Application Note

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

Video Processing Demonstrator with Full HD Sensor and 8xSIMD EdkDSP Accelerator IP Core

Trenz Electronic TE0720-2IF Module, TE0701 Carrier

Jiří Kadlec, Zdeněk Pohl, Lukáš Kohout
kadlec@utia.cas.cz , xpohl@utia.cas.cz , kohoutl@utia.cas.cz
phone: +420 2 6605 2216
UTIA AV CR, v.v.i.

Revision history:

Rev.	Date	Author	Description
1	08.03.2017	Jiří Kadlec	Evaluation package for Xilinx SDK 2015.4
2	30.01.2018	Jiří Kadlec	Updated for use of the embedded EdkDSP C compiler

Acknowledgements:

This work has been partially supported by ENIAC JU project THINGS2DO No. 621221 and by related MEYS (CZ NFA) project 7H14007.

Table of Contents

Table of Figures

Table of Tables

1. Evaluation Designs

1.1 Objectives

This application describes use of an evaluation package for the Xilinx SDK 2015.4 SW environment with these standalone, HW accelerated video processing demos designed in the Xilinx SDSoC 2015.4 environment [9]:

- 3 edge detection video processing designs (sh01, sh02, sh03) with separate HW accelerated data paths.
  - These demos present the possibility to define different HW paths by different source C/C++ functions. This is important for covering of the borders lines of the parallel processed parts of the frame.
  - HW accelerators can process parts of the frame with variable width.
  - These demos enable efficient, synchronised parallel execution with ARM C user code.

- 1 motion detection video processing design (md01).
  - This demonstrates the pipelined parallel execution of HW video processing accelerators.
  - HW accelerators work with fixed number of processed micro-lines (1080 micro-lines).

All demos work in parallel with single 8xSIMD EdkDSP run-time reprogrammable floating point accelerator IP developed within the ECSEL THINGS2DO project.

- The ARM Cortex A9 C programs can be compiled for in the Xilinx SDK 2015.4 environment.
- The MicroBlaze C programs can be compiled for in the Xilinx SDK 2015.4 environment.
- The PicoBlaze6 C/ASM programs for the 8xSIMD EdkDSP accelerator IP can be compiled in the embedded EdkDSP C compiler toolchain running on the TE0720-2I development module as PetaLinux 2017.1 applications.
- User can compile and optimize/customize the final system without the need to re-compile the design in Vivado 2015.4 and without the need to use the commercial SDSoC 2015.4 development system.

All demos are designed for the Trenz Electronic TE0701-06 platform [3] with industrial grade Zynq XC7Z020-2I device on System on Module TE0720-2IF [1].

All demonstrated video processing algorithms have been developed, debugged and tested as SW algorithms for ARM in Xilinx SDSoC 2015.4 environment [9]. Selected SW functions have been compiled by the Xilinx SDSoC 2015.4 system level compiler (based on the Xilinx HLS compiler) to the Vivado 2015.4 HW projects, and compiled by Vivado 2015.4 [8] to the bitstreams for Zynq XC7Z020-2I device. The new SDSoC functions controlling the HW accelerators have been exported from the Xilinx SDSoC 2015.4 projects to the standalone Xilinx SDK 2015.4 [8] projects as static .a libraries for the ARM Cortex A9 processor.

Main objectives of this application note:

- To demonstrate how to install, compile, modify and use the enclosed SW projects in the SDK 2015.4 [8].
- To demonstrate the HW accelerated video processing algorithms and the energy per pixel reduction in comparison to the original SW versions.
- To demonstrate parallel execution of predefined video processing HW paths with C user code on ARM.
- To demonstrate HW accelerated video processing working in parallel with the 8xSIMD EdkDSP run-time re-programmable floating point accelerator IP core.

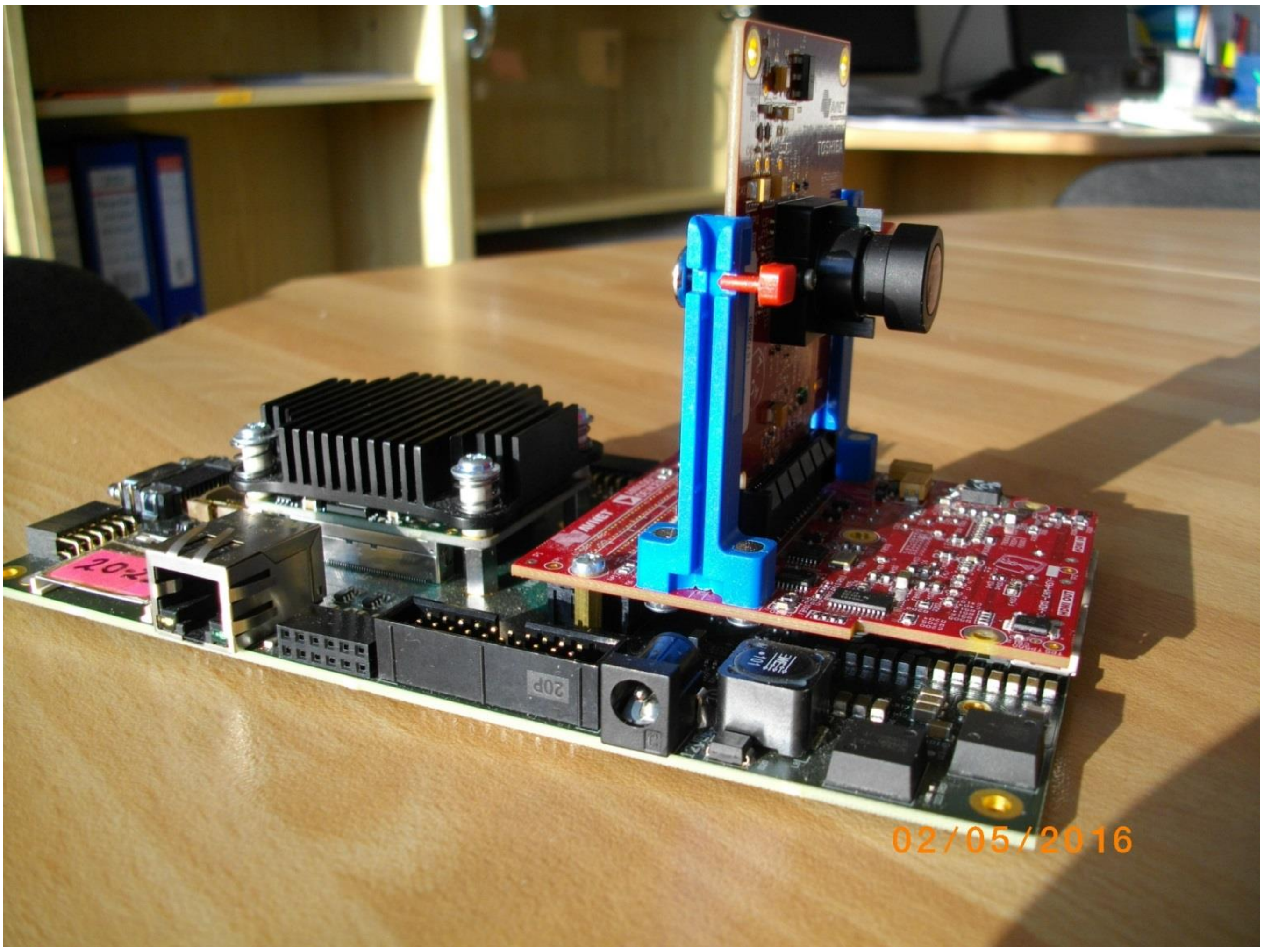


Figure 1: Full HD Toshiba evaluation platform with video processing in HW and EdkDSP accelerator.

Common setup for demos:

- The ARM Cortex A9 processor of Xilinx Zynq device XC7Z020-2I executes standalone C application programs performing initialisation and synchronisation of the HW accelerated video processing chains.
- Enclosed C programs can be modified by the user and recompiled in Xilinx SDK 2015.4.
- Full HD Toshiba sensor with Full HD resolution 1920x1080p60 provides raw colour video data.
- Data are processed in HW into the YCrCb 4:2:2 (16 bit per pixel) format and stored by video DMA (VDMA) controller to input video frame buffers (VFBs) reserved in the DDR3.
- HW DMA controller(s) send data from the input VFBs to the processing HW accelerators in the programmable logic (PL) part of Zynq.
- Another HW DMA controller(s) send processed data from HW to output VFBs in DDR3.
- Second part of the HW VDMA writes processed data to the Full HD HDMI display with Full HD resolution 1920x1080p60.

1.2 Full HD Toshiba platform with EdkDSP Accelerator

This application note describes HW platform performing integration of the runtime reprogrammable 8xSIMD EdkDSP floating point accelerator with edge detection and motion detection video processing of raw data from the Full HD Toshiba colour video sensor with fixed Full HD resolution 1920x1080p60.

- The Xilinx Zynq device xc7z020-2I has two ARM Cortex A9 processors (operating at 766.7 MHz). Memory controller provides interface to DDR3 memory access ports. The Zynq device provides also the programmable logic area used for:
    - UTIA EdkDSP (8xSIMD) floating point accelerator (operating at 120 MHz) connected to Xilinx MicroBlaze 32 bit processor. The MicroBlaze works with 100 MHz clock.
    - Input chain of video processing IPs connects Full HD Toshiba video sensor with Full HD resolution 1920x1080p60 to the input video frame buffers. The sensor pixel clock is set to 148.5 MHz.
    - The input video DMA (VDMA) controller operates at 150 MHz.
    - Part of PL is reserved for the HLS HW accelerators and data movers defined in Xilinx SDSoC 2015.4 environment. These accelerators are controlled from ARM Cortex A9 C programs in SDK 2015.4 C projects. These HLS accelerators are work with 150 MHz clock.
    - The output VDMA controller operates also with the 150 MHz clock.
    - Finally, chain of output video processing IPs connects the output frame buffer with pixel clock 148.5 MHz to the HDMI display with the Full HD resolution 1920x1080p60.

- UTIA EdkDSP is 8xSIMD floating point accelerator IP. It is reprogrammable in runtime by change of firmware of build in PicoBlaze6 8bit controller. It takes the role of an scheduler of vector operations performed in the EdkDSP is 8xSIMD floating point processor data paths. This scheduler is programmed by simple C programs compiled by simple C compiler and assembler, respecting the minimal resources of the PicoBlaze6 controller.
- UTIA EdkDSP is 8xSIMD floating point accelerator IP is controlled by the 32bit MicroBlaze processor. The MicroBlaze processor executes C programs from the DDR3 memory. Complex C algorithms can benefit from execution of selected operations efficiently on the EdkDSP coprocessor connected to the MicroBlaze via local dual ported memories. MicroBlaze C programs can take benefit of overlap of its data communication from DDR3 to the EdkDSP dual-ported memories with parallel computations in the EdkDSP accelerator.
- Platform also includes the video processing chain consisting of IPs controlled by ARM Cortex A9 processor.
- ARM Cortex A9 processor of Xilinx Zynq performs initialisation and synchronisation of the video processing chain. Program and the FPGA bitstream file is downloaded to the board from the Xilinx SDK 2015.4 via USB JTAG. The program is located in the 1GB DDR3 memory on the Zynq SoM. System can be also started directly from the SD card. ARM processor initiates the IP cores in the programmable logic (PL) part of the Zynq. It also initiates the Full HD Toshiba video sensor and the video output to a monitor with Full HD 1920x1080p60 resolution.

Details of the video processing video chain:

- Data are provided by the Full HD Toshiba video sensor with the Full HD 1920x1080p60 resolution. The raw video data received from video sensor are coded in Bayer format.
- Data are processed into the YCrCb 4:2:2 format (this is 16 bit per pixel) and stored by Video DMA (VDMA) to the input video frame buffers (VFBs) defined in the DDR3.
- HW DMA controller(s) send data from/to the VFBs to the processing accelerators. Clock is 150 MHz.

The evaluation designs with HW accelerators have been created from C/C++ functions tested in the Xilinx SDSoC 2015.4 system level compiler.

Projects described in the next section summarise the energy per frame measured on the platform for different HW accelerated image processing algorithms as defined by the individual C projects in these four main configurations:

1. MicroBlaze with EdkDSP coprocessor computes FIR filter in floating point (in parallel to the dedicated video processing accelerator chain).

2. The MicroBlaze with EdkDSP coprocessor computes LMS adaptive filter in floating point (in parallel to the dedicated video processing accelerator chain).
3. The MicroBlaze computes in SW (only with its floating point unit) the FIR or LMS filter (in parallel to the dedicated video processing accelerator chain) but EdkDSP accelerator is not used.
4. The MicroBlaze is present and computes in SW (only with its floating point unit) the FIR or LMS filter. The EdkDSP is not present in the PL at all. The dedicated video processing accelerator chain processes the video from the Full HD Toshiba video sensor.

The figures provide comparison of the energy/pixel consumed by the complete system in case of computation in ARM. C/C++ code was compiled with -O3 optimisation (but without NEON) in the SDSoC 2015.4 environment [9] with no video processing HW accelerators present and the EdkDSP computing FIR filter. It is labled as “SW”.

This “SW” energy/pixel result is compared with the energy/pixel consumed by s complete system in case of HW accelerated video processing and different modes of the EdkDSP accelerator: (1) FIR or (2) LMS on EdkDSP or (3) on MicroBlaze with EdkDSP present but not used of, finally, (4) with computation of FIR or LMS on MicroBlaze with no EdkDSP accelerator HW logic present in the PL part of the Zynq device.

1.3 Introduction to the demos

Edge detection

The edge detection algorithm produces B/W video stream. Edges in each frame are marked as white and remaining part of the figure is set as black. The edges are detected by a Sobel filter. Each pixel is filtered by a 3x3 2D FIR filter. A nonlinear decision on the output of the filter provides decision if the pixel is part of an edge or not. All computations are performed in fixed point. Input to the Sobel filter is the video signal with each pixel converted to the monochrome 8bit format.

Demos **sh01**, **sh02** and **sh03** provide HW accelerated computation of edge detection with 1, 2 or 3 parallel HW data paths. Computation of horizontal border line is resolved in case of sh02 and sh03. All these demos support synchronised parallel execution of user defined C code on ARM while the HW data paths perform accelerated video processing. HW demos use 1, 2 or 3 DMA HW channels as input from DDR3 to 1, 2 or 3 Sobel filters. Another 1, 2 or 3 DMA HW channels support output from Sobel filters to the DDR3. Demos are linked with static libraries libsh01.a, libsh02.a or libsh03.a. Zynq PL resources and the accelerations reached for these HW designs are summarised in sections 1.4, 1.5 and 1.6.

Motion detection

The motion detection algorithm detects and performs visualisation of moving edges. The moving edges are identified by two Sobel filters performing FIR filtering (similar to the above described edge detection) on pixels with identical coordinates but from two subsequent video frames. A difference of these filtered results is computed a noise in that signal is filtered by a Median filter. Resulting signal is used for the nonlinear binary decision if the analysed pixel is part of a moving edge or not. If the pixel is part of a moving edge, it is assigned red colour and merged with the original colour video signal. Resulting output video signal is unchanged, with the exception of red colour marked moving edges.

Demo **md01** provides accelerated HW computation with one parallel HW data path. HW demo uses 2 DMA HW channels for reading from two sub sequent video frame buffers; both located both in the DDR3, to pass the data to the video processing chain of accelerators performing the motion detection. Another DMA HW channel performs parallel write of results to the DDR3. Demo is linked with static library libmd01.a. Zynq PL resources and accelerations reached for these HW designs are summarised in section 1.7.

Frames per Second (FPS) Measurements

We report the measured frames per second (FPS) reached by the accelerator and the FPS reached by the initial SW implementation on ARM in the SDSoC 2015.4. In case of SW implementation –O3 optimisation was used. The Full HD Toshiba input and the HDMI output data movements are supported by the dedicated VDMA HW channels in all cases.

1.4 Project sh01: EdkDSP accelerator with edge detection in single HLS accelerator

Zynq module:	TE0720-03-2IF (Trenz electronic)
Carrier board:	TE0701-06 (Trenz electronic)
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)
FMC card input:	Toshiba Full HD video sensor (color 1920x1080p60)
FMC card output:	Full HD HDMI output to display (color 1920x1080p60)
Accelerator:	8xSIMD EdkDSP (120 MHz) with MicroBlaze (100 MHz)
Accelerator:	Edge detection single data path (150 MHz)
Platform (SDSoC 2015.4):	C:\S\t20i2tm4\hio\
Demos (SDK 2015.4):	C:\VM07\t20i2tm4\sh01_rows_resize_25_to_100\ C:\VM07\t20i2tm4\sh01_rows_fixed_100\

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2017_02_28

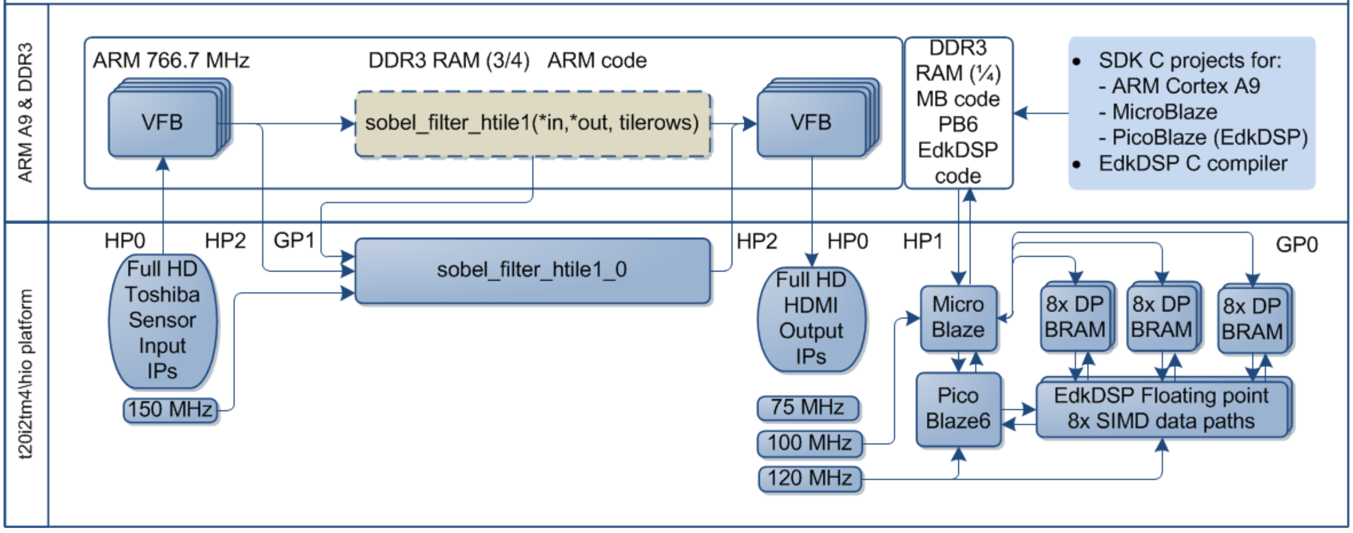
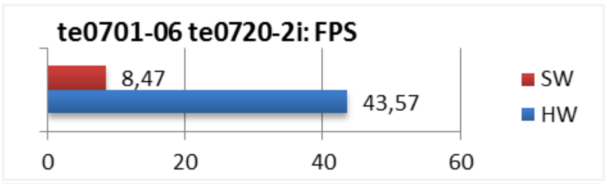
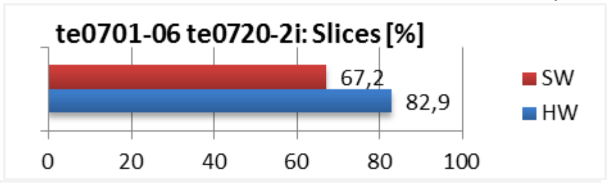
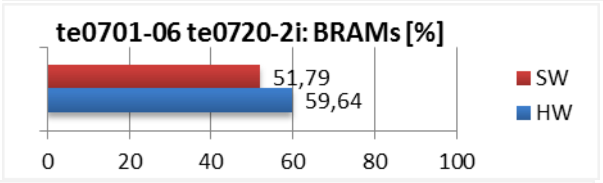


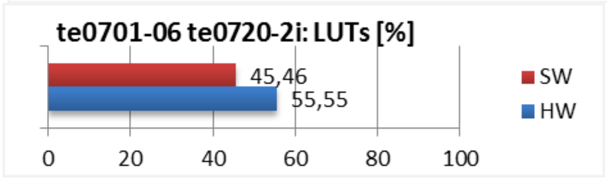
Figure 2: Project sh01 - Edge detection with single HW accelerator and EdkDSP accelerator.

Energy per pixel (nJ/p = nano Joule/pixel)		**Reduced:**
EdkDSP FIR SW: 417.86 nJ/p	HW: 132.77 nJ/p	**3.14 x**
EdkDSP LMS	HW: 132.29 nJ/p	
MB and no active EdkDSP	HW: 129.66 nJ/p	
MB, no EdkDSP in the PL	HW: 127.27 nJ/p	









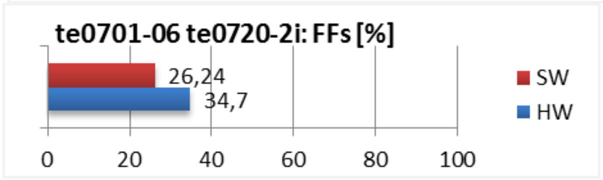


Figure 3: Project sh01 - Energy per frame reduction and used HW resources.

1.5 Project sh02: EdkDSP accelerator with edge detection in two HLS accelerators

Zynq module:	TE0720-03-2IF (Trenz electronic)
Carrier board:	TE0701-06 (Trenz electronic)
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)
FMC card input:	Full HD Toshiba video sensor (color 1920x1080p60)
FMC card output:	Full HD HDMI output to display (color 1920x1080p60)
Accelerator:	8xSIMD EdkDSP (120 MHz) with MicroBlaze (100 MHz)
Accelerator:	Edge detection two data paths (150 MHz)
Platform (SDK 2015.4):	C:\S\t20i2tm4\hio\
Demos (SDK 2015.4):	C:\VM07\t20i2tm4\sh02_rows_resize_25_to_100\
	C:\VM07\t20i2tm4\sh02_rows_fixed_100\

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2017_02_28

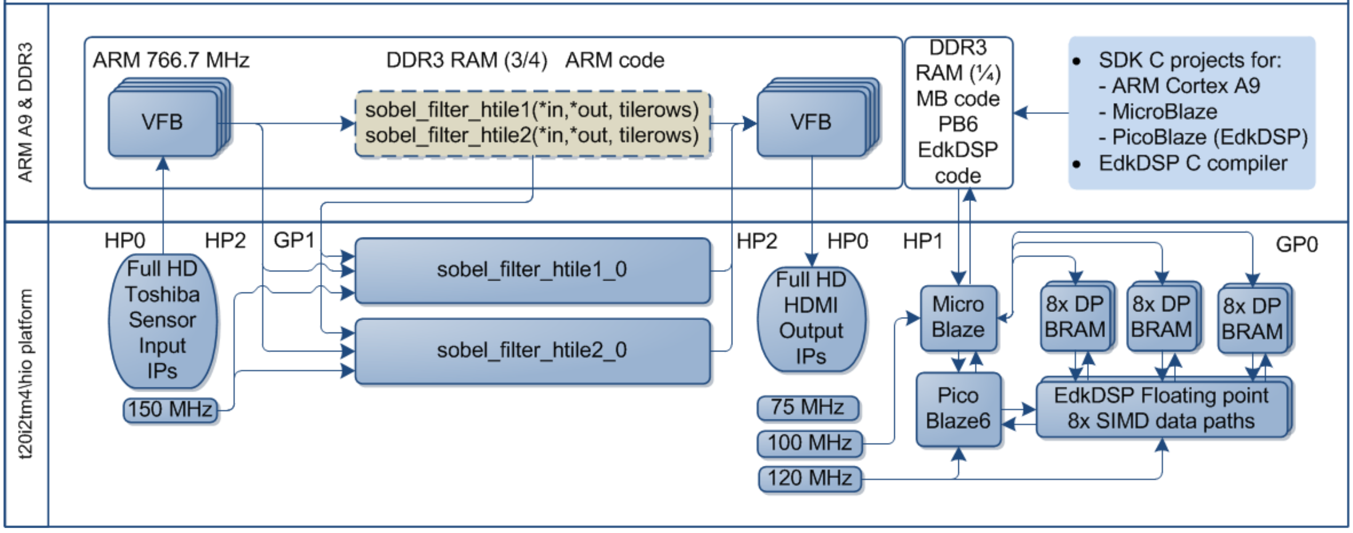
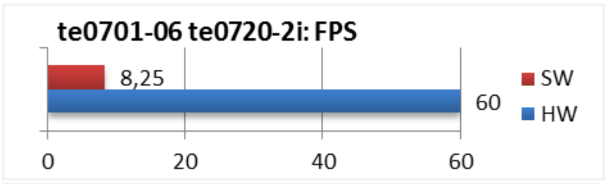
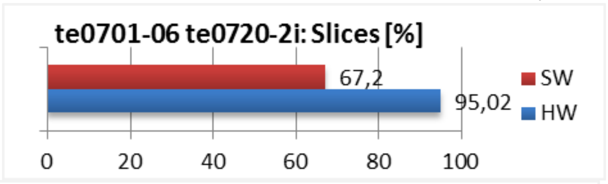
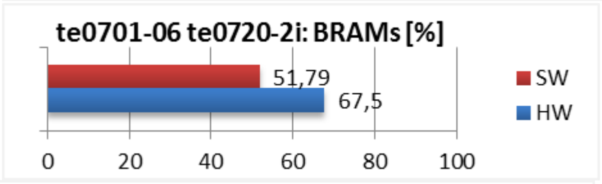


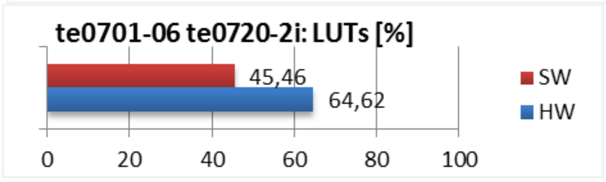
Figure 4: Project sh02 - Edge detection with two HW accelerators and 1x 8xSIMD EdkDSP IP.

Energy per pixel (nJ/p = nano Joule/pixel)			**Reduced:**
EdkDSP FIR	SW: 438.10 nJ/p	HW: 86.67 nJ/p	**5.05 x**
EdkDSP LMS		HW: 86.36 nJ/p	
MB and no active EdkDSP		HW: 84.69 nJ/p	
MB, no EdkDSP in the PL		HW: 82.86 nJ/p	









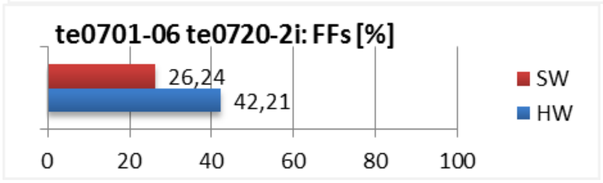


Figure 5: Project sh02 - Energy per frame reduction and used HW resources.

