Newton FPGA Platform Description and User Guide

TOF Imager

Exported on 02/12/2020

Table of Contents

1 Requirements

- Provide a vehicle for testing the HSP firmware with a relevant subset of the Newton design.
- The relevant subset includes:
    - SPI Slave
    - I2C Slave (optional)
    - APB Fabric
    - The following APB Fabric end-points:
        - Micro-sequencer
        - Analog Interface
        - SPI Master
        - MIPI Controller
        - eFuse Controller
        - LPS1
        - LPS2
        - Spread Spectrum PLL
        - PCM
        - Datapath
        - Dump Engine
    - eFuse array and controller
    - HSP
    - System TAP Controller
    - JTAG to APB Master
- Testing of the following external interfaces:
    - SPI Slave
    - I2S Slave (optional)
    - UART
    - JTAG (System TAP Controller and HSP)
- Provide a console program with commands for accessing the Newton FPGA functions.
- Provide C and Python programming interfaces.

2 Documentation Available on Raspberry Pi

A PDF version of this document is available on the Raspberry here : file:///home/pi/host_api/docs/pdf/Newton_FPGA_Platform_Description_and_User_Guide.pdf

3 FPGA Board

The FPGA board that is supported is the Xilinx Virtex UltraScale+ FPGA VCU118 Evaluation Kit. The user guide can be found here[1].

1 https://www.xilinx.com/support/documentation/boards_and_kits/vcu118/ug1224-vcu118-eval-bd.pdf

4 host_api

The host_api is the software that provides:

- Functions for accessing the Newton FPGA.
- C and Python programming interfaces.
- A console program with commands for accessing the Newton FPGA.

5 Raspberry Pi 4b

The Raspberry Pi 4b[2] hardware and supporting software provides the environment for testing the Newton FPGA. This environment will support all of the external interfaces listed in the requirements section.

Product Brief[3].

5.1 Software Supporting the External Interfaces

5.1.1 SPI Slave

The host_api uses the the default linux SPI driver. Documentation of the Raspberry Pi SPI interface can be found here[4].

5.1.2 I2C Slave

Use of the I2C interface is optional and not supported yet. *Documentation to be written.*

5.1.3 UART

Documentation to be written

5.1.4 JTAG

The host_api uses the openocd[5] software for providing an API for JTAG access.

5.2 External Interfaces

The external interfaces to the FPGA:

- SPI - Connected to Raspberry Pi
- UART - Can be connected to Raspberry Pi or some other device
- JTAG - We assume that the HSP firmware team will be connecting the JTAG interface to a USB to JTAG device, for example Fly Swatter2. We do have the option to drive this interface from the Raspberry Pi.

5.2.1 J8 Connector Pin Assignment

The following table shows the default pinout of the J8 connector of the Raspberry Pi 4b. Note that the JTAG pins TCK, TDO, TDI, TRST and TMS are not standard, but GPIOs 16, 19, 20, 21 and 26 will be used for this purpose. For this project we are using the default GPIO pinout.

The default pinout will be used.

2 https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
3 https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf
4 https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md
5 http://openocd.org/

Pin	GPIO / Function	Default Resistor State	Default
1	3v3		
2	5v		
3	2	High	SDA1
4	5v		
5	3	High	SCL1
6	GND		
7	4	High	GPCLK0
8	14	Low	TXD0
9	GND		
10	15	Low	RXD0
11	17	Low	HSP_INTERRUPT
12	18	Low	PWM0
13	27	Low	-
14	GND		
15	22	Low	-
16	23	Low	-
17	3v3		
18	24	Low	-
19	10	Low	SPI0_MOS1
20	GND		
21	9	Low	SPI0_MISO
22	25	Low	-
23	11	Low	SPI0_CLK
24	8	High	SPI0_CE0_N

25	GND		
26	7	High	SPI0_CE1_N
27	0	High	SDA0
28	1	High	SCL0
29	5	High	-
30	GND		
31	6	High	-
32	12	Low	PWM0
33	13	Low	PWM1
34	GND		
35	19	Low	TDO
36	16	Low	TCK
37	26	Low	TMS
38	20	Low	TRST
39	GND		
40	21	Low	TDI

