

Nuclei

FPGA Evaluation Boards

and Debugger Kit

Introduction

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

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 <u>support@nucleisys.com</u>, or visit "Nuclei User Center" website <u>http://user.nucleisys.com</u> 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.	 Add MCU200T Evaluation FPGA Board Add DDR200T Evaluation FPGA Board



Table of Contents

COPY NOTIO	RIGH CE	FT		0
CONT. INFOI	ACT RMA	ΓΙΟΝ		0
REVIS HIST(SION ORY .			1
TABL	E OF			
CONT	ENT	S	•••••	2 LIST OF
FIGUI	RES .		,	4
1. II 	NTRO .5	ODUCTION	•••••	
2. H K	IUMI SIT	MINGBIRD DEBUGGER	}	
2.1.	Ov	ERVIEW OF DEBUGGER		
K _{IT}		n V	8 2.2.	FEATURES OF
THE	DRIV	ER FOR DEBUGGER K _{it}		
2.	.3.1.	Install the Driver in Linux PC	9	
2.	.3.2.	Install the Driver in Windows PC 10		
3. H	IUMI 2	MINGBIRD EVALUATION FPGA BOARD	,	
3.1. 3.2.	O _V Bo	erview of FPGA _{board} Ard Hardware R _{esources}		
3.	.2.1.	Overall Introduction of Hardware Resources 13		
3.	.2.2.	As the SoC prototype board directly 15 3.2.3. As regular FPGA board resource		
3.3.	Gene	RATE THE BITSTREAM FILE (MCS FORMAT)		
3.4.	Prog	RAM THE BITSTREAM (MCS FORMAT) INTO FPGA		
4. N 24	4 4	COOT EVALUATION FPGA BOARD		
4.1. 4.2.	Ov Bo	erview of FPGA _{board} ard Hardware R _{esources}		
4.	.2.1.	Overall Introduction of Hardware Resources		

4.2.2.	As the SoC prototype board directly	
	28 4.2.3 . As regular FPGA board	
	resource	
4.3. GENE	ERATE THE BITSTREAM FILE (MCS FORMAT)	30
4.4. Proc	GRAM THE BITSTREAM (MCS FORMAT) INTO FPGA	30
5. DDR2 35	200T EVALUATION FPGA BOARD	•••••
5.1. $O_{\rm V}$	verview of FPGA board	35
5.2.1.	Overall Introduction of Hardware Resources	
5.2.2.	As the SoC prototype board directly	
5.2.3.	As regular FPGA board resource	
5.3. GENE	ERATE THE BITSTREAM FILE (MCS FORMAT)	41
5.4. Proc	GRAM THE BITSTREAM (MCS FORMAT) INTO FPGA	42

List of Figures

FIGURE 1-1 HUMMINGBIRD EVALUATION KIT AND HUMMINGBIRD DEBUGGER	
KIT	ATION KIT AND HUMMINGBIRD DEBUGGER
KIT	LUATION KIT AND HUMMINGBIRD
DEBUGGER KIT	IMMINGBIRD DEBUGGER
K _{IT}	8 Figure 2-2 The Ubuntu OS
RECOGNIZED USB	
THE HUMMINGBIRD DEBUGGER KIT'S DRIVER FOR WINDOWS	11 FIGURE 3-1 BOARD
HARDWARE RESOURCES	
CONNECTING HUMMINGBIRD FPGA BOARD AND DEBUGGER KIT WITH PC	
3-3 EXAMPLE OF USING OTHER BOARD RESOURCE TO CONTROL SOC PINS	
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	
MEMORY DEVICE	
FLASH Types	
CONFIGURATION FILES	23 FIGURE 4-1
BOARD HARDWARE RESOURCES	
2 CONNECTING MCU200T FPGA BOARD AND DEBUGGER KIT WITH PC	
4-3 THE FPGA JTAG CONNECTOR AND THE 12V POWER INTERFACE	
FIGURE 4-4 THE HARDWARE MANAGER OF VIVADO	
32 FIGURE 4-5 USE VIVADO HARDWARE MANAGER TO CONNECT	
FPGA	DD CONFIGURATION MEMORY
DEVICE	33 FIGURE 4-7 SELECT THE FLASH
Түреб	
CONFIGURATION FILES.	
BOARD HARDWARE RESOURCES	
CONNECTING DDR200T FPGA BOARD AND DEBUGGER KIT WITH PC	
3 THE FPGA JTAG INTERFACE AND THE 12V POWER INTERFACE	
5-4 THE HARDWARE MANAGER OF VIVADO	
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
T _{YPES}	
CONFIGURATION FILES	

