Application Note

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

SDK 2015.4 Projects for Evaluation of HW Accelerated Video Processing with Python 1300 Sensor and (8xSIMD) EdkDSP Accelerator on TE0720-03-2IF Module and TE0701-06 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	03.03.2017	Jiří Kadlec	Evaluation package for Xilinx SDK 2015.4

Acknowledgements:

This work has been partially supported by: ARTEMIS JU project ALMARVI No. 621439 [10] and by related MEYS (CZ NFA) project 7H14004 [11].

Table of Contents

Table of Figures

Table of Tables

1. Evaluation Designs

1.1 Objectives

This application describes use of an evaluation package with these demos:

- 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 be programmed for the number of processed micro-lines.
  - 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 (1024 micro-lines).

All demos work in parallel with single 8xSIMD EdkDSP run-time reprogrammable floating point accelerator.

- C programs can be compiled for the MicroBlaze and for the EdkDSP accelerator and used in the accelerator, without need to re-compile the design in Vivado 2015.4.
- C programs for the MicroBlaze processor and for the EdkDSP accelerator can be edited in the same SDK 2015.4 environment.

All demos are designed for the Trenz Electronic TE0701-06 platform [3] with industrial grade Zynq XC7Z020-2I device on System on Module TE0720-03-2I [1].

All demonstrated video processing algorithms have been developed, debugged and tested in Xilinx SDSoC 2015.4 environment [9].

SW algorithms have been compiled by Xilinx SDSoC 2015.4 system level compiler (based on the Xilinx HLS compiler) to Vivado 2015.4 HW projects, and compiled by Vivado 2015.4 [8] to the bitstreams for Zynq XC7Z020-2I device.

Created SW access functions controlling the HW accelerators have been exported from the Xilinx SDSoC 2015.4 projects to the Xilinx SDK 2015.4 [8] SW C projects as static .a libraries for standalone 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.

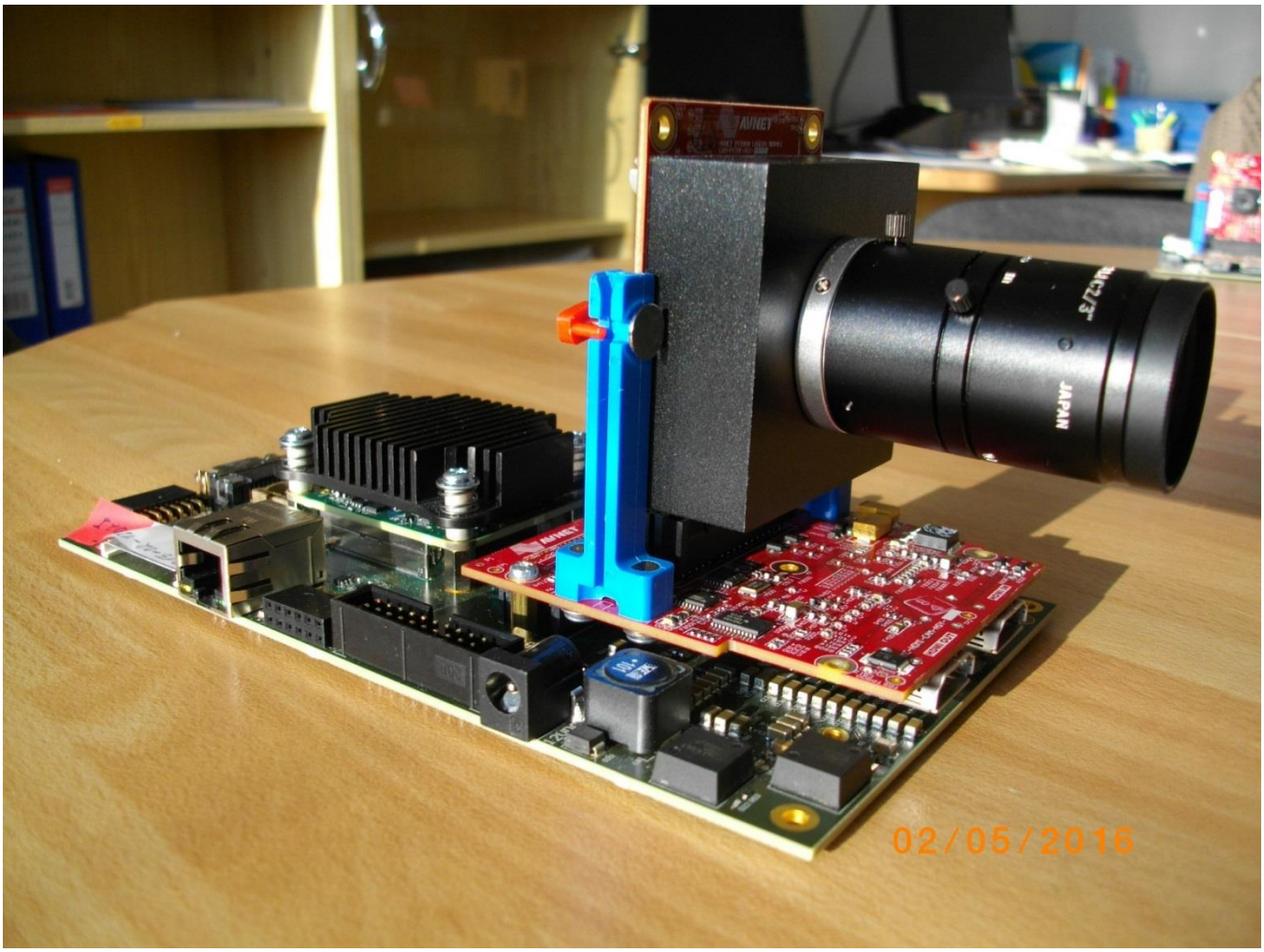


Figure 1: Python 1300 evaluation platform with video processing in HW and EdkDSP accelerator.

