with “Nuclei evaluation SoC”, this board can be worked as a SoC prototype directly. Since the board has been designed with buttons and extended ports names in line with the SoC GPIO pin name, the embedded software engineers can directly use this board without knowing any FPGA hardware knowledge.
- To easy the writing, the “Nuclei evaluation SoC” will be shorted as “the SoC” in this document thereby.
- As the regular FPGA board.
  - For those hardware engineers who know FPGA hardware knowledge, this board can be used as regular FPGA board, which can be used to burn (program) any Verilog HDL design.

### 4.2. Board Hardware Resources

#### 4.2.1. Overall Introduction of Hardware Resources



**Figure 4-1 Board Hardware Resources**

MCU200T Evaluation Kit is an intermediate-level FPGA board based on Xilinx FPGA, as depicted in Figure 3-1 the hardware resource (marked as with different numbers in above picture) of this board is described as follows.

The basic hardware resources for FPGA:

- Xilinx FPGA Chip (XC7A200T-2FBG484I).
- Two Crystal Oscillators, Y1 is for 100MHz Clock, Y2 is for 32.768K RTC Clock.
- FPGA\_FLASH, independent SPI Flash chip for FPGA to store its Bitstream (MCS format).
  - The Xilinx Vivado support the Bitstream to be burned into external SPI Flash with MCS format, the FPGA hardware will automatically re-load the Bitstream from

external Flash into the FPGA each time after reset. By this way, FPGA can make sure its Bitstream will not be lost after power-down (since Flash will retain its Bitstream). Please refer to Section 3.4 for the steps how to burn MCS format Bitstream into external Flash.

- The  $V_{adj}$  jumpers, which can be used to configure the voltage level of FPGA\_GPIO, as 1.8V, 2.5V, or 3.3V.( Mark: [26])
- The FPGA\_JTAG connector(2.54mm 10PIN), used to connect USB-JTAG Programming cables for programming the Bitstream of the FPGA. ( Mark: [7])
- Separated **DC 12V** power supply and a power switch. ( Mark:[1] & [2])
- The FPGA\_PROG button to force the FPGA to re-load the Bitstream from the external Flash. ( Mark:[20])

The pre-defined hardware resources for the RISC-V Core and the SoC are as below. Please refer to Section 3.2.2 for more details of how these resource work.

- The MCU\_FLASH, independent Flash chip for RISC-V core (burned inside FPGA) to store its instruction programs or read-only data.
- The MCU\_JTAG socket, which is the on-board JTAG interface socket for RISC-V Core debugging. ( Mark:[4])
- The MCU\_GPIO ports x4 (Compatible with PMOD interface), for 32 GPIO pins of the SoC. ( Mark:[6])
- The FPGA\_RESET button for PoR reset of the SoC. ( Mark: [18])
- The MCU\_RESET button for System reset of the SoC. ( Mark:[19])

Other hardware resources as regular FPGA board resource:

- Differential SMA clock input . ( Mark:[24])

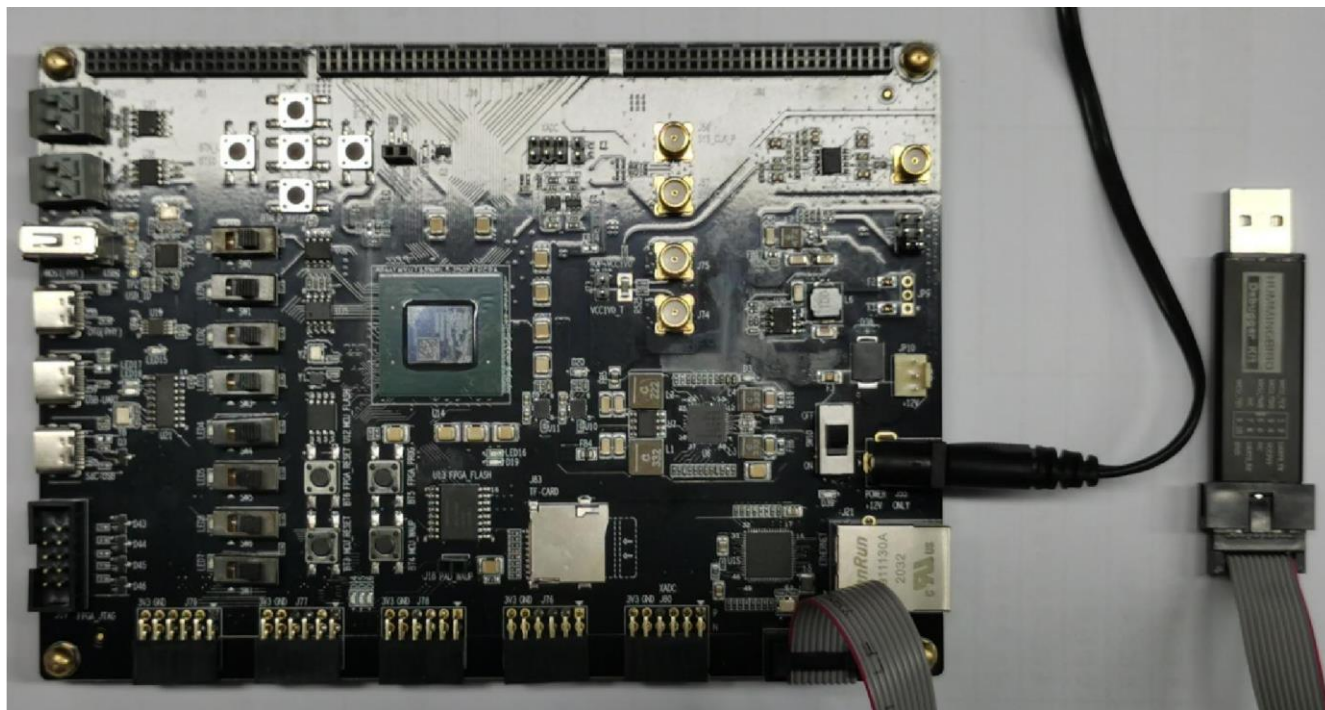
- User LEDs x8. ( Mark:[15])
- User Push Buttons x5 . ( Mark:[17])
- User DIP Switch (8-position) . (Mark:[16])
- XADC port (Compatible with PMOD interface). ( Mark:[5])
- User CAN port. ( Mark:[12])
- User RS485 port. ( Mark:[13])
- User 64Mb PSRAM. (SPI interface) ■ User IIC EEPROM: 24LC04 .
- FPGA SD Card Slot . (Mark:[25])
- SoC USB port . (Mark:[8])
- UART To USB Bridge .( Mark:[9])
- FPGA USB-OTG port .( Mark:[10])
- FPGA USB-HOST port .( Mark:[11])
- 10/100/1000 Mbps Ethernet RJ45 port(GMII). ( Mark:[3])
- FPGA GPIO ports x3 (2.0mm). ( Mark:[14])
- The RGB LEDs.