1.6 Project sh03: EdkDSP accelerator with edge detection in three HW accelerators

Zynq module:	TE0720-03-2IF (Trenz electronic)
Carrier board:	TE0701-05 (Trenz electronic)
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)
FMC card input:	Full HD Toshiba video sensor (color 1920x1080p60)
FMC card output:	Full HD HDMI output to display (color 1920x1080p60)
Accelerator:	8xSIMD EdkDSP (120 MHz) with MicroBlaze (100 MHz)
Accelerator:	Edge detection three data paths (150 MHz)
Platform (SDSoC 2015.4):	C:\S\t20i2tm4\hio\
Demos (SDK 2015.4):	C:\VM07\t20i2tm4\sh03_rows_resize_25_to_100\
	C:\VM07\t20i2tm4\sh03_rows_fixed_100\

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2017_02_28

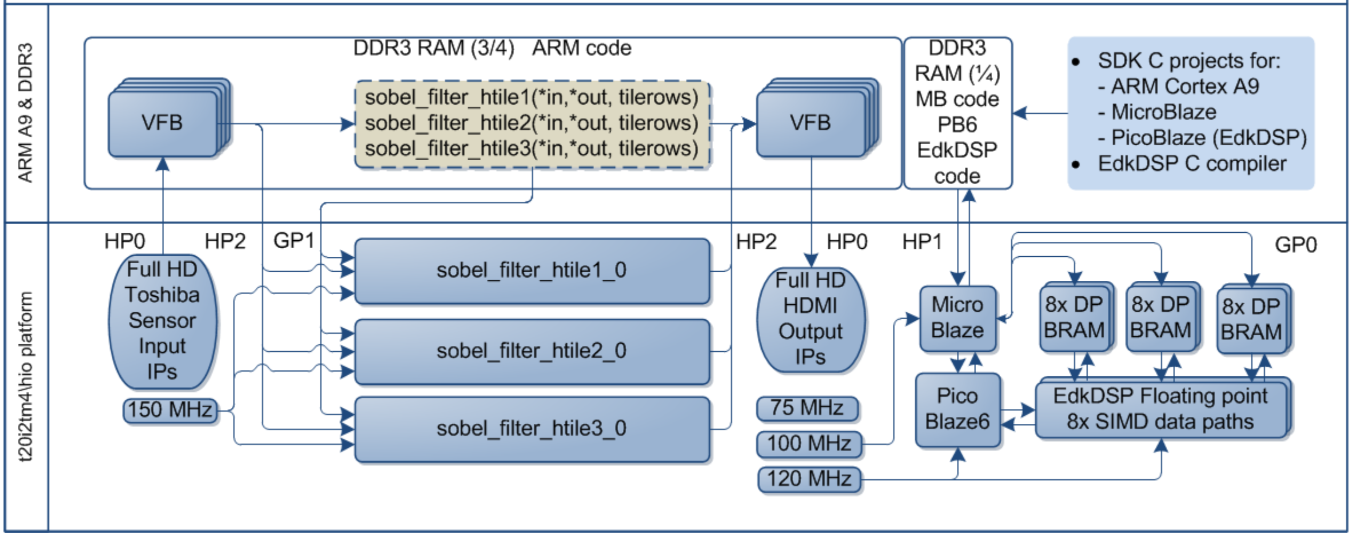
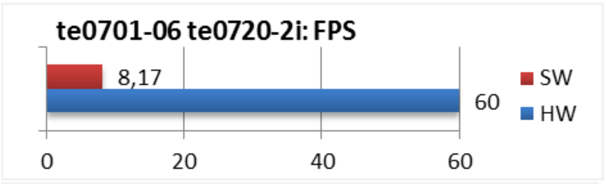
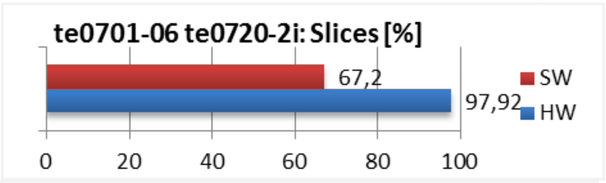
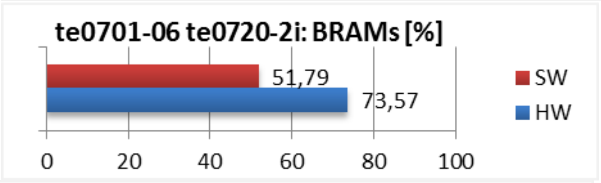


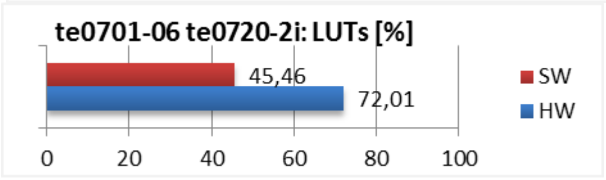
Figure 6: Project sh03 - Edge detection with three HW accelerators and 1x 8xSIMD EdkDSP IP.

Energy per pixel (nJ/p = nano Joule/pixel)			**Reduced:**
EdkDSP FIR	SW: 441.87 nJ/p	HW: 86.98 nJ/p	**5.08 x**
EdkDSP LMS		HW: 86.67 nJ/p	
MB and no active EdkDSP		HW: 84.99 nJ/p	
MB, no EdkDSP in the PL		HW: 83.31 nJ/p	









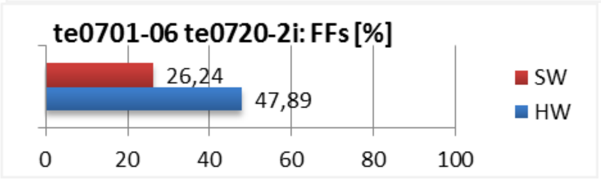
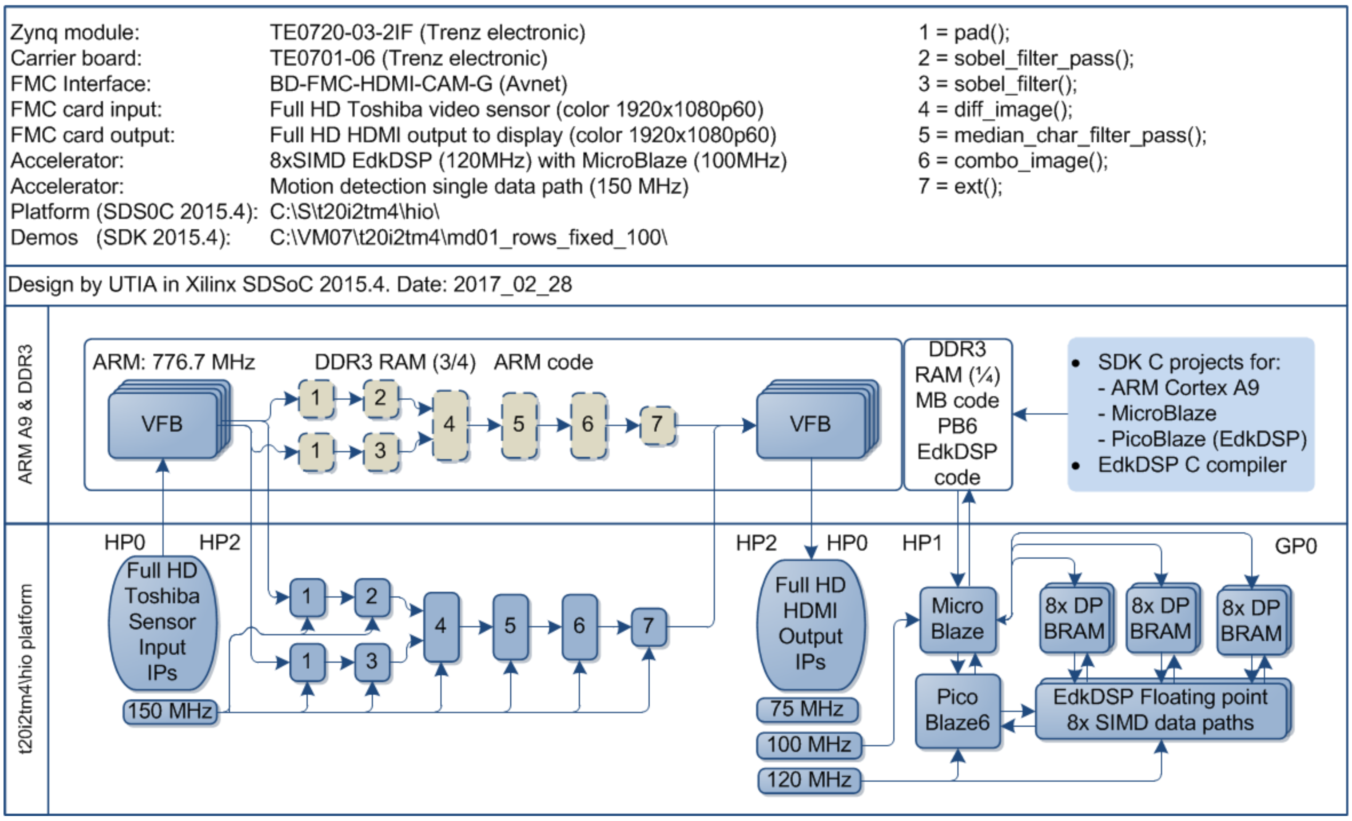


Figure 7: Project sh03 - Energy per frame reduction and used HW resources.

1.7 Project md01: EdkDSP accelerator with motion detection in HLS accelerators



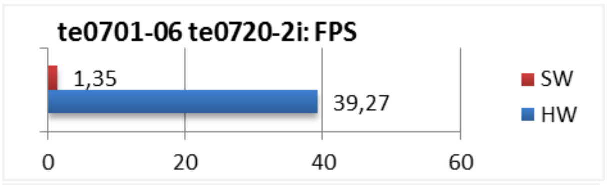
Zynq module:	TE0720-03-2IF (Trenz electronic)
Carrier board:	TE0701-06 (Trenz electronic)
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)
FMC card input:	Full HD Toshiba video sensor (color 1920x1080p60)
FMC card output:	Full HD HDMI output to display (color 1920x1080p60)
Accelerator:	8xSIMD EdkDSP (120MHz) with MicroBlaze (100MHz)
Accelerator:	Motion detection single data path (150 MHz)
Platform (SDS0C 2015.4):	C:\S\t20i2tm4\hio\
Demos (SDK 2015.4):	C:\VM07\t20i2tm4\md01_rows_fixed_100\

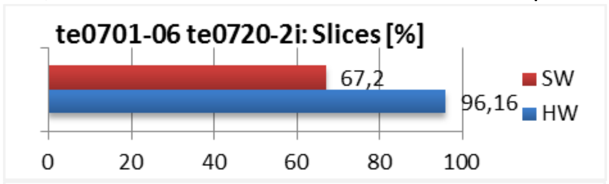
1 = pad();
2 = sobel_filter_pass();
3 = sobel_filter();
4 = diff_image();
5 = median_char_filter_pass();
6 = combo_image();
7 = ext();

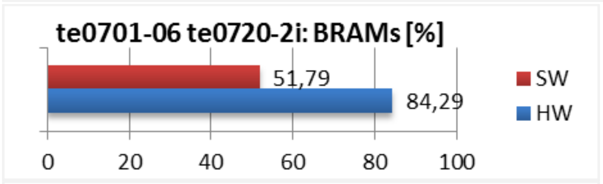
Design by UTIA in Xilinx SDSoC 2015.4. Date: 2017_02_28

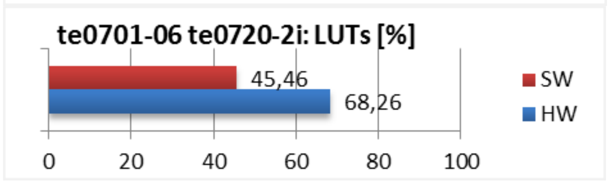
Figure 8: Project md01 - Motion detection with single HW accelerator and 1x 8xSIMD EdkDSP IP.

Energy per pixel (nJ/p = nano Joule/pixel)		**Reduced:**
EdkDSP FIR SW: 2575.7 nJ/p	HW: 130.00 nJ/p	**19.81 x**
EdkDSP LMS	HW: 129.56 nJ/p	
MB and no active EdkDSP	HW: 127.13 nJ/p	
MB, no EdkDSP in the PL	HW: 124.47 nJ/p	









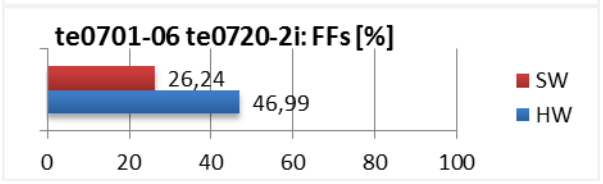


Figure 9: Project md01 - Energy per frame reduction and HW resources.

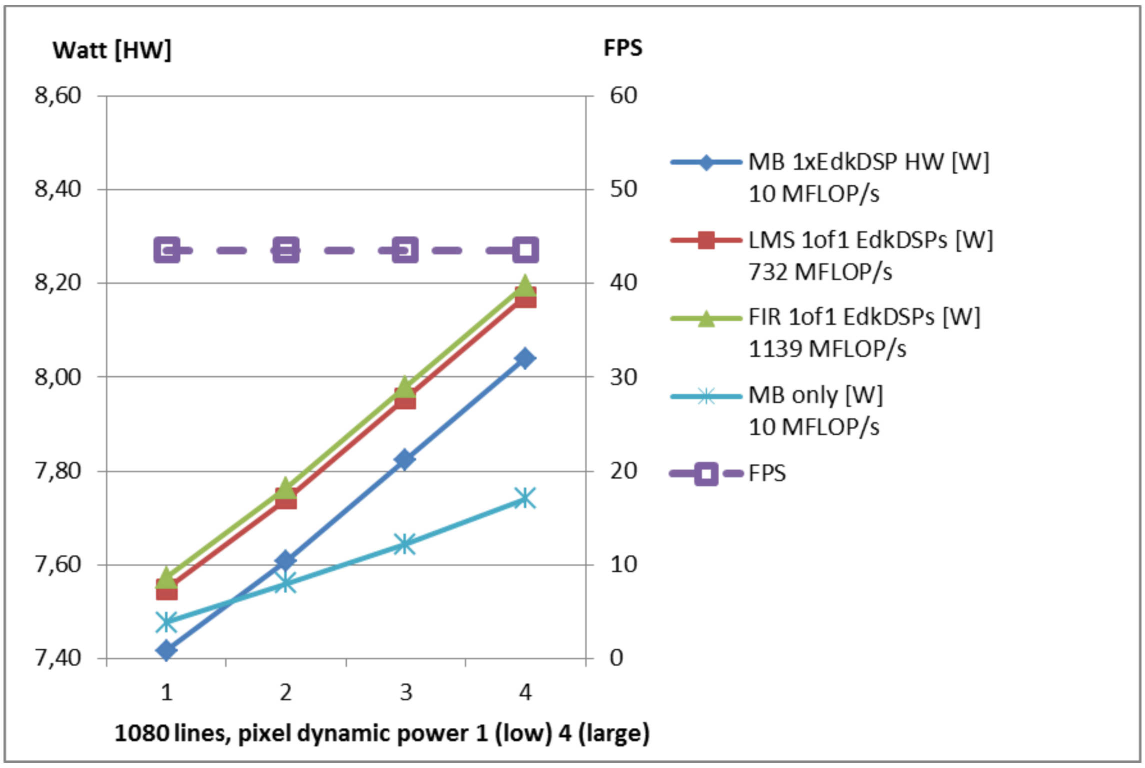


Figure 10: Project sh01 - Edge detection in 1x HW accelerator. 1x 8xSIMD EdkDSP IP.

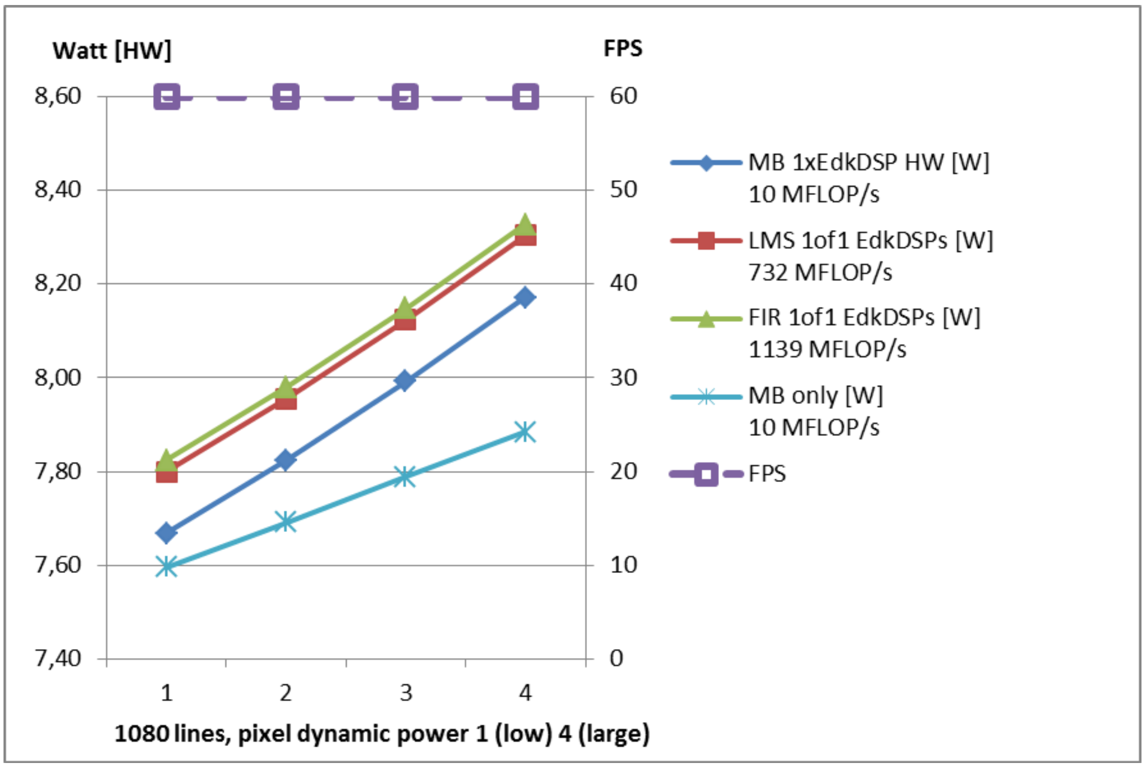


Figure 11: Project sh02 - Edge detection in 2x HW accelerators. 1x 8xSIMD EdkDSP IP.

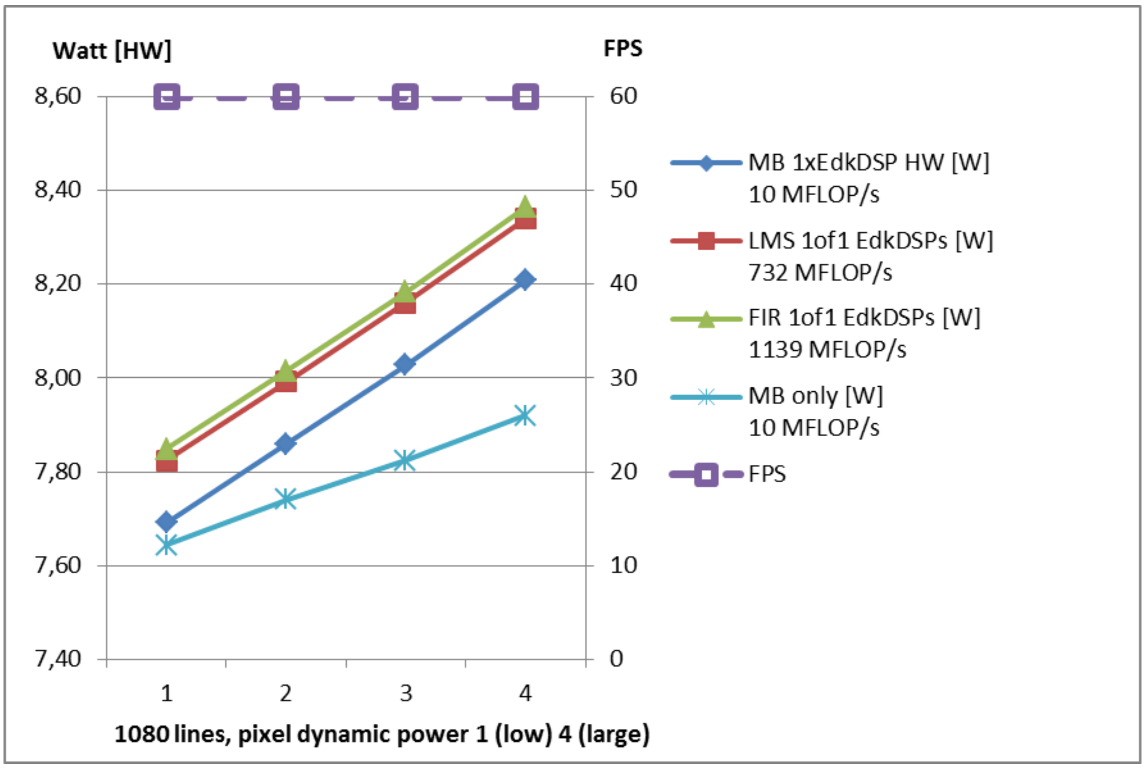


Figure 12: Project sh03 - Edge detection in 3x HW accelerator. 1x 8xSIMD EdkDSP IP.

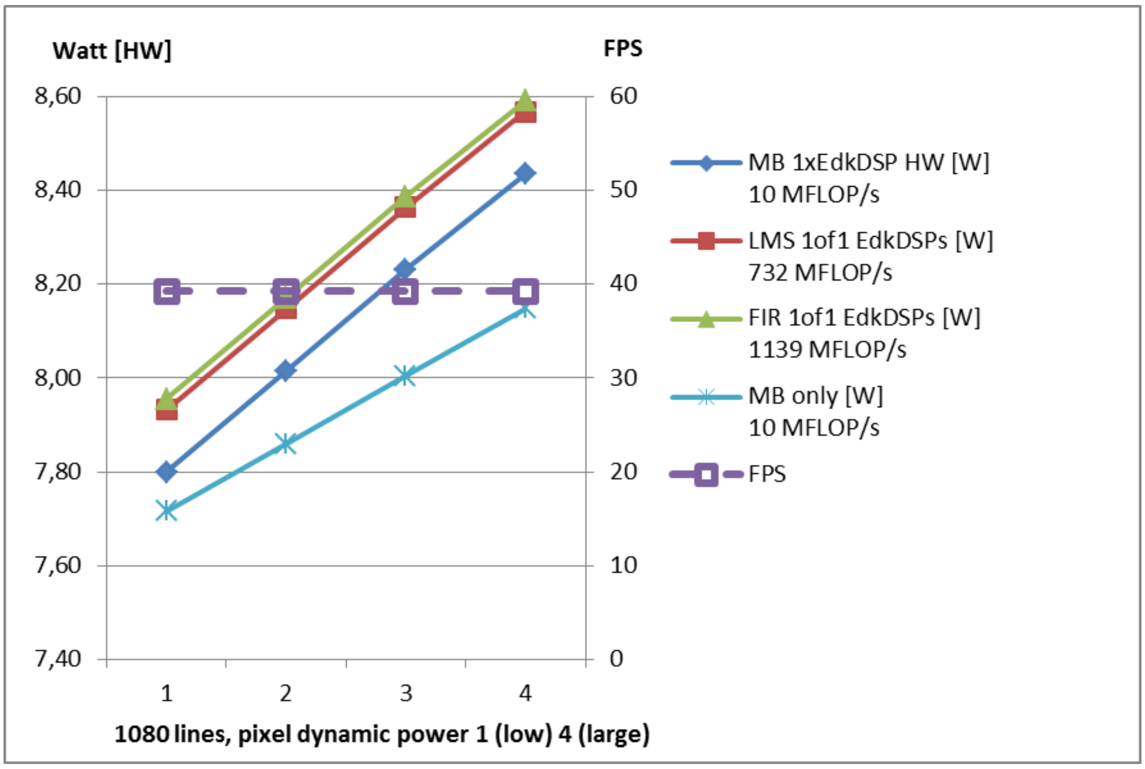


Figure 13: Project md01 - Motion detection in 8 chained HW accelerators. 1x 8xSIMD EdkDSP IP

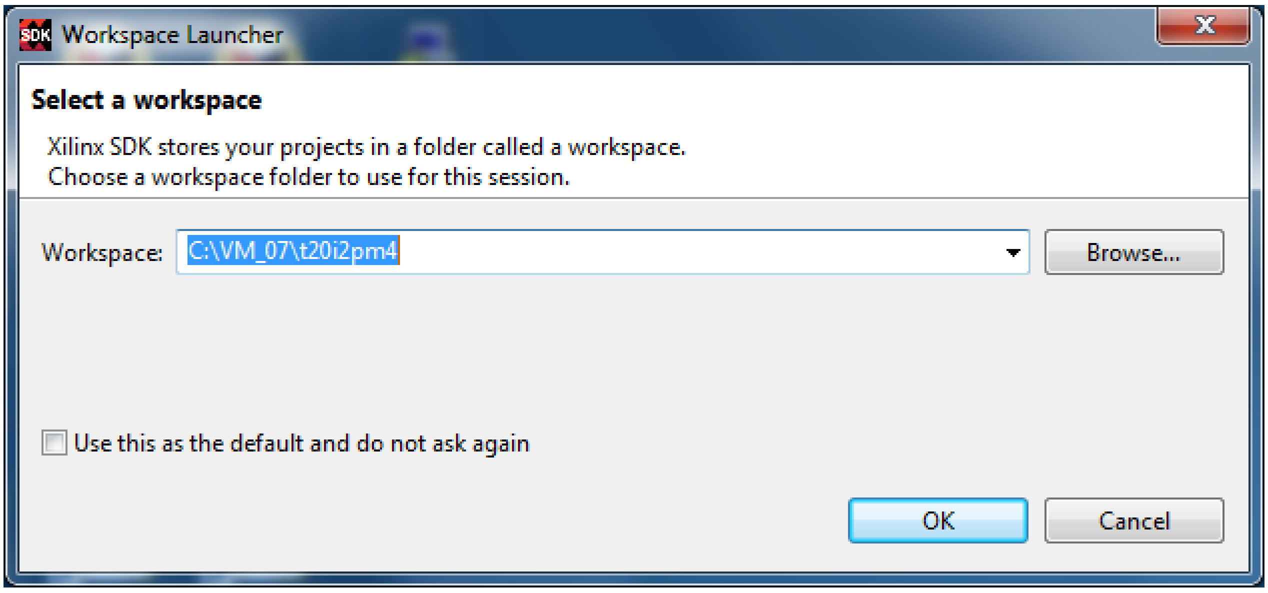
2. Installation of the Evaluation Package

2.1 Import of SW projects in Xilinx SDK 2015.4

Unzip the evaluation package to the directory
C:\VM_07\t20i2tm4_V54_IMPORT

Create empty directory for Xilinx SDK workspace.
C:\VM_07\t20i2tm4

Start Xilinx SDK 2015.4 and select the directory for the SDK 2015.4 workspace. See Figure 14
Select **C:\VM_07\t20i2tm4**


Workspace Launcher

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\t20i2pm4

Figure 14: Select the SDK Workspace.

HW and SW projects can be imported into SDK now. Select:

File -> Import -> General -> Existing Projects into Workspace

Click on Next button. See Figure 15.

