CON-FMC Example of AMBA AXI BFM with Block Memory

Version 0 Revision 1

May 5, 2019 (May 10, 2018)

Future Design Systems, Inc.
www.future-ds.com / contact@future-ds.com

Abstract

This document addresses an example of CON-FMC with ML605 FPGA board consisting of AMBA AXI BFM and memory.

Table of contents

1 Introduction

This example introduces an example as shown in Figure 1, where the design consists of a simple AMBA AXI memory that is accessed by the host program through AXI BFM over USB 2.0 or 3.0.

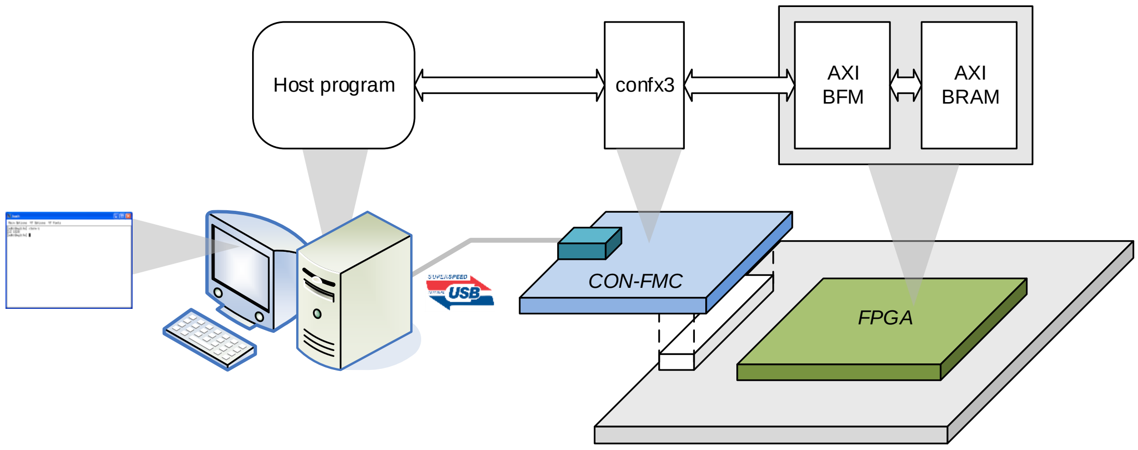

Figure 1: AMBA AXI Memory Platform

2 Getting started

2.1 Prerequisites

- Mandatory
  - ✧ CON-FMC board
  - ✧ CON-FMC SW package
  - ✧ FPGA board: Xilinx ML605 board
  - ✧ USB device driver: usbfs for Linux
  - ✧ LIBUSB: libusb-1.0
- Optional
  - ✧ Simulator: industry standard Verilog simmulator, e.g., ModelSim
  - ✧ FPGA development package: Xilinx ISE 14.7