#### 4.2.2. As the SoC prototype board directly

As described in Section 3.2.1, one of the purposes of “MCU200T Evaluation Kit” is to be as the SoC board directly. To achieve this goal, there are several key points here:

- The “MCU200T Evaluation Kit” is supposed to be programmed with the Nuclei evaluation SoC (included RISC-V core). For the details of the SoC, please refer to document <Nuclei\_Eval\_SoC\_Intro.pdf>, this document can be easily acquired in Nuclei User Center <http://user.nucleisys.com>.
- There is a separated SPI Nor Flash on the board, this SPI Nor Flash will be connected to the SoC’s SPI interface, which support the XiP mode. The RISC-V Core in the FPGA will start to fetch the instructions (with XiP mode) from this Flash after reset.
- The SoC need two clock inputs, which are allocated at board as below:
  - The RTC clock is directly from the FPGA board 32.768KHz clock source.
  - The high-speed clock is from the PLL output (in FPGA project). The original clock source of PLL is from the FPGA board 100MHz clock source.
- The SoC have the “System Reset” pin, and the FPGA project mapped the pin to the board MCU\_RESET button, as depicted in the Figure 3-1 mark [19].
- The SoC have the “PoR Reset” pin, and the FPGA project mapped the pin to the board FPGA\_RESET button, as depicted in the Figure 3-1 mark [18].
- The SoC have 32 GPIO pins, and the FPGA project mapped these GPIO pins to the board exposed connectors.
- There are 3 LED lights on the board, they are wired to 3 GPIO pins respectively, Red light to GPIO 19, Green light to GPIO 21, and Blue light to GPIO22.
- The SoC have JTAG pins and UART0 pins (pin-mux to GPIO\_16 and GPIO\_17 of the SoC), and the FPGA project mapped these pins to the board MCU\_JTAG interface socket, as depicted in the Figure 3-1 mark [4]. The “MCU200T Debugger Kit” can be connected to this socket, as depicted in Figure 3-2, and then be used to debug the RISC-V core (burned inside FPGA).





**Figure 4-2 Connecting MCU200T FPGA board and Debugger Kit with PC**

#### 4.2.3. As regular FPGA board resource

As described in Section 3.2.1, one of the purposes of “MCU200T Evaluation Kit” is to be as the regular FPGA board. To achieve this goal, there are push buttons , switches and LED lights, as depicted in the Figure 3-1 mark [17] [16] [15]. User can utilize these on-board resource to control the SoC, for examples as below:

- The switches (as depicted in the Figure 3-1 mark [16]) connect directly to FPGA GPIO. User can rewrite the constraint file(.xdc) to connect it to MCU\_GPIOx pins, then to use the switch to control the MCU\_GPIO input value.
- The LED lights (as depicted in the Figure 3-1 mark [15]) connect directly to FPGA GPIO. User can rewrite the constraint file(.xdc) to connect it to MCU\_GPIOx pins, then to use the MCU\_GPIO output value to control the LED lights.

- The push buttons (as depicted in the Figure 3-1 mark [17]) connect directly to FPGA GPIO. User can rewrite the constraint file(.xdc) to connect it to MCU\_GPIOx pins, then to use the button to control the MCU\_GPIO input value.

### 4.3. Generate the Bitstream file (MCS format)

The FPGA board has the SPI Flash to store FPGA's Bitstream (MCS format). If the users want to re-program the FPGA with the Bitstream file, Nuclei have provided the completed FPGA project (for each IP product), and then use Xilinx Vivado to generate the Bitstream file (MCS format).

For the detailed steps how to generate the Bitstream, please refer to each IP product's FPGA Prototype QuickStart document, which can be easily acquired in Nuclei User Center <http://user.nucleisys.com>.

### 4.4. Program the Bitstream (MCS format) into FPGA

After generating the Bitstream file (MCS format), then we can program it into the FPGA's SPI Flash.

Here are the steps how to program it with Vivado:

```
// Step 1: Use the USB-JTAG Programming cables to connect the PC and the FPGA board's FPGA_JTAG connector, the FPGA_JTAG connector is the left yellow color highlighted box as depicted in the Figure 3-4.
```

```
// Step 2: Use the USB cable to connect the Power source and the FPGA board's 12V Power interface, the Power interface is the right yellow color highlighted box as depicted in the Figure 3-4. And then manually turn on the power by switching it to ON status, the LED light along with it will be on after it.
```

```
// Step 3: Open Xilinx Vivado, and open its Hardware Manager (as depicted in Figure 3-5), it will automatically recognize the board (via USB-JTAG Programming cables), as depicted in Figure 3-6.
```

```
// Step 4: Right click the FPGA Device name, and choose the "Add Configuration Memory Device", as depicted in Figure 3-7.
```



// Step 5: Select the Flash parameters, as depicted in Figure 3-8.

```
Part n25q128-3.3v
Manufacturer Micron
Family n25q
Type spi
Density 128
Width x1 x2 x4
```

// Step 6: There will be a box pop out: "Do you want to program the configuration memory device now?" choose "OK".

// Step 7: Choose the "Configuration file" as depicted in Figure 3-9, and add your MCS files (e.g., n205\_rls\_pkg/n205\_cct/fpga/hbirdkit/system.mcs), and then select OK, it will start to program FPGA's SPI Flash, it will take a few seconds to wait.

// Step 8: If the above step succeeded, user can push FPGA board's "FPGA\_PROG" button (as depicted in the Figure 3-1 mark [20]), it will trigger the FPGA to load the Bitstream from the SPI Flash into the FPGA chip and make it start to work.

// Step 9: If the above steps succeeded, then user can pull out the USB-JTAG Programming cables from "FPGA JTAG" connector, unless user want to re-program FPGA again.