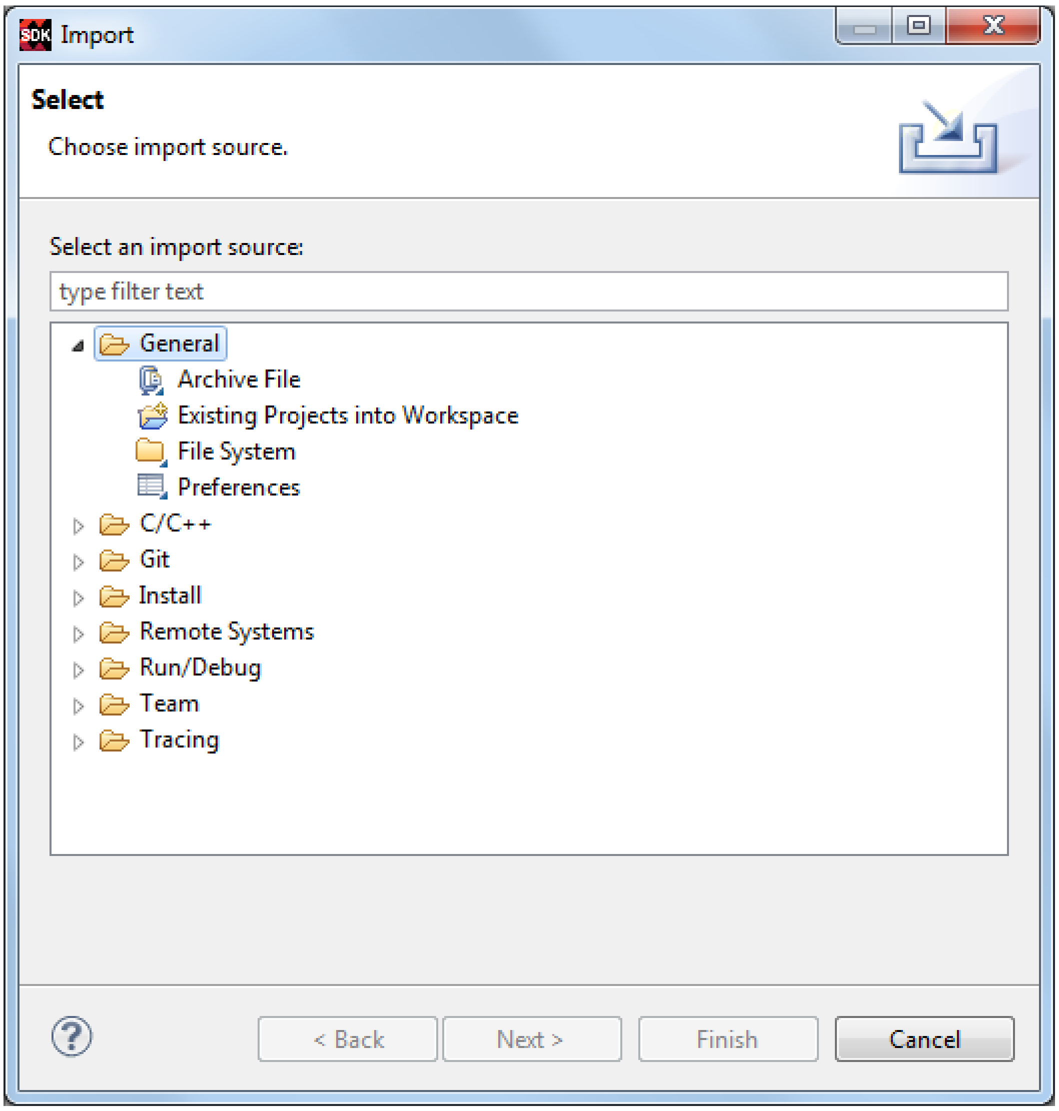


Figure 15: Import Existing Projects into Workspace.

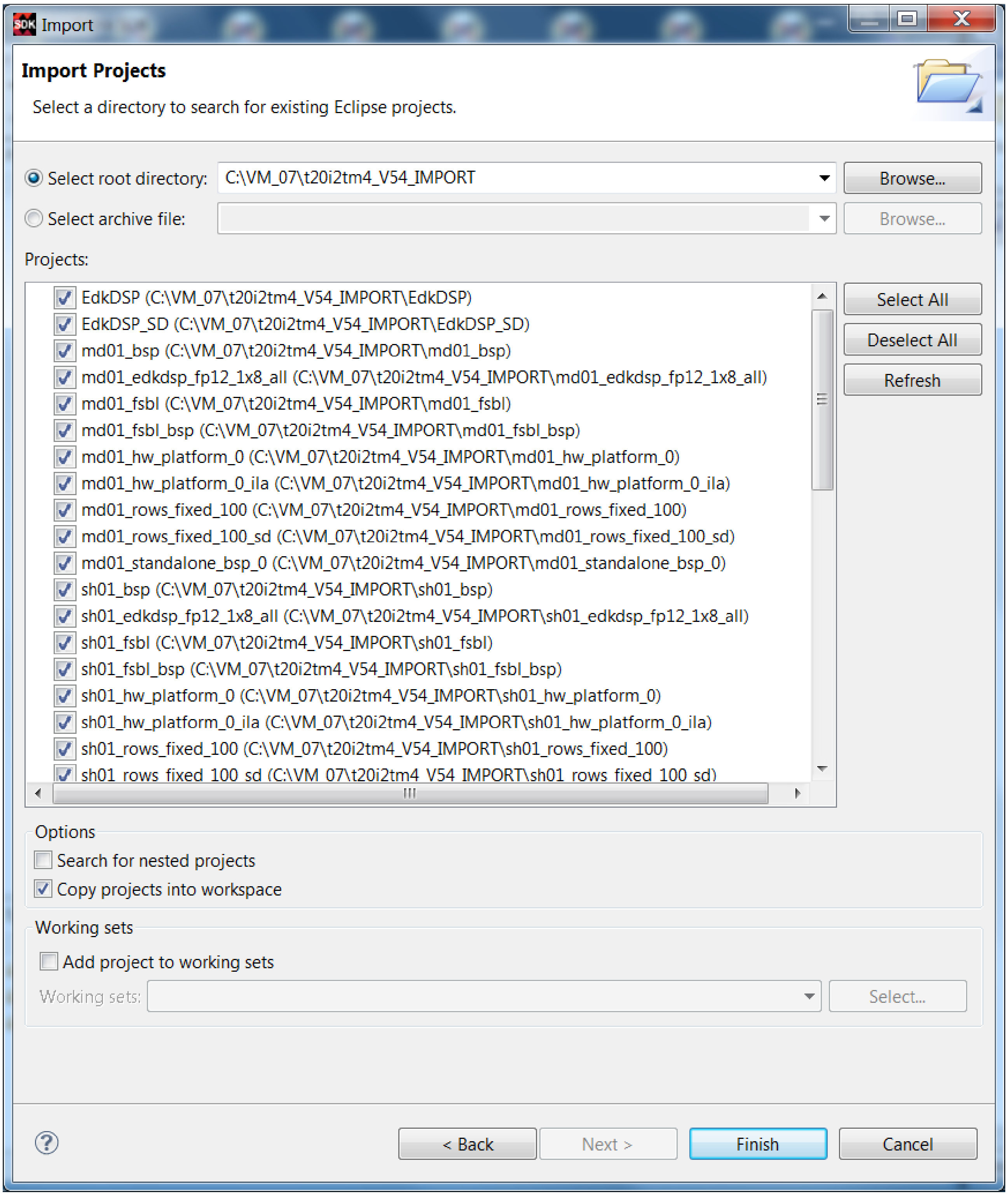
Type the directory with projects to be imported. See Figure 16.

C:\VM_07\t20i2tm4_V54_IMPORT

Set the "**Copy projects into workspace**" check box.

Click on Finish button. See Figure 16.

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.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

(● Select root directory:) C:\VM_07\t20i2tm4_V54_IMPORT Browse...

○ Select archive file: Browse...

Projects:

- ✓ EdkDSP (C:\VM_07\t20i2tm4_V54_IMPORT\EdkDSP)
- ✓ EdkDSP_SD (C:\VM_07\t20i2tm4_V54_IMPORT\EdkDSP_SD)
- ✓ md01_bsp (C:\VM_07\t20i2tm4_V54_IMPORT\md01_bsp)
- ✓ md01_edkdsp_fp12_1x8_all (C:\VM_07\t20i2tm4_V54_IMPORT\md01_edkdsp_fp12_1x8_all)
- ✓ md01_fsbl (C:\VM_07\t20i2tm4_V54_IMPORT\md01_fsbl)
- ✓ md01_fsbl_bsp (C:\VM_07\t20i2tm4_V54_IMPORT\md01_fsbl_bsp)
- ✓ md01_hw_platform_0 (C:\VM_07\t20i2tm4_V54_IMPORT\md01_hw_platform_0)
- ✓ md01_hw_platform_0_ila (C:\VM_07\t20i2tm4_V54_IMPORT\md01_hw_platform_0_ila)
- ✓ md01_rows_fixed_100 (C:\VM_07\t20i2tm4_V54_IMPORT\md01_rows_fixed_100)
- ✓ md01_rows_fixed_100_sd (C:\VM_07\t20i2tm4_V54_IMPORT\md01_rows_fixed_100_sd)
- ✓ md01_standalone_bsp_0 (C:\VM_07\t20i2tm4_V54_IMPORT\md01_standalone_bsp_0)
- ✓ sh01_bsp (C:\VM_07\t20i2tm4_V54_IMPORT\sh01_bsp)
- ✓ sh01_edkdsp_fp12_1x8_all (C:\VM_07\t20i2tm4_V54_IMPORT\sh01_edkdsp_fp12_1x8_all)
- ✓ sh01_fsbl (C:\VM_07\t20i2tm4_V54_IMPORT\sh01_fsbl)
- ✓ sh01_fsbl_bsp (C:\VM_07\t20i2tm4_V54_IMPORT\sh01_fsbl_bsp)
- ✓ sh01_hw_platform_0 (C:\VM_07\t20i2tm4_V54_IMPORT\sh01_hw_platform_0)
- ✓ sh01_hw_platform_0_ila (C:\VM_07\t20i2tm4_V54_IMPORT\sh01_hw_platform_0_ila)
- ✓ sh01_rows_fixed_100 (C:\VM_07\t20i2tm4_V54_IMPORT\sh01_rows_fixed_100)
- ✓ sh01_rows_fixed_100_sd (C:\VM_07\t20i2tm4_V54_IMPORT\sh01_rows_fixed_100_sd)

Select All | Deselect All | Refresh

Options

- ☐ Search for nested projects
- ✓ Copy projects into workspace

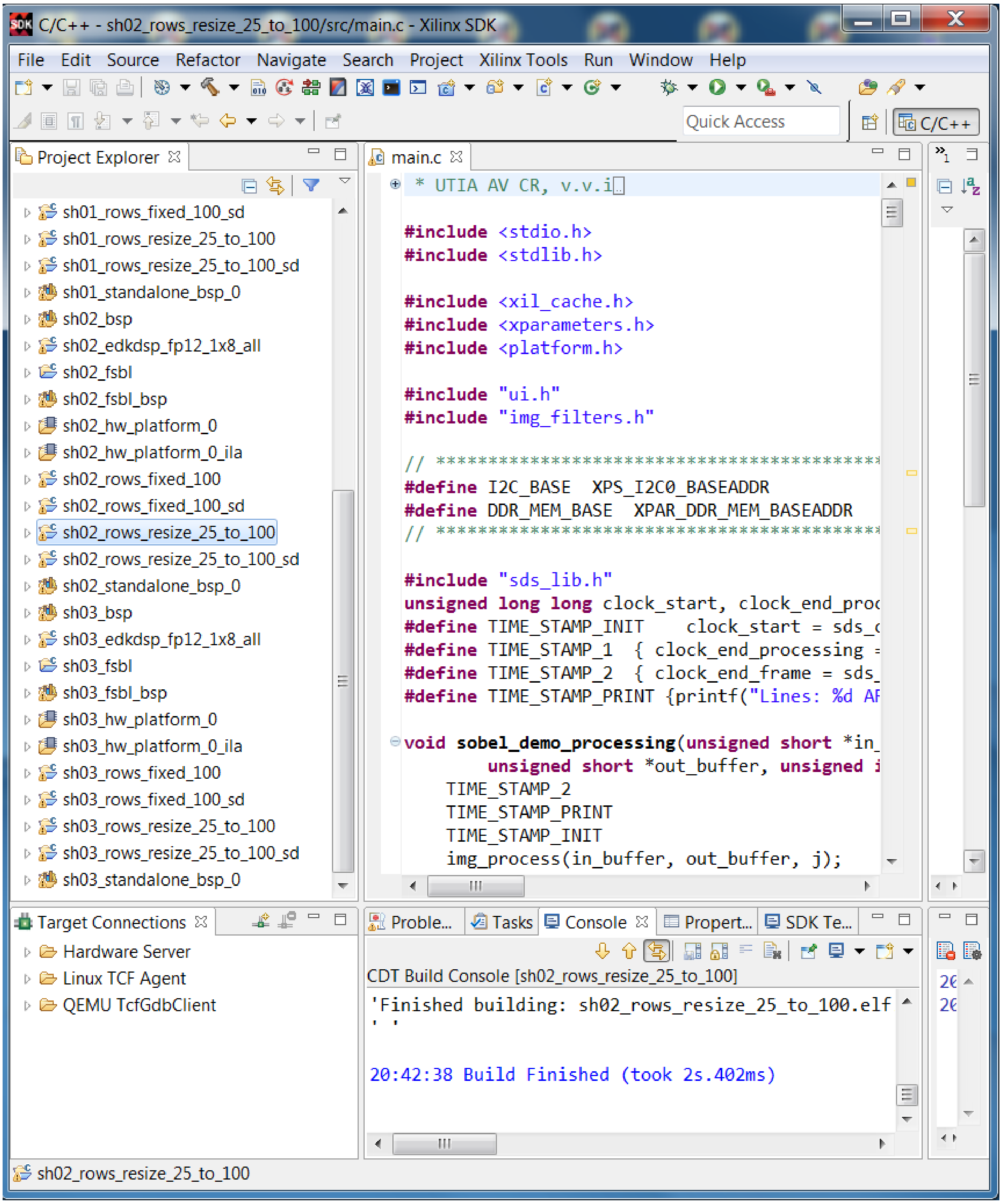
Working sets

- ☐ Add project to working sets

Working sets: Select...

< Back | Next > | Finish | Cancel

Figure 16: Select "Copy projects into workspace" and finish the import of all projects.



```
sh01_rows_fixed_100_sd
sh01_rows_resize_25_to_100
sh01_rows_resize_25_to_100_sd
sh01_standalone_bsp_0
sh02_bsp
sh02_edkdsp_fp12_1x8_all
sh02_fsbl
sh02_fsbl_bsp
sh02_hw_platform_0
sh02_hw_platform_0_ila
sh02_rows_fixed_100
sh02_rows_fixed_100_sd
sh02_rows_resize_25_to_100
sh02_rows_resize_25_to_100_sd
sh02_standalone_bsp_0
sh03_bsp
sh03_edkdsp_fp12_1x8_all
sh03_fsbl
sh03_fsbl_bsp
sh03_hw_platform_0
sh03_hw_platform_0_ila
sh03_rows_fixed_100
sh03_rows_fixed_100_sd
sh03_rows_resize_25_to_100
sh03_rows_resize_25_to_100_sd
sh03_standalone_bsp_0
```

```c
 * UTIA AV CR, v.v.i.

#include <stdio.h>
#include <stdlib.h>

#include <xil_cache.h>
#include <xparameters.h>
#include <platform.h>

#include "ui.h"
#include "img_filters.h"

// *******************************************
#define I2C_BASE   XPS_I2C0_BASEADDR
#define DDR_MEM_BASE   XPAR_DDR_MEM_BASEADDR
// *******************************************

#include "sds_lib.h"
unsigned long long clock_start, clock_end_proc
#define TIME_STAMP_INIT     clock_start = sds_c
#define TIME_STAMP_1  { clock_end_processing =
#define TIME_STAMP_2  { clock_end_frame = sds_
#define TIME_STAMP_PRINT {printf("Lines: %d AF

void sobel_demo_processing(unsigned short *in_
        unsigned short *out_buffer, unsigned i
    TIME_STAMP_2
    TIME_STAMP_PRINT
    TIME_STAMP_INIT
    img_process(in_buffer, out_buffer, j);
```

```
CDT Build Console [sh02_rows_resize_25_to_100]
'Finished building: sh02_rows_resize_25_to_100.elf
' '

20:42:38 Build Finished (took 2s.402ms)
```

sh02_rows_resize_25_to_100

Figure 17: All projects are compiled in debug mode.

The SDK 2015.4 tool compiles SW of all imported demos for the default debug mode. This will take several minutes.

2.2 HW setup

HW setup is based on commercially accessible components [1], [2], [3], [4], [5], [6]:

TE0720-2IF; Part: XC7Z020-2CLG484I; 1 GByte DDR; Industrial Grade (-40°C to +85°C) [1].
Heatsink for TE0720, spring-loaded embedded [2].
TE0701 Carrier Board for Trenz Electronic 7 Series [3].
AES-FMC-HDMI-CAM-G FMC card with HDMI I/O and CAM interface [4].
FULL HD TOSHIBA Color Image Sensor Camera Module [5].
PmodRS232: Serial converter & interface [6].

See the technical reference manuals (TRM) for the description of the TE0720-02-2IF or TE0720-03-2IF module revision [1] and TE0701-04, TE0701-05 or TE0701-06 revision of carrier boards [3].

Set the TE0701 carrier board switches and jumpers for the TE0720-03-2IF module as follows:

- **Set switch S3**
  TE701-04 switch S3: 1=ON 2=OFF 3=ON 4=OFF
  TE701-05 switch S3: 1=ON 2=OFF 3=ON 4=OFF
  TE701-06 switch S3: 1=ON 2=OFF 3=ON 4=OFF

- **Set jumpers to generate FMC_VADJ=2.5V**
  TE701-04: (VIOTA=2.5V VIOTB=FMC_VADJ=VADJ=2.5V) J17: connect 2-3; J21: connect 2-3
  TE701-05: (VIOTA=2.5V VIOTB=FMC_VADJ=VADJ=2.5V) J17: connect 2-3; J21: connect 2-3
  TE701-06: (VIOTA=VIOTB=FMC_VADJ=VADJ=2.5V) J16: closed; J17: open; J21: connect 2-3

- **Set switch S4** (only in case of TE701-06)
  TE701-06 switch S4: (set FMC_VADJ=2.5V as power source): 1=OFF 2=ON 3=ON 4=OFF

The TE0720-2IF Zynq device works with all IO-bank supply-voltages 2.5V. It is highly recommended to set switches of the TE0701 carrier board and measure the PL IO-bank supply-voltage before mounting of the module on TE0701. See locations and switch positions in the TE0701 TRMs [3].

Note:
In case of TE0701-04 or TE0701-05, the PmodRS232 serial converter & interface [6] is powered by 3.3V.

In case of the TE0701-06 [3], the PmodRS232 serial converter & interface [6] will be powered by VIOTA=2.5V of the PMOD header J5. This voltage is below the power supply specification (3V … 5V) of the PmodRS232 serial converter. Based on our experiments the PmodRS232 serial converter works OK with the power supply 2.5V.

2.3 Test demos

To test demos follow these steps:

- Insert the Full HD Toshiba video sensor to the connector on the AES-FMC-HDMI-CAM-G board.
- Connect HDMI (or DVI) monitor by HDMI cable to the HDMI OUT on the AES-FMC-HDMI-CAM-G card.
- Switch the monitor ON.
- Connect the carrier board by USB-to-microUSB cable to PC to support JTAG serial link and the standard serial terminal.
- Connect the PmodRS232 serial converter & interface module to the carrier board as indicated in Figure 18. Connect the RS232 cable to COM1 serial terminal of your PC. This serial line will support serial terminal for the MicroBlaze processor.
- Connect power supply (DC 12V).

- Open and configure the standard serial terminal client (PuTTY or similar) on PC for the ARM serial terminal (USB emulated). Set up:
  Speed: 115200 baud; Data bits: 8; Stop bits: 1; Parity: None; Flow control: None.
- Open and configure the standard serial terminal client (PuTTY or similar) on PC for MicroBlaze It is COM1. Set up:
  Speed: 115200 baud; Data bits: 8; Stop bits: 1; Parity: None; Flow control: None.
- Reset the board. Board will start first stage boot loader from internal flash as set up by Trenz Electronic. It writes messages to the serial terminal. On request, “Hit any key to stop autoboot” press any key to stop the auto-boot of Linux.
- If you need to switch-off the power, close first the serial terminal on the PC. This will help to avoid problems with lost communication.

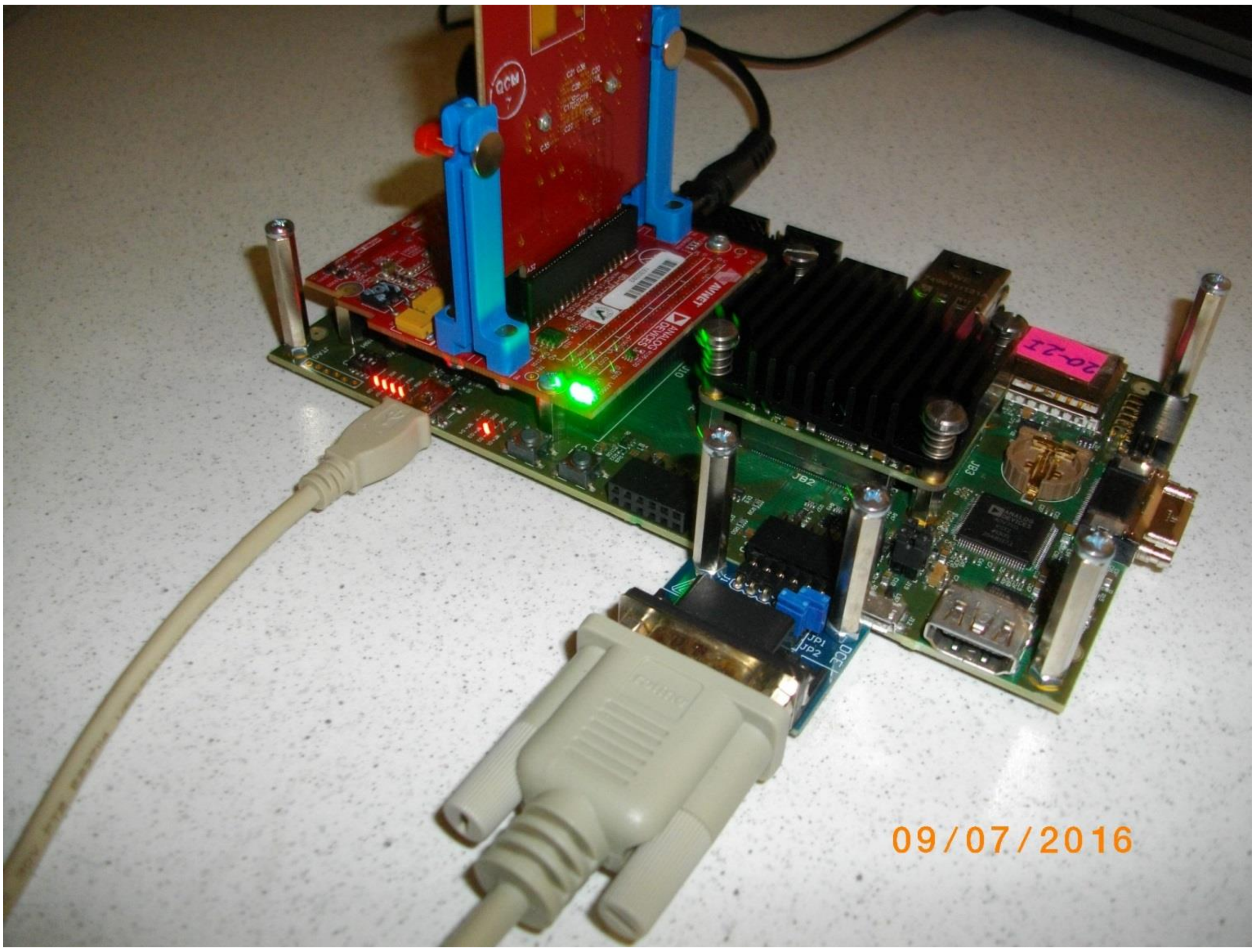
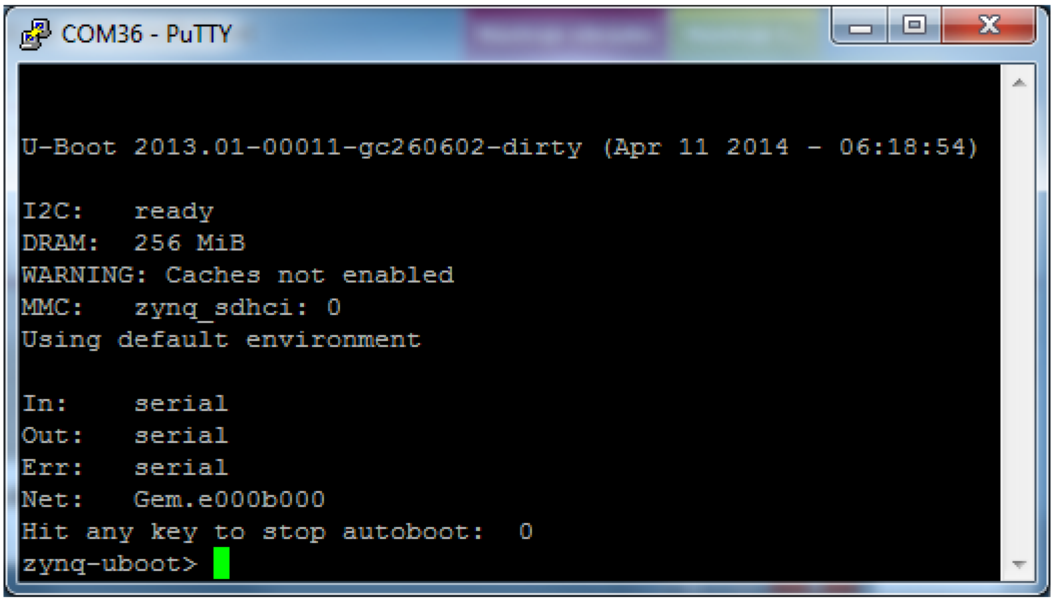


Figure 18: USB for ARM terminal and JTAG. RS232C Pmod for MicroBlaze.

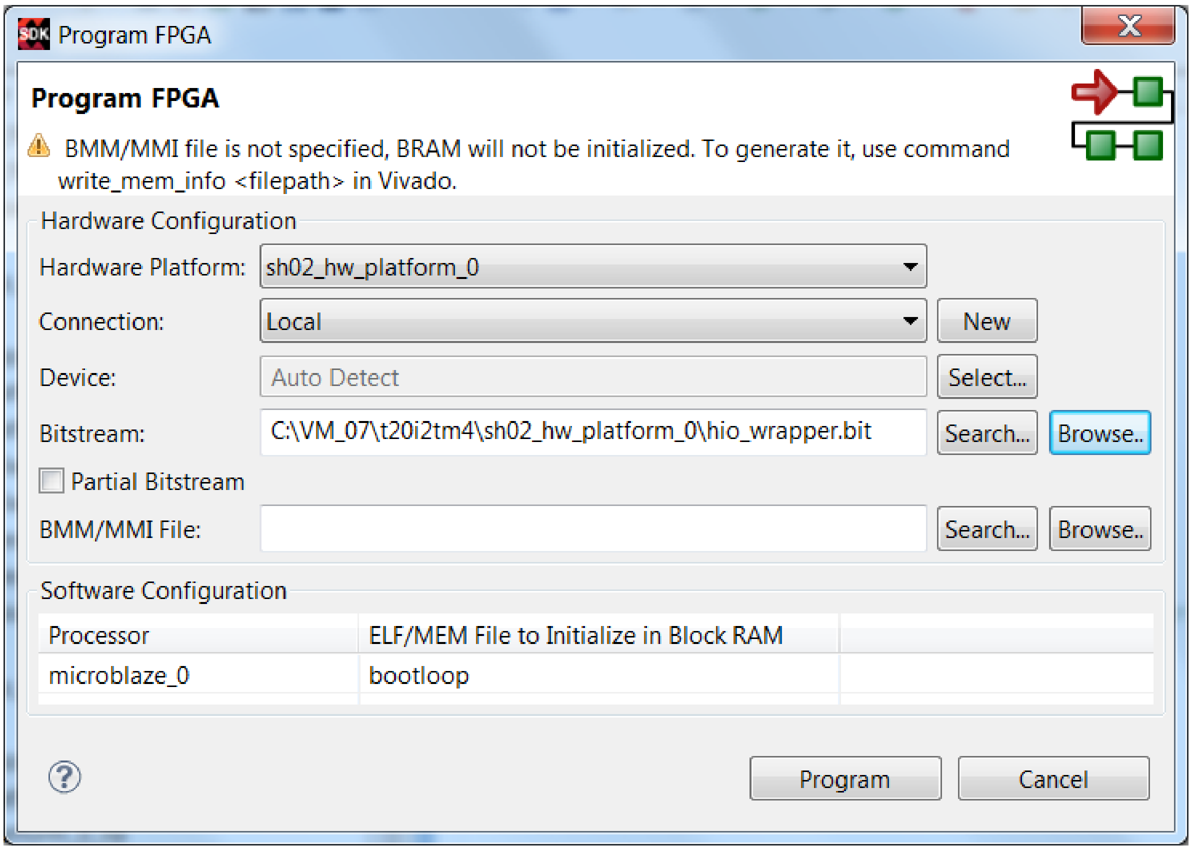


```
U-Boot 2013.01-00011-gc260602-dirty (Apr 11 2014 - 06:18:54)

I2C:   ready
DRAM:  256 MiB
WARNING: Caches not enabled
MMC:   zynq_sdhci: 0
Using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
Hit any key to stop autoboot:  0
zynq-uboot>
```

Figure 19: Serial console. Reset board and stop autoboot by any key.