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 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



... 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. ← → C 🔒 nucleisys.com/developboard.php QA UU 蜂鸟调试器 专为基于Nuclei处理器内核所实现的SoC原型及MCU芯片的开发调试而设计 • 一端为A型USB接口便于直连PC机,另一端为10PIN牛角插座,可直连蜂 鸟FPGA开发板 • 同时支持标准的四线JTAG、两线JTAG调试协议,并支持两线串口 (购买链接 一) (详细介绍 一 (下载驱动

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 UARTO 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, SWo 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 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

😣 🖨 🗐 🛛 Vivado 2015.4			
<u>File Flow Tools Window Help</u>			Q- Search commands
VIVADO.^>	Productivity. Multiplied.		E XILINX ALL PROGRAMMABLE.
Quick Start			Recent Projects
Create New Project	Open Project	Open Example Project	e200_fpga_proj /home/zhenbohu/jk_work/riscv/e20
Tasks			
Manage IP	Open Hardware Manager	Xilinx Tcl Store	
4			
Documentation and Tutorials	Quick Take Videos	Release Notes Guide	
Tcl Console			
File			

Figure 3-5 The Hardware Manager of Vivado



🙉 🖨 💿 🛛 Vivado 2015.4		
File Edit Flow Tools Window Lavout View	Help	Q- Search commands
👌 🕼 🖉 🖿 🖿 🗙 🕺 🚱 🖳 Logic Analyze	r	ashboard + (E)
Har Undo (Ctrl+Z) her - localhost/vilipy tcf/Digile	pt/21.031.9A2CE08A	
There are no dobug corres. Program double	Defrech device	^
There are no debug cores. Program device	<u>Kellesil device</u>	
Hardware	_ 0 2	×
< ă₩∎ ▶ ▶ ₩ ■		_
Name	Status	
	irogrammed	
Properties	- 0 2	×
← →		
Select an object to see pr	operties	
Tcl Console		- 🗆 🕑 ×
 INFO: [Labtoolstcl 44-466] Opening hw current hw_device [Lindex [get hw_dev C refresh hw_device -update hw_probes f INFO: [Labtools 27-1434] Device xc7a3 WARNING: [Labtools 27-3123] The debug Resolution: Awake sure the clock connected to t Canually launch hw_server with -e 	_target localhost:3121/xi ices] 0] alse [lindex [get_hw_devi 5t (JTAG device index = C hub core was not detecte he debug hub (dbg_hub) cc "set xsdb-user-bscan <c_l< td=""><td><pre>llinx_tcf/Digilent/210319A2CE08A _cces] 0])) is programmed with a design that has no sur id at User Scan Chain 1 or 3. rr is a free running clock and is active OR ISER_SCAN_CHAIN scan_chain_number>" to detect</pre></td></c_l<>	<pre>llinx_tcf/Digilent/210319A2CE08A _cces] 0])) is programmed with a design that has no sur id at User Scan Chain 1 or 3. rr is a free running clock and is active OR ISER_SCAN_CHAIN scan_chain_number>" to detect</pre>
Type a Ici command here		
🔚 Tcl Console 🖉 Messages		
Undo the last edit		

