department of signal processing

http://zs.utia.cas.cz

Application Note

ÚTIA Akademie věd České republiky
Ústav teorie informace a automatizace AV ČR, v.v.i.

Computation and Communication Blocks for Xilinx Kintex7 FPGA with UTIA EdkDSP Accelerators. Vivado 2013.4 Designs with SW Demos.

Jiří Kadlec
kadlec@utia.cas.cz
phone: +420 2 6605 2216
UTIA AV CR, v.v.i.

Revision history:

Rev.	**Date**	**Author**	**Description**
1	17.11.2014	Jiří Kadlec	Description of precompiled Vivado 2013.4 Kintex7 designs with EdkDSP accelerators and examples of use.
2	13.12.2014	Jiří Kadlec	Updated for EdkDSP accelerator clock 175 MHz

Acknowledgements:

This work has been partially supported by the Artemis JU project ALMARVI "Algorithms, Design Methods, and Many-Core Execution Platform for Low-Power Massive Data-Rate Video and Image Processing", project number Artemis JU 621439 and 7H14004 (Ministry of Education Youth and Sports of the Czech Republic [6].

Table of contents

Computation and Communication Blocks for Xilinx Kintex7 FPGA with UTIA EdkDSP Accelerators. Vivado 2013.4 Designs with SW Demos. 1
1. Summary 3
1.1 Communication and Computation blocks for Video and Image Processing 3
1.2 What is included 4
2. Description of EdkDSP Accelerators and Demos 5
2.1 Description of EdkDSP accelerators and evaluation designs 5
2.2 Resources used by the designs 9
2.3 Use of external DDR3 memory 10
2.4 Re-programmability of EdkDSP accelerators 10
2.5 Debug of evaluation designs with EdkDSP accelerators 10
3. Installation and use of the evaluation package 11
3.1 Import of precompiled HW and SW projects into Xilinx SDK 2013.4 11
3.2 Evaluation of demo projects 18
3.3 Ethernet point to point connection with PC 22
3.4 Boot of the bitstream 22
3.5 Boot of the application 22
3.6 Use of the C compiler for the EdkDSP firmware with download from Ethernet 35
3.7 Use of the C compiler for the EdkDSP firmware witout Ethernet 49
4. References 55
5. Evaluation version of Vivado 2013.4 Kintex7 designs 56
6. Release version of Vivado 2013.4 Kintex7 designs for Almarvi project partners 57
7. Release version of Vivado 2013.4 Kintex7 designs 59
Disclaimer 61

1. Summary

1.1 Communication and Computation blocks for Video and Image Processing

This application note describes precompiled Vivado 2013.4 Kintex7 designs with the floating point EdkDSP accelerators and examples of use of basic communication and computation blocks used in the video processing and image processing applications. The MicroBlaze SoC design with the AXI bus is based on the Xilinx BIST (build in self-test) provided by Xilinx for the Kintex7 KC705 board and the Vivado 2014.3 design flow. The network HW controller is supporting 1Gbit/100Mbit/10Mbit standards with HW DMA and a SW stack based on the lwIP library described in the Xilinx application note XAPP1026 [3], [4]. The MicroBlaze processor is controlling 8 EdkDSP floating point accelerators. Each accelerator is organised as 8xSIMD reconfigurable data path, controlled by a PicoBlaze6 controller. This evaluation package is provided by UTIA for the Xilinx KC705 board with the 28nm Kintex7 xc7k325t-2 FPGA part. This application note explains how to install and use the demonstrator on Windows7, (32 or 64 bit) and Xilinx KC705 board [1], [2]. The evaluation package can be also installed and used on PC running Linux (32 or 64 bit) or Win XP (32 or 64 bit) with the corresponding Xilinx SDK 2013.4.

These key features are demonstrated:

- WWW server running on Kintex7 KC705 board with the lwIP stack running in RAW mode or SOCKET mode with the Xilkernel support of POSIX compatible threads.
- TFTP server running on Kintex7 KC705 board with the lwIP stack running in RAW mode or SOCKET mode.
- RAM based file system with files in the DDR3 memory on the KC705 board.
- 8 reprogrammable floating point accelerators for local embedded computing on the Kintex7 28nm chip.
- Demo implementation of an adaptive acoustic noise cancellation on 1 of 8 accelerators is computing the recursive adaptive LMS algorithm for identification of regression filter with 2000 coefficients in single precision floating point arithmetic with this sustained performance
    - 1012,0 MFLOP/s on a single 175 MHz (8xSIMD) EdkDSP accelerator (only 1 of the 8 units is used)
    - 7,6 MFLOP/s on the 100 MHz MicroBlaze processor with the floating point HW unit
- The EdkDSP accelerators can be reprogrammed by the firmware. The programming is possible in C with the use of the UTIA EDKDSP C compiler. Accelerators can be programmed with two firmware programs. Designs can swap in the real time the firmware in only few clock cycles in the runtime.
- The alternative firmware can be downloaded to the EdkDSP accelerators from the internet in parallel with the execution of the current firmware. This is demonstrated by the download of firmware by the TFTP server and by swap of the firmware for the FIR filter room-response to the firmware for the adaptive LMS identification of the filter coefficients in the acoustic noise cancellation demo.
- The EdkDSP accelerator is providing single-precision floating point results bit-exact identical to the reference software implementations running on the MicroBlaze with the Xilinx HW single precision floating point unit.
- Single 175 MHz (8xSIMD) EdkDSP accelerator is 132x faster than computation on the performance optimized 100 MHz MicroBlaze with HW floating point unit, in the presented case of the 2000 tap adaptive LMS filter.
- The floating point 2000 tap coefficients FIR filter (acoustics room model) is computed by single 175 MHz (8xSIMD) EdkDSP accelerator with the floating point performance of 1403 MFLOP/s. The peak performance (only theoretical) of a single 175 MHz (8xSIMD) EdkDSP accelerator is 2,8 GFLOP/s.
- The peak performance of eight 175 MHz (8xSIMD) EdkDSP accelerators implemented in this demo design is 22,4 GFLOP/s (this is only theoretical peek figure).
- This evaluation package presents two (8xSIMD) EdkDSP accelerator families: one family without pipelined floating point divider data path and one family with a single pipelined floating point divider data path. The members of both families differ by size and by supported vector floating point operations.
- The floating point applications are scheduled inside of the EdkDSP accelerator by the Xilinx PicoBlaze6 processor [5]. Each PicoBlaze6 firmware program has maximal size of 4096 (18 bit wide words).

1.2 What is included

The evaluation package includes precompiled Vivado 2013.4 Kintex7 designs with floating point EdkDSP accelerators and SW examples of communication and computation in form of Xilinx SDK 2013.4 SW projects for Windows 7 (32 or 64bit) or a PC running Linux (32 or 64 bit) or Win XP (32 or 64 bit) with the corresponding installation of Xilinx SDK 2013.4:

- 8 evaluation versions of precompiled Kintex7 designs. Each design contains one MicroBlaze and eight instances of the EdkDSP accelerators. Each accelerator has 8xSIMD floating point data paths and programmable PicoBlaze6 controller for scheduling of floating point vector operations in the accelerator. The MicroBlaze works with 100 MHz system clock and EdkDSP accelerators use 175 MHz clock. The Microblaze processor works with 1 Gb Ethernet with DMA controller and 1 GB DDR3 memory. Designs are compiled in Xilinx Vivado 2013.4.
- UTIA is providing source code for the demo applications and SW projects for the Xilinx SDK 2013.4. These source code projects are compiled with the UTIA library libwal.a serving for the EdkDSP communication and the library libmfsimage.a with the initial file system supporting the simple www server GUI.
- The included evaluation versions of the UTIA EdkDSP accelerators have HW limitation of maximal number of performed vector operations.
- The UTIA EDKDSPC C compiler is provided as 4 binary applications for Ubuntu in the VMware Player.
- The firmware for accelerators is provided in source code and also in format of binary files to enable the initial evaluation of the EdkDSP accelerators without the need to install the EDKDSPCC C compiler.

- UTIA partners of the Artemis Almarvi [6] project can get from UTIA the release version of Vivado 2013.4 HW design projects with the evaluation versions of the EdkDSP accelerators (in the Vivado 2013.4 IP netlist format) for free. See chapter 6 for specification of deliverables for the Artemis Almarvi [6] project partners with license details.

- Release versions of Vivado 2013.4 HW design projects and release version of EdkDSP accelerators for the Xilinx KC705 board is offered by UTIA. All customers can order and buy from UTIA the release version of this demo. It includes the Vivado 2013.4 HW design projects with the EdkDSP accelerators (in the Vivado 2013.4 IP netlist format) with the HW limitation of maximal number of performed vector operations removed. See sections 7 of this application note for specification of deliverables and license details.

2. Description of EdkDSP Accelerators and Demos

2.1 Description of EdkDSP accelerators and evaluation designs

This application note describes how to set-up and use of 8 HW designs running on one MicroBlaze processor, each design with eight (8xSIMD) EdkDSP accelerators on Xilinx KC705 board. See Figure 1 and Figure 2.

Demonstrators serve for evaluation of the communication and computation processor infrastructure. Utia is providing two floating point accelerator families for the Xilinx Kintex7 xc7k325t-2 part:

- **bce_fp11_1x8_0_axiw_v1_[10|20|30|40]_b** ibs a family of four versions of floating point EdkDSP accelerators with 8 SIMD data paths.
- **bce_fp12_1x8_0_axiw_v1_[10|20|30|40]_b** is similar family of four versions of floating point EdkDSP accelerators with 8 SIMD data paths extended by a pipelined floating point division (FPDIV) in a single data path.

The four grades [10|20|30|40] of the EdkDSP accelerator differ in HW-supported vector computing capabilities:

The area optimized accelerators bce_fp11_1x8_0_axiw_v1_10_b and bce_fp12_1x8_0_axiw_v1_10_b perform vector floating point operations FPADD, FPSUB in 8 SIMD data paths.

The accelerators bce_fp11_1x8_0_axiw_v1_20_b and bce_fp12_1x8_0_axiw_v1_20_b perform vector floating point operations FPADD, FPSUB in 8 SIMD data paths plus the vector floating point MAC operations in 8 SIMD data paths for length of the vector 1 up to 10. These accelerators can be used in applications like floating point matrix multiplication with row and column dimensions <= 10.

The accelerators bce_fp11_1x8_0_axiw_v1_30_b and bce_fp12_1x8_0_axiw_v1_30_b support identical operations as the bce_fp11_1x8_0_axiw_v1_20_b and bce_fp12_1x8_0_axiw_v1_20_b plus the floating point vector by vector dot products performed in 8 SIMD data paths. It is optimized for parallel computation of up to 8 FIR or LMS filters, each with size up to 255 coefficients. It is also effective in case of floating point matrix by matrix multiplications, where one of the dimensions is large (in the range from 11 to 255).

Finally, the accelerators bce_fp11_1x8_0_axiw_v1_40_b and bce_fp12_1x8_0_axiw_v1_40_b support identical operations as the bce_fp11_1x8_0_axiw_v1_30_b and bce_fp12_1x8_0_axiw_v1_30_b plus an additional HW support of dot product. It is computed in 8 data paths with the HW supported wind-up into single scalar result.

The bce_fp11 versions of 8xSIMD accelerators has no support for pipelined vector floating point division and it is suitable for applications like FIR filters or adaptive LMS filters with no need for floating point division.

The bce_fp12 versions of 8xSIMD accelerators are larger in comparison to the bce_fp11 equivalents and support in a single data path the pipelined vector floating point division. Accelerators are suitable for applications like adaptive normalised NLMS filters and the square root free versions of adaptive RLS QR filters and adaptive RLS LATTICE filters.

See Figure 3 and Figure 4 for the resources used by the evaluation designs included in this package.

Photo 1: Demonstration of 1 Gb ethernet, www server and TFTP server with 8x (8xSIMD) EdkDSP floating point accelerators on Xilinx KC705 board with Kintex7 FPGA.

Ten HW designs precompiled in Vivado 2013.4 combine MicroBlaze and eight 8xSIMD EdkDSP accelerators. All designs demonstrate use of single instance of 8xSIMD EdkDSP floating point accelerator on 32bit AXI-lite bus of the Xilinx MicroBlaze soft-core processor on the Xilinx Kintex7 KC705 FPGA board with system clock of MicroBlaze 100 MHz and EdkDSP accelerators 175 MHz. See Figure 2.

Common properties of precompiled Vivado 2013.4 evaluation designs:

- The EdkDSP floating point accelerators are reconfigurable during runtime by change of firmware.
- All HW evaluation designs have been compiled in Xilinx VIVADO 2013.4 with SW projects for SDK 2013.4.

Presented HW accelerators can results in better POWER per MFLOPS ratio for certain class of DSP applications in comparison to the computation on MicroBlaze with HW floating point support.

The demonstrator includes source code of set of SW demos prepared for easy import of projects and compilation in the Xilinx SDK 2013.4.

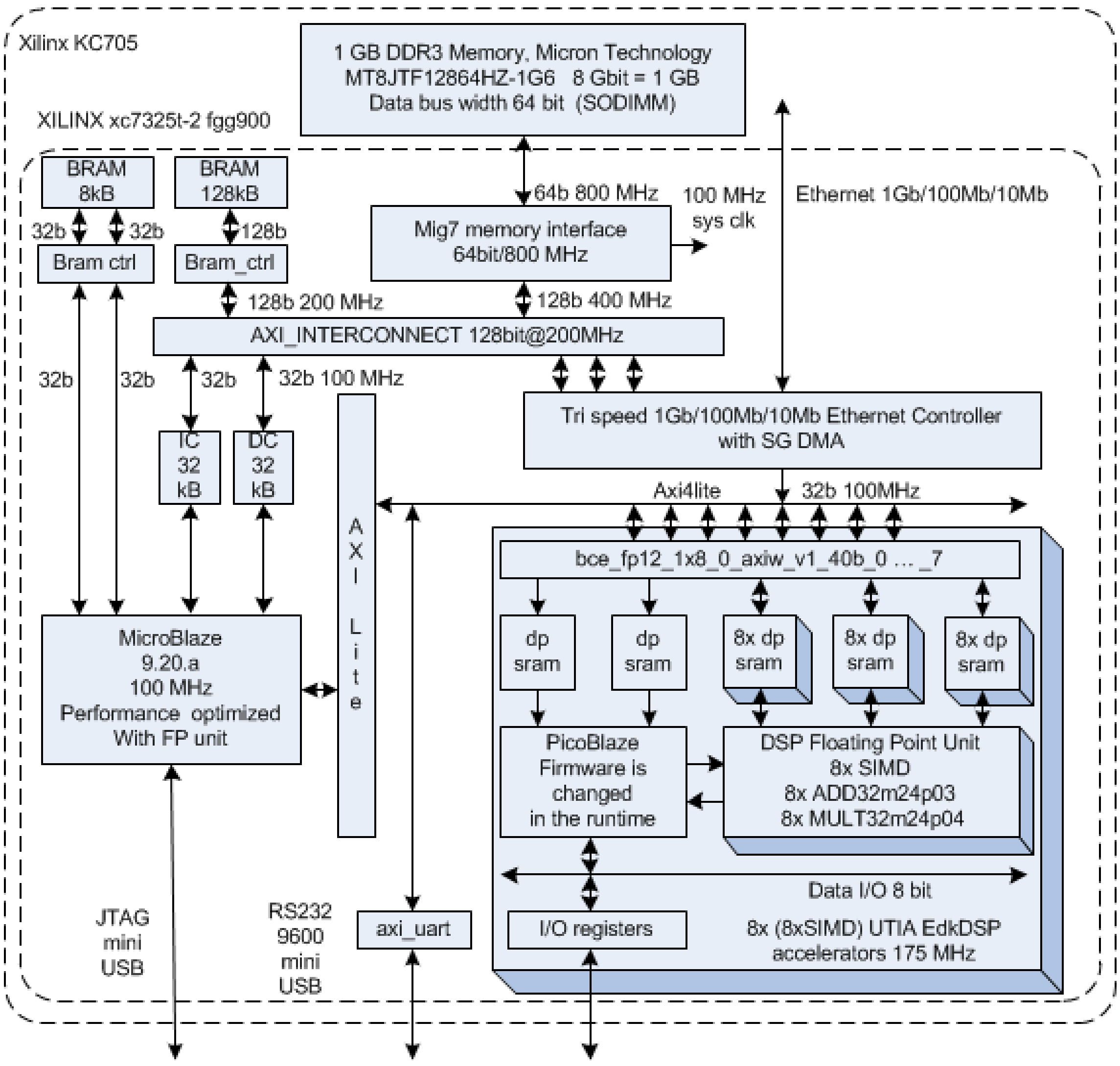

Xilinx KC705
1 GB DDR3 Memory, Micron Technology MT8JTF12864HZ-1G6 8 Gbit = 1 GB Data bus width 64 bit (SODIMM)
XILINX xc7325t-2 fgg900
BRAM 8kB
BRAM 128kB
32b
32b
128b
Bram ctrl
Bram_ctrl
64b 800 MHz
100 MHz sys clk
Ethernet 1Gb/100Mb/10Mb
Mig7 memory interface 64bit/800 MHz
128b 200 MHz
128b 400 MHz
AXI_INTERCONNECT 128bit@200MHz
32b
32b
32b
32b 100 MHz
IC 32 kB
DC 32 kB
AXI Lite
Tri speed 1Gb/100Mb/10Mb Ethernet Controller with SG DMA
Axi4lite
32b 100MHz
bce_fp12_1x8_0_axiw_v1_40b_0 ... _7
dp sram
dp sram
8x dp sram
8x dp sram
8x dp sram
MicroBlaze 9.20.a 100 MHz Performance optimized With FP unit
PicoBlaze Firmware is changed in the runtime
DSP Floating Point Unit 8x SIMD 8x ADD32m24p03 8x MULT32m24p04
Data I/O 8 bit
JTAG mini USB
RS232 9600 mini USB
axi_uart
I/O registers
8x (8xSIMD) UTIA EdkDSP accelerators 175 MHz

Figure 1: Key building blocks of the Kintex demonstrator with 8 EdkDSP accelerators.

Figure 1 describes the the key building blocks of the design. It is SoC with MicroBlaze, Tri speed ethernet (1Gb/100Mb/10Mb) with SG central DMA and eight EdkDSP accelerators. Each accelerator works with 8xSIMD floating point unit controlled by a reprogrammable PicoBlaze6 controller. There are in total nine programmable processors, all capable of HW supported parallel floating point computation.

Demo designs are based on the modified Vivado 2013.4 KC705 BIST (built-in self-test) reference design from Xilinx. See the complete design in the IP Integrator on Figure 2.

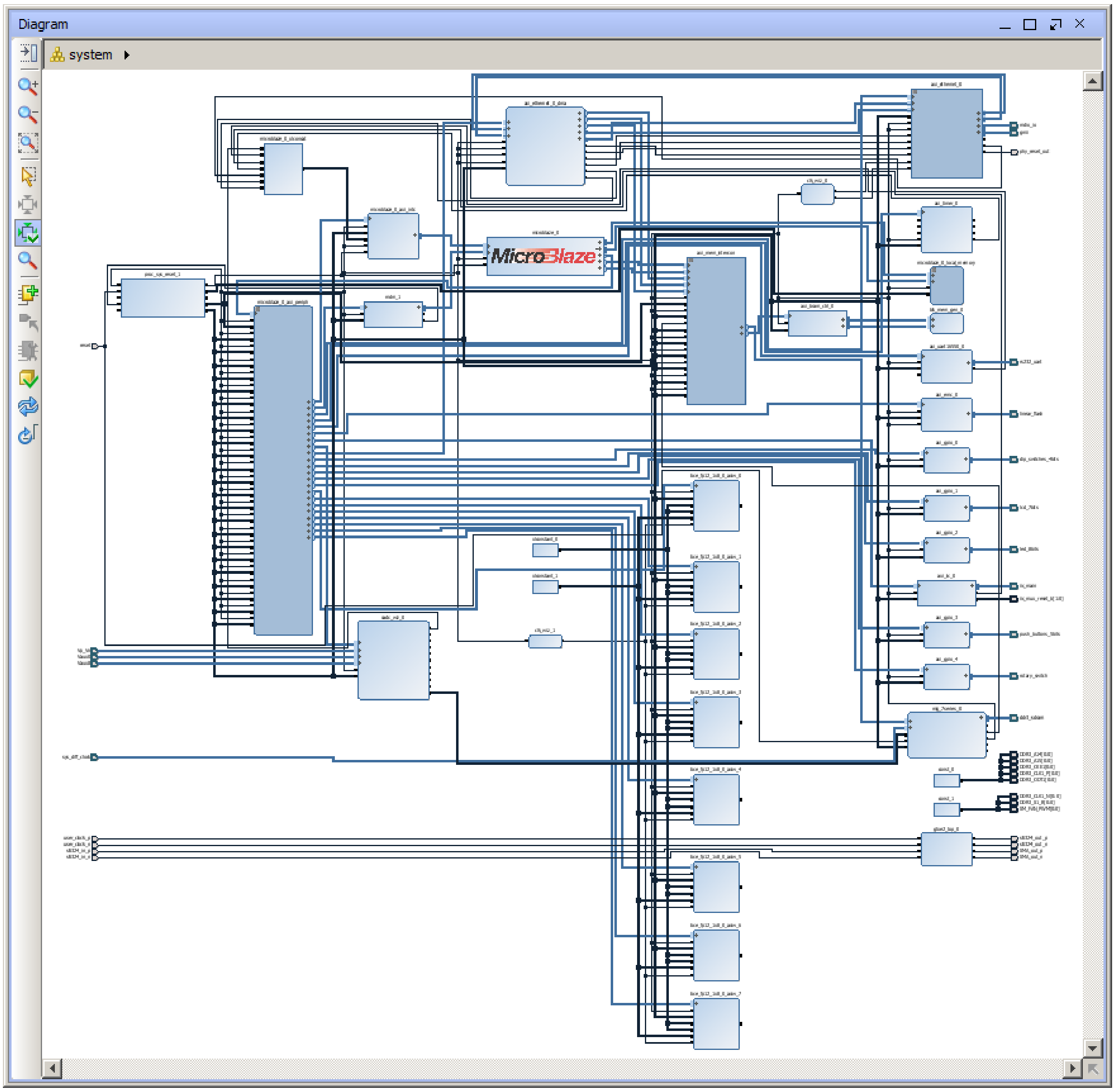


Figure 2: Design with 8 EdkDSP accelerators in Xilinx Vivado; 2013.4 IP Integrator.

Figure 2 describes the SoC with MicroBlaze, 1Gb Ethernet and eight EdkDSP accelerators. The design is based on the Vivado 2013.4 KC705 BIST (built-in self-test) reference design from Xilinx.

The initial MicroBlaze boot block RAM is set to 32 KB. The internal program and data RAM memory is set to 128 KB size. Data width of this AXI interfaced memory is set to 128 bit to support burst operations.

The 8 EdkDSP (8xSIMD) floating point accelerators are memory mapped on the 32 bit AXI-lite bus.
Each accelerator has reserved 1 MB of address space. See Figure 3 and Figure 4 for the resources used by the designs.

2.2 Resources used by the designs

The resources used by the 8 presented designs are summarised in Figure 3 and Figure 4.

7k325t-2	fp	fp	fp	fp	fp	Design size			Performance	
	Add Mul	Mac	Dot Prod	S8 Prod	Div	FFs %	LUTs %	Bram No (of)	LMS Mflop/s	FIR Mflop/s
kc705_bist						7	15	63 (445)		
(6x) fp11_1x8_10	8x					15	41	303 (445)		
(8x) fp11_1x8_20	8x	8x				17	44	303 (445)		
(8x) fp11_1x8_30	8x	8x	8x			19	51	303 (445)		
(8x) fp11_1x8_40	8x	8x	8x	1x		19	52	303 (445)	**(8x) 1012**	**(8x) 1403**

Figure 3: Resources used by MicroBlaze and 8x (8xSIMD) EdkDSP, no FP division

7k325t-2	fp	fp	fp	fp	fp	Design size			Performance	
	Add Mul	Mac	Dot Prod	S8 Prod	Div	FFs %	LUTs %	Bram No (of)	LMS Mflop/s	FIR Mflop/s
kc705_bist						7	15	63 (445)		
(8x) fp12_1x8_10	8x				1x	17	45	303 (445)		
(8x) fp12_1x8_20	8x	8x			1x	19	48	303 (445)		
(8x) fp12_1x8_30	8x	8x	8x		1x	21	55	303 (445)		
(8x) fp12_1x8_40	8x	8x	8x	1x	1x	21	56	303 (445)	**(8x) 1012**	**(8x) 1403**

Figure 4: Resources used by MicroBlaze and 8x (8xSIMD) EdkDSP, with FP division

The kc705_bist design describes resources used by the MicroBlaze SoC without EdkDSP accelerators. The internal block RAM memory is set to 32 KB and 128 KB. Please, notice, that the Xilinx reference kc705_bist design [2] works with internal block RAM memory set to 32 KB and 1 MB.

All designs with EdkDSP accelerators (fp11 and fp12) work with

- 64 single precision 3-stage pipelined floating point add/sub units each performing up to 175 MFLOP/s
- 64 single precision 4-stage pipelined floating point multiply units each performing up to 175 MFLOP/s
- 8 PicoBlaze6 controllers with 175 MHz system clock, each executing 87,5 Mil. instructions/s
- The 100 MHz MicroBlaze performance optimized processor is using one single precision 3-stage pipelined floating point add/sub unit and one single precision 4-stage pipelined floating point multiply unit, 32 KB data cache and 32 KB instruction cache.

The fp12 designs work in addition with

- 8 single precision 16-stage pipelined floating point divide units each performing up to 175 MFLOP/s.

Designs use accelerators with different HW supported operation. This is reflected in the difference of resources used by the designs. See Figure 3 and Figure 4.

2.3 Use of external DDR3 memory

Presented FPGA designs are running on the Xilinx KC705 development board [1], [2]. See Figure 1. It is using the 1 GB DDR3 memory with clock signal 800 MHz. The DDR3 is connected to Xilinx Kintex7 xc7k325t-2 FPGA by 64 bit wide data path. The maximal theoretical peak performance of this external DDR3 memory with 800 MHz clock is therefore 1600(DDR2 transactions) x 8(bytes) = 12,8 GB/s. This is 102,4 Gb/s.

2.4 Re-programmability of EdkDSP accelerators

Each (8xSIMD) EdkDSP floating point accelerator subsystem contains one reprogrammable Xilinx PicoBlaze6 8-bit controller and the floating point (8xSIMD) DSP unit. The performance of the accelerator is application specific. In this demo, a single (8xSIMD) EdkDSP unit is delivering sustained 1403 MFLOP/s in case of 2000 tap FIR filter computation and 1012 MFLOP/s in case of the adaptive 2000 tap LMS filter identification demo. All designs have eight (8xSIMD) EdkDSP units.

The Xilinx PicoBlaze6 processor has fixed configuration with size of the program memory 4096 (18 bit wide) words, 64 Bytes scratch pad RAM memory and the interrupt vector in the address 1023.

The (8xSIMD) EdkDSP accelerator works with 2 program memories. Each program memory has 4096 (18bit wide) words. Both program memories are accessible by MicroBlaze processor via AXI-lite bus. The MicroBlaze application can write new firmware to the currently unused program memory, while the PicoBlaze6 is executing firmware from the second program memory.

The peak performance of data memories of all 8 (8xSIMD) EdkDSP accelerators in the included evaluation designs is 175 MHz (clock) x 4(bytes) x 3(mems) x 8(simd) x 8(instances) = 134,4 GB/s. This is 1075,2 Gb/s.

2.5 Debug of evaluation designs with EdkDSP accelerators

All EdkDSP accelerators can communicate with MicroBlaze program. The communication is using the Worker Abstraction Layer (WAL) library API. This API is used for support of writing of the debug information from the worker to the MicroBlaze terminal.

The PicoBlaze6 processors [5] can exchange data and text via the 8 bit communication data path with the MicroBlaze processor. This path is used to communicate parameters to the accelerators and to get messages or reports from accelerators for debugging. Text file with information from the accelerator can be stored in the RAM based file system of MicroBlaze. It can be downloaded to PC via Ethernet for inspection.

Floating point data are accessed by the MicroBlaze processor via the dual ported block memories of accelerators. The MicroBlaze side of the dual-ported memories is mapped into the MicroBlaze memory. The MicroBlaze processor can copy data from the dual ported memories to the DDR3 global workspace and display floating point data in the debugger. The computation in the (8xSIMD) EdkDSP units can overlap with the communication with the DDR3 performed by MicroBlaze. It is supported by data and program cache.

A Ping-Pong swap of memory banks is used by the accelerator firmware. The (8xSIMD) EdkDSP firmware is computing (in parallel) in some banks of all dual ported memories and the MicroBlaze is communicating (sequentially) to/from DDR3 in another set of banks of the dual-ported memories. This process can be stopped, inspected and debugged by the MicroBlaze debugger from the SDK 2013.4.

3. Installation and use of the evaluation package

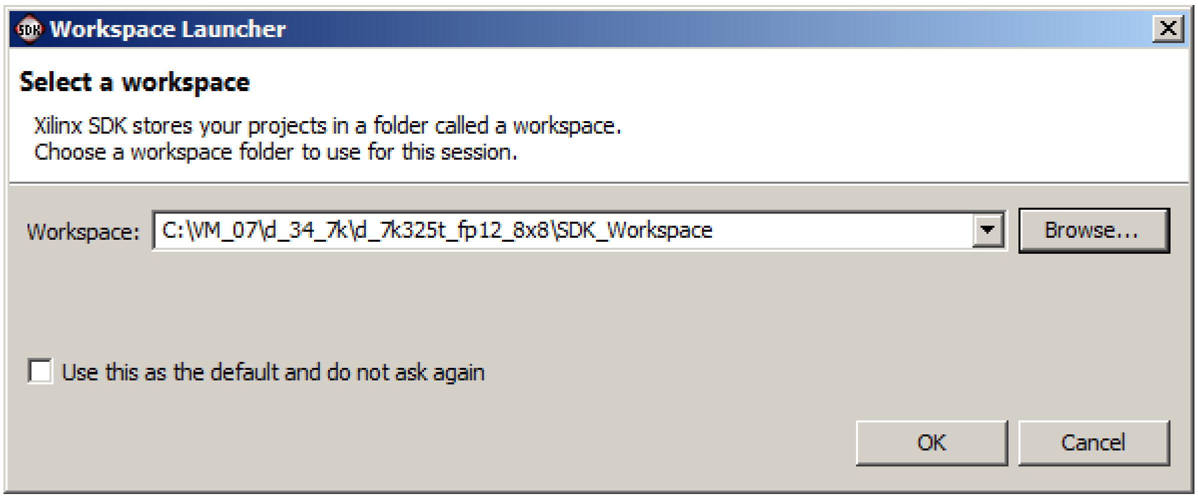
3.1 Import of precompiled HW and SW projects into Xilinx SDK 2013.4

Unzip the evaluation package to directory of your choice. The directory c:\VM_07 will be used in this application note. You will get these directories:

c:\VM_07\d_34_7k

```
01.11.2014  16:11   <DIR>       .
01.11.2014  16:11   <DIR>       ..
01.11.2014  16:09   <DIR>       d_7k325t_fp11_6x8
01.11.2014  16:09   <DIR>       d_7k325t_fp11_6x8_IMPORT
01.11.2014  16:12   <DIR>       d_7k325t_fp11_6x8_v1_10b
01.11.2014  16:12   <DIR>       d_7k325t_fp11_6x8_v1_20b
01.11.2014  16:12   <DIR>       d_7k325t_fp11_6x8_v1_30b
01.11.2014  16:12   <DIR>       d_7k325t_fp11_6x8_v1_40b
01.11.2014  16:10   <DIR>       d_7k325t_fp12_6x8
01.11.2014  16:10   <DIR>       d_7k325t_fp12_6x8_IMPORT
31.10.2014  14:25   <DIR>       d_7k325t_fp12_6x8_v1_10b
31.10.2014  14:24   <DIR>       d_7k325t_fp12_6x8_v1_20b
31.10.2014  14:24   <DIR>       d_7k325t_fp12_6x8_v1_30b
31.10.2014  14:23   <DIR>       d_7k325t_fp12_6x8_v1_40b
```

Select SDK 2013.4 workspace in c:\VM_07\d_34_7k\d_7k325t_fp12_6x8\SDK_Workspace.
See Figure 5.



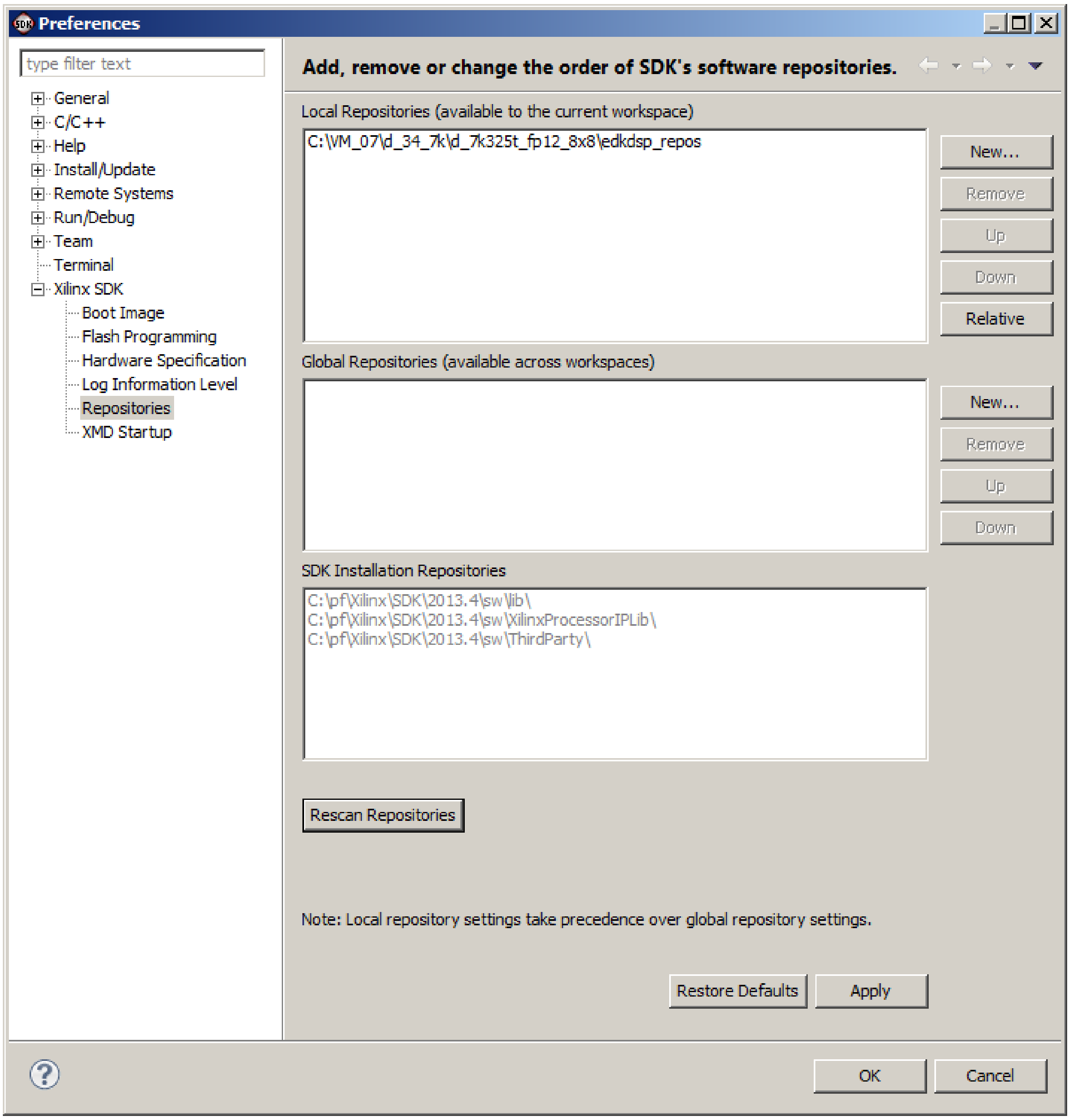
Select a workspace

Xilinx SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\VM_07\d_34_7k\d_7k325t_fp12_8x8\SDK_Workspace

Figure 5: Select the SDK Workspace

Add **c:\VM_07\d_34_7k\d_7k325t_fp12_6x8\repo_edkdsp** path to the UTIA EdkDSP repository. See Figure 6.