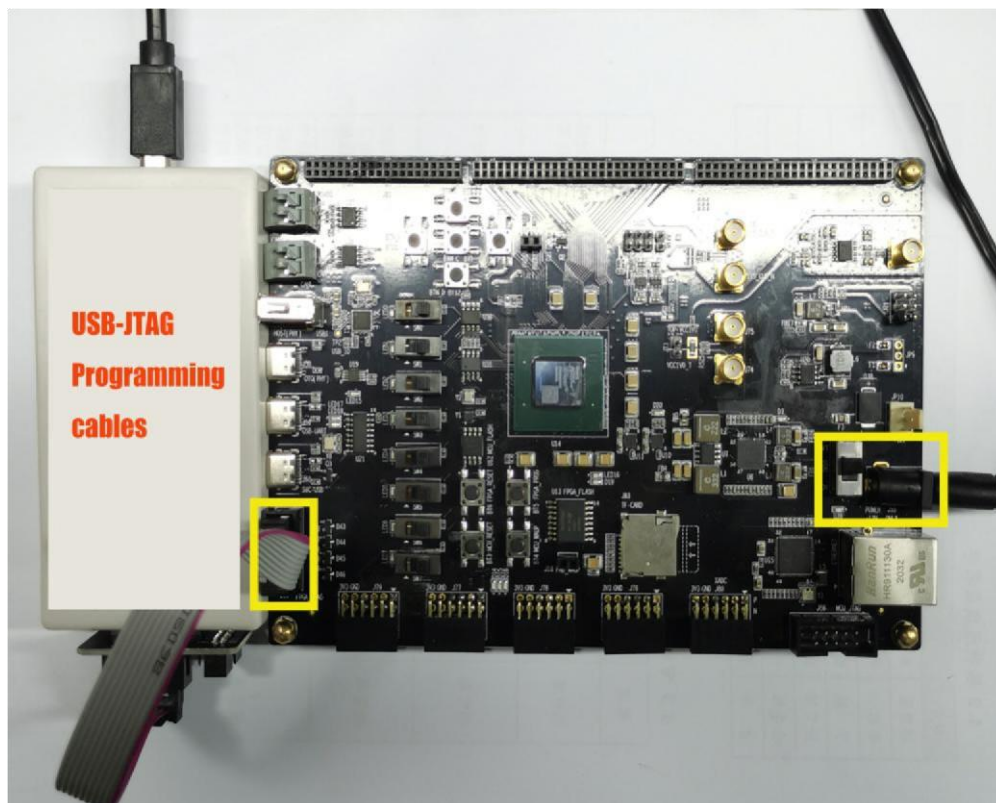
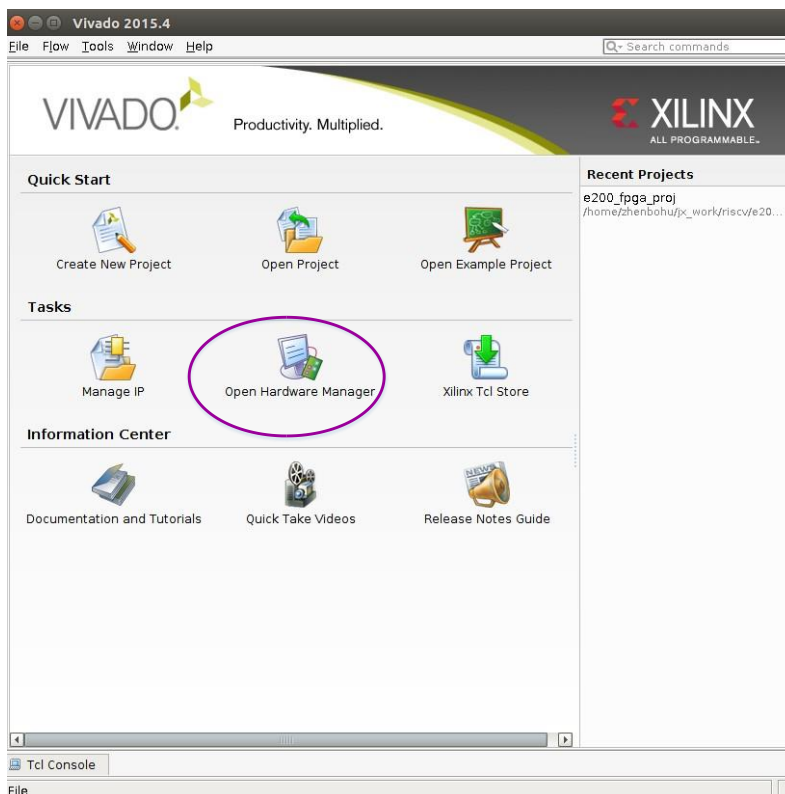
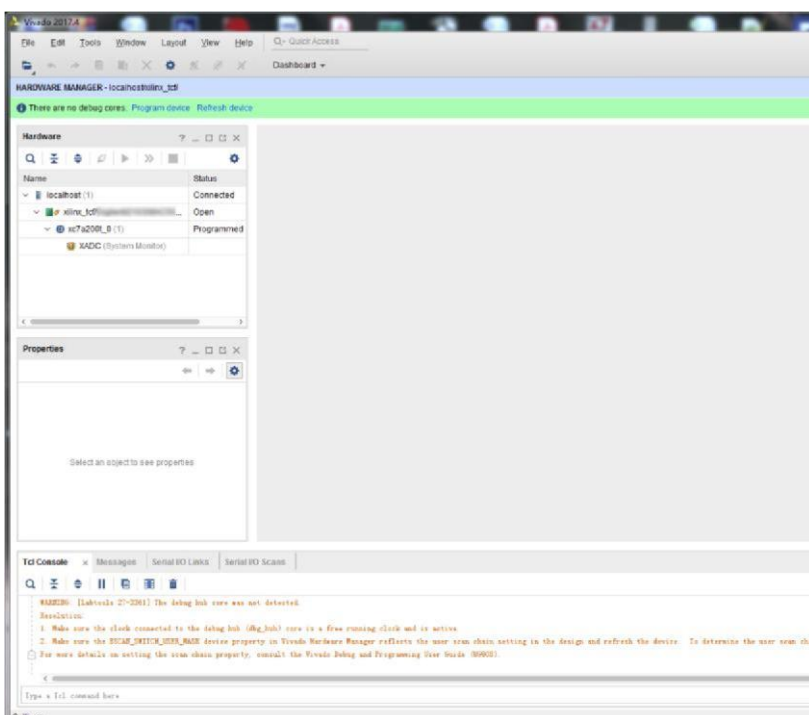