Figure 3-6 Use Vivado Hardware Manager to connect FPGA

😣 🖨 💷 Vivado 2015.4
Eile Edit Flow Tools Window Layout View Help
🚵 📾 💷 🗎 🐂 🗙 🙀 🖳 Logic Analyzer 💿 🗸 🗶 🗞 🔪 Dashboard 🗸 🍳
Hardware Manager · localhost/xilinx_tcf/Digilent/210319A2CE08A ×
There are no debug cores. Program device Refresh device
Hardware D 🖄 ×
Name Status Image: Status Connected Image: Status <td< td=""></td<>
@ Refresh Device
Hardware Device Add Configuration Memory Device
xc7a35t_0 Program BBR Key Clear BBR Key
Name: Program eFUSE Registers
Part: Export to Spreadsheet
D code: U3020093
General Properties
Tel Console – 🗆 Lé ×
 INFO: [Labtoolstcl 44-466] Opening hw target localhost:3121/xilinx_tcf/Digilent/210319A2CE08A Current hw device (lindex (get hw devices) 0) Prefresh hw device - update hw probes false [lindex (get hw devices) 0] INFO: [Labtools 27-1434] Device xc7a35t (JTAG device index = 0) is programmed with a design that has no su; WARNING: [Labtools 27-3123] The debug hub core was not detected at User Scan Chain 1 or 3. Resolution: Akek sure the clock connected to the debug hub (dbg_hub) core is a free running clock and is active OR Anaully launch hw_server with -e "set xsdb-user-bscan << USER_SCAN_CHAIN scan_chain_number>" to detect
Type a Tcl command here
Tcl Console O Messages
Open .

Figure 3-7 Add Configuration Memory Device



Add Configuration Mem	nory Device	t/210319A2CE08A		-	-	-	-
ce: ₩ xc/a35t_U							
Manufacturer All		-		Ту	pe All		
Density (Mb) All		-		Wic	th All		
t Configuration Memory Part		·					
t Configuration Memory Part earch: Q-n25		🖸 (8 matches)					
: Configuration Memory Part earch: Q- n25 Name	Part	(8 matches)	Alias	Family	Туре	Density (Mb)	Width
: Configuration Memory Part earch: Q-n25 Name n25q128-1.8%spix1_x2_x4	Part n25q128-1.8v	(8 matches) Manufacturer Micron	Alias	Family n25q	Type	Density (Mb)	Width x1_x2_x4
Configuration Memory Part earch: C n25 Name n25q128-1.8v-spi-x1_x2_x4 n25q128-3.3v-spi-x1_x2_x4 n25q128-3.3v-spi-x1_x2_x4	Part n25q128-1.8v n25q128-3.3v	 (8 matches) Manufacturer Micron Micron Micron 	Alias	Family n25q n25q	Type spi spi	Density (Mb) 128 128	Width x1_x2_x4 x1_x2_x4
Configuration Memory Part earch: Q- n25 Name N25q128-1.8V-spix1_x2_x4 n25q128-3.3V-spix1_x2_x4 n25q256-1.8V-spix1_x2_x4 n25q256-1.8V-spix4_x2_x4	Part n25q128-1.8v n25q128-3.3v n25q256-1.8v n25q256-3.2v	(8 matches) Manufacturer Micron Micron Micron	Alias	Family n25q n25q n25q n25q	Spi spi spi	Density (Mb) 128 128 256	Width x1 x2 x4 x1 x2 x4 x1 x2 x4
Configuration Memory Part aarch: <u>0: n25</u> <u>Name</u> n25q128-3.8v-spixd_x2_x4 n25q128-3.8v-spixd_x2_x4 n25q256-3.3v-spixd_x2_x4 n25q256-3.3v-spixd_x2_x4	Part n25q128-1.8v n25q128-3.3v n25q256-1.8v n25q256-3.3v n25q256-3.3v	(8 matches) Manufacturer Micron Micron Micron Micron	Alias	Eamily n25q n25q n25q n25q n25q	spi spi spi spi spi	Density (Mb) 128 128 256 256 32	Width x1_x2_x4 x1_x2_x4 x1_x2_x4 x1_x2_x4 x1_x2_x4
Configuration Memory Part earch: Q-n25 Name n25q128-1.8v-spixd_x2_x4 n25q128-3.3v-spixd_x2_x4 n25q256-1.8v-spixd_x2_x4 n25q256-3.3v-spixd_x2_x4 n25q32-3.3v-spixd_x2_x4	Part n25q128-1.8v n25q128-3.3v n25q256-1.8v n25q256-3.3v n25q32-1.8v n25q32-3.3v	(8 matches) Manufacturer Micron Micron Micron Micron Micron Micron Micron	Alias	Family n25q n25q n25q n25q n25q n25q n25q	Type spi spi spi spi spi spi spi	Density (Mb) 128 128 256 256 32 32	Width x1 x2 x4 x1 x2 x4 x1 x2 x4 x1 x2 x4 x1 x2 x4 x1 x2 x4 x1 x2 x4
t Configuration Memory Part earch: Q= n25 Name n25q1281.8v-spixd_v2_v4 n25q2561.8v-spixd_v2_v4 n25q2561.8v-spixd_v2_v4 n25q2561.8v-spixd_v2_v4 n25q321.8v-spixd_v2_v4 n25q323.3v-spixd_v2_v4	Part n25q128-1.8v n25q128-3.3v n25q256-1.8v n25q256-3.3v n25q32-1.8v n25q32-3.3v	(8 matches) Manufacturer Micron Micron Micron Micron Micron Micron	Alias	Family n25q n25q n25q n25q n25q n25q n25q n25q	Type spi spi spi spi spi spi	Density (Mb) 128 128 256 256 32 32 32 64	Width x1 x2 x4 x1 x2 x4
Configuration Memory Part earch: Q: n25 Name n25q128-1.8%-spi+d_x2_x4 n25q128-3.3%-spi+d_x2_x4 n25q256-3.3%-spi+d_x2_x4 n25q256-3.3%-spi+d_x2_x4 n25q323.4%-spi+d_y2_x4 n25q34-1.8%-spi+d_y2_x4 n25q64-3.3%-spi+d_x2_x4	Part n25q128-1.8v n25q266-1.8v n25q256-3.3v n25q32-1.8v n25q32-3.3v n25q64-1.8v n25q64-3.3v	(8 matches) Manufacturer Micron Micron Micron Micron Micron Micron Micron	Alias	Family n25q n25q n25q n25q n25q n25q n25q n25q	Type spi spi spi spi spi spi spi spi	Density (Mb) 128 256 256 32 32 64 64	Width x1_x2_x x1_x2_x x1_x2_x x1_x2_x x1_x2_x x1_x2_x x1_x2_x x1_x2_x x1_x2_x

