Everest UART to SPI-FLASH Demo

Getting Started

Project: Everest UART to SPI-FLASH Demo Getting Started	created:	S. Rieche	Date	2018-12-18
	edited:	S. Rieche	Date:	2019-01-29
	approved:		Date:	
Filename:	LAB9 Getting_Started_v1p1.docx			
Arrow Central Europe GmbH	Version:	1.1	Page 1 of 13	

Contents

Figures

Tables

1. Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 1.1

The document was updated for Libero SoC v12.0.

1.2 Revision 1.0

Revision 1.0 is the first publication of this document.

2. Getting Started

The proprietary Debug-SPI-Flash-Tool tool can be used to execute *device-id*/*erase*/*read*/*write* operations on configuration SPI flash devices currently not supported by the Libero Software. A UART-to-SPI flash client running on a Mi-V Soft CPU is required to be flashed on the FPGA Board. The Host (PC) side uses Phython scripts to exchange commands and binary data with the UART-to-SPI flash client in the FPGA.

2.1 Prerequisites

For the Everest Tiny Yolo v2.0 Demo the following is needed:

Item	**Quantity**
Everest DEV Board	1
12 V / 5 A wall-mounted power adapter	1
USB 2.0 A male to mini-USB B cable for UART / Programming interface to PC	1
Flash Pro PolarFire v2.0 or later	1

Note 1: The Everest DEV Board offers an on-board FlashPro5 programmer, which can be used to program and debug with Identify, SmartDebug and embedded application software using SoftConsole.

Note 2: This tool should only be used for debugging purposes, the official tool for SPI Flash programming is the Libero Software.

2.2 Handling the Board

Pay attention to the following points while handling or operating the board:

Handle the board with electrostatic discharge (ESD) precautions to avoid damage.

For information about ESD precautions see https://www.microsemi.com/documentportal/doc_view/126483-esd-appnote.

2.3 Board-Setup Revision PROTO

2.3.1 Toggle-Switch S1 – PCIe

Warning: S1-1 and S1-2 must not be at position on at the same time!

SWITCH ON	**PCIe LANES**
S1-1	x1
S1-2	x4

2.3.2 Toggle -Switch S5 – SC SPI-Flash enable

Warning: S5-1 and S5-2 must not be at position on at the same time!

SWITCH ON	**SC SPI-FLASH**
S5-1	ENABLE
S5-2	DISABLE

2.3.3 DIP-Switch S8 – FMC Voltage Selector

Warning: S8-1 to S8-4 must not be at position on at the same time!

SWITCH ON	**FMC VOLTAGE**
S8-1	3.3 V
S8-2	2.5 V
S8-3	1.8 V
S8-4	undefined (not connected)

2.3.4 Toggle -Switch S9 – VDDAUX2 & VDDAUX5 Voltage

Warning: S9-1 and S9-2 must not be at position on at the same time!

SWITCH ON	**VDDAUX2 & VDDAUX5**
S9-1	2.5 V
S9-2	FMC voltage

2.4 Board-Setup Revision A and B

2.4.1 Toggle-Switch S1 – PCIe

SWITCH	**PCIe LANES**
S1-1 (marking)	x4
S1-2	x1

2.4.2 Toggle -Switch S5 – SC SPI-Flash enable

SWITCH	**SC SPI-FLASH**
S5-1 (marking)	DISABLE
S5-2	ENABLE

2.4.3 DIP-Switch S8 – FMC Voltage Selector

SWITCH	**FMC VOLTAGE**
S8-1 off, S8-2 off	1.8 V
S8-1 on, S8-2 off	2.5 V
S8-1 off, S8-2 on	undefined (not recommended)
S8-1 on, S8-2 on	3.3 V