Figure 4-3 The FPGA\_JTAG connector and the 12V Power interface



**Figure 4-4 The Hardware Manager of Vivado**



**Figure 4-5 Use Vivado Hardware Manager to connect FPGA**

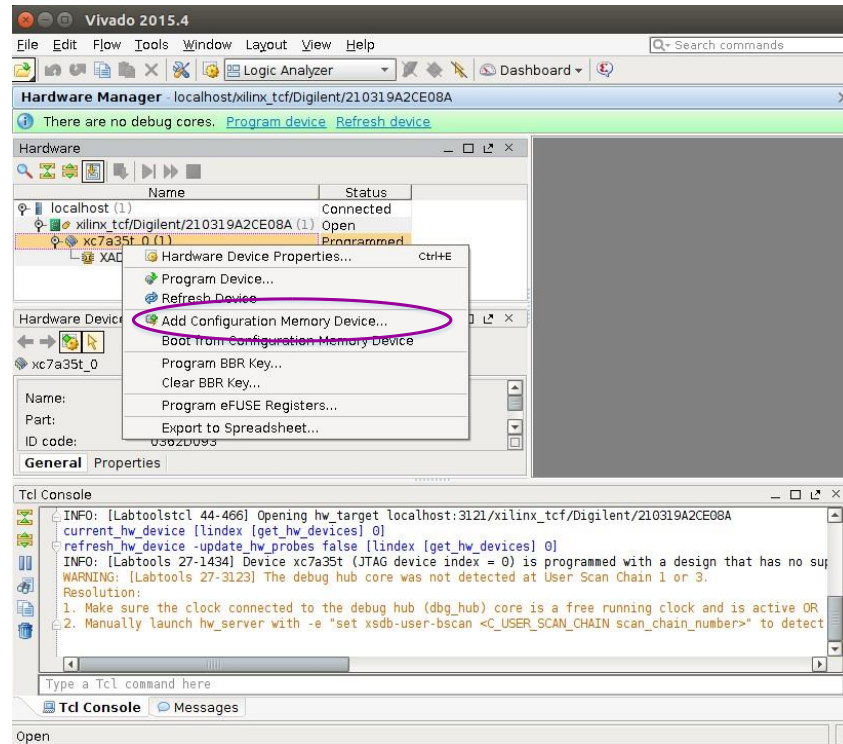


Figure 4-6 Add Configuration Memory Device

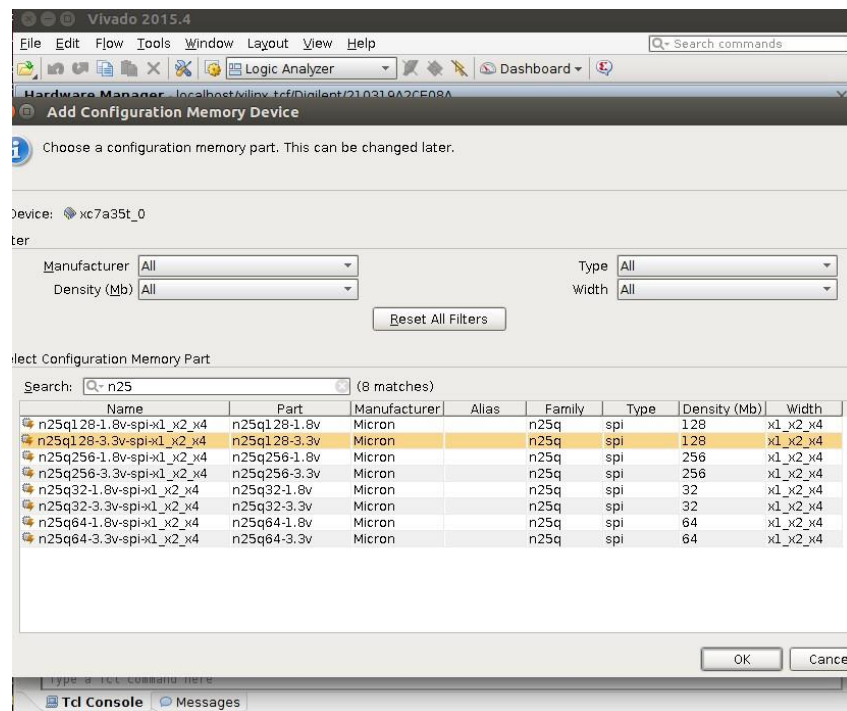


Figure 4-7 Select the Flash Types

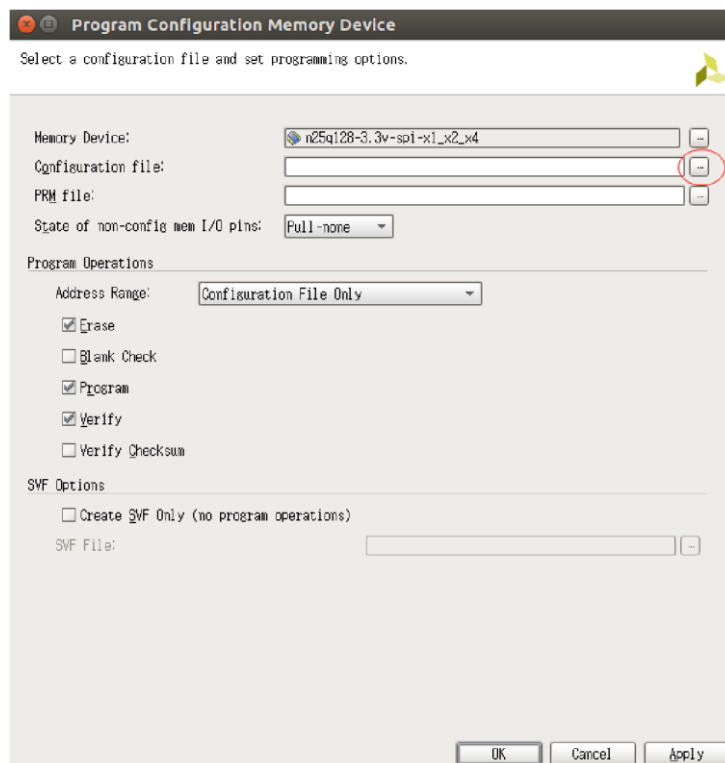


Figure 4-8 Add the Configuration Files

## 5. DDR200T Evaluation FPGA Board

### 5.1. Overview of FPGA board

Nuclei have customized a FPGA evaluation board (called DDR200T Evaluation Kit), which is to make “One board serves two purposes”:

- As the SoC prototype board directly:
  - If the FPGA have been pre-burned (programmed) with “Nuclei evaluation SoC”, this board can be worked as a SoC prototype directly. Since the board has been designed with buttons and extended ports names in line with the SoC GPIO pin name, the embedded software engineers can directly use this board without knowing any FPGA hardware knowledge.
- To easy the writing, the “Nuclei evaluation SoC” will be shorted as “the SoC” in this document thereby.



- As the regular FPGA board.

- For those hardware engineers who know FPGA hardware knowledge, this board can be used as regular FPGA board, which can be used to burn (program) any Verilog HDL design.

