TE03xx Series Application Notes

Xilinx Spartan-3 Industrial-Grade FPGA Micromodules*

AN-TE03xx (v2.02) 3 August 2012 Trenz Electronic GmbH

1 Introduction

The TE03xx product family comes with some reference designs built using Xilinx EDK version 10.1.03 or superior.

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

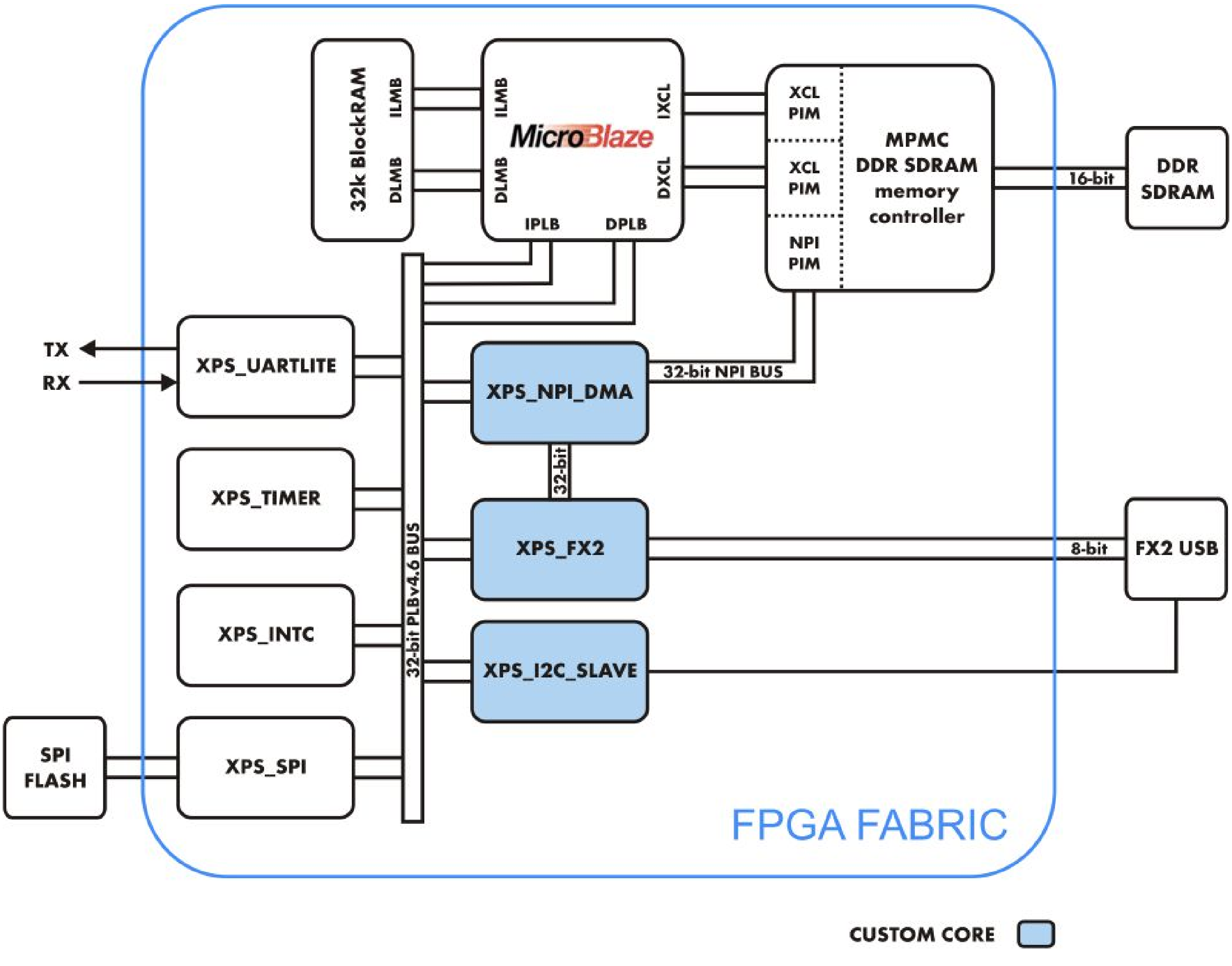


Figure 1: Trenz Electronic TE0300-01 reference architecture.