Add, remove or change the order of SDK's software repositories.

Local Repositories (available to the current workspace)

C:\VM_07\d_34_7k\d_7k325t_fp12_8x8\edkdsp_repos

Global Repositories (available across workspaces)

SDK Installation Repositories

C:\pf\Xilinx\SDK\2013.4\sw\lib\
C:\pf\Xilinx\SDK\2013.4\sw\XilinxProcessorIPLib\
C:\pf\Xilinx\SDK\2013.4\sw\ThirdParty\

Note: Local repository settings take precedence over global repository settings.

Figure 6: Include the UTIA EdkDSP Repository

Click on the “Rescan Repositories” button. Click on the “Apply button”, and finally click on the OK button. The path to the SW drivers has been defined.

In SDK, select File -> New -> Project … -> Xilinx -> Hardware Platform Specification. See Figure 7.
Click on the Next button.

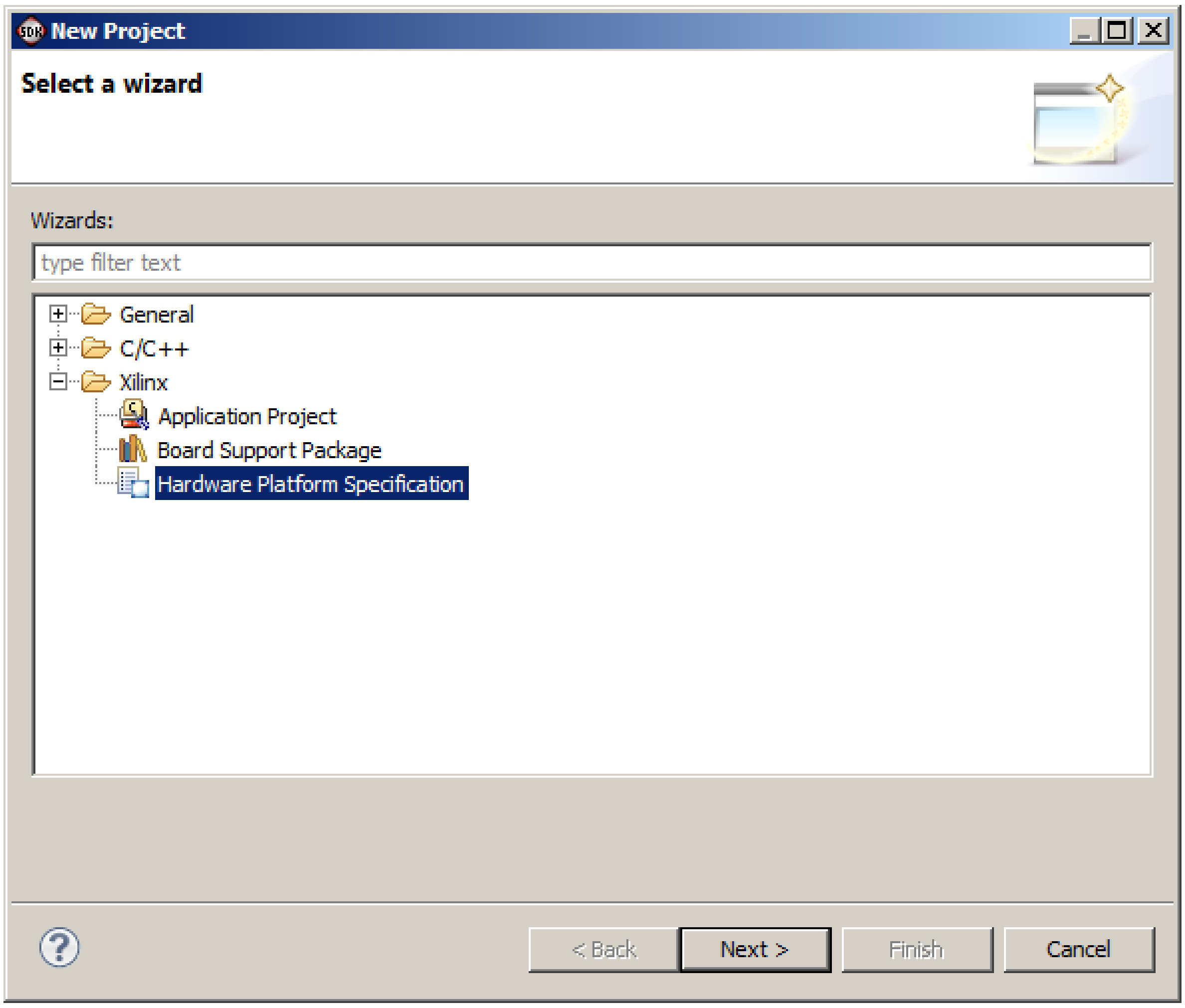


Figure 7: Specify the hardware platform

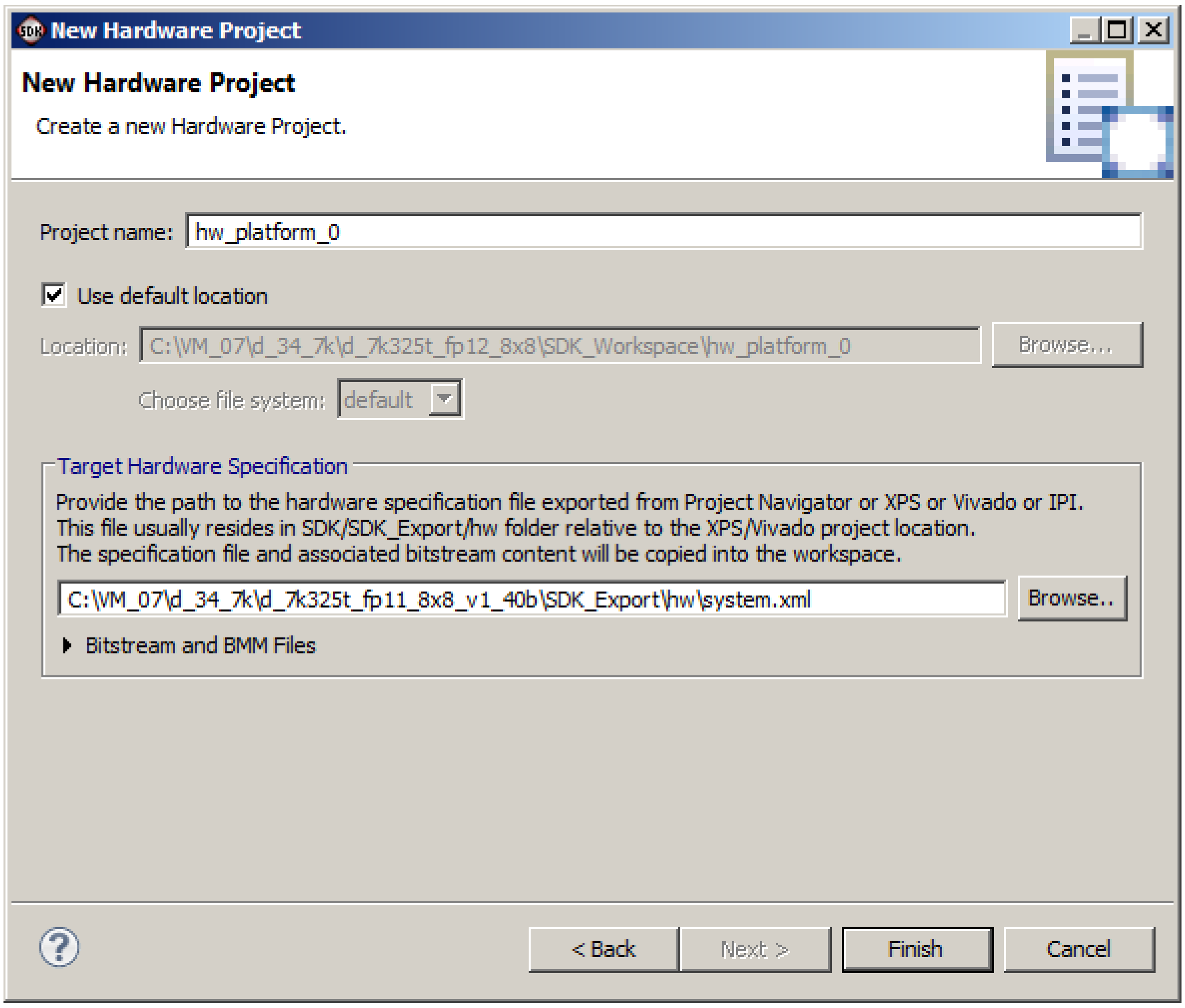
In the "New Hardware Project" screen, fill into the Project name: hw_platform_0
In the New Hardware Project screen, fill into the Target Hardware Specification:

c:\VM_07\d_34_7k\d_7k325t_fp12_6x8_v1_40b\SDK\SDK_Export\hw\system.xml

This will specify one of the 8 precompiled HW designs present in the evaluation package. See Figure 8.

We have selected the **d_7k325t_fp12_6x8_v1_40b** design, demonstrating the use of eight instances the UTIA EdkDSP accelerators, all with 8xSIMD data path, with floating point single data path division. All eight (8xSIMD) accelerators compiled in this design have identical capabilities defined by the IP core: bce_fp12_1x8_0_axiw_v1_40_b.

Click on "Finish" button to finalize the selection of the precompiled HW design. See Figure 8.


New Hardware Project

New Hardware Project

Create a new Hardware Project.

Project name: hw_platform_0

[✓] Use default location

Location: C:\VM_07\d_34_7k\d_7k325t_fp12_8x8\SDK_Workspace\hw_platform_0 Browse...

Choose file system: default

Target Hardware Specification

Provide the path to the hardware specification file exported from Project Navigator or XPS or Vivado or IPI.
This file usually resides in SDK/SDK_Export/hw folder relative to the XPS/Vivado project location.
The specification file and associated bitstream content will be copied into the workspace.

C:\VM_07\d_34_7k\d_7k325t_fp11_8x8_v1_40b\SDK_Export\hw\system.xml Browse..

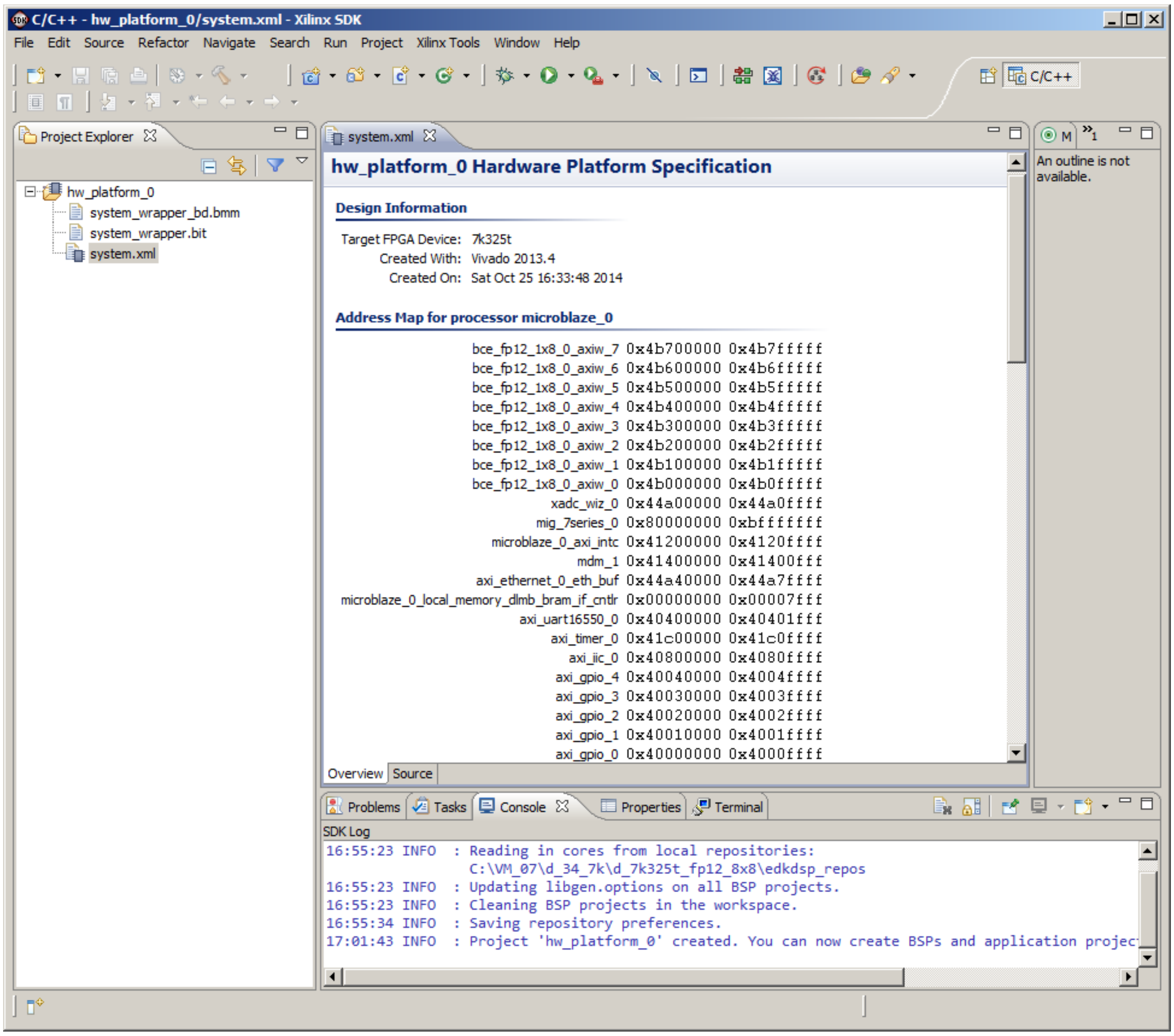
Bitstream and BMM Files

< Back | Next > | Finish | Cancel

Figure 8: Use the name "hw_platform_0" and select one of the provided xml design descriptions

SDK is interpreting the system.xml and presents HW cores of in the design. See Figure 9.

The hardware platform "hw_platform_0" has been created.


C/C++ - hw_platform_0/system.xml - Xilinx SDK
File Edit Source Refactor Navigate Search Run Project Xilinx Tools Window Help
C/C++
Project Explorer
hw_platform_0
system_wrapper_bd.bmm
system_wrapper.bit
system.xml
system.xml
An outline is not available.
Overview
Source
Problems
Tasks
Console
Properties
Terminal

hw_platform_0 Hardware Platform Specification

Design Information

Target FPGA Device:	7k325t
Created With:	Vivado 2013.4
Created On:	Sat Oct 25 16:33:48 2014

Address Map for processor microblaze_0

bce_fp12_1x8_0_axiw_7	0x4b700000	0x4b7fffff
bce_fp12_1x8_0_axiw_6	0x4b600000	0x4b6fffff
bce_fp12_1x8_0_axiw_5	0x4b500000	0x4b5fffff
bce_fp12_1x8_0_axiw_4	0x4b400000	0x4b4fffff
bce_fp12_1x8_0_axiw_3	0x4b300000	0x4b3fffff
bce_fp12_1x8_0_axiw_2	0x4b200000	0x4b2fffff
bce_fp12_1x8_0_axiw_1	0x4b100000	0x4b1fffff
bce_fp12_1x8_0_axiw_0	0x4b000000	0x4b0fffff
xadc_wiz_0	0x44a00000	0x44a0ffff
mig_7series_0	0x80000000	0xbfffffff
microblaze_0_axi_intc	0x41200000	0x4120ffff
mdm_1	0x41400000	0x41400fff
axi_ethernet_0_eth_buf	0x44a40000	0x44a7ffff
microblaze_0_local_memory_dlmb_bram_if_cntlr	0x00000000	0x00007fff
axi_uart16550_0	0x40400000	0x40401fff
axi_timer_0	0x41c00000	0x41c0ffff
axi_iic_0	0x40800000	0x4080ffff
axi_gpio_4	0x40040000	0x4004ffff
axi_gpio_3	0x40030000	0x4003ffff
axi_gpio_2	0x40020000	0x4002ffff
axi_gpio_1	0x40010000	0x4001ffff
axi_gpio_0	0x40000000	0x4000ffff

SDK Log
```
16:55:23 INFO  : Reading in cores from local repositories:
                 C:\VM_07\d_34_7k\d_7k325t_fp12_8x8\edkdsp_repos
16:55:23 INFO  : Updating libgen.options on all BSP projects.
16:55:23 INFO  : Cleaning BSP projects in the workspace.
16:55:34 INFO  : Saving repository preferences.
17:01:43 INFO  : Project 'hw_platform_0' created. You can now create BSPs and application projec
```

Figure 9: Hardware platform with the MicroBlaze processor and the address map

SW projects can be imported into SDK now. Select:

File -> Import -> General -> Existing Projects into Workspace
Click on Next button. See Figure 10.

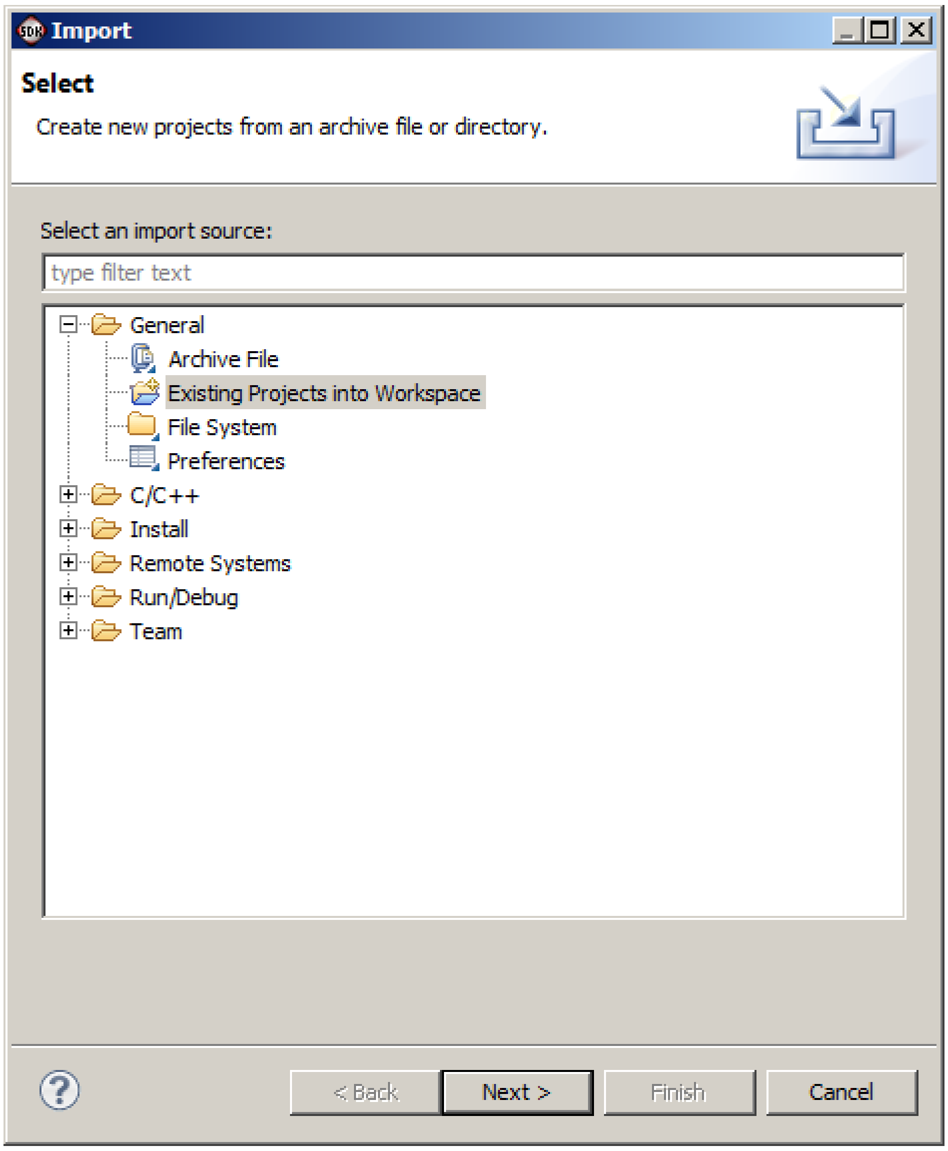
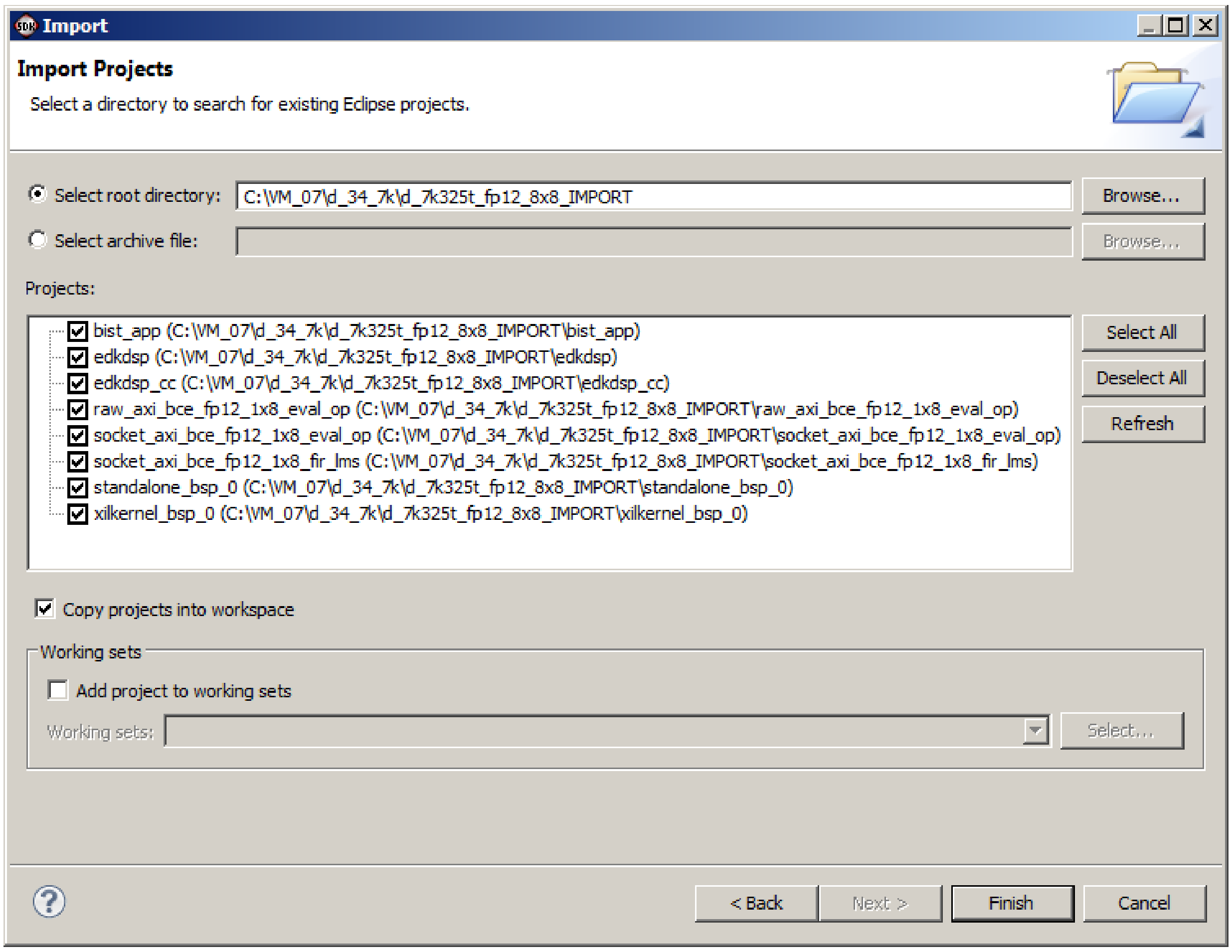


Figure 10: Import existing projects into workspace

Select the directory with projects to be imported. See Figure 11.

c:\VM_07\d_34_7k\d_7k325t_fp12_6x8_IMPORT

Set the "Copy projects into workspace" check box.
Click on Finish button. See Figure 11.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

(●) Select root directory: C:\VM_07\d_34_7k\d_7k325t_fp12_8x8_IMPORT

( ) Select archive file:

Projects:

- [✓] bist_app (C:\VM_07\d_34_7k\d_7k325t_fp12_8x8_IMPORT\bist_app)
- [✓] edkdsp (C:\VM_07\d_34_7k\d_7k325t_fp12_8x8_IMPORT\edkdsp)
- [✓] edkdsp_cc (C:\VM_07\d_34_7k\d_7k325t_fp12_8x8_IMPORT\edkdsp_cc)
- [✓] raw_axi_bce_fp12_1x8_eval_op (C:\VM_07\d_34_7k\d_7k325t_fp12_8x8_IMPORT\raw_axi_bce_fp12_1x8_eval_op)
- [✓] socket_axi_bce_fp12_1x8_eval_op (C:\VM_07\d_34_7k\d_7k325t_fp12_8x8_IMPORT\socket_axi_bce_fp12_1x8_eval_op)
- [✓] socket_axi_bce_fp12_1x8_fir_lms (C:\VM_07\d_34_7k\d_7k325t_fp12_8x8_IMPORT\socket_axi_bce_fp12_1x8_fir_lms)
- [✓] standalone_bsp_0 (C:\VM_07\d_34_7k\d_7k325t_fp12_8x8_IMPORT\standalone_bsp_0)
- [✓] xilkernel_bsp_0 (C:\VM_07\d_34_7k\d_7k325t_fp12_8x8_IMPORT\xilkernel_bsp_0)

[✓] Copy projects into workspace

Working sets

[ ] Add project to working sets

Working sets:

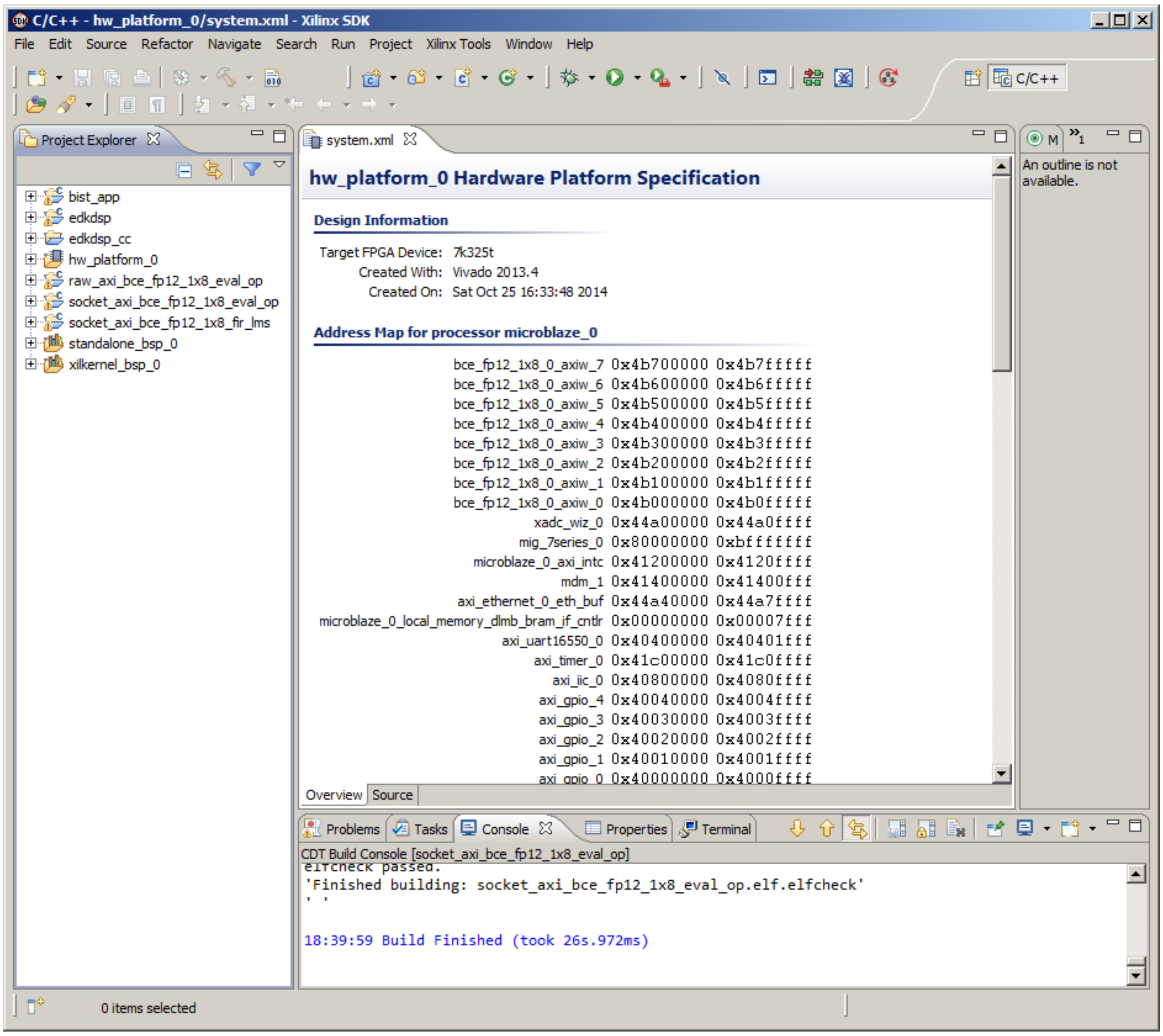
Figure 11: Select copy projects into workspace and finish the import of all projects.

All the UTIA EdkDSP SW projects are imported into SDK workspace from the directory
c:\VM_07\d_34_7k\d_7k325t_fp12_6x8_IMPORT

Process of compilation will start automatically. This first compilation of all SDK SW projects can take several minutes to finish. It should finish without errors. See Figure 12.

3.2 Evaluation of demo projects

The “bist_app” project in the “Project Explorer” window of the SDK 2013.4 is only slightly modified version of the Xilinx BIST SW application project. The RAM memory test is adjusted for the 128 KB RAM. See Figure 12.
The “edkdsp” project is extending the “bist_app” with tests of the EdkDSP accelerator, without Ethernet.
The “raw_axi_bce_fp12_1x8_eval_op” project is extending the “edkdsp” with RAW version of the lwIP Ethernet www server GUI, the TFTP file server and the RAM based file system.
The “socket_axi_bce_fp12_1x8_eval_op” project is extending the “edkdsp” with SOCKET version of the lwIP Ethernet www server GUI, the TFTP file server and the RAM based file system.
The “socket_axi_bce_fp12_1x8_fir_lms” project is demonstrating the floating point FIR filter and LMS filter computation on a single (8xSIMD) EdkDSP accelerator with the SOCKET version of the lwIP Ethernet www server GUI, the SOCKET version of the TFTP file server and the RAM based file system.