Common setup for demos:

- 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.
- Python 1300 sensor with resolution 1280x1024p60 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 HDMI display with fixed resolution 1280x1024p60.

1.2 Python 1300 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 Python 1300 colour video sensor with fixed resolution 1280x1024p60.

- 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 Python 1300 video sensor with resolution 1280x1024p60 to the input video frame buffers. The sensor pixel clock is set to 108 MHz.
- The input video DMA (VDMA) controller operates at 150 MHz.
- Part of PL are 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 108 MHz to the HDMI display with the resolution 1280x1024p60.

- UTIA EdkDSP is 8xSIMD floating point accelerator. 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 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 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 Python 1300 video sensor and the video output to a monitor with fixed 1280x1024p60 resolution.

Details of the video processing video chain:

- Data are provided by the Python 1300 video sensor with the 1280x1024p60 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 Python 1300 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 Python 1300 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:	Python 1300 video sensor (color 1280x1024p60) (Avnet)
FMC card output:	HDMI output to display (color 1280x1024p60)
Accelerator:	8xSIMD EdkDSP (120 MHz) with MicroBlaze (100 MHz)
Accelerator:	Edge detection single data path (150 MHz)
Platform (SDSoC 2015.4):	C:\S\t20i2pm4\hio\
Demos (SDK 2015.4):	C:\VM07\t20i2pm4\sh01_rows_resize_25_to_100\ C:\VM07\t20i2pm4\sh01_rows_fixed_100\

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

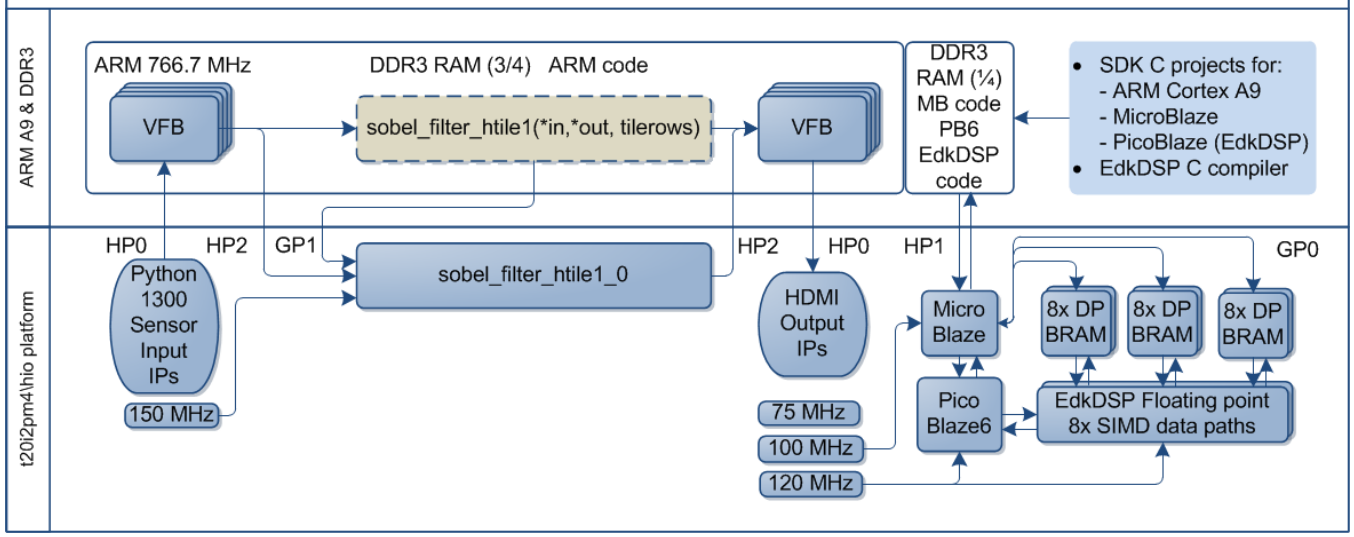


Figure 2: Project sh01 - Edge detection with single HW accelerator and EdkDSP accelerator.