2 Reference Architecture

The Xilinx FPGA itself on the Trenz Electronic TE03xx family by default is blank and has no architecture. To define an FPGA functionality, a logic architecture should be defined and loaded into the device. The reference design system was built using Xilinx Embedded Development Kit (EDK). Basically, it is an embedded system with a MicroBlaze 32-bit soft microprocessor. The MicroBlaze initializes and sets up the system. The XPS_I2C_SLAVE block sends commands coming from the USB bus towards the MicroBlaze processor (low speed communication channel). The horsepower for high bandwidth data streaming is a Multiport Memory Controller (MPMC). A custom-built DMA (direct memory access) engine (XPS_NPI_DMA) streams data between multiple sources and external RAM simultaneously. Standard EDK cores are used to implement a serial interface (XPS_UARTLITE), an SPI FLASH interface (XPS_SPI), a timer / counter block (XPS_TIMER) and an interrupt controller (XPS_INTC).

When data is sent from the USB-host to the TE03xx family USB high-speed endpoint (high speed communication channel), it is automatically stored into the RAM by the DMA at a specified buffer location. The reference design software running on the MicroBlaze verifies the transferred data at the end of transmission and sends to the USB host a notification about the data test (pass/fail).

When data is sent form the TE03xx family USB high-speed endpoint to the USB-host, it is automatically fetched from the RAM via the DMA engine and forwarded to the XPS_FX2 core in 1 kB packets. Microblaze does the throttling to prevent XPS_FX2 TX FIFO overflow.

3 Custom Logic Blocks

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

- XPS_NPI_DMA
- XPS_FX2
- XPS_I2C_SLAVE

XPS_NPI_DMA 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 is a logic block 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.

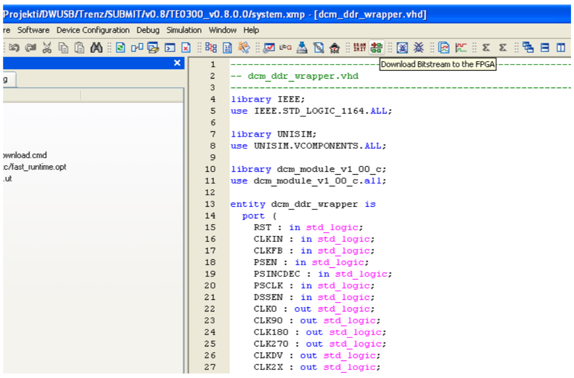
XPS_I2C_SLAVE is a logic block for low speed bidirectional communication between the FPGA and a host PC. It is usually used for command, settings and status communication. It contains 6 x 32-bit memory mapped registers:

- 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.

4 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.