Download bitstream to the board. Demo **sh02_rows_resize_25_to_100** will be used as an example.
The **bitstream.bit** file for demo **sh02** is located in the directory: **C:\VM_07\t20i2tm4\sh02_hw_platform_0**

Select Program to download the bitstream to the PL part of Zynq via the USB cable in JTAG mode.



Program FPGA

BMM/MMI file is not specified, BRAM will not be initialized. To generate it, use command write_mem_info <filepath> in Vivado.

Hardware Configuration

Hardware Platform: sh02_hw_platform_0

Connection: Local

Device: Auto Detect

Bitstream: C:\VM_07\t20i2tm4\sh02_hw_platform_0\hio_wrapper.bit

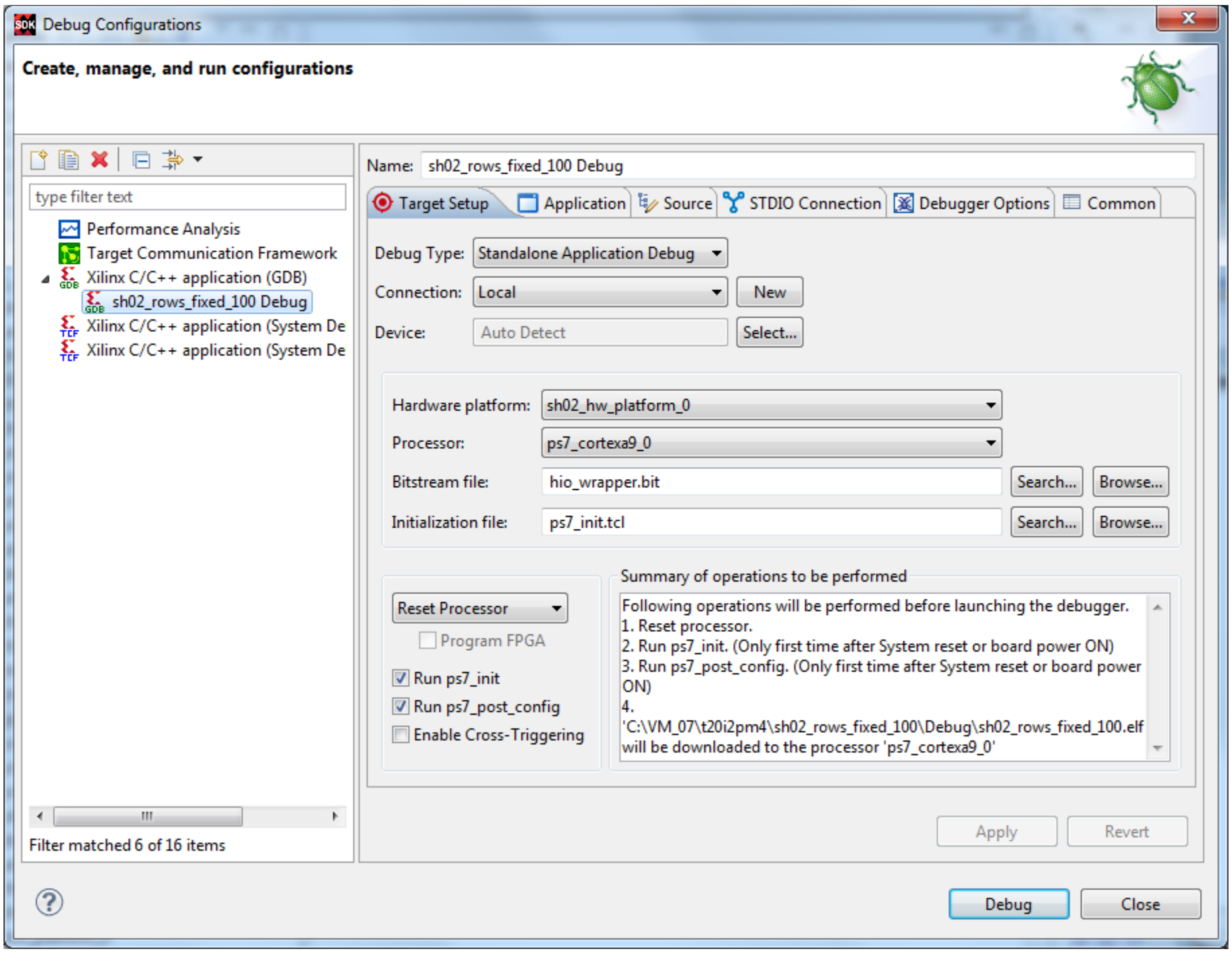
Partial Bitstream

BMM/MMI File:

Software Configuration

Processor	ELF/MEM File to Initialize in Block RAM
microblaze_0	bootloop

Figure 20: Download bitstream to the PL part of Zynq.



Debug Configurations

Create, manage, and run configurations

Name: sh02_rows_fixed_100 Debug

Target Setup | Application | Source | STDIO Connection | Debugger Options | Common

Debug Type: Standalone Application Debug

Connection: Local

Device: Auto Detect

Hardware platform: sh02_hw_platform_0

Processor: ps7_cortexa9_0

Bitstream file: hio_wrapper.bit

Initialization file: ps7_init.tcl

Reset Processor

☐ Program FPGA

☑ Run ps7_init

☑ Run ps7_post_config

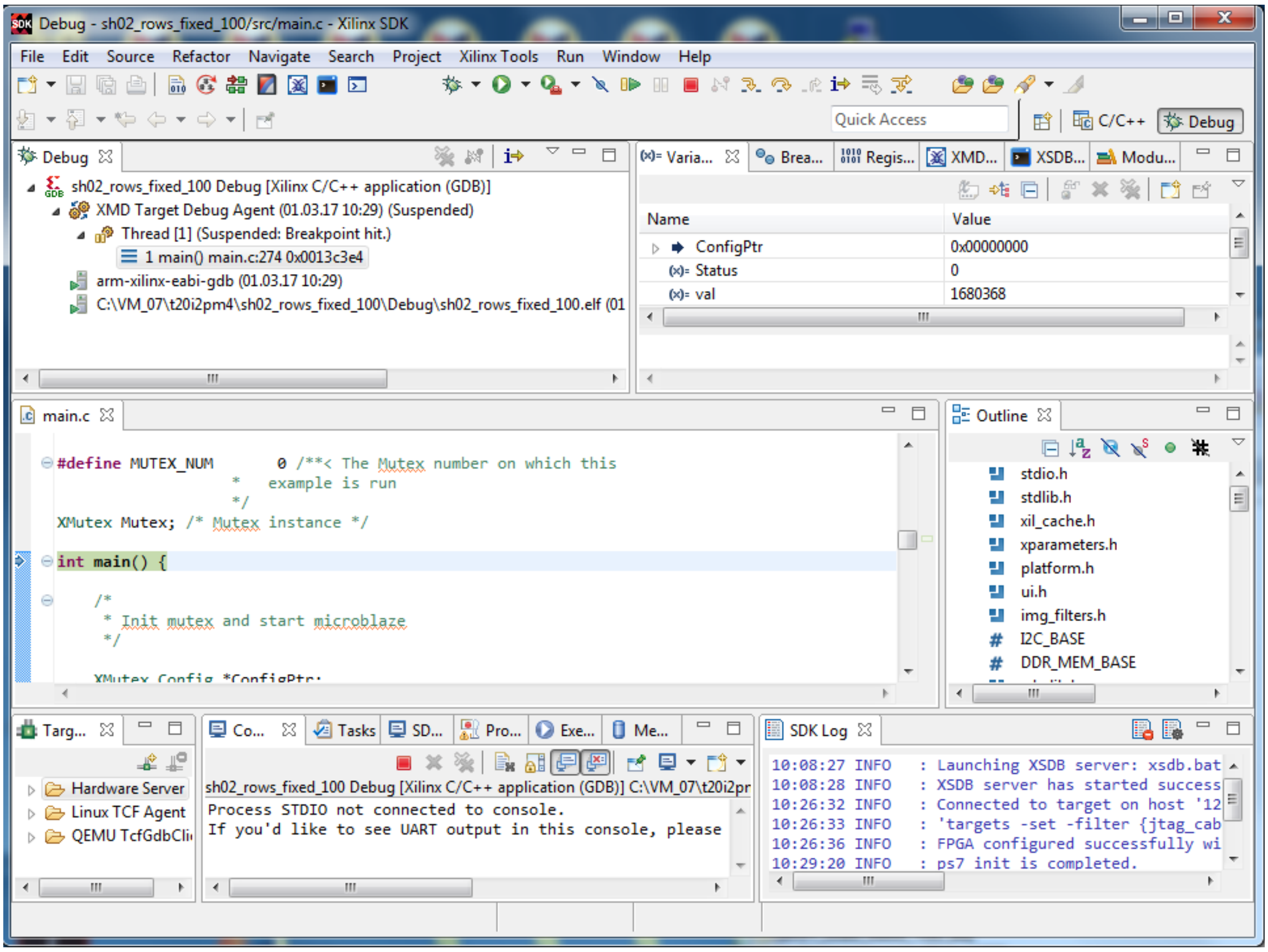
☐ Enable Cross-Triggering

Summary of operations to be performed

Following operations will be performed before launching the debugger.
1. Reset processor.
2. Run ps7_init. (Only first time after System reset or board power ON)
3. Run ps7_post_config. (Only first time after System reset or board power ON)
4.
'C:\VM_07\t20i2pm4\sh02_rows_fixed_100\Debug\sh02_rows_fixed_100.elf
will be downloaded to the processor 'ps7_cortexa9_0'

type filter text
Performance Analysis
Target Communication Framework
Xilinx C/C++ application (GDB)
sh02_rows_fixed_100 Debug
Xilinx C/C++ application (System De
Xilinx C/C++ application (System De
Filter matched 6 of 16 items
New
Select...
Search...
Browse...
Apply
Revert
Debug
Close

Figure 21: Select demo application for debug.



```
sh02_rows_fixed_100 Debug [Xilinx C/C++ application (GDB)]
  XMD Target Debug Agent (01.03.17 10:29) (Suspended)
    Thread [1] (Suspended: Breakpoint hit.)
      1 main() main.c:274 0x0013c3e4
  arm-xilinx-eabi-gdb (01.03.17 10:29)
  C:\VM_07\t20i2pm4\sh02_rows_fixed_100\Debug\sh02_rows_fixed_100.elf (01
```

Name	Value
ConfigPtr	0x00000000
Status	0
val	1680368

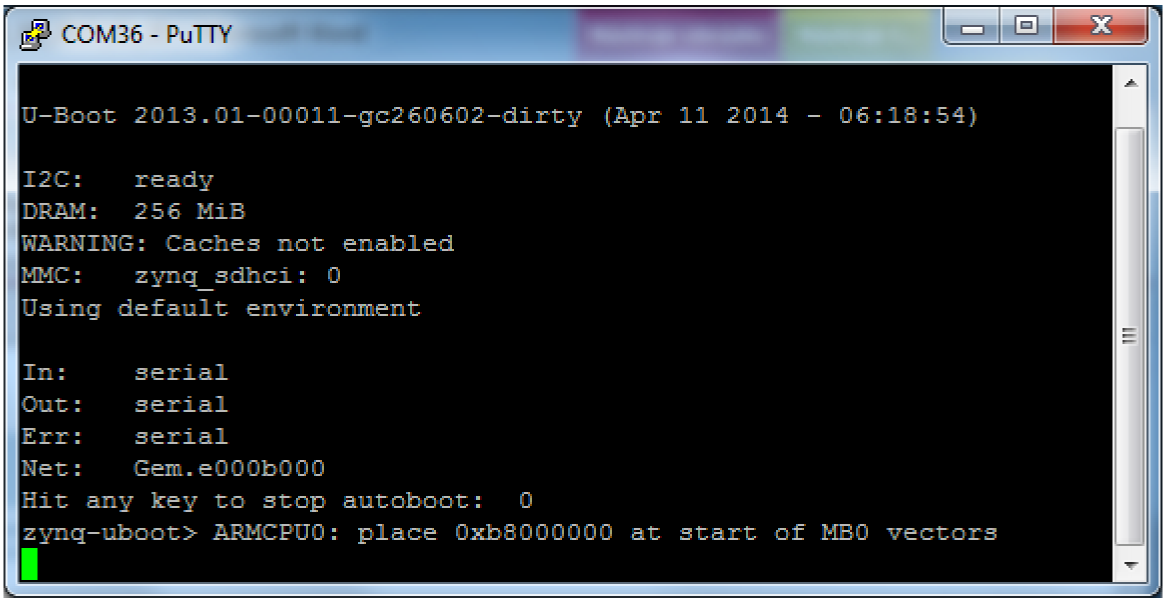
```c
#define MUTEX_NUM       0 /**< The Mutex number on which this
                   *   example is run
                   */
XMutex Mutex; /* Mutex instance */

int main() {

    /*
     * Init mutex and start microblaze
     */

    XMutex_Config *ConfigPtr;
```

```
stdio.h
stdlib.h
xil_cache.h
xparameters.h
platform.h
ui.h
img_filters.h
I2C_BASE
DDR_MEM_BASE
```

```
sh02_rows_fixed_100 Debug [Xilinx C/C++ application (GDB)] C:\VM_07\t20i2pr
Process STDIO not connected to console.
If you'd like to see UART output in this console, please
```

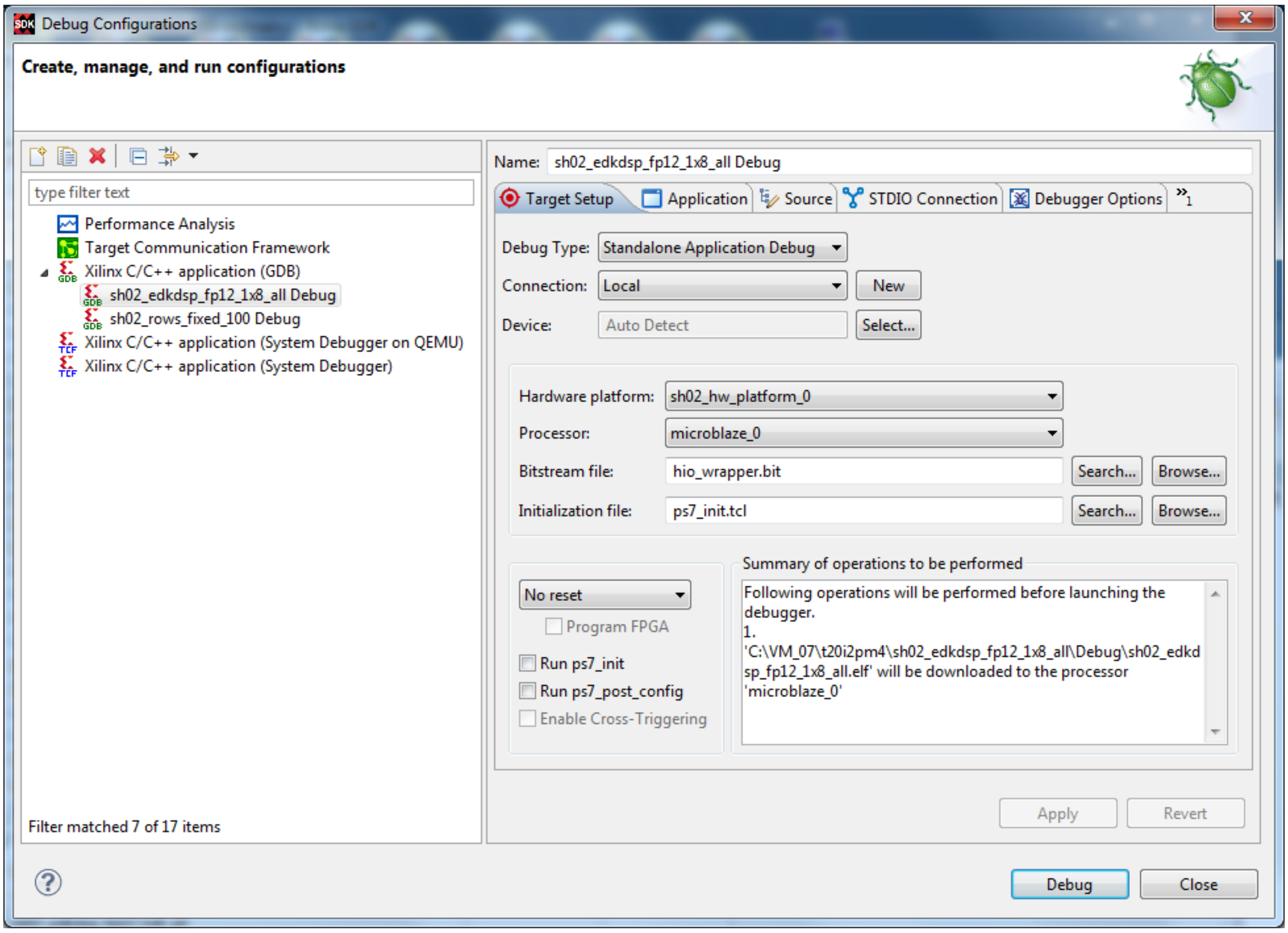
```
10:08:27 INFO   : Launching XSDB server: xsdb.bat
10:08:28 INFO   : XSDB server has started success
10:26:32 INFO   : Connected to target on host '12
10:26:33 INFO   : 'targets -set -filter {jtag_cab
10:26:36 INFO   : FPGA configured successfully wi
10:29:20 INFO   : ps7 init is completed.
```

Figure 22: Demo app is booted to ARM and the debugger is waiting on the first executable line.



```
U-Boot 2013.01-00011-gc260602-dirty (Apr 11 2014 - 06:18:54)

I2C:   ready
DRAM:  256 MiB
WARNING: Caches not enabled
MMC:   zynq_sdhci: 0
Using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
Hit any key to stop autoboot:  0
zynq-uboot> ARMCPU0: place 0xb8000000 at start of MB0 vectors
```

Figure 23: ARM is waiting on HW Mutex for the MicroBlaze start.


Debug Configurations

Create, manage, and run configurations

Name: sh02_edkdsp_fp12_1x8_all Debug

Debug Type: Standalone Application Debug

Connection: Local

Device: Auto Detect

Hardware platform: sh02_hw_platform_0

Processor: microblaze_0

Bitstream file: hio_wrapper.bit

Initialization file: ps7_init.tcl

No reset

Program FPGA [ ]

Run ps7_init [ ]

Run ps7_post_config [ ]

Enable Cross-Triggering [ ]

Summary of operations to be performed

Following operations will be performed before launching the debugger.
1.
'C:\VM_07\t20i2pm4\sh02_edkdsp_fp12_1x8_all\Debug\sh02_edkdsp_fp12_1x8_all.elf' will be downloaded to the processor 'microblaze_0'

Filter matched 7 of 17 items

type filter text
Performance Analysis
Target Communication Framework
Xilinx C/C++ application (GDB)
sh02_edkdsp_fp12_1x8_all Debug
sh02_rows_fixed_100 Debug
Xilinx C/C++ application (System Debugger on QEMU)
Xilinx C/C++ application (System Debugger)
Target Setup
Application
Source
STDIO Connection
Debugger Options
New
Select...
Search...
Browse...
Apply
Revert
Debug
Close

Figure 24: Select the MicroBlaze application (with the EdkDSP accelerator code) for debug.

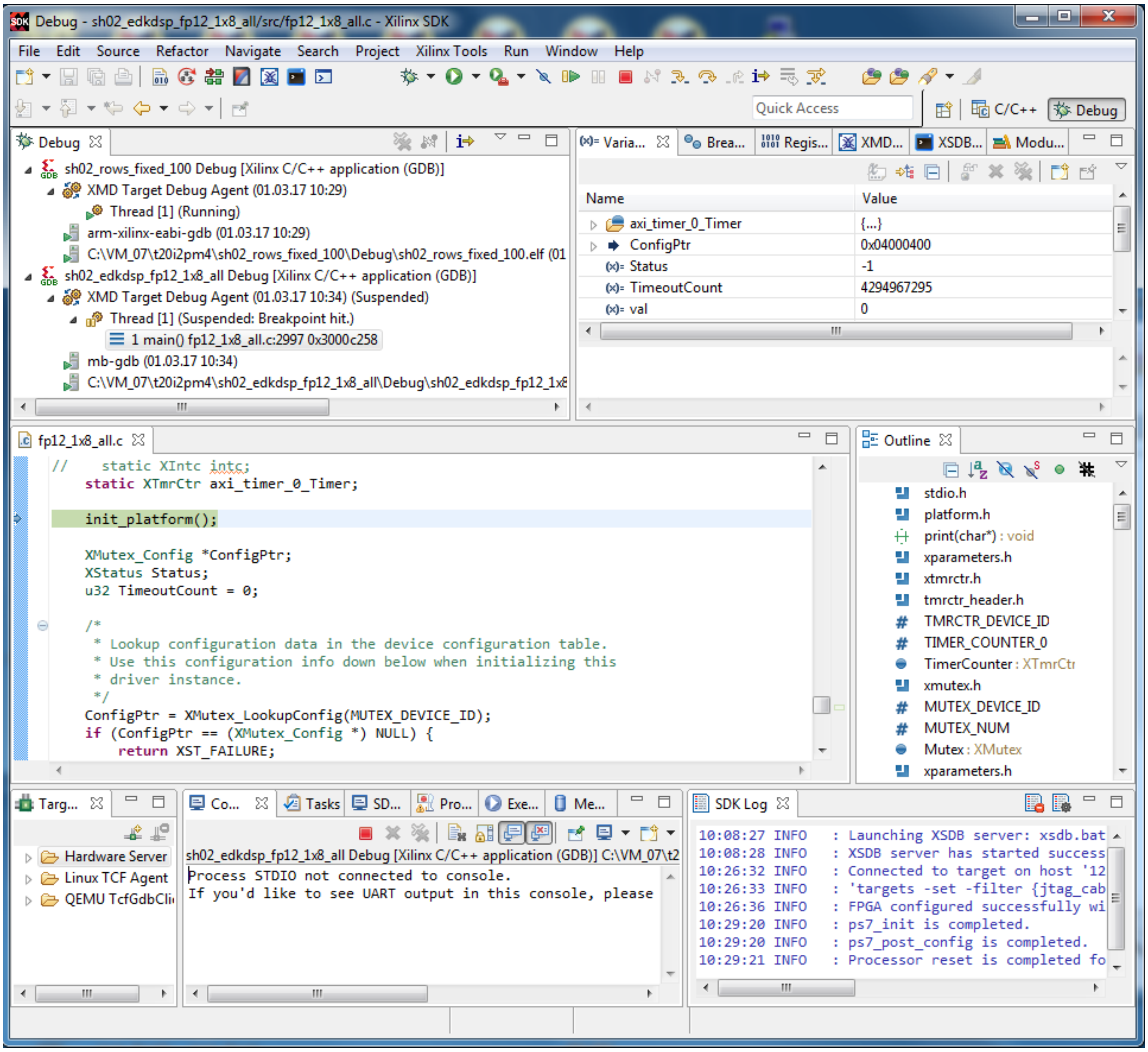
We are downloading program for MicroBlaze by JTAG, while ARM is already running.

- Unselect “Run ps7_init”
- Unselect “Run ps7_post_config”
- Select No reset

Click on “Apply” button. See Figure 24.

Click on “Debug” to download the **sh02_edkdsp_fp12_1x8_all.elf** to DDR3 as program for MicroBlaze.

The debugger will download this code by JTAG (connected to PC by the USB cable shared with the serial terminal) and stop MicroBlaze at the first executable instruction. See Figure 25.



```
sh02_rows_fixed_100 Debug [Xilinx C/C++ application (GDB)]
  XMD Target Debug Agent (01.03.17 10:29)
    Thread [1] (Running)
  arm-xilinx-eabi-gdb (01.03.17 10:29)
  C:\VM_07\t20i2pm4\sh02_rows_fixed_100\Debug\sh02_rows_fixed_100.elf (01
sh02_edkdsp_fp12_1x8_all Debug [Xilinx C/C++ application (GDB)]
  XMD Target Debug Agent (01.03.17 10:34) (Suspended)
    Thread [1] (Suspended: Breakpoint hit.)
      1 main() fp12_1x8_all.c:2997 0x3000c258
  mb-gdb (01.03.17 10:34)
  C:\VM_07\t20i2pm4\sh02_edkdsp_fp12_1x8_all\Debug\sh02_edkdsp_fp12_1x8
```

Name	Value
axi_timer_0_Timer	{...}
ConfigPtr	0x04000400
Status	-1
TimeoutCount	4294967295
val	0

fp12_1x8_all.c

```
//    static XIntc intc;
    static XTmrCtr axi_timer_0_Timer;

    init_platform();

    XMutex_Config *ConfigPtr;
    XStatus Status;
    u32 TimeoutCount = 0;

    /*
     * Lookup configuration data in the device configuration table.
     * Use this configuration info down below when initializing this
     * driver instance.
     */
    ConfigPtr = XMutex_LookupConfig(MUTEX_DEVICE_ID);
    if (ConfigPtr == (XMutex_Config *) NULL) {
        return XST_FAILURE;
```

```
stdio.h
platform.h
print(char*) : void
xparameters.h
xtmrctr.h
tmrctr_header.h
TMRCTR_DEVICE_ID
TIMER_COUNTER_0
TimerCounter : XTmrCtr
xmutex.h
MUTEX_DEVICE_ID
MUTEX_NUM
Mutex : XMutex
xparameters.h
```

```
sh02_edkdsp_fp12_1x8_all Debug [Xilinx C/C++ application (GDB)] C:\VM_07\t2
Process STDIO not connected to console.
If you'd like to see UART output in this console, please
```

```
10:08:27 INFO  : Launching XSDB server: xsdb.bat
10:08:28 INFO  : XSDB server has started success
10:26:32 INFO  : Connected to target on host '12
10:26:33 INFO  : 'targets -set -filter {jtag_cab
10:26:36 INFO  : FPGA configured successfully wi
10:29:20 INFO  : ps7_init is completed.
10:29:20 INFO  : ps7_post_config is completed.
10:29:21 INFO  : Processor reset is completed fo
```

Figure 25: MicroBlaze application is loaded and debugger stops on the first instruction.

- The ARM SW thread is running.
- The MicroBlaze SW thread is currently suspended at breakpoint hit. See Figure 25.