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

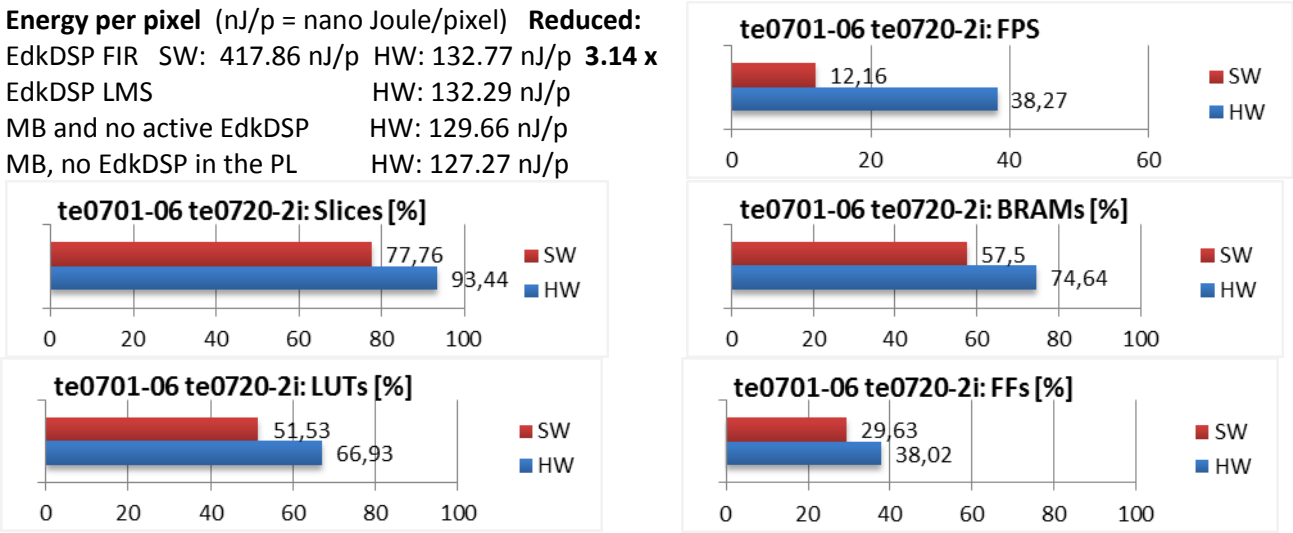


Figure 3: Project sh01 - Energy per frame reduction and used HW resources.

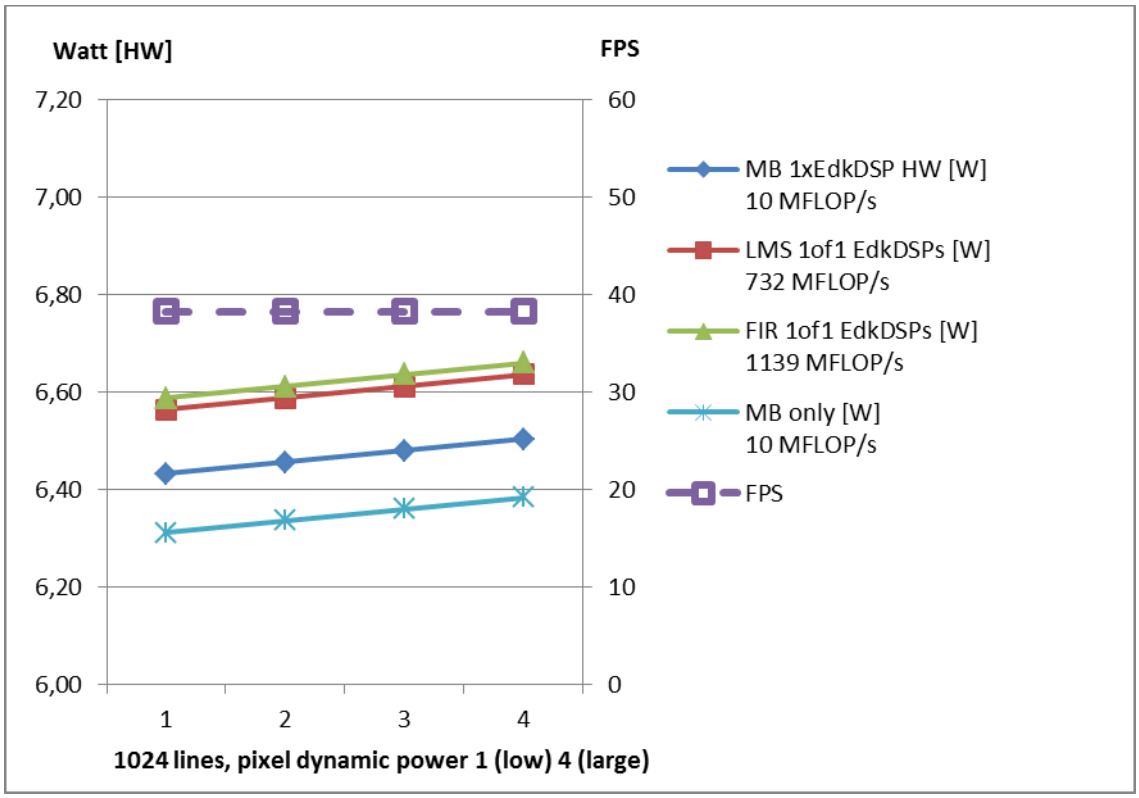


Figure 4: Project sh01 - Edge detection in 1x HW accelerator. 1x EdkDSP accelerator.