- bist_app
- edkdsp
- edkdsp_cc
- hw_platform_0
- raw_axi_bce_fp12_1x8_eval_op
- socket_axi_bce_fp12_1x8_eval_op
- socket_axi_bce_fp12_1x8_fir_lms
- standalone_bsp_0
- xilkernel_bsp_0

hw_platform_0 Hardware Platform Specification

Design Information

Target FPGA Device:	7k325t
Created With:	Vivado 2013.4
Created On:	Sat Oct 25 16:33:48 2014

Address Map for processor microblaze_0

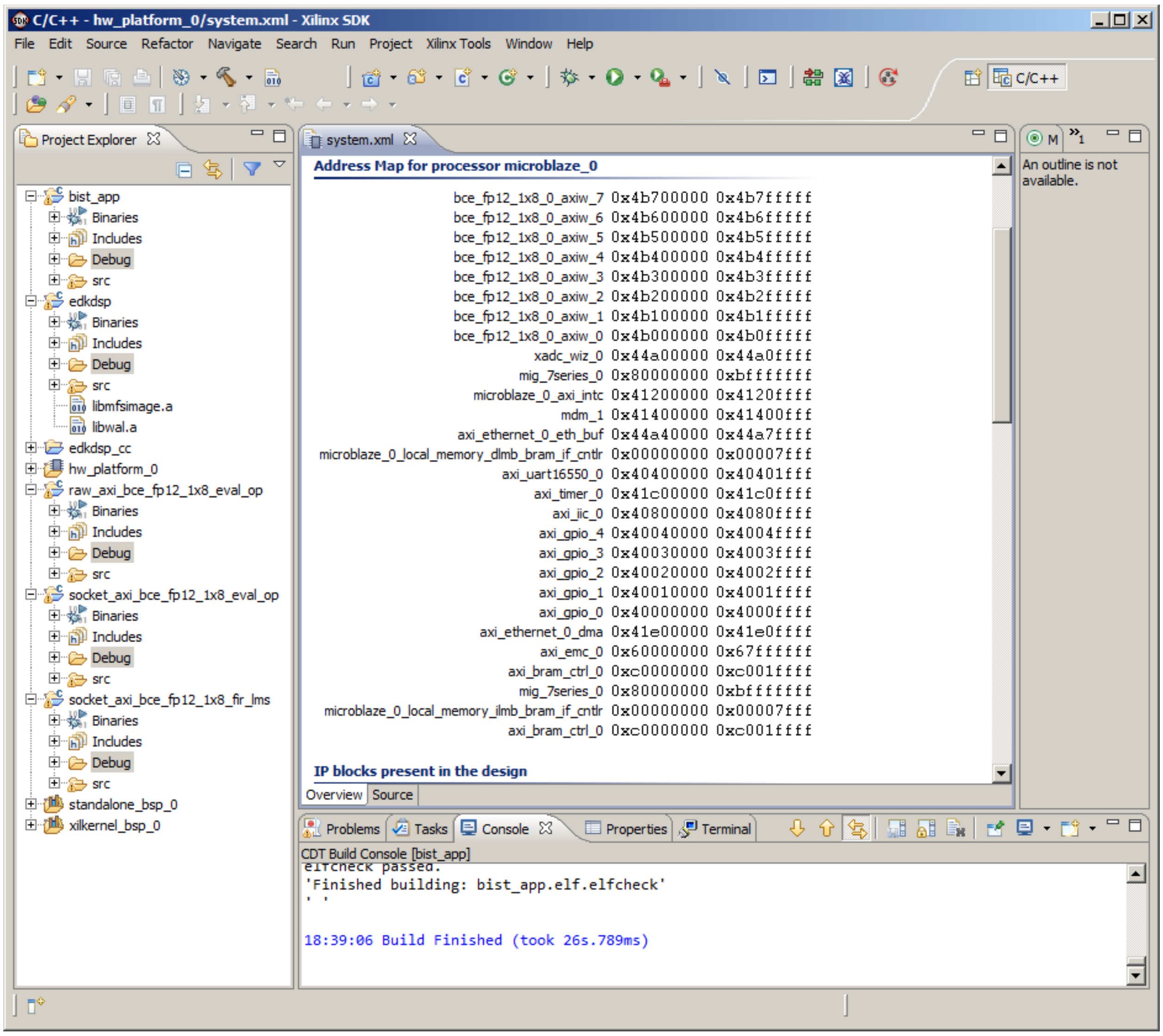
bce_fp12_1x8_0_axiw_7	0x4b700000	0x4b7fffff
bce_fp12_1x8_0_axiw_6	0x4b600000	0x4b6fffff
bce_fp12_1x8_0_axiw_5	0x4b500000	0x4b5fffff
bce_fp12_1x8_0_axiw_4	0x4b400000	0x4b4fffff
bce_fp12_1x8_0_axiw_3	0x4b300000	0x4b3fffff
bce_fp12_1x8_0_axiw_2	0x4b200000	0x4b2fffff
bce_fp12_1x8_0_axiw_1	0x4b100000	0x4b1fffff
bce_fp12_1x8_0_axiw_0	0x4b000000	0x4b0fffff
xadc_wiz_0	0x44a00000	0x44a0ffff
mig_7series_0	0x80000000	0xbfffffff
microblaze_0_axi_intc	0x41200000	0x4120ffff
mdm_1	0x41400000	0x41400fff
axi_ethernet_0_eth_buf	0x44a40000	0x44a7ffff
microblaze_0_local_memory_dlmb_bram_if_cntlr	0x00000000	0x00007fff
axi_uart16550_0	0x40400000	0x40401fff
axi_timer_0	0x41c00000	0x41c0ffff
axi_iic_0	0x40800000	0x4080ffff
axi_gpio_4	0x40040000	0x4004ffff
axi_gpio_3	0x40030000	0x4003ffff
axi_gpio_2	0x40020000	0x4002ffff
axi_gpio_1	0x40010000	0x4001ffff
axi_gpio_0	0x40000000	0x4000ffff

CDT Build Console [socket_axi_bce_fp12_1x8_eval_op]

```
elfcheck passed.
'Finished building: socket_axi_bce_fp12_1x8_eval_op.elf.elfcheck'
' '

18:39:59 Build Finished (took 26s.972ms)
```

Figure 12: All projects are compiled. See IP blocks present in the design.

Connect the jtag and serial line USB cables to your KC705 board. Switch ON the board.



Address Map for processor microblaze_0

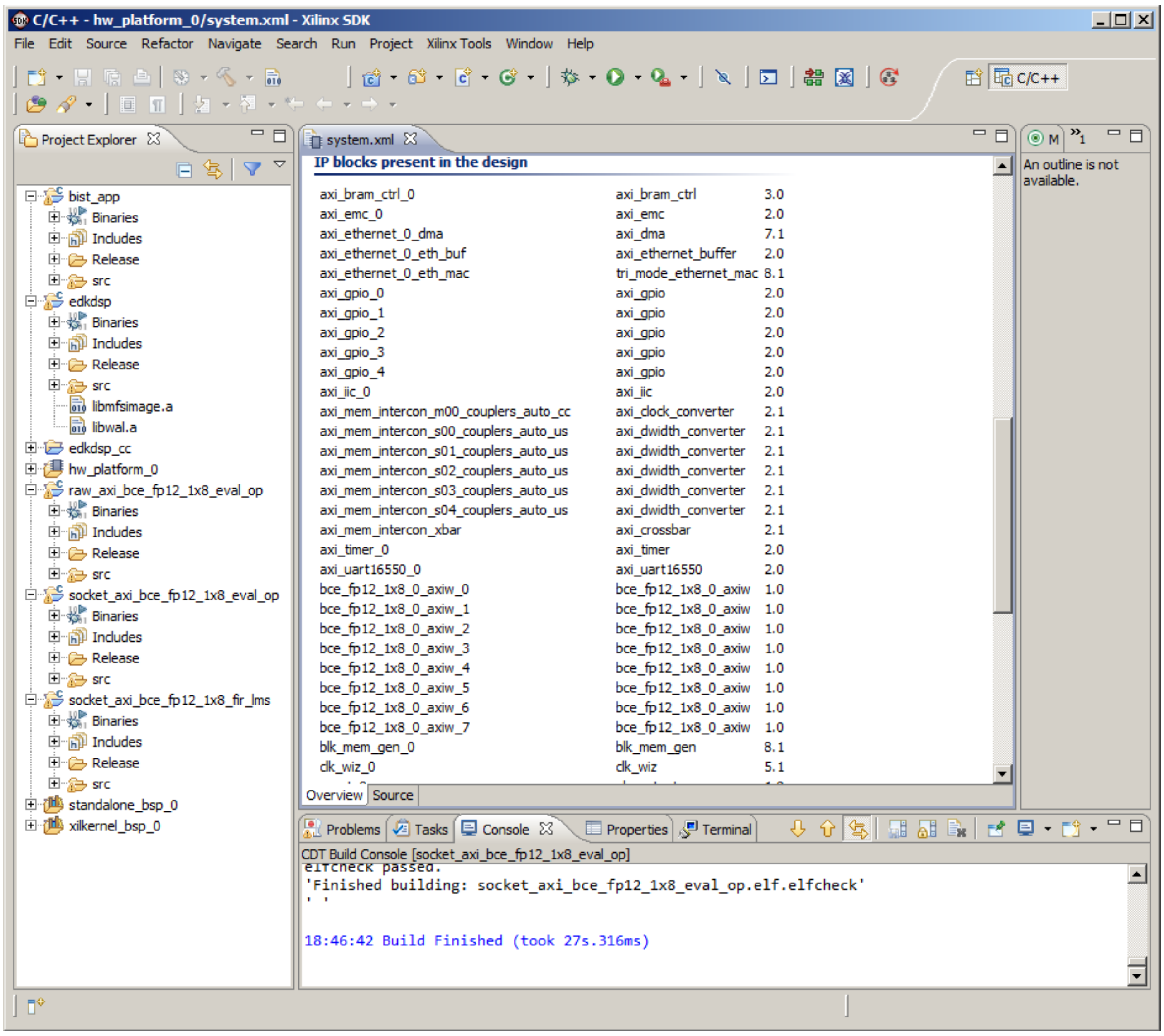
```
                    bce_fp12_1x8_0_axiw_7 0x4b700000 0x4b7fffff
                    bce_fp12_1x8_0_axiw_6 0x4b600000 0x4b6fffff
                    bce_fp12_1x8_0_axiw_5 0x4b500000 0x4b5fffff
                    bce_fp12_1x8_0_axiw_4 0x4b400000 0x4b4fffff
                    bce_fp12_1x8_0_axiw_3 0x4b300000 0x4b3fffff
                    bce_fp12_1x8_0_axiw_2 0x4b200000 0x4b2fffff
                    bce_fp12_1x8_0_axiw_1 0x4b100000 0x4b1fffff
                    bce_fp12_1x8_0_axiw_0 0x4b000000 0x4b0fffff
                               xadc_wiz_0 0x44a00000 0x44a0ffff
                              mig_7series_0 0x80000000 0xbfffffff
                     microblaze_0_axi_intc 0x41200000 0x4120ffff
                                    mdm_1 0x41400000 0x41400fff
                       axi_ethernet_0_eth_buf 0x44a40000 0x44a7ffff
microblaze_0_local_memory_dlmb_bram_if_cntlr 0x00000000 0x00007fff
                           axi_uart16550_0 0x40400000 0x40401fff
                              axi_timer_0 0x41c00000 0x41c0ffff
                                axi_iic_0 0x40800000 0x4080ffff
                               axi_gpio_4 0x40040000 0x4004ffff
                               axi_gpio_3 0x40030000 0x4003ffff
                               axi_gpio_2 0x40020000 0x4002ffff
                               axi_gpio_1 0x40010000 0x4001ffff
                               axi_gpio_0 0x40000000 0x4000ffff
                         axi_ethernet_0_dma 0x41e00000 0x41e0ffff
                                axi_emc_0 0x60000000 0x67ffffff
                            axi_bram_ctrl_0 0xc0000000 0xc001ffff
                              mig_7series_0 0x80000000 0xbfffffff
microblaze_0_local_memory_ilmb_bram_if_cntlr 0x00000000 0x00007fff
                            axi_bram_ctrl_0 0xc0000000 0xc001ffff
```

IP blocks present in the design

CDT Build Console [bist_app]

```
elfcheck passed.
'Finished building: bist_app.elf.elfcheck'
' '

18:39:06 Build Finished (took 26s.789ms)
```

Figure 13: Set all projects for Release and delete all Debug subdirectories


C/C++ - hw_platform_0/system.xml - Xilinx SDK
File Edit Source Refactor Navigate Search Run Project Xilinx Tools Window Help
C/C++
Project Explorer
system.xml
An outline is not available.
Overview Source
Problems Tasks Console Properties Terminal

```
bist_app
    Binaries
    Includes
    Release
    src
edkdsp
    Binaries
    Includes
    Release
    src
    libmfsimage.a
    libwal.a
edkdsp_cc
hw_platform_0
raw_axi_bce_fp12_1x8_eval_op
    Binaries
    Includes
    Release
    src
socket_axi_bce_fp12_1x8_eval_op
    Binaries
    Includes
    Release
    src
socket_axi_bce_fp12_1x8_fir_lms
    Binaries
    Includes
    Release
    src
standalone_bsp_0
xilkernel_bsp_0
```

IP blocks present in the design

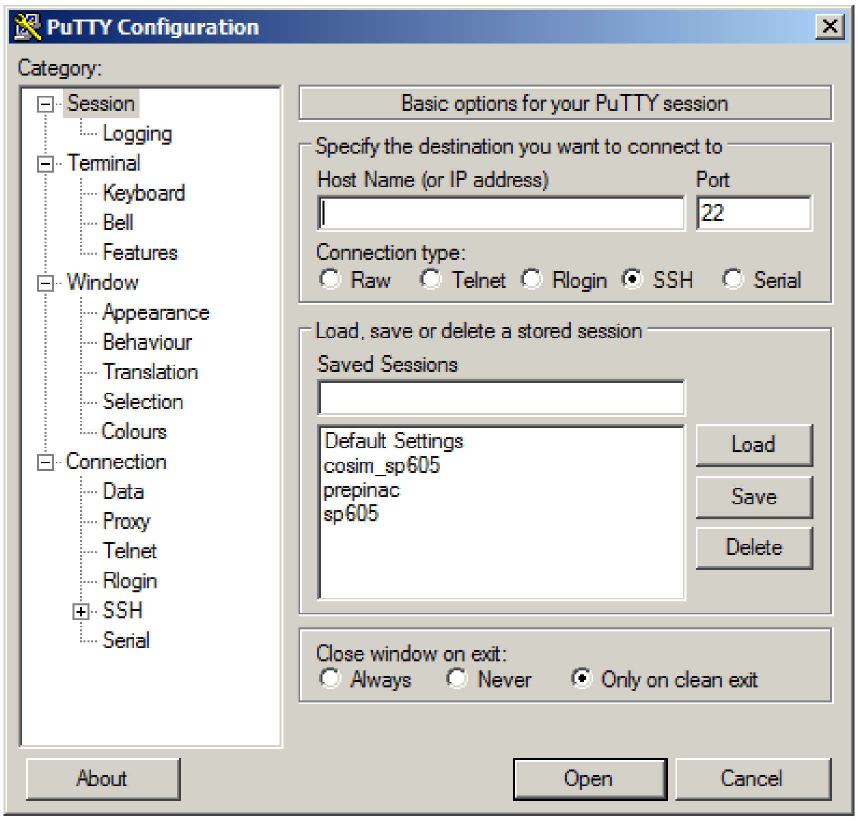
axi_bram_ctrl_0	axi_bram_ctrl	3.0
axi_emc_0	axi_emc	2.0
axi_ethernet_0_dma	axi_dma	7.1
axi_ethernet_0_eth_buf	axi_ethernet_buffer	2.0
axi_ethernet_0_eth_mac	tri_mode_ethernet_mac	8.1
axi_gpio_0	axi_gpio	2.0
axi_gpio_1	axi_gpio	2.0
axi_gpio_2	axi_gpio	2.0
axi_gpio_3	axi_gpio	2.0
axi_gpio_4	axi_gpio	2.0
axi_iic_0	axi_iic	2.0
axi_mem_intercon_m00_couplers_auto_cc	axi_clock_converter	2.1
axi_mem_intercon_s00_couplers_auto_us	axi_dwidth_converter	2.1
axi_mem_intercon_s01_couplers_auto_us	axi_dwidth_converter	2.1
axi_mem_intercon_s02_couplers_auto_us	axi_dwidth_converter	2.1
axi_mem_intercon_s03_couplers_auto_us	axi_dwidth_converter	2.1
axi_mem_intercon_s04_couplers_auto_us	axi_dwidth_converter	2.1
axi_mem_intercon_xbar	axi_crossbar	2.1
axi_timer_0	axi_timer	2.0
axi_uart16550_0	axi_uart16550	2.0
bce_fp12_1x8_0_axiw_0	bce_fp12_1x8_0_axiw	1.0
bce_fp12_1x8_0_axiw_1	bce_fp12_1x8_0_axiw	1.0
bce_fp12_1x8_0_axiw_2	bce_fp12_1x8_0_axiw	1.0
bce_fp12_1x8_0_axiw_3	bce_fp12_1x8_0_axiw	1.0
bce_fp12_1x8_0_axiw_4	bce_fp12_1x8_0_axiw	1.0
bce_fp12_1x8_0_axiw_5	bce_fp12_1x8_0_axiw	1.0
bce_fp12_1x8_0_axiw_6	bce_fp12_1x8_0_axiw	1.0
bce_fp12_1x8_0_axiw_7	bce_fp12_1x8_0_axiw	1.0
blk_mem_gen_0	blk_mem_gen	8.1
clk_wiz_0	clk_wiz	5.1

CDT Build Console [socket_axi_bce_fp12_1x8_eval_op]

```
elfcheck passed.
'Finished building: socket_axi_bce_fp12_1x8_eval_op.elf.elfcheck'
' '

18:46:42 Build Finished (took 27s.316ms)
```

Figure 14: All projects are recompiled for release.

On PC, start PuTTY terminal. Set 9600 baud and “Flow control” to None. See Figure 15 and Figure 16.



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

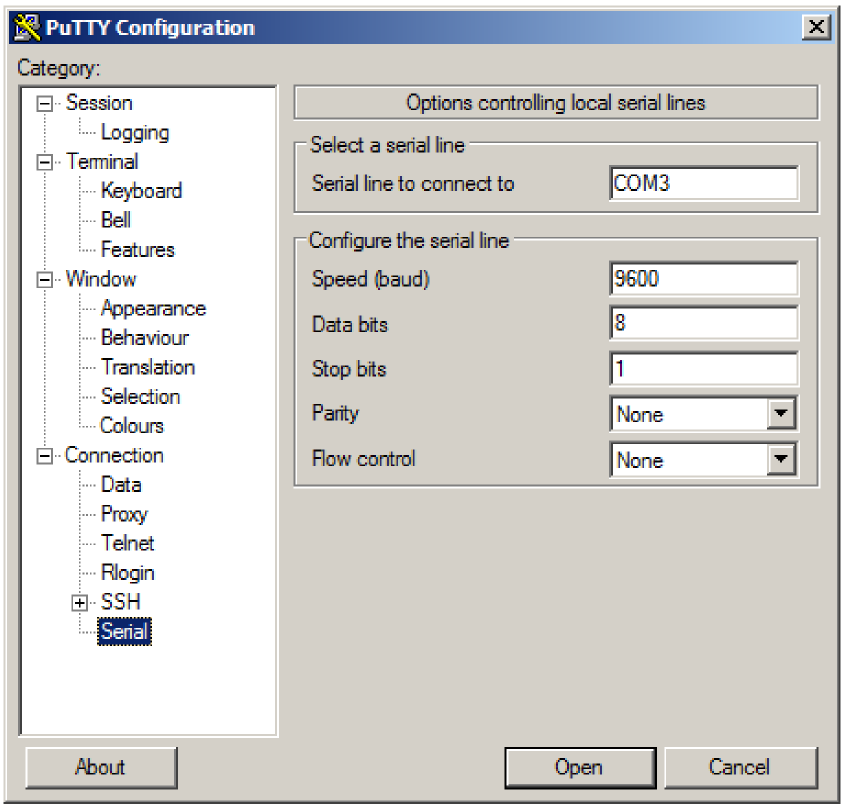
Load, save or delete a stored session

Saved Sessions

Default Settings
cosim_sp605
prepinac
sp605

Close window on exit: Always Never [✓ Only on clean exit]

Figure 15: Open PuTTY terminal.



Options controlling local serial lines

Select a serial line

Serial line to connect to	COM3

Configure the serial line

Speed (baud)	9600
Data bits	8
Stop bits	1
Parity	None
Flow control	None

Figure 16: Select “Serial”, select your COL port, set speed to 9600 and flow control to None.

3.3 Ethernet point to point connection with PC

The SDK SW projects included in this evaluation package demonstrate integration of the UTIA EdkDSP accelerator together with the Xilinx 1 Gb Ethernet controller. The connection to the Ethernet is based on two versions of the LwIP SW:

- Raw versions of SDK SW projects use raw version of the LwIP library without real-time OS.
- Socket versions of SW projects use the socket version of LwIP on top of the Xilinx XilKernel.

Set your PC Ethernet connection to point-to-point with the fixed IP address:

192.168.8.2

All included UTIA EdkDSP projects are setting the IP address of the KC705 board to:

192.168.8.10

This setting enables the direct point to point Ethernet connection.

3.4 Boot of the bitstream

Program the KC705 board by selecting in SDK:
Xilinx Tools -> Program FPGA

C:\VM_07\d_34_7k\d_7k325t_fp12_1x8\SDK_Workspace\hw_platform\system.xml

Click on the "Program" button. See Figure 17.

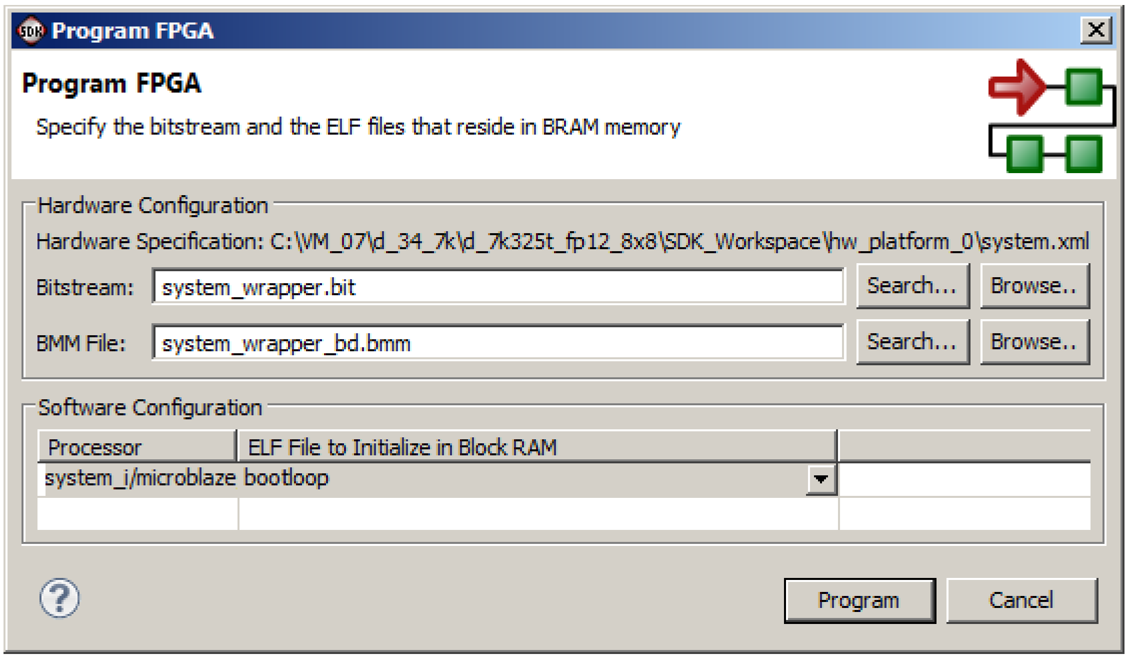
The KC705 board is programmed with the system_wrapper.bit. The MicroBlaze is running in the initial bootloop from internal FPGA RAM.

3.5 Boot of the application

The SW bist_app.elf application from the "bist_app" project can be downloaded to the DDR3 memory and started. Select the "bist_app" project in the project navigator.

In SDK, select:
Run -> Run Configuration -> Xilinx C/C++ ELF

Click on the "New launch configuration" in the Run configuration screen and the bist_app.elf project executable is ready for download to DDR3 via the jtag cable. Click on "Run" button to download the executable. See Figure 18.



Program FPGA

Specify the bitstream and the ELF files that reside in BRAM memory

Hardware Configuration

Hardware Specification: C:\VM_07\d_34_7k\d_7k325t_fp12_8x8\SDK_Workspace\hw_platform_0\system.xml

Bitstream: system_wrapper.bit

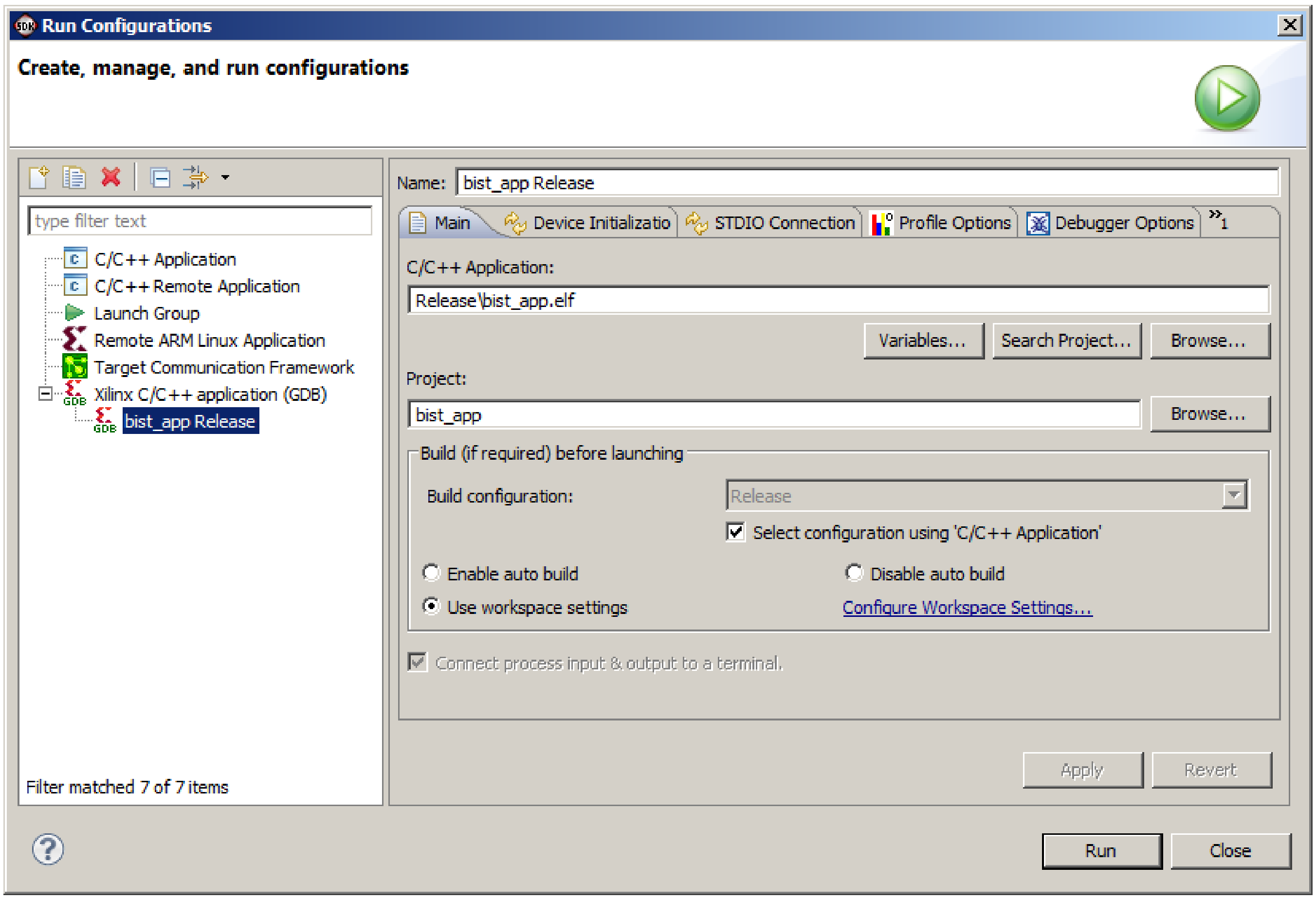
BMM File: system_wrapper_bd.bmm

Software Configuration

Processor	ELF File to Initialize in Block RAM
system_i/microblaze	bootloop

Figure 17: Program KC705 board.

Click on the “Program” button.



Create, manage, and run configurations

type filter text

Filter matched 7 of 7 items

Name: bist_app Release

C/C++ Application: Release\bist_app.elf

Project: bist_app

Build (if required) before launching

Build configuration: Release

[✓] Select configuration using 'C/C++ Application'

Enable auto build / Disable auto build

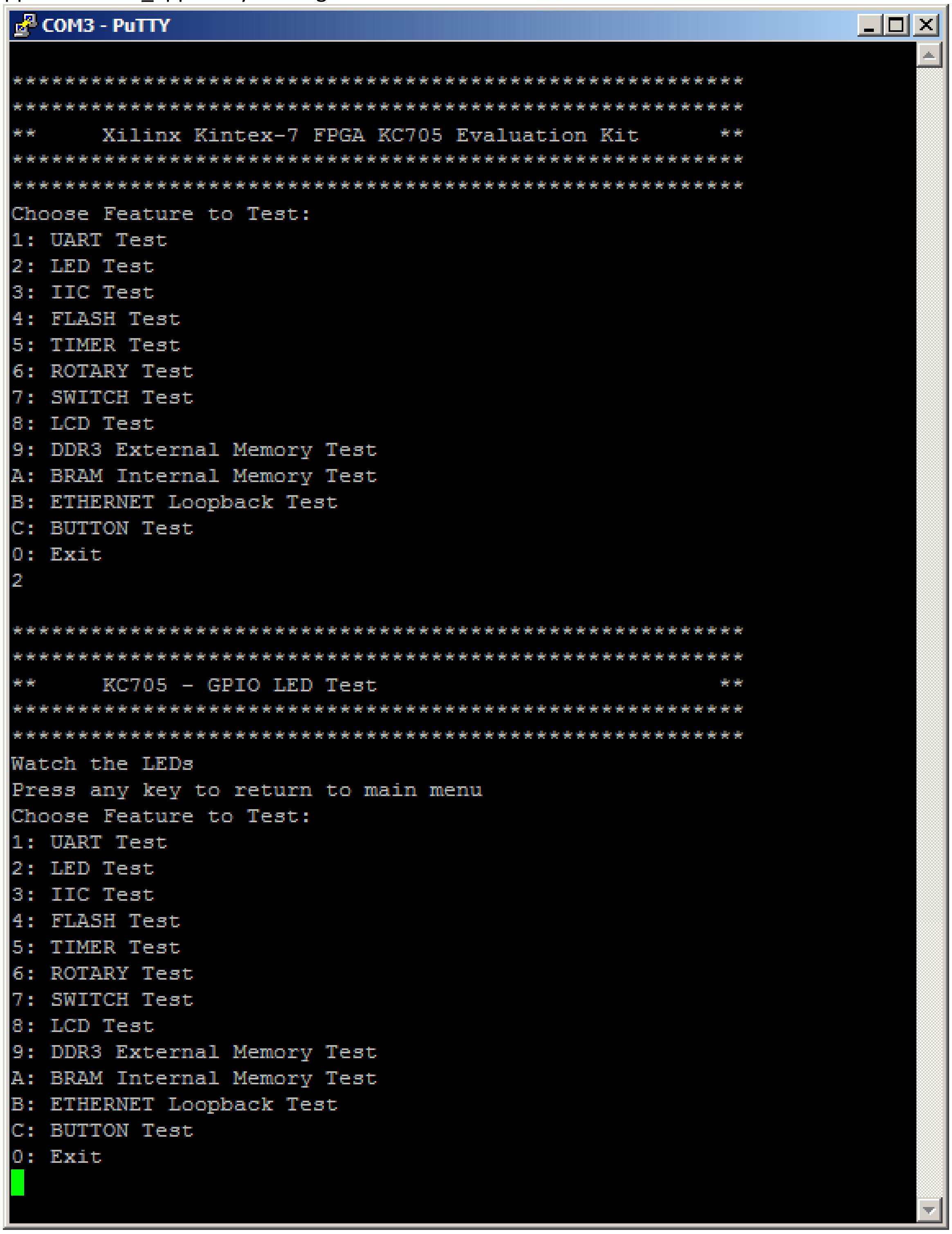
[✓ Use workspace settings]

Configure Workspace Settings...

[✓] Connect process input & output to a terminal.

Figure 18: Select “bist_app.elf” code.

Run the application bist_app.elf by clicking on Run.

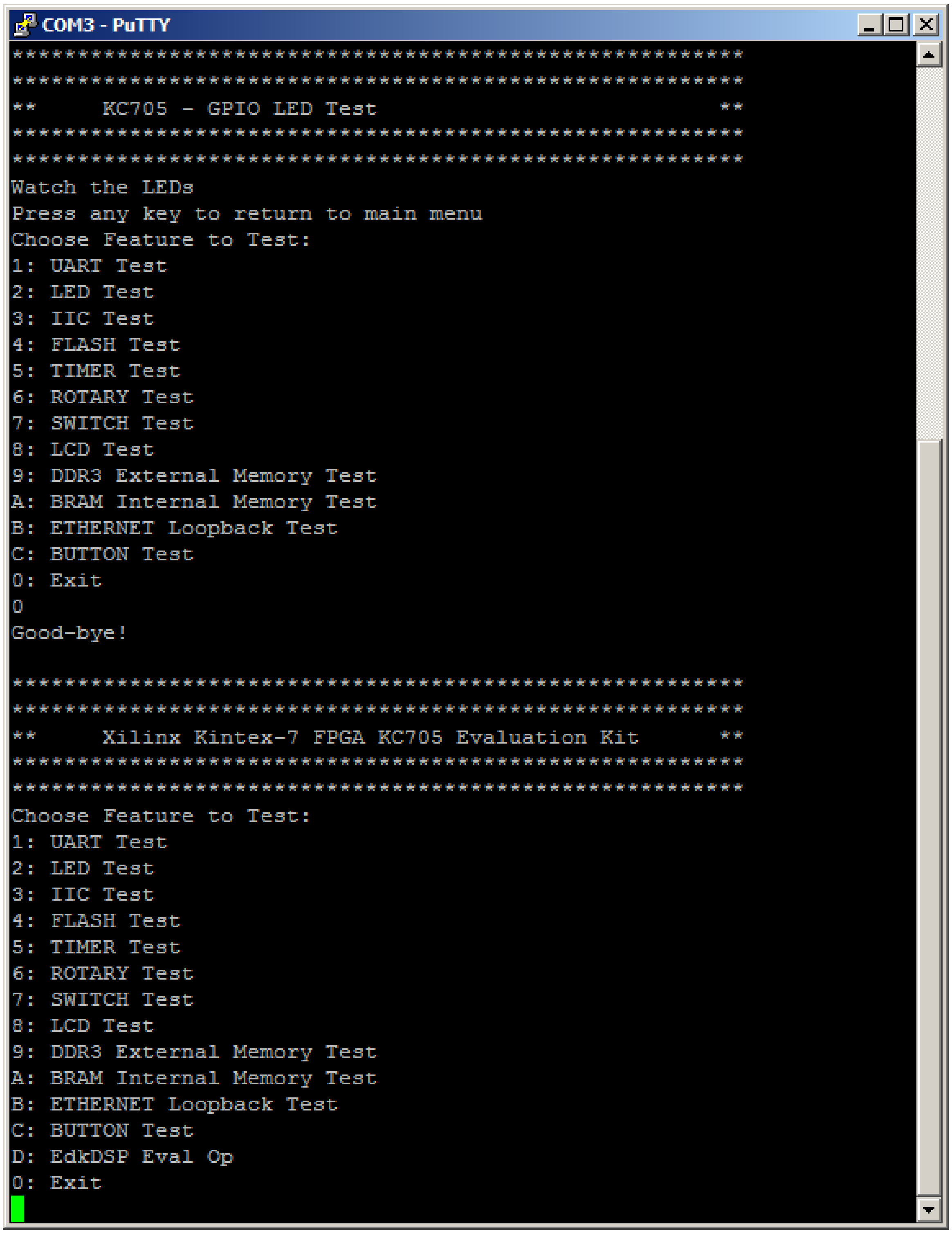


```
**********************************************************
**********************************************************
**      Xilinx Kintex-7 FPGA KC705 Evaluation Kit      **
**********************************************************
**********************************************************
Choose Feature to Test:
1: UART Test
2: LED Test
3: IIC Test
4: FLASH Test
5: TIMER Test
6: ROTARY Test
7: SWITCH Test
8: LCD Test
9: DDR3 External Memory Test
A: BRAM Internal Memory Test
B: ETHERNET Loopback Test
C: BUTTON Test
0: Exit
2

**********************************************************
**********************************************************
**      KC705 - GPIO LED Test                          **
**********************************************************
**********************************************************
Watch the LEDs
Press any key to return to main menu
Choose Feature to Test:
1: UART Test
2: LED Test
3: IIC Test
4: FLASH Test
5: TIMER Test
6: ROTARY Test
7: SWITCH Test
8: LCD Test
9: DDR3 External Memory Test
A: BRAM Internal Memory Test
B: ETHERNET Loopback Test
C: BUTTON Test
0: Exit
```

Figure 19: Run bist_app.elf and select tests from the terminal keyboard (PC).

