

## Introduction

The TE0300 comes with some reference designs built using Xilinx EDK version 10.1.03.

- DMA reference design;
- MPMC4 reference design;
- OPB reference design.

The instructions contained in this document can be applied to all reference designs. Besides standard IP cores, they contain three custom IP cores:

- xps\_vfbc\_dma\_v1\_00\_a
- xps\_fx2\_v1\_00\_a
- xps\_i2c\_slave\_v1\_00\_a

**xps\_vfbc\_dma\_v1\_00\_a** is a high speed DMA (direct memory access) engine which connects to the MPMC (Multi-Port Memory Controller) VFBC (Video Frame Buffer Controller) port. It enables high speed data streaming to/from external memory (DDR SDRAM). It can be controlled by a processor using 6 x 32-bit memory mapped registers attached to the PLB (peripheral local bus). For more information about registers, see the Xilinx MPMC Product Specification (mpmc.pdf), "Video Frame Buffer Controller PIM" section .

**xps\_fx2\_v1\_00\_a** is a core for high speed bidirectional communication between the FPGA and a host PC. It contains two 2 kB FIFOs for data buffering. For more information about the 5 x 32-bit memory mapped registers see the #project\_root#\pcores\xps\_fx2\_v1\_00\_a\doc.

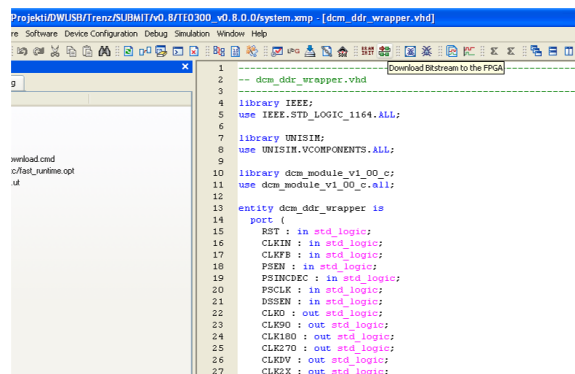
**xps\_i2c\_slave\_v1\_00\_a** is a core for low speed bidirectional communication between the FPGA and a host PC. It is usually used for command, settings and status communication. It contains 6 x 32-bit memory mapped registers:

- 3 for PC -> FPGA communication (FX2MB regs)
- 3 for FPGA -> PC communication (MB2FX2 regs)

When the PC sends commands to the Microblaze (MB) soft embedded processor, an interrupt is triggered. When the MB writes data to MB2FX2\_reg0, the interrupt (INT0) is sent to the Cypress EZ-USB FX2LP USB microcontroller. When the FX2 microcontroller receives an interrupt, it reads all MB2FX2 regs.

## Building the project

Open the project by double-clicking on the *system.xmp* file. The Xilinx Platform Studio is opened. To compile the project press the "Download Bitstream to the FPGA" button.

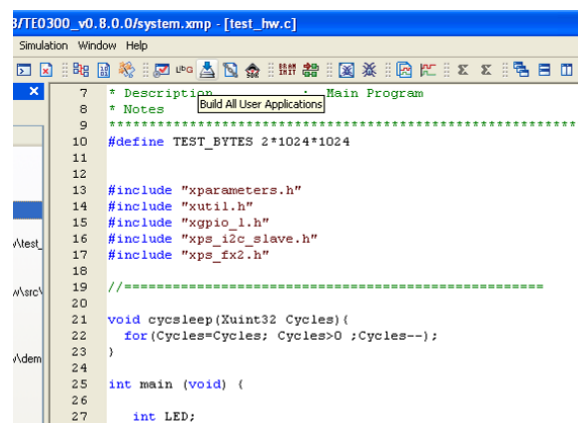


```
1  -- dcm_ddr_wrapper.vhd
2
3  -----
4  library IEEE;
5  use IEEE.STD_LOGIC_1164.ALL;
6
7  library UNISIM;
8  use UNISIM.VCOMPONENTS.ALL;
9
10 library dcm_module_v1_00_c;
11 use dcm_module_v1_00_c.all;
12
13 entity dcm_ddr_wrapper is
14 port (
15     RST : in std_logic;
16     CLKIN : in std_logic;
17     CLKFB : in std_logic;
18     PSEN : in std_logic;
19     PSINDEC : in std_logic;
20     PSCLK : in std_logic;
21     DSSEN : in std_logic;
22     CLRD : out std_logic;
23     CLKS0 : out std_logic;
24     CLK180 : out std_logic;
25     CLK270 : out std_logic;
26     CLKS90 : out std_logic;
27     CLK2X : out std_logic;
```

The HW implementation usually takes some time. The FX2 microcontroller on the TE0300 module should contain valid firmware before proceeding. If the FX2 microcontroller has not been programmed before, please follow the instructions in the TE0300 User Manual.