2.4.4 Toggle -Switch S9 – VDDAUX2 & VDDAUX5 Voltage

SWITCH	**VDDAUX2 & VDDAUX5**
S9-1 (marking)	2.5 V
S9-2	FMC voltage

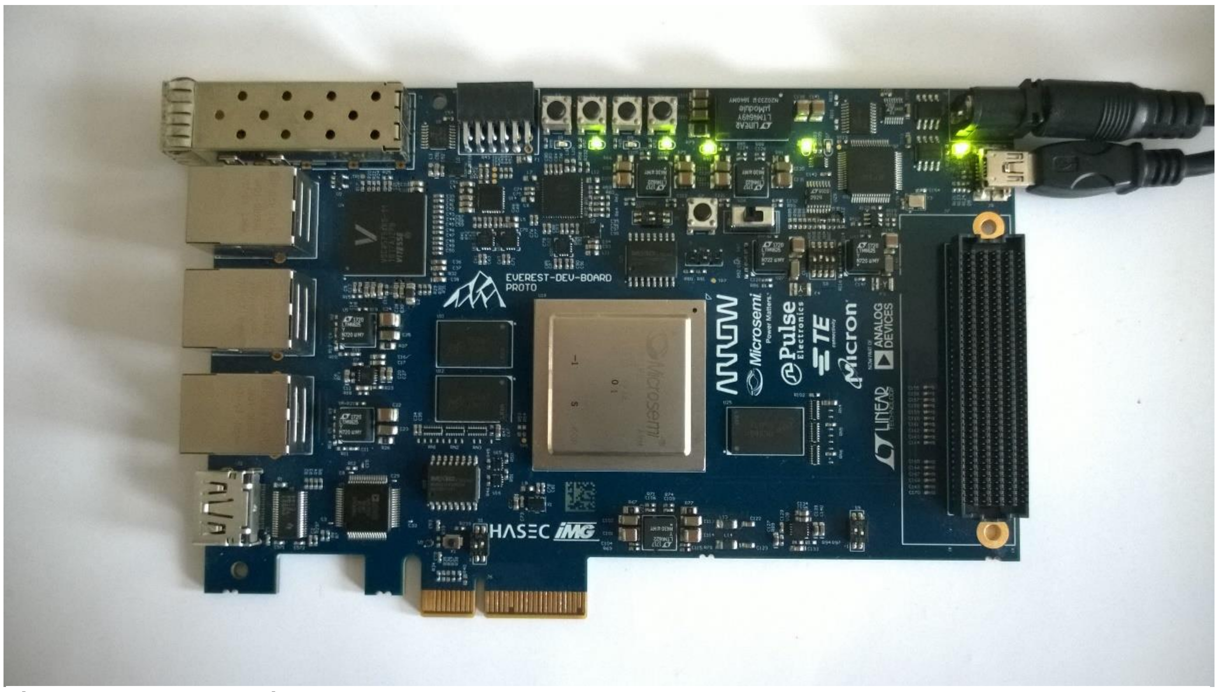
Figure 1: Everest Board

2.5 Powering up the Board

The Everest DEV Board is powered up using the 12 V DC jack. For programming connect it although with your computer using USB mini B connector J9.

3. Demo Design

3.1 Prerequisites

Table 1: Software / IP Requirements

Software	**Version**
Libero SoC PolarFire	V12.0
Synplify Pro	L2017.09M-SP1-1
FlashPro PolarFire	V2.0
IP	
PF_CCC	1.0.115
PF_INIT_MONITOR	2.0.103
PF_OSC	1.0.102
CoreAHBLite	5.3.101
PF_SRAM_ABHL_AXI	1.1.127
CoreAHBTOAPB3	3.1.100
CoreAPB3	4.1.100
CORESPI	5.2.104
CoreGPIO	3.2.102
CoreUARTapb	5.6.102
COREJTAGDEBUG	3.0.100
MIV_RV32IMA_L1_AHB	2.3.100

Before you start you have to make sure, that all cores are downloaded to your local vault.

3.2 FPGA-Design

The FPGA-Design consists mainly of a Mi-V-CPU and a JTAG, UART- and SPI-interface. To get the FPGA-Design into the PolarFire FPGA, you could either use the Libero PolarFire project files to generate the bitstream or the precompiled staple-file in the "*Bitstream*" folder, that has to be programmed with the FlashPro tool.
The main functionality of the UART-to-SPI-Bridge is done by the firmware, executed by the Mi-V-CPU.

