AXI-FX3-Interface v1.2

Cesys IP Core Product Guide for AXI-FX3-Interface v1.2.

IP Facts

Introduction

The AXI-FX3-Interface v1.2 IP core facilitates an easy and efficient connectivity between a host PC with Superspeed Universal Serial Bus (USB3.0) and a number of AXI-based FPGA peripherals. It is designed to be used in Xilinx's Vivado design Suite. The IP provides a ready solution for USB by interfacing the widely used Cypress FX3's slave FIFOs and AXI4-Memory-Mapped peripherals like Memory Interface Generator, Block RAM Controller, General Purpose IOs, Quad SPI etc. Such peripherals are readily available in the Vivado Block design methodology. The IP core can be used in FPGA designs where an interface between FX3-slave FIFOs and AXI4 based peripherals is necessary. The IP Core is designed to be used with Cesys GmbH's Embedded FPGA Module "EFM03 Beastboard" (https://www.cesys.com/en/our-products/fpga-boards/efm-03.html). It can also be used in custom FPGA designs requiring a solution between FX3 GPIF-II and AXI interfaces.

The AXI-FX3-Interface v1.2 IP Core is part of Cesys's Unified Development Kit (UDK3) that is available with the EFM03 Beastboard module. The host-software can read and write all the functional blocks of the reference design by calling the read or write API functions using UDK3. The referenced addresses are embedded in the UDK3 protocol, transferred serially over the Universal Serial Bus (USB) and are stored intermediately in the FX3's Slave FIFOs. The AXI-FX3-Interface module reads/writes these Slave FIFOs over a General Programmable Interface (GPIF-II) and translates the commands/data into AXI4-Full cycles to access the on-chip registers or on-board peripherals.

Advantages

The IP Core has several advantages like:

1. Provides ready solution for systems using Superspeed USB (USB-3.0)
2. Saves valuable amount of development time in custom USB-based designs
3. Interfaces 2 widely used technologies; AXI bus system and FX3 Chipset
4. Facilitates high data-rates upto 315 MB/s for Read and Write accesses
5. Proven stable performance for long durations with rigorous tests
6. Backward compatibility with USB2.0

Basic Information

IP Information

Particular	Further Details
Supported Devices	Xilinx 7 Series FPGAs
Supported User Interface	AXI4-Memory-Mapped and GPIF-II
Supported Tool	Vivado IDE 2017.2.1 onwards

Resource Utilization

This IP-Core is synthesized and implemented in Vivado version 2017.2.1. It was tested on the "xc7a200tfbg676-2" part on Cesys EFM03 Beastboard. The Post-implementation Resource Utilization report can be seen in the below table.

Resource	Utilization
LUT	1797
LUTRAM	175
FF	2023
BRAM	26.50
IO	261
BUFG	2

What is provided with the core?

Particular	Further Details
Design files	Encrypted IP (VHDL)
Reference design	Block Design + VHDL wrapper
Simulation files	Block Design + VHDL wrapper + VHDL test-bench
Constraints files	Xilinx design Constraits (XDC)
Support	Provided by Cesys @ http://cesys.com/download/fpga/

Features

- Supports AXI-4 Memory-Mapped Interface specifications
- Supports FX3's GPIF-II slave FIFO interface
- Uses DMA transfers to transfer data to and from Memory-mapped peripherals

- Supports transfer sizes of upto 8 Megabytes
- Supports AXI-4 Full and Lite protocols
- Connects seamlessly with AXI based peripherals available in Xilinx LOGICore IP repository
- Supports all Xilinx 7 series FPGA parts

Applications

- High performance PC peripherals
- Image processing
- DSP co-processing
- Embedded Control
- Custom test equipment
- Data acquisition daughter cards

Quick Start

Obtaining the IP-Core files

The AXI-FX3-Interface v1.2 IP Core is available as part of the Cesys UDK3 kit that is delivered with the Cesys EFM03 Beastboard. The core's design files and the reference design are packed in an archive "efm03_reference_design-v1.2.1.zip". Using the login information that you received with your board, please download the archive from our website (https://www.cesys.com).

Adding the AXI-FX3-Interface v1.2 IP to the IP Catalog

The IP core can be added to the Vivado IP Catalog by following the below steps:

1. Create a new Vivado project or open the Vivado project in which you want to add the AXI-FX3-Interface v1.2 IP
2. In the **Flow Navigator** window, under **Project Manager** tab, open **IP Catalog**
3. In the IP Catalog Window, Right-Click and select **Add Repository...** option
4. Navigate to the location where you have unzipped the above mentioned archive
5. Select **Cesys_IP_repo** and press **Select**

Now, in the **IP Catalog** window, you must be able to see a section called User Repository. Under **User Repository->UserIP** you will find **axi_fx3_if_v1_2**. You can now close the IP Catalog window. To test if the IP is added or not, follow the below steps:

1. Select **Create Block design** under **IP Integrator** tab in **Flow Navigator** window
2. If you already have an existing Block design open, move to the **Diagram** window
3. Right click on the canvas and select **Add IP...** option (Ctrl + I)
4. In the IP selection box, search for or navigate through the list to find *axi_fx3_if_v1_2*
5. Select *axi_fx3_if_v1_2* and the IP should appear on the canvas

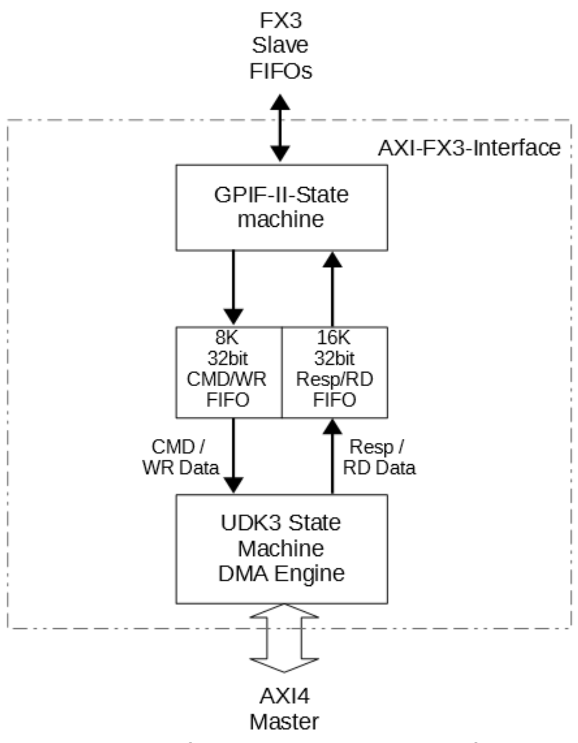
Overview

This chapter briefly describes the IP-Core and its internal structure.