If you are sure that the FX2 microcontroller connects properly, you can connect connector J2 on the TE0300 module to the JTAG cable. We recommend using the Xilinx Platform Cable USB. Then connect the TE0300 module to a USB cable.

If the HDL design was successfully implemented and downloaded to the TE0300, you can proceed to compile the MB software. Press the "build all user applications" button.

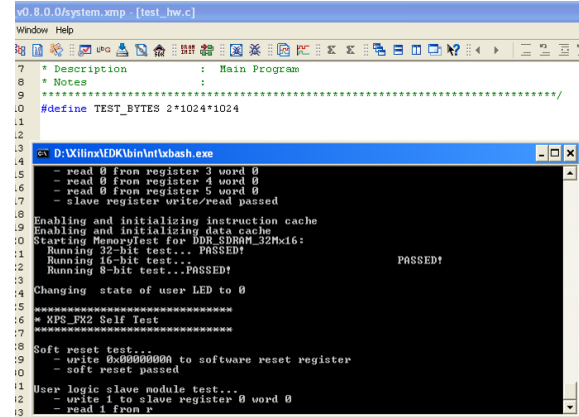


```

3/TE0300_v0.8.0.0/system.xmp - [test_hw.c]
Simulation Window Help
7 * Description      : Main Program
8 * Notes           : Build All User Applications
9
10 #define TEST_BYTES 2*1024*1024
11
12
13 #include "xparameters.h"
14 #include "xutil.h"
15 #include "xgpio_1.h"
16 #include "xps_i2c_slave.h"
17 #include "xps_fx2.h"
18
19 //=====
20
21 void cycsleep(Xuint32 Cycles){
22     for(Cycles=Cycles; Cycles>0 ;Cycles--);
23 }
24
25 int main (void) {
26     int LED;
27

```

When both applications (*hw\_test* and *demo*) are compiled, you can click on the "Start XMD" button to download the *hw\_test* application and open a UART terminal.



```

v0.8.0.0/system.xmp - [test_hw.c]
Window Help
7 * Description      : Main Program
8 * Notes           :
9
10 #define TEST_BYTES 2*1024*1024
11
12
13
14
15 - read 0 from register 3 word 0
16 - read 0 from register 4 word 0
17 - read 0 from register 5 word 0
18 - slave register write/read passed
19
20 Enabling and initializing instruction cache
21 Enabling and initializing data cache
22 Starting MemoryTest For DDR_SDRAM_32Mx16:
23 Running 32-bit test... PASSED!
24 Running 16-bit test... PASSED!
25 Running 8-bit test... PASSED!
26
27 Changing state of user LED to 0
28
29 =====
30 * XPS_FX2 Self Test
31 =====
32
33 Soft reset test...
34 - write 0x00000000 to software reset register
35 - soft reset passed
36
37 User logic slave module test...
38 - write 1 to slave register 0 word 0
39 - read 1 from r
40

```

Before running the *demo* application, open the `#project_root#\xmd.ini` file:

- 1.rst
- 2.dow sw/test\_hw.elf
- 3.#dow sw/demo.elf
- 4.run
- 5.terminal -jtag\_uart\_server 4321