2.2 Setup

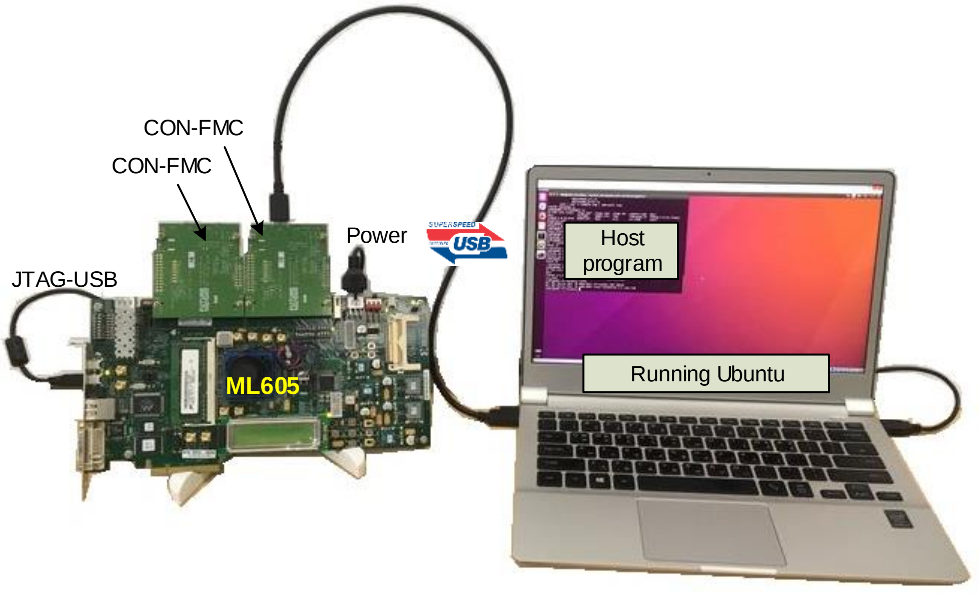

Figure 2: Setup under operation (one of two CON-FMC is used)

2.3 Quick start

1. Make sure the FPGA board is turned off
2. Connect CON-FMC to the FPGA board on HPC or LPC connector
3. Connect CON-FMC to the host computer through 3.0
4. Connect JTAG-USB to the host computer
5. Turn on FPGA board[1]
6. Down load bit-stream under the $project directory[2]
   - hw.single/pnr/ise.ml605.lpc/fpga.bit
   - hw.single/pnr/ise.ml605.hpc/fpga.bit
   - hw.dual/pnr/ise.ml605.dual/fpga.bit
7. Go to '$project/sw.native/test_mem' directory
8. Run 'make cleanup; make; make run'
9. Then, host-program writes and read the memory in the FPGA.

3 Design in detail

3.1 Hardware structure

Figure 3 shows internal structure of the design; i.e., FPGA design, where 'bram_AXI' is directly connected to 'trx_AXI' through AMB AXI bus signals.

[1] After turn on, CON-FMC should be detected by '$ lsusb' command. It should list '0x04B4:0x00F3' device.
[2] Xilinx 'impact' tool can be used or simple run '$ make' in $project/hw.single/pnr/ise.ml605.lpc/impact' directory. Bit-stream should be carefully selected depending on which FMC connector is used.

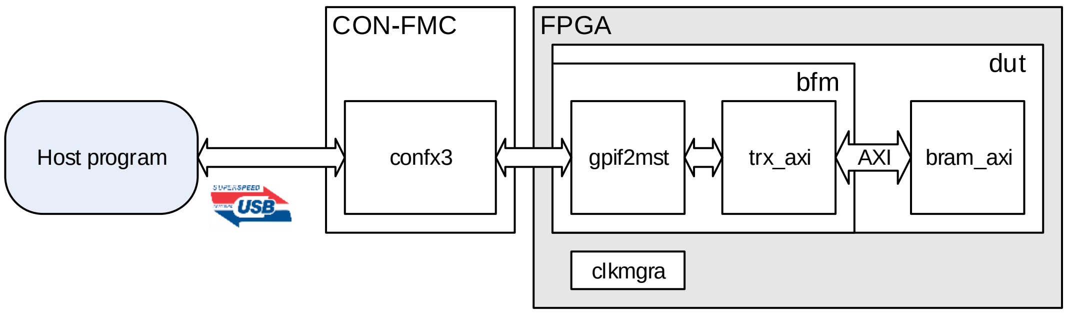

Figure 3: Hardware structure

3.1.1 Address map

AXI BFM bypasses all AMBA transactions. For this example, AMB BRAM, i.e. memory has 512Kbyte space.

3.1.2 Clock and reset

As shown in Figure 4, one clock is fed into the FPGA, which is 66Mhz USER_CLK_IN. This clock is adjusted by FPGA internal MMCM and two clocks are provided; one is SYS_CLK and the other is SYS_CLK.

SYS_CLK: it is for USB interface and can be 80Mhz or 100Mhz depending on defining macro, 'SL_PCLK_FREQ' in 'syn_define.v'.
USR_CLK: it is for user design and can be 80Mhz or 100Mhz depending on defining macro, 'USR_CLK_FREQ' in 'syn_define.v'.