## 5.2. Board Hardware Resources

### 5.2.1. Overall Introduction of Hardware Resources

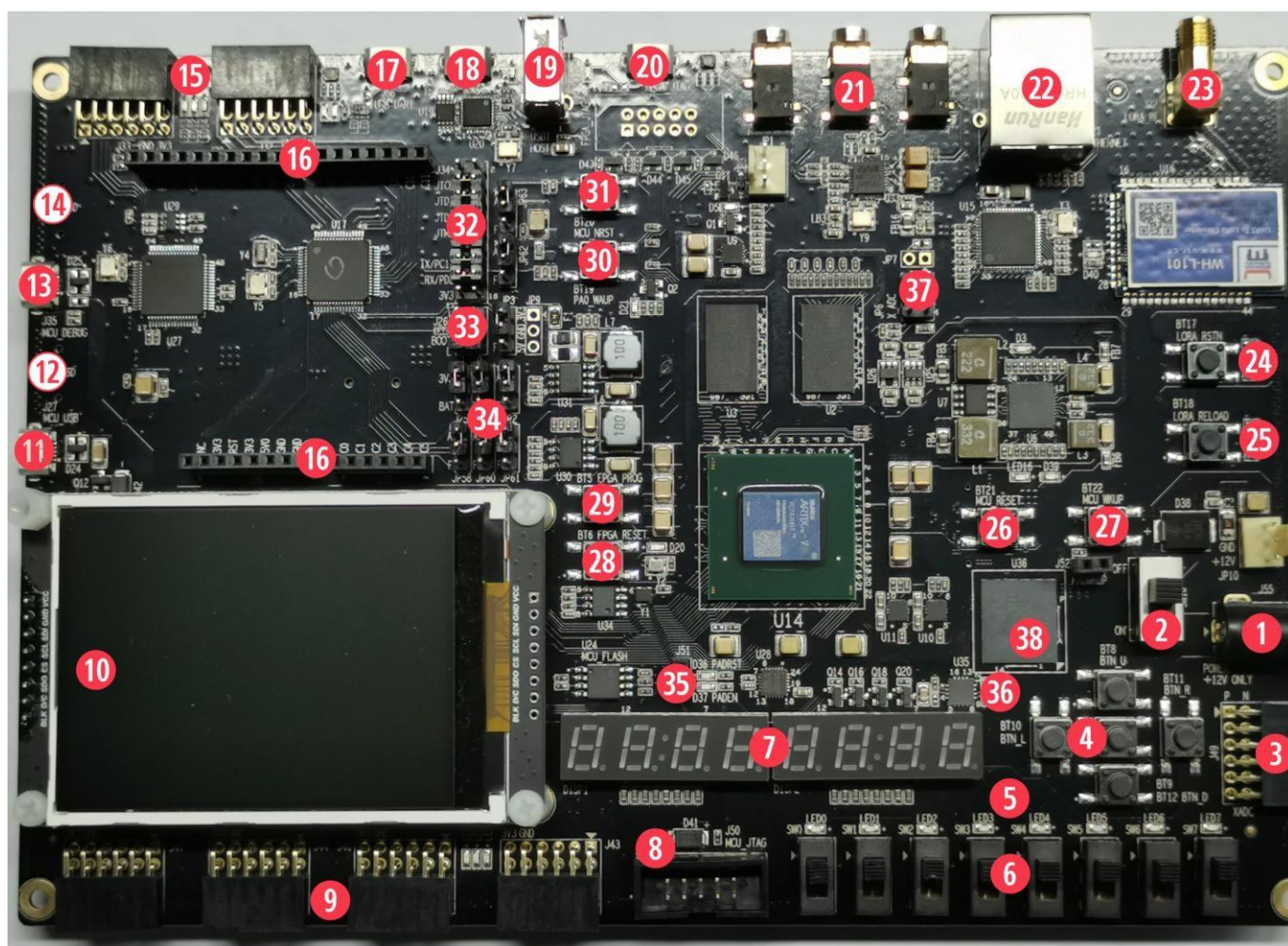


Figure 5-1 Board Hardware Resources

DDR200T Evaluation Kit is an high-level FPGA board based on Xilinx FPGA, as depicted in Figure 3-1, the hardware resource (marked as with different numbers in above picture) of this board is described as follows.

The basic hardware resources for FPGA:

- Xilinx FPGA Chip (XC7A200T-2FBG484I).
- 4Gb DDR3L SDRAM with 32-bit bus (MT41K128M16JT-125K DDR III)..
- Two Crystal Oscillators, Y1 is for 100MHz Clock, Y2 is for 32.768K RTC Clock.
- FPGA\_FLASH, independent SPI Flash chip for FPGA to store its Bitstream (MCS format).
  - The Xilinx Vivado support the Bitstream to be burned into external SPI Flash with MCS format, the FPGA hardware will automatically re-load the Bitstream from external Flash into the FPGA each time after reset. By this way, FPGA can make sure its Bitstream will not be lost after power-down (since Flash will retain its Bitstream). Please refer to Section 3.4 for the steps how to burn MCS format Bitstream into external Flash.
- The FPGA\_JTAG interface, which is the on-board USB JTAG programmer for programming the Bitstream of the FPGA. ( Mark: [20])
- Separated **DC 12V** power supply and a power switch. ( Mark:[1] & [2])
- The FPGA\_PROG button to force the FPGA to re-load the Bitstream from the external Flash. ( Mark:[29])

The pre-defined hardware resources for the RISC-V Core and the SoC are as below. Please refer to Section 3.2.2 for more details of how these resource work.

- The MCU\_FLASH, independent Flash chip for RISC-V core (burned inside FPGA) to store its instruction programs or read-only data.
- The MCU\_JTAG socket, which is the on-board JTAG interface socket for RISC-V Core debugging. ( Mark:[8])
- The MCU\_GPIO ports x4 (Compatible with PMOD interface), for 32 GPIO pins of the SoC. ( Mark:[9])

- The FPGA\_RESET button for PoR reset of the SoC. ( Mark: [28])
- The MCU\_RESET button for System reset of the SoC. ( Mark: [26])

Other hardware resources as regular FPGA board resource:

- User LEDs x8. ( Mark:[5])
- User Push Buttons x5 . ( Mark:[4])
- User DIP Switch (8-position) . (Mark:[6])
- XADC port (Compatible with PMOD interface). ( Mark:[3])
- Digit 7-Segment LED Display(8-position). (Mark:[7])
- 2.8inch LCD module . (Mark:[10]) ■ User IIC EEPROM: 24LC04 .
- FPGA SD Card Slot . (Mark:[12])
- UART To USB Bridge .( Mark:[17])
- FPGA USB-OTG port .( Mark:[18])
- FPGA USB-HOST port .( Mark:[19])
- FPGA Digital audio codec interface . ( Mark:[21])
- 10/100/1000 Mbps Ethernet RJ45 port(GMII). ( Mark:[22])
- LORA module RF SMA port . ( Mark:[23])
- LORA module RESET Push Buttons . ( Mark:[24])
- LORA module RELOAD Push Buttons . ( Mark:[25])
- FPGA 8G EMMC. ( Mark:[38]) RISC-V MCU (GD32VF103) resource:
- GD32VF103 USB port . ( Mark:[11])

- GD32VF103 on-board USB JTAG Debugger port . ( Mark:[13])
- GD32VF103 SD Card Slot . (Mark:[14])
- GD32VF103 PMOD port . ( Mark:[15])
- GD32VF103 Arduino port . ( Mark:[16])
- GD32VF103 MCU\_NRST Push Buttons . ( Mark:[38])
- GD32VF103 PAO\_WKUP Push Buttons . ( Mark:[31])
- GD32VF103 Debugger Select-jumper Header . ( Mark:[32])
- GD32VF103 BOOT-jumper Header . ( Mark:[33])
- GD32VF103 GPIO jumper Header(To FPGA GPIO or Arduino port) . ( Mark:[34])