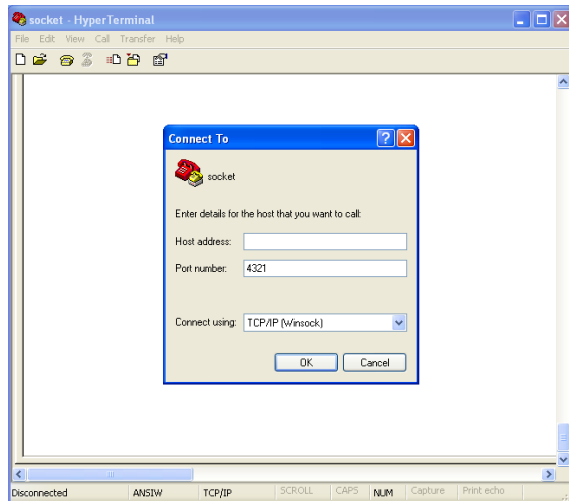
To run the *demo* application

- uncomment line 3 (remove "#")
- comment line 2 (add "#" as first character)
- save *xmd.ini*.
- type "exit" in XMD command window
- restart XMD by clicking again the "Start XMD" button in the XPS toolbar.

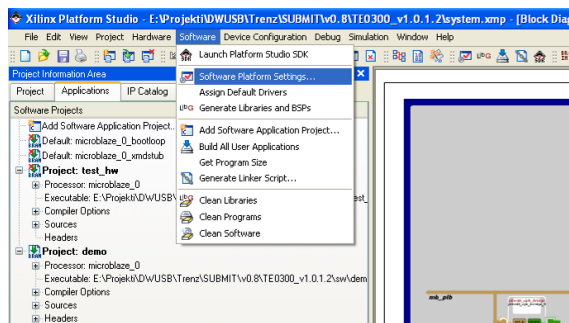
With this application, you can test the PC ↔ FPGA communication using a provided API.

If you want to input some characters to the XMD UART, then open some terminal emulators, such as Microsoft / Hilgraeve HyperTerminal (usually included in Windows START MENU / All programs / Accessories / Communications / HyperTerminal). Connect using the following settings:

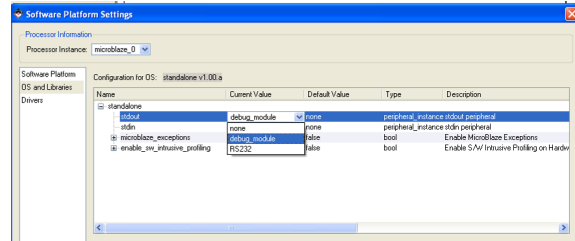
- No Host address
- Port Number: 4321
- TCP/IP connection type



**Note:** To use the *demo* project without the XMD UART, you need to use "RS232" instead of "debug\_module" as standard in/out port. Otherwise the application running on the Microblaze processor freezes if you disconnect the XMD. To accomplish that you need to set up the Microblaze "Software Platform Settings".



In the dialog window select "OS and libraries" in the left window and pick "RS232" as a stdout and stdin interface. Then rebuild the software and download again the project to the FPGA.



The UART is then redirected to external pins, which are defined in the *data/system.ucf* file:

```
#### Module RS232 constraints

Net fpga_0_RS232_RX_pin LOC=B13;
Net fpga_0_RS232_RX_pin IOSTANDARD
= LVCMOS33;
Net fpga_0_RS232_RX_pin PULLUP;
Net fpga_0_RS232_TX_pin LOC=B14;
Net fpga_0_RS232_TX_pin IOSTANDARD
= LVCMOS33;
```

The UART settings are:

- 115200 Baud
- 8 bits
- 1 stop bit
- No parity

Microblaze should work normally even though the RS232 interface is not connected.

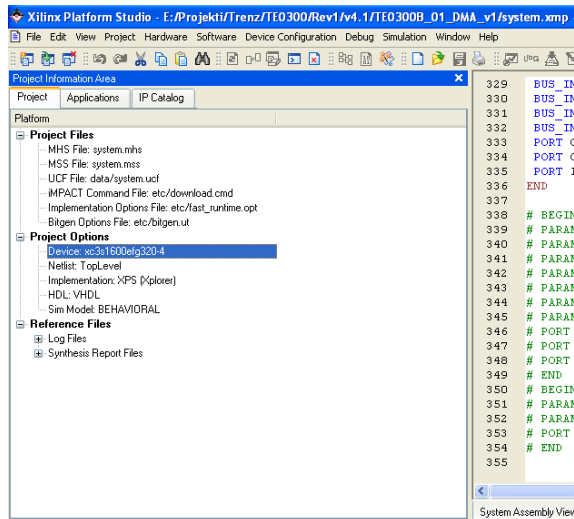
## Porting to different modules

The supplied reference designs were built for TE0300-01 which uses a 125MHz oscillator and a Spartan-3E XC3S1200E-4FG320 FPGA. Other module assembly versions are listed in Table 1.