The Xilinx **bist_app** demo serves for test of the MicroBlaze peripherals. Stop hardware from SDK.

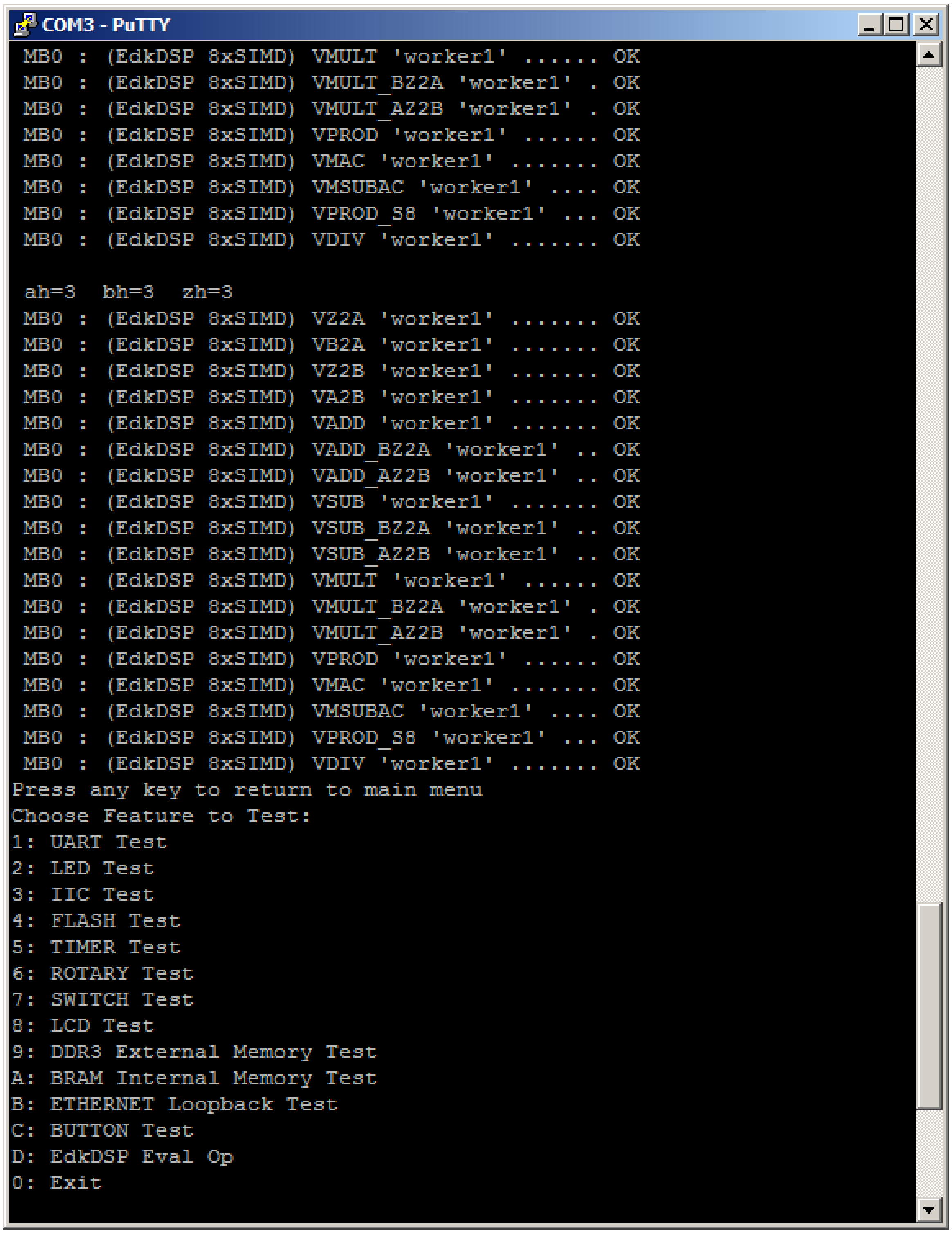
Download again the bitstream (chapter 3.4), select the **edkdsp** project for download (chapter 3.5), run it to see the extended menu enabling tests of the EdkDSP accelerator. See Figure 20.



```
************************************************************
************************************************************
**      KC705 - GPIO LED Test                             **
************************************************************
************************************************************
Watch the LEDs
Press any key to return to main menu
Choose Feature to Test:
1: UART Test
2: LED Test
3: IIC Test
4: FLASH Test
5: TIMER Test
6: ROTARY Test
7: SWITCH Test
8: LCD Test
9: DDR3 External Memory Test
A: BRAM Internal Memory Test
B: ETHERNET Loopback Test
C: BUTTON Test
0: Exit
0
Good-bye!

************************************************************
************************************************************
**      Xilinx Kintex-7 FPGA KC705 Evaluation Kit         **
************************************************************
************************************************************
Choose Feature to Test:
1: UART Test
2: LED Test
3: IIC Test
4: FLASH Test
5: TIMER Test
6: ROTARY Test
7: SWITCH Test
8: LCD Test
9: DDR3 External Memory Test
A: BRAM Internal Memory Test
B: ETHERNET Loopback Test
C: BUTTON Test
D: EdkDSP Eval Op
0: Exit
```

Figure 20: Run the edkdsp.elf application and select the EdkDSP Eval Op test.

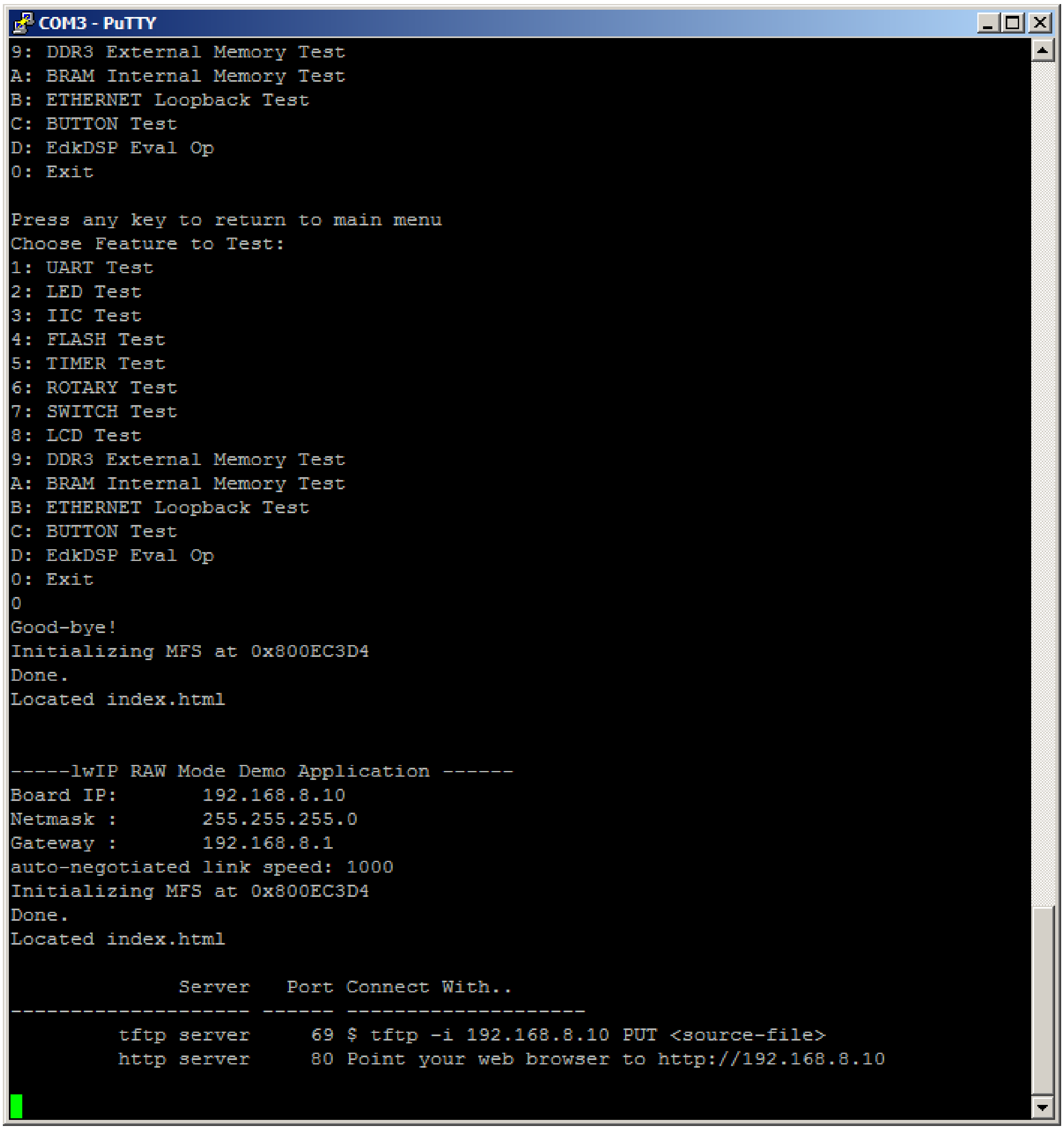
Select the C option from the terminal keyboard to run test of the EdkDSP accelerator. See Figure 21.



```
 MB0 : (EdkDSP 8xSIMD) VMULT 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMULT_BZ2A 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VMULT_AZ2B 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VPROD 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMAC 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VMSUBAC 'worker1' .... OK
 MB0 : (EdkDSP 8xSIMD) VPROD_S8 'worker1' ... OK
 MB0 : (EdkDSP 8xSIMD) VDIV 'worker1' ....... OK

 ah=3  bh=3  zh=3
 MB0 : (EdkDSP 8xSIMD) VZ2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VB2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VZ2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VA2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VADD_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VSUB_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VMULT 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMULT_BZ2A 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VMULT_AZ2B 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VPROD 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMAC 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VMSUBAC 'worker1' .... OK
 MB0 : (EdkDSP 8xSIMD) VPROD_S8 'worker1' ... OK
 MB0 : (EdkDSP 8xSIMD) VDIV 'worker1' ....... OK
Press any key to return to main menu
Choose Feature to Test:
1: UART Test
2: LED Test
3: IIC Test
4: FLASH Test
5: TIMER Test
6: ROTARY Test
7: SWITCH Test
8: LCD Test
9: DDR3 External Memory Test
A: BRAM Internal Memory Test
B: ETHERNET Loopback Test
C: BUTTON Test
D: EdkDSP Eval Op
0: Exit
```

Figure 21: The EdkDSP basic vector floating point operations have been tested.

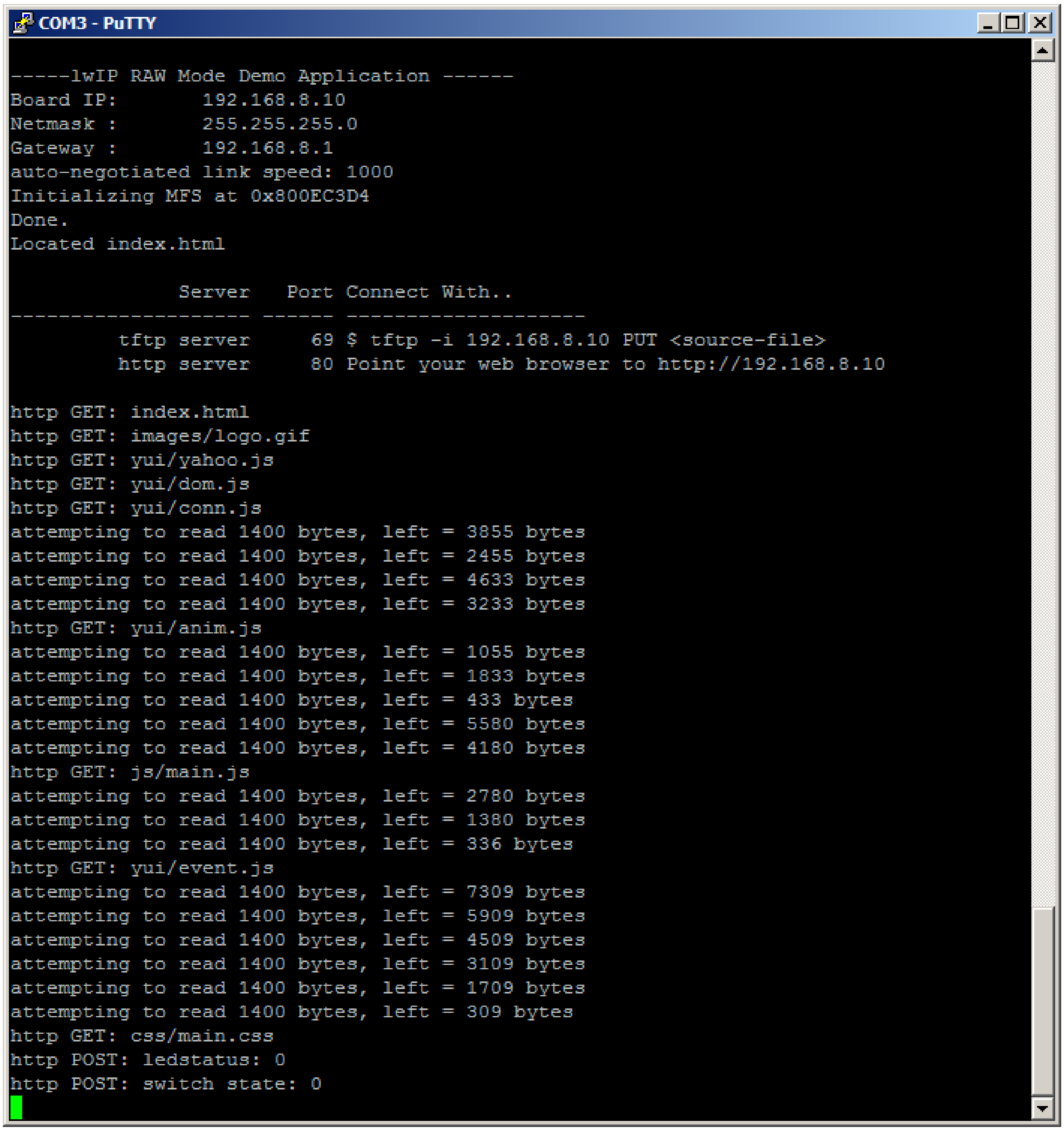
Stop hardware from the SDK. Download again the bitstream (chapter 3.4), select the **raw_axi_bce_fp12_1x8_eval_op** project for download (chapter 3.5) and run it. See Figure 22.



```
9: DDR3 External Memory Test
A: BRAM Internal Memory Test
B: ETHERNET Loopback Test
C: BUTTON Test
D: EdkDSP Eval Op
0: Exit

Press any key to return to main menu
Choose Feature to Test:
1: UART Test
2: LED Test
3: IIC Test
4: FLASH Test
5: TIMER Test
6: ROTARY Test
7: SWITCH Test
8: LCD Test
9: DDR3 External Memory Test
A: BRAM Internal Memory Test
B: ETHERNET Loopback Test
C: BUTTON Test
D: EdkDSP Eval Op
0: Exit
0
Good-bye!
Initializing MFS at 0x800EC3D4
Done.
Located index.html


-----lwIP RAW Mode Demo Application ------
Board IP:       192.168.8.10
Netmask :       255.255.255.0
Gateway :       192.168.8.1
auto-negotiated link speed: 1000
Initializing MFS at 0x800EC3D4
Done.
Located index.html

              Server   Port Connect With..
-------------------- ------ --------------------
         tftp server     69 $ tftp -i 192.168.8.10 PUT <source-file>
         http server     80 Point your web browser to http://192.168.8.10
```

Figure 22: Select "raw_axi_bce_fp12_eval_opl.elf application to test the lwIP services in RWW mode.

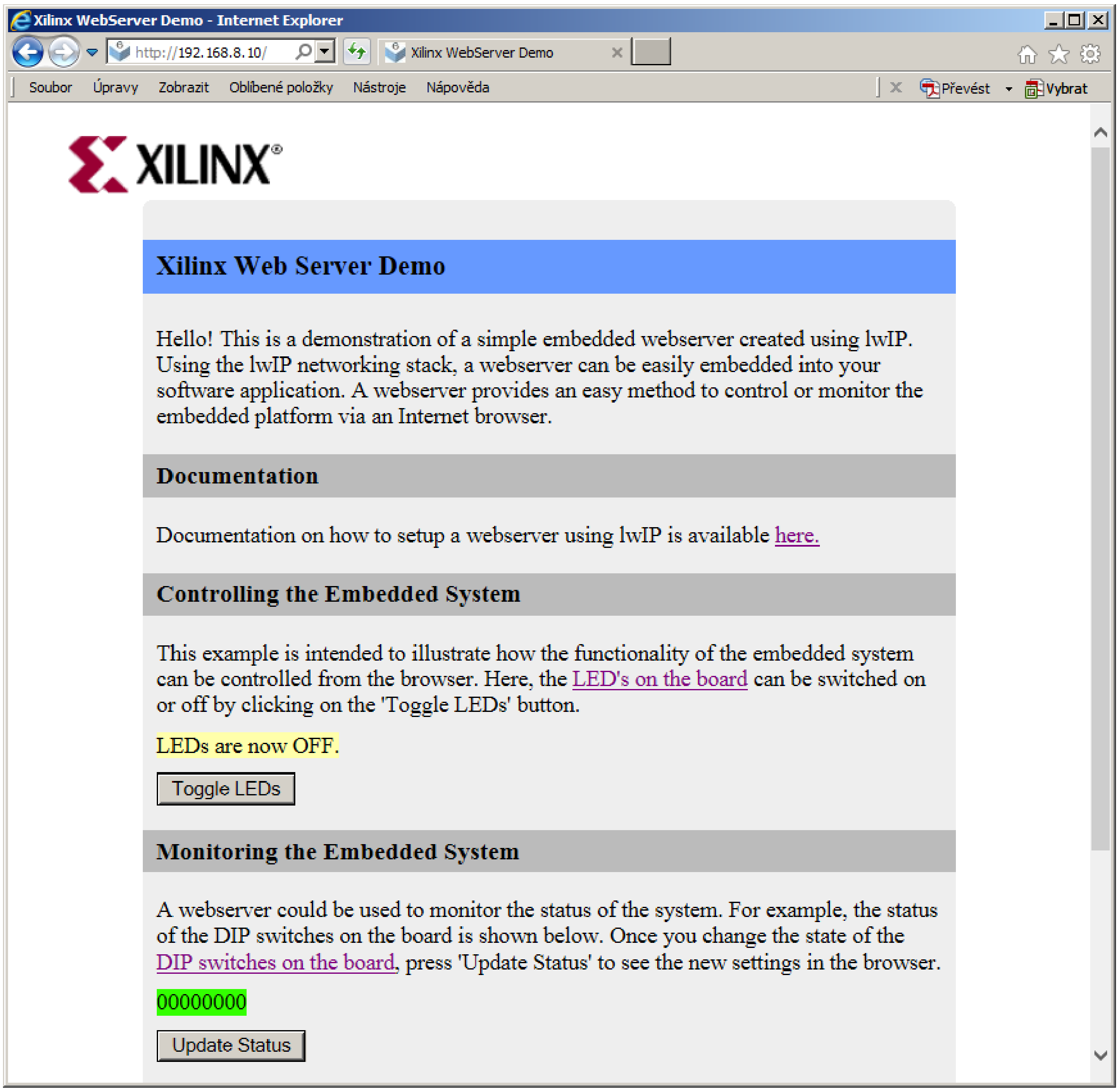
The RAW version of the tftp server and the RAW version of the http server have been started on the Kintex7 MicroBlaze processor. Open www browser in (Internet Explorer) client and connect to the board address http://192.168.8.10/



```
-----lwIP RAW Mode Demo Application ------
Board IP:       192.168.8.10
Netmask :       255.255.255.0
Gateway :       192.168.8.1
auto-negotiated link speed: 1000
Initializing MFS at 0x800EC3D4
Done.
Located index.html

              Server   Port Connect With..
-------------------- ------ --------------------
         tftp server     69 $ tftp -i 192.168.8.10 PUT <source-file>
         http server     80 Point your web browser to http://192.168.8.10

http GET: index.html
http GET: images/logo.gif
http GET: yui/yahoo.js
http GET: yui/dom.js
http GET: yui/conn.js
attempting to read 1400 bytes, left = 3855 bytes
attempting to read 1400 bytes, left = 2455 bytes
attempting to read 1400 bytes, left = 4633 bytes
attempting to read 1400 bytes, left = 3233 bytes
http GET: yui/anim.js
attempting to read 1400 bytes, left = 1055 bytes
attempting to read 1400 bytes, left = 1833 bytes
attempting to read 1400 bytes, left = 433 bytes
attempting to read 1400 bytes, left = 5580 bytes
attempting to read 1400 bytes, left = 4180 bytes
http GET: js/main.js
attempting to read 1400 bytes, left = 2780 bytes
attempting to read 1400 bytes, left = 1380 bytes
attempting to read 1400 bytes, left = 336 bytes
http GET: yui/event.js
attempting to read 1400 bytes, left = 7309 bytes
attempting to read 1400 bytes, left = 5909 bytes
attempting to read 1400 bytes, left = 4509 bytes
attempting to read 1400 bytes, left = 3109 bytes
attempting to read 1400 bytes, left = 1709 bytes
attempting to read 1400 bytes, left = 309 bytes
http GET: css/main.css
http POST: ledstatus: 0
http POST: switch state: 0
```

Figure 23: The Java Script has been loaded from the FPGA RAM based file system to your brawser.

Support script files are downloaded to the PC from the Kintex7 file system and the interface page is started. See Figure 23 and Figure 24.



Xilinx Web Server Demo

Hello! This is a demonstration of a simple embedded webserver created using lwIP. Using the lwIP networking stack, a webserver can be easily embedded into your software application. A webserver provides an easy method to control or monitor the embedded platform via an Internet browser.

Documentation

Documentation on how to setup a webserver using lwIP is available here.

Controlling the Embedded System

This example is intended to illustrate how the functionality of the embedded system can be controlled from the browser. Here, the LED's on the board can be switched on or off by clicking on the 'Toggle LEDs' button.

LEDs are now OFF.

Toggle LEDs

Monitoring the Embedded System

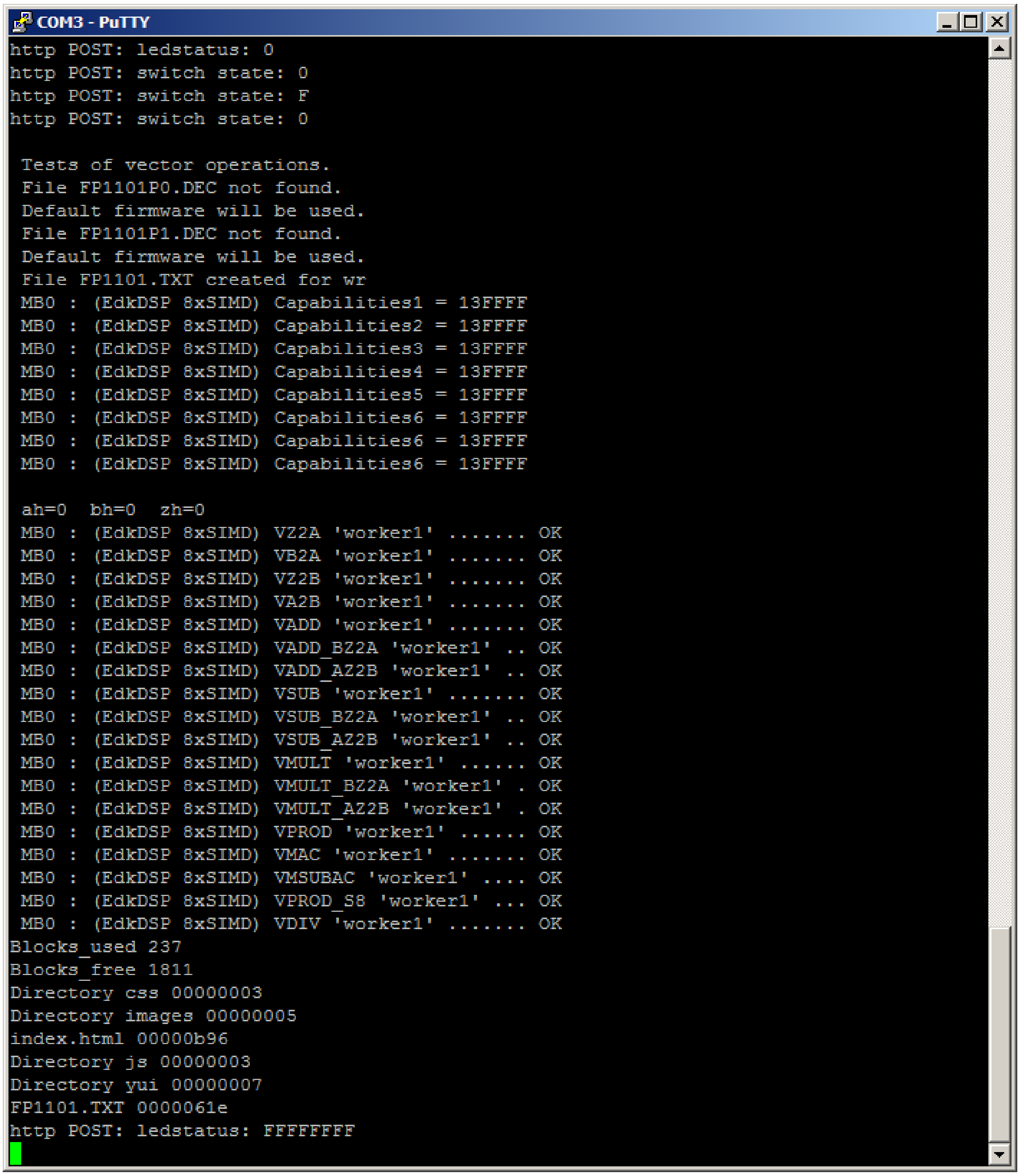
A webserver could be used to monitor the status of the system. For example, the status of the DIP switches on the board is shown below. Once you change the state of the DIP switches on the board, press 'Update Status' to see the new settings in the browser.

00000000

Update Status

Figure 24: The demo www server is evaluating the basic GUI for communication from the web browser client to the Kintex7 application working as an embedded server providing 1 G bit point to point connection.

The **Update Status** button serves to get the DIP switches status. The **Toggle LEDs** button is toggling the led output on the board and starts the EdkDSP accelerator evaluation. See Figure 25. The SW application is testing presence of an updated firmware in the RAM based file system of the board. If it is not present, the default firmware is used.
The file FP1101.TXT is open for WR in the RAM based file system. It will store text messages from the tested EdkDSP accelerator.
The capabilities of all 6 EdkDSP accelerators are displayed next. This information is based on the reply from the initialised accelerators. Test is performed. Finally the top directory of the RAM based file system is listed together with the information about used and free blocks in the RAM based file system. See Figure 25.

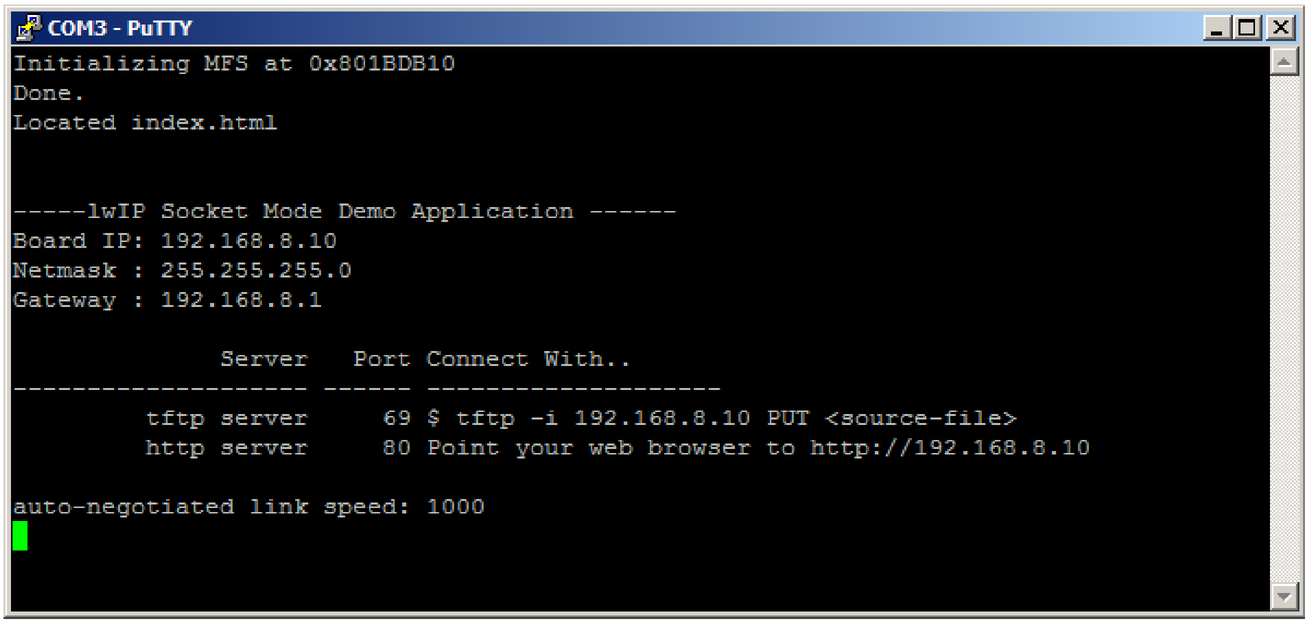


```
http POST: ledstatus: 0
http POST: switch state: 0
http POST: switch state: F
http POST: switch state: 0

 Tests of vector operations.
 File FP1101P0.DEC not found.
 Default firmware will be used.
 File FP1101P1.DEC not found.
 Default firmware will be used.
 File FP1101.TXT created for wr
 MB0 : (EdkDSP 8xSIMD) Capabilities1 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities2 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities3 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities4 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities5 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities6 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities6 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities6 = 13FFFF

 ah=0  bh=0  zh=0
 MB0 : (EdkDSP 8xSIMD) VZ2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VB2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VZ2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VA2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VADD_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VSUB_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VMULT 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMULT_BZ2A 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VMULT_AZ2B 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VPROD 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMAC 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VMSUBAC 'worker1' .... OK
 MB0 : (EdkDSP 8xSIMD) VPROD_S8 'worker1' ... OK
 MB0 : (EdkDSP 8xSIMD) VDIV 'worker1' ....... OK
Blocks_used 237
Blocks_free 1811
Directory css 00000003
Directory images 00000005
index.html 00000b96
Directory js 00000003
Directory yui 00000007
FP1101.TXT 0000061e
http POST: ledstatus: FFFFFFFF
```

Figure 25: Test of basic operations has been started from the web browser GUI TOGLE LED button. The listing of the top level directory of the RAM based file system is provided to the terminal.

Close the web browser. Close the application running on the Kintex7 from the SDK (click on the Red square icon on top of the console and next on the X icon to close the debugger session).

Download again the bitstream (chapter 3.4), select the **socket_axi_bce_fp12_1x8_eval_op** project for download (chapter 3.5) and run it. See Figure 26.

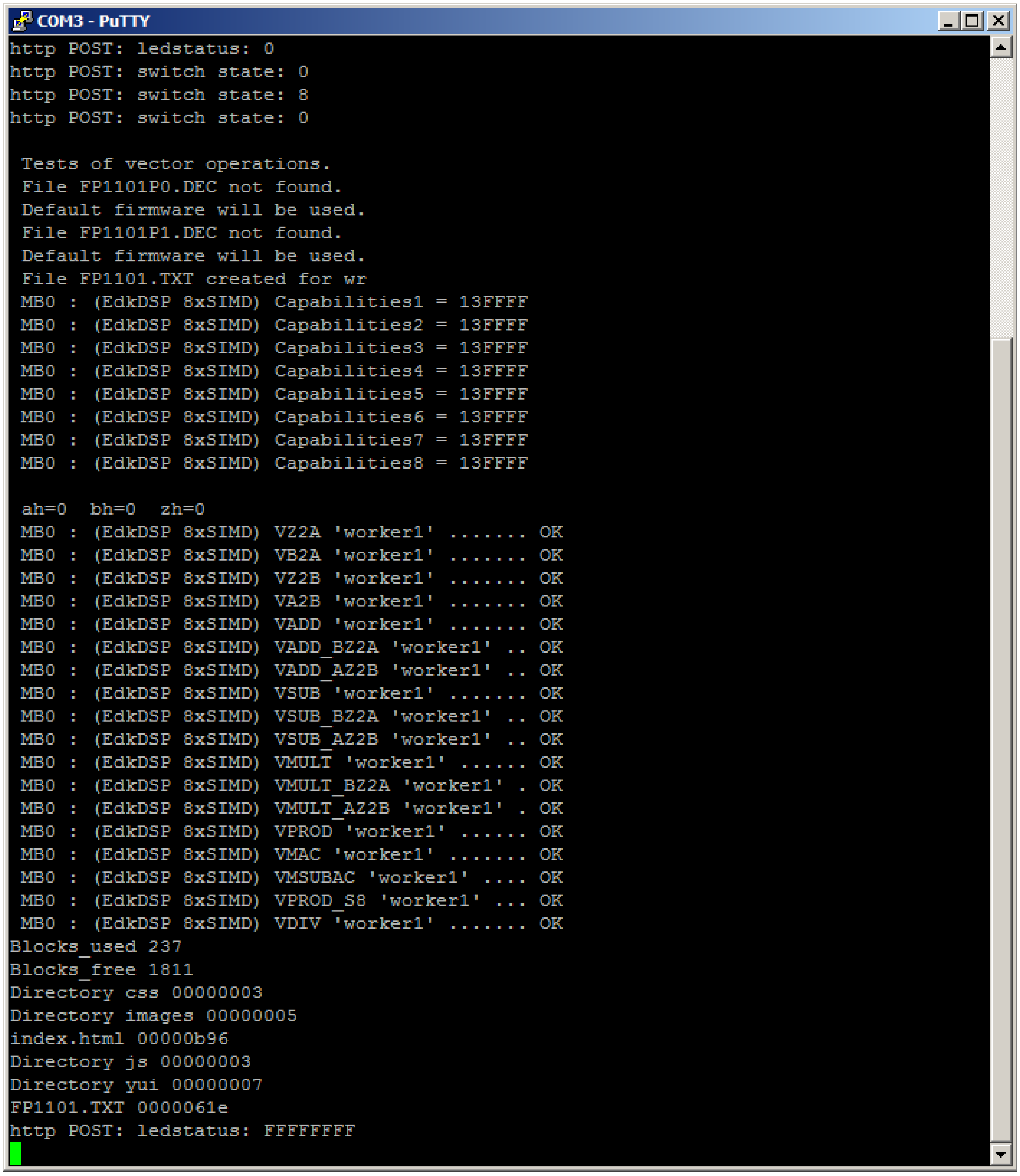


```
Initializing MFS at 0x801BDB10
Done.
Located index.html


-----lwIP Socket Mode Demo Application ------
Board IP: 192.168.8.10
Netmask : 255.255.255.0
Gateway : 192.168.8.1

              Server   Port Connect With..
-------------------- ------ --------------------
         tftp server     69 $ tftp -i 192.168.8.10 PUT <source-file>
         http server     80 Point your web browser to http://192.168.8.10

auto-negotiated link speed: 1000
```

Figure 26: Start the socket_axi_bce_fp12_1x8_eval_op.elf demo application, working on top of the Xilkernel OS..