Color coding of the cells in the table above:

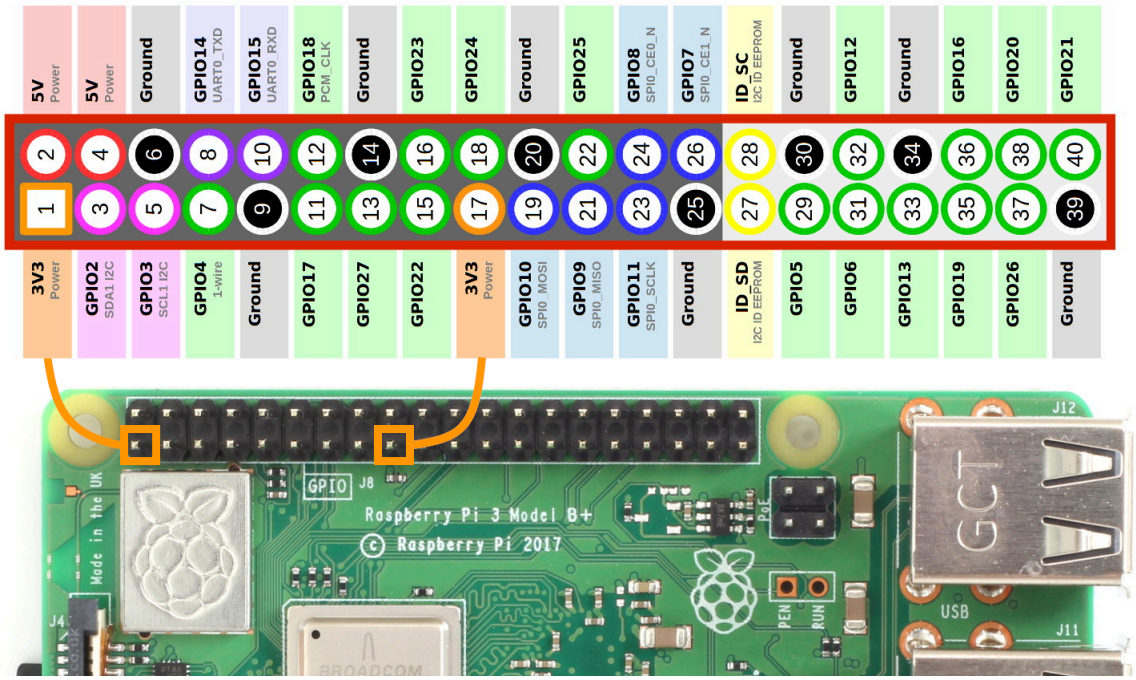
- Blue - UART interface
- Green - SPI Interface
- Red - I2C interface
- Yellow - JTAG interface
- Gray - HSP Interrupt

5.2.2 Connector J8 GPI Layout

3V3	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	TXD0
GND	9	10	RXD0
HSP_INTERRUPT	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3V3	17	18	GPIO24
SPI0_MOSI	19	20	GND
SPI0_MISO	21	22	GPIO25
SPI0_CLK	23	24	SPI0_CE0_N
GND	25	26	GPIO7
SDA0	27	28	SCL0
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
TDO	35	36	TCK
TMS	37	38	TRST
GND	39	40	TDI

5.2.3 GPIO Connector (J8) Image

The following image shows the layout of the GPIO connector (J8). Not that the colors in this image do NOT relate to the color coding used in the tables of this document.



5.2.4 Voltage Levels

The Raspberry Pi 4b and VCU118 FPGA use different IO voltage levels so voltage level shifters must be used. The Raspberry Pi 4b GPIOs are LVCMOS33 IOs. The The voltage levels of the VCU118 FPGA IOs depends on the voltage applied to the VADJ_1V8 rail pin. The valid values of the VADJ_1V8 rail are 1.2V, 1.5V, and 1.8V.

We plan to use the following level shifter board:

- Logic Level Shifter, 4-Channel, Bidirectional, item 2595[6] from Pololu Robotics and Electronics[7]

5.2.5 Connecting the FPGA board to the Raspberry Pi 4b

The FPGA board is connected to the Raspberry Pi 4b using off the shelf parts.