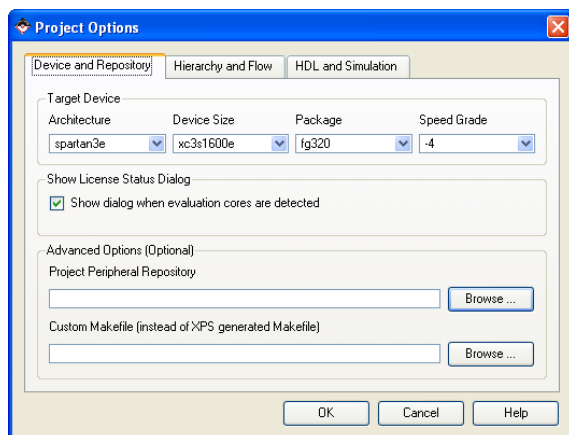
To change the FPGA device

- open the project in Xilinx Platform Studio
- click on the "Project" tab

- under "Project Options" double click on "Device":



Select a suitable FPGA device:



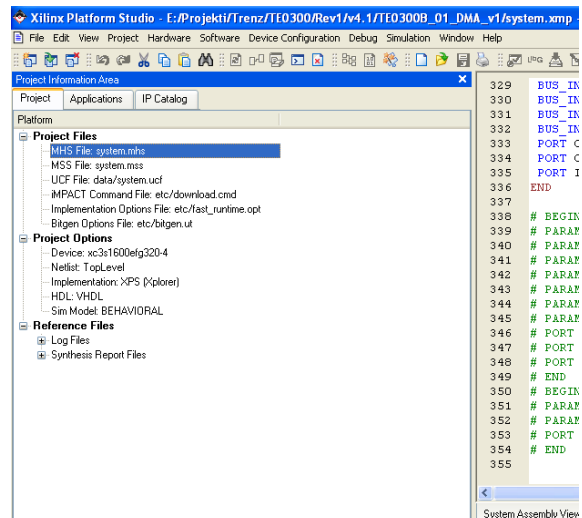
The example shows the case of a Spartan-3E xc3s1200e (or xc3s1600e)/fg320/-4.

For MPMC designs (MPMC4 and DMA design) you also need to replace the *system.ucf* (constraints file) which is located under #project\_root#\data with:

- *system.ucf.1200* for 1200 devices
- *system.ucf.1600* for 1600 devices

The DDR constraints are different for different device sizes. Otherwise you get timing / routing errors.

To change oscillator frequency, we advise you to manually edit *system.mhs*. You can open it by double clicking on "MHS File" under "Project Files":



Edit the input clock freq in Hz (100000000 or 125000000):

```
PORT sys_clk_pin = dcm_clk_s, DIR =
I, SIGIS = DCMCLK, CLK_FREQ =
125000000
```

Adjust clock generator frequencies by replacing all "125" occurrences by 100 and all "500" occurrences by 625 or the other way around:

```
BEGIN clock_generator
PARAMETER INSTANCE = clock_generator_0
PARAMETER HW_VER = 2.01.a
PARAMETER C_EXT_RESET_HIGH = 1
PARAMETER C_CLKIN_FREQ = 125000000
PARAMETER C_CLKOUT0_FREQ =
62500000
PARAMETER C_CLKOUT0_PHASE = 0
PARAMETER C_CLKOUT0_GROUP = NONE
```

```

PARAMETER C_CLKOUT1_FREQ =
125000000
PARAMETER C_CLKOUT1_PHASE = 0
PARAMETER C_CLKOUT1_GROUP = NONE
PARAMETER C_CLKOUT2_FREQ =
125000000
PARAMETER C_CLKOUT2_PHASE = 90
PARAMETER C_CLKOUT2_GROUP = NONE
PARAMETER C_CLKIN_BUF = FALSE
PARAMETER C_CLKOUT0_BUF = TRUE
PARAMETER C_CLKOUT1_BUF = TRUE
PARAMETER C_CLKOUT2_BUF = TRUE
PORT CLKOUT0 = sys_clk_s
PORT CLKOUT1 =
DDR_SDRAM_mpmc_clk_s
PORT CLKOUT2 =
DDR_SDRAM_mpmc_clk_90_s
PORT CLKIN = dcm_clk_s
PORT LOCKED =
clock_generator_locked
PORT RST = net_gnd
END