Figure 3-8 Select the Flash Types

Memory Device:		
C <u>o</u> nfiguration file:		
PR <u>M</u> file:		
S <u>t</u> ate of non-config mem I/O pi	ns: Pull-none 🔻	
rogram Operations		
Address Range: Config	uration File Only 👻	
✓ Erase		
🔲 <u>B</u> l amk Check		
፼ P <u>r</u> ogram		
✓ Verify		
🗌 Verify <u>C</u> hecksum		
VF Options		
Create SVF Only (no pro	ram operations)	
SVF File:		
	·	

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 UARTO 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



🥝 🗇 🔍 Vivado 2015.4	
File Edit Flow Tools Window Layout View Help Q- Search commands	
🛃 📾 💵 📔 🛍 🗙 🎉 📴 Logic Analyzer 💿 🔻 🎉 🐟 🍡 🔊 Dashboard 🗸 🥸	
Hardware Manager - localhost/xilinx_tcf/Digilent/210319A2CE08A	×
I There are no debug cores. Program device Refresh device	
Hardware _ 🗆 🖄 🗡	
Name Status	
P localhost (1) Connected	
La XAL G Hardware Device Properties Ct/+E	
Program Device	
@ Refresh Device	
Hardware Device Add Configuration Memory Device	
←→ S Boot Trom Configuration Memory DeVice	
xc7a35t 0 Program BBR Kev	
Clear BBR Key	
Name: Program eFUSE Registers	
Part: Export to Spreadsheet	
ID code:	
General Properties	
Tcl Console _ D L	×
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210319A2CE08A [urrent_hw_device [lindex [get_hw_devices] 0] Orefresh_hw_device -update_hw_probes false [lindex [get_hw_devices] 0] INFO: [Labtools 27-1434] Device xc7a35t (JTAG device index = 0) is programmed with a design that has no sug WARNING: [Labtools 27-3123] The debug hub core was not detected at User Scan Chain 1 or 3.	-
Resolution: 1. Make sure the clock connected to the debug hub (dbg_hub) core is a free running clock and is active OR 2. Manually launch hw_server with -e "set xsdb-user-bscan <c_user_scan_chain scan_chain_number="">" to detect</c_user_scan_chain>	-
Type a Tol command have	-
	_
messages	
Open	