- FMC test boards for connecting to the FPGA board:
  - FMC XM105 Debug Card[8] from Xilinx[9]. XM105 Debug Card Image[10].
- Female to female fly wires for connecting the FMC board to the voltage level shifter boards.
- Female to female fly wires for connecting the voltage level shifter boards to connector J8 of the Raspberry Pi 4b.

6 https://www.pololu.com/product/2595
7 https://www.pololu.com
8 https://www.xilinx.com/support/documentation/boards_and_kits/ug537.pdf
9 https://www.xilinx.com
10 https://confluence.analog.com/download/attachments/121146016/fmc_105.png?api=v2&modificationDate=1580923982820&version=1

The following table shows the connections between the XM105 Debug Card and the J8 connector of the Raspberry Pi 4b.

	Signals	FPGA Pin	FMC PIN	FMC PIN NAME	X105 Pin name	Level Shifter	Level Shifter Pin		RPI J8 Pin	Wire Color	Notes
							LV	HV			
1	SPIS_SCLK	AL14	C26	FMC_LA27_P	J20-14	S0	L1	H1	23	red	
2	SPIS_MOSI	AK15	D26	FMC_LA26_P	J20-10	S0	L2	H2	19	yellow	
3	SPIS_MISO	AL15	D27	FMC_LA26_N	J20-12	S0	L3	H3	21	orange	
4	SPIS_SCS	AM14	C27	FMC_LA27_N	J20-16	S0	L4	H4	24	brown	
5	3v3					S0	--	HV	17		
6	1v8			-	J6-5 / J6-6 / J15-1	S0	LV	--	--		
7	UART_SOUT	BF9	D8	FMC_LA01_CC_P	J1-5	S1	L1	H1	10	green	
8	UART_SIN	BF10	D9	FMC_LA01_CC_N	J1-7	S1	L2	H2	8	blue	
9	HSP_INTERRUPT	BF15		FMC_LA08_N	J1-35	S1	L3	H3	11	purple	
10						S1	L4	H4	--		Spare level shifter
11	3v3					S1	--	HV	17		
12	1v8			-	J6-5 / J6-6 / J15-1	S1	LV	--	--		
13	FPGA_TDO_HSP_SP	AW7	C19	FMC_LA14_N	J1-20	S2	L1	H1	35	gray	

14	FPGA_TCK_HSP_SP	BC16	H20	FMC_LA15_N	J1-24	S2	L2	H2	36	white	
15	FPGA_TMS_HSP_SP	BB16	H19	FMC_LA15_P	J1-22	S2	L3	H3	37	black	
16	FPGA_TRST_HSP_SP	AV9	G18	FMC_LA16_P	J1-26	S2	L4	H4	38	brown	
17	3v3					S2	--	HV	17		
18	1v8				J6-5 / J6-6 / J15-1		LV	--	--		
19	FPGA_TDI_HSP_SP	AV8	G19	FMC_LA16_N	J1-28	S3	L1	H1	40	red	
20	GND				J16-3 / J16-4 / J15-2				25 / 20 / 30		
21						S3	L2	H2	--		Spare level shifter
22						S3	L3	H3	--		Spare level shifter
23						S3	L4	H4	--		Spare level shifter
24	3v3					S3	--	HV	17		
25	1v8					S3	LV	--	--		