```

Adjust memory controller parameters to appropriate values:

```

BEGIN mpmc
PARAMETER INSTANCE = DDR_SDRAM
PARAMETER HW_VER = 4.03.a
PARAMETER C_NUM_PORTS = 3
PARAMETER C_PIM0_BASETYPE = 1
PARAMETER C_PIM1_BASETYPE = 1
PARAMETER C_MEM_PARTNO = HY-
B25D512160BF-6 (or MT46V32M16-6)
PARAMETER C_MEM_DATA_WIDTH = 16
PARAMETER C_MEM_TYPE = DDR
PARAMETER C_XCL0_WRITEXFER = 0
PARAMETER C_PIM2_BASETYPE = 6
PARAMETER C_MPMC_CLK0_PERIOD_PS =
8000 (or 10000)
PARAMETER C_MPMC_BASEADDR =
0x1C000000
PARAMETER C_MPMC_HIGHADDR =
0x1FFFFFFF
PARAMETER C_PIM2_DATA_WIDTH = 32

```

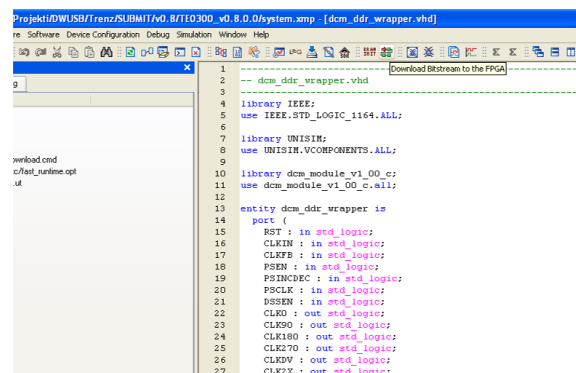
Adjust also UARTLITE system clock frequency (if you need UART on external pins of course):

```

BEGIN xps_uartlite
PARAMETER INSTANCE = RS232
PARAMETER HW_VER = 1.00.a
PARAMETER C_SPLB_CLK_FREQ_HZ =
62500000
PARAMETER C_BAUDRATE = 115200
PARAMETER C_ODD_PARITY = 0
PARAMETER C_USE_PARITY = 0
PARAMETER C_BASEADDR = 0x84000000
PARAMETER C_HIGHADDR = 0x8400ffff
BUS_INTERFACE SPLB = mb_plb
PORT RX = fpga_0_RS232_RX
PORT TX = fpga_0_RS232_TX
PORT Interrupt = RS232_Interrupt
END

```

That is all. Then download the bitstream file to the FPGA:



## Revision History

Rev	Date	Who	Description
1.0	2009-06-22	FDR	created

module version	kilo gates	env.	clock [MHZ]	memory (DDR SDRAM)
TE0300-01	1200	Com	125	Qimonda HYB25DC512160CF-6
TE0300-01M	1200	Com	125	Micron MT46V32M16BN-6:F
TE0300-01B	1600	Com	125	Qimonda HYB25DC512160CF-6
TE0300-01BM	1600	Com	125	Micron MT46V32M16BN-6:F
TE0300-01BLP	1600	Com	100	Qimonda HYB25DC512160CF-6
TE0300-01BMLP	1600	Com	100	Micron MT46V32M16BN-6:F
TE0300-01I	1200	Ind	125	Micron MT46V32M16BN-6 IT:F
TE0300-01IBM	1600	Ind	125	Micron MT46V32M16BN-6 IT:F

**Table 1: TE0300 assembly versions.**

"Com" is "commercial grade" and "Ind" is "industrial grade"; 100 or 125 MHz are oscillator frequencies. MT46V32M16BN-6 IT:F is a Micron Technologies industrial DDR SDRAM memory, while the others are commercial ones.