Figure 4-6 Add Configuration Memory Device

💿 🖨 🕘 Vivado 2015.4							
<u>File Edit Flow Tools Windo</u>	ow Layout ⊻iew	<u>H</u> elp			Q	- Search comm	hands
🔁 🙋 🖓 🖬 🦍 🕺 🧖	R Logic Analyzer	- X & Y	🔌 🖾 Da	shboard -	(2)		
		1/21 021 0420E004					~
Add Configuration Mem	ory Device	A A A A A A A A A A A A A A A A A A A					
Add Configuration Men	iory Device						
Choose a configuration mer	nory part. This car	be changed later.					
Device: 🚸 xc7a35t_0							
ter							
Manufacturer All		-		TV			
		-		(y			
		<u> </u>		WIC			
		Reset All F	ilters				
		.					
lect Configuration Memory Part							
Search: Q-n25		(8 matches)					
Name	Part	Manufacturer	Alias	Family	Type	Density (M	b) Width
n25q128-1.8v-spi-x1_x2_x4	n25q128-1.8v	Micron		n25q	spi	128	x1_x2_x4
n25q128-3.3v-spi-x1_x2_x4	n25q128-3.3v	Micron		n25q	spi	128	x1_x2_x4
n25q256-1.8v-spi-x1_x2_x4	n25q256-1.8v	Micron		n25q	spi	256	x1_x2_x4
n25q256-3.3v-spi-x1_x2_x4	n25q256-3.3v	Micron		n25q	spi	256	x1_x2_x4
n25q32-1.8v-spi-x1_x2_x4	n25q32-1.8v	Micron		n25q	spi	32	x1_x2_x4
n25q32-3.3v-spi-x1_x2_x4	n25q32-3.3v	Micron		n25q	spi	32	x1_x2_x4
n25q64-1.8v-spi-x1_x2_x4	n25q64-1.8v	Micron		n25q	spi	64	x1_x2_x4
n25q64-3.3v-spi-x1_x2_x4	n25q64-3.3v	Micron		n25q	spi	64	x1_x2_x4
						ОК	Cance
Type a rec command here							
🔚 Tcl Console 🛛 🔎 Messa	ges						

Figure 4-7 Select the Flash Types



🔒 💷 Program Configuration M	Memory Device					
Select a configuration file and set programming options.						
		-				
Memory Device:	Image: Image	-				
Configuration file:						
PRM file:		-				
State of non-config mem I/O pins:	Pull-none 👻					
Program Operations						
Address Range: Configurati	tion File Only 👻					
✓ Erase						
🔲 Blank Check						
✓ Program						
✓ Verify						
🗌 Verify <u>C</u> hecksum						
SVF Options						
□ Create <u>S</u> VF Only (no program	n operations)					
SVF File:						
	OK Cancel App	ly				

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 PA0_WKUP Push Buttons . (Mark:[31])
- GD32VF103 Debugger Select-jumper Hearder . (Mark:[32])
- GD32VF103 BOOT-jumper Hearder . (Mark:[33])
- GD32VF103 GPIO jumper Hearder(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 <u>http://user.nucleisys.com</u>.
- 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 UARTO 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 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