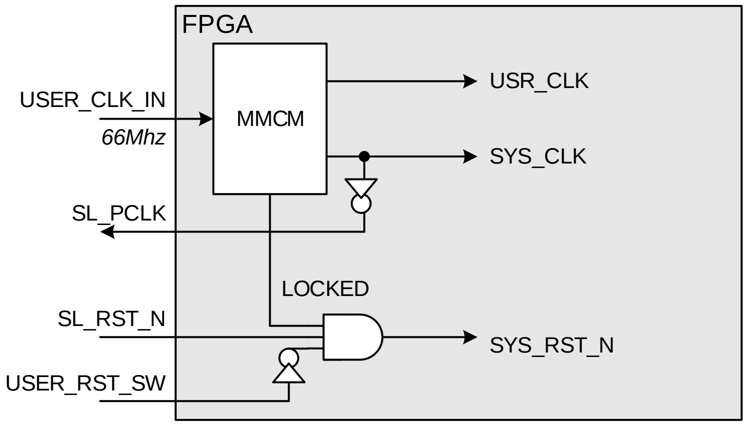

Figure 4: Clock structure

As shown in Figure 4, three reset sources are fed into the FPGA; SL_RST_N (active-low), USER_RST_SW (active-high), LOCKED (active-low in terms of reset). These two are active-low. In addition to this MMCM LOCED result is used to build reset.

3.2 Software

3.2.1 Host-program driven BFM architecture

Transactor (i.e., BFM: Bus Functional Model) in the FPGA generates system bus specific bus transaction under the command that comes from the host program through USB channel.

As shown in Figure 5, C program sends bus commands to the BFM through transparent communication channel over USB and then the commands direct BFM to carry out corresponding bus transactions, which can be AXI or AXI depending on BFM and system bus.

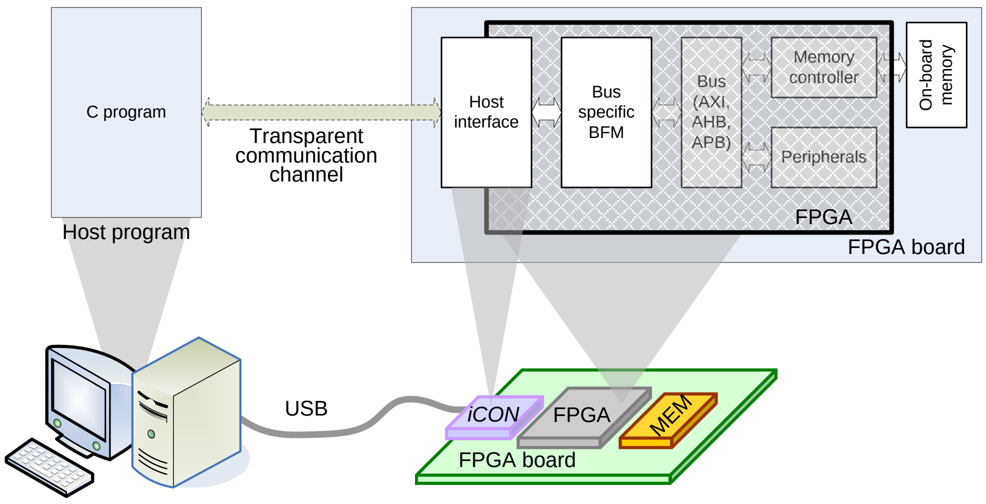

Figure 5: Host-program driven BFM architecture

3.2.2 C API

It should be noted that following C APIs are just macro and you should have proper C header and library. If you are not sure, please contact technical support of Future Design Systems.

- $CONFMC_HOME/include/conapi.h
- trx_AXI/drv/c/trx_axi_api.h

BfmWrite() generates write transaction on the system bus, where the buffer pointed by ‘data’ carries justified data and each item carries ‘size’ bytes up to 4.

```
void BfmWrite( unsigned int  addr   // address to write
             , unsigned int *data   // pointer to buffer contains data to write
             , unsigned int  size   // num of bytes for each element in the buffer
             , unsigned int  length); // num of items to write
```

BfmRead() generates read transaction on the system bus, where the buffer pointed by ‘data’ carries justified data and each item carries ‘size’ bytes up to 4.

```
void BfmRead ( unsigned int  addr // address to read
                        , unsigned int *data // pointer to buffer containing data read
                        , unsigned int  size // num of bytes for each item
                        , unsigned int  length); // num of items to read
```