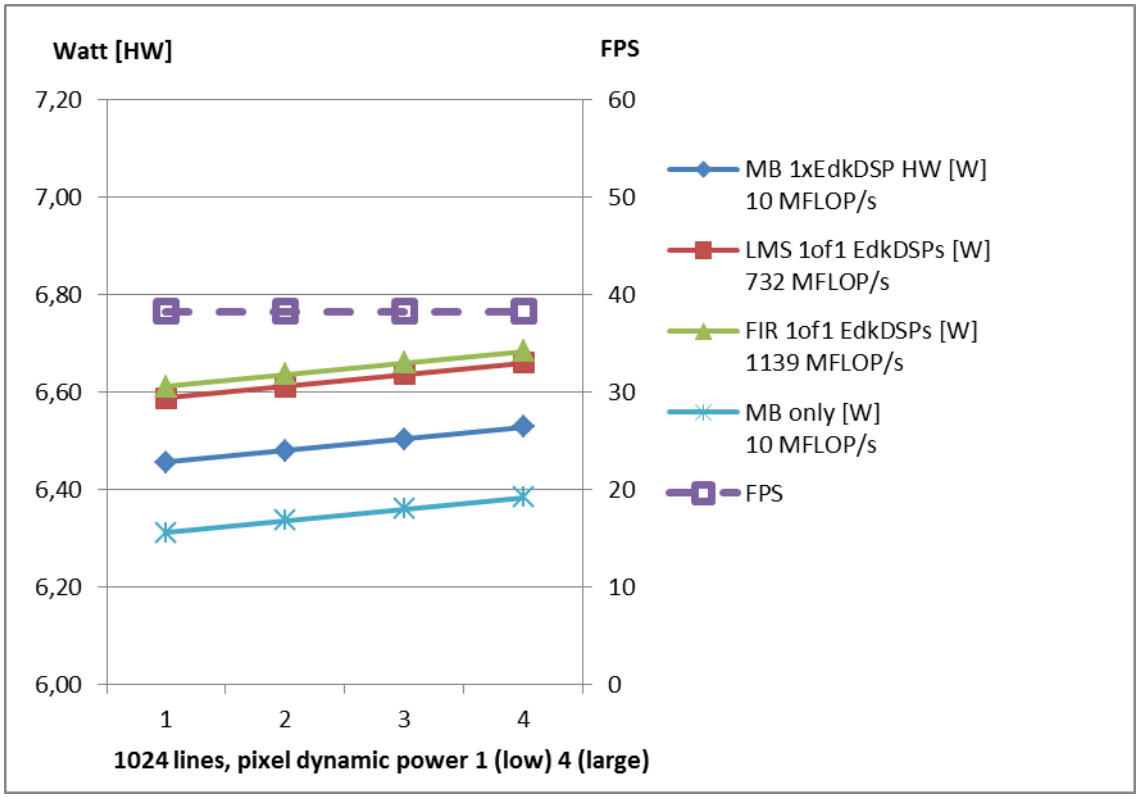


Figure 5: Project sh01 - Edge detection in 1x HW accelerator. 1x EdkDSP accelerator and ILA.

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:	Python 1300 video sensor (color 1280x1024p60) (Avnet)
FMC card output:	HDMI output to display (color 1280x1024p60)
Accelerator:	8xSIMD EdkDSP (120 MHz) with MicroBlaze (100 MHz)
Accelerator:	Edge detection two data paths (150 MHz)
Platform (SDK 2015.4):	C:\S\t20i2pm4\hio\
Demos (SDK 2015.4):	C:\VM07\t20i2pm4\sh02_rows_resize_25_to_100\ C:\VM07\t20i2pm4\sh02_rows_fixed_100\