Click on the |> icon to start the execution of MicroBlaze. The handshake between ARM and MicroBlaze by HW mutex IP is completed and both processors start to run. Interrupts are disabled.

ARM will initiate the Full HD Toshiba video sensor and all Video processing IP cores. It controls status of VDMA units and sets correct pointers to the active video frame buffers in SW. Video processing is performed by HLS IP cores in HW. Data are moved from video frame buffers to HW and back to output video frame buffers by HW data mover IPs. All video I/O IP cores are set-up by the ARM SW via the Axi-Lite interface.

Input HW data movers act as HW masters controlling the DMA engines moving data from DDR3 as input to the chain(s) of HLS IP cores. Output HW data movers act as HW masters controlling the DMA engines moving data from the output of chain(s) of HLS IP cores to DDR3 output video frame buffers.

The MicroBlaze processor executes in parallel program from DDR3, downloads firmware and cares about moving of data to the (8xSIMD) EdkDSP floating point accelerator.

Microblaze application tests all basic floating point operations supported by the EdkDSP and compares the EdkDSP results with the MicroBlaze floating point reference results.

In next stage it programs EdkDSP to perform FIR filter and LMS adaptive filter.

The performance of the combination of MicroBlaze with EdkDSP accelerator is measured by HW timer instantiated as MicroBlaze AXI-Lite IP core. See Figure 27, lines ending with MFLOPs.

- All evaluation demos can be also compiled into release versions with optimisation set to -O2 or -O3. These optimisations can be set for ARM and for MicroBlaze.

- Demos like sh01_rows_fixed_100 work on complete frame with single HW accelerator data path. Demos like sh01_rows_resize_25_to_100 work with identical HW, but the ARM SW dynamically scales the number of lines to be processed. This is scaling from ¼ of frame to the complete frame.

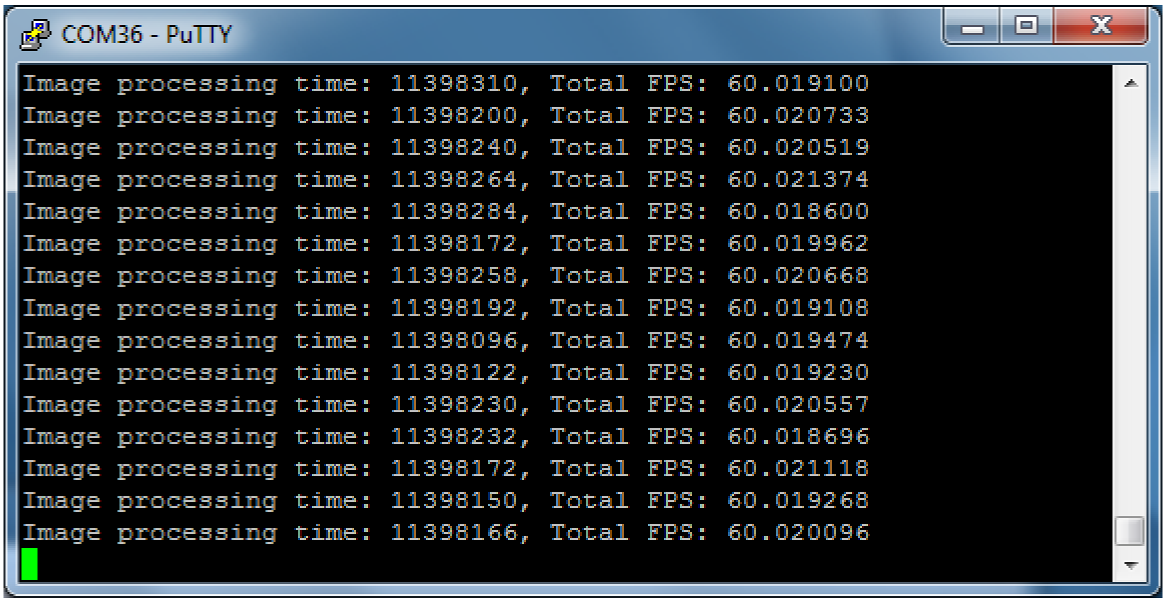
  Part of the frame which is not processed in HW is propagated to the output via the cyclic structure of the 8 video frame buffers.

  The HW data movers are instructed about the number of lines to be processed. SW writes this information to an AXI-lite configuration register of the data mover IP core.

- Demos sh02_rows_fixed_100 and sh02_rows_resize_25_to_100 work with 2 data paths.

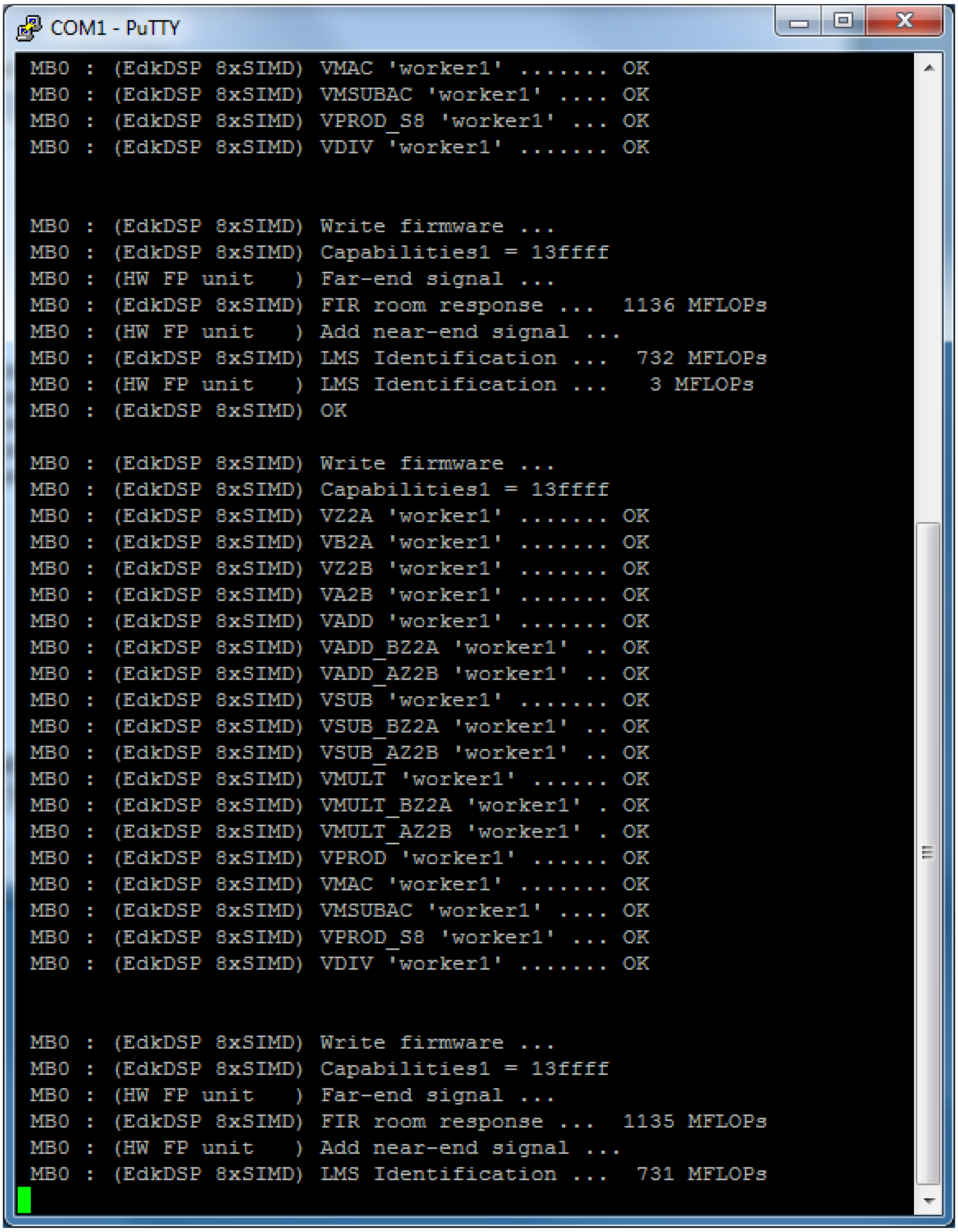
- Demos sh03_rows_fixed_100 and sh03_rows_resize_25_to_100 work with 3 data paths.

- Demo md01_rows_fixed_100 works with one HW video processing chain with fixed set of processed lines.



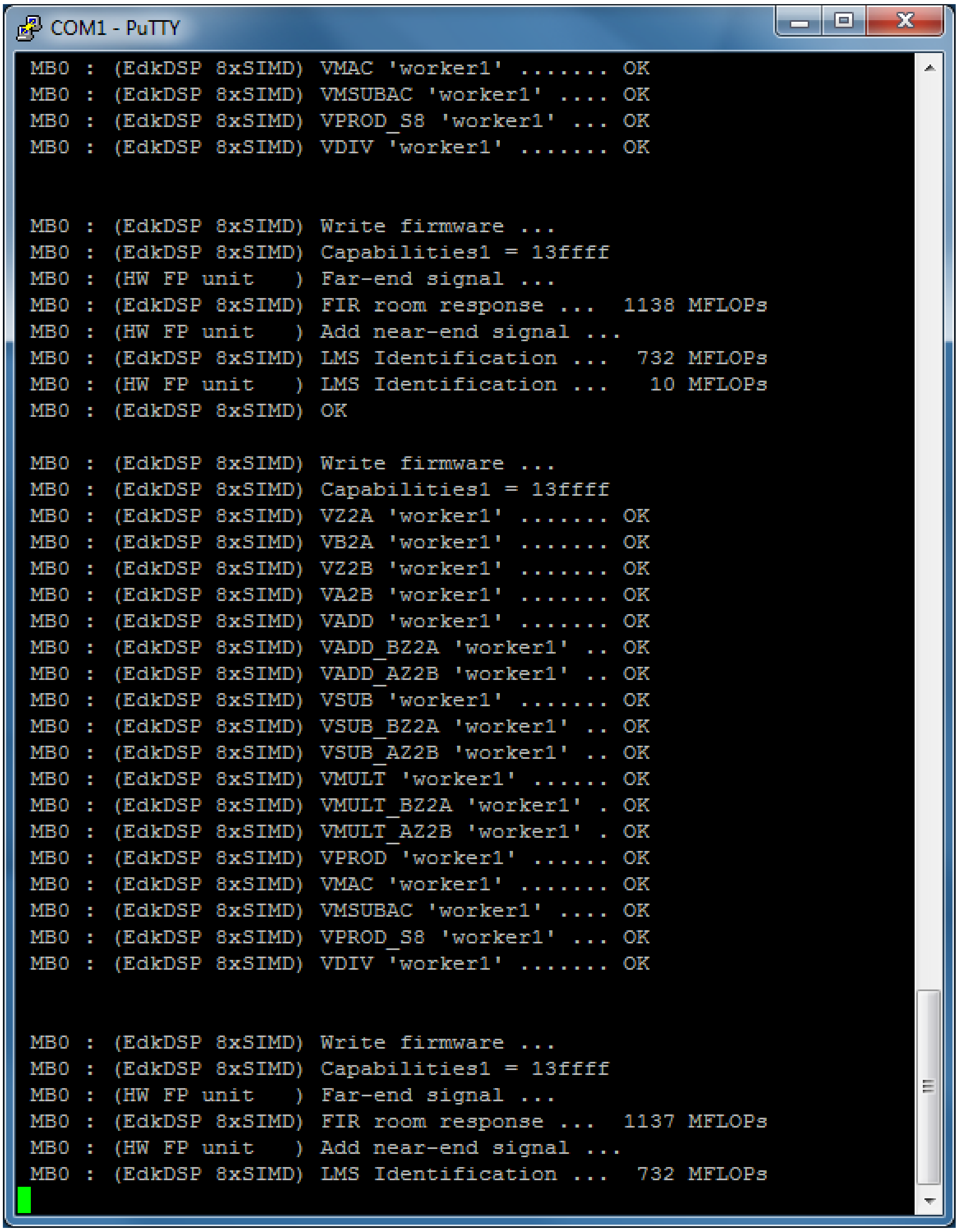
```
Image processing time: 11398310, Total FPS: 60.019100
Image processing time: 11398200, Total FPS: 60.020733
Image processing time: 11398240, Total FPS: 60.020519
Image processing time: 11398264, Total FPS: 60.021374
Image processing time: 11398284, Total FPS: 60.018600
Image processing time: 11398172, Total FPS: 60.019962
Image processing time: 11398258, Total FPS: 60.020668
Image processing time: 11398192, Total FPS: 60.019108
Image processing time: 11398096, Total FPS: 60.019474
Image processing time: 11398122, Total FPS: 60.019230
Image processing time: 11398230, Total FPS: 60.020557
Image processing time: 11398232, Total FPS: 60.018696
Image processing time: 11398172, Total FPS: 60.021118
Image processing time: 11398150, Total FPS: 60.019268
Image processing time: 11398166, Total FPS: 60.020096
```

Figure 26: ARM is running in debug mode. It indicates the number of frames per second.



```
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


MB0 : (EdkDSP 8xSIMD) Write firmware ...
MB0 : (EdkDSP 8xSIMD) Capabilities1 = 13ffff
MB0 : (HW FP unit    ) Far-end signal ...
MB0 : (EdkDSP 8xSIMD) FIR room response ...  1136 MFLOPs
MB0 : (HW FP unit    ) Add near-end signal ...
MB0 : (EdkDSP 8xSIMD) LMS Identification ...  732 MFLOPs
MB0 : (HW FP unit    ) LMS Identification ...   3 MFLOPs
MB0 : (EdkDSP 8xSIMD) OK

MB0 : (EdkDSP 8xSIMD) Write firmware ...
MB0 : (EdkDSP 8xSIMD) Capabilities1 = 13ffff
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


MB0 : (EdkDSP 8xSIMD) Write firmware ...
MB0 : (EdkDSP 8xSIMD) Capabilities1 = 13ffff
MB0 : (HW FP unit    ) Far-end signal ...
MB0 : (EdkDSP 8xSIMD) FIR room response ...  1135 MFLOPs
MB0 : (HW FP unit    ) Add near-end signal ...
MB0 : (EdkDSP 8xSIMD) LMS Identification ...  731 MFLOPs
```

Figure 27: MicroBlaze is running in debug mode. It indicates MFLOPs.



```
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


MB0 : (EdkDSP 8xSIMD) Write firmware ...
MB0 : (EdkDSP 8xSIMD) Capabilities1 = 13ffff
MB0 : (HW FP unit    ) Far-end signal ...
MB0 : (EdkDSP 8xSIMD) FIR room response ...  1138 MFLOPs
MB0 : (HW FP unit    ) Add near-end signal ...
MB0 : (EdkDSP 8xSIMD) LMS Identification ...  732 MFLOPs
MB0 : (HW FP unit    ) LMS Identification ...   10 MFLOPs
MB0 : (EdkDSP 8xSIMD) OK

MB0 : (EdkDSP 8xSIMD) Write firmware ...
MB0 : (EdkDSP 8xSIMD) Capabilities1 = 13ffff
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


MB0 : (EdkDSP 8xSIMD) Write firmware ...
MB0 : (EdkDSP 8xSIMD) Capabilities1 = 13ffff
MB0 : (HW FP unit    ) Far-end signal ...
MB0 : (EdkDSP 8xSIMD) FIR room response ...  1137 MFLOPs
MB0 : (HW FP unit    ) Add near-end signal ...
MB0 : (EdkDSP 8xSIMD) LMS Identification ...  732 MFLOPs
```

Figure 28: MicroBlaze is running in release mode. It indicates MFLOPs.

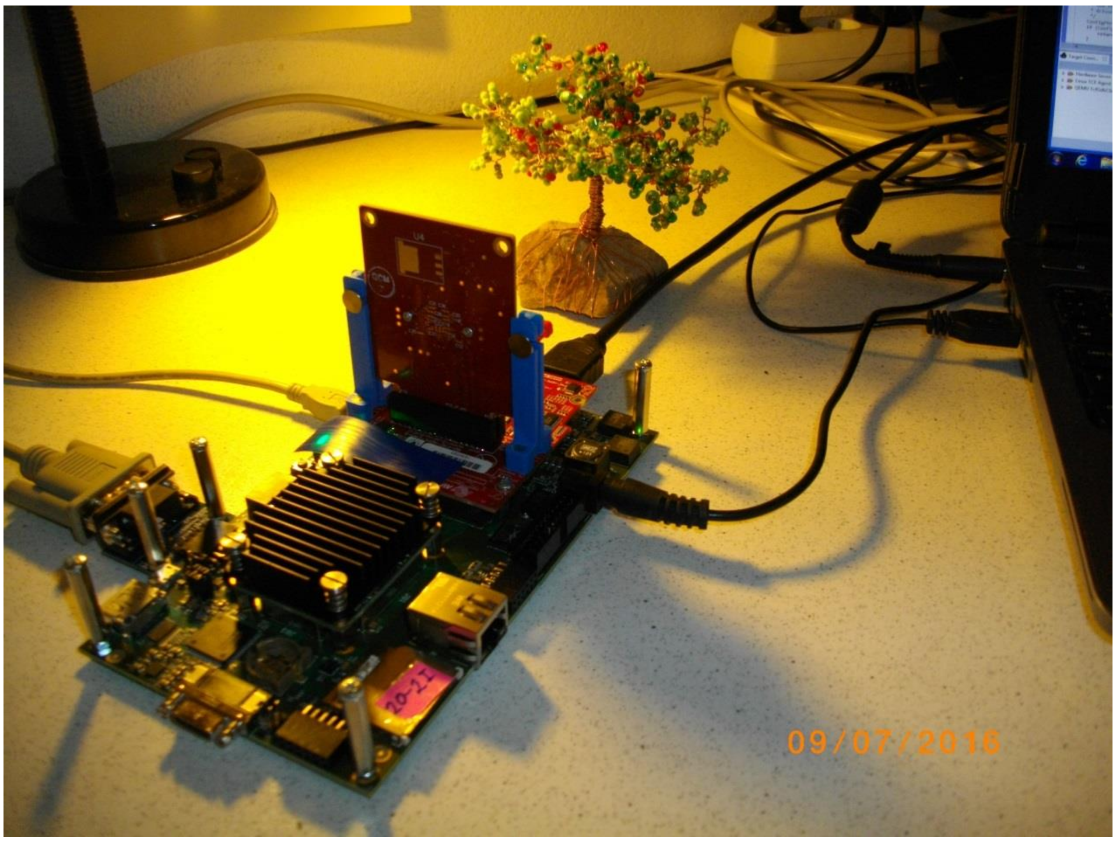


Figure 29: Accelerated edge detection Full HD Toshiba sensor and Zynq with EdkDSP IP.

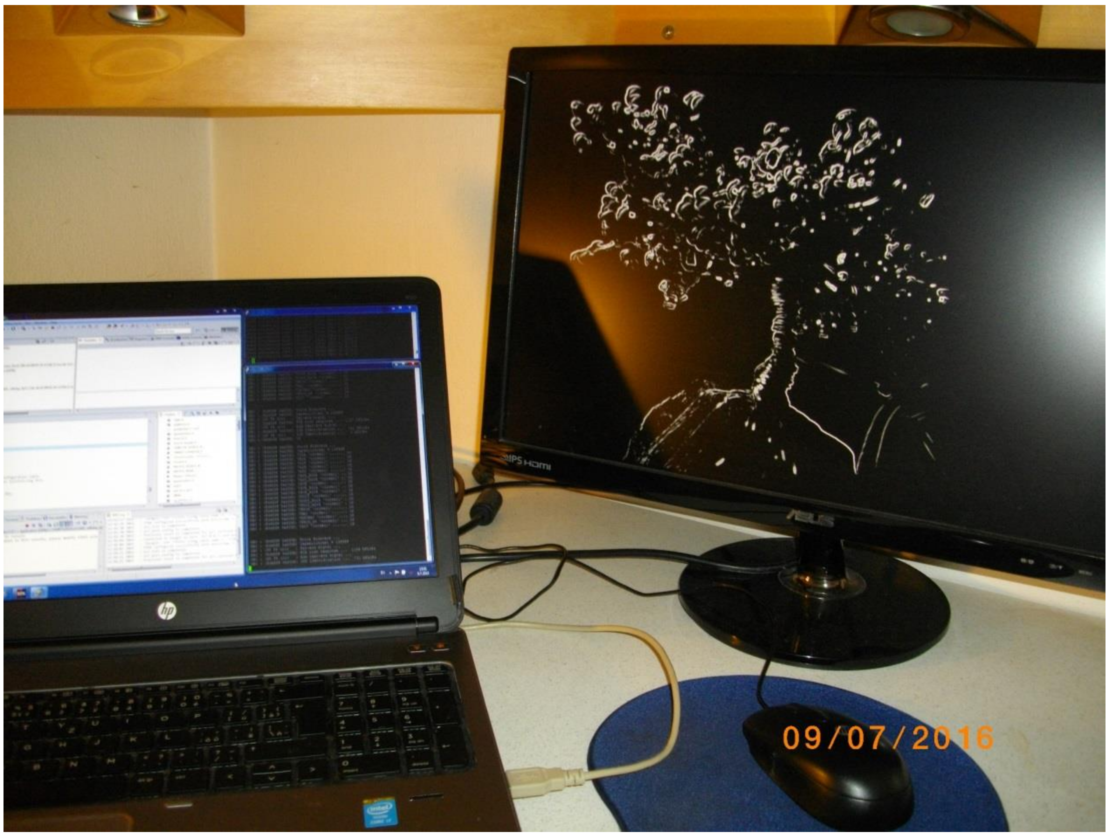


Figure 30: Edge detection (sh02 - Sobel filters, 2 variable HW data paths) output on HDMI monitor.

2.4 Synchronisation of user C code with the video processing HW accelerators

This section describes synchronisation of ARM C code with parallel video processing HW accelerators.

Two programming models are described:

- User defined synchronisation with parallel HW data paths (SW barrier).
- Internal synchronisation with parallel HW data paths.

User defined synchronisation with parallel HW data paths (SW barrier)

Consider **sh02_rows_fixed_100** project as an example. Two HW data paths perform edge detection in parallel on 2 separate areas of one video frame.

C:\VM_07\t20i2tm4\sh02_rows_fixed_100\sobel\img_filters.c

```
#include <stdio.h>
#include "frame_size.h"
#include "hw_sobel.h"

void img_process(unsigned short *fb_in, unsigned short *fb_out) {


#pragma SDS async(2)
      _p0_sobel_filter_htile2_0(fb_in + (NUMTILEROWS)*NUMPADCOLS, fb_out +
(NUMTILEROWS)*NUMPADCOLS, NUMTILEROWS);

#pragma SDS async(1)
      _p0_sobel_filter_htile1_0(fb_in, fb_out, (NUMTILEROWS+2));

// Parallel ARM code here

      sds_wait(2);
      sds_wait(1);
}
```

Figure 31: ARM C function with an user defined synchronization of 2 HW accelerators.

Arm C code calls two functions (see Figure 11 and Figure 31):

_p0_sobel_filter_htile2_0()   // Not blocking, Starts HW path 2
_p0_sobel_filter_htile1_0()   // Not blocking, Starts HW path 1.

They correspond to two HW video acceleration paths. These functions are independent. Each of functions only starts its HW data path. Both functions are not blocking. Both functions have been defined in the original SDSoC 2015.4 project with the #pragma SDS async and exported in the libsh02.a static library. The synchronisation point (similar to a barrier in case of SW threads) is implemented separately by the two calls to the functions sds_wait(2) and sds_wait(1). These functions are blocking and each of the functions terminates when the corresponding HW accelerated data path is done. ARM processor can be programmed by user C code and this code can be executed in parallel to the started HW accelerated data paths.

Internal synchronisation with parallel HW data paths

Figure 32 presents interface to HW pipeline of accelerators with fixed data path used in md01 demo. The HW pipeline serves for direct communication of accelerators with the final synchronisation in function _p0_ext_0().

The sequence of function calls in is fixed. It cannot be changed. It is related to the md01 HW pipeline See Figure 32.

C:\VM_07\t20i2tm4\md01_rows_fixed_100\motion_detect\img_filters.c

```
#include <stdio.h>
#include "frame_size.h"
#include "hw_motion_detect.h"

unsigned short yc_data_prev[NUMROWS*NUMCOLS], yc_data_in[NUMROWS*NUMCOLS],
                yc_out_tmp1[NUMROWS*NUMCOLS], yc_out_tmp2[NUMROWS*NUMCOLS],
                yc_out_tmp3[NUMROWS*NUMCOLS], yc_out_tmp4[NUMROWS*NUMCOLS];
unsigned char sobel_curr[NUMROWS*NUMCOLS], sobel_prev[NUMROWS*NUMCOLS],
              motion_image_tmp1[NUMROWS*NUMCOLS], motion_image_tmp2[NUMROWS*NUMCOLS];

void img_process(unsigned short *rgb_data_prev, unsigned short *rgb_data_in,
                 unsigned short *rgb_data_out, int param0, int param1, int param2)
{
    unsigned char pass_through;
    unsigned char threshold = 100;
    pass_through = 0; // always force combo

    _p0_pad_1(rgb_data_prev, yc_data_prev);
    _p0_pad_0(rgb_data_in, yc_data_in);
    _p0_sobel_filter_pass_0(yc_data_in, sobel_curr, yc_out_tmp1);
    _p0_sobel_filter_0(yc_data_prev, sobel_prev);
    _p0_diff_image_0(sobel_curr, sobel_prev,yc_out_tmp1, yc_out_tmp2, motion_image_tmp1);
    _p0_median_char_filter_pass_0(threshold, motion_image_tmp1, yc_out_tmp2,
                              motion_image_tmp2, yc_out_tmp3);
    _p0_combo_image_0(pass_through, motion_image_tmp2, yc_out_tmp3, yc_out_tmp4);
    _p0_ext_0(yc_out_tmp4, rgb_data_out);
}
```

Figure 32: ARM C function with internal synchronization of a pipeline of 8 HW accelerators.

2.5 EdkDSP C compiler API

The PicoBlaze6 controller acts as programmable finite state machine in the (8xSIMD) EdkDSP accelerator. It sets the sequences of wide instructions for the 8xSIMD floating point data path of the EdkDSP accelerator.

The EdkDSP accelerators are connected to Xilinx MicroBlaze. The Microblaze processor is responsible for implementation of desired sequences of operations composed of:

- accelerator firmware programming,
- starting and synchronization of accelerators
- data communication.

These operations are supported by the Worker Abstraction Layer API. The API functions are summarized in Table 1.