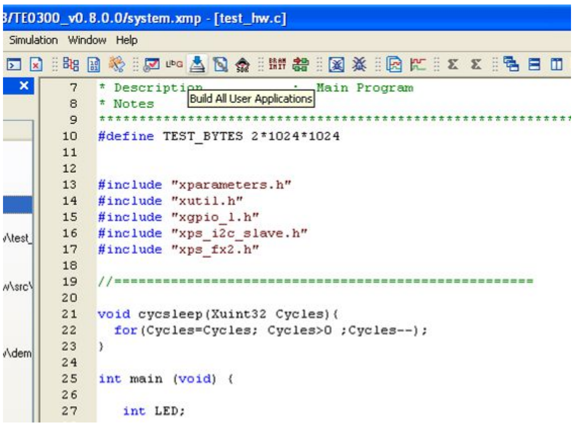

Projekti/DWUSB/Trenz/SUBMIT/v0.8/TE0300_v0.8.0.0/system.xmp - [dcm_ddr_wrapper.vhd]
re Software Device Configuration Debug Simulation Window Help
Download Bitstream to the FPGA
g
ownload.cmd
c/fast_runtime.opt
.ut

```
-------------------------------------------------------------
-- dcm_ddr_wrapper.vhd
-------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

library dcm_module_v1_00_c;
use dcm_module_v1_00_c.all;

entity dcm_ddr_wrapper is
  port (
    RST : in std_logic;
    CLKIN : in std_logic;
    CLKFB : in std_logic;
    PSEN : in std_logic;
    PSINCDEC : in std_logic;
    PSCLK : in std_logic;
    DSSEN : in std_logic;
    CLK0 : out std_logic;
    CLK90 : out std_logic;
    CLK180 : out std_logic;
    CLK270 : out std_logic;
    CLKDV : out std_logic;
    CLK2X : out std_logic;
```

The HW implementation usually takes some time. The FX2 microcontroller on the TE03xx family should contain valid firmware before proceeding. If the FX2 microcontroller has not been programmed before, please follow the instructions in the TE03xx family User Manuals.

If you are sure that the FX2 microcontroller connected properly, you can connect to the TE0300 module with a JTAG adapter cable. We recommend using the Xilinx Platform Cable USB. Then connect the TE03xx module to a USB cable.

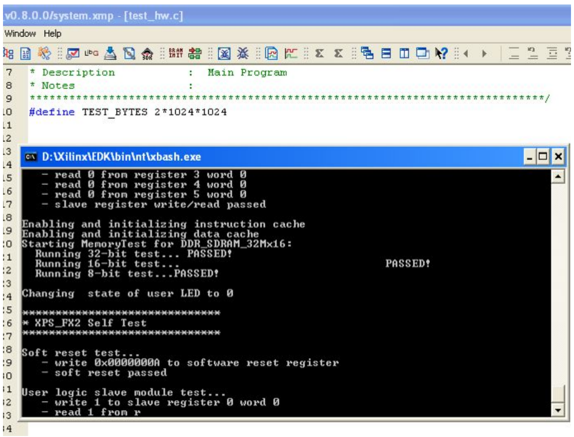
If the HDL design was successfully implemented and downloaded to the TE03xx family module, you can proceed to compile the MB software. Press the "build all user applications" button.



3/TE0300_v0.8.0.0/system.xmp - [test_hw.c]
Simulation Window Help
Build All User Applications
\test_
\src\
\dem

```
* Description                    :  Main Program
* Notes
*************************************************************
#define TEST_BYTES 2*1024*1024


#include "xparameters.h"
#include "xutil.h"
#include "xgpio_l.h"
#include "xps_i2c_slave.h"
#include "xps_fx2.h"

//====================================================

void cycsleep(Xuint32 Cycles){
  for(Cycles=Cycles; Cycles>0 ;Cycles--);
}

int main (void) {

   int LED;
```

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
D:\Xilinx\EDK\bin\nt\xbash.exe

```
* Description              :  Main Program
* Notes                    :
*******************************************************************************/
#define TEST_BYTES 2*1024*1024
```

```
   - read 0 from register 3 word 0
   - read 0 from register 4 word 0
   - read 0 from register 5 word 0
   - slave register write/read passed

Enabling and initializing instruction cache
Enabling and initializing data cache
Starting MemoryTest for DDR_SDRAM_32Mx16:
  Running 32-bit test... PASSED!
  Running 16-bit test...                              PASSED!
  Running 8-bit test...PASSED!

Changing  state of user LED to 0

******************************
* XPS_FX2 Self Test
******************************

Soft reset test...
   - write 0x0000000A to software reset register
   - soft reset passed

User logic slave module test...
   - write 1 to slave register 0 word 0
   - read 1 from r
```