3.3 Firmware

The firmware is a SoftConsole v5.2 project and could be found in the folder "*SoftConsole*". Please open the SoftConsole project, compile it and execute it by starting a debug session.

The Everest DEV Board is now ready to listen on the UART-interface and waiting for commands and data.

3.4 Determine the COM Port Number

Please open the *device manager* on your computer to find out the com port numbers, that were used by the operating system for your Everest DEV Board. You should see at least four ascending numbers, like three, four, five and six. In this case, com port number five should be used. If this dose not work, try number six.

3.5 Python Interpreter

Please check out, if a *Python Interpreter* (3.7.0 or above) is already installed on your computer. If not, you can download and install it from:
https://www.python.org/downloads/release/python-370/

Than please install the additional *xmodem* and *pyserial packeges*. They are located in the subfolder "*flashtool\Addon_Python_Packages*".

1. Change to folder "*xmodem-0.4.4*"
2. Double-click on "*setup.py*"
3. Change to folder "*pyserial-3.0.1*"
4. Double-click on "*setup.py*"

You can also use your command prompt by typing:

```
python setup.py install
```

4. Usage Instructions

4.1 General Notes

The Debug-SPI-Flash-Tool supports the following four operations:

- Read SPI Device ID Registers
- Erase/Read/Program, in multiples of 4096 Byte chunks

The four LEDs on the Everest DEV Board will inform you about the state of the current operation in the following way:

- One LED will alternate during the selected operation
- All four LEDs will alternate on Error conditions
- All four LEDs will be constant-on after successful operation

Attention: Adjust the serial com port used for Mi-V communication in the example batch-files (com43) to match your PC.

4.2 Read SPI Device Identification Registers

If you execute the example: "*SPI_READ_DEVICE_ID.bat*", you should see the following output on your command prompt for the Macronix Flash MX66U1G45G:

```
Manufacturer-ID: 0xc2
Device-ID: 0x3b
Device-Capacity: 0x3b
```

Please refer to the Macronix device datasheet for details (table 11: ID Definitions, RDID 0x9F).

4.3 Read SPI Flash

The example batch file "*SPI_READ_1MByte_0x400.bat*" uses com port 5 to read 1 MByte in 256 chunks of 4 kByte, starting at address 0x400. The output will be written into the file "*outfile.bin*" in the subfolder "*file_output*".

```
python ./scripts/flash_read_chunks.py com5 0x00000400 256 outfile.bin
```

The general syntax is:

```
flash_read_chunks.py <com-port> <start-address> <chunks> <output-file>
```

4.4 Erase SPI Flash (required before Program operation)

Batch-file "*SPI_ERASE_1MByte_0x0.bat*" uses com port 5 to erase 1 MByte in 256 chunks of 4 kByte, starting at address 0x0.

```
python ./scripts/flash_erase_chunks.py com5 0x00000000 256
```

The general syntax is:

```
flash_erase_chunks.py <com-port> <start-address> <chunks>
```

While erasing the indicator LED is alternating. When the process is finished all four LEDs will be ON. If the erasing process was not successful, all four LEDs will alternate.

Note: The start-address for erase needs to be on 4KByte Boundary, so it will be masked with 0xFFFF0000.

4.5 Program SPI Flash

The example batch file “*SPI_PROGRAM_1MByte_0x400.bat*” uses com port 5 to write the file “*./init_data/test1MByte.bin*” into the SPI-FLASH, starting at address 0x400.

The general syntax is:

```
flash_write_chunks.py <com-port> <start-address> <output-file>
```

Note: Make sure to erase at least a 1 MByte portion of the flash before programming, otherwise the data will not be programmed correctly.

4.6 Expected Performance

Table 2 shows the expected performance:

Table 2: expected performance of SPF-FLASH operations

Size	**Erase**	**Program**	**Read**
128 kByte	< 1 s	17 s	19 s
1 MByte	2 s	2 min, 16 s	1 min, 30 s
8 MByte	1 min	18 min, 3 s	18 min, 37 s