The SOCKET version of the tftp server and the http server have been started on the Kintex7 MicroBlaze processor. Open www browser (Internet Explorer) client and connect to the board address: http://192.168.8.10/

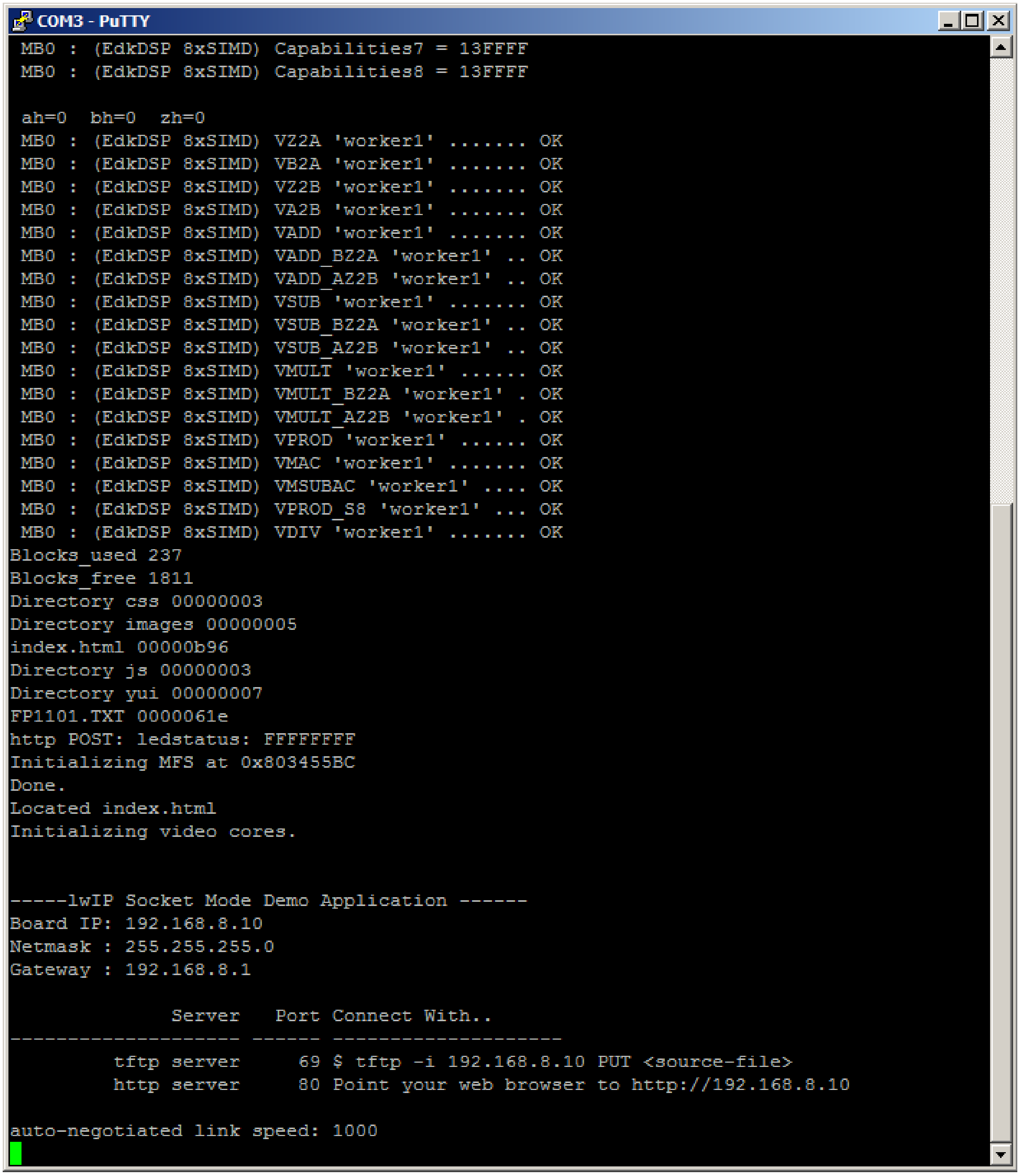
Click on the **Toggle LEDs** button to toggle the led output on the board and to starts the EdkDSP accelerator evaluation. The SOCKET version of the server supports both buttons in parallel. See Figure 27.



```
http POST: ledstatus: 0
http POST: switch state: 0
http POST: switch state: 8
http POST: switch state: 0

 Tests of vector operations.
 File FP1101P0.DEC not found.
 Default firmware will be used.
 File FP1101P1.DEC not found.
 Default firmware will be used.
 File FP1101.TXT created for wr
 MB0 : (EdkDSP 8xSIMD) Capabilities1 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities2 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities3 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities4 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities5 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities6 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities7 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities8 = 13FFFF

 ah=0  bh=0  zh=0
 MB0 : (EdkDSP 8xSIMD) VZ2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VB2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VZ2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VA2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VADD_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VSUB_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VMULT 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMULT_BZ2A 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VMULT_AZ2B 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VPROD 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMAC 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VMSUBAC 'worker1' .... OK
 MB0 : (EdkDSP 8xSIMD) VPROD_S8 'worker1' ... OK
 MB0 : (EdkDSP 8xSIMD) VDIV 'worker1' ....... OK
Blocks_used 237
Blocks_free 1811
Directory css 00000003
Directory images 00000005
index.html 00000b96
Directory js 00000003
Directory yui 00000007
FP1101.TXT 0000061e
http POST: ledstatus: FFFFFFFF
```

Figure 27: Test of vector operations is started from the www browser GUI. It is served by the lwIP library working on top of the Xilkernel.

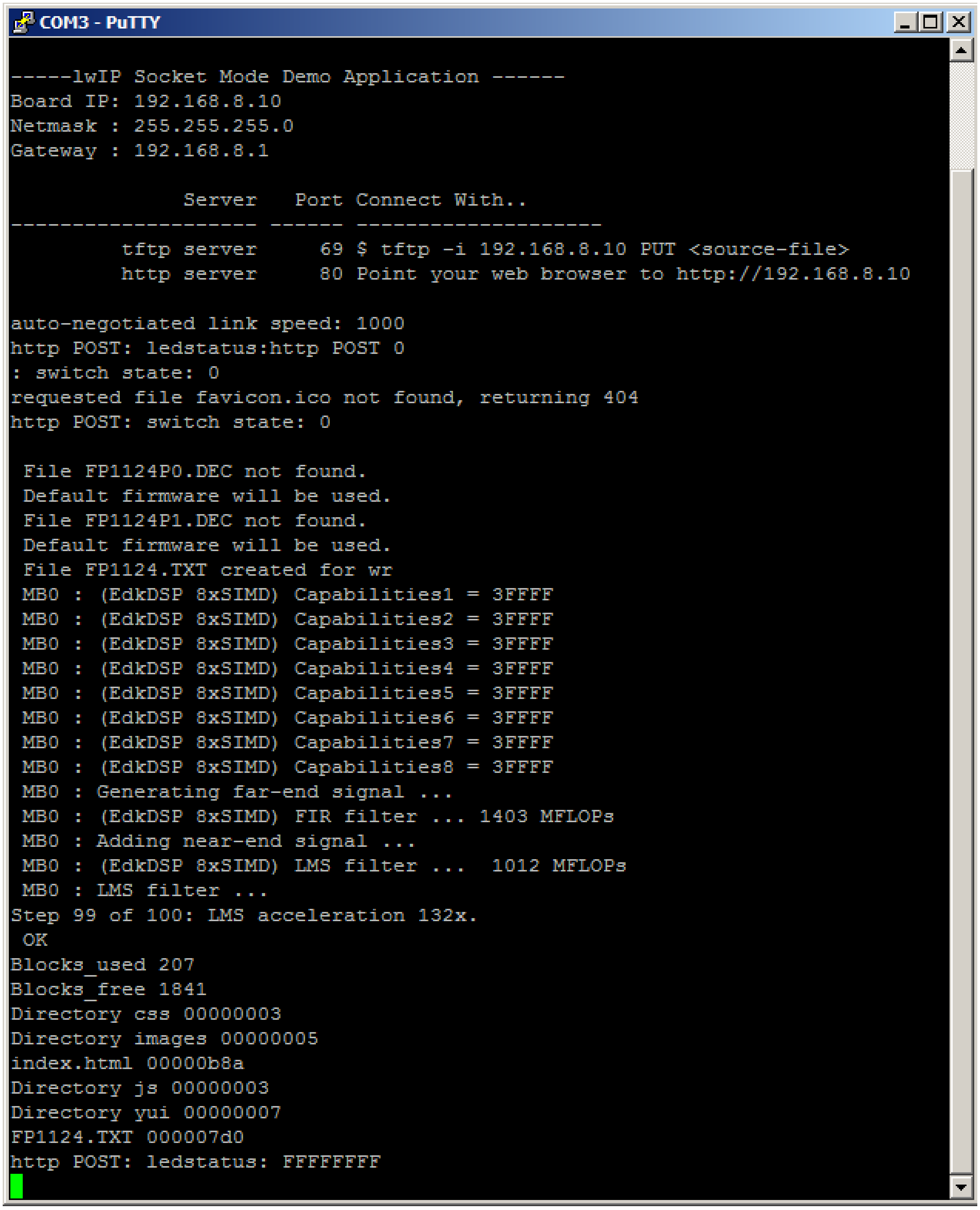
Close the web browser. Close the socket based application running on the Kintex7 from the SDK.
Download again the bitstream (chapter 3.4), select the socket_axi_bce_fp12_1x8_fir_lms project for download (chapter 3.5) and run it.



```
 MB0 : (EdkDSP 8xSIMD) Capabilities7 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities8 = 13FFFF

 ah=0  bh=0  zh=0
 MB0 : (EdkDSP 8xSIMD) VZ2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VB2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VZ2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VA2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VADD_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VSUB_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VMULT 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMULT_BZ2A 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VMULT_AZ2B 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VPROD 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMAC 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VMSUBAC 'worker1' .... OK
 MB0 : (EdkDSP 8xSIMD) VPROD_S8 'worker1' ... OK
 MB0 : (EdkDSP 8xSIMD) VDIV 'worker1' ....... OK
Blocks_used 237
Blocks_free 1811
Directory css 00000003
Directory images 00000005
index.html 00000b96
Directory js 00000003
Directory yui 00000007
FP1101.TXT 0000061e
http POST: ledstatus: FFFFFFFF
Initializing MFS at 0x803455BC
Done.
Located index.html
Initializing video cores.


-----lwIP Socket Mode Demo Application ------
Board IP: 192.168.8.10
Netmask : 255.255.255.0
Gateway : 192.168.8.1

              Server   Port Connect With..
-------------------- ------ --------------------
         tftp server     69 $ tftp -i 192.168.8.10 PUT <source-file>
         http server     80 Point your web browser to http://192.168.8.10

auto-negotiated link speed: 1000
```

Figure 28: Start the socket_axi_bce_fp12_1x8_fir_lms.elf application.

The SOCKET version of the TFTP and HTTP servers have been started on the Kintex7 MicroBlaze processor. Open www browser (Internet Explorer) client and connect to the board address: http://192.168.8.10/
Click on the **Toggle LEDs** button to toggle the led output on the board and starts the FIR and LMS filter computation on single (8xSIMS) EdkDSP accelerator. See Figure 29.



```
-----lwIP Socket Mode Demo Application ------
Board IP: 192.168.8.10
Netmask : 255.255.255.0
Gateway : 192.168.8.1

              Server   Port Connect With..
-------------------- ------ --------------------
         tftp server     69 $ tftp -i 192.168.8.10 PUT <source-file>
         http server     80 Point your web browser to http://192.168.8.10

auto-negotiated link speed: 1000
http POST: ledstatus:http POST 0
: switch state: 0
requested file favicon.ico not found, returning 404
http POST: switch state: 0

 File FP1124P0.DEC not found.
 Default firmware will be used.
 File FP1124P1.DEC not found.
 Default firmware will be used.
 File FP1124.TXT created for wr
 MB0 : (EdkDSP 8xSIMD) Capabilities1 = 3FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities2 = 3FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities3 = 3FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities4 = 3FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities5 = 3FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities6 = 3FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities7 = 3FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities8 = 3FFFF
 MB0 : Generating far-end signal ...
 MB0 : (EdkDSP 8xSIMD) FIR filter ... 1403 MFLOPs
 MB0 : Adding near-end signal ...
 MB0 : (EdkDSP 8xSIMD) LMS filter ...  1012 MFLOPs
 MB0 : LMS filter ...
Step 99 of 100: LMS acceleration 132x.
 OK
Blocks_used 207
Blocks_free 1841
Directory css 00000003
Directory images 00000005
index.html 00000b8a
Directory js 00000003
Directory yui 00000007
FP1124.TXT 000007d0
http POST: ledstatus: FFFFFFFF
```

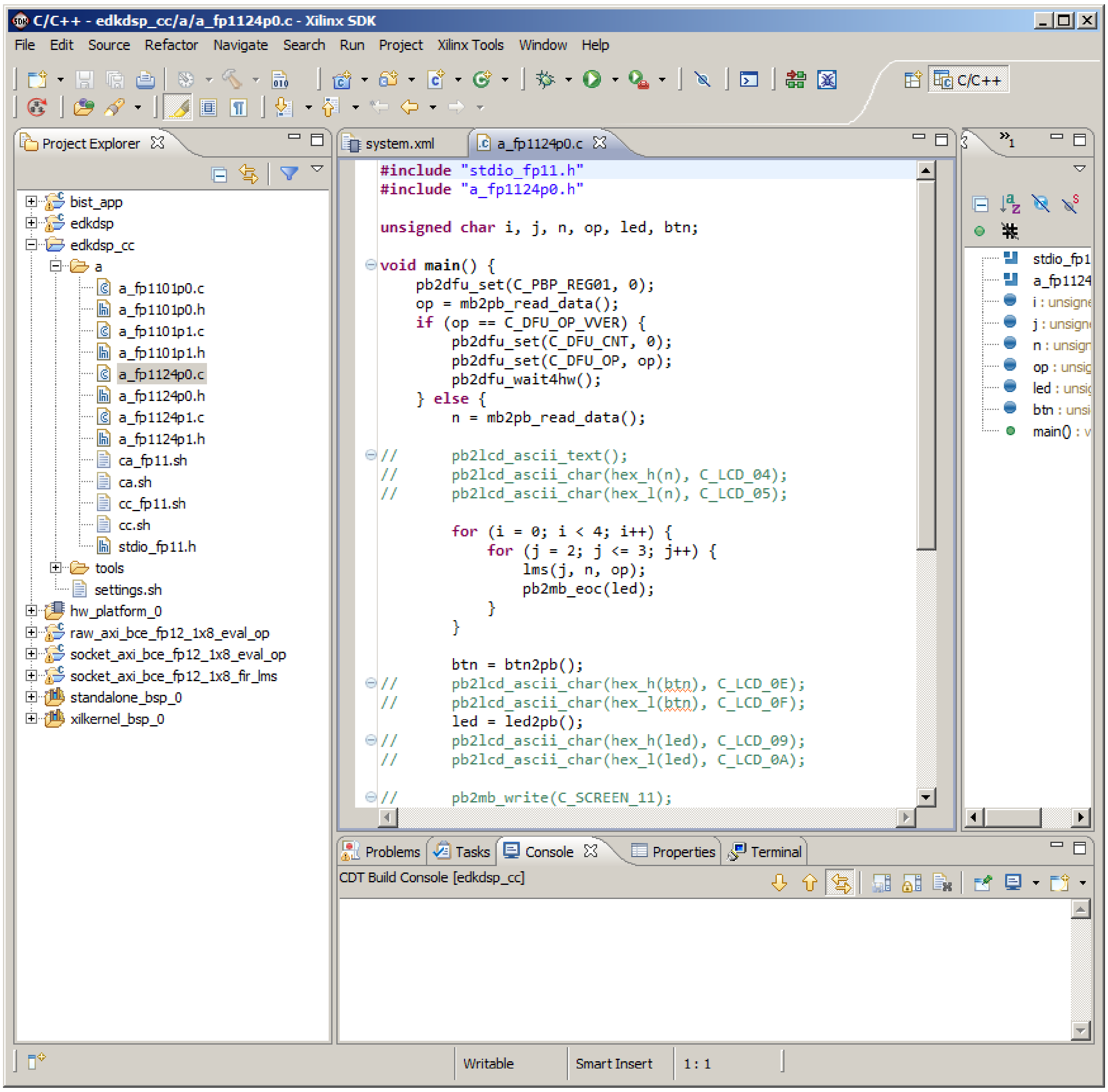
Figure 29: The FIR and LMS computation is started from the web browser GUI. The performance of single EdkDSP accelerator is measured and compared to the performance of MicroBlaze processor with HW floating point unit.

The performance for FIR and LMS is displayed and the speedup in comparison to the MicroBlaze is reported during the MicroBlaze verification run. The result from the EdkDSP is identical to the MicroBlaze result.

3.6 Use of the C compiler for the EdkDSP firmware with download from Ethernet

This section is describing the use of the UTIA EdkDSP C compiler to recompile the firmware for the PicoBlaze6 controller present in each of the eight (8xSIMD) EdkDSP accelerators in the KC705 board.

In SDK Project Explorer, open the project edkdsp_cc and the subdirectory edkdsp_cc/a. See Figure 30. It contains C source code of the EdkDSP accelerator firmware and Ubuntu scripts for the compilation. The compiled versions of firmware are already present in the demonstrated applications in form of headers for the MicroBlaze applications. This helps to evaluate the EdkDSP accelerators without installation of the C compiler for the EdkDSP.



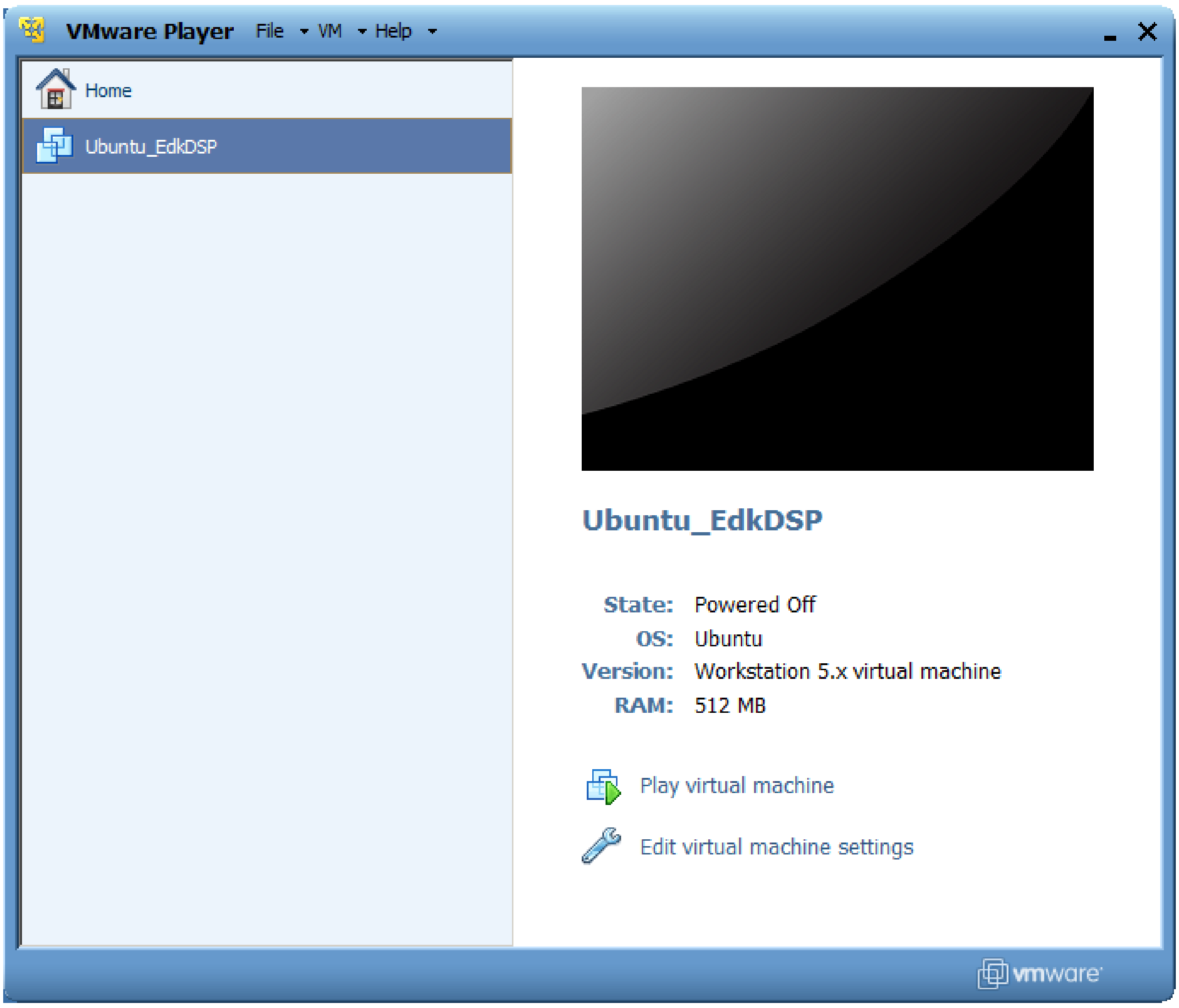
```c
#include "stdio_fp11.h"
#include "a_fp1124p0.h"

unsigned char i, j, n, op, led, btn;

void main() {
    pb2dfu_set(C_PBP_REG01, 0);
    op = mb2pb_read_data();
    if (op == C_DFU_OP_VVER) {
        pb2dfu_set(C_DFU_CNT, 0);
        pb2dfu_set(C_DFU_OP, op);
        pb2dfu_wait4hw();
    } else {
        n = mb2pb_read_data();

//      pb2lcd_ascii_text();
//      pb2lcd_ascii_char(hex_h(n), C_LCD_04);
//      pb2lcd_ascii_char(hex_l(n), C_LCD_05);

        for (i = 0; i < 4; i++) {
            for (j = 2; j <= 3; j++) {
                lms(j, n, op);
                pb2mb_eoc(led);
            }
        }

        btn = btn2pb();
//      pb2lcd_ascii_char(hex_h(btn), C_LCD_0E);
//      pb2lcd_ascii_char(hex_l(btn), C_LCD_0F);
        led = led2pb();
//      pb2lcd_ascii_char(hex_h(led), C_LCD_09);
//      pb2lcd_ascii_char(hex_l(led), C_LCD_0A);

//      pb2mb_write(C_SCREEN_11);
```

Figure 30: Evaluate the included C code for reprograming of the EdkDSP accelerators.

The UTIA EdkDSP C compiler is provided as implemented as several Ubuntu binary applications.
The “VMware player” software and the compatible Ubuntu image version is needed to run the UTIA EdkDSP C compiler on Windows 7 (64bit or 32bit) PC, Linux (32 or 64 bit) or the legacy Win XP (32 or 64 bit).

The Ubuntu image used in UTIA needs two DVD disks (8GB) for installation. That is why it is not included as part of the evaluation package. If you would need this image, write an email request to kadlec@utia.cas.cz to get these two DVD with correct Ubuntu image from UTIA (free of charge).

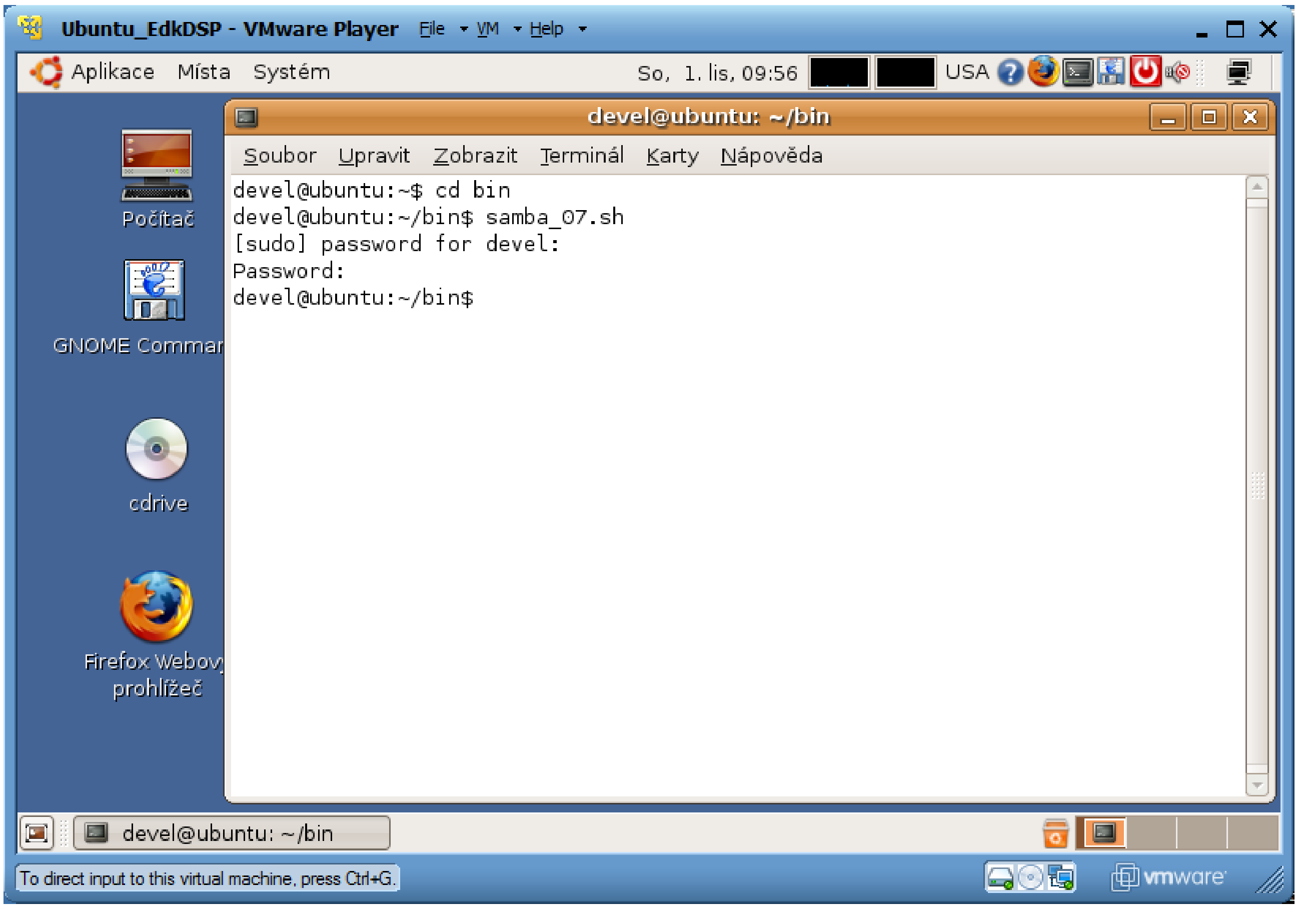
Install the VMware Player software (64bit or 32bit) on your PC.
In VMware Player open the Ubuntu_EdkDSP package. See Figure 31.



Ubuntu_EdkDSP

State: Powered Off
OS: Ubuntu
Version: Workstation 5.x virtual machine
RAM: 512 MB

Figure 31: Start the VMware Player to run the C compiler for the EdkDSP accelerators as an Ubuntu binary user application.



```
devel@ubuntu:~$ cd bin
devel@ubuntu:~/bin$ samba_07.sh
[sudo] password for devel:
Password:
devel@ubuntu:~/bin$
```

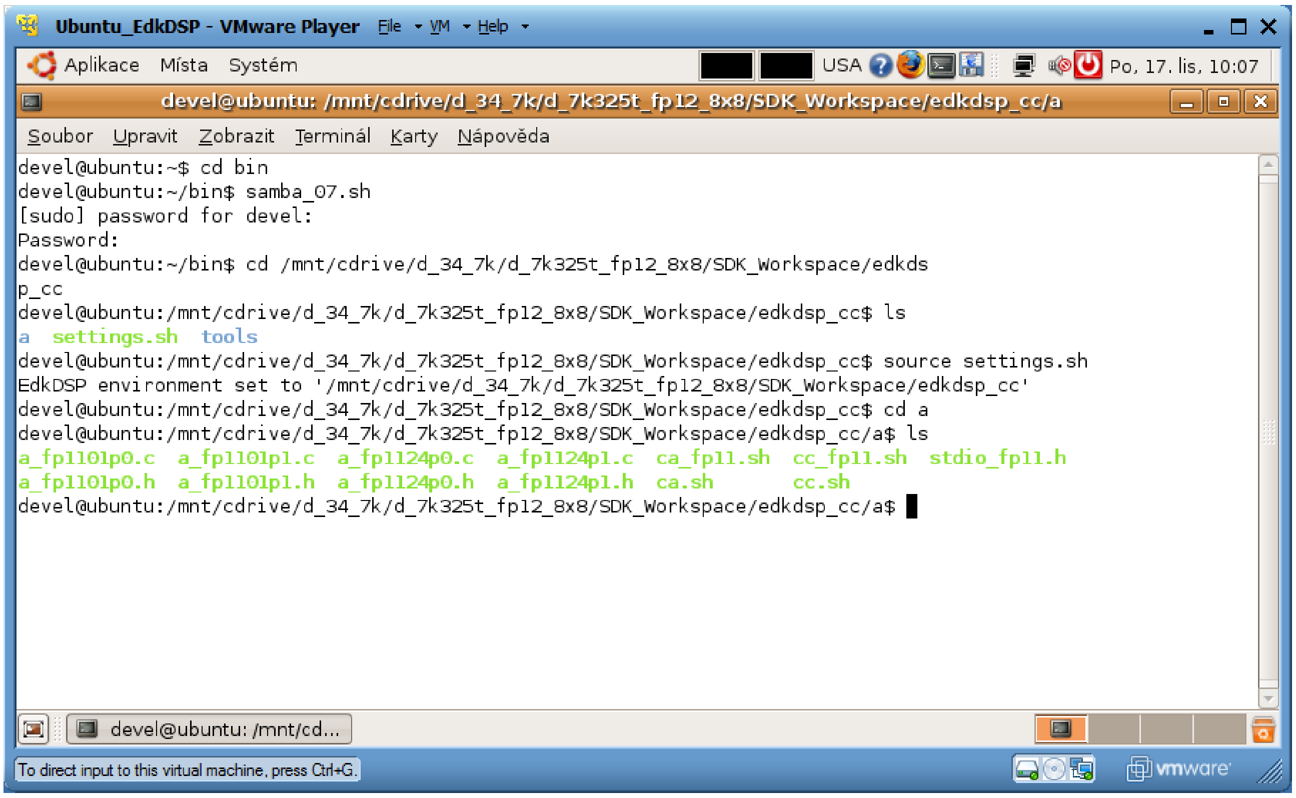
Figure 32: Mount the Windows 7 directoy c:\VM_07as /mnt/cdrive in Ubuntu

Open the VMware Player and select the “Ubuntu_EdkDSP” image. The Ubuntu will start.
Login as:
User: devel
Pswd: devuser

The PC directory c:\VM_07 needs to be shared by Windows 7 with Ubuntu.
In Windows 7, set the directory c:\VM_07 and its subdirectories as shared with the __vmware_user__ for Read and Write. In Ubuntu, open terminal and mount the PC directory c:\VM_07 to Ubuntu.
The Windows 7 c:/VM_07 directory is mounted to the Ubuntu OS as: /mnt/cdrive
This process has been automated by the script samba_07.sn in my case. See Figure 32.