Functional Description

Current FPGA design trends in Xilinx have mostly moved towards the block design methodology. The AXI-FX3-Interface v1.2 IP Core is designed to fit into this trend by providing an easy and efficient translation between the FX3's GPIF-II slave FIFO interface and the widely used AXI4 bus system. The IP core communicates with the FX3 slave FIFOs over a GPIF-II interface, interprets the commands issued by the host using the UDK3 protocol (See also **AN101** for *UDK3 api specifications*) and issues DMA transfers to read from or write to AXI4 peripherals.

The figure below shows the internal structure of the IP core.



Broad overview of the internal structure of AXI-FX3-Interface v1.2 IP-Core

GPIF-II State Machine

The Cypress FX3 USB controller transfers data to its Master over a slave FIFO interface and this is called the 2nd generation General Programmable Interface (GPIF-II). The FX3 controller sets output flags to indicate the slave FIFO levels to the Master requesting or giving data to it. The AXI-FX3-Interface v1.2 IP Core is the master and based on the FX3's flags, it reads the FX3 slave FIFOs if a Host-2-Peripheral command is issued and stores it in the CMD/Write Data FIFO. If a Peripheral-2-Host command is issued, the IP Core checks the FX3's FIFO levels and if there is an 8Kbyte space available, it initiates a read transfer to FX3 slave FIFOs.

CMD/Write and Resp/Read FIFOs

The IP-Core contains an 8K deep (CMD/Write FIFO) and a 16K deep FIFO (Resp/Read FIFO), both with a data-width of 32-bits. Therefore, each of the FIFOs can hold atleast 4 complete UDK3 frames of size 8KByte. The GPIF-II State machine writes the commands or write data to the CMD/Write FIFO and reads peripheral data through the Resp/Read Data FIFO. These FIFOs are setup to hold atleast one complete USB3.0 Data field of size 8Kbyte at a time. The FIFOs are configured as "Independent Clocks, Block RAM" so that they operate at independent Read and Write clocks and instantiates Block RAMs for storage. The transaction begins as soon as there is data in the CMD/Write Data FIFO. It is the responsibility of the user to provide 8KByte frames.

UDK3 State machine

This module decodes the Cesys UDK3 protocol (See also **AN101** for *UDK3 api specifications*), issues commands and provides data to the DMA engine. Based on the commands issued and the data given, the DMA Engine initiates a transfer to or from the addressed peripheral.

If the host issues a Host-2-Peripheral command, then the command is followed by data. The UDK3 state machine reads the CMD/Write Data FIFO and decodes the command. It then waits for the data to arrive. As soon as the data is in the CMD/Write Data FIFO, the UDK3 state-machine isssues a WRITE command to the DMA engine. The DMA engine then reads data directly from the Write Data FIFO and transfers it to the Peripheral which was addressed in the command.

If the host issues a Peripheral-2-Host command, then the GPIF-II stores this command in the CMD/Write Data FIFO. The UDK3 State-machine reads this command and issues a READ command to the DMA Engine. The DMA engine in turn reads the Peripheral addressed by the command and stores the read data directly in the Resp/Read Data FIFO.

According to the UDK3 protocol, each transfer is 8KByte long and if the whole 8KByte is not used, the remaining data fields must be stuffed with DUMMY values (0x00). For a CMD/Write transfer, the DUMMY insertion has to be done by the HOST. For Resp/Read transfers, the UDK3 State-machine does this automatically once the requested number of bytes has been written into the Resp/Read Data FIFO.

Elimination of Read-Write Collision

The host PC sends commands and data to (or expects data from) the FPGA design sequentially over a USB interface. The DMA Engine on the other hand, consists of separate FIFOs for both, Read and Write accesses, thus facilitating an actual parallel execution of Read and Write accesses which is a characteristic feature of the AXI Bus system. However, most peripheral modules consist of one single bus to Read or Write to the on-board peripheral and do not support parallel execution of Read and Write accesses. When a slower peripheral is connected to the system and a Write followed by a Read command is executed, a Read-Write collision may occur at the AXI-Slave Peripheral. This is because a Write access is still in progress and a Read access is serviced by the AXI Slave. This collision occurs especially when writing to very slow peripherals over an AXI-Lite Interface such as a Flash Controller, in which write accesses typically take 10-15us to complete.

When an AXI-Lite Slave is connected to the Master interface of the AXI Interconnect, the interconnect automatically instantiates an AXI Protocol Converter (auto PC) in order to translate AXI-Full bus cycles (from DMA engine) into AXI-Lite cycles. The auto PC has a latency of 2 AWREADY-AWVALID handshakes (or ARREADY-ARVALID handshakes). Such an implementation may cause a Read-Write collision between the last write and the first read access at the peripheral's AXI Slave Interface which is not in the AXI-FX3-Interface v1.2 IP Core's control.

The UDK3 state machine eliminates any collision that could be caused at the AXI-FX3-Interface v1.2 IP Core's Master which is driving the AXI Interconnect. It is acheived by ensuring a sequential execution of host commands on the DMA's Memory-Map write interface(M_AXI_S2MM). This is accomplished by simply waiting for the DMA Engine to completely forward the current write access to the AXI Interconnect. Only after the DMA engine signals the end of the write transfer (xfer_cmplt), the next command will be processed. However, when an AXI4-Lite interface is connected to the master interface of the AXI Interconnect, as explained earlier, it is out of scope of the UDK3 state machine. The Slave Peripheral should not accept read accesses when write accesses are present at the interface.