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

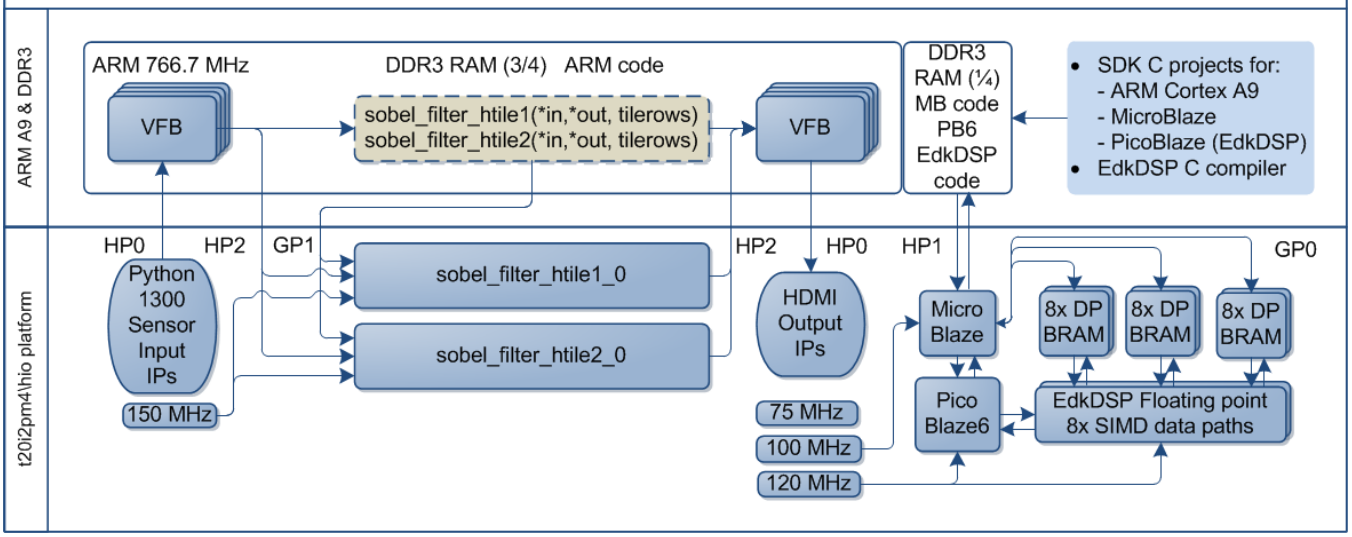


Figure 6: Project sh02 - Edge detection with two HW accelerators and EdkDSP accelerator.

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

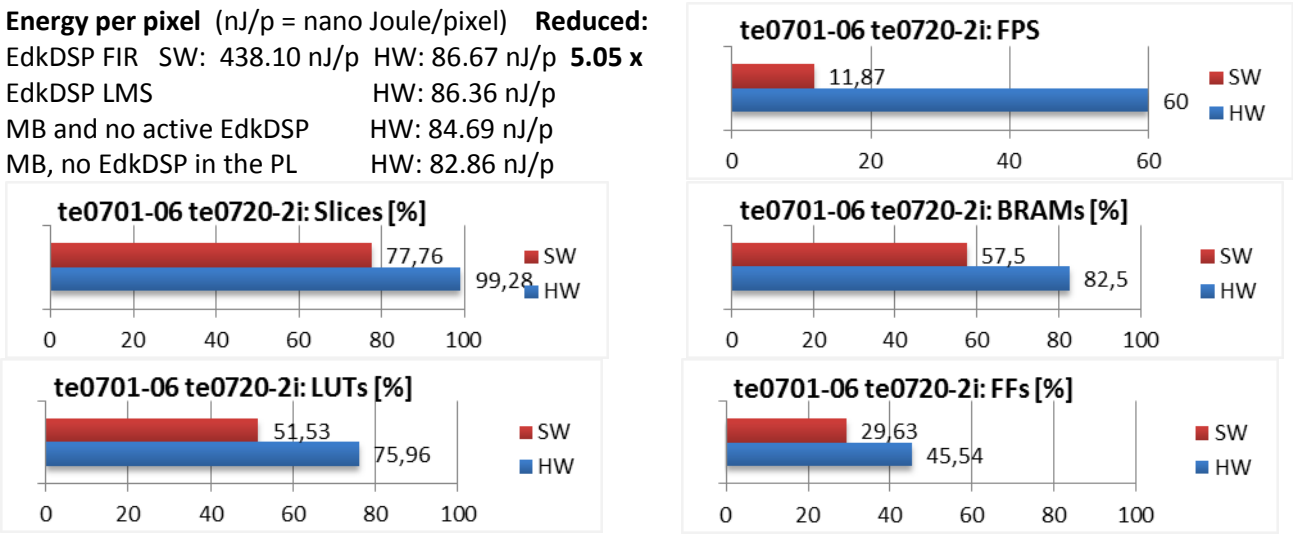


Figure 7: Project sh02 - Energy per frame reduction and used HW resources.