In Ubuntu terminal, change the directory to:
$ cd /mnt/cdrive/d_34_7k/d_7k325t_fp12_6x8/SDK_Workspace/edkdsp_cc

The EdkDSP C compiler utilities have to be on the Ubuntu PATH. This is done by sourcing the settings.sh script in this directory. Type in Ubuntu terminal (See Figure 33):
$ source settings.sh
In Ubuntu terminal, change the directory to the example directory (See Figure 33):
$ cd a
devel@ubuntu:/mnt/cdrive/d_34_7z/d_7z020_fp12_4x8/SDK_Workspace/edkdsp_cc/a$


Ubuntu_EdkDSP - VMware Player
File
VM
Help
Aplikace
Místa
Systém
USA
Po, 17. lis, 10:07
devel@ubuntu: /mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc/a
Soubor
Upravit
Zobrazit
Terminál
Karty
Nápověda
devel@ubuntu: /mnt/cd...
To direct input to this virtual machine, press Ctrl+G.
vmware

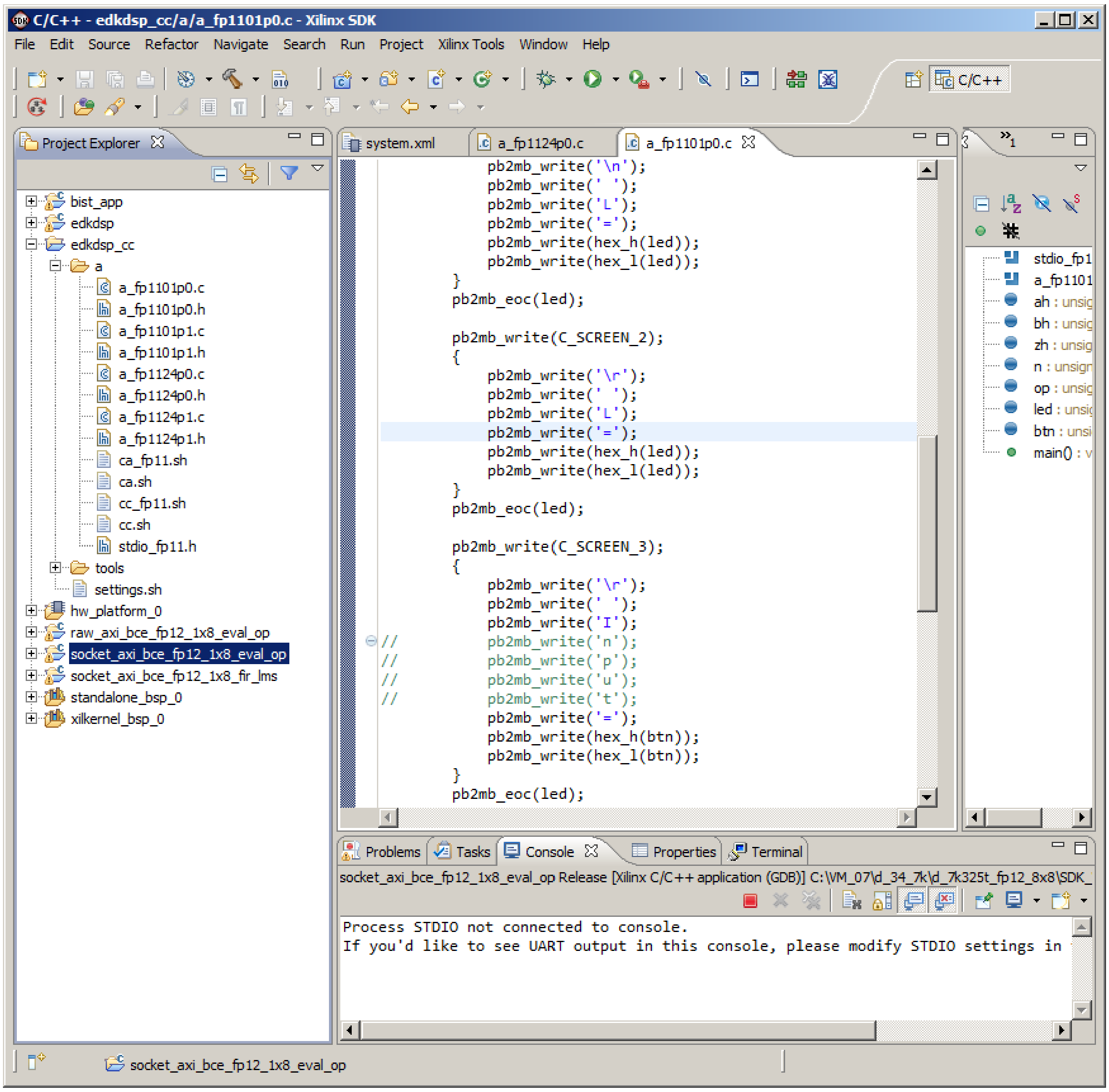
```
devel@ubuntu:~$ cd bin
devel@ubuntu:~/bin$ samba_07.sh
[sudo] password for devel:
Password:
devel@ubuntu:~/bin$ cd /mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkds
p_cc
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc$ ls
a  settings.sh  tools
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc$ source settings.sh
EdkDSP environment set to '/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc'
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc$ cd a
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc/a$ ls
a_fp1101p0.c  a_fp1101p1.c  a_fp1124p0.c  a_fp1124p1.c  ca_fp11.sh  cc_fp11.sh  stdio_fp11.h
a_fp1101p0.h  a_fp1101p1.h  a_fp1124p0.h  a_fp1124p1.h  ca.sh       cc.sh
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc/a$ 
```

Figure 33: Source the path to the EdkDSP C compiler tools.

In SDK, open the C source code of the current firmware for the EdkDSP accelerator in the file edkdsp_cc/a/a_fp1101p0.c

See the original listing in Figure 34.
To demonstrate the compilation and new firmware download via Ethernet, We will change the message going from EdkDSP PicoBlaze processor to the MicroBlaze and to the FP1101.TXT log file from I=00; to Input=00.

Uncomment the four commented lines from // pb2mb_Write ('n'); to // pb2mb_Write ('t');
See Figure 34.
Save the modifications.



```
            pb2mb_write('\n');
            pb2mb_write(' ');
            pb2mb_write('L');
            pb2mb_write('=');
            pb2mb_write(hex_h(led));
            pb2mb_write(hex_l(led));
        }
        pb2mb_eoc(led);

        pb2mb_write(C_SCREEN_2);
        {
            pb2mb_write('\r');
            pb2mb_write(' ');
            pb2mb_write('L');
            pb2mb_write('=');
            pb2mb_write(hex_h(led));
            pb2mb_write(hex_l(led));
        }
        pb2mb_eoc(led);

        pb2mb_write(C_SCREEN_3);
        {
            pb2mb_write('\r');
            pb2mb_write(' ');
            pb2mb_write('I');
//          pb2mb_write('n');
//          pb2mb_write('p');
//          pb2mb_write('u');
//          pb2mb_write('t');
            pb2mb_write('=');
            pb2mb_write(hex_h(btn));
            pb2mb_write(hex_l(btn));
        }
        pb2mb_eoc(led);
```

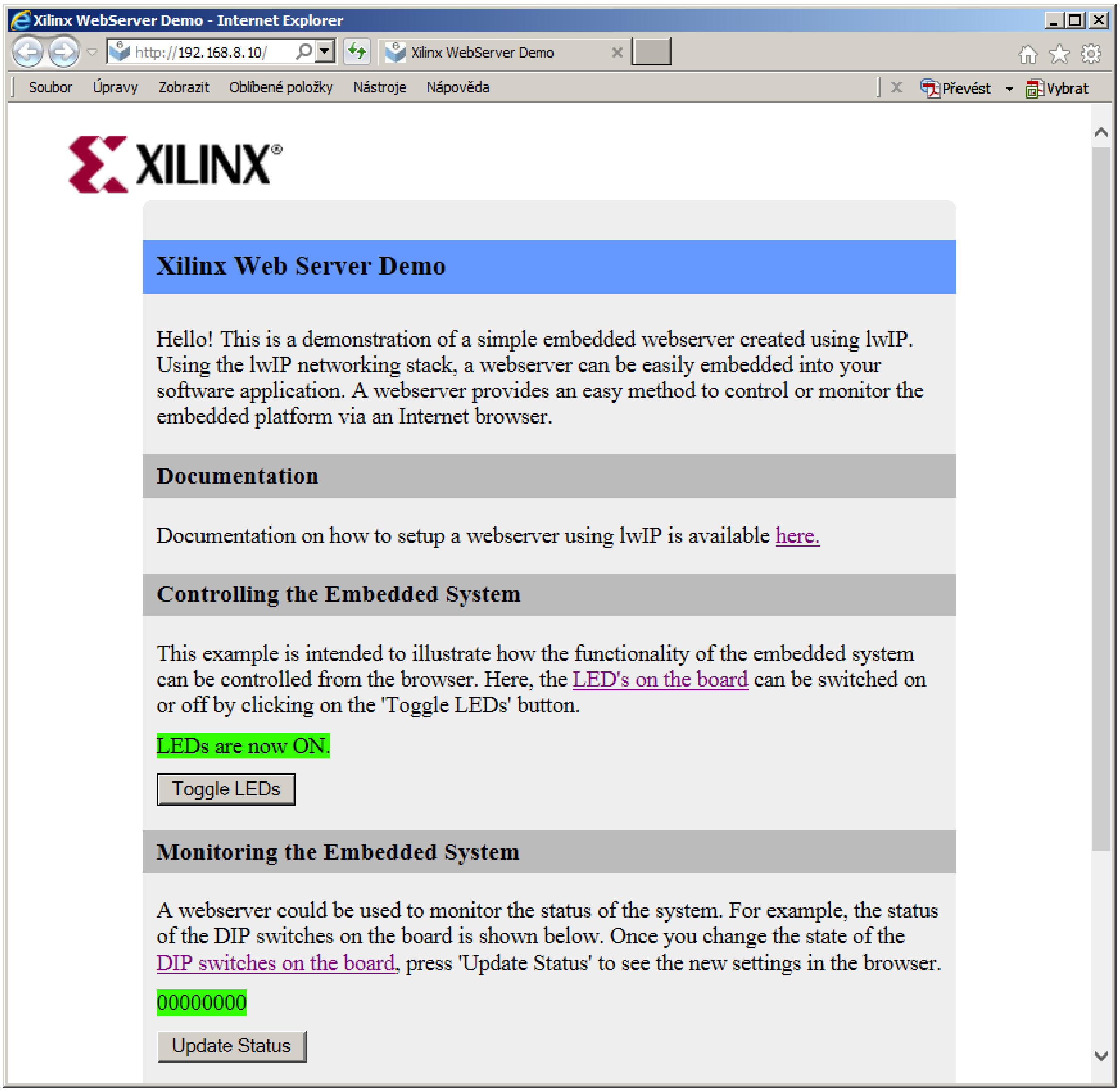
```
socket_axi_bce_fp12_1x8_eval_op Release [Xilinx C/C++ application (GDB)] C:\VM_07\d_34_7k\d_7k325t_fp12_8x8\SDK_
```

```
Process STDIO not connected to console.
If you'd like to see UART output in this console, please modify STDIO settings in
```

Figure 34: See the details of communication from the accelerator to MicroBlaze in the original code.

We will demonstrate the complete process related to the compilation, download of results from Kintex7 to the PC and upload of the bitstreem to the Kintex7 now.

Start the application socket_axi_bce_fp12_1x8_eval_op.elf and open the www browser and start the demo run by clicking on the Toggle LEDs button.
See Figure 35 and Figure 36.


Xilinx WebServer Demo - Internet Explorer
http://192.168.8.10/
Xilinx WebServer Demo
Soubor Úpravy Zobrazit Oblíbené položky Nástroje Nápověda
Převést Vybrat
XILINX

Xilinx Web Server Demo

Hello! This is a demonstration of a simple embedded webserver created using lwIP. Using the lwIP networking stack, a webserver can be easily embedded into your software application. A webserver provides an easy method to control or monitor the embedded platform via an Internet browser.

Documentation

Documentation on how to setup a webserver using lwIP is available here.

Controlling the Embedded System

This example is intended to illustrate how the functionality of the embedded system can be controlled from the browser. Here, the LED's on the board can be switched on or off by clicking on the 'Toggle LEDs' button.

LEDs are now ON.

Toggle LEDs

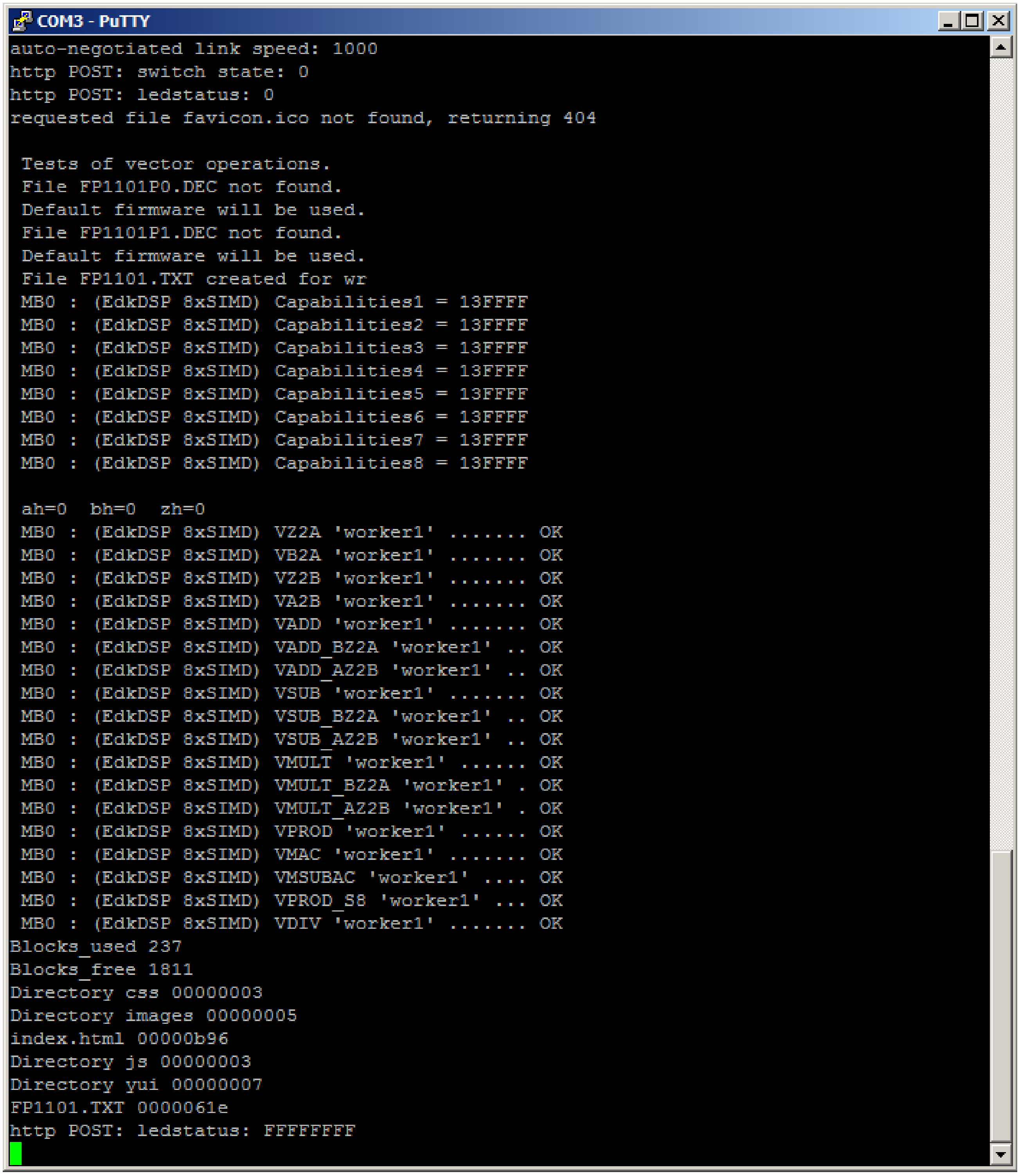
Monitoring the Embedded System

A webserver could be used to monitor the status of the system. For example, the status of the DIP switches on the board is shown below. Once you change the state of the DIP switches on the board, press 'Update Status' to see the new settings in the browser.

00000000

Update Status

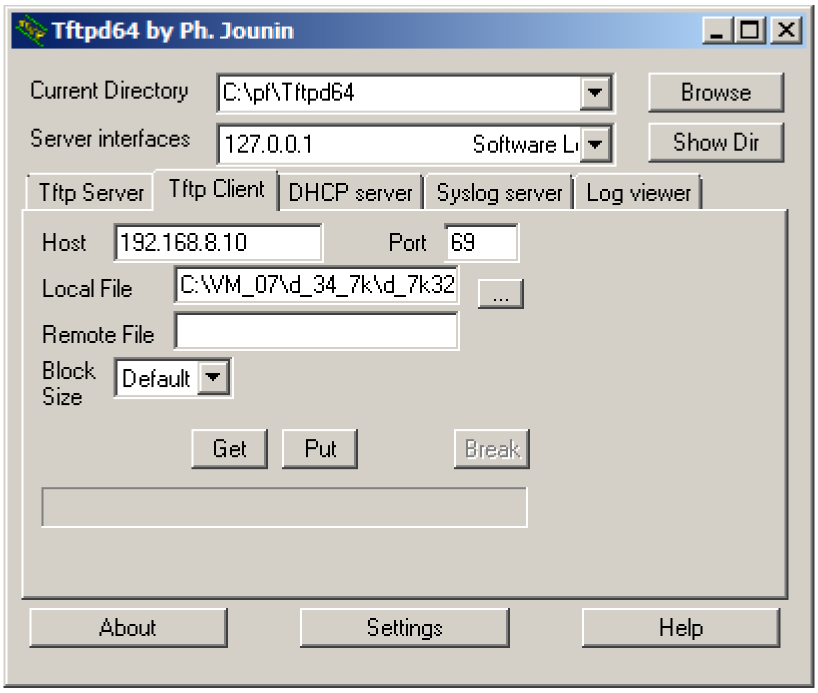
Figure 35: Start test from the web brawser GUI by Toggle LEDs button.



```
auto-negotiated link speed: 1000
http POST: switch state: 0
http POST: ledstatus: 0
requested file favicon.ico not found, returning 404

 Tests of vector operations.
 File FP1101P0.DEC not found.
 Default firmware will be used.
 File FP1101P1.DEC not found.
 Default firmware will be used.
 File FP1101.TXT created for wr
 MB0 : (EdkDSP 8xSIMD) Capabilities1 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities2 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities3 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities4 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities5 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities6 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities7 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities8 = 13FFFF

 ah=0  bh=0  zh=0
 MB0 : (EdkDSP 8xSIMD) VZ2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VB2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VZ2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VA2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VADD_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VSUB_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VMULT 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMULT_BZ2A 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VMULT_AZ2B 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VPROD 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMAC 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VMSUBAC 'worker1' .... OK
 MB0 : (EdkDSP 8xSIMD) VPROD_S8 'worker1' ... OK
 MB0 : (EdkDSP 8xSIMD) VDIV 'worker1' ....... OK
Blocks_used 237
Blocks_free 1811
Directory css 00000003
Directory images 00000005
index.html 00000b96
Directory js 00000003
Directory yui 00000007
FP1101.TXT 0000061e
http POST: ledstatus: FFFFFFFF
```

Figure 36: Test has been performed and the tested EdkDSP accelerator created data file FP1101.TXT in the RAM file system located in the DDR3 of the KC705 board.

Open the TFTP application on your PC as a TFTP client connected to the Kintex7 host 192.168.8.10 with Port 69. See Figure 37. Select Local (PC) file to:
c:\VM_07\ d_34_7z\d_7z020_fp12_4x8\SDK_Workspace\edkdsp_cc\a\FP1101.TXT
and Remote File (Kintex7 file system) to:
FP1101.TXT
See Figure 37 and Figure 38 for the selection of the PC file location. Click on Get to download the file.



Tftpd64 by Ph. Jounin

Current Directory: C:\pf\Tftpd64

Server interfaces: 127.0.0.1 Software L

Tftp Client

Host: 192.168.8.10 Port: 69

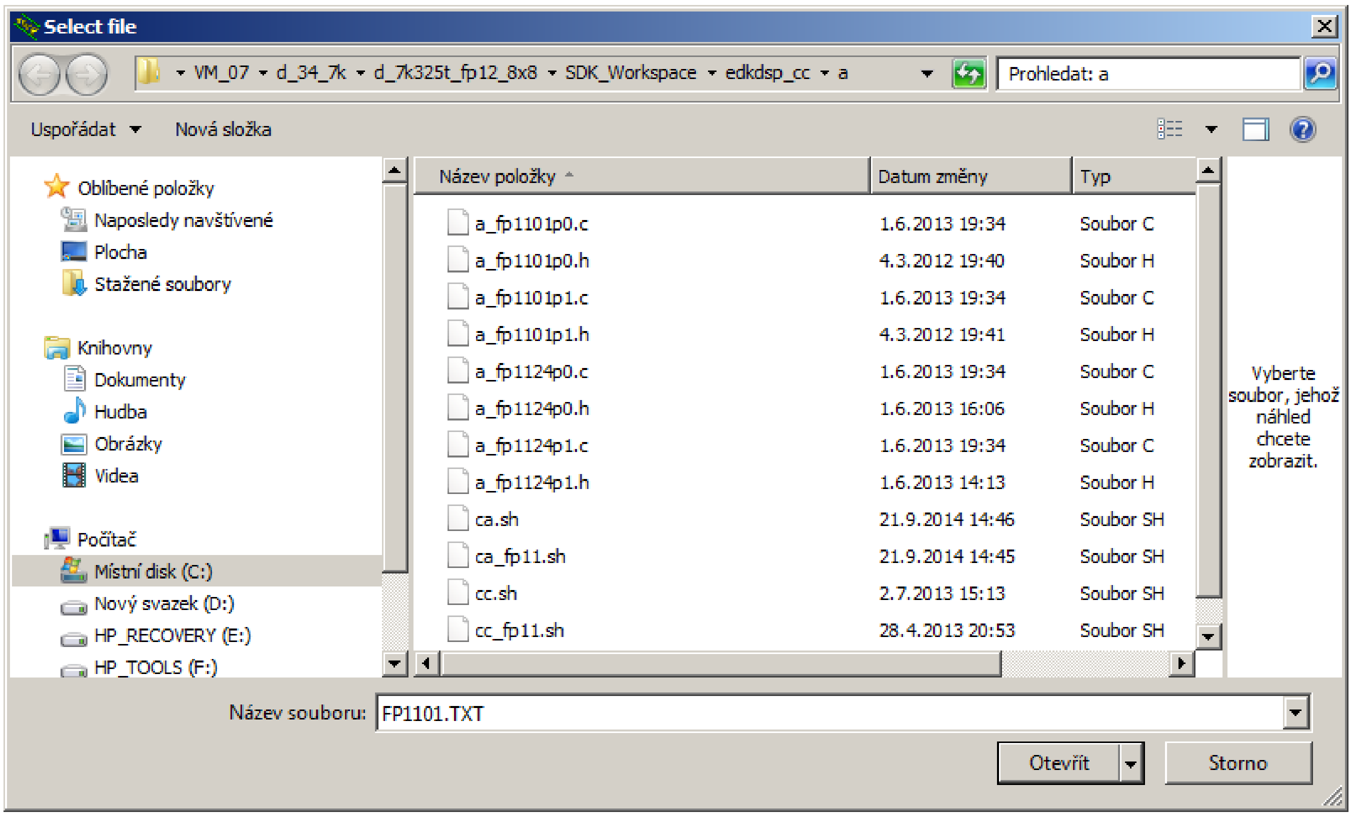
Local File: C:\VM_07\d_34_7k\d_7k32

Remote File:

Block Size: Default

Figure 37: Start TFTP client and get the file FP1101.TXR from the Kintex7 FPGA to PC via Ethernet.

The EdkDSP firmware after the compilation is presented in Figure 36.



Select file

VM_07 ▸ d_34_7k ▸ d_7k325t_fp12_8x8 ▸ SDK_Workspace ▸ edkdsp_cc ▸ a

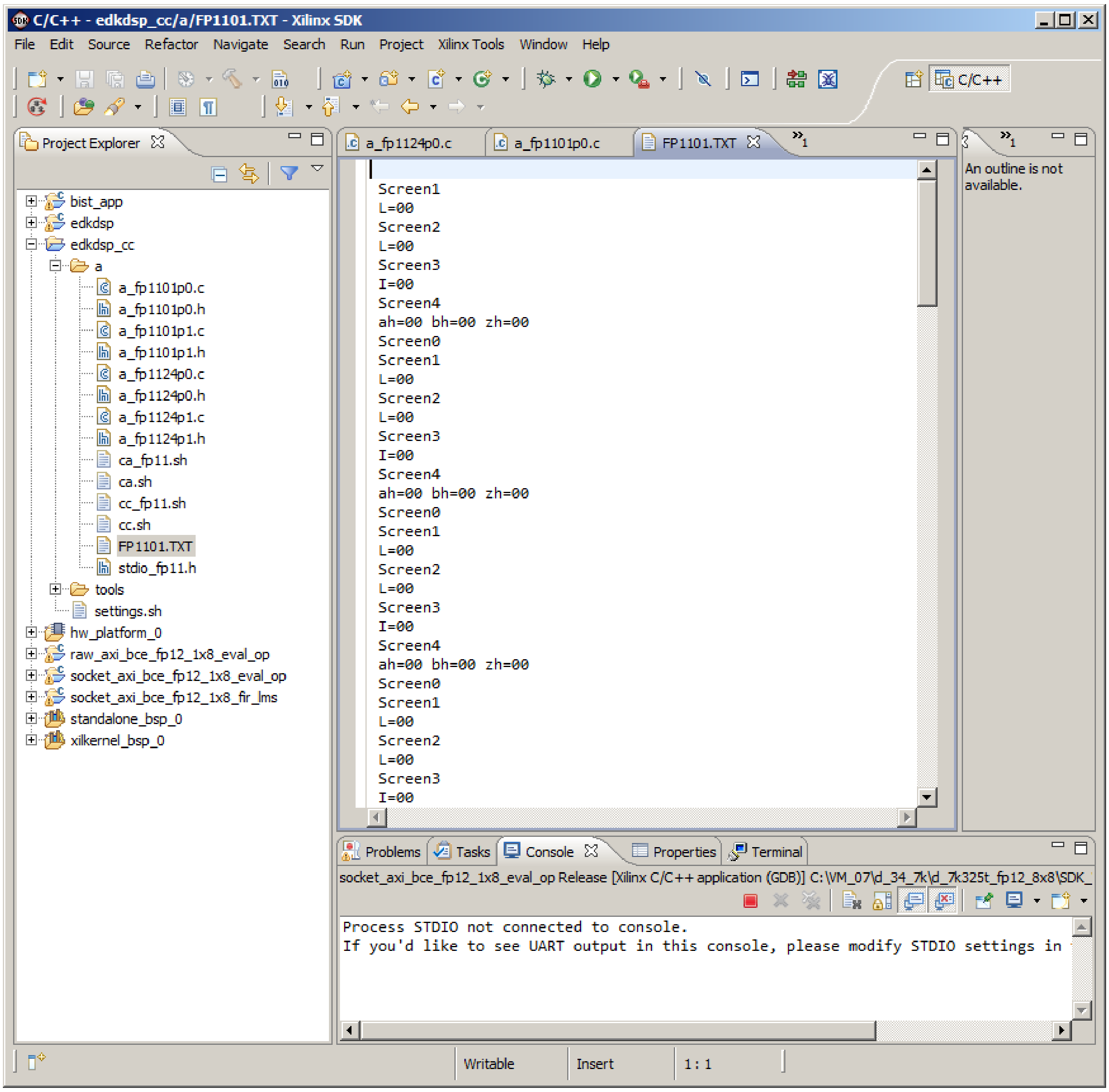
Prohledat: a

Název položky	Datum změny	Typ
a_fp1101p0.c	1.6.2013 19:34	Soubor C
a_fp1101p0.h	4.3.2012 19:40	Soubor H
a_fp1101p1.c	1.6.2013 19:34	Soubor C
a_fp1101p1.h	4.3.2012 19:41	Soubor H
a_fp1124p0.c	1.6.2013 19:34	Soubor C
a_fp1124p0.h	1.6.2013 16:06	Soubor H
a_fp1124p1.c	1.6.2013 19:34	Soubor C
a_fp1124p1.h	1.6.2013 14:13	Soubor H
ca.sh	21.9.2014 14:46	Soubor SH
ca_fp11.sh	21.9.2014 14:45	Soubor SH
cc.sh	2.7.2013 15:13	Soubor SH
cc_fp11.sh	28.4.2013 20:53	Soubor SH

Vyberte soubor, jehož náhled chcete zobrazit.

Název souboru: FP1101.TXT

Figure 38: Select the directory where you want to get the FP1101.TXT file.



```
Screen1
L=00
Screen2
L=00
Screen3
I=00
Screen4
ah=00 bh=00 zh=00
Screen0
Screen1
L=00
Screen2
L=00
Screen3
I=00
Screen4
ah=00 bh=00 zh=00
Screen0
Screen1
L=00
Screen2
L=00
Screen3
I=00
Screen4
ah=00 bh=00 zh=00
Screen0
Screen1
L=00
Screen2
L=00
Screen3
I=00
```

socket_axi_bce_fp12_1x8_eval_op Release [Xilinx C/C++ application (GDB)] C:\VM_07\d_34_7k\d_7k325t_fp12_8x8\SDK_

```
Process STDIO not connected to console.
If you'd like to see UART output in this console, please modify STDIO settings in
```

Figure 39: In SDK, Refresh the edkdsp_cc/a directory (by F5) to see the received FP1101.TXT file downloaded from the server running on the Kintex7 FPGA. Notice that the input data are printed as I=00.

Refresh the project explorer view by F5. The uploaded log file FP1101.TXT can be open. See Figure 39. The PicoBlaze6 original firmware is writing I=00 to the log file as expected.

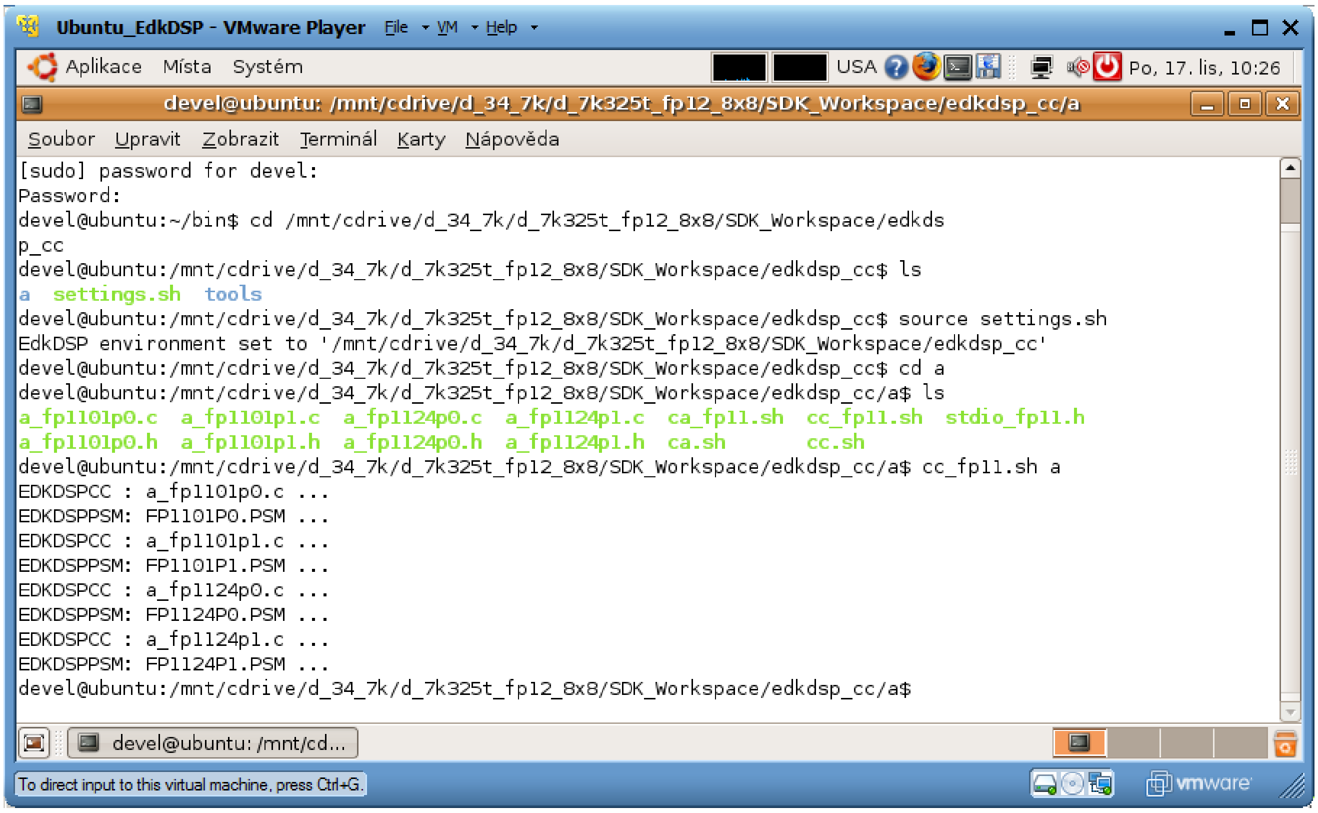
Keep the application running on the Kintex7 together with the browser GUI.

Compile the modified firmware source code by script cc_fp11.sh with parameter a.
Type in the Ubuntu terminal:
$ cc_fp11.sh a

This will compile and assemble all four C firmware programs to header files with the firmware binary code (See Figure 40):

a_fp1101p0.c	is compiled to FP1101P0.DEC
a_fp1101p1.c	is compiled to FP1101P1.DEC
a_fp1124p0.c	is compiled to FP1124P0.DEC
a_fp1124p1.c	is compiled to FP1124P0.DEC

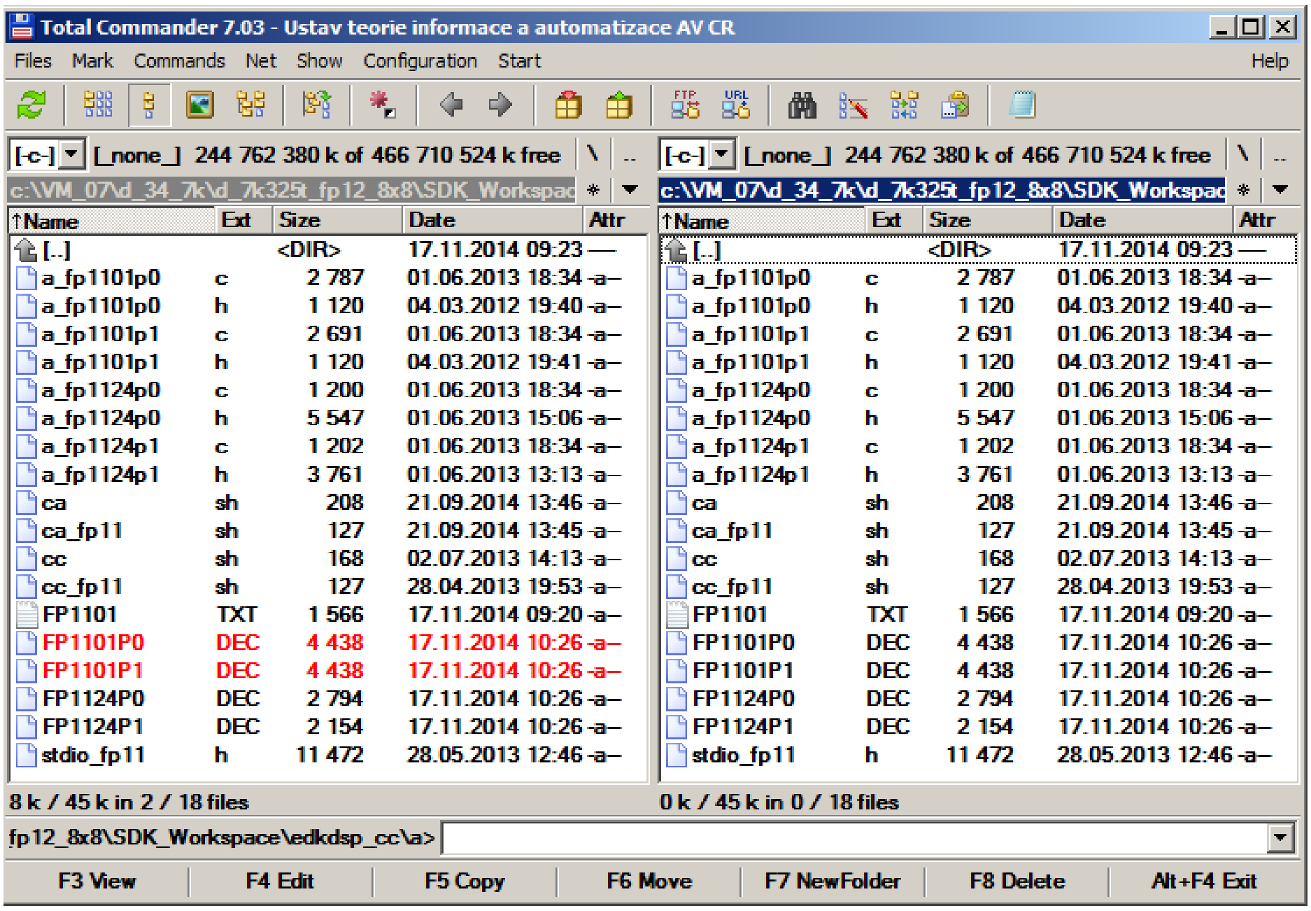
This compiled firmware can be uploaded from PC to the running demo application in the Kintex7 chip.