In custom peripherals based on AXI-Lite Slave interface, users can acheive sequential accesses by waiting for the write transfers to complete before asserting the 'ready' signal of the Read Address channel (ARREADY) of the Slave AXI Bus. This will stall the read access until the peripheral has finished writing. An example code is shown below:

```
if (axi_arready = '0' and S_AXI_ARVALID = '1'  -- Previous handshake completed
  and  peripheral_ready = '1'                  -- peripheral is ready
  and S_AXI_AWVALID = '0' and axi_awready = '0') then  -- when no writes (SEQ)
  -- slave has accepted valid RD_addr
  axi_arready <= '1';
  axi_araddr  <= S_AXI_ARADDR;
else
  axi_arready <= '0';
end if;
```

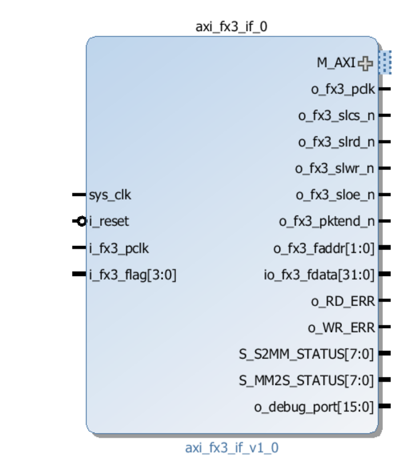
The above code snippet is an extraction from the AXI-Lite Slave interface to generate the ARREADY signal. It is generated by Xilinx Vivado tool. The line marked 'SEQ' ensures that the read access is accepted only when there are no more writes at the AXI-Slave interface.

The DMA Engine

The DMA Engine used in this IP Core converts AXI4 Streaming data to AXI4 Memory-mapped protocol and vice versa, based on the commands issued to it. The data for a Host-2-Peripheral transfer is provided by the CMD/Write Data FIFO. The UDK3 State-machine provides the FIFO data to the DMA engine over an AXI4 streaming protocol. The DMA engine writes this data to the peripherals via an AXI-Interconnect based on AXI4-Full protocol. After it has transferred the write data completely onto the AXI-Interconnect, the DMA Engine signals the end of transfers (xfer_cmplt) signal to the AXI-FX3-Interface and waits for the next command. For a Peripheral-2-Host transfer, the DMA Engine reads the peripherals over AXI4-Full protocol and provides this data to the UDK3 State-machine over an AXI4 streaming interface. The DMA generates a 'tlast' (AXI4-Streaming interface) signal when it is transferring the last 32-bit data which signals the end of transfer during a read access.

Port Description

The AXI-FX3-Interface IP Core looks as shown in the figure below:



The AXI-FX3-Interface v1.2 IP-Core

AXI-FX3-Interface Port Description.

Signal	Interface	I/O	Default	Description
sys_clk	Clock	I	NA	System Clock.
i_fx3_pclk	Clock	I	NA	FX3 GPIF state machine clock. This clock can either externally be provided or the on-board loop-back clock can be used. Please Refer Hardware reference guide for EFM03 Beastboard for further details. This is a 100MHz clock.
i_reset	Reset	I	NA	System Reset, active Low.
i_fx3_flag (3:0)		I	0000	Flags from the FX3 slave FIFOs. They can be programmed to display different levels of the slave FIFO. For this IP core, the flags must be programmed as follows:

Signal	Interface	I/O	Default	Description
				Bit '0': FX3_POP_IS_EMPTY FX3 slave FIFOs are empty Bit '1': FX3_POP_CAN_GIVE_A_BUFFER FX3 slave FIFO has at least 8KByte data to be read Bit '2': FX3_PUSH_IS_FULL FX3 slave FIFOs are full Bit '3': FX3_PUSH_BUFFER_POSSIBLE FX3 slave FIFO has at least 8KByte space
o_fx3_faddr (1:0)	Slave FIFO Address	O	00	FX3 slave FIFO Address. The FX3 slave consists of 4 slaves which can be used to achieve high performance. For this IP, only 2 slave FIFOs are utilized. Bit '0': Always 0 Bit '1': 0 -> Host-2-Peripheral transfers Use slave FIFO socket-0 for H2P transfers Bit '1': 1 -> Peripheral-2-Host transfers Use slave FIFO socket-2 for P2H transfers
io_fx3_fdata	Slave FIFO Data bus	IO	Z	FX3 controller's data bus. This bus is a bidirectional bus. It needs an additional signal to control the data direction. See o_fx3_sloe_n.
o_fx3_slwr_n	write enable	O	1	The FX3 slave Interface's Write Enable signal. The IP core asserts this signal when it has data to write to FX3 slave FIFO. This signal is asserted after the o_fx3_slcs_n signal is asserted. active low signal.
o_fx3_slrd_n	read enable	O	1	The FX3 slave Interface's Read Enable signal. The IP core asserts this signal when it wants to read data from the FX3 slave FIFOs. This signal is asserted after the o_fx3_slcs_n and o_fx3_sloe_n signals are asserted.

Signal	Interface	I/O	Default	Description
				active low signal.
o_fx3_sloe_n	output enable	O	1	The FX3 slave Interface's Output Enable signal. The IP core asserts this signal when it wants to read data from the FX3 slave FIFOs. This signal is asserted after the o_fx3_slcs_n signal and before the o_fx3_slrd_n signals are asserted. active low signal. This signal is the output enable for the Bidirectional data port. '0' -> io_fx3_fdata behaves as input port The slave drives the data bus. If there is no input from the slave, the IP core pulls the data bus into a HIGH Impedence state (Z). '1' -> io_fx3_fdata behaves as output port. The Master drives the data bus. The slave pulls the data bus into a HIGH Impedence state (Z).
o_fx3_slcs_n	Slave FIFO Chip Select	O	1	FX3 controller's Chip Select Signal. active low signal.
o_fx3_pktend_n	Slave FIFO packet end sig	O	1	FX3 controller's Packet end signal. This design does not make use of packet end signalling. This signal is always deasserted in this IP-Core. active low signal.
o_fx3_pclk	PCLK for FX3	O	NA	FX3 controller's Clock Signal. Maximum of 100MHz clock frequency can be supplied to the FX3 controller. This IP Core provides a 100MHz Clock through this port.
M_AXI	AXI Master	NA	NA	AXI4 Master bus.