6 Board setup steps

1. Make sure the FPGA board switches and jumpers are in default positions. Refer 'Board Setup and Configuration' section in user guide[11].
2. Connect XM105 to FMC HPC1 Connector J2. Turn on power switch (SW1) and check if 'Power Good LEDs' are on DS5 or DS7. If these LEDs are not on, refer instructions in step 3 to adjust the voltage on VCU118 board else proceed with step 4.
3. Enabling VADJ voltage to FMC on VCU118. Xilinx Answer Record has detailed description of the problem and solution (https://www.xilinx.com/support/answers/67308.html). Here is a summary instructions for the same.
    a. Install System controller utility. (Please refer Chapter 3 Board Component Descriptions, System Controller section for installation steps.)
    b. Connect USB cable to the UARt port and start the utility. Make sure the application is able to communicate with FPGA by clicking on get 'VADJ_1V8 voltage' option under Voltages. The application should display its value as 1.8V.
    c. Click on 'Set VADJ to 1.8V' option under FMC/set VADJ/ Current. Check if 'Power Good LEDs' on XM105 are on.
    d. If step iii is successful, then you can use 'Set VADJ to 1.8V' option in FMC/Set VADJ/Boot-up to preserve the settings after power cycle (optional but useful).
    e. Measure voltage available on FMC pinout. Pin J6-5, J6-6 and J15-1 are connected to 1.8V while J16-3, J16-4, J15-2 are connected to ground.
4. Please check if the level shifters are working properly or not.
5. Make connections between FMC, level shifter and Raspberry Pi as mentioned in the table above. (Please follow wire color coding mentioned in the table).
6. Program the bit file in FPGA.
7. After programming is finished GPIO_LED[0], GPIO_LED[1], GPIO_LED[2] should start blinking.
    a. GPIO_LED[0] : Indicates crystal clock is running. (12 MHz).
    b. GPIO_LED[1] : Indicates system pll clock is running. (24 MHz).
    c. GPIO_LED[2] : Indicates debug core clock is running.
    d. GPIO_LED[3] : Backdoor is enabled and path is active.
8. Use newton_control to load efuse and ROM memories. Reset the FPGA using push-button SW6 or reset HSP by writing 1 to address 80 using spi_write command. This control bit is not self clearing hence it must be set to 0 to remove reset on HSP.

Note : Please take precaution when booting or rebooting raspberry pi after all the connections are made (FPGA ↔ Level Shifter ↔ Raspberry Pi). The pin direction of GPIO pins don't match to pin directions of the design on FPGA. This might damage the level shifter. The workaround for this problem is to make changes in raspberry pi boot up. This workaround is not tried or tested.

FPGA debug probes.

There are two APB bus which are connected to the debug core.

1. APB bus connected to HSP mailbox.
2. APB bus connected to different slaves in Newton controlled by HSP.

In addition to these probes, there are probes on following signals.

1. SiFive BReset
2. SiFive Exception register (reg_mcause_code[9:0])
3. SiFive PC (Only bits 29 to 0 are connected).
4. SPRom0En

11 https://www.xilinx.com/support/documentation/boards_and_kits/vcu118/ug1224-vcu118-eval-bd.pdf

5. HSP ROM address
6. HSP ROM data (38 bit)
7. SPRom0Data (32 bit)

Note: The design has a virtual input / output core which controls polarity of TRST signal connected to SiFive core. The default value of this input is set to 1 which might affect JTAG working. If required please change it to 0.

FPGA Backdoor

Use load command in newton_control to load FMC efuse memory, ROM and RAM memory. The HSP logic can be put in reset state during backdoor load by writing 1 to address 80 using spi_write command. This control bit is not self clearing hence it must be set to 0 to remove reset on HSP. The FPGA backdoor path always remains accessible. Hence the HSP memories can be written and read back at any time using commands provided in newton_control application.

The backdoor logic also has a debug register (Address 84) to which some of the output signals of HSP are connected.

debug register bit	HSP output signal
0	hsp_mbox_interrupt_o[0]
1	hsp_mbox_interrupt_o[1]
2	fir_datal_err_o
3	-
4	aeb_mst_dbg_o
5	aeb_fuse_dft_en
6	aeb_top_dft_en
7	aeb_hsp_dft_en

7 Installing Software on Raspberry Pi 4b Board

The following is a list of software to be installed on the Raspberry Pi 4b base software:

- Android cutils
- JSON Parser[12]
- WiringPi[13]
- swig[14]
- openocd[15]
- i2c-tools

7.1 Android cutils

Directions for installing Android cutils:

- cd ~/Downloads
- wget https://android.googlesource.com/platform/system/core/+archive/master/liblog.tar.gz
- gunzip liblog.tar.gz
- cd ../
- mkdir libcutils
- cd libcutils
- tar vxf ~/Downloads/liblog.tar