😣 🖨 🗊 Vivad	2015.4					
<u>File Edit</u> Flow	Tools Window Layout View Help Q- Search commands					
🔁 IO 🖉 🗎 🛙	👔 🗙 😼 🔚 Logic Analyzer 🛛 🔻 🕷 🔌 🔊 Dashboard 👻 🧐					
Hardware Man	Hardware Manager - localhost/xilinx_tcf/Digilent/210319A2CE08A					
There are no	There are no debug cores. Program device Refresh device					
Hardware	- 🗆 🕑 ×					
S 😂 📳 🔳						
ଡ଼-∎ localhost (1 ଡ଼-≣ø xilinx_tc ଡ଼-ଭ xc7a3: େହୁ XAE	Name Status Connected Diglient/210319A2CE08A (1) Open to (1) Pronrammed @ Hardware Device Properties Ctrl+E @ Program Device @ Refract Device					
Hardware Device	Add Configuration Memory Device					
← →	Boot from Configuration Memory Device					
xc7a35t_0	Program BBR Key					
Name:	Clear BBR Key					
Part:	Program eFUSE Registers					
ID code:						
General Prop	rties					
Tcl Console		Ŀ"×				
INFO: LLa current r efresh f INFO: LLa WARUING: Resolutio 1. Make s 2. Manual	<pre>toolstcl 44-466] Opening hw target localhost:3121/xilinx_tcf/Digilent/210319A2CE08A v_device [lindex [get_hw_devices] 0] v_device -update hw probes false [lindex [get_hw_devices] 0] tools 27-1434] Device xc7a35t (JTAG device index = 0) is programmed with a design that has no [Labtools 27-3123] The debug hub core was not detected at User Scan Chain 1 or 3. to the clock connected to the debug hub (dbg_hub) core is a free running clock and is active 0 y launch hw_server with -e "set xsdb-user-bscan <c_user_scan_chain scan_chain_number="">" to dete</c_user_scan_chain></pre>	sur Rect				
Type a Tol	omeand here					
Open						
open		1. 11				

Figure 5-6 Add Configuration Memory Device

🕲 🖨 🗉 Vivado 2015.4									
<u>Eile Edit Flow Tools Windo</u>	w Layout ⊻iew	Help			0	- Search comma	nds		
💌 🕼 🕼 📉 💥 🦉 🖳 Logic Analyzer 💿 🔻 🕷 🗞 🔊 Dashboard 🛪 🛞									
Hardware Manager localboct/killov tc/Dialent/2103198/2/5088 V									
D Choose a configuration memory part. This can be changed later.									
Device: ∲xc7a35t_0									
ter									
Manufacturer All		-		Tva	ae All		*		
Dapoitu (Mb) All		-		1466	th All				
Density (MD) All		<u> </u>		401CI					
		Reset All F	ilters						
lect Configuration Memory Part									
Annut D and		(o sector sec)							
Search: QC+125		(8 matches)							
Name	Part	Manufacturer	Alias	Family	Туре	Density (Mb)	Width		
9 n25q128-1.8v-spi-x1_x2_x4	n25q128-1.8v	Micron		n25q	spi	128	x1_x2_x4		
n25q128-3.3v-spi-x1_x2_x4	n25q128-3.3v	Micron		n25q	spi	128	x1_x2_x4		
9 n25q25b-1.8V-spi-x1_x2_x4	n25q256-1.8v	Micron		nz5q	spi	256	x1_x2_x4		
n25q25b-3.3v-spi-x1_x2_x4	n25q256-3.3v	Micron		n25q	spi	256	x1_x2_x4		
9 n25q32-1.8v-spi-x1_x2_x4	n25q32-1.8v	Micron		n25q	spi	32	x1_x2_x4		
n25q32-3.3v-spi-x1_x2_x4	n25q32-3.3v	Micron		n25q	spi	32	x1_x2_x4		
9 n25d64-1.8v-spi-x1_x2_x4	n25q64-1.8v	Micron		n25q	spi	64	H1_H2_H4		
n25q64-3.3v-spi-x1_x2_x4	n25q64-3.3v	Micron		n25q	spi	64	X1_X2_X4		
						ОК	Cancel		



😣 🗊 Program Configuration Memory Device						
Select a configuration file and set programming options.						
Memory Device: Configuration file: PRM_file: State of non-config mem I/O pins: Paramar Gameration	<pre> n25q128-3.3v-spi-x1_x2_x4 - - Pull-none * </pre>					
Address Range: Configuration Erase Blank Check Program Yerify Verify Checksum SVF Octions	on File Only 👻					
□ Create SVF Only (no program c SVF File:	operations)					
_	OK Cancel Apply					

Figure 5-8 Add the Configuration Files