Signal	Interface	I/O	Default	Description
				These signals follow AXI Specifications as mentioned in the *Appendix A* of the Vivado AXI reference Guide (UG1037)for AXI4, AXI4-Lite and AXI-Stream Signals.

Clocking

The AXI-FX3-Interface v1.2 IP-Core operates on the **sys_clk**. It is a **100MHz** Clock.

IMPORTANT NOTE: The same clock must be used for sys_clk and all the AXI-peripherals. This clock will also be used to generate the "o_fx3_pclk" which drives the on-board FX3. The "o_fx3_pclk" is looped back to the FPGA to pin "i_fx3_pclk". This looped back clock must be connected to "i_fx3_pclk" port of the IP-Core. This drives the IP-Core's GPIF-II State Machine used to communicate with the FX3's slave FIFOs.

Resets

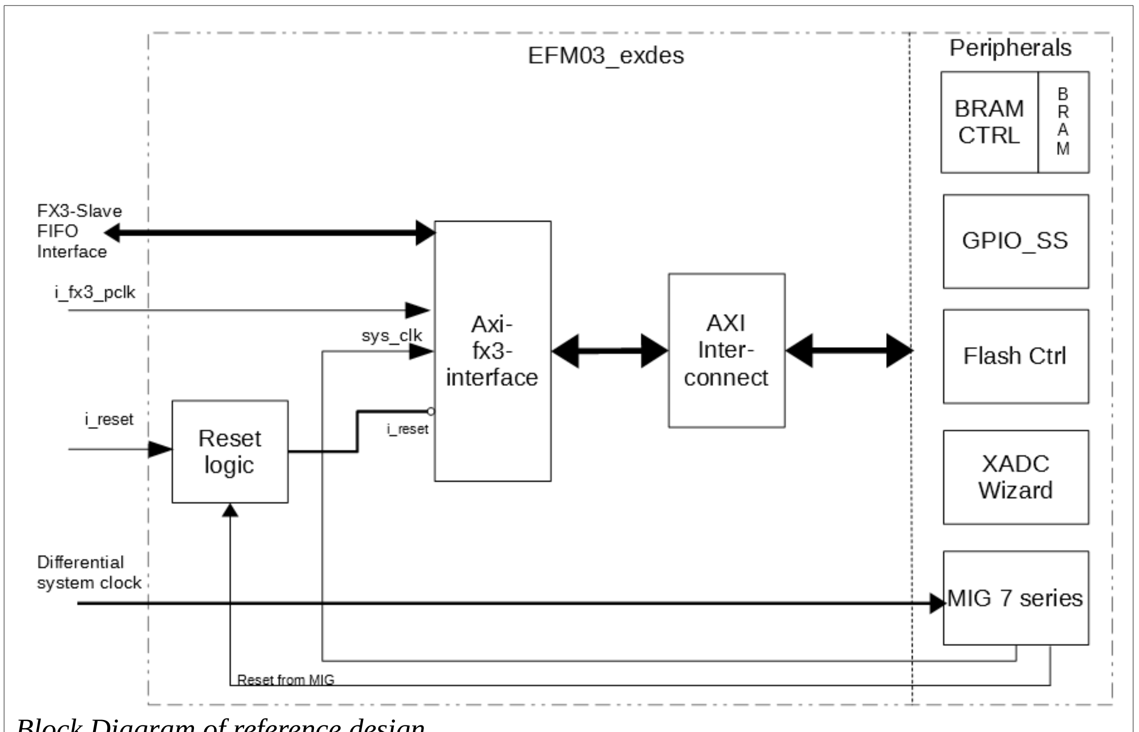
The AXI-FX3-Interface v1.2 IP-Core is reset when the **i_reset** is asserted. This signal is an active-LOW reset and is synchronous to **sys_clk** clock.

IMPORTANT NOTE: The reset on the EFM03 Beastboard provides an active-HIGH reset to the FPGA on pin **K15**. However, the reset port *i_reset* in the AXI-FX3-Interface v1.2 IP-Core is an active-LOW reset. It is the responsibility of the user to convert the active-HIGH signal to an active-LOW reset.

Reference design

This chapter contains information about the reference design provided with the AXI-FX3-Interface v1.2 IP-Core. It serves as a starting point for user designs. It is designed in such a way that it reduces development time in your own system. You can easily expand the reference design by adding your own blocks/ IPs to the AXI bus system or modifying the existing ones. The reference design is delivered as a block design with VHDL sources. The FPGA bitstream can be generated with Xilinx's Vivado Design Suite toolset (2017.2.1 onwards).

The top module instantiates all components and IP-cores that are essential for implementing the reference design on EFM03 Beastboard Hardware. The reference design is named as **EFM03_exdes.xpr** and can be found in the **EFM03_exdes/** folder in the downloaded zip file. The figure below gives an overview of the reference design and different components in it.



Block Diagram of reference design.

The reference design receives commands from the host, and based on the received command, performs a READ or a WRITE operation on the addressed peripheral. A brief explanation is provided below about the components used.

Reset Logic: The reference design uses a combinatorial reset logic to ensure proper resets are given to the various components and IP-cores. It is necessary because the EFM03 Beastboard provides an active-HIGH reset to the FPGA at pin **K15**. This reset is directly given to the Memory Interface Generator (MIG), which generates an active HIGH reset (it is dependent on the polarity of Reset Output selected while generating the MIG).