Table 1: API for MicroBlaze C code

Function	Description
Init/Done functions	
wal_init_worker	Initiate and claim worker in an application
wal_done_worker	Cleanup and release data structures allocated for the worker
Basic control functions	
wal_reset_worker	Send hard reset to the worker. Set control part if the worker to the default state
wal_start_operation	Select and run preloaded firmware in the worker
wal_end_operation	Send request to stop worker operation. It is followed by request to worker reset
wal_is_busy	Test if the worker is currently busy (It is a non-blocking operation)
wal_mb2pb	Set control word to the worker
wal_pb2mb	Read status word of the worker
Functions for working with control memories	
wal_mb2cmem	Copy block of data from MicroBlaze user defined C array to the control memory shared by worker with MicroBlaze (worker firmware, 2 memories)
wal_cmem2mb	Copy block of data from control memory of worker shared with MicroBlaze to MicroBlaze user defined C array
Functions for working with data memories	
wal_mb2dmem	Copy block of data from MicroBlaze user defined C array to selected data memory of the worker shared with the MicroBlaze (for 8xSIMD EdkDSP – 24 memories)
wal_dmem2mb	Copy block of data from the selected data memory of the worker shared with the MicroBlaze (for 8xSIMD EdkDSP – 24 memories)
Common support functions	
wal_set_firmware	Copy worker firmware to selected position
wal_get_id	Read worker ID
wal_get_capabilities	Read worker capabilities
wal_get_license	Read worker license

Table 2: EdkDSP accelerator I/O API functions used by the PicoBlaze6 controller firmware

Function	Description
mb2pb_read_data	Read value from MicroBlaze (blocking, includes hands-hake with MicroBlaze)
pb2mb_write	Write data value from PicoBlaze to MicroBlaze (blocking, includes handshake)
pb2mb_eoc	Write data value from PicoBlaze to MicroBlaze and indicate the end of string flag (blocking, includes handshake with MicroBlaze)
pb2mb_req_reset	Write data value from PicoBlaze to MicroBlaze and indicate request from to reset PicoBlaze (blocking, includes handshake with MicroBlaze)
pb2mb_reset	Activate PicoBlaze reset from PicoBlaze program with MicroBlaze support (blocking, includes handshake with MicroBlaze)
led2pb	Read PicoBlaze LED port
btn2pb	Read PicoBlaze BTN port
hex_h	Write hexadecimal ascii representation of the high 4bit of the input 8bit argument from PicoBlaze to MicroBlaze (blocking, includes handshake with MicroBlaze)
hex_l	Write hexadecimal ascii representation of the low 4bit of the input 8bit argument from PicoBlaze to MicroBlaze (blocking, includes handshake with MicroBlaze)
pb2dfu_set	Write 8bit data to the PicoBlaze I/O port mem
pb2dfu_wait4hw	Wait for the end of the EdkDSP floating point, vector operation (blocking, waits for the end of the FP vector operation)
pb2lcd_ascii_char	Write to local 2x16 lcd display.

The last layer is the basic I/O library prepared for the (8xSIMD) EdkDSP accelerator for communication of its PicoBlaze6 controller with the MicroBlaze processor.

Each EdkDSP I/O API function has been optimised in assembler to provide low footprint and maximum performance at this low-level hardware layer. The EdkDSP I/O library functions are listed in Table 2.

EdkDSP C compiler

This section briefly describes how to use the UTIA embedded EdkDSP C compiler. This compiler runs as PetaLinux application on the evaluation board. It serves for compilation of simple C/ASM programs for the PicoBlaze6 controller. The evaluation package includes also already compiled firmware files for the PicoBlaze6 controller in form of headers in the MicroBlaze SDK projects. These files can be used for the first evaluations of the 8xSIMD EdkDSP accelerator IP. The EdkDSP compiler is provided in form of SD card image in the directory:

c:\VM_07\t20i2tm4\EdkDSP_SD\

(1) In PC, copy content of the directory to the SD card. Insert this SD card with the PetaLinux image and the C/ASM source code files and the EdkDSP compiler tools to the TE0701 carrier board.

(2) Reset the carrier board.
- The boot process of the PetaLinux 2017.1 OS will start.
- Type the linux user name and password:

```
root
root
```

- Mount SD card to the directory (See Figure 33) /mnt by typing:

```
mount   /dev/mmcblk0p1   /mnt
```

- Change directory to /mnt

```
cd /mnt
```

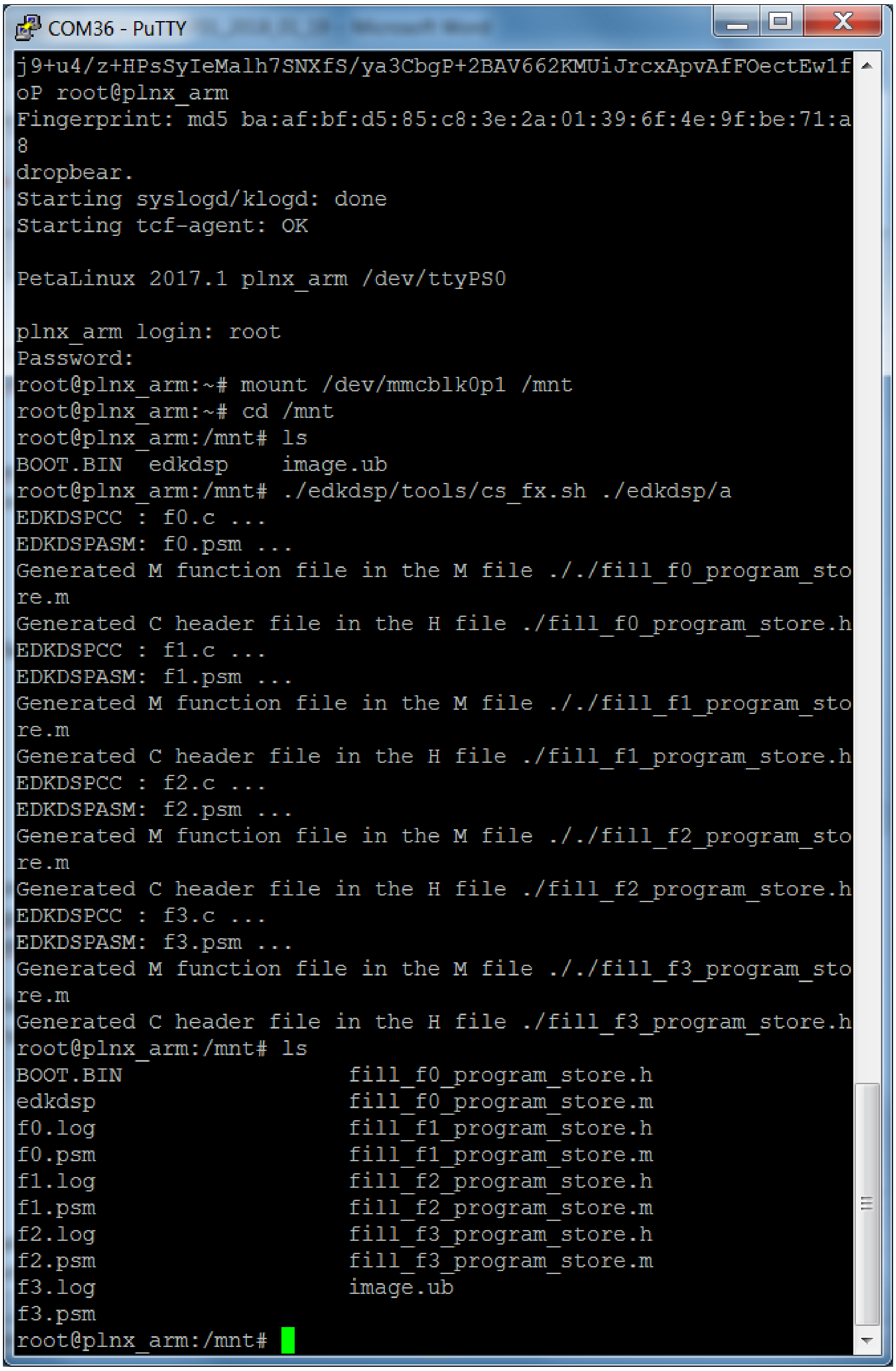
-Compile firmware for the PicoBlaze6 by the EdkDSP C compiler (see Figure 33) :

```
./edkdsp/tools/cs_fx.sh   ./edkdsp/a
(or ./edkdsp/tools/cs_fx.sh ./edkdsp/b  or ./edkdsp/tools/cs_fx.sh ./edkdsp/c   etc.)
```

Generated header files with PicoBlaze6 firmware for the standalone target EdkDSP IP target are created and stored in the SD card root directory:

```
./fill_f0_program_store.h
./fill_f1_program_store.h
./fill_f2_program_store.h
./fill_f3_program_store.h
```

These headers serve for the standalone MicroBlaze projects. Headers are compiled into the MicroBlaze standalone application as described above.



```
j9+u4/z+HPsSyIeMalh7SNXfS/ya3CbgP+2BAV662KMUiJrcxApvAfFOectEw1f
oP root@plnx_arm
Fingerprint: md5 ba:af:bf:d5:85:c8:3e:2a:01:39:6f:4e:9f:be:71:a
8
dropbear.
Starting syslogd/klogd: done
Starting tcf-agent: OK

PetaLinux 2017.1 plnx_arm /dev/ttyPS0

plnx_arm login: root
Password:
root@plnx_arm:~# mount /dev/mmcblk0p1 /mnt
root@plnx_arm:~# cd /mnt
root@plnx_arm:/mnt# ls
BOOT.BIN  edkdsp    image.ub
root@plnx_arm:/mnt# ./edkdsp/tools/cs_fx.sh ./edkdsp/a
EDKDSPCC : f0.c ...
EDKDSPASM: f0.psm ...
Generated M function file in the M file ././fill_f0_program_sto
re.m
Generated C header file in the H file ./fill_f0_program_store.h
EDKDSPCC : f1.c ...
EDKDSPASM: f1.psm ...
Generated M function file in the M file ././fill_f1_program_sto
re.m
Generated C header file in the H file ./fill_f1_program_store.h
EDKDSPCC : f2.c ...
EDKDSPASM: f2.psm ...
Generated M function file in the M file ././fill_f2_program_sto
re.m
Generated C header file in the H file ./fill_f2_program_store.h
EDKDSPCC : f3.c ...
EDKDSPASM: f3.psm ...
Generated M function file in the M file ././fill_f3_program_sto
re.m
Generated C header file in the H file ./fill_f3_program_store.h
root@plnx_arm:/mnt# ls
BOOT.BIN                 fill_f0_program_store.h
edkdsp                   fill_f0_program_store.m
f0.log                   fill_f1_program_store.h
f0.psm                   fill_f1_program_store.m
f1.log                   fill_f2_program_store.h
f1.psm                   fill_f2_program_store.m
f2.log                   fill_f3_program_store.h
f2.psm                   fill_f3_program_store.m
f3.log                   image.ub
f3.psm
root@plnx_arm:/mnt#
```

Figure 33: EdkDSP C/ASM compilation of firmware on the SD card.

(3) Remove SD card and copy generated firmware to the SDK 2015.4 MicroBlaze project.

Recompile SDK project in the PC and test new firmware on the evaluation board. Recompile the MicroBlaze project “**sh02_edkdsp_fp12_1x8_all**”.

The compiled firmware for the (8xSIMD) EdkDSP IP is used by the MicroBlaze C code of the demo as data for the runtime (re)configurations of the (8xSIMD) EdkDSP accelerator IP with changed firmware of the PicoBlaze6 controller. The run time change of firmware is demonstrated by swapping of firmware for computation of FIR and LMS filters in the EdkDSP accelerator IP in all demos in the evaluation package.

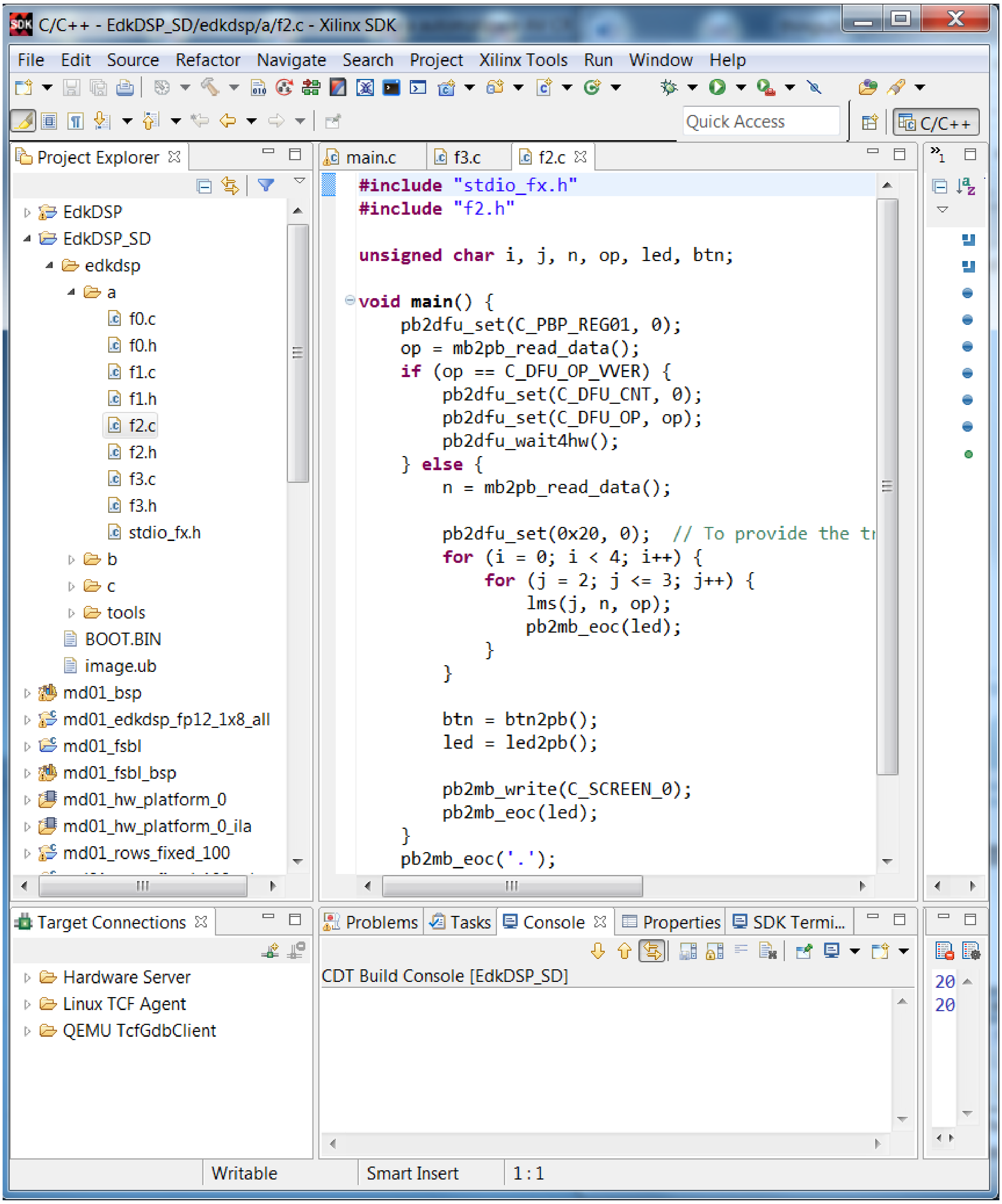
The evaluation design used in this application note works with single instance of the (8xSIMD) EdkDSP floating point accelerator IP core **bce_fp12_1x8_40**.

UTIA maintains four grades **[10|20|30|40]** of the (8xSIMD) EdkDSP accelerator IP. Cores differ in HW-supported vector floating point computing capabilities:

- **bce_fp12_1x8_10** is area optimized and supports local vector data transfers (HW supported 8xSIMD transfers inside of the accelerator IP) and vector floating point operations FPADD, FPSUB in 8xSIMD data paths.
- **bce_fp12_1x8_20** performs identical operations as bce_fp12_1x8_10 plus the vector floating point MAC operations in 8xSIMD data paths. MAC is supported for length of vectors 1 up to 13. This accelerator is optimized for applications like floating point matrix multiplication with one row and column dimensions <= 13.
- **bce_fp12_1x8_30** supports identical operations as bce_fp12_1x8_0_20 plus HW accelerated computation the floating point vector by vector dot products performed in 8xSIMD data paths. It is optimized for parallel computation of up to 8 FIR or LMS filters, each with size up to 250 coefficients. It is also efficient in case of floating point matrix by matrix multiplications, where one of the dimensions is large (from 11 to 250).
- **bce_fp12_1x8_40** supports identical operations as bce_fp12_1x8_30 plus an additional HW support of dot product. It is computed in 8xSIMD data paths with HW-supported pipeline wind-up into single scalar result. This result is propagated into all 8 SIMD data planes.

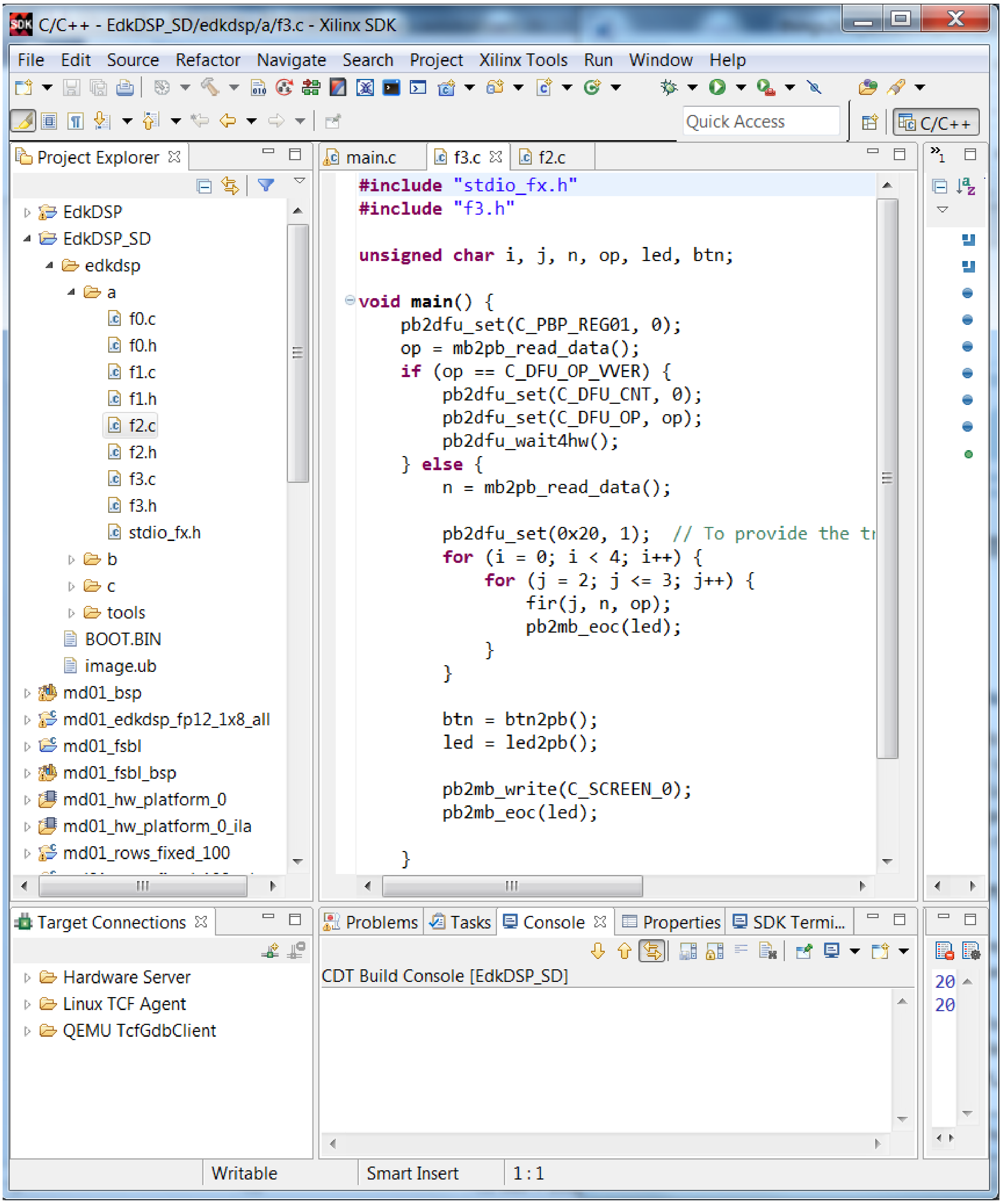
All **bce_fp12_1x8_[10|20|30|40]** accelerators IP cores support single data path for, pipelined, floating-point division (FPDIV) with vector operands taken from the first SIMD plain and the result vector propagated into all 8 SIMD data plains.

All **bce_fp12_1x8_[10|20|30|40]** accelerator IP core versions are suitable for applications like adaptive normalised NLMS filters, Square-root-free versions of adaptive RLS QR filters and Adaptive RLS LATTICE filters.



```
#include "stdio_fx.h"
#include "f2.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();

        pb2dfu_set(0x20, 0);  // To provide the tr
        for (i = 0; i < 4; i++) {
            for (j = 2; j <= 3; j++) {
                lms(j, n, op);
                pb2mb_eoc(led);
            }
        }

        btn = btn2pb();
        led = led2pb();

        pb2mb_write(C_SCREEN_0);
        pb2mb_eoc(led);
    }
    pb2mb_eoc('.');
```

Figure 34: C listing of the LMS filter firmware for the EdkDSP IP.



```c
#include "stdio_fx.h"
#include "f3.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();

        pb2dfu_set(0x20, 1);  // To provide the tr
        for (i = 0; i < 4; i++) {
            for (j = 2; j <= 3; j++) {
                fir(j, n, op);
                pb2mb_eoc(led);
            }
        }

        btn = btn2pb();
        led = led2pb();

        pb2mb_write(C_SCREEN_0);
        pb2mb_eoc(led);

    }
```

Figure 35: C listing of the FIR filter firmware for the EdkDSP IP.

2.6 Debug of the EdkDSP accelerator IP firmware with the In-circuit Logic Analyser

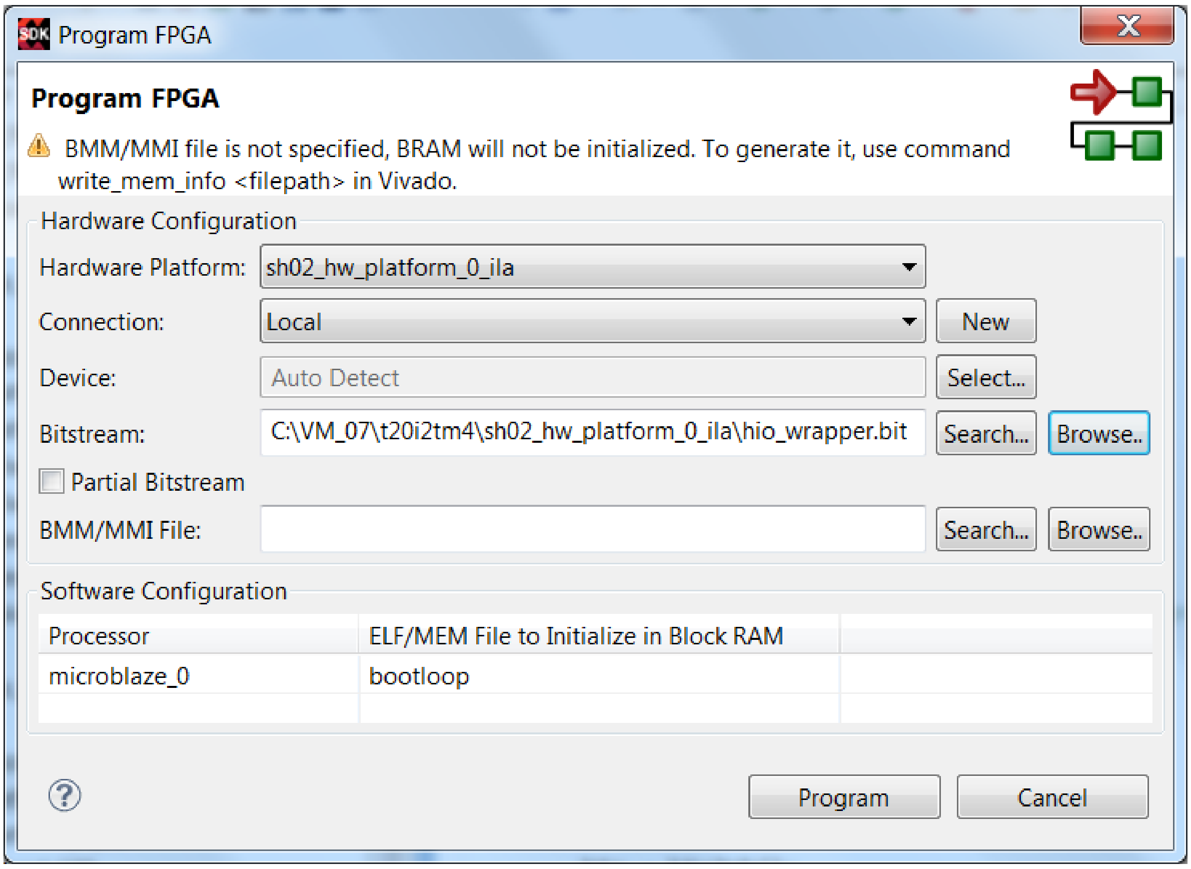
This application note includes evaluation version of designs with ARM Cortex A9 processor and MicroBlaze processor controlling single (8xSIMD) EdkDSP accelerator IP. The accelerator can be configured with the Xilinx

In-circuit Logic Analyser (ILA) for debug in the Vivado 2015.4 Lab Edition tool. The tool can be downloaded from Xilinx support webpage [8] for free. The platform with single (8xSIMD) EdkDSP accelerators and ILA support is compiled in the directory:

C:\VM_07\ t20i2tm4\sh02_hw_platform_0_ila

You can repeat all evaluation and compilation steps as described in this application note for the demos without ILA, but use the bitstream from the ***_hw_platfor_0_ila** directory.

Example for the sh02 demo:
In SDK, select: **Xilinx Tools -> Program FPGA** select **"sh02_hw_platform_0_ila" (**instead of **"sh02_hw_platform_0").** Click on the "**Program**" button.



Program FPGA

BMM/MMI file is not specified, BRAM will not be initialized. To generate it, use command write_mem_info <filepath> in Vivado.

Hardware Configuration

Hardware Platform:	sh02_hw_platform_0_ila
Connection:	Local
Device:	Auto Detect
Bitstream:	C:\VM_07\t20i2tm4\sh02_hw_platform_0_ila\hio_wrapper.bit
Partial Bitstream	☐
BMM/MMI File:	

Software Configuration

Processor	ELF/MEM File to Initialize in Block RAM
microblaze_0	bootloop

Figure 36: Change the default hw_platform_0 to the hw_platform_0_ila.

The implemented In-Circuit Logic Analyser (ILA) stores 8192 samples of all output of the (8xSIMD) EdkDSP Accelerator debug ports.

The debug ports provide the basic visibility of the vector (8xSIMD) EdkDSP accelerator IP. Prepared debug ILA environment provides synchronised time records of addresses and schedule of executed floating point vector operations.

The processed floating point data are not stored. These data can be better analysed in the MicroBlaze debugger. MicroBlaze and its debugger can access all dual-ported memories of the (8xSIMD) EdkDSP accelerator IP at synchronising points defined in the MicroBlaze application software.

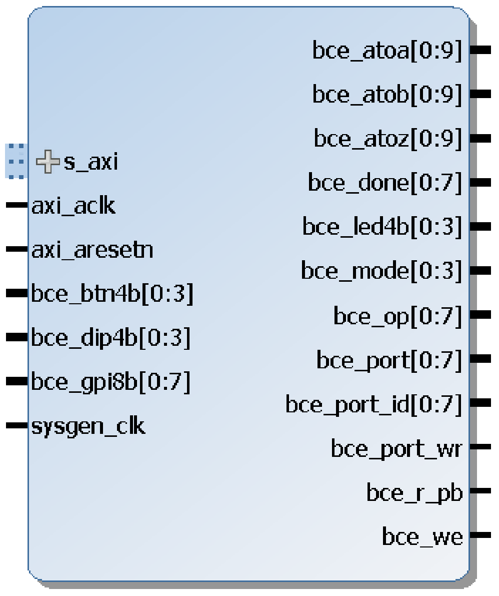


Figure 37: Debug ports of the (8xSIMD) EdkDSP floating point accelerator IP core.

Debug ports of the (8xSIMD) EdkDSP accelerator
All debug ports are stored with depth of 8192 samples with the sample frequency 120 MHz. See Figure 37.