Ubuntu_EdkDSP - VMware Player
File
VM
Help
Aplikace
Místa
Systém
USA
Po, 17. lis, 10:26
devel@ubuntu: /mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc/a
Soubor
Upravit
Zobrazit
Terminál
Karty
Nápověda
devel@ubuntu: /mnt/cd...
To direct input to this virtual machine, press Ctrl+G.
vmware

```
[sudo] password for devel:
Password:
devel@ubuntu:~/bin$ cd /mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkds
p_cc
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc$ ls
a  settings.sh  tools
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc$ source settings.sh
EdkDSP environment set to '/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc'
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc$ cd a
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc/a$ ls
a_fp1101p0.c  a_fp1101p1.c  a_fp1124p0.c  a_fp1124p1.c  ca_fp11.sh  cc_fp11.sh  stdio_fp11.h
a_fp1101p0.h  a_fp1101p1.h  a_fp1124p0.h  a_fp1124p1.h  ca.sh       cc.sh
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc/a$ cc_fp11.sh a
EDKDSPCC : a_fp1101p0.c ...
EDKDSPPSM: FP1101P0.PSM ...
EDKDSPCC : a_fp1101p1.c ...
EDKDSPPSM: FP1101P1.PSM ...
EDKDSPCC : a_fp1124p0.c ...
EDKDSPPSM: FP1124P0.PSM ...
EDKDSPCC : a_fp1124p1.c ...
EDKDSPPSM: FP1124P1.PSM ...
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc/a$
```

Figure 40: Compile the C code with uncommented lines to display Input=00 instead of I=00

Upload the compiled firmware from PC to the Kintex7 File system. See Figure 41 - Figure 44.


Total Commander 7.03 - Ustav teorie informace a automatizace AV CR
Files Mark Commands Net Show Configuration Start Help
[-c-] [_none_] 244 762 380 k of 466 710 524 k free
c:\VM_07\d_34_7k\d_7k325t_fp12_8x8\SDK_Workspac

↑Name	Ext	Size	Date	Attr
[..]		<DIR>	17.11.2014 09:23	—
a_fp1101p0	c	2 787	01.06.2013 18:34	-a–
a_fp1101p0	h	1 120	04.03.2012 19:40	-a–
a_fp1101p1	c	2 691	01.06.2013 18:34	-a–
a_fp1101p1	h	1 120	04.03.2012 19:41	-a–
a_fp1124p0	c	1 200	01.06.2013 18:34	-a–
a_fp1124p0	h	5 547	01.06.2013 15:06	-a–
a_fp1124p1	c	1 202	01.06.2013 18:34	-a–
a_fp1124p1	h	3 761	01.06.2013 13:13	-a–
ca	sh	208	21.09.2014 13:46	-a–
ca_fp11	sh	127	21.09.2014 13:45	-a–
cc	sh	168	02.07.2013 14:13	-a–
cc_fp11	sh	127	28.04.2013 19:53	-a–
FP1101	TXT	1 566	17.11.2014 09:20	-a–
FP1101P0	DEC	4 438	17.11.2014 10:26	-a–
FP1101P1	DEC	4 438	17.11.2014 10:26	-a–
FP1124P0	DEC	2 794	17.11.2014 10:26	-a–
FP1124P1	DEC	2 154	17.11.2014 10:26	-a–
stdio_fp11	h	11 472	28.05.2013 12:46	-a–

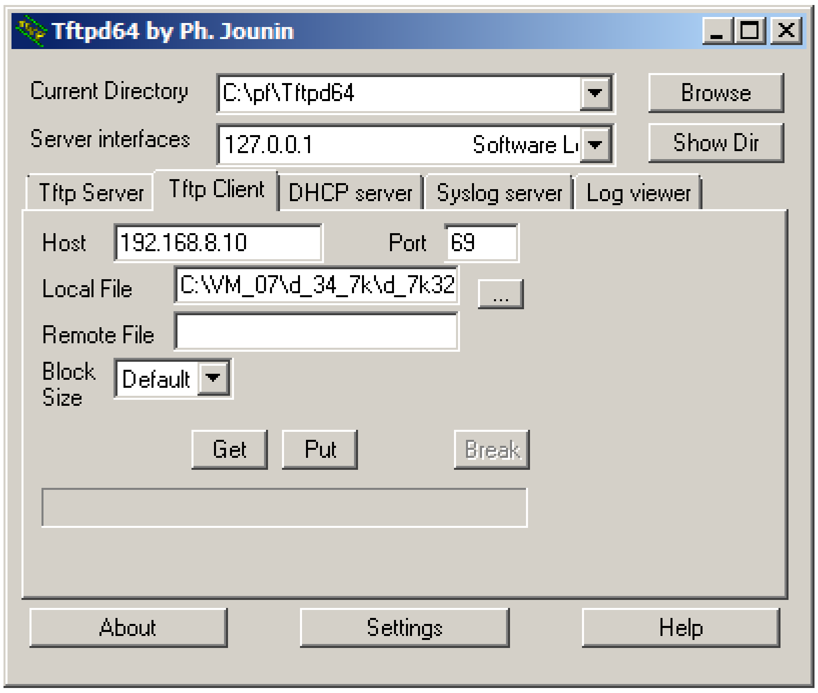
8 k / 45 k in 2 / 18 files

↑Name	Ext	Size	Date	Attr
[..]		<DIR>	17.11.2014 09:23	—
a_fp1101p0	c	2 787	01.06.2013 18:34	-a–
a_fp1101p0	h	1 120	04.03.2012 19:40	-a–
a_fp1101p1	c	2 691	01.06.2013 18:34	-a–
a_fp1101p1	h	1 120	04.03.2012 19:41	-a–
a_fp1124p0	c	1 200	01.06.2013 18:34	-a–
a_fp1124p0	h	5 547	01.06.2013 15:06	-a–
a_fp1124p1	c	1 202	01.06.2013 18:34	-a–
a_fp1124p1	h	3 761	01.06.2013 13:13	-a–
ca	sh	208	21.09.2014 13:46	-a–
ca_fp11	sh	127	21.09.2014 13:45	-a–
cc	sh	168	02.07.2013 14:13	-a–
cc_fp11	sh	127	28.04.2013 19:53	-a–
FP1101	TXT	1 566	17.11.2014 09:20	-a–
FP1101P0	DEC	4 438	17.11.2014 10:26	-a–
FP1101P1	DEC	4 438	17.11.2014 10:26	-a–
FP1124P0	DEC	2 794	17.11.2014 10:26	-a–
FP1124P1	DEC	2 154	17.11.2014 10:26	-a–
stdio_fp11	h	11 472	28.05.2013 12:46	-a–

0 k / 45 k in 0 / 18 files

fp12_8x8\SDK_Workspace\edkdsp_cc\a>

Figure 41: Select compiled binaries and download them to the Kintex7 FPGA by the TFTP client.


Tftpd64 by Ph. Jounin

Current Directory: C:\pf\Tftpd64

Server interfaces: 127.0.0.1 Software L

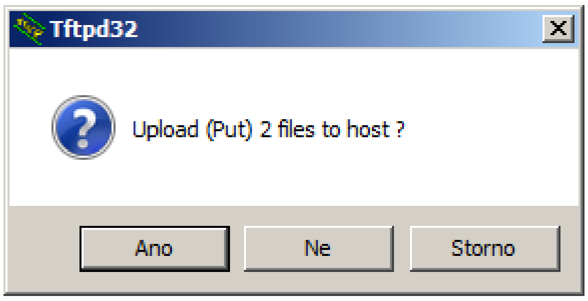
Host: 192.168.8.10 Port: 69

Local File: C:\VM_07\d_34_7k\d_7k32

Remote File:

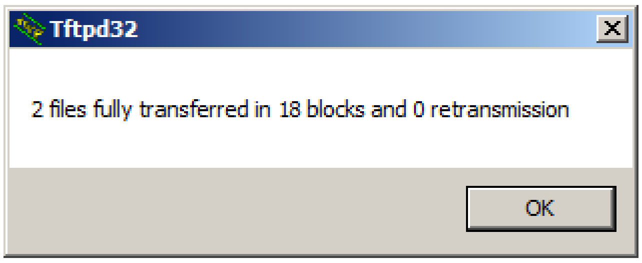
Block Size: Default

Figure 42: Drag and drop the 2 binary program files to upload them to the Kintex7 file system.


Tftpd32

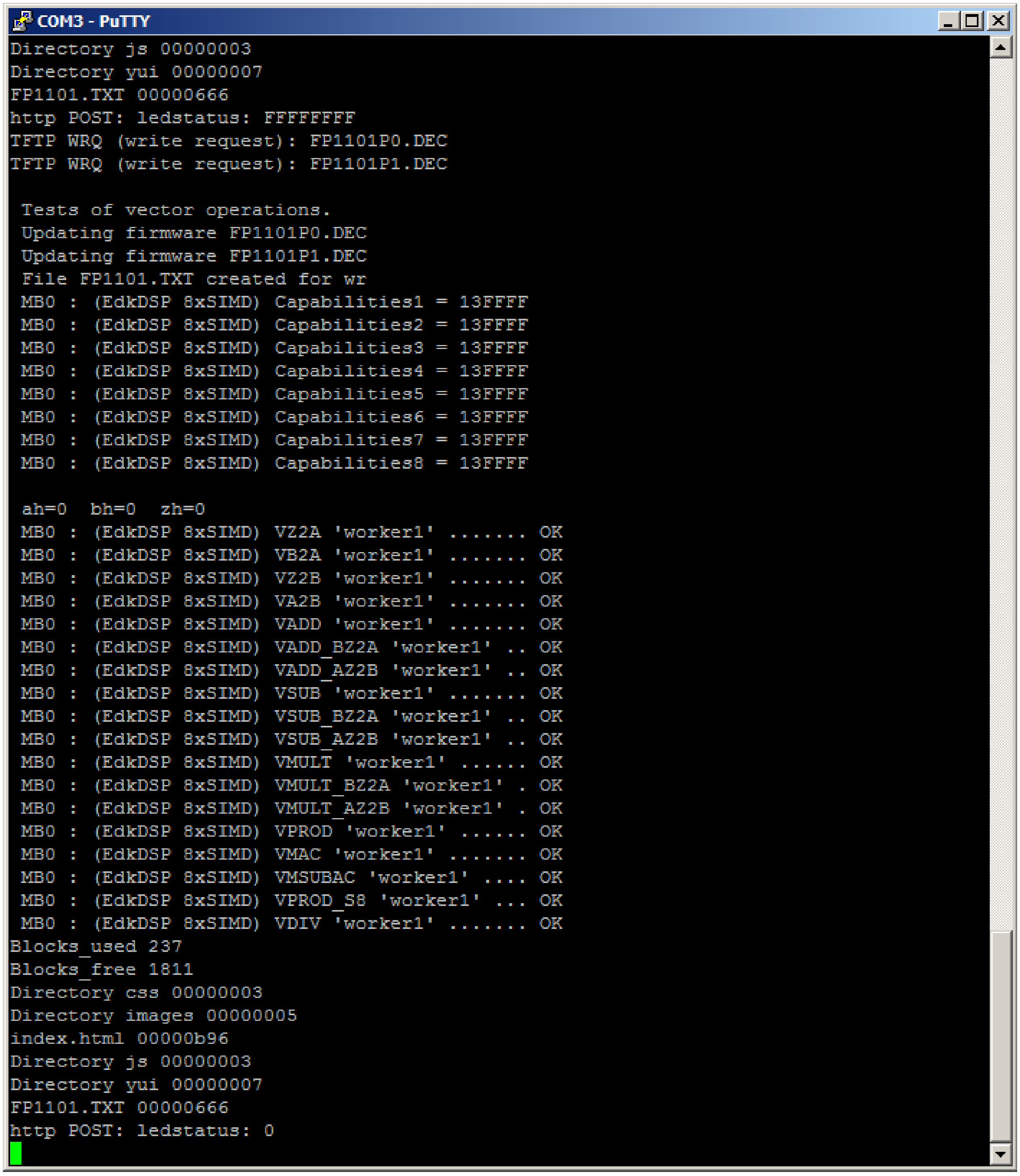
Upload (Put) 2 files to host ?

Figure 43: Confirm Ano (yes in Czech…)


Tftpd32

2 files fully transferred in 18 blocks and 0 retransmission

Figure 44: The TFTP server is indicating number of blocks uploaded to Kintex7 file system.



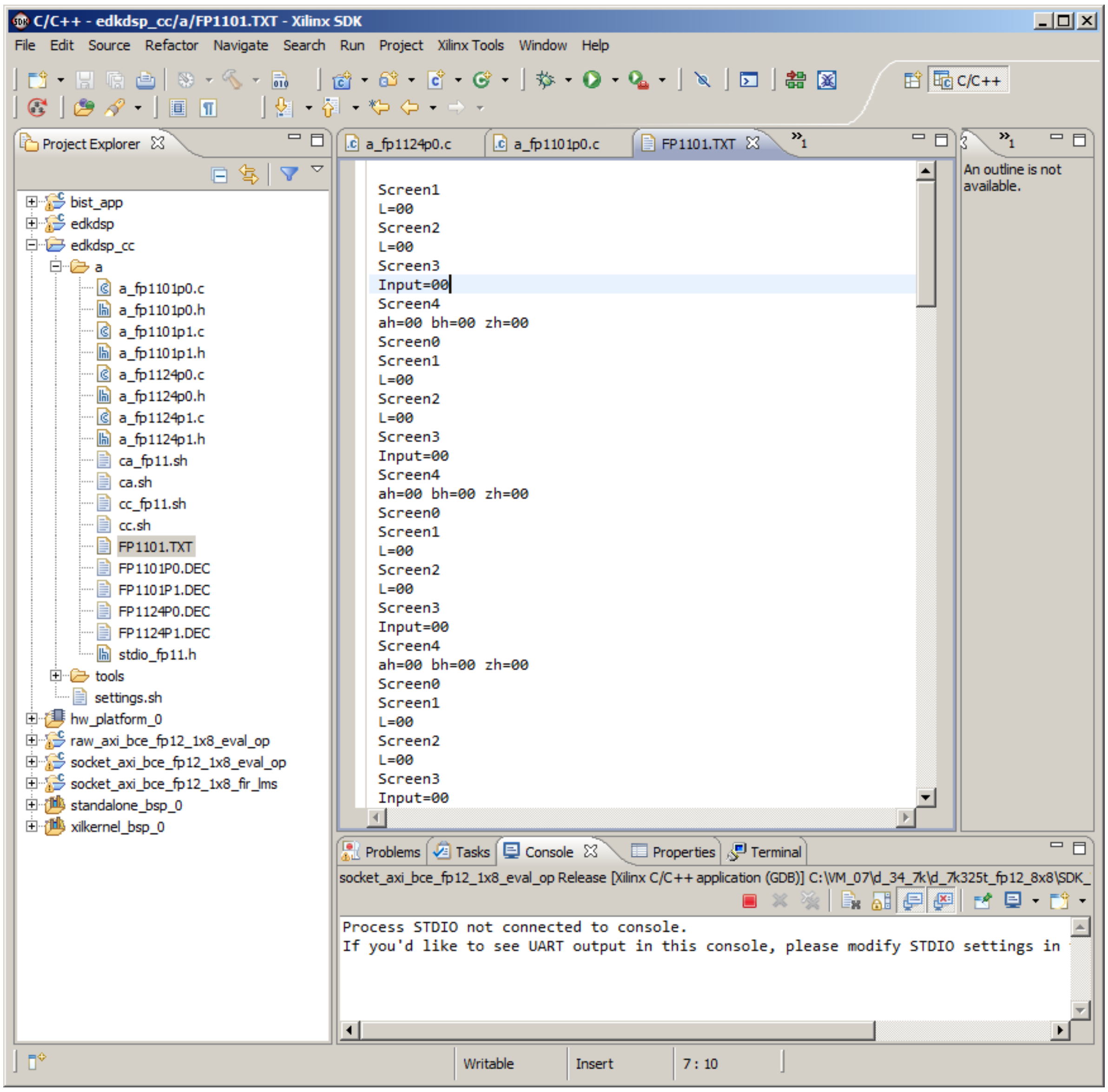
```
Directory js 00000003
Directory yui 00000007
FP1101.TXT 00000666
http POST: ledstatus: FFFFFFFF
TFTP WRQ (write request): FP1101P0.DEC
TFTP WRQ (write request): FP1101P1.DEC

 Tests of vector operations.
 Updating firmware FP1101P0.DEC
 Updating firmware FP1101P1.DEC
 File FP1101.TXT created for wr
 MB0 : (EdkDSP 8xSIMD) Capabilities1 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities2 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities3 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities4 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities5 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities6 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities7 = 13FFFF
 MB0 : (EdkDSP 8xSIMD) Capabilities8 = 13FFFF

 ah=0  bh=0  zh=0
 MB0 : (EdkDSP 8xSIMD) VZ2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VB2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VZ2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VA2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VADD_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VSUB_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VMULT 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMULT_BZ2A 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VMULT_AZ2B 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VPROD 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMAC 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VMSUBAC 'worker1' .... OK
 MB0 : (EdkDSP 8xSIMD) VPROD_S8 'worker1' ... OK
 MB0 : (EdkDSP 8xSIMD) VDIV 'worker1' ....... OK
Blocks_used 237
Blocks_free 1811
Directory css 00000003
Directory images 00000005
index.html 00000b96
Directory js 00000003
Directory yui 00000007
FP1101.TXT 00000666
http POST: ledstatus: 0
```

Figure 45: The TFTP server is indicating on the terminal that the 2 files have been received.

The TFTP server running on the Kintex7 MicroBlaze is informing about the uploaded firmware files. See the last two lines in Figure 45.

Start second test of the EdkDSP accelerator by clicking on the Toggle LEDs button in the www browser user interface. Firmware files have been found, and firmware of the tested EdkDSP accelerator have been updated. Tests have been performed and the log file FP1100.TXT stored in the Kintex7 RAM based file system. See Figure 46.

Refresh the edkdsp_cc/a directory and see the file FP1100.TXT in the SDK. Text Input=00 is now written to the file. This corresponds to the modified C source for the EdkDSP accelerator.



```
Screen1
L=00
Screen2
L=00
Screen3
Input=00
Screen4
ah=00 bh=00 zh=00
Screen0
Screen1
L=00
Screen2
L=00
Screen3
Input=00
Screen4
ah=00 bh=00 zh=00
Screen0
Screen1
L=00
Screen2
L=00
Screen3
Input=00
Screen4
ah=00 bh=00 zh=00
Screen0
Screen1
L=00
Screen2
L=00
Screen3
Input=00
```

socket_axi_bce_fp12_1x8_eval_op Release [Xilinx C/C++ application (GDB)] C:\VM_07\d_34_7k\d_7k325t_fp12_8x8\SDK_

```
Process STDIO not connected to console.
If you'd like to see UART output in this console, please modify STDIO settings in
```

Figure 46: The resulting FP1101.TXT is bigger due to the longer output text: (I=00) replaced by (Input=00).

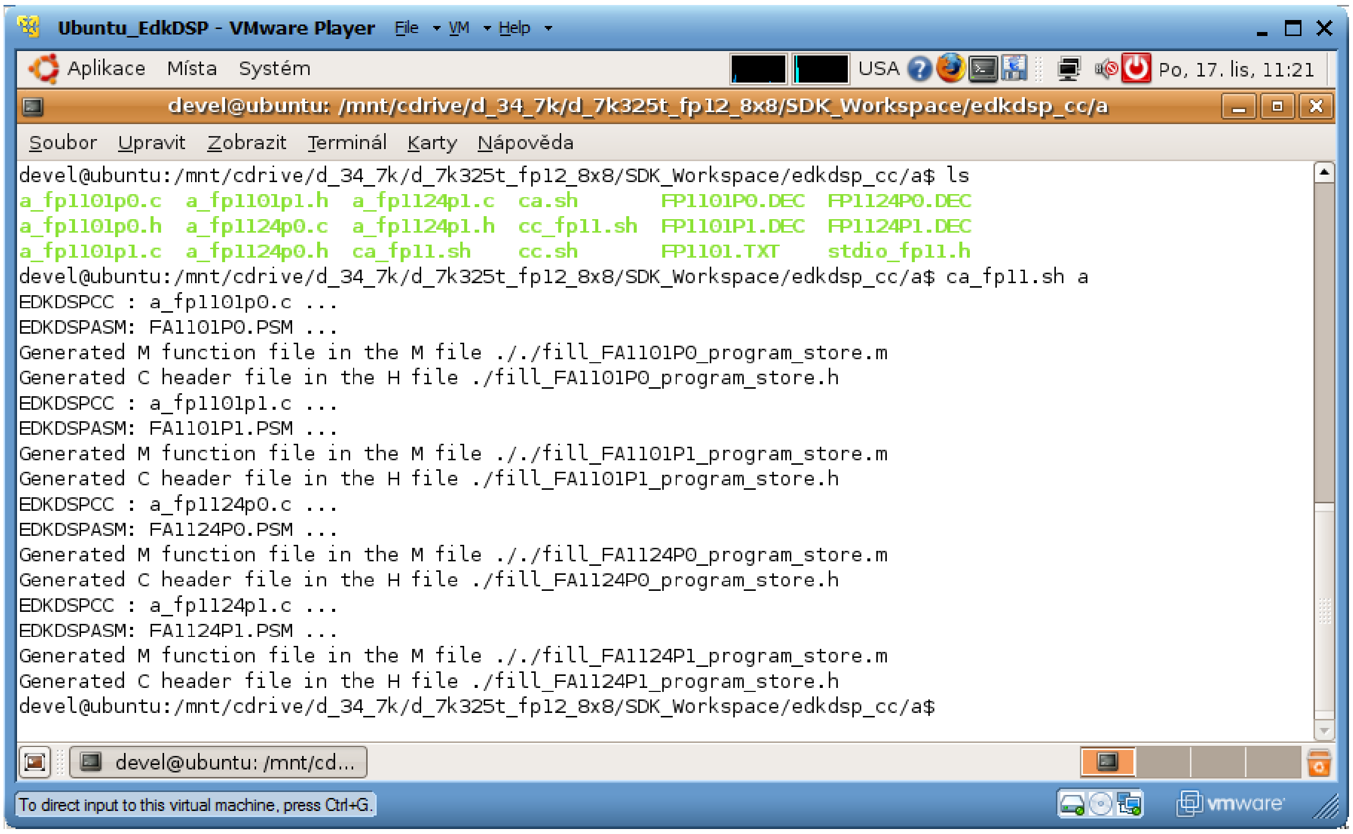
We have demonstrated the process of compilation, download of files from the Kintex7 chip to PC and upload of compiled firmware from PC to the Kintex7 and its EdkDSP accelerators.

Close the browser application and stop the application on the Kintex7 MicroBlaze processor.

3.7 Use of the C compiler for the EdkDSP firmware witout Ethernet

This section is describing the use of the UTIA EdkDSP C compiler to recompile the firmware for the PicoBlaze6 controller present in each of the eight (8xSIMD) EdkDSP accelerators in the KC705 board for simple application without internet connectivity. The edkdsp project in the SDK project explorer will be used as an example.

The firmware C source code examples can be compiled by the script ca_fp11.sh with parameter a.
Type in the Ubuntu terminal (See Figure 47):
$ ca_fp11.sh a


Ubuntu_EdkDSP - VMware Player
File VM Help
Aplikace Místa Systém
USA
Po, 17. lis, 11:21
devel@ubuntu: /mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc/a
Soubor Upravit Zobrazit Terminál Karty Nápověda
devel@ubuntu: /mnt/cd...
To direct input to this virtual machine, press Ctrl+G.
vmware

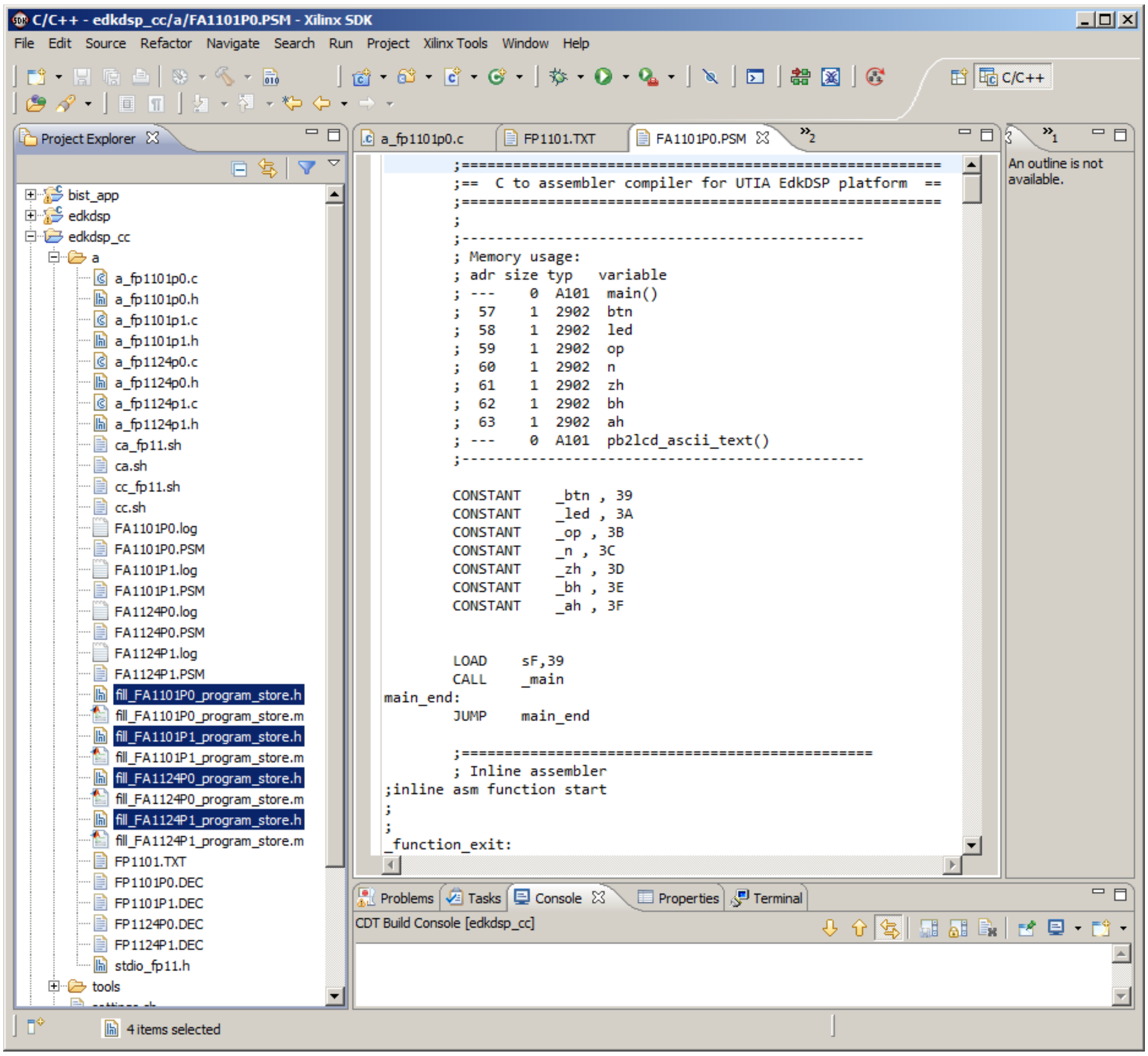
```
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc/a$ ls
a_fp1101p0.c  a_fp1101p1.h  a_fp1124p1.c  ca.sh       FP1101P0.DEC  FP1124P0.DEC
a_fp1101p0.h  a_fp1124p0.c  a_fp1124p1.h  cc_fp11.sh  FP1101P1.DEC  FP1124P1.DEC
a_fp1101p1.c  a_fp1124p0.h  ca_fp11.sh    cc.sh       FP1101.TXT    stdio_fp11.h
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc/a$ ca_fp11.sh a
EDKDSPCC : a_fp1101p0.c ...
EDKDSPASM: FA1101P0.PSM ...
Generated M function file in the M file ././fill_FA1101P0_program_store.m
Generated C header file in the H file ./fill_FA1101P0_program_store.h
EDKDSPCC : a_fp1101p1.c ...
EDKDSPASM: FA1101P1.PSM ...
Generated M function file in the M file ././fill_FA1101P1_program_store.m
Generated C header file in the H file ./fill_FA1101P1_program_store.h
EDKDSPCC : a_fp1124p0.c ...
EDKDSPASM: FA1124P0.PSM ...
Generated M function file in the M file ././fill_FA1124P0_program_store.m
Generated C header file in the H file ./fill_FA1124P0_program_store.h
EDKDSPCC : a_fp1124p1.c ...
EDKDSPASM: FA1124P1.PSM ...
Generated M function file in the M file ././fill_FA1124P1_program_store.m
Generated C header file in the H file ./fill_FA1124P1_program_store.h
devel@ubuntu:/mnt/cdrive/d_34_7k/d_7k325t_fp12_8x8/SDK_Workspace/edkdsp_cc/a$
```

Figure 47: Compile the C source code for the accelerator by the EDKDSPCC compiler with the edkdspasm assembler. It will create the assembler source code and firmware binary in format of C .h header files. These headers can be used for inclusion into the edkdsp demo project (without the TFTP file server).

This will compile and assemble all four C firmware programs to header files with the firmware binary code:

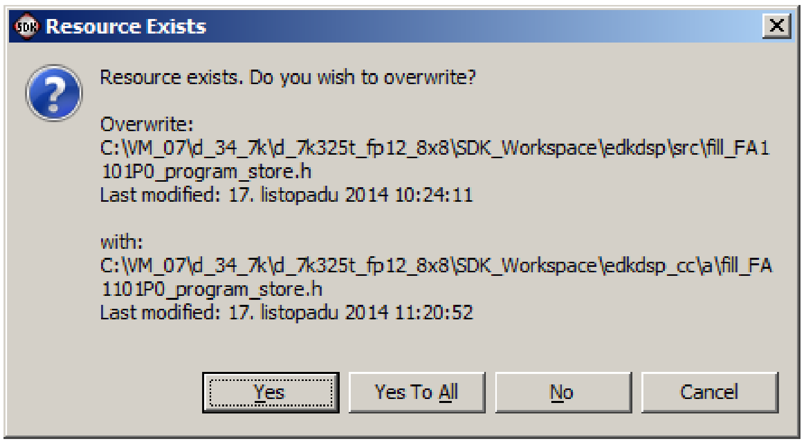
a_fp1101p0.c	is compiled to fill_FA1101P0_program_store.h
a_fp1101p1.c	is compiled to fill_FA1101P1_program_store.h
a_fp1124p0.c	is compiled to fill_FA1124P0_program_store.h
a_fp1124p1.c	is compiled to fill_FA1124P0_program_store.h

Copy and paste the compiled headers into the src directory of the MicroBlaze project “edkdsp” of the SDK 2013.4. See Figure 48 - Figure 50.



```
        ;========================================================
        ;==  C to assembler compiler for UTIA EdkDSP platform  ==
        ;========================================================
        ;
        ;----------------------------------------------
        ; Memory usage:
        ; adr size typ   variable
        ; ---     0  A101  main()
        ;  57     1  2902  btn
        ;  58     1  2902  led
        ;  59     1  2902  op
        ;  60     1  2902  n
        ;  61     1  2902  zh
        ;  62     1  2902  bh
        ;  63     1  2902  ah
        ; ---     0  A101  pb2lcd_ascii_text()
        ;----------------------------------------------

        CONSTANT    _btn , 39
        CONSTANT    _led , 3A
        CONSTANT    _op , 3B
        CONSTANT    _n , 3C
        CONSTANT    _zh , 3D
        CONSTANT    _bh , 3E
        CONSTANT    _ah , 3F


        LOAD    sF,39
        CALL    _main
main_end:
        JUMP    main_end

        ;================================================
        ; Inline assembler
;inline asm function start
;
;
_function_exit:
```

Figure 48: *Select firmware header files and Ctrl-C Ctrl-V them to the edkdsp/src directory.*

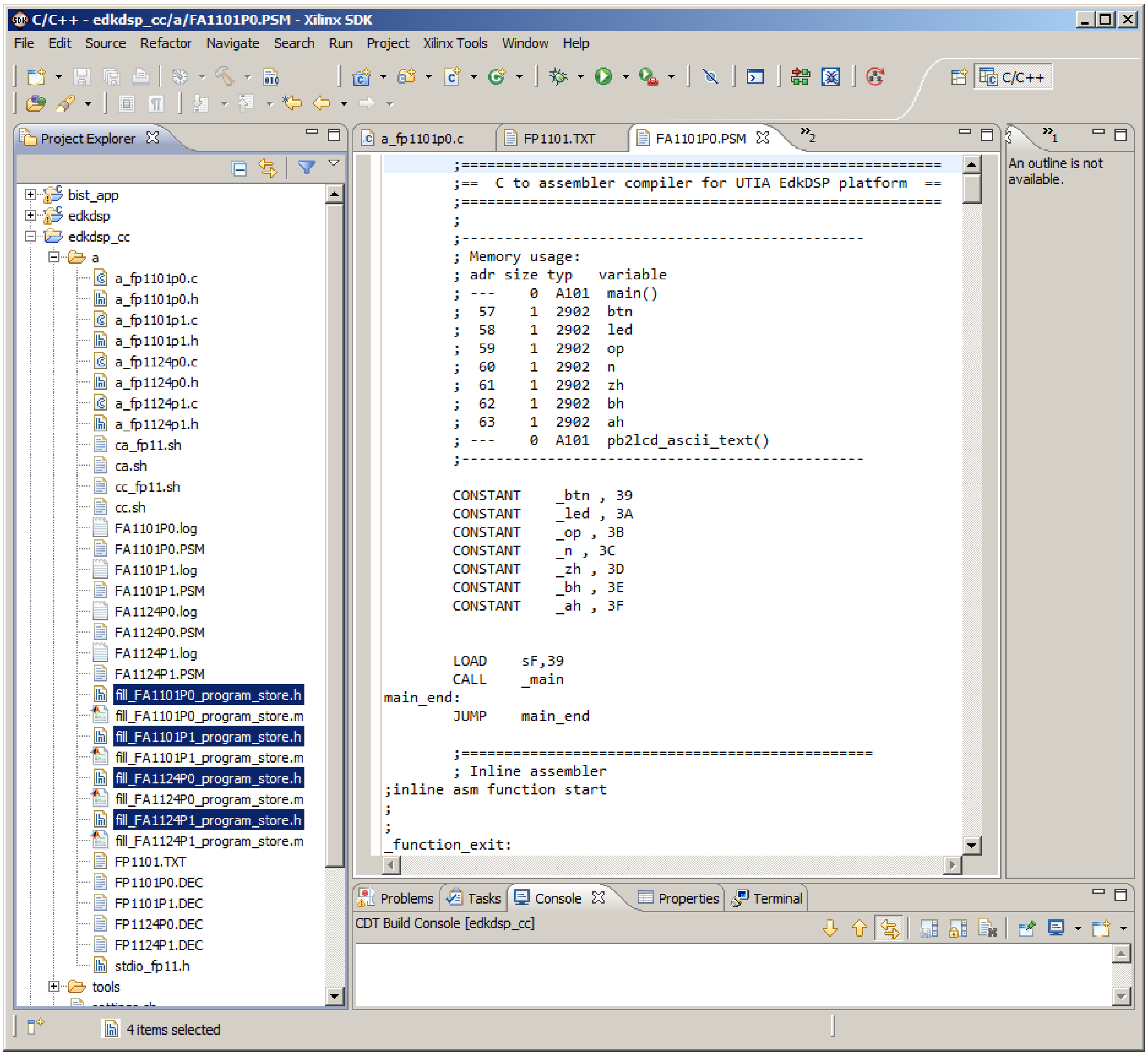


Resource exists. Do you wish to overwrite?