However, the AXI-FX3-Interface v1.2 IP-Core requires an active-LOW reset, as is the case with almost all AXI based IP Cores available in Xilinx Vivado. The Reset Logic receives MMCM-locked and the *init_calib_complete* signals from the MIG IP. It also takes into account the *End_Of_Startup (EOS)* signal from the AXI Flash Controller IP. Based on simple combinatorial logic, it generates an active-LOW reset which can be used to reset the AXI-FX3-Interface v1.2 IP-Core, AXI Interconnect and the AXI based peripherals.

BRAM Controller and Block Memory Generator: The reference design instantiates Block RAM using the *Block Memory Generator* IP Core available in Vivado. This IP uses Block RAM primitives present on the FPGA fabric. These Block RAMs can be accessed using the *AXI-BRAM Controller* IP-Core. By varying the address range in the *Address Editor* tab in Vivado, the size of the Block RAM can be varied. In this reference design, a 64KByte Block RAM module is instantiated. For further details on the AXI-BRAM Controller and Block Memory Generator IP-cores, please refer to *PG078* and *PG058* programming guides from Xilinx, respectively.

GPIO Sub System (GPIO_SS): The reference design uses AXI based GPIO IP cores that are readily available in Vivado. Each IP Core can access 32 IOs parallelly. Therefore, a total of six AXI-GPIO cores are required in order to address all the 191 General Purpose IOs available on the EFM03 Beastboard and also the USER_LED. The GPIOs are named *axi_gpio_X*, where 'X' is 0 to 5. In the reference design, the User LED has been configured to Bit 31 (MSB) of GPIO_5. At Reset/ Power-up, all the GPIO pins are configured to PULLDOWN. For further details on the AXI-GPIO IP-core, please refer to *PG144* AXI GPIO programming guide from Xilinx.

Memory Interface Generator (MIG): The EFM03 Beastboard houses a 2GB DDR3 RAM which is accessible through a Xilinx 7-Series Memory Interface Generator (MIG) IP core. The reference design instantiates an AXI4 based MIG IP core.

The IP takes in the external 200MHz differential clock available on the EFM03 Beastboard as input and generates a 100MHz internal clock for user-logic. This internal 100MHz clock signal provides clock to the AXI-FX3-Interface v1.2 IP-core (sys_clk), the AXI-Interconnect and all the AXI-based peripherals in the reference design. The MIG IP-Core takes the active-HIGH reset as sys_rst and provides an active-HIGH *ui_clk_sync_rst*, which is synchronous to the 100MHz *ui_clk* output of the MIG IP-Core. The *ui_clk_sync_rst* provides an active-HIGH reset to the clock manager. It is also used by the Reset Logic to generate a common active-LOW reset signal for the AXI-FX3-Interface v1.2 IP-Core, the AXI-Interconnect and all the AXI peripherals.

For further details on the MIG IP-core, please refer to *UG586* Zynq-7000 AP SoC and 7 Series FPGAs user guide from Xilinx.

AXI Flash Controller: The EFM03 Beastboard houses a 32MB NOR Flash Memory which can be accessed over an SPI interface. The reference design instantiates an AXI-based Controller IP Core module to access this Flash Memory Device. This Core provides access to the entire 32MBytes as an addressable peripheral. It consists of a very simple command set supporting only SECTOR_ERASE, PAGE_PROGRAM and FAST_READ commands. The IP can access only 4 bytes per address and all the operations are blocking, i.e., when the module is accessing the flash memory device, it does not allow any further access. Even the DMA engine's S2MM stream interface is blocked till the access completes. No pipelining of commands and data is supported.

The Controller has a 26-bit address bus and a 32-bit data bus. Most significant bit (MSB) of the address bus is used as an ERASE-bit. The remaining 25 bits are used as address bits. This means that the IP can access 32MB addresses. However, the flash memory device is configured for 24-bit addressing only. It implies that we can address only 16MB at a time. To address the higher 16MBytes, the IP automatically sets the Extended Address Register when it sees addresses at the AXI interface which are greater than 0x00FFFFFF (when Bit 24 is HIGH) and continues accessing the higher 16MB for any further access. When the address at AXI interface falls below 0x00FFFFFF again, it resets the Extended Address Register and starts accessing the Lower 16MB of the Flash Memory. This is accomplished by the WRITE_EXTENDED_ADDRESS_REGISTER command. By default, at Power-on/ Reset, the lower 16MB are accessed. When the ERASE-bit is set, the core sends SECTOR_ERASE command to erase the sector containing the address in the bits(24:0). Any address in that particular sector is valid.

The IP core runs on the 100MHz clock generated by MIG. The core generates a 50MHz clock with the Flash Slave Select signal as a clock-enable. It is provided to the flash memory device through Xilinx's STARTUPE2 primitive. In EFM03 Beastboard, the EMCCLK is used to clock the flash memory device in FPGA-configuration mode. EMCCLK can not be used to clock the flash memory when FPGA is in user mode. Therefore, this IP instantiates the STARTUPE2 primitive which facilitates mapping of a user-defined clock signal at the FPGA's CFGCLK pin after FPGA configuration is done. The STARTUPE2 primitive asserts the End_Of_Startup (EOS) output signalling the end of configuration. After assertion of EOS signal and subsequent deassertion of reset input (s00_axi_aresetn), the IP Core asserts the Slave Select (o_flash_ss) signal for five clock cycles ensuring a proper switching of the CFGCLK-pin output from config-clock (CFGCLK) to user-clock. For more information about STARTUPE2 primitive and Start Up sequence in 7 Series FPGA, please refer *UG470* 7 Series FPGAs Configuration from Xilinx.

The AXI Flash Controller IP Port description.