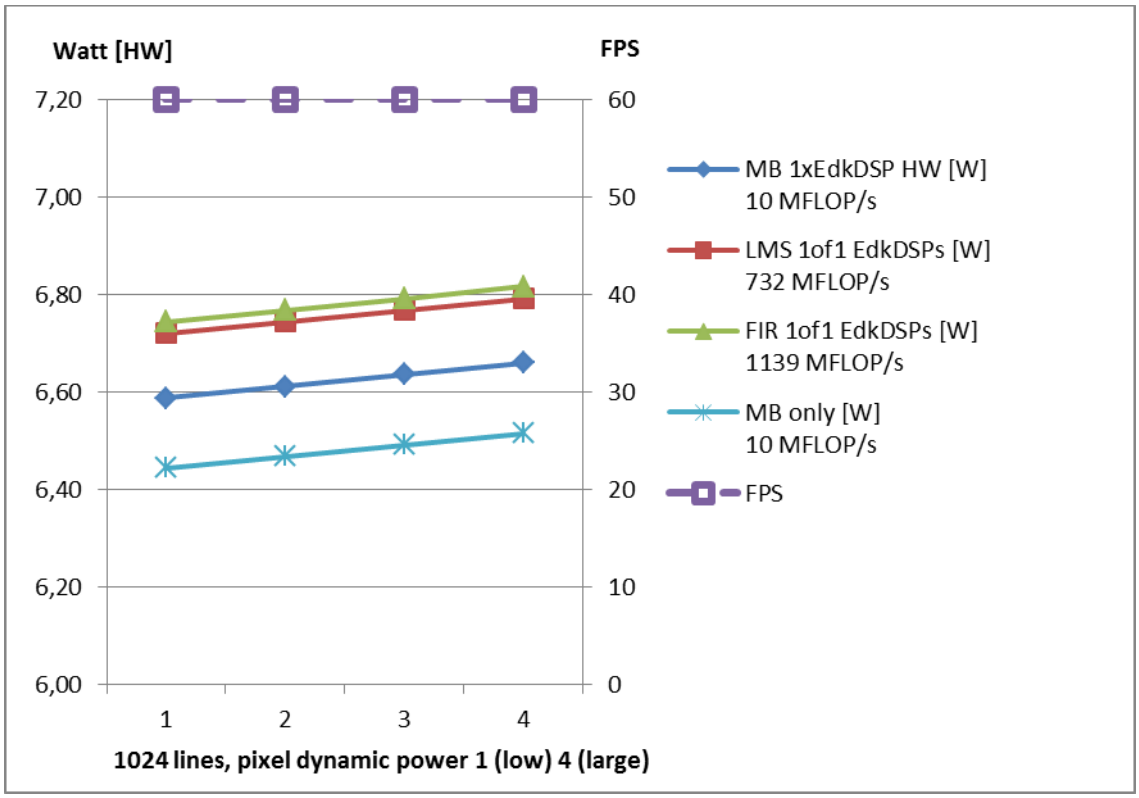


Figure 8: Project sh02 - Edge detection in 2x HW accelerators. 1x EdkDSP accelerator.

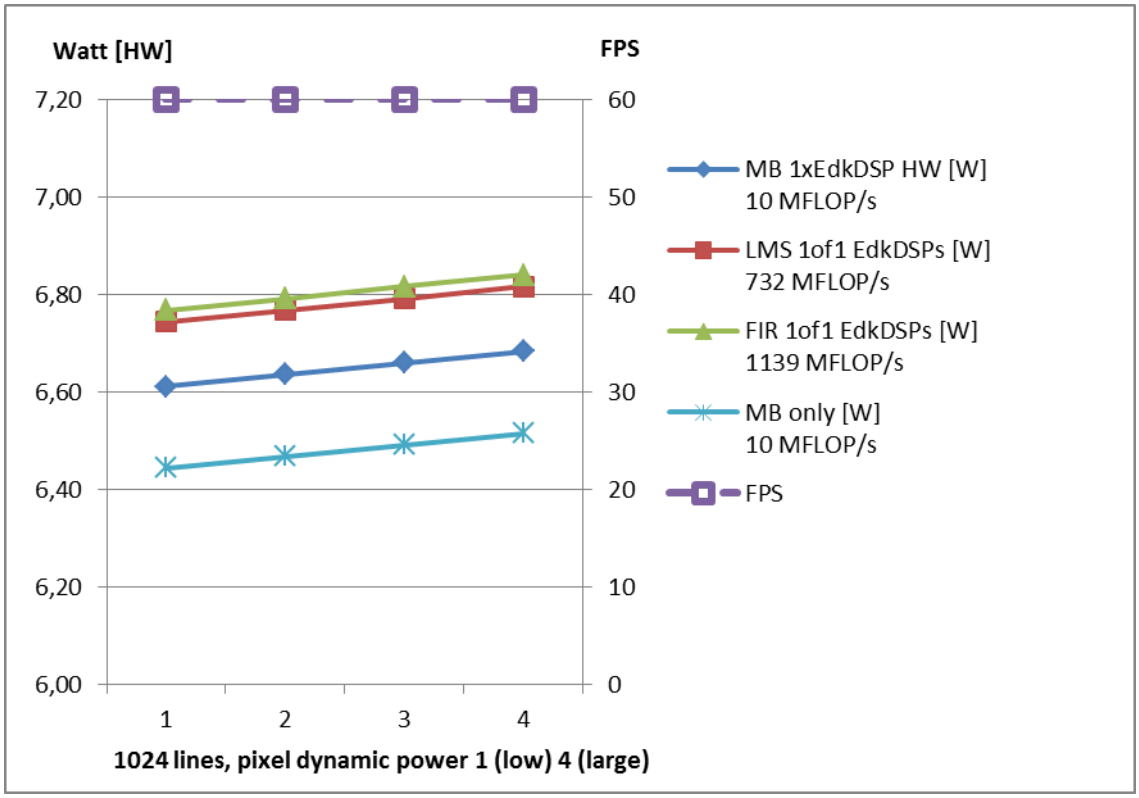


Figure 9: Project sh02 - Edge detection in 2x HW accelerators. 1x EdkDSP accelerator and ILA.