7.2 JSON Parser

Directions for installing json-parser

- cd
- git clone https://github.com/udp/json-parser
- cd /home/$USER/json-parser
- mkdir -p /home/$USER/json-parser-install/lib
- mkdir -p /home/$USER/json-parser-install/include
- ./configure -prefix=/home/$USER/json-parser-install
- make
- make install-static

7.3 WiringPi

WiringPi is already installed. If not execute the following:

- cd
- sudo apt-get install wiringpi

7.4 SWIG

Directions for installing SWIG:

12 https://github.com/udp/json-parser
13 http://wiringpi.com/
14 http://www.swig.org/
15 http://openocd.org/

- cd
- sudo apt-get install swig

7.5 OPENOCD

Directions for installing openocd:

- cd
- sudo apt-get update
- sudo apt-get install -y autoconf libtool libftdi-dev
- git clone --recursive git://git.code.sf.net/p/openocd/code openocd-git
- cd openocd-git
- # Follow directions in README
- ./bootstrap
- ./configure
- make
- sudo make install

7.6 i2c-tools

Directions for installing i2c-tools:

- cd
- sudo apt-get install -y i2c-tools

7.7 Newton FPGA host_api

7.7.1 Documentation on Raspberry Pi

A HTML version of the host_api documentation is available on the Raspberry here : file:///home/pi/host_api/docs/doxygen/html/index.html

7.7.2 C API and newton_control application

The newton_control is a console program for interacting with the Newton FPGA via SPI. To build the newton_control application perform the following steps

- ssh pi@<ip_address>
- cd host_spi/examples/c
- make clean
- make

7.7.3 Python API and Examples

To be written.

7.7.4 Console Application and Commands

Usage:

- newton_control.exe load target file_name
- newton_control.exe verify target file_name
- newton_control.exe unload target file_name
- newton_control.exe spi_write address data
- newton_control.exe spi_read address
- newton_control.exe reg_write address data
- newton_control.exe reg_read address
- newton_control.exe reg_dump address word_count
- newton_control.exe reg_fill address data word_count
- newton_control.exe console

Where:

- load : Loads the contents of the specified file into the target memory.
- verify : Compares the contents of the specified file to the contents of the target memory.
- unload : Dumps the content of the target memory to the specified file.
- target is one of the following:
    - useq_seq_ram : Microsequencer Sequence RAM
    - useq_map_ram : Microsequencer MAP RAM
    - useq_wave_ram : Microsequencer Wave RAM
    - datapath_ram : Gain Correction RAM
    - de_ram : Dump Engine RAM
    - lps1_ram : LPS1
    - lps2_ram : LPS2
    - hsp_rom : HSP ROM (FPGA only)
    - hsp_ram : HSP RAM (FPGA only)
    - efuse : eFuse (FPGA only)
- spi_write : Write data to the address. For accessing HSP MBOX registers.
- spi_read : Read data at address. For accessing HSP MBOX registers.
- reg_write : Write data to the Newton register at address
- reg_read : Read the Newton register at address
- reg_dump : Read and display word_count Newton registers starting at address.
- reg_fill : Write word_count word_count Newton registers with data starting at address.
- console : Enter console mode

Options:

--help Shows this help message.

7.7.5 Command Line Examples

Here are a few examples of the command line interface:

- Loading new HSP ROM and eFuse data:
    - newton_control.exe load hsp_rom ../../firmware/ADI_Firmware_Drop_0.4/ADI_HSP_ROM_0.4.4.vhx
    - newton_control.exe load efuse ../../efuse/hsp_fuse_file.vhx
- Writing the HSP mailbox S2H_MBX_VALID register:
    - spi_write 8 1

- Reading the HSP mailbox S2H_MBX_FIFO_CNT register:
    - spi_read a
- Writing the newton microsequencer gprR0 register:
    - reg_write 60 01234
- Reading the newton microsequencer gprR0 register:
    - reg_read 60

7.7.6 Directions for Installing the host_api

To install the host_api on the Raspberry Pi:

- scp -r -p host_api pi@<ip_address>:/home/pi
- ssh pi@<ip_address>