Overwrite:
C:\VM_07\d_34_7k\d_7k325t_fp12_8x8\SDK_Workspace\edkdsp\src\fill_FA1101P0_program_store.h
Last modified: 17. listopadu 2014 10:24:11

with:
C:\VM_07\d_34_7k\d_7k325t_fp12_8x8\SDK_Workspace\edkdsp_cc\a\fill_FA1101P0_program_store.h
Last modified: 17. listopadu 2014 11:20:52

Figure 49: *Confirm to overwrite multiple files*


C/C++ - edkdsp_cc/a/FA1101P0.PSM - Xilinx SDK
File Edit Source Refactor Navigate Search Run Project Xilinx Tools Window Help
Project Explorer
bist_app
edkdsp
edkdsp_cc
a
a_fp1101p0.c
a_fp1101p0.h
a_fp1101p1.c
a_fp1101p1.h
a_fp1124p0.c
a_fp1124p0.h
a_fp1124p1.c
a_fp1124p1.h
ca_fp11.sh
ca.sh
cc_fp11.sh
cc.sh
FA1101P0.log
FA1101P0.PSM
FA1101P1.log
FA1101P1.PSM
FA1124P0.log
FA1124P0.PSM
FA1124P1.log
FA1124P1.PSM
fill_FA1101P0_program_store.h
fill_FA1101P0_program_store.m
fill_FA1101P1_program_store.h
fill_FA1101P1_program_store.m
fill_FA1124P0_program_store.h
fill_FA1124P0_program_store.m
fill_FA1124P1_program_store.h
fill_FA1124P1_program_store.m
FP1101.TXT
FP1101P0.DEC
FP1101P1.DEC
FP1124P0.DEC
FP1124P1.DEC
stdio_fp11.h
tools
a_fp1101p0.c
FP1101.TXT
FA1101P0.PSM
An outline is not available.
Problems Tasks Console Properties Terminal
CDT Build Console [edkdsp_cc]
4 items selected

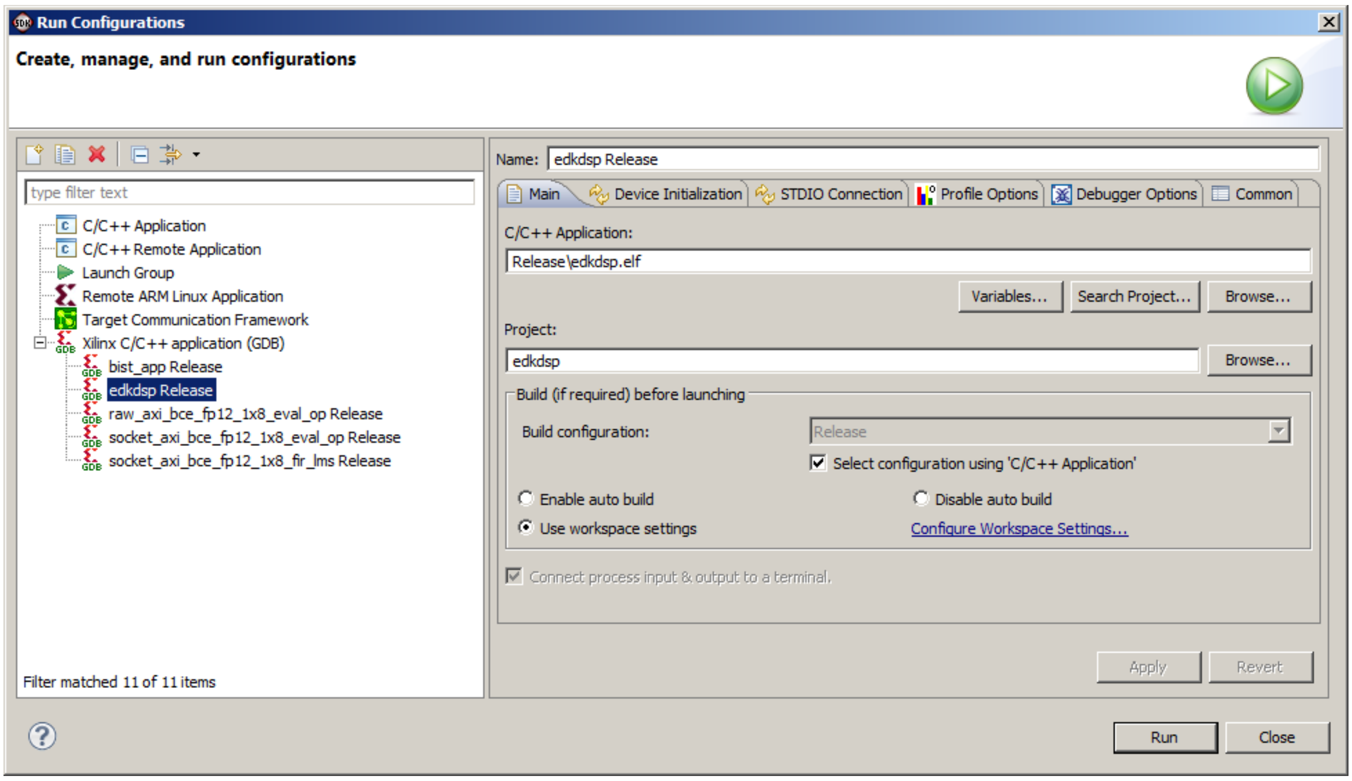
```
        ;=========================================================
        ;==  C to assembler compiler for UTIA EdkDSP platform  ==
        ;=========================================================
        ;
        ;---------------------------------------------
        ; Memory usage:
        ; adr size typ   variable
        ; ---     0  A101  main()
        ;  57     1  2902  btn
        ;  58     1  2902  led
        ;  59     1  2902  op
        ;  60     1  2902  n
        ;  61     1  2902  zh
        ;  62     1  2902  bh
        ;  63     1  2902  ah
        ; ---     0  A101  pb2lcd_ascii_text()
        ;---------------------------------------------

        CONSTANT    _btn , 39
        CONSTANT    _led , 3A
        CONSTANT    _op , 3B
        CONSTANT    _n , 3C
        CONSTANT    _zh , 3D
        CONSTANT    _bh , 3E
        CONSTANT    _ah , 3F


        LOAD    sF,39
        CALL    _main
main_end:
        JUMP    main_end

        ;==============================================
        ; Inline assembler
;inline asm function start
;
;
_function_exit:
```

Figure 50: See the updated edkdsp/src directory and section of the Microblaze source code, where the recompiled modified firmware is updated and EdkDSP accelerators are programmed.

Notice also the listing of the firmware in the assembler in Figure 48.
Figure 50 is presenting the firmware update section of the C code in the Microblaze edkdsp project.

In SDK, recompile the edkdsp project, to reflect the change of the firmware in header files.

To test new firmware, download the bitstream, and run the recompiled edkdsp.elf application. See Figure 51.


Run Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Remote Application
- Launch Group
- Remote ARM Linux Application
- Target Communication Framework
- Xilinx C/C++ application (GDB)
  - bist_app Release
  - edkdsp Release
  - raw_axi_bce_fp12_1x8_eval_op Release
  - socket_axi_bce_fp12_1x8_eval_op Release
  - socket_axi_bce_fp12_1x8_fir_lms Release

Filter matched 11 of 11 items

Name: edkdsp Release

Main | Device Initialization | STDIO Connection | Profile Options | Debugger Options | Common

C/C++ Application: Release\edkdsp.elf

Variables... Search Project... Browse...

Project: edkdsp

Browse...

Build (if required) before launching

Build configuration: Release

[✓] Select configuration using 'C/C++ Application'

Enable auto build [✓ Use workspace settings] Disable auto build

Configure Workspace Settings...

[✓] Connect process input & output to a terminal.

Apply Revert

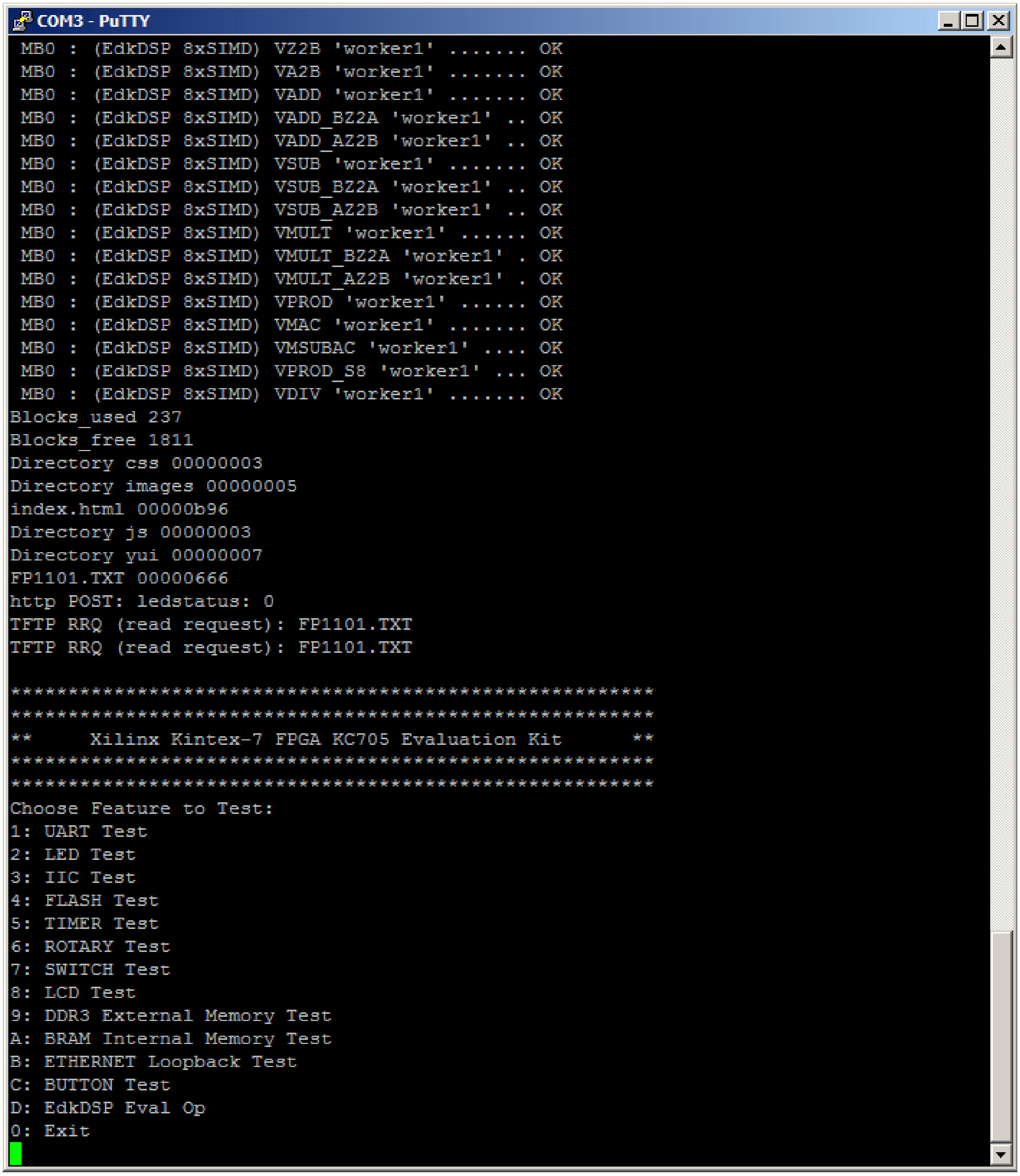
Run Close

Figure 51: Recompile edkdsp project, download the .bit file and run the edkdsp.elf on Kintex.

Figure 52 is presenting the initial menu of the edkdsp application.

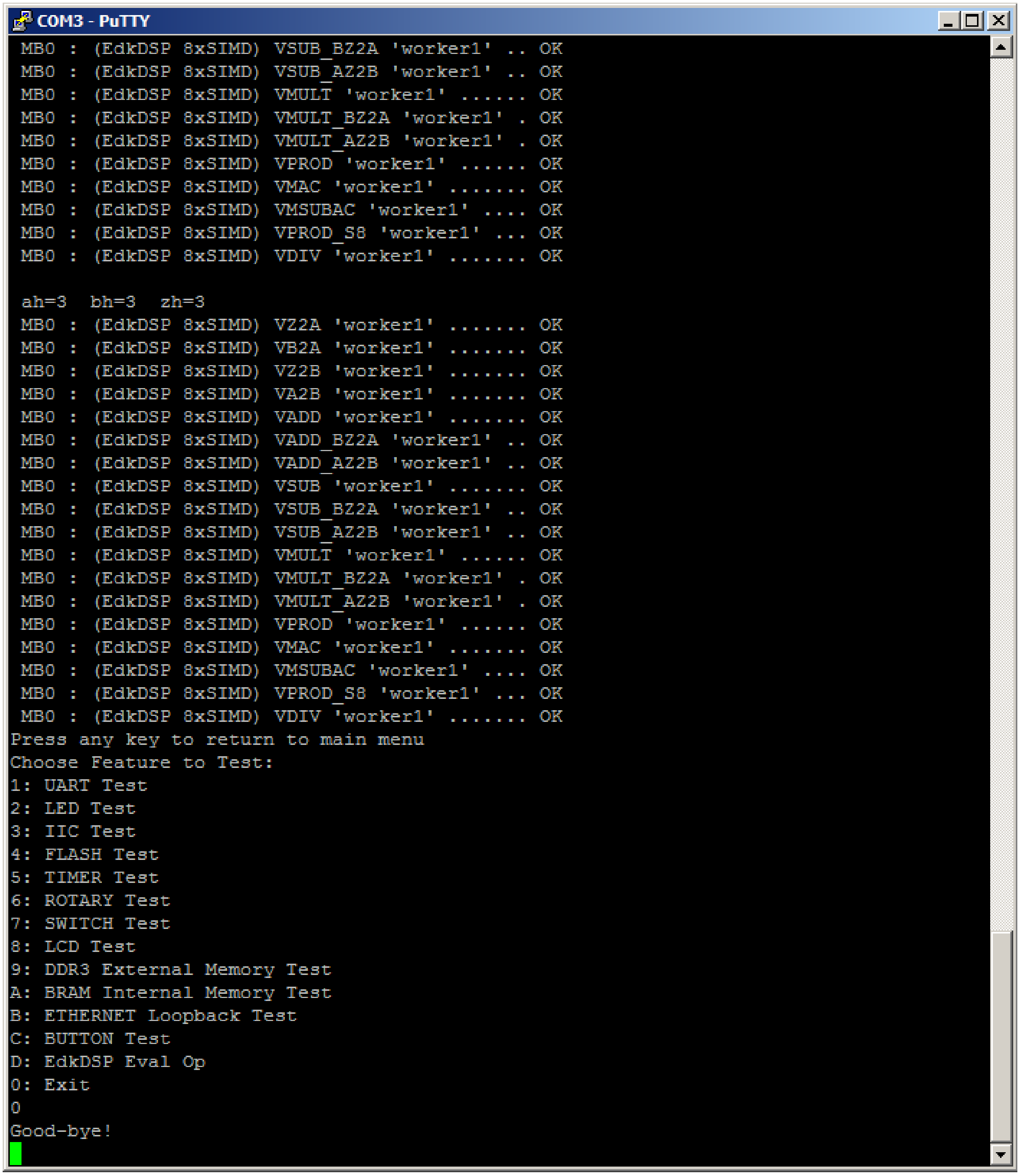
Type D to select test of the EdkDSP operations.
Figure 53 is presenting results of the test of the EdkDSP accelerator with modified firmware.
Type 0 to wxit from the edkdsp simple menu.

Close the debug session from SDK console (the X icon).



```
 MB0 : (EdkDSP 8xSIMD) VZ2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VA2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VADD_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VSUB_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VMULT 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMULT_BZ2A 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VMULT_AZ2B 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VPROD 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMAC 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VMSUBAC 'worker1' .... OK
 MB0 : (EdkDSP 8xSIMD) VPROD_S8 'worker1' ... OK
 MB0 : (EdkDSP 8xSIMD) VDIV 'worker1' ....... OK
Blocks_used 237
Blocks_free 1811
Directory css 00000003
Directory images 00000005
index.html 00000b96
Directory js 00000003
Directory yui 00000007
FP1101.TXT 00000666
http POST: ledstatus: 0
TFTP RRQ (read request): FP1101.TXT
TFTP RRQ (read request): FP1101.TXT

********************************************************
********************************************************
**     Xilinx Kintex-7 FPGA KC705 Evaluation Kit      **
********************************************************
********************************************************
Choose Feature to Test:
1: UART Test
2: LED Test
3: IIC Test
4: FLASH Test
5: TIMER Test
6: ROTARY Test
7: SWITCH Test
8: LCD Test
9: DDR3 External Memory Test
A: BRAM Internal Memory Test
B: ETHERNET Loopback Test
C: BUTTON Test
D: EdkDSP Eval Op
0: Exit
```

Figure 52: Test the EdkDSP accelerator with the new firmware from the menu (type D).


COM3 - PuTTY

```
 MB0 : (EdkDSP 8xSIMD) VSUB_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VMULT 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMULT_BZ2A 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VMULT_AZ2B 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VPROD 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMAC 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VMSUBAC 'worker1' .... OK
 MB0 : (EdkDSP 8xSIMD) VPROD_S8 'worker1' ... OK
 MB0 : (EdkDSP 8xSIMD) VDIV 'worker1' ....... OK

 ah=3  bh=3  zh=3
 MB0 : (EdkDSP 8xSIMD) VZ2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VB2A 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VZ2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VA2B 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VADD_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VADD_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VSUB_BZ2A 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VSUB_AZ2B 'worker1' .. OK
 MB0 : (EdkDSP 8xSIMD) VMULT 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMULT_BZ2A 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VMULT_AZ2B 'worker1' . OK
 MB0 : (EdkDSP 8xSIMD) VPROD 'worker1' ...... OK
 MB0 : (EdkDSP 8xSIMD) VMAC 'worker1' ....... OK
 MB0 : (EdkDSP 8xSIMD) VMSUBAC 'worker1' .... OK
 MB0 : (EdkDSP 8xSIMD) VPROD_S8 'worker1' ... OK
 MB0 : (EdkDSP 8xSIMD) VDIV 'worker1' ....... OK
Press any key to return to main menu
Choose Feature to Test:
1: UART Test
2: LED Test
3: IIC Test
4: FLASH Test
5: TIMER Test
6: ROTARY Test
7: SWITCH Test
8: LCD Test
9: DDR3 External Memory Test
A: BRAM Internal Memory Test
B: ETHERNET Loopback Test
C: BUTTON Test
D: EdkDSP Eval Op
0: Exit
0
Good-bye!
```

Figure 53: See the result of test of all basic vector operations performed on the EdkDSP accelerator with the recompiled firmware. Results remain identical with the Microblaze reference.

4. References

[1] KC705 Evaluation Board for the Kintex-7 FPGA User Guide UG810 (v1.5), July 11, 2014.
http://www.xilinx.com/support/documentation/boards_and_kits/kc705/ug810_KC705_Eval_Bd.pdf

[2] KC705 Built-In Self Test Flash Application, December 2013.
http://www.xilinx.com/support/documentation/boards_and_kits/kc705/2013_4/xtp195-kc705-bist-c-2013-4.pdf

[3] LightWeight IP (lwIP) Application Examples, Author: Anirudha Sarangi and Stephen MacMahon; XAPP1026 (v3.2); October 28, 2012.
http://www.xilinx.com/support/documentation/application_notes/xapp1026.pdf

[4] LightWeight IP Application Examples. Author: Anirudha Sarangi, Stephen MacMahon, and Upender Cherukupaly, XAPP1026 (v5.0) October 24, 2014.
http://www.xilinx.com/support/documentation/application_notes/xapp1026.pdf

[5] PicoBlaze 8-bit Embedded Microcontroller User Guide for Extended Spartan 3 and Virtex5 FPGAs; Introducing PicoBlaze for Spartan-6, Virtex-6, and 7 Series FPGAs.
UG129 June 22, 2011.
http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf

[6] Artemis JU project Almarvi “Algorithms, Design Methods, and Many-Core Execution Platform for Low-Power Massive Data-Rate Video and Image Processing”, project number ENIAC JU 621439.
http://www.almarvi.eu
http://sp.utia.cas.cz/index.php?ids=projects/almarvi

5. Evaluation version of Vivado 2013.4 Kintex7 designs

The enclosed **Evaluation version of precompiled Vivado 2013.4 Kintex7 designs with evaluation versions of UTIA (8xSIMD) EdkDSP accelerator cores** can be downloaded from UTIA www pages free of charge and used for evaluation together with the eight UTIA (8xSIMD) EdkDSP accelerators.

The evaluation package includes one DVD or the www download package with these deliverables:

8 precompiled designs with UTIA (8xSIMD) EdkDSP accelerators for Xilinx Kintex7 KC705 board [1], [2] compiled in Xilinx Vivado 2013.4. The UTIA (8xSIMD) EdkDSP accelerators are compiled with HW limit on number of vector operations. The termination of the evaluation license is reported in advance by the demonstrator on the terminal.

The evaluation package includes SDK 2013.4 SW projects with source code for MicroBlaze processor. SW projects support the family of UTIA (8xSIMD) EdkDSP accelerators for the Xilinx KC705 board [1], [2].

The evaluation package includes this compiled library:

libwal.a	EdkDSP api (SDK 2013.4, MicroBlaze) for EdkDSP accelerators on KC705 board.
libmfsimage.a	The library with file system supporting simple www server GUI.

The library **libwal.a** has no time restriction. The evaluation license is provided by UTIA only for the use with the family of UTIA EdkDSP accelerators designed for the Xilinx KC705 board. Source code of this library is owned by UTIA and it is not provided in this evaluation package.

The evaluation package includes these binary applications for Ubuntu:

edkdsppp	EdkDSP C pre-processor binary for Ubuntu (x86 PC) under the VMware Player.
edkdspcc	EdkDSP C compiler binary for Ubuntu (x86 PC) under the VMware Player.
edkdspasm	EdkDSP ASM compiler binary for Ubuntu (x86 PC) under the VMware Player.
edkdsppsm	EdkDSP ASM compiler binary for Ubuntu (x86 PC) under the VMware Player.

These binary applications have no time restriction. The user of the evaluation package has license from UTIA to use these utilities for compilation of the firmware for the Xilinx PicoBlaze6 processor inside of the UTIA EdkDSP accelerators in the 8 precompiled designs for the Xilinx KC705 board. The source code of these compilers is owned by UTIA and it is not provided in the evaluation package.

The evaluation package includes demonstration firmware in C source code for the Xilinx PicoBlaze6 processor for the family of UTIA EdkDSP accelerators for the Xilinx KC705 board.

The evaluation package also includes compiled versions of this firmware in form of header files .h. These compiled firmware files can be used for initial test of the UTIA EdkDSP accelerators on the Xilinx KC705 board without the need to install the UTIA compiler binaries and the Ubuntu (x86 PC) OS image under the VMware Player.

On email request to kadlec@utia.cas.cz , UTIA will send 2 DVD CDs (8GB) with the Ubuntu (x86 PC) image for the VMware Player free of charge.

6. Release version of Vivado 2013.4 Kintex7 designs for Almarvi project partners

The release version of Vivado 2013.4 Kintex7 designs with evaluation versions of UTIA (8xSIMD) EdkDSP accelerator cores for Almarvi [6] project partners can be ordered from UTIA AV CR, v.v.i., by email request for quotation to kadlec@utia.cas.cz. UTIA will provide quotation by email. After the confirmed order received by email to kadlec@utia.cas.cz, UTIA AV CR, v.v.i. will deliver (by standard mail to the Almarvi project partners) a printed version of this application note together with 3 DVDs with deliverables described in this section. UTIA AV CR, v.v.i., will also send to the Almarvi project partner (by email) and by the standard mail the invoice for:

Release version of Vivado 2013.4 Kintex7 designs	
with evaluation versions of UTIA (8xSIMD) EdkDSP accelerator cores	
for Almarvi [6] project partners (without VAT)	**0,00 Eur**

The package includes this application note and the EdkDSP DVD with these deliverables:

8 precompiled designs with UTIA (8xSIMD) EdkDSP accelerators for Xilinx KC705 board, compiled in Xilinx Vivado 2013.4. The UTIA (8xSIMD) EdkDSP accelerators are compiled with HW limit on number of vector operations. The termination of the evaluation license is reported in advance by the demonstrator on the terminal.

The Release version of Vivado 2013.4 Kintex7 designs with evaluation versions of UTIA (8xSIMD) EdkDSP accelerator cores for Almarvi [6] project partners include all 8 Vivado 2013.4 design projects and the evaluation versions of the UTIA (8xSIMD) EdkDSP accelerators provided in form of netlisted IP cores generated in Xilinx VIVADO 2013.4:

bce_fp11_1x8_0_axiw_v1_10_b
bce_fp11_1x8_0_axiw_v1_20_b
bce_fp11_1x8_0_axiw_v1_30_b
bce_fp11_1x8_0_axiw_v1_40_b
bce_fp12_1x8_0_axiw_v1_10_b
bce_fp12_1x8_0_axiw_v1_20_b
bce_fp12_1x8_0_axiw_v1_30_b
bce_fp12_1x8_0_axiw_v1_40_b

These evaluation versions of UTIA (8xSIMS) EdkDSP netlist pcores are compiled with an HW limit on number of vector operations. Almarvi [6] project partners have license from UTIA to integrate these evaluation netlists into their own VIVADO 2013.4 designs and to compile them to unlimited number of bit-streams for designs on Xilinx Kintex7 FPGAs. This license has no time restriction. The source code of the evaluation versions of (8xSIMS) EdkDSP accelerators is an IP owned by UTIA and it is not provided in the release package to the Almarvi project partners.

The package for the Almarvi [6] project partners includes the SDK 2013.4 SW projects in source code for MicroBlaze as described in this application note. Projects support the evaluation versions of the UTIA (8xSIMD) EdkDSP accelerators (in the netlist pcore format) for the Xilinx KC705 board.

The package for the Almarvi project partners includes the library:

libwal.a	EdkDSP api (SDK 2013.4, MicroBlaze) for EdkDSP accelerators on KC705 board.
libmfsimage.a	The library with file system supporting simple www server GUI.

The library **libwal.a** has has no time restriction. The evaluation license is provided by UTIA only for the use with the family of UTIA EdkDSP accelerators designed for the Xilinx KC705 board. Source code of this library is owned by UTIA and it is not provided in this evaluation package.

The package for the Almarvi project partners includes these binary applications for Ubuntu:

edkdsppp	EdkDSP C pre-processor binary for Ubuntu (x86 PC) under the VMware Player.
edkdspcc	EdkDSP C compiler binary for Ubuntu (x86 PC) under the VMware Player.
edkdspasm	EdkDSP ASM compiler binary for Ubuntu (x86 PC) under the VMware Player.
edkdsppsm	EdkDSP ASM compiler binary for Ubuntu (x86 PC) under the VMware Player.

These binary applications have no time restriction. The Almarvi project partners have license from UTIA to use these utilities for compilation of the firmware for the Xilinx PicoBlaze6 processor inside of the UTIA EdkDSP accelerators in the 10 precompiled designs for the Xilinx KC705 board. The source code of these binaries is owned by UTIA and it is not provided in the evaluation package.

The package includes demonstration firmware in C source code for the Xilinx PicoBlaze6 processor for the family of UTIA EdkDSP accelerators for the Xilinx KC705 board.

The package also includes compiled versions of this firmware in form of header files .h. These compiled firmware files can be used to evaluate the UTIA EdkDSP accelerators on the Xilinx KC705 board without the need to install the UTIA compiler binaries and the Ubuntu (x86 PC) OS image under the VMware Player.

The release package deliverables also includes two DVDs with the Ubuntu (x86 PC) image for the VMware Player (free of charge). This image is provided to ease the installation of the UTIA EdkDSP C compiler on Windows 7 (32bit or 64bit) in the VMware Player.

Any and all legal disputes that may arise from or in connection with the use, intended use of or license for the software provided hereunder shall be exclusively resolved under the regional jurisdiction relevant for UTIA AV CR, v. v. i. and shall be governed by the law of the Czech Republic.

7. Release version of Vivado 2013.4 Kintex7 designs

The release version of Vivado 2013.4 Kintex7 designs with the release version of the UTIA (8xSIMD) EdkDSP accelerator cores can be ordered from UTIA AV CR, v.v.i., by email request for quotation to kadlec@utia.cas.cz. UTIA will provide quotation by email. After the confirmed order received by email to kadlec@utia.cas.cz, UTIA AV CR, v.v.i. will deliver (by standard mail) to the customer the printed version of this application note together with 3 DVDs with deliverables described in this section. UTIA AV CR, v.v.i., will send to the customer (by email) and by the standard mail the invoice for:

Release version of Vivado 2013.4 Kintex7 designs with the evaluation version of the UTIA (8xSIMD) EdkDSP accelerator cores (without VAT)	**400,00 Eur**

The release package includes this application note and the EdkDSP DVD with these deliverables:

8 precompiled designs with UTIA (8xSIMD) EdkDSP accelerators for Xilinx KC705 board [2], compiled in Xilinx Vivado 2013.4. The UTIA (8xSIMD) EdkDSP accelerators included in these designs are compiled with **no HW limit on number of vector operations.** Therefore, all these precompiled designs of the release package run on KC705 without limitations of the evaluation package.

The release package includes all 8 Vivado 2013.4 design projects. The UTIA (8xSIMD) EdkDSP accelerators are provided in the form of netlist IP cores generated in Xilinx VIVADO 2013.4:

bce_fp11_1x8_0_axiw_v1_10_b
bce_fp11_1x8_0_axiw_v1_20_b
bce_fp11_1x8_0_axiw_v1_30_b
bce_fp11_1x8_0_axiw_v1_40_b
bce_fp12_1x8_0_axiw_v1_10_b
bce_fp12_1x8_0_axiw_v1_20_b
bce_fp12_1x8_0_axiw_v1_30_b
bce_fp12_1x8_0_axiw_v1_40_b

These UTIA (8xSIMS) EdkDSP netlist pcores have **no HW limit on number of vector operations.** The user of the release package has license from UTIA to integrate these netlists into its own VIVADO 2013.4 designs and to compile them to unlimited number of bit-streams. This license has no time restriction. The source code of the (8xSIMS) EdkDSP accelerators is an IP owned by UTIA and it is not provided in the release package to the customer.

The release package includes SDK 2013.4 SW projects in source code for MicroBlaze as described in this application note. Projects support the family of UTIA (8xSIMD) EdkDSP accelerators for Xilinx KC705 board [2].

The release package includes the library:

libwal.a	EdkDSP api (SDK 2013.4, MicroBlaze) for EdkDSP accelerators on KC705 board.
libmfsimage.a	The library with file system supporting simple www server GUI.

The library **libwal.a** has has no time restriction. The evaluation license is provided by UTIA only for the use with the family of UTIA EdkDSP accelerators designed for the Xilinx KC705 board. Source code of this library is owned by UTIA and it is not provided in this release package.

The release package includes these binary applications for Ubuntu:

edkdsppp	EdkDSP C pre-processor binary for Ubuntu (x86 PC) under the VMware Player.
edkdspcc	EdkDSP C compiler binary for Ubuntu (x86 PC) under the VMware Player.
edkdspasm	EdkDSP ASM compiler binary for Ubuntu (x86 PC) under the VMware Player.
edkdsppsm	EdkDSP ASM compiler binary for Ubuntu (x86 PC) under the VMware Player.

These binary applications have no time restriction. The user of the evaluation package has license from UTIA to use these utilities for compilation of the firmware for the Xilinx PicoBlaze6 processor inside of the UTIA EdkDSP accelerators in the 8 precompiled designs for the Xilinx KC705 board. The source code of these compilers is owned by UTIA and it is not provided in the release package.

The release package includes demonstration firmware in C source code for the Xilinx PicoBlaze6 processor for the family of UTIA EdkDSP accelerators for the Xilinx KC705 board.

The release package also includes compiled versions of this firmware in form of header files .h. These compiled firmware files can be downloaded into the UTIA EdkDSP accelerators for the Xilinx KC705 board without the need to install UTIA compiler binaries and the Ubuntu (x86 PC) OS under the VMware Player.

The release package deliverables also includes two DVDs with the Ubuntu (x86 PC) image for the VMware Player (free of charge). This image is provided to ease the installation of the UTIA EdkDSP C compiler on Windows 7 (32bit or 64bit) in the VMware Player.

Any and all legal disputes that may arise from or in connection with the use, intended use of or license for the software provided hereunder shall be exclusively resolved under the regional jurisdiction relevant for UTIA AV CR, v. v. i. and shall be governed by the law of the Czech Republic.

Disclaimer