Signal	Interface	I/O	Default	Description
s00_axi_aclk	Clock	I	NA	100MHz input clock signal for AXI slave interface. THE SPI logic is also clocked using this clock.
s00_axi_aresetn	Reset	I	NA	active LOW reset to the IP Core.
s00_axi*	S_AXI		NA	AXI4-Lite slave bus. These signals follow AXI Specifications as mentioned in the *Appendix A* of the Vivado AXI reference Guide (UG1037)for AXI4, AXI4-Lite and AXI-Stream Signals.
CFGCLK	Clock	O	NA	Clock output to the Flash Memory Device. Using the STARTUPE2 primitive, a 50MHz User Clock is provided to the CFGCLK pin. This clock is active when the o_flash_ss signal is asserted. After the system comes out of Power-on or Reset state, the IP Core asserts the Slave Select pin and gives out the 50MHz Clock for 5 clock cycles to ensure the transition of CFGCLK from Configuration Clock to User Clock. The FPGA logic takes 3 clock-cycles to achieve

Signal	Interface	I/O	Default	Description
				this. It is a one-time process only (After the system gets out of Power-on/ reset state). Please refer *UG470* 7 Series FPGAs Configuration from Xilinx.
EOS	End of Startup	O	NA	This port signals the End of the Startup sequence during FPGA configuration. This pin is active HIGH and is asserted once the FPGA is completely configured. After assertion of this pin, the User Clock can be given to CFGCLK. Please refer *UG470* 7 Series FPGAs Configuration from Xilinx.
o_flash_dq0	Q0	O	0	The serial data output pin to the Flash Memory device. Slave Select must be asserted before sending data to memory. Please refer N25Q 256Mb User Guide for further details.
i_flash_dq1	D1	I	0	The serial data input pin from the Flash Memory device. Slave Select must be asserted before receiving data from the memory. Please refer N25Q 256Mb User Guide for further details.
o_flash_dq2	W#/Vpp/ DQ2	O	1	The Write Protection control input. It is always set to '1' in this design. Please refer N25Q 256Mb User Guide for further details.
o_flash_dq3	HOLD#/ DQ3	O	1	The HOLD# signal. Pauses Communication with the device without deselecting it. It is always set to '1' in this design. Please refer N25Q 256Mb User Guide for further details.
o_flash_ss	slave Select	O	1	The Chip Select signal to Flash memory device. This is an active LOW signal. When deasserted, i_flash_dq1 input pin is tri-stated. Driving the Slave Select LOW enables the device. After Power-on/reset a falling edge on

Signal	Interface	I/O	Default	Description
				this signal is required prior startin any command. Please refer N25Q 256Mb User Guide for further details.

XADC Wizard: The reference design instantiates an XADC Wizard. The registers in the XADC module can be read regularly as part of a system-monitor functionality. This helps in determining varaibles such as FPGA-Temperature and Supply Voltages. It can also be used to generate alarms based on user defined parameters. Also, the MIG IP Core samples the device temperature periodically and uses this information for Data Alignment by compensating for the temperature drifts (see also *UG586* Zynq-7000 AP SoC and 7 Series FPGAs). This application instantiates the XADC Wizard for system monitoring purposes. No alarms are generated. The Host Software can read all the register addresses generated in the Wizard.

AXI-FX3-Interface: The reference design instatiates the AXI-FX3-Interface v1.2 IP-Core. This IP translates the host commands and data that are via FX3 slave FIFO into AXI transactions using a DMA Engine. The AXI Master has a 32-bit address bus (4G addresses) and a 32-bit data bus. The IP Core can access all the slave peripherals via an AXI-Interconnect IP core which is readily available in the Xilinx IP Suite. However, only 16 different slave peripherals can be accessed. If more slaves need to be interfaced, another AXI-Interconnect must be instantiated. For further details about the AXI-FX3-Interface IP Core, please refer to chapter **Overview** (chapter 3) of this document. Please refer to the table below for address ranges of all the slave peripherals that are connected to the AXI-FX3-Interface IP Core.

Address Range of all peripherals

Slave Name	Interface	Offset Address	Range	High Address
MIG 7 Series	S_AXI	0x0000_0000	2GB	0x7FFF_FFFF
axi_gpio_0	S_AXI	0x8000_0000	64KB	0x8000_FFFF
axi_gpio_1	S_AXI	0x8001_0000	64KB	0x8001_FFFF
axi_gpio_2	S_AXI	0x8002_0000	64KB	0x8002_FFFF
axi_gpio_3	S_AXI	0x8003_0000	64KB	0x8003_FFFF
axi_gpio_4	S_AXI	0x8004_0000	64KB	0x8004_FFFF
axi_gpio_5	S_AXI	0x8005_0000	64KB	0x8005_FFFF

Slave Name	Interface	Offset Address	Range	High Address
XADC Wizard	S_AXI	0x8006_0000	64KB	0x8006_FFFF
BRAM Controller	S_AXI	0xC000_0000	64KB	0xC000_FFFF
Flash Controller	S_AXI	0x8400_0000	64MB	0x87FF_FFFF

Using the reference design

The reference design is a part of the Cesys UDK that is provided with the EFM03 Beastboard. All the relevant files and sources are packed in an archive "efm03_reference_design-v1.2.1.zip". Using the login information that you received with your board, please download the archive from our website (http://www.cesys.com) and follow the below steps.

1. Unzip the archive and navigate to **<installation path>/EFM03_exdes/**.
2. Open the **EFM03_exdes.xpr** file by double-clicking on it or in **Vivado->Open project**.
3. in the **Flow Navigator** window, under **IP Integrator**, click on **Open Block Design**
4. The Block Design of the reference design should appear on the canvas.

Simulating the design

The **EFM03_exdes** project consists of two workflows, one for synthesis/ implementation and another for simulation. This section describes the design in the simulation workflow.

The simulation workflow contains a block design *efm03_sim*, a VHDL wrapper for the block design (efm03_sim_wrapper.vhd), a VHDL test-bench (TB_efm03_ sim_wrapper.vhd) and a Waveform Configuration File (TB_efm03_ sim_wrapper_behav.wcfg). The block design itself consists of the AXI-FX3-Interface v1.2 IP-Core, an AXI-BRAM Controller and a Block Memory Generator. Further, the block design does not consist of the other IP components that are present in the reference design for synthesis and implementation workflow because it uses the MIG IP-Core output "*init_calib_complete*" and the STARTUPE2 primitive output "*End_Of_Startup*" signals in order to generate the active-LOW reset signal to the AXI-FX3-Interface v1.2 IP-Core and the AXI bus system. These signals are hardware dependent and can not be simulated with a test-bench.