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:	Python 1300 video sensor (color 1280x1024p60) (Avnet)
FMC card output:	HDMI output to display (color 1280x1024p60)
Accelerator:	8xSIMD EdkDSP (120 MHz) with MicroBlaze (100 MHz)
Accelerator:	Edge detection three data paths (150 MHz)
Platform (SDSoC 2015.4):	C:\S\t20i2pm4\hio\
Demos (SDK 2015.4):	C:\VM07\t20i2pm4\sh03_rows_resize_25_to_100\
	C:\VM07\t20i2pm4\sh03_rows_fixed_100\

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

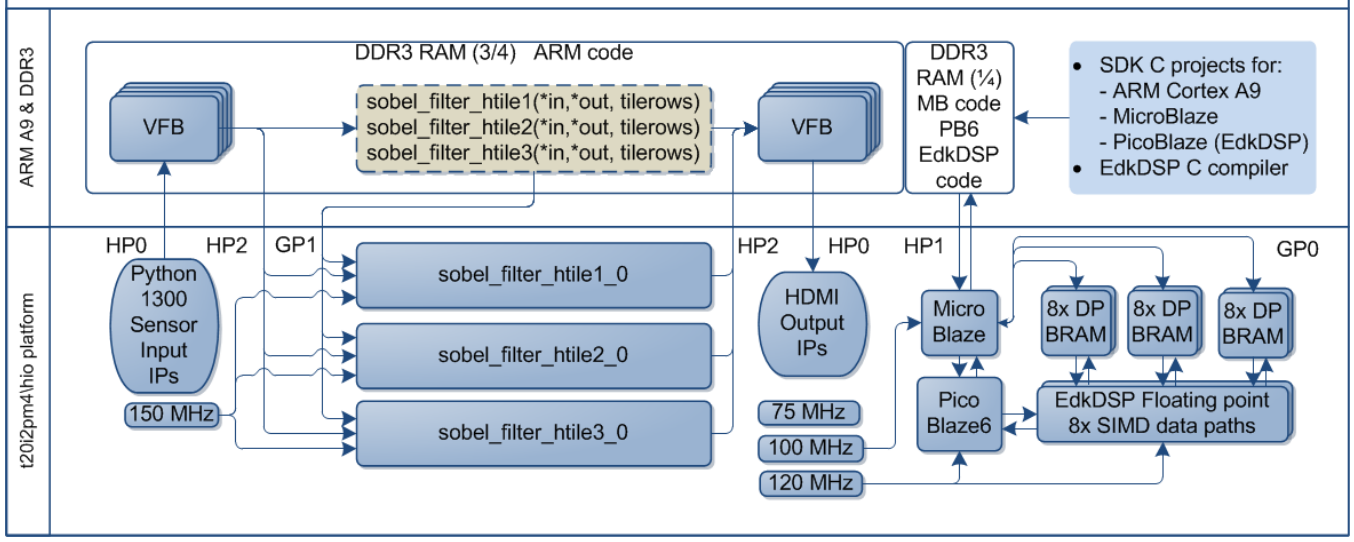


Figure 10: Project sh03 - Edge detection with three HW accelerators and EdkDSP accelerator.

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

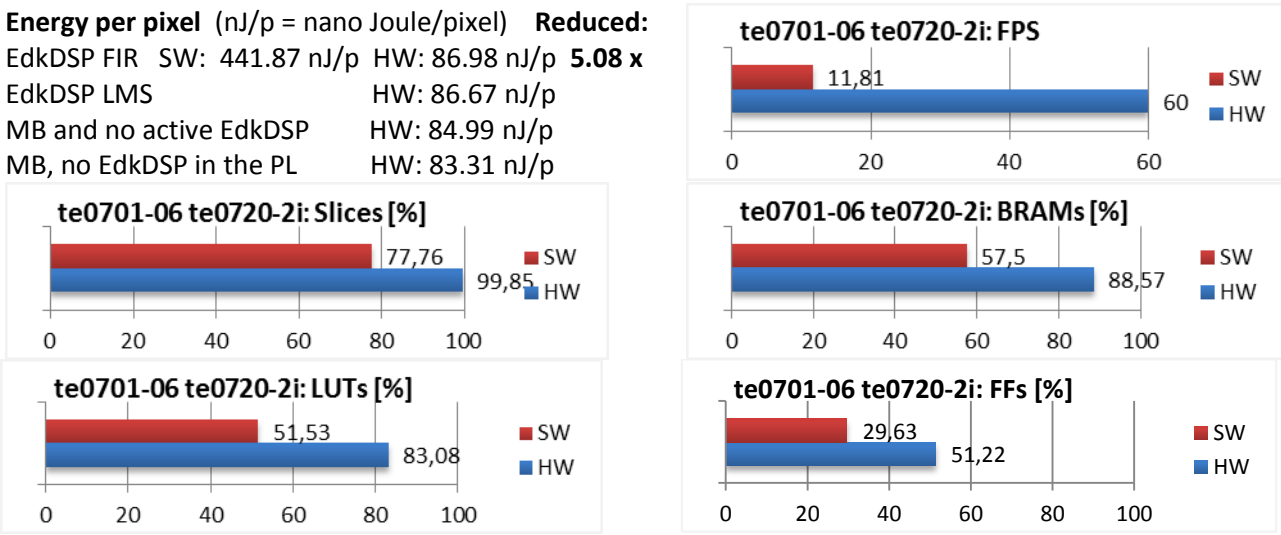


Figure 11: Project sh03 - Energy per frame reduction and used HW resources.