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
```

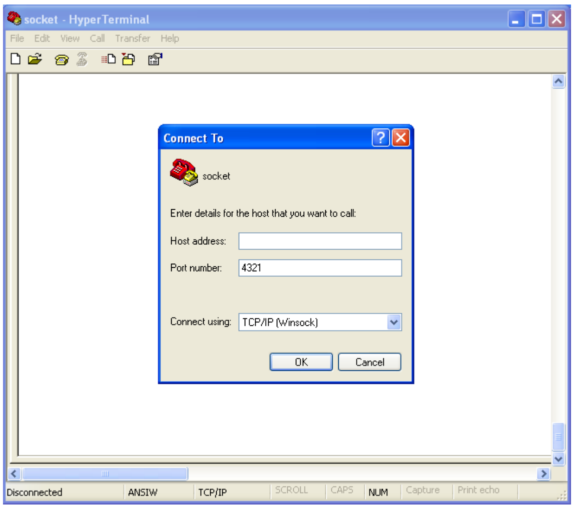
To run the *demo* application

- 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 / Acessories / Communications / Hyper Terminal). Connect using the following settings:

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



socket - HyperTerminal

Connect To

socket

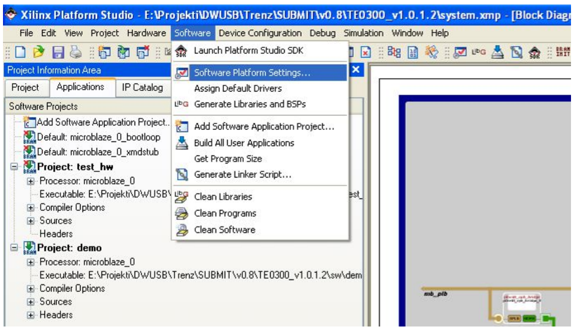
Enter details for the host that you want to call:

Host address:

Port number: 4321

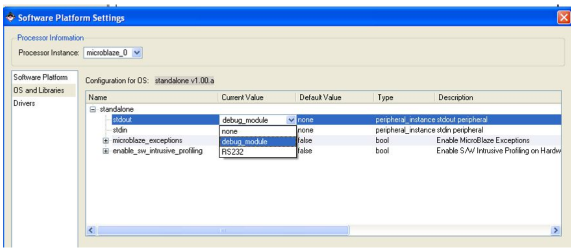
Connect using: TCP/IP (Winsock)

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".



Xilinx Platform Studio - E:\Projekti\DWUSB\Trenz\SUBMIT\v0.8\TE0300_v1.0.1.2\system.xmp - [Block Diag

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.



Software Platform Settings

Processor Information

Processor Instance: microblaze_0

Configuration for OS: standalone v1.00.a

Name	Current Value	Default Value	Type	Description
standalone				
stdout	debug_module	none	peripheral_instance	stdout peripheral
stdin		none	peripheral_instance	stdin peripheral
microblaze_exceptions		false	bool	Enable MicroBlaze Exceptions
enable_sw_intrusive_profiling		false	bool	Enable S/W Intrusive Profiling on Hardw

The UART is then redirected to external pins, which are defined in the *data/system.ucf* file. The following snippet shows the case of the TE0300 series modules:

```
#### Module RS232 constraints
```

```
Net fpga_0_RS232_RX_pin LOC=B13;
Net fpga_0_RS232_TX_pin LOC=B14;
```

Please refer to Table 1 for other module series relevant to this application note.

TE series	RS232_RX FPGA ball	RS232_RX module pin	RS232_TX FPGA ball	RS232_TX module pin
TE0300	R6	J5-29	P6	J5-31
TE0320	V17	J5-IO18	W17	J5-IO19
TE0630	Y7	J5-29	AB7	J5-31
TE0304		J1-3		J1-2
TE0323	J4-35	J4-35	J4-37	J4-37
host (PC)		TX		RX

Table 1: location of UART pins examples.

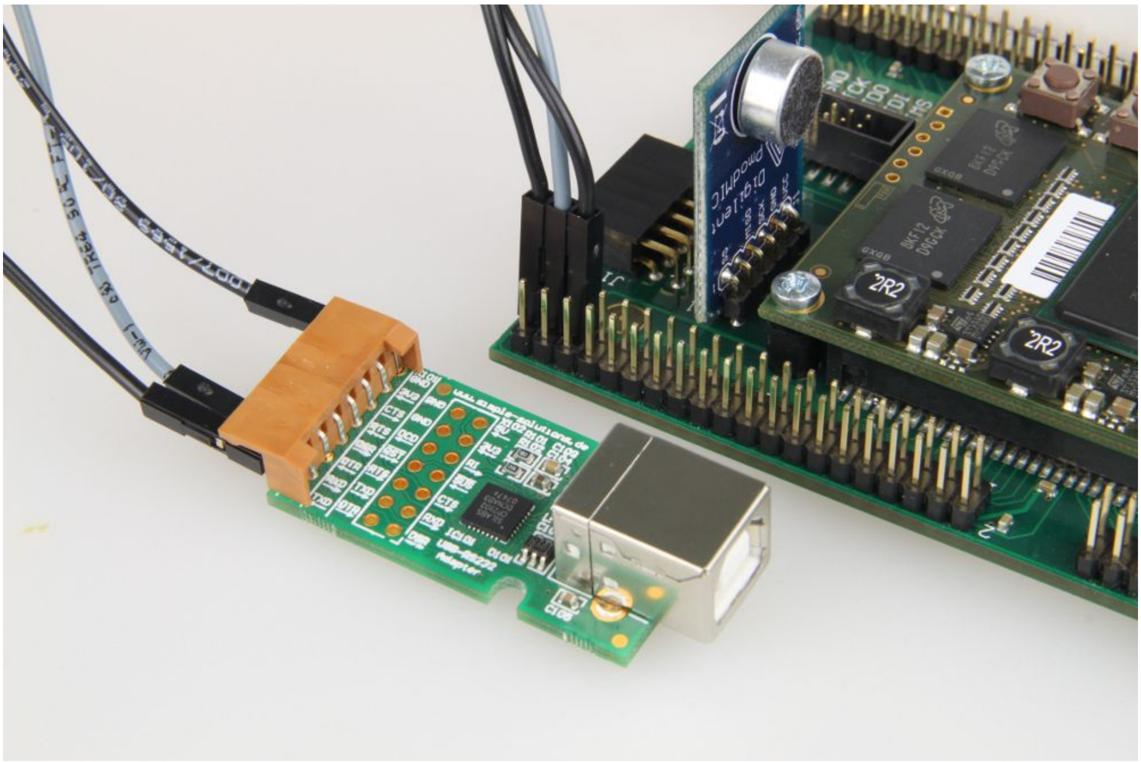
The UART settings are:

- bits per seconds: 115,200
- data bits: 8
- parity: none
- stop bits: 1
- flow control: none (otherwise you will not be able to enter commands)

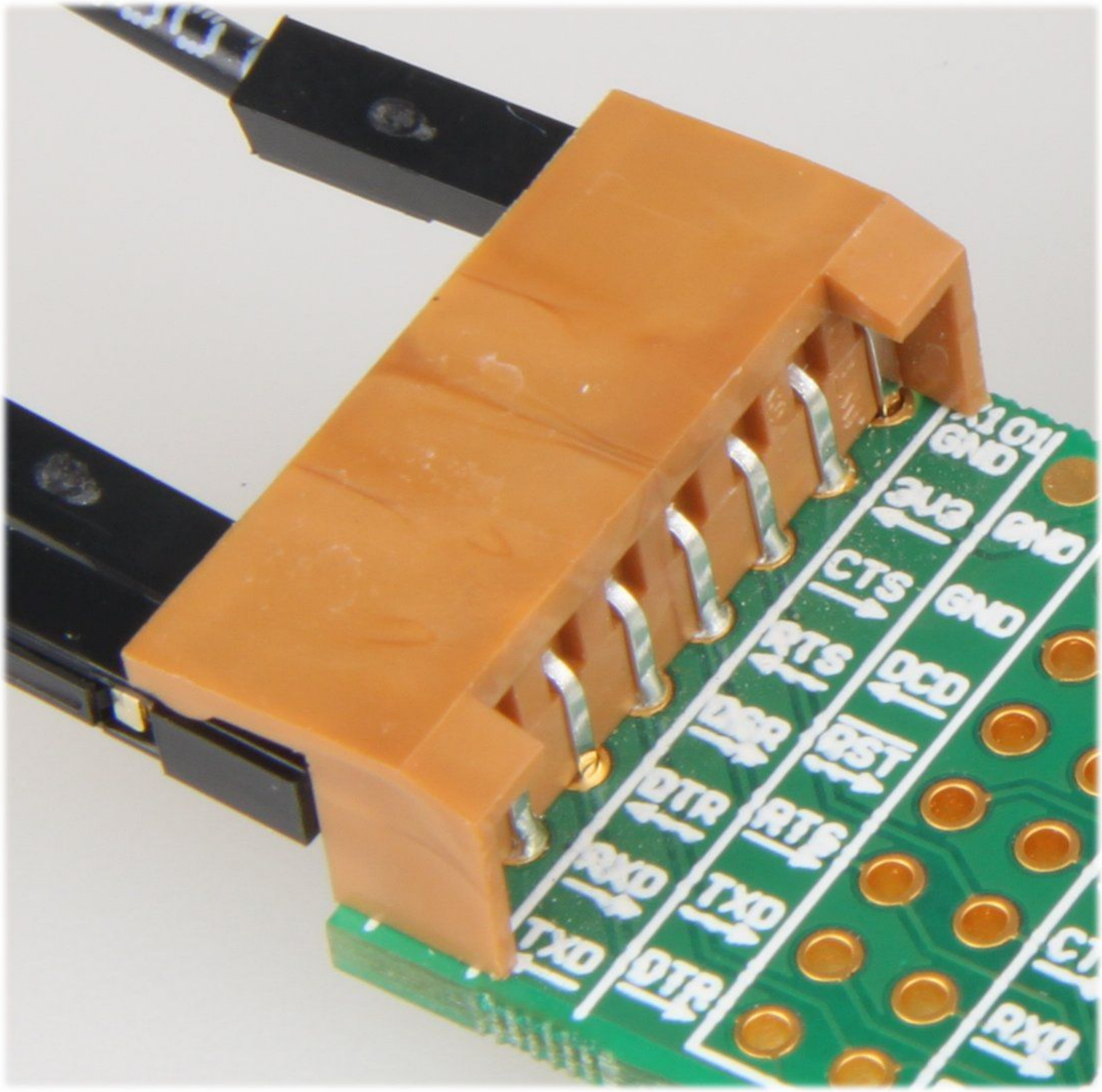
The UART port will output something of tis kind:

```
--Entering main TE0300 DEMO ver 0x07010218--

Setting up Interrupt Controller:
     Initialize exception handling
     Register external interrupt handler
     Register I2C_SLAVE interrupt handler
     Enable interrupts in the interrupt controller
     Start the interrupt controller
Enabling and initializing instruction cache
Enabling and initializing data cache

Type:
    'a' RAM test
    'f' RAM Ftest
    'c' toggles caching
    'g' prints switches state and board revision
    't' starts TX transmission
    'r' starts RX transmission
    's' stops all transmissions
    'm' for the redraw menu
```

MicroBlaze will work even in case the UART port is left unconnected.



Sample UART to USB virtual COM port converter.



Sample UART to USB virtual COM port converter: signal detail.

5 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 2.

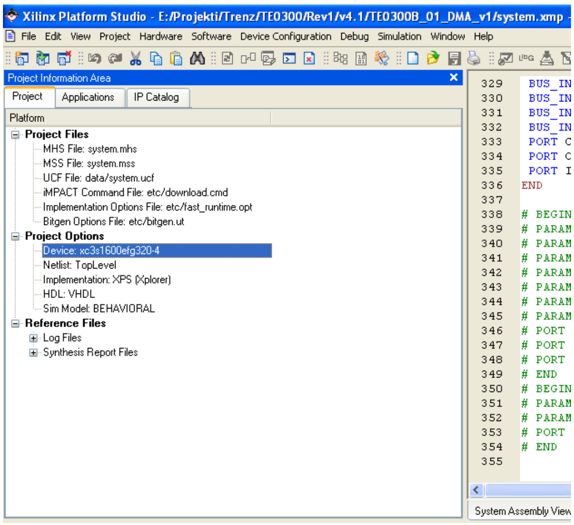
module version	kilo gates	env.	clock [MHZ]	memory (DDR SDRAM)
TE0300-01	1200	Com	125	Qimonda HYB25DC512160CF-6
TE0300-01M	1200	Com	125	Micron MT46V32M16BN-6:F
TE0300-01B	1600	Com	125	Qimonda HYB25DC512160CF-6
TE0300-01BM	1600	Com	125	Micron MT46V32M16BN-6:F
TE0300-01BLP	1600	Com	100	Qimonda HYB25DC512160CF-6
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 2: 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.

To change the FPGA device

- open the project in Xilinx Platform Studio
- click on the "Project" tab
- under "Project Options" double click on "Device":



Xilinx Platform Studio - E:/Projekti/Trenz/TE0300/Rev1/v4.1/TE0300B_01_DMA_v1/system.xmp -

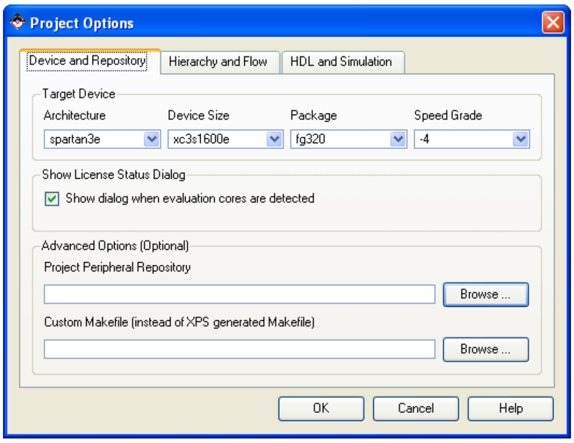
Project Files
- MHS File: system.mhs
- MSS File: system.mss
- UCF File: data/system.ucf
- iMPACT Command File: etc/download.cmd
- Implementation Options File: etc/fast_runtime.opt
- Bitgen Options File: etc/bitgen.ut

Project Options
- Device: xc3s1600efg320-4
- Netlist: TopLevel
- Implementation: XPS (Xplorer)
- HDL: VHDL
- Sim Model: BEHAVIORAL

Reference Files
- Log Files
- Synthesis Report Files