• bce_atoa[0:9]	Memory A address (addressing 1024 32 bit floating point values)
• bce_atob[0:9]	Memory B address (addressing 1024 32 bit floating point values)
• bce_atoz[0:9]	Memory Z address (addressing 1024 32 bit floating point values)
• bce_done[0:7]	Vector operation in progress or finished
• bce_led4b[0:3]	4 bit output, intended for led signalling. (Unconnected to external pins).
• bce_mode[0:3]	Mode of the communication protocol PicoBlaze6 - MicroBlaze
• bce_op[0:7]	Vector operation to be performed.
• bce_port[0:7]	8 bit output port. (Unconnected to external pins).
• bce_port_id[0:7]	8 bit output External port address. Address space [0x0 ... 0x1F] are reserved for internal construction of the VLIW instruction to the 8xSIMD vector processing unit of the EdkDSP. Address space [0x20 ... 0xFF] can be used by the user.
• bce_port_wr	1 bit output. Write strobe for write of 8 bit data to the external port address.
• bce_r_pb	1 bit output. Reset of the PicoBlaze6.
• bce_we	1 bit output. Write strobe signals start of execution of a VLIW instruction by the 8xSIMD vector processing unit of the EdkDSP.

These debug ports are used for the real-time visualisation, debug and analysis of the computation implemented inside of the 8xSIMD vector processing unit of the (8xSIMD) EdkDSP accelerator IP. This makes easier to debug the compiled PicoBlaze6 firmware code.

The sequence of executed vector operations of the (8xSIMD) EdkDSP accelerator IP **bce_fp12_1x8_40** can be monitored at the **bce_op[0:7]** debug port. Code of these 8xSIMD vector operations is defined in Table 3:

Table 3: (8xSIMD) EdkDSP bce_fp12_1x8_40 accelerator vector operations.

Name in MicroBlaze C value (dec)	**8xSIMD Floating point Operation**
WAL_BCE_JK_VVER = 0	Return capabilities of the (8xSIMD) EdkDSP accelerator
WAL_BCE_JK_VZ2A = 1	8xSIMD copy $a_m[i] <= z_m[j]$; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VB2A = 2	8xSIMD copy $a_m[i] <= b_m[j]$; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VZ2B = 3	8xSIMD copy $b_m[i] <= z_m[j]$; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VA2B = 4	8xSIMD copy $b_m[i] <= a_m[j]$; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VADD = 5	8xSIMD add $z_m[i] <= a_m[j] + b_m[k]$]; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VADD_BZ2A = 6	8xSIMD add $a_m[i] <= b_m[j] + z_m[k]$]; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VADD_AZ2B = 7	8xSIMD add $b_m[i] <= a_m[j] + z_m[k]$]; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VSUB = 8	8xSIMD sub $z_m[i] <= a_m[j] - b_m[k]$; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VSUB_BZ2A = 9	8xSIMD sub $a_m[i] <= b_m[j] - z_m[k]$; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VSUB_AZ2B = 10	8xSIMD sub $b_m[i] <= a_m[j] - z_m[k]$; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VMULT = 11	8xSIMD mult $z_m[i] <= a_m[j] * b_m[k]$; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VMULT_BZ2A = 12	8xSIMD mult $a_m[i] <= b_m[j] * z_m[k]$; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VMULT_AZ2B = 13	8xSIMD mult $b_m[i] <= a_m[j] * z_m[k]$; m=1..8 IP core: 10,20,30,40
WAL_BCE_JK_VPROD = 14	8xSIMD vector products: IP core: 30,40 $z_m[i] <= a_m'[j..j+nn]*b_m[k..k+nn]$; m=1..8; nn range 1..255
WAL_BCE_JK_VMAC = 15	8xSIMD vector MACs: IP core: 20,30,40 $z_m[i..i+nn] <= z_m[i..i+nn] + a_m[j..j+nn] * b_m[k..jk+nn]$; nn range 1..13
WAL_BCE_JK_VMSUBAC = 16	8xSIMD vector MSUBACs IP core: 20,30,40 $z_m[i..i+nn] <= z_m[i..i+nn] - a_m[j..j+nn] * b_m[k..jk+nn]$; nn range 1..13
WAL_BCE_JK_VPROD_S8 = 17	8xSIMD vector product (extended) IP core: 40 $z_m[i] <= ((a_1'[j..j+nn]*b_1[k..k+nn]+a_2'[j..j+nn]*b_2[k..k+nn])$ $+ (a_3'[j..j+nn]*b_3[k..k+nn]+a_4'[j..j+nn]*b_4[k..k+nn]))$ $+$ $((a_5'[j..j+nn]*b_5[k..k+nn]+a_6'[j..j+nn]*b_6[k..k+nn])$ $+ (a_7'[j..j+nn]*b_7[k..k+nn]+a_8'[j..j+nn]*b_8[k..k+nn]))$; m=1..8; nn range 1..255
WAL_BCE_JK_VDIV = 20	vector division (extended) IP core: 10,20,30,40 $z_m[i] <= a_1[j] / b_1[k]$; m=1..8

2.7 Use of In-circuit Logic Analyser (ILA)

Start demo design with ILA from the Vivado 2015.4 SDK. Start both terminals. The AMP demo design is running.

It executes AMP demo SW on ARM, MicroBlaze with single (8xSIMD) EdkDSP accelerators (each with PicoBlaze6 reprogrammable firmware) and the ILA HW interface configured for debugging of the first EdkDSP accelerator.

Start Vivado Lab Edition 2015.4 and select "Open Hardware Manager". See Figure 38.

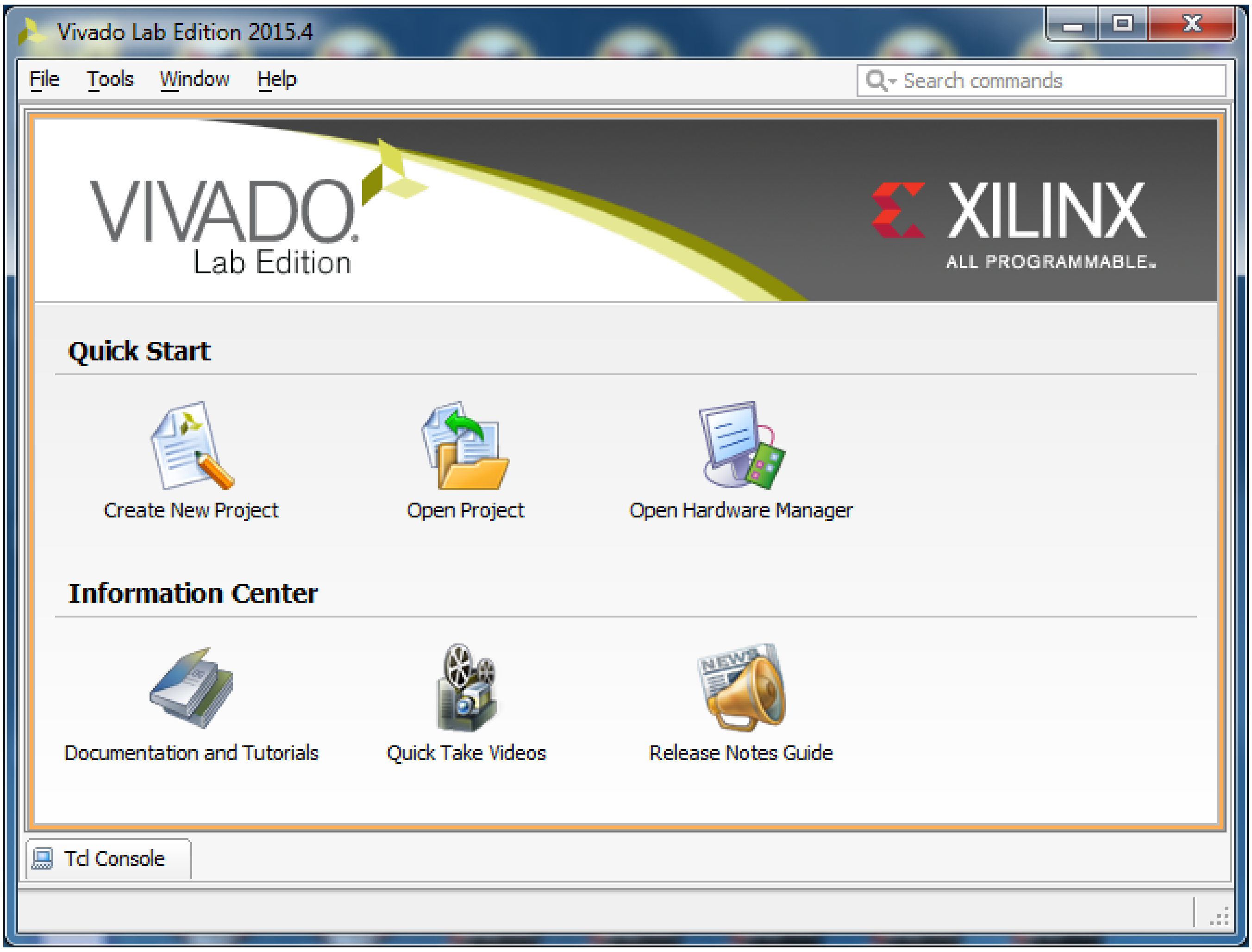


Figure 38: Vivado Lab Edition 2015.4.

Select: **Open Target** See Figure 39. Take all defaults by clicking **Next** button in coming screens.

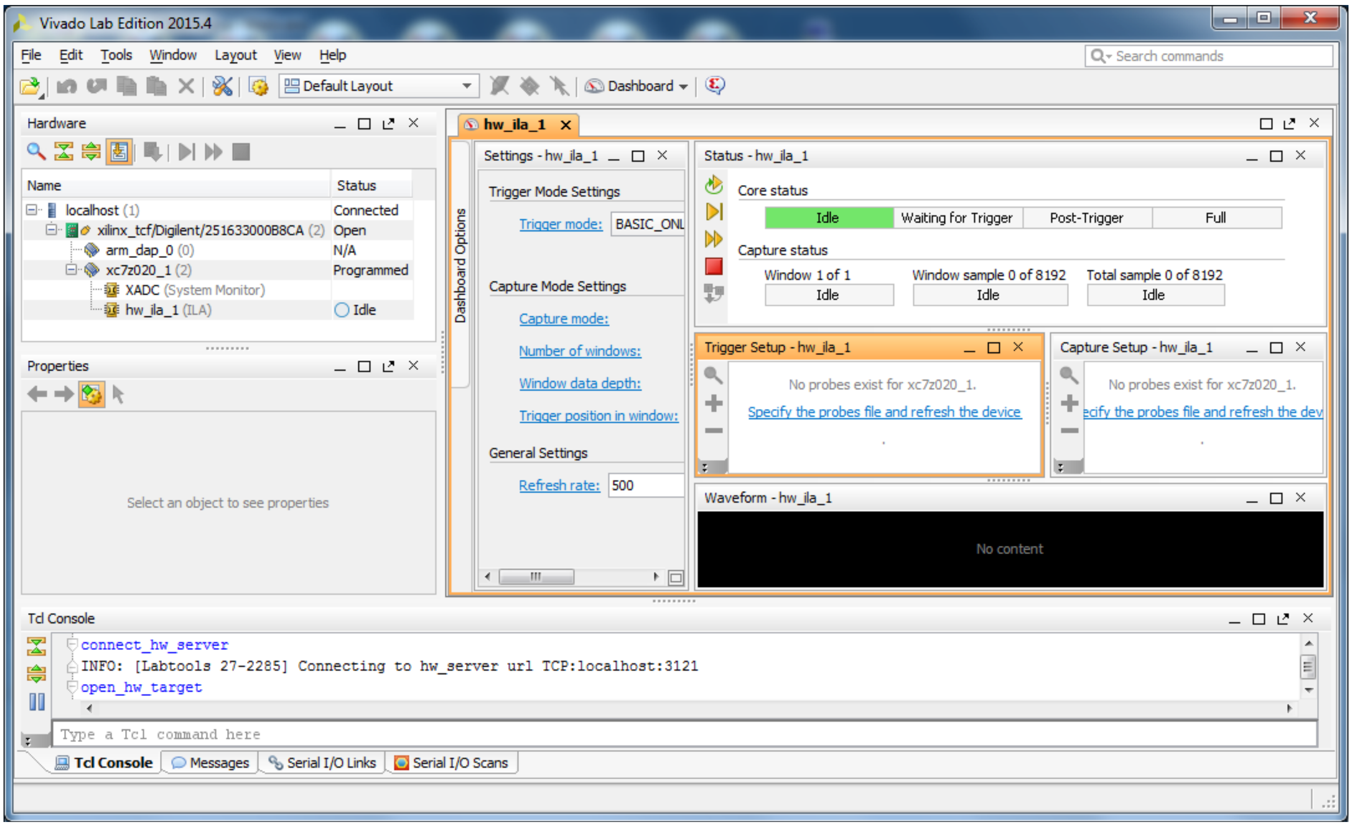
The Vivado Lab Edition 2015.4 is at this stage connected to the debugged board by the jtag.
Names and parameters of probes (see Figure 37) which can be captured by the ILA configuration on HW and visualised by Vivado Lab Edition 2015.4 are stored in file **debug_nets.ltx.**

In Vivado Lab Edition 2015.4, click on the "specify the probes file and refresh the device" link in the Trigger setup hw_ila_1 window.

Specify file

C:/VM_07/t20i2tm4/sh02_hw_platform_0_ila/ debug_nets.ltx
See Figure 40.

This will add the names and parameters of probes (see Figure 37) to the ILA Waveform window.
Use **+** to select probes used for triggering, and select the condition for trigger for each probe and their combinations (use AND as default).



Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/251633000B8CA (2)	Open
arm_dap_0 (0)	N/A
xc7z020_1 (2)	Programmed
XADC (System Monitor)	
hw_ila_1 (ILA)	Idle

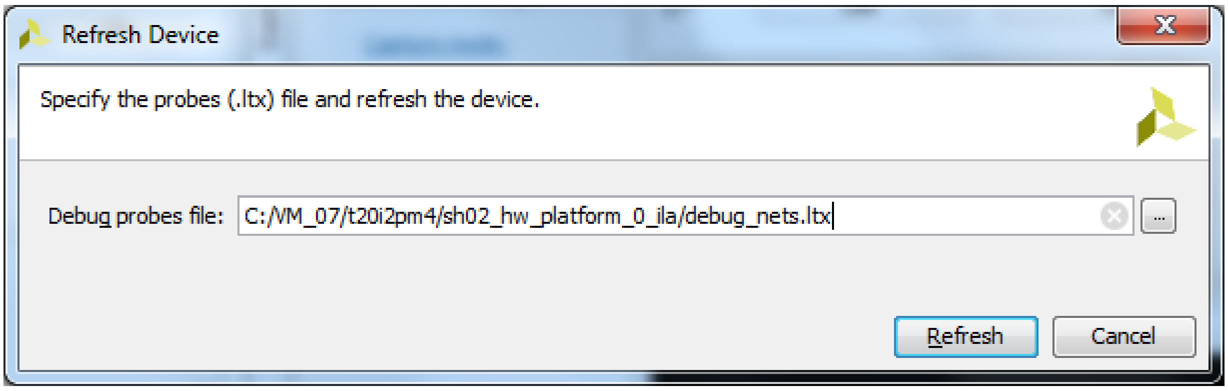
Trigger Mode Settings
Trigger mode: BASIC_ONL
Capture Mode Settings
Capture mode:
Number of windows:
Window data depth:
Trigger position in window:
General Settings
Refresh rate: 500

Core status: Idle | Waiting for Trigger | Post-Trigger | Full
Capture status
Window 1 of 1: Idle
Window sample 0 of 8192: Idle
Total sample 0 of 8192: Idle

No probes exist for xc7z020_1.
Specify the probes file and refresh the device

```
connect_hw_server
INFO: [Labtools 27-2285] Connecting to hw_server url TCP:localhost:3121
open_hw_target
```

Figure 39: Select Open Target.



Specify the probes (.ltx) file and refresh the device.

Debug probes file: C:/VM_07/t20i2pm4/sh02_hw_platform_0_ila/debug_nets.ltx

Figure 40: Select file with definition of probes present in HW.

Some of debug probes can be used to trigger the capturing of data. The ILA can be triggered from the firmware running on the PicoBlaze6 running inside of the 8xSIMD EdkDSP accelerator IP.

In SDK, open the **EdkDSP_SD/edkdsp/a/f3.c** file. See section of the **FIR** C code firmware. The code includes the additional call to the **pb2dfu_set()** function. We will use it for selective triggering of the ILA in this specified point of computation of the EdkDSP accelerator.

File **fill_f2_program_store.h** contains firmware resulting from compilation of C source code **f2.c** (see Figure 35).

```
…
                    pb2dfu_set(0x20, 1);  // To provide the trigger (0x01 on port 0x20) for the  ILA
                    for (i = 0; i < 4; i++) {
                              for (j = 2; j <= 3; j++) {
                                        fir(j, n, op);
                                        pb2mb_eoc(led);
                              }
                    }
                    …
```

In Vivado Lab Edition, in the ILA configuration page, change the trigger condition to:

(bce_port_wr ==1) AND (bce_port_id[0:7]==0x20) AND (bce_port[0:7]==**0x01**)

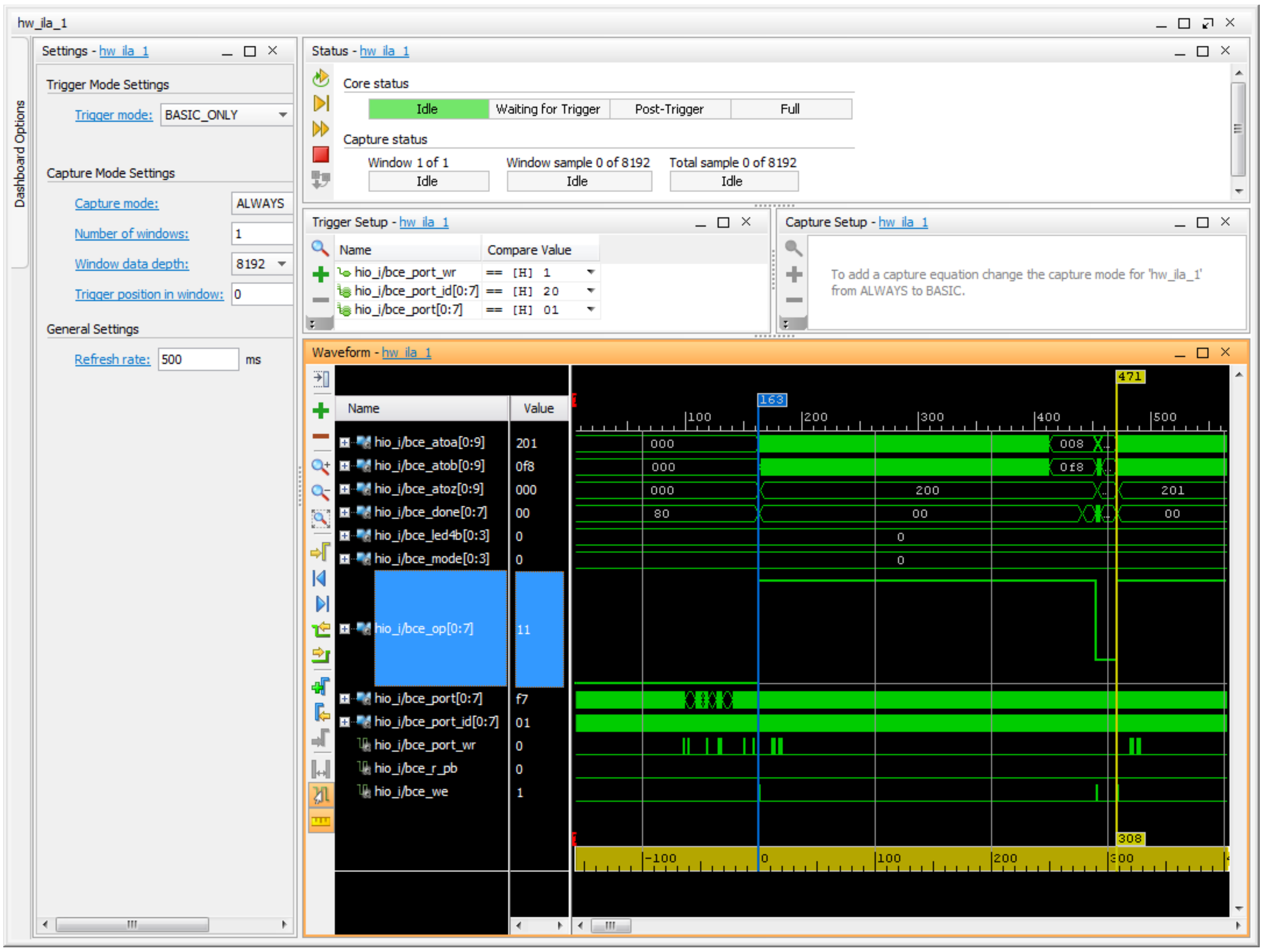
In Vivado Lab Edition 2015.4, arm the hw_ila_1 core by pressing **Run Trigger** button in **Hardware** window.

Armed hw_ila_1 core will wait until the recompiled EdkDSP firmware comes to the point, where PicoBlaze6 calls function **pb2dfu_set(0x20, 1).**

ILA core starts to capture 8K samples of all debug signals with the sampling rate 120 MHz. Data are captured and sent via jtag to Vivado Lab Edition 2015.4 for visualisation and analysis in the waveform window. This snapshot stores the detailed trace of the initial 8192 clock cycles of the FIR filter computation as defined by the SW displayed in Figure 35. The red trigger corresponds to the event. See Figure 41.

The user can zoom in the data and define additional markers. Selected markers indicate single step of the FIR filter. It takes 308 clock cycles (120 MHz = 8.333 ns clock period) to compute the vector product of two floating point vectors (coefficients and data), both with length 250*8=2000 elements and to update the input data vector (in a circular buffer).

This demonstrates how the Vivado Lab Edition 2015.4 [8] supports visibility and debug capabilities for the developer of the first (8xSIMD) EdkDSP accelerator firmware through the ILA core instantiated in the design.


hw_ila_1

Settings - hw_ila_1

Trigger Mode Settings

Trigger mode: BASIC_ONLY

Capture Mode Settings

Capture mode: ALWAYS

Number of windows: 1

Window data depth: 8192

Trigger position in window: 0

General Settings

Refresh rate: 500 ms

Status - hw_ila_1

Core status

Idle | Waiting for Trigger | Post-Trigger | Full

Capture status

Window 1 of 1 — Idle

Window sample 0 of 8192 — Idle

Total sample 0 of 8192 — Idle

Trigger Setup - hw_ila_1

Name	Compare Value
hio_i/bce_port_wr	== [H] 1
hio_i/bce_port_id[0:7]	== [H] 20
hio_i/bce_port[0:7]	== [H] 01

Capture Setup - hw_ila_1

To add a capture equation change the capture mode for 'hw_ila_1' from ALWAYS to BASIC.

Waveform - hw_ila_1

Name	Value
hio_i/bce_atoa[0:9]	201
hio_i/bce_atob[0:9]	0f8
hio_i/bce_atoz[0:9]	000
hio_i/bce_done[0:7]	00
hio_i/bce_led4b[0:3]	0
hio_i/bce_mode[0:3]	0
hio_i/bce_op[0:7]	11
hio_i/bce_port[0:7]	f7
hio_i/bce_port_id[0:7]	01
hio_i/bce_port_wr	0
hio_i/bce_r_pb	0
hio_i/bce_we	1

Figure 41: FIR filter waveforms after the trigger in Vivado Lab Edition 2015.4.

While the system is running, we can modify the trigger condition to capture the initial phase of the LMS filter. Filter runs on the same EdkDSP accelerator in a next stage of the application code.

In SDK, see **EdkDSP_SD/edkdsp/a/f2.c** code for the LMS filter on the identical 8xSIMD EdkDSP IP. See Figure 34.

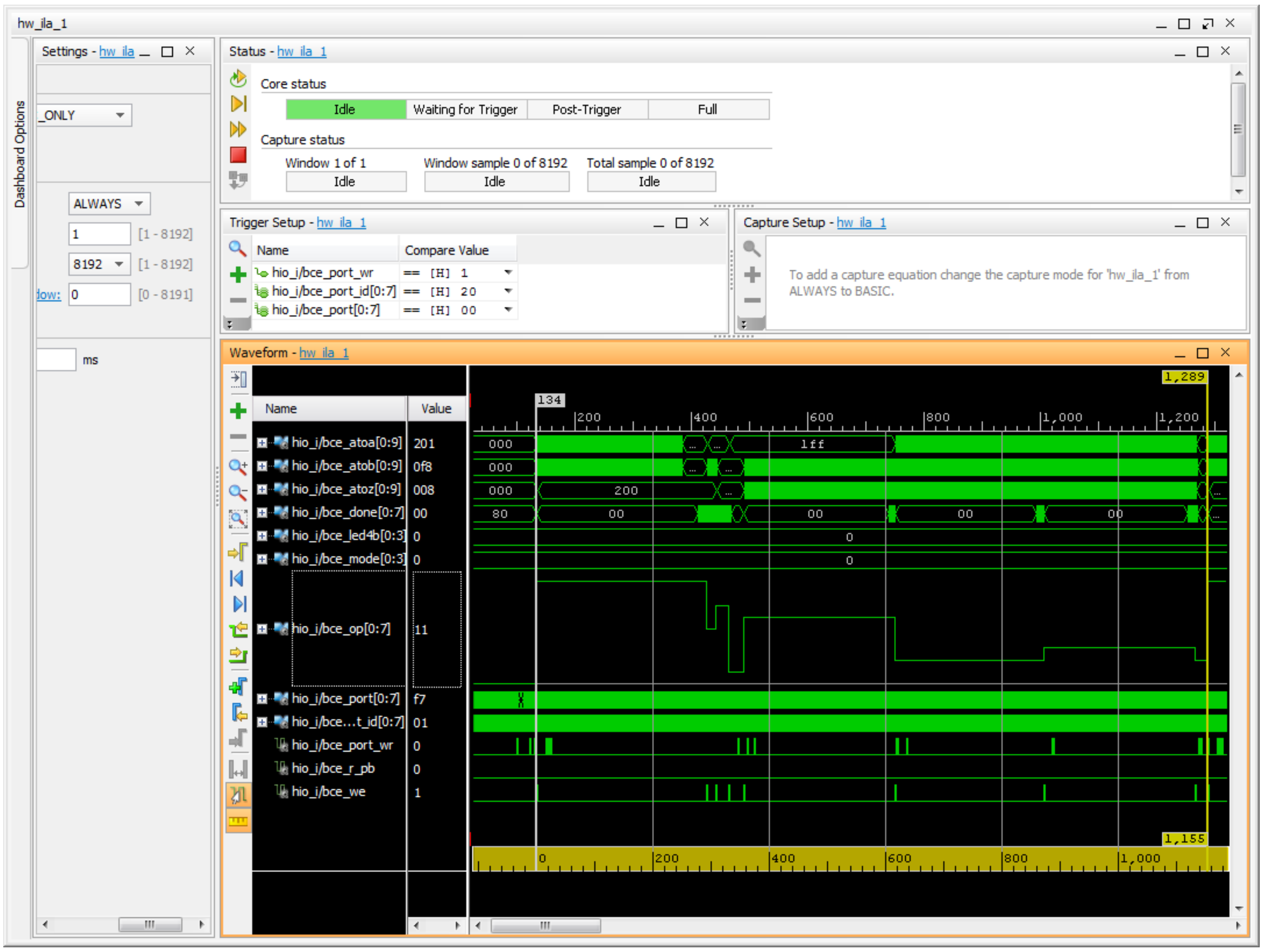
```
…
pb2dfu_set(0x20, 0);   // To provide dedicated trigger (0x00 on port 0x20) for the  ILA
for (i = 0; i < 4; i++) {
        for (j = 2; j <= 3; j++) {
                lms(j, n, op);
                pb2mb_eoc(led);
        }
}
…
```

The output function **pb2dfu_set()** writes 0x00 to port 0x20 this time. In Vivado Lab Edition 2015.4, modify the trigger condition in the Trigger window to:

(bce_port_wr ==1) AND (bce_port_id[0:7]==0x20) AND (bce_port[0:7]==**0x00**)

In Vivado Lab Edition 2015.4, arm the hw_ila_1 core again by pressing the **Run Trigger** button in the **Hardware** window.