Following shows an example,

```
unsigned int addr;
unsigned int dataR, dataW;

addr = 0x80000000;
dataW = 0x12345678;
BfmWrite(addr, dataW, 4, 1); // write 4-byte of data to the address 0x80000000
BfmWrite(addr, dataR, 4, 1); // read 4-byte of data from 0x80000000

addr = 0x80000002;
dataW = 0x00001234; // two bytes of data in justified fashion
BfmWrite(addr, dataW, 2, 1); // write 2-byte of data to the address 0x80000002
BfmWrite(addr, dataR, 2, 1); // read 2-byte of data from 0x80000002

addr = 0x80000003;
dataW = 0x00000012; // one byte of data in justified fashion
BfmWrite(addr, dataW, 1, 1); // write 1-byte of data to the address 0x80000003
BfmWrite(addr, dataR, 1, 1); // read 1-byte of data from 0x80000003
```

API argument 'addr' should be aligned with the size and data in the 'data' argument is justified.

4 Example script and program

4.1 Directory structure

Table 1: Directory structure

Directory					Remarks
fex_0003_amba_axi_mem					
	doc				document
	hw.dual				Dual FMC case
		beh			
		bench			
		design			
		pnr			
		sim			
		syn			
	hw.single				Single FMC case
		beh	verilog/gpif2slv.v		Behavioral tester
		bench	verilog/top.v		Top-level for simulation
		design	verilog/fpga.v		Design for FPGA

			verilog/dut.v verilog/clkmgra.v		
		pnr	ise.ml605.hpc ise.ml605.lpc		PnR for Xilinx ML605 board
		sim	modelsim		HDL simulation for ModelSim
		syn	xst.v6		RLT synthesis for Xilinx Virtex 6
	iplib				HW IP
		mem_axi			AMBA AXI memory IP
			beh		Behavioral model
			bram_dual_port		Xilinx Dual-Port BRAM
			rtl/verilog		RTL
	sw.native				Native-mode program to test the design
		test_mem			Memory testing
			src		

4.2 RTL simulation

As shown in Figure 6, the design can be simulated using tasks. It uses ‘gpif2slv’ behavioral model, which generates GPIF2 Interface transactions.

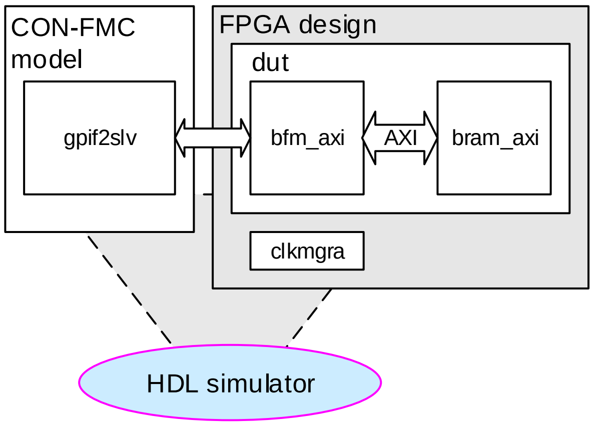

Figure 6: HW/SW co-simulation

Do as follows:

① go to $project/hw.single/sim/modelsim
② run ‘make’.

4.3 Logic synthesis

Look ‘$(project)/hw.single/syn/xst.v6’ directory and run ‘make’ to get net-list for Xilinx Virtex-7 FPGA.

4.4 FPGA PnR

Look '$(PROJECT)/hw.single/pnr/ise.ml605.lpc' directory and run 'make' to get bitstream.

4.5 Memory test

Look '$(project)/sw.native/test_mem' directory and run 'make cleanup; make run' to run the program.

Wish list

-

References

[1] Future Design Systems, Cypress EZ-USB FX3 GPIF-II Master Controller, FDS-TD-2018-04-002, 2018.
[2] Futue Design Systens, TRX_AXI: AMBA AXI Transactor for GPIF2MST, FDS-TD-2018-04-006, 2018.
[3] Futue Design Systens, CON-FMC API based on LIBUSB, FDS-TD-2018-04-004, 2018.

Revision history

- ☐ 2019.05.05: Version 0 Revision 1 is prepare by Ando Ki.
  - ✧ Minor correction.
- ☐ 2018.10.10: Version 0 Revision 0 is prepared by Ando Ki.

-- End of Document --