The test-bench is designed in such a way as to provide commands and data to the design. It follows the Cesys UDK3 protocol (See also **AN101** for *UDK3 api specifications*). The test-bench provides all the clocks and resets required to simulate the design. A 180° phase shifted clock is provided for the loopback clock *i_fx3_pclk* in order to test the FX3 Slave interface operation when used with a different clock.

At time 0s, the test-bench asserts and holds the reset signal to '0' (active-LOW). It waits for 25 clock cycles (250ns) before deasserting it. The test-bench then sets the FX3 Flags for a write operation ("0010") and writes the header and dummy bytes followed by 1024 data bytes and the remaining dummy bytes to the BRAM starting from address 0x00000000. After writing the data, the FX3 Flags are set to "0000" which signifies no opertion. After 25us, the FX3 Flags are set to "0010" in order to transfer a command to read the previously written 1024 bytes from BRAM starting from address 0x00000000. After transferring the read command, the FX3 Flags are set to "1000" which signifies that the FPGA can transfer the 1024 bytes read from BRAM over the FX3 data interface. The AXI-FX3-Interface v1.2 IP-Core inserts the required amount of dummy bytes while transferring the data over the FX3 data interface.

The simulation design is completely built and the user can analyze the waveforms by just running the simulation. To simulate the design, please click on **Run Simulation** under **SIMULATION** section in the **Flow Navigator** window of Vivado. All the necessary signals are copied to the Waveform Configuration File. Please run simulation for at least 125 micro seconds in order to analyze the write and read operations.

Synthesis and Implementation

For synthesizing the design, click on **Run Synthesis** in **Flow Navigator** under **Synthesis**.

For implementing (Place And Route) the design, click on **Run Implementation** in **Flow Navigator** under **Implementation**.

To generate a bitstream, click on **Generate Bitstream** in **Flow Navigator** under **Program and Debug**.

Verifying the design

After generating the bitstream in Vivado toolset follow the below steps to verify the design:

1. Copy the generated **efm03_wrapper.bin** and paste it in the **<installation path>/udk3-tools-windows-1.5.1/** folder.
2. Run the **UDK3perfMon.exe** tool in the same folder
3. Select *EFM03(XCA200T)SOC Block RAM* or *EFM03(XCA200T)SOC DDR3L RAM* in **Preset** section.

4. In **Device**, select EFM03
5. Browse and select the efm03_wrapper.bin in **FPGA design file** field (it is also selected by default)
6. Select the Transfer Block Size either by clicking on the options or by entering the size manually
7. Make sure the Transfer Block Size is a power of 2 or Area Size is a multiple of Transfer Block Size
8. Select Input to read, Output to write or all three to write data and then read and verify the written data.
9. Press **Start**; Data transfer should begin and run continuously without any errors.

Constraints

The reference design comes with a Xilinx Design Constraints (XDC) file and it can be found under:

<installation_path>/EFM03_exdes/EFM03_exdes.srcs/constrs_1/new/efm03_constr.xdc.

This file contains all the IO and delay constraints required in order to acheive the best performance with this reference design. The most important constraints are described below.

These constraints state that the output clock (o_user_clk; N21) is generated from the sys_clk of the IP Core and sets up the i_fx3_pclk to be a 100MHz clock with 50% duty cycle at FPGA ball location M21. It also sets up the reset pin at FPGA ball location K15.

```
# Output clock to FX3 Chipset. This clock is looped back in hardware to ball N21
set_property PACKAGE_PIN N21 [get_ports o_user_clk]
set_property IOSTANDARD LVCMOS33 [get_ports o_user_clk]

# Generated clock constraint for the loopback clock o_user_clk
create_generated_clock -name o_fx3_pclk -source \
[get_pins -filter REF_PIN_NAME=~sys_clk* -of [get_cells -hier axi_fx3_if*]] \
-multiply_by 1 [get_ports o_user_clk]

# Hardware-loop-back clock for 'i_fx3_pclk' port of AXI-FX3-Interface
set_property PACKAGE_PIN M21 [get_ports i_fx3_pclk]
set_property IOSTANDARD LVCMOS33 [get_ports i_fx3_pclk]
create_clock -period 10.000 -name i_fx3_pclk -waveform {0.000 5.000} [get_ports
i_fx3_pclk]

# reset
```

```
set_property PACKAGE_PIN K15 [get_ports i_reset]
set_property IOSTANDARD LVCMOS33 [get_ports i_reset]
```

These constraints set up the FPGA configuration mode. They state that the FPGA will be configured by an external master device, in this case the FX3 chipset. These constraints are very important and have to be present if the FPGA needs to be configured by host PC through USB. Also, the FPGA is instructed to use the 90MHz External Master Configuration Clock (EMCCLK) during configuration and not the custom CCLK (3MHz).

```
# Configuration Mode Slave Select Map 16 bit data bus
set_property CONFIG_MODE S_SELECTMAP16 [current_design]

# Use 90MHz External Master Configuration Clock during FPGA Configuration
set_property BITSTREAM.CONFIG.EXTMASTERCCLK_EN DIV-1 [current_design]
```