```
 BUS_IN
 BUS_IN
 BUS_IN
 BUS_IN
 PORT C
 PORT C
 PORT I
END

# BEGIN
# PARAM
# PARAM
# PARAM
# PARAM
# PARAM
# PARAM
# PARAM
# PORT
# PORT
# PORT
# END
# BEGIN
# PARAM
# PARAM
# PORT
# END
```

Select a suitable FPGA device:



Project Options

Target Device

Architecture	Device Size	Package	Speed Grade
spartan3e	xc3s1600e	fg320	-4

Show License Status Dialog

☑ Show dialog when evaluation cores are detected

Advanced Options (Optional)

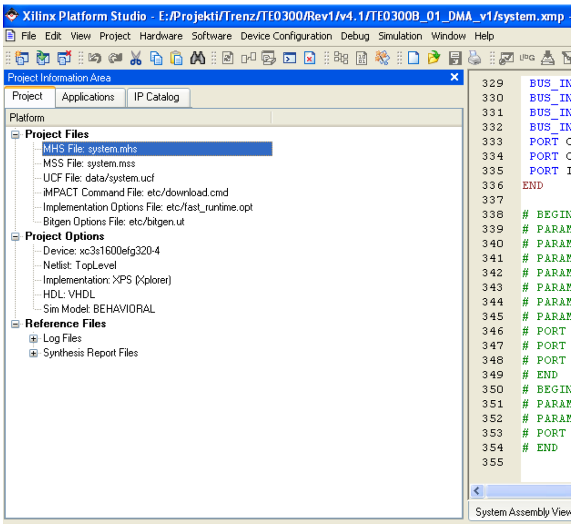
Project Peripheral Repository

Custom Makefile (instead of XPS generated Makefile)

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

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

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



Xilinx Platform Studio - E:/Projekti/Trenz/TE0300/Rev1/v4.1/TE0300B_01_DMA_v1/system.xmp

Platform

- Project Files
  - MHS File: system.mhs
  - MSS File: system.mss
  - UCF File: data/system.ucf
  - iMPACT Command File: etc/download.cmd
  - Implementation Options File: etc/fast_runtime.opt
  - Bitgen Options File: etc/bitgen.ut
- Project Options
  - Device: xc3s1600efg320-4
  - Netlist: TopLevel
  - Implementation: XPS (Xplorer)
  - HDL: VHDL
  - Sim Model: BEHAVIORAL
- Reference Files
  - Log Files
  - Synthesis Report Files

```
 BUS_IN
 BUS_IN
 BUS_IN
 BUS_IN
 PORT C
 PORT C
 PORT I
END

# BEGIN
# PARAM
# PARAM
# PARAM
# PARAM
# PARAM
# PARAM
# PARAM
# PORT
# PORT
# PORT
# END
# BEGIN
# PARAM
# PARAM
# PORT
# END
```

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
```

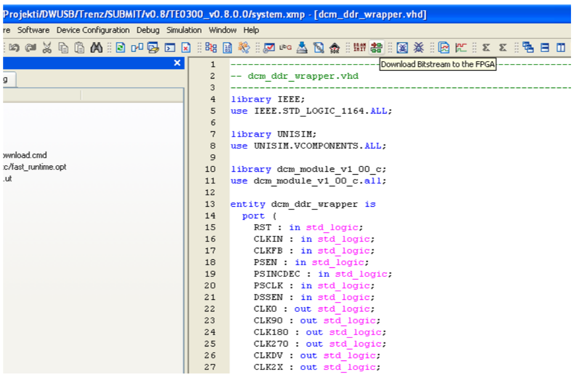
Adjust memory controller parameters to appropriate values:

```
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 = HYB25D512160BF-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:


Projekti/DWUSB/Trenz/SUBMIT/v0.8/TE0300_v0.8.0.0/system.xmp - [dcm_ddr_wrapper.vhd]
re Software Device Configuration Debug Simulation Window Help
Download Bitstream to the FPGA
ownload.cmd
c/fast_runtime.opt
.ut

```
-----------------------------------------------------------------
-- dcm_ddr_wrapper.vhd
-----------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

library dcm_module_v1_00_c;
use dcm_module_v1_00_c.all;

entity dcm_ddr_wrapper is
  port (
    RST : in std_logic;
    CLKIN : in std_logic;
    CLKFB : in std_logic;
    PSEN : in std_logic;
    PSINCDEC : in std_logic;
    PSCLK : in std_logic;
    DSSEN : in std_logic;
    CLK0 : out std_logic;
    CLK90 : out std_logic;
    CLK180 : out std_logic;
    CLK270 : out std_logic;
    CLKDV : out std_logic;
    CLK2X : out std_logic;
```

6 Revision History

Rev	Date	Who	Description
1.00	2009-06-22	FDR	Created
1.01	2009-10-20	FDR	Extended references to TE03xx family
1.02	2009-10-23	FDR	Generalized description about UCF file
1.03	2009-10-23	FDR	Now the reference designs do not need to be modified to work with any of the boards.
2.00	2010-03-09	FDR	Applied new template. Extended UART information.
2.01	2011-04-06	FDR	Improved UART information
2,02	2012-04-18	AIK	Added TE0630 UART information