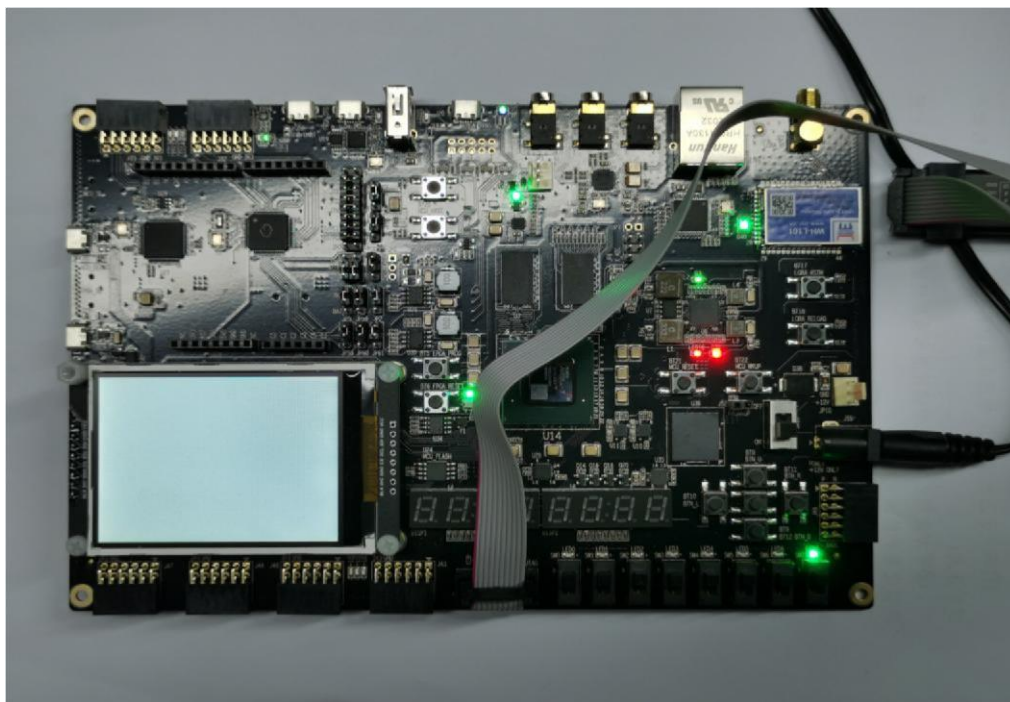
### 5.2.2. As the SoC prototype board directly

As described in Section 3.2.1, one of the purposes of “DDR200T Evaluation Kit” is to be as the SoC board directly. To achieve this goal, there are several key points here:

- The “DDR200T Evaluation Kit” is supposed to be programmed with the Nuclei evaluation SoC (included RISC-V core). For the details of the SoC, please refer to document <Nuclei\_Eval\_SoC\_Intro.pdf>, this document can be easily acquired in Nuclei User Center <http://user.nucleisys.com>.
- There is a separated SPI Nor Flash on the board, this SPI Nor Flash will be connected to the SoC’s SPI interface, which support the XiP mode. The RISC-V Core in the FPGA will start to fetch the instructions (with XiP mode) from this Flash after reset.
- The SoC need two clock inputs, which are allocated at board as below:
  - The RTC clock is directly from the FPGA board 32.768KHz clock source.



- The high-speed clock is from the PLL output (in FPGA project). The original clock source of PLL is from the FPGA board 100MHz clock source.
- The SoC have the “System Reset” pin, and the FPGA project mapped the pin to the board MCU\_RESET button, as depicted in the Figure 3-1 mark [26].
- The SoC have the “PoR Reset” pin, and the FPGA project mapped the pin to the board FPGA\_RESET button, as depicted in the Figure 3-1 mark [28].
- The SoC have 32 GPIO pins, and the FPGA project mapped these GPIO pins to the board exposed PMOD connectors.
- There are 3 LED lights on the board, they are wired to 3 GPIO pins respectively, Red light to GPIO 19, Green light to GPIO 21, and Blue light to GPIO22.
- The SoC have JTAG pins and UART pins (pin-mux to GPIO\_16 and GPIO\_17 of the SoC), and the FPGA project mapped these pins to the board MCU\_JTAG interface socket, as depicted in the Figure 3-1 mark [8]. The “DDR200T Debugger Kit” can be connected to this socket, as depicted in Figure 3-2, and then be used to debug the RISC-V core (burned inside FPGA).



**Figure 5-2 Connecting DDR200T FPGA board and Debugger Kit with PC**

### 5.2.3. As regular FPGA board resource

As described in Section 3.2.1, one of the purposes of “DDR200T Evaluation Kit” is to be as the regular FPGA board. To achieve this goal, there are push buttons , switches and LED lights, as depicted in the Figure 3-1 mark [4] [6] [5]. User can utilize these on-board resource to control the SoC, for examples as below:

- The switches (as depicted in the Figure 3-1 mark [6]) connect directly to FPGA GPIO. User can rewrite the constraint file(.xdc) to connect it to MCU\_GPIOx pins, then to use the switch to control the MCU\_GPIO input value.
- The LED lights (as depicted in the Figure 3-1 mark [5]) connect directly to FPGA GPIO. User can rewrite the constraint file(.xdc) to connect it to MCU\_GPIOx pins, then to use the MCU\_GPIO output value to control the LED lights.
- The buttons (as depicted in the Figure 3-1 mark [4]) connect directly to FPGA GPIO. User can rewrite the constraint file(.xdc) to connect it to MCU\_GPIOx pins, then to use the button to control the MCU\_GPIO input value.

### **5.3. Generate the Bitstream file (MCS format)**

The FPGA board has the SPI Flash to store FPGA's Bitstream (MCS format). If the users want to re-program the FPGA with the Bitstream file, Nuclei have provided the completed FPGA project (for each IP product), and then use Xilinx Vivado to generate the Bitstream file (MCS format).

For the detailed steps how to generate the Bitstream, please refer to each IP product's FPGA Prototype QuickStart document, which can be easily acquired in Nuclei User Center <http://user.nucleisys.com>.

### **5.4. Program the Bitstream (MCS format) into FPGA**

After generating the Bitstream file (MCS format), then we can program it into the FPGA's SPI Flash.

Here are the steps how to program it with Vivado:

// Step 1: Use the USB cable to connect the PC and the FPGA board's FPGA\_JTAG interface, the FPGA\_JTAG interface is the left red color highlighted box as depicted in the Figure 3-4.

// Step 2: Use the USB cable to connect the Power source and the FPGA board's 12V Power interface, the Power interface is the right red color highlighted box as depicted in the Figure 3-4. And then manually turn on the power by switching it to ON status, the LED light along with it will be on after it.

// Step 3: Open Xilinx Vivado, and open its Hardware Manager (as depicted in Figure 3-5), it will automatically recognize the board (via USB interface), as depicted in Figure 3-6.

// Step 4: Right click the FPGA Device name, and choose the "Add Configuration Memory Device", as depicted in Figure 3-7.

// Step 5: Select the Flash parameters, as depicted in Figure 3-8.

Part n25q128-3.3v  
Manufacturer Micron  
Family n25q  
Type spi  
Density 128  
Width x1 x2 x4

// Step 6: There will be a box pop out: "Do you want to program the configuration memory device now?" choose "OK".

// Step 7: Choose the "Configuration file" as depicted in Figure 3-9, and add your MCS files (e.g., n205\_rls\_pkg/n205\_cct/fpga/hbirdkit/system.mcs), and then select OK, it will start to program FPGA's SPI Flash, it will take a few seconds to wait.

// Step 8: If the above step succeeded, user can push FPGA board's "FPGA\_PROG" button (as depicted in the Figure 3-1 mark [29]), it will trigger the FPGA to load the Bitstream from the SPI Flash into the FPGA chip and make it start to work.

// Step 9: If the above steps succeeded, then user can pull out the USB cable from "FPGA JTAG" interface, unless user want to re-program FPGA again.