The FX3 Chipset has a tight setup/hold timing requirement on its input data bus and has a large delay before it outputs data to FPGA. The FPGA design must comply to these timing requirements. The below *set_input_delay* constraints state that FX3 data is valid at the FPGA input port only after 8ns and is valid for 2 ns. The *set_output_delay* constraints establish the setup/hold requirement of the FX3's control input and data input ports. Retaining these constraints is very important for the functionality of the reference design. The FPGA output data ports are constrained for a drive strength of 12mA and for a fast slew-rate so that the data is available at the FX3's inputs as soon as possible and with very less transition time. All address/ control ports are configured for a higher drive strength of 16mA.

```
# FX3 Data Output Delay Constraints
set_input_delay -clock i_fx3_pclk -max 8 [get_ports io_fx3_fdata]
set_input_delay -clock i_fx3_pclk -min 2 [get_ports io_fx3_fdata]

# FX3 Control In and Data In setup/hold requirements
# SETUP is 2 ns, so max is 2
# HOLD is 0.5 ns, so min is -0.5
set_output_delay -clock o_fx3_pclk -max 2.000 [get_ports io_fx3_fdata]
set_output_delay -clock o_fx3_pclk -min -0.500 [get_ports io_fx3_fdata]

set_output_delay -clock o_fx3_pclk -max 2.000 [get_ports o_fx3_slcs_n]
set_output_delay -clock o_fx3_pclk -min -0.500 [get_ports o_fx3_slcs_n]

set_output_delay -clock o_fx3_pclk -max 2.000 [get_ports o_fx3_slwr_n]
set_output_delay -clock o_fx3_pclk -min -0.500 [get_ports o_fx3_slwr_n]

set_output_delay -clock o_fx3_pclk -max 2.000 [get_ports o_fx3_slrd_n]
set_output_delay -clock o_fx3_pclk -min -0.500 [get_ports o_fx3_slrd_n]
set_output_delay -clock o_fx3_pclk -max 2.000 [get_ports o_fx3_sloe_n]
set_output_delay -clock o_fx3_pclk -min -0.500 [get_ports o_fx3_sloe_n]

set_output_delay -clock o_fx3_pclk -max 2.000 [get_ports o_fx3_pktend_n]
set_output_delay -clock o_fx3_pclk -min -0.500 [get_ports o_fx3_pktend_n]

set_output_delay -clock o_fx3_pclk -max 2.000 [get_ports o_fx3_faddr]
set_output_delay -clock o_fx3_pclk -min -0.500 [get_ports o_fx3_faddr]

# Slew Rate and drive strength constraints
set_property DRIVE 12 [get_ports {io_fx3_fdata[0]}]
set_property SLEW FAST [get_ports {io_fx3_fdata[0]}]
..
set_property DRIVE 12 [get_ports {io_fx3_fdata[31]}]
set_property SLEW FAST [get_ports {io_fx3_fdata[31]}]

set_property DRIVE 16 [get_ports o_fx3_slcs_n]
set_property DRIVE 16 [get_ports o_fx3_sloe_n]
set_property DRIVE 16 [get_ports o_fx3_slrd_n]
set_property DRIVE 16 [get_ports o_fx3_slwr_n]
set_property DRIVE 16 [get_ports o_fx3_pktend_n]
set_property DRIVE 16 [get_ports {o_fx3_faddr[1]}]
set_property DRIVE 16 [get_ports {o_fx3_faddr[0]}]
```

All the GPIO ports are configured for 3.3V LVCMOS standards. They are pulled-down at configuration (except for gpio_5_tri_io[31] which is connected to User-LED). This is important because some of these FPGA balls are used during FPGA Configuration and any other setting may cause the FPGA Configuration to fail. Please refer to UG120-EFM03-Hardware-Reference Guide for a complete mapping of GPIO pins to FPGA balls.

```
# All GPIO ports are 3.3V LVCMOS and are PULLDOWN at configuration
set_property IOSTANDARD LVCMOS33 [get_ports {gpio_0_tri_io[0]}]
set_property PULLDOWN true [get_ports {gpio_0_tri_io[0]}]
...
set_property IOSTANDARD LVCMOS33 [get_ports {gpio_5_tri_io[30]}]
set_property PULLDOWN true [get_ports {gpio_5_tri_io[30]}]

set_property IOSTANDARD LVCMOS33 [get_ports {gpio_5_tri_io[31]}]
```

IMPORTANT: Ignoring these constraints may result in undefined behaviour. The constraints to set up the differential system clock are described in the MIG IP's constraints. The Memory Interface Controller's (MIG) ports are constrained by the MIG IP's constraints file and they are fixed. They can be seen in MIG IP's customization tool under *Pin Selection* window. The MIG IP's pin configuration should not be changed if the design is intended to be programmed onto EFM03 Beastboard.

Performace benchmarking

The design and its performance was thoroughly tested using the UDK3 Performance Monitor tool available with the UDK3 toolchain under **<UDK3_installation_path>/udk3-tools-windows-1.5.x** folder. The Performace Monitor was run on an Intel Core i7-4790 processor running a 64-bit Windows-7 Ultimate operating system. The design's performance was recorded and is presented in the table below:

Peripheral	Transfer Size (Bytes)	Area Size	Write Speed (MBps)	Read Speed (MBps)	Read & verify (MBps)
DDR3L RAM	2048	0x8000_0000	32.25	15.60	21.10
DDR3L RAM	8192	0x8000_0000	124.10	62.25	82.65
DDR3L RAM	65536 (64K)	0x8000_0000	266.75	222.50	242.50
DDR3L RAM	262144 (256K)	0x8000_0000	290.25	289.50	282.25
DDR3L RAM	1048576 (1M)	0x8000_0000	304.30	320.50	302.50
Block RAM	2048	0x0001_0000	32.10	15.60	21.00
Block RAM	8192	0x0001_0000	127.85	62.60	84.20
Block RAM	65536 (64K)	0x0001_0000	272.75	221.50	245.10

Copyright Notice

Disclaimer

Disclaimer

CRITICAL APPLICATIONS

Address

CESYS Gesellschaft für angewandte Mikroelektronik mbH
Gustav-Hertz-Str. 4

D - 91074 Herzogenaurach - GERMANY

Revision history

Version	Date	Details	Author	Approved by
1.0	Jun 01, 2017	Initial Release	vv	mh
2.0	Aug 31, 2017	1. Performance improvement 2. Lesser Resources 3. AXI-Flash controller 4. XADC Wizard	vv	mh
3.0	Oct 12, 2017	1. XDC Constraints 2. Verifying the design 3. Performace benchmarking	vv	mk
4.0	Nov 21, 2017	Added "Simulating the design". Changed order of subsections in "Using the reference design".	vv	mk