The armed hw_ila_1 core will wait until the running demo design comes to the point, where PicoBlaze6 calls this dedicated function call pb2dfu_set(0x20, 0); as defined in the LMS C code (See Figure 34). This C code is executed by the corresponding the PicoBlaze6 controller inside of the first EdkDSP accelerator. This will trigger capturing of new 8192 samples of all debug signals with the sampling rate 120 MHz and provide detailed trace of the initial 8192 samples of the LMS filter computation. See Figure 42.



Core status

Idle	Waiting for Trigger	Post-Trigger	Full

Capture status

Window 1 of 1	Window sample 0 of 8192	Total sample 0 of 8192
Idle	Idle	Idle

_ONLY	
ALWAYS	
1	[1 - 8192]
8192	[1 - 8192]
0	[0 - 8191]

Name	Compare Value
hio_i/bce_port_wr	== [H] 1
hio_i/bce_port_id[0:7]	== [H] 20
hio_i/bce_port[0:7]	== [H] 00

To add a capture equation change the capture mode for 'hw_ila_1' from ALWAYS to BASIC.

Name	Value
hio_i/bce_atoa[0:9]	201
hio_i/bce_atob[0:9]	0f8
hio_i/bce_atoz[0:9]	008
hio_i/bce_done[0:7]	00
hio_i/bce_led4b[0:3]	0
hio_i/bce_mode[0:3]	0
hio_i/bce_op[0:7]	11
hio_i/bce_port[0:7]	f7
hio_i/bce…t_id[0:7]	01
hio_i/bce_port_wr	0
hio_i/bce_r_pb	0
hio_i/bce_we	1

Figure 42: LMS filter waveforms after the trigger in Vivado Lab Edition 2015.4.

The red trigger corresponds to the event. We can zoom in the data and define additional markers. Selected markers to indicate single elementary step of the LMS filter. It takes 1154 clock cycles (120 MHz = 8.333 ns clock period) to compute the vector product of two floating point vectors (coefficients and data), both with length 250*8=2000 elements, update the data vector (in a circular buffer), compute the prediction error and adapt the coefficients of the floating point LMS filter.

The bce_op[0:7] debug signal is displayed in the analogue/hold mode. This helps to indicate the sequence of vector operations issued by the PicoBlaze6 firmware. Each step of the LMS algorithm is implemented as a sequence of seven vector operations of (8xSIMD) EdkDSP deined as in the PicoBlaze6 function lms(). See Figure 34, and the definition of operations summarised in Table 3.

The ARM code and the MicroBlaze code can be compiled with –O0, … , -O3 optimisations and executed under both debuggers in combination with the ILA HW debug and visualisation.

The –O0 option provides lower performance on ARM and MicroBlaze processors, but the corresponding binary code includes no transformations. This makes the co-debugging of the ARM and MicroBlaze C code easier.

The MicroBlaze debugger helps also in debugging of the interactions of the MicroBlaze with the (8xSIMD) EdkDSP accelerators. Blocks of floating point data can be inspected and verified with support of the MicroBlaze debugger before and after the synchronisation points of the MicroBlaze API interface.

The (8xSIMD) EdkDSP accelerator code and the floating point data path are deterministic. All operations can be also emulated in the MicroBlaze C code, including the exact sequence of all floating point operations.

The floating point HW unit of the MicroBlaze supports the single precision floating point ADD and MULT operations with bit-exact identical results to the floating point units used in the (8xSIMD) EdkDSP accelerators.

This determinism secures, that the MicroBlaze “golden C code” can deliver floating point results which are bit-exact identical to the (8xSIMD) EdkDSP accelerators. This is used for verification of algorithms executed by the (8xSIMD) EdkDSP accelerator.

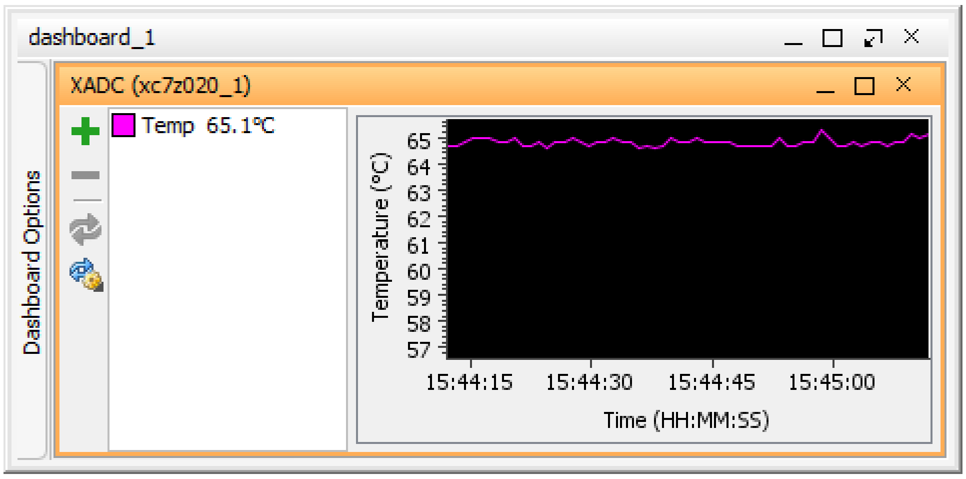


Figure 43: Separate dashboard with display of temperature in Vivado Lab Edition 2015.4.

The Vivado Lab Edition 2015.4 jtag based interface supports also the continuous download of some additional signals measured in the ZYNQ fabric. Data can be opened in a separate dashboard.

The dashboard is presented in Figure 43. It displays the temperature inside of the ZYNQ fabric. It is oscillating around 62 degrees Celsius. The default sampling rate is 0,5 sec.

These measurements run in the background and do not influence the HW and SW running on the monitored device.

See the running video processing system performing the HW accelerated full HD motion detection together with floating point computation of FIR and LMS filter in Figure 44.

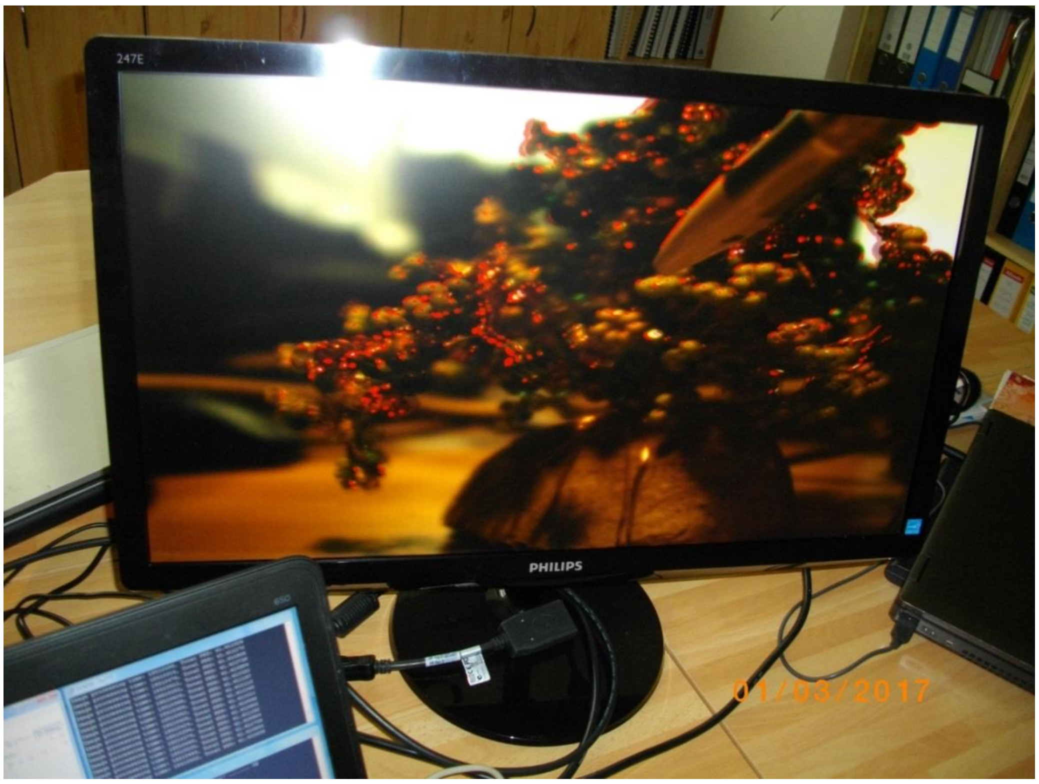


Figure 44: HW accelerated motion detection algorithm – moving edges marked by red.

3. Conclusions

This application note and related evaluation package document following general observations and conclusions:

- Programmable logic part of the Zynq xc7z020-2I device implements in parallel:
    - Run-time reprogrammable (8xSIMD) EdkDSP floating point accelerator IP
    - Several HW accelerated video processing algorithms for the Full HD Toshiba color sensor with the Full HD resolution 1920x1080p60.

- The total power consumption for the HW accelerated video processing (measured at the 12V DC power supply) is up to 7.05 W. This requires passive heat sink [2] for the Zynq module.

- The energy per pixel is significantly reduced for the HW accelerated designs with HW accelerators. Energy per pixel reduction from 3x to 29.8x can be reached in comparison to ARM SW solution. Best energy/pixel reduction is reached for the HW motion detection.

- The energy per pixel measurements are valid for the system running in parallel single (8xSIMD) EdkDSP floating point accelerator computing the FIR filter.

- Main source of the energy per pixel saving in comparison to the SW solution is the increased frame rate of the video processing achieved by the HW accelerators.

- The combination of 32bit MicroBlaze with single (8xSIMD) EdkDSP floating point accelerator IP brings additional capability to compute (in single precision floating point) with performance in the range of around one GFLOP/s (1.13 GFLOP/s in case of FIR filter) at the expense of only moderate increase of the total power consumption of the system:

    - 6.75 W with MicroBlaze and no (8xSIMD) EdkDSP IP - 10 MFLOP/s
    - 7.05 W with MicroBlaze + (8xSIMD) EdkDSP - 1139 MFLOP/s

This application note documents how designs debugged and developed in the high level SDSoC 2015.4 environment can be exported to the end-user in form of SDK 2015.4 SW projects with precompiled HW designs.

Enclosed SDK 2015.4 projects provide space for the end-user to make adaptations and customisations of the final SW application in the C source code for ARM and MicroBlaze. The compilation from C/ASM source code and programming of the PicoBlaze6 controller located in the 8xSIMD EdkDSP IP core is also supported and demonstrated.

This user customisation is possible without detailed information about the used IP cores, Vivado 2015.4 projects and without the SDSoC 2015.4 board support package for the Full HD Toshiba and the 8xSIMD EdkDSP IP core.

Application note explained integration of the Full HD Toshiba video sensor with the HW accelerated video processing algorithms and single run-time reprogrammable 8xSIMD EdkDSP floating point accelerator IP core maintained by UTIA in the ECSEL THINGS2DO project.

4. References

[1] TE0720-03-2IF; Part: XC7Z020-2CLG484I; 1 GByte DDR; Grade: Industrial;
http://shop.trenz-electronic.de/en/TE0720-03-2IF-Xilinx-Zynq-module-XC7Z020-2CLG484I-ind.-temp.-range-1-Gbyte
https://www.trenz-electronic.de/fileadmin/docs/Trenz_Electronic/TE0720/REV02/Documents/TE0720%20User%20Manual-v45-20150323_1407.pdf
https://www.trenz-electronic.de/fileadmin/docs/Trenz_Electronic/TE0720/REV03/Documents/TRM-TE0720-03.pdf

[2] Heatsink for TE0720, spring-loaded embedded;
https://shop.trenz-electronic.de/en/26922-Heatsink-for-TE0720-spring-loaded-embedded?c=38

[3] TE0701-06 Carrier Board for Trenz Electronic 7 Series;
https://shop.trenz-electronic.de/en/TE0701-06-Carrier-Board-for-Trenz-Electronic-7-Series?c=261
https://www.trenz-electronic.de/fileadmin/docs/Trenz_Electronic/carrier_boards/TE0701/REV04/Documents/TRM-TE0701-04.pdf
https://www.trenz-electronic.de/fileadmin/docs/Trenz_Electronic/carrier_boards/TE0701/REV05/Documents/TRM-TE0701-05.pdf
https://www.trenz-electronic.de/fileadmin/docs/Trenz_Electronic/carrier_boards/TE0701/REV06/Documents/TRM-TE0701-06.PDF

[4] AES-FMC-HDMI-CAM-G HDMI Input/Output FMC Module
https://products.avnet.com/shop/en/ema/kits-and-tools/development-kits/aes-fmc-hdmi-cam-g-3074457345628965509/?categoryId=&fromPage=autoSuggest

[5] Toshiba Industrial 1080P60 Camera Module
https://www.avnet.com/opasdata/d120001/medias/docus/7/AES-CAM-TOSH-1080P-G-product-brief.pdf

[6] PmodRS232: Serial converter & interface.
https://shop.trenz-electronic.de/de/23331-PmodRS232-Serial-converter-und-interface?c=215

[7] VMware Workstation Player Documentation
https://www.vmware.com/support/pubs/player_pubs.html

[8] Vivado HLx Web Install Client - 2015.4.
https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2015-4.html

[9] SDSoC - 2015.4 Full Product Installations.
https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/sdx-development-environments/sdsoc/2015-4.html

[10] THINGS2DO Algorithms, Design Methods, and Many-core Execution Platform for Low- Power Massive Data-Rate Video and Image Processing Artemis 2013 GA 621439
http://www.THINGS2DO.eu/

[11] THINGS2DO Project www page in UTIA with pointers to evaluation packages for download
http://sp.utia.cz/index.php?ids=projects/THINGS2DO

5. Free Evaluation Version of the Package

The **evaluation version of the package** can be downloaded from UTIA www pages [11] free of charge.

Deliverables:
The evaluation package [11] includes evaluation bitstreams with single (8xSIMD) EdkDSP accelerator IP core working in parallel with the HW-accelerated edge detection and motion detection algorithms for video processing on the Trenz Electronic TE0720-2IF module [1] located on the Trenz Electronic TE0701 carrier [3] with the FMC card [4], FULL HD TOSHIBA Color Image Sensor Camera Module [5] and the PMODRS232 adapter [6].

The evaluation package [11] includes bitstreams compiled with the evaluation version of the UTIA (8xSIMD) EdkDSP HW accelerator IP core. Evaluation IPs compiled in the enclosed bitstreams:

bce_fp12_1x8_0_axiw_v1_10_c	Evaluation version of the AXI-lite interface
bce_fp12_1x8_40	Evaluation version of the floating point data path

This evaluation version of the UTIA (8xSIMS) EdkDSP accelerator is compiled into bitstream with an HW limit on number of vector operations.

The termination of the nonexclusive, non-transferable evaluation license of this evaluation IP core is reported in advance by the demonstrator on the RS232 terminal. The evaluation designs run again after the reset.

The evaluation package [11] includes SDK 2015.4 SW projects with source code for MicroBlaze processor and the 1GHz ARM processor in a single Zynq device. SW projects support single (8xSIMD) EdkDSP floating point accelerators for the Trenz Electronic TE0720-2IF module [1] on Trenz Electronic TE701 carrier board [3].

The evaluation package [11] includes SDK 2015.4 SW projects with C source code for the ARM Cortex A9 processor (32bit) in standalone mode, C source code for MicroBlaze and C/ASM source code for the PicoBlaze6 controller in the 8xSIMD EdkDSP IP core.

The evaluation package [11] includes these static libraries for ARM Cortex A9 processor (32bit) for standalone mode:

libfmc_imageon.a	SDK 2015.4 UTIA static library with interface functions for video IP cores
libsh01.a	SDSoC 2015.4 static library for HW accelerator in project sh01
libsh02.a	SDSoC 2015.4 static library for HW accelerator in project sh02
libsh03.a	SDSoC 2015.4 static library for HW accelerator in project sh03
libmd01.a	SDSoC 2015.4 static library for HW accelerator in project md01

and static library for the MicroBlaze processor (32bit) for standalone mode:

libwal.a	SDK 2015.4 UTIA static library with 8xSIMD EdkDSP API for the MicroBlaze processor

These libraries have no time restriction. Source code of these libraries is not provided in this evaluation package.

The evaluation package [11] includes these binary applications running on the evaluation board:

edkdsppp	EdkDSP C pre-processor binary for the TE0720-2IF module on TE0701 Carrier board.

edkdspcc	EdkDSP C compiler binary for the TE0720-2IF module on TE0701 Carrier board.
edkdspasm	EdkDSP ASM compiler binary for the TE0720-2IF module on TE0701 Carrier board

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

The evaluation package [11] includes demonstration firmware in C/ASM source code for the Xilinx PicoBlaze6 processor for the family of UTIA EdkDSP accelerator IP for the Trenz Electronic TE0720-2IF module [1] on Trenz Electronic TE0701 carrier board [3].

HW boards are not part of deliverables. HW can be ordered separately from [1] – [6].

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. See also the Disclaimer section.

6. Vivado Projects with the Evaluation Version of the (8xSIMD) EdkDSP IP for the Eniac THINGS2DO Project Partners

This evaluation package includes **Vivado 2015.4 projects** for the Trenz Electronic TE0720-2IF module [1] located on the Trenz Electronic TE0701 carrier [3] with the FMC card [4], FULL HD TOSHIBA Color Image Sensor Camera Module [5] and the PMODRS232 adapter [6] **with the evaluation version of the (8xSIMD) EdkDSP accelerator IP for the partners in the Eniac THINGS2DO project [10]** can be ordered from UTIA AV CR, v.v.i., by email request for quotation to kadlec@utia.cas.cz.

UTIA AV CR, v.v.i., will provide to the THINGS2DO project partner quotation by email. After confirmation of the quotation by the customer, UTIA AV CR, v.v.i., will send to the customer this invoice:

> **The Vivado 2015.4 projects for the Trenz Electronic TE0720-2IF module [1] located on the Trenz Electronic TE0701-06 carrier [3] with the FMC card [4], FULL HD TOSHIBA Color Image Sensor Camera Module [5] and the PMODRS232 adapter [6] with the evaluation version of the (8xSIMD) EdkDSP accelerator IP for the partners in the Eniac THINGS2DO project**
> **(Without VAT)** **0,00 Eur**

After receiving confirmation from the THINGS2DO project partner about the zero-invoice received, UTIA AV CR, v.v.i. will send within 5 working days by standard mail printed version of this application note together with DVD with the Deliverables described in this section.

Deliverables:
The evaluation package for THINGS2DO partners [10] includes the Vivado 2015.4 design projects which can be modified and recompiled by the THINGS2DO project partner. The evaluation version of the UTIA (8xSIMD) EdkDSP accelerator is provided as part of the Xilinx Vivado 2015.4 design projects. Evaluation IPs included:

bce_fp12_1x8_0_axiw_v1_10_c	Netlist of the evaluation version of the AXI-lite interface
bce_fp12_1x8_40	Netlist of the evaluation version of the floating point data path

This netlist evaluation version of the UTIA 8xSIMD EdkDSP accelerator IP has an HW limit on number of vector operations.

THINGS2DO project [10] partners have nonexclusive, non-transferable license from UTIA to integrate this evaluation netlist into their own Vivado 2015.4 designs and to compile them to unlimited number of bit-streams for the Xilinx ZYNQ XC7Z020-2IF devices. This nonexclusive, non-transferable license has no time restriction.

The source code of the evaluation versions of the (8xSIMS) EdkDSP accelerator is the IP core owned by UTIA and the source code of it is not provided in the evaluation package to the THINGS2DO partners. The (8xSIMD) EdkDSP HW accelerator IP core is compiled with an HW limit on the number of vector operations.

The termination of the nonexclusive, non-transferable evaluation license is reported in advance by the demonstrator on the RS232 terminal. The evaluation designs run again after the reset.

The evaluation package for THINGS2DO partners includes SDK 2015.4 SW projects with C source code for the ARM Cortex A9 processor (32bit) in standalone mode, C source code for MicroBlaze and C source code for the EdkDSP PicoBlaze6 controller.

The evaluation package [11] includes these static libraries for ARM Cortex A9 processor (32bit) for standalone mode:

libfmc_imageon.a	SDK 2015.4 UTIA static library with interface functions for video IP cores
libsh01.a	SDSoC 2015.4 static library for HW accelerator in project sh01
libsh02.a	SDSoC 2015.4 static library for HW accelerator in project sh02
libsh03.a	SDSoC 2015.4 static library for HW accelerator in project sh03
libmd01.a	SDSoC 2015.4 static library for HW accelerator in project md01

and static library for the MicroBlaze processor (32bit) for standalone mode:

libwal.a	SDK 2015.4 UTIA static library with 8xSIMD EdkDSP API for the MicroBlaze processor

These libraries have no time restriction. Source code of these libraries is not provided in this evaluation package.

The evaluation package for THINGS2DO partners includes SDK 2015.4 SW projects with source code for MicroBlaze processor and ARM processor applications. SW projects support the UTIA 8xSIMD EdkDSP accelerator IP for the Trenz Electronic TE0720-2IF module [1] on Trenz Electronic TE701 carrier board [3].

The evaluation package [11] includes these binary applications running on the evaluation board:

edkdsppp	EdkDSP C pre-processor binary for the TE0720-2IF module on TE0701 Carrier board.
edkdspcc	EdkDSP C compiler binary for the TE0720-2IF module on TE0701 Carrier board.
edkdspasm	EdkDSP ASM compiler binary for the TE0720-2IF module on TE0701 Carrier board

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

The evaluation package for THINGS2DO partners includes demonstration firmware in C source code for the Xilinx PicoBlaze6 processor for the family of UTIA 8xSIMD EdkDSP accelerator IP for the Trenz Electronic TE0720-2IF module [1] on Trenz Electronic TE0701 carrier board [3].

HW boards are not part of deliverables. HW can be ordered separately from references [1] – [6].

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. See also the Disclaimer section.

7. Vivado Projects with the Release Version of the 8xSIMD EdkDSP IP with no HW Limit on Number of Vector Operations

This release package includes **Vivado 2015.4 projects** for the Trenz Electronic TE0720-2IF module [1] located on the Trenz Electronic TE0701 carrier [3] with the FMC card [4], the FULL HD TOSHIBA Color Image Sensor Camera Module [5] and the PMODRS232 adapter [6] **with the release version of the (8xSIMD) EdkDSP accelerator IP with no HW limit on number of vector operations** can be ordered by a customer from UTIA AV CR, v.v.i., by sending email request for quotation to kadlec@utia.cas.cz.

UTIA AV CR, v.v.i., will provide quotation by email. After confirmation of the quotation by the customer, UTIA AV CR, v.v.i., will send to the customer this invoice:

> **Vivado 2015.4 projects for the Trenz Electronic TE0720-2IF module [1] located on the Trenz Electronic TE0701 carrier [3] with the FMC card [4], the FULL HD TOSHIBA Color Image Sensor Camera Module [5] and the PMODRS232 adapter [6] with the release version of the 8xSIMD EdkDSP accelerator IP with no HW limit on number of vector operations.**
> **(Without VAT)** **400,00 Eur**

After receiving payment, UTIA AV CR, v.v.i. will send to the customer within 5 working days (by standard mail) the printed version of the application note together with a DVD with deliverables described in this section.

Deliverables:
The release package includes the Vivado 2015.4 design projects which can be modified and recompiled by the customer. Release IPs included:

bce_fp12_1x8_0_axiw_v1_10_c	Release netlist of the AXI-lite interface
bce_fp12_1x8_40	Release netlist of the floating point data path
bce_fp12_1x8_30	Release netlist of the floating point data path
bce_fp12_1x8_20	Release netlist of the floating point data path
bce_fp12_1x8_10	Release netlist of the floating point data path

This release netlist versions of the UTIA (8xSIMS) EdkDSP accelerators have **no HW limit on number of vector operations.**

The customer has a nonexclusive, non-transferable license from UTIA to integrate these netlists into own Vivado 2015.4 designs and to compile these netlists to an unlimited number of bit-streams for designs for the Xilinx ZYNQ XC7Z020-2I devices. This nonexclusive, non-transferable license has no time restriction.

The source code of the 8xSIMD EdkDSP accelerator IP is owned by UTIA and it is not provided in the release package to the customer.

The release package includes SDK 2015.4 SW projects with C source code for the ARM Cortex A9 processor (32bit) in standalone mode, C source code for MicroBlaze and C source code for the PicoBlaze6 controller inside of the 8xSIMD EdkDSP IP.

The release package includes these static libraries for ARM Cortex A9 processor (32bit) for standalone mode:

libfmc_imageon.a	SDK 2015.4 UTIA static library with interface functions for video IP cores
libwal.a	SDK 2015.4 UTIA static library with EdkDSP API for MicroBlaze
libsh01.a	SDSoC 2015.4 static library for HW accelerator in project sh01
libsh02.a	SDSoC 2015.4 static library for HW accelerator in project sh02
libsh03.a	SDSoC 2015.4 static library for HW accelerator in project sh03
libmd01.a	SDSoC 2015.4 static library for HW accelerator in project md01

These libraries have no time restriction. Source code of these libraries is not provided in the release package.

The release package includes SDK 2015.4 SW projects with source code for MicroBlaze processor and for the 1 GHz ARM Cortex A9 processor. SW projects support the family of UTIA (8xSIMD) EdkDSP accelerators for the Trenz Electronic TE0720-03-2IF module [1] on Trenz Electronic TE701-06 carrier board [3].

The release package includes these static libraries for ARM Cortex A9 processor (32bit) for standalone mode:

libfmc_imageon.a	SDK 2015.4 UTIA static library with interface functions for video IP cores
libsh01.a	SDSoC 2015.4 static library for HW accelerator in project sh01
libsh02.a	SDSoC 2015.4 static library for HW accelerator in project sh02
libsh03.a	SDSoC 2015.4 static library for HW accelerator in project sh03
libmd01.a	SDSoC 2015.4 static library for HW accelerator in project md01

and static library for the MicroBlaze processor (32bit) for standalone mode:

libwal.a	SDK 2015.4 UTIA static library with 8xSIMD EdkDSP API for the MicroBlaze processor

These libraries have no time restriction. Source code of these libraries is not provided in this evaluation package.

The evaluation package for THINGS2DO partners includes SDK 2015.4 SW projects with source code for MicroBlaze processor and ARM processor applications. SW projects support the UTIA 8xSIMD EdkDSP accelerator IP for the Trenz Electronic TE0720-2IF module [1] on Trenz Electronic TE701 carrier board [3].

The release package [11] includes these binary applications running on the evaluation board:

edkdsppp	EdkDSP C pre-processor binary for the TE0720-2IF module on TE0701 Carrier board.
edkdspcc	EdkDSP C compiler binary for the TE0720-2IF module on TE0701 Carrier board.
edkdspasm	EdkDSP ASM compiler binary for the TE0720-2IF module on TE0701 Carrier board

These binary applications have no time restriction. The user of the evaluation package has nonexclusive, non-transferable license from UTIA to use these utilities for compilation of the firmware for the Xilinx PicoBlaze6 processor inside of the UTIA 8xSIMD EdkDSP accelerator IP in precompiled designs. The source code of these compilers is owned by UTIA and it is not provided in the evaluation 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 Trenz Electronic TE0720-03-2IF module [1] on Trenz Electronic TE0701-06 carrier board [3].

HW boards are not part of deliverables. HW can be ordered separately from references [1] – [6].

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. See also the Disclaimer section.

Disclaimer