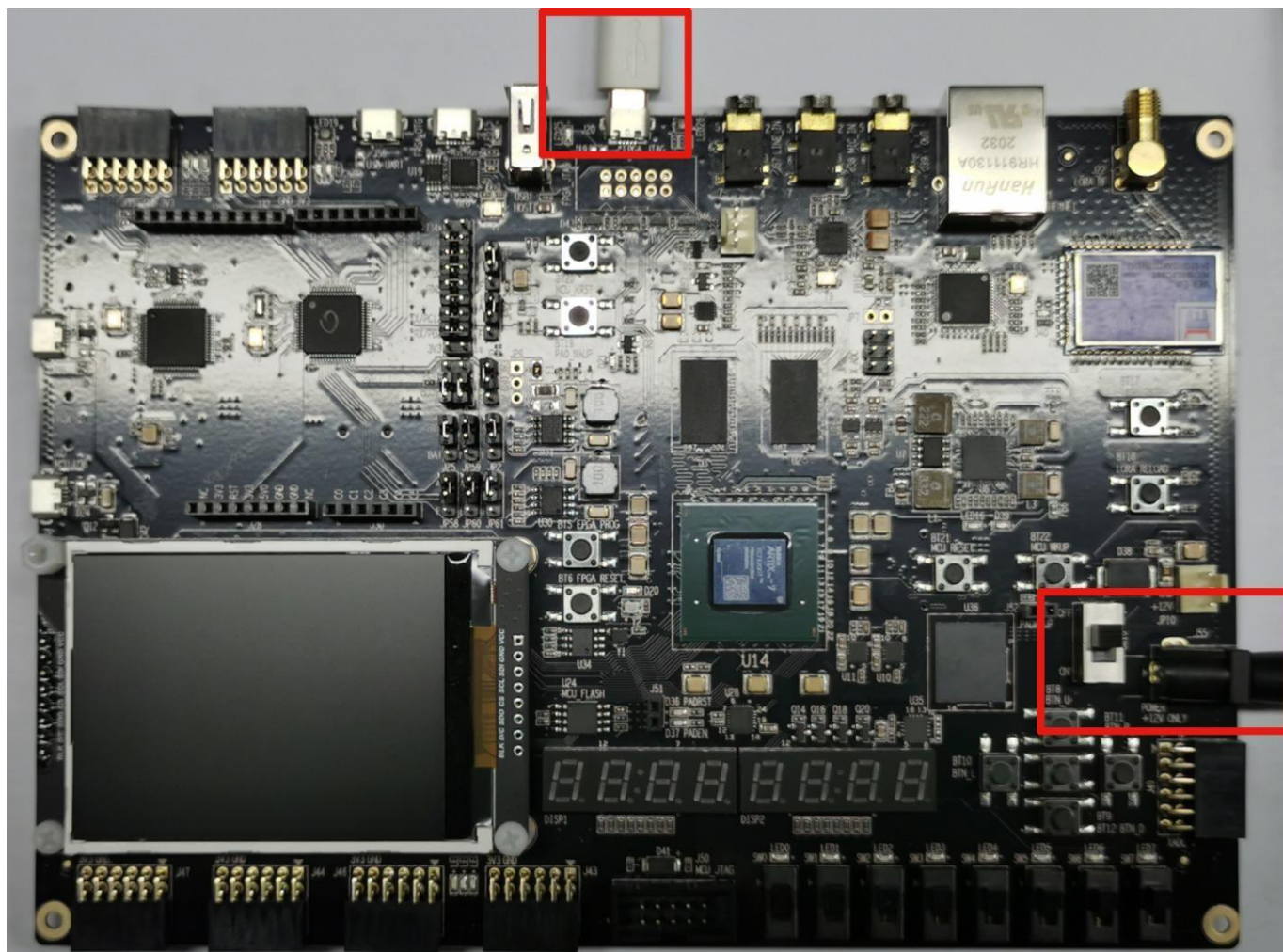


Figure 5-3 The FPGA\_JTAG interface and the 12V Power interface

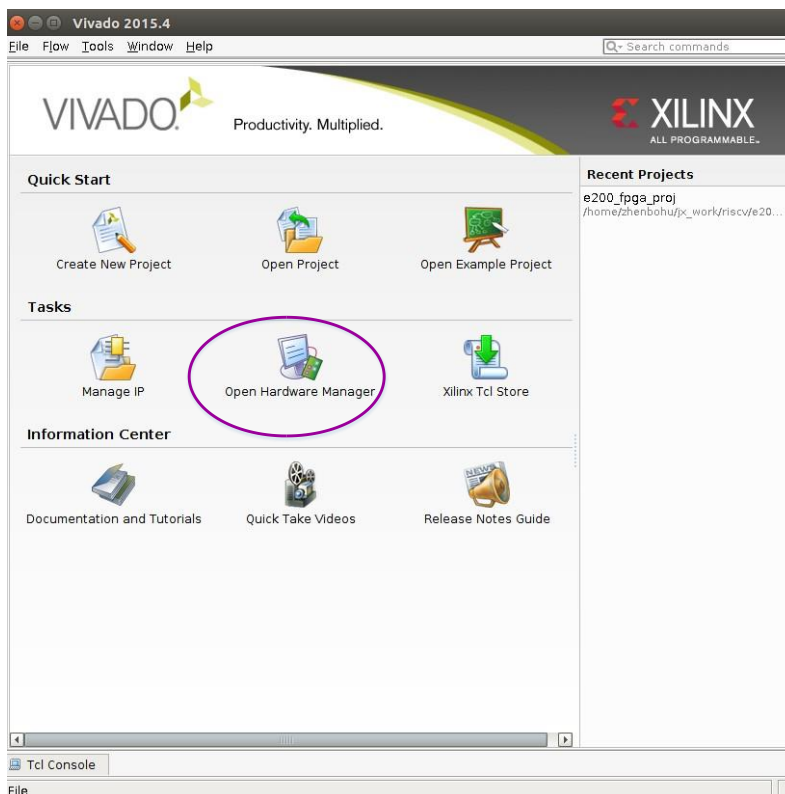


Figure 5-4 The Hardware Manager of Vivado

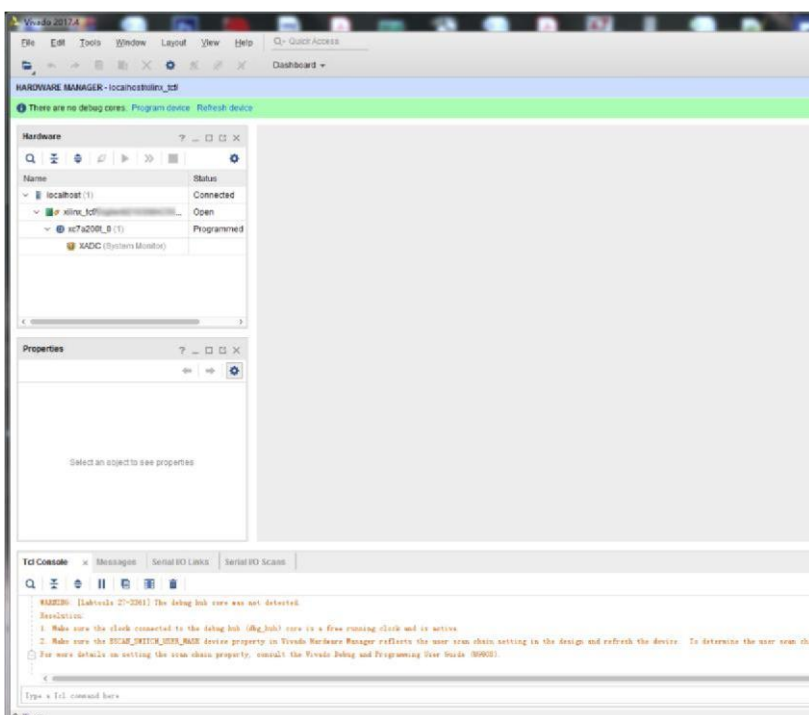


Figure 5-5 Use Vivado Hardware Manager to connect FPGA



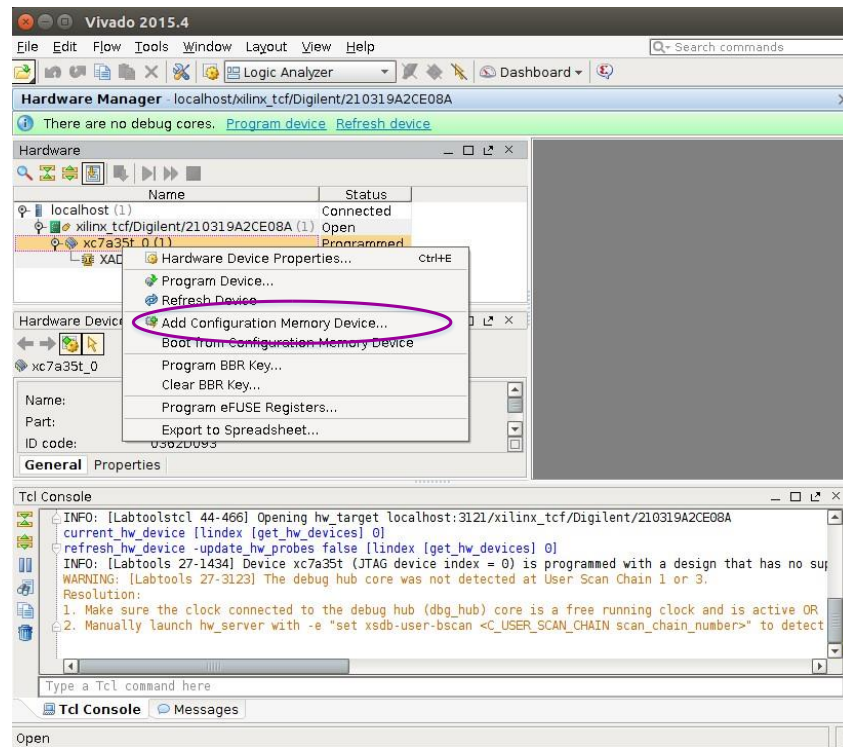


Figure 5-6 Add Configuration Memory Device

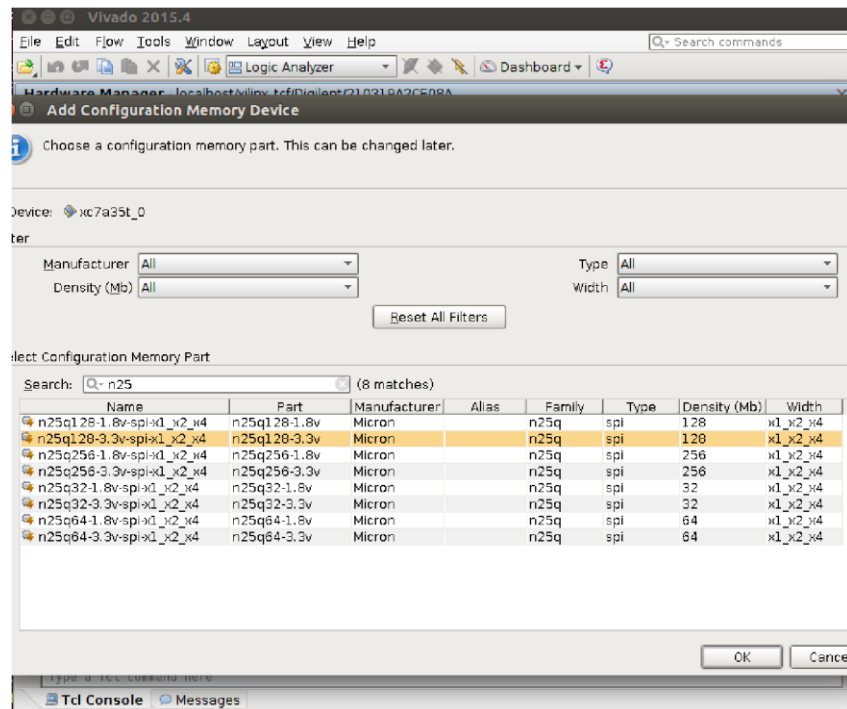
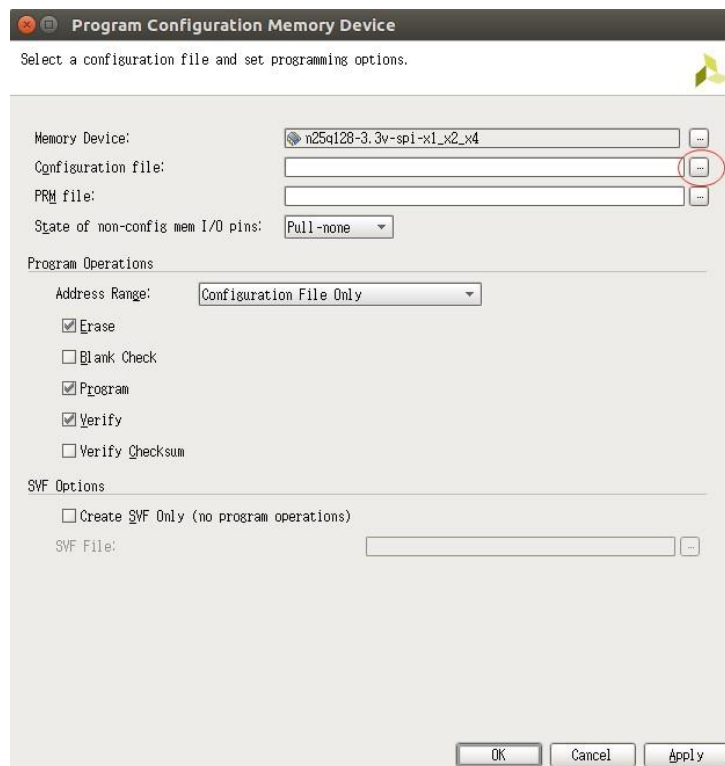


Figure 5-7 Select the Flash Types



**Figure 5-8 Add the Configuration Files**