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

Zynq module:	TE0720-03-2IF (Trenz electronic)	1 = pad();
Carrier board:	TE0701-06 (Trenz electronic)	2 = sobel_filter_pass();
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)	3 = sobel_filter();
FMC card input:	Python 1300 video sensor (color 1280x1024p60) (Avnet)	4 = diff_image();
FMC card output:	HDMI output to display (color 1280x1024p60)	5 = median_char_filter_pass();
Accelerator:	8xSIMD EdkDSP (120MHz) with MicroBlaze (100MHz)	6 = combo_image();
Accelerator:	Motion detection single data path (150 MHz)	7 = ext();
Platform (SDS0C 2015.4):	C:\S\t20i2pm4\hio\	
Demos (SDK 2015.4):	C:\VM07\t20i2pm4\md01_rows_fixed_100\	

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

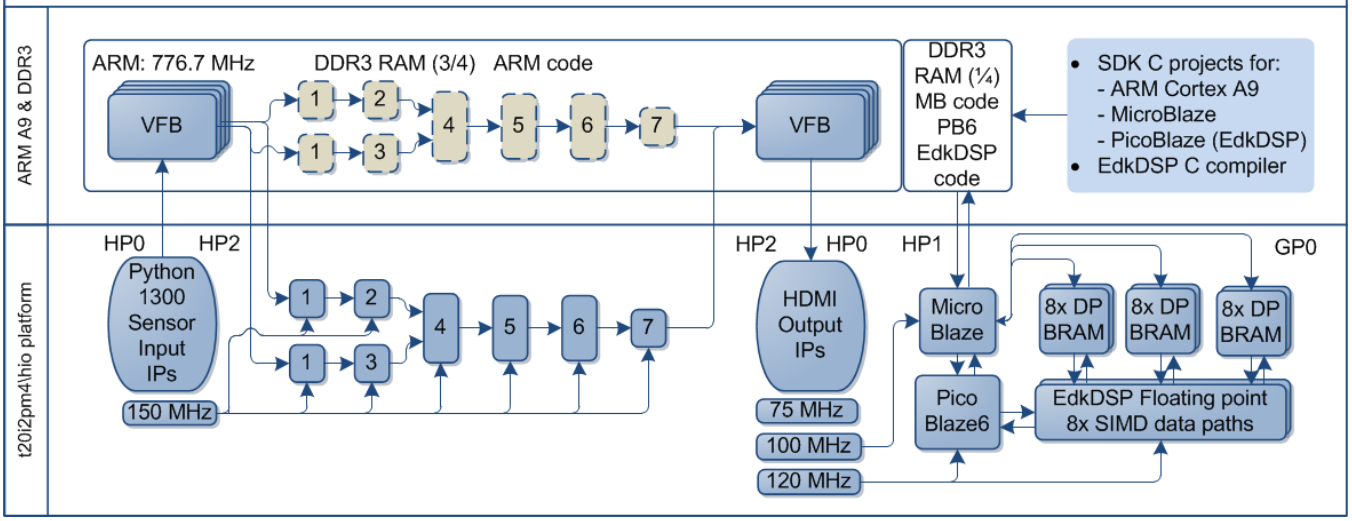
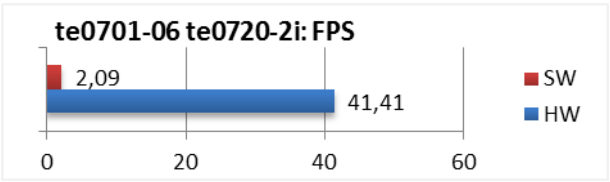
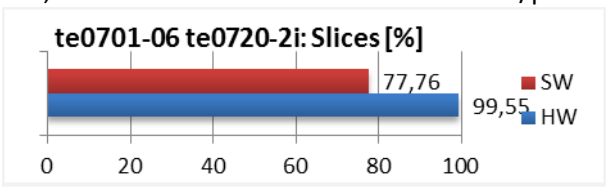


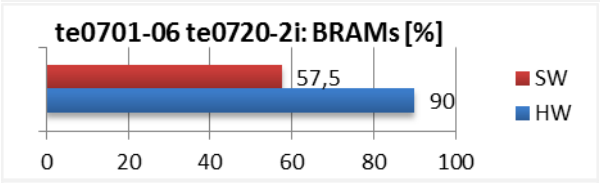
Figure 12: Project md01 - Motion detection with single HW accelerator and EdkDSP accelerator.

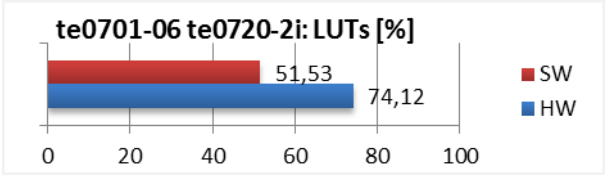
Energy per pixel (nJ/p = nano Joule/pixel) **Reduced:**

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	









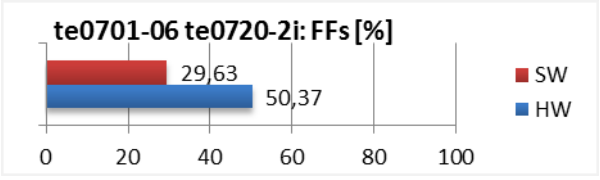


Figure 13: Project md01 - Energy per frame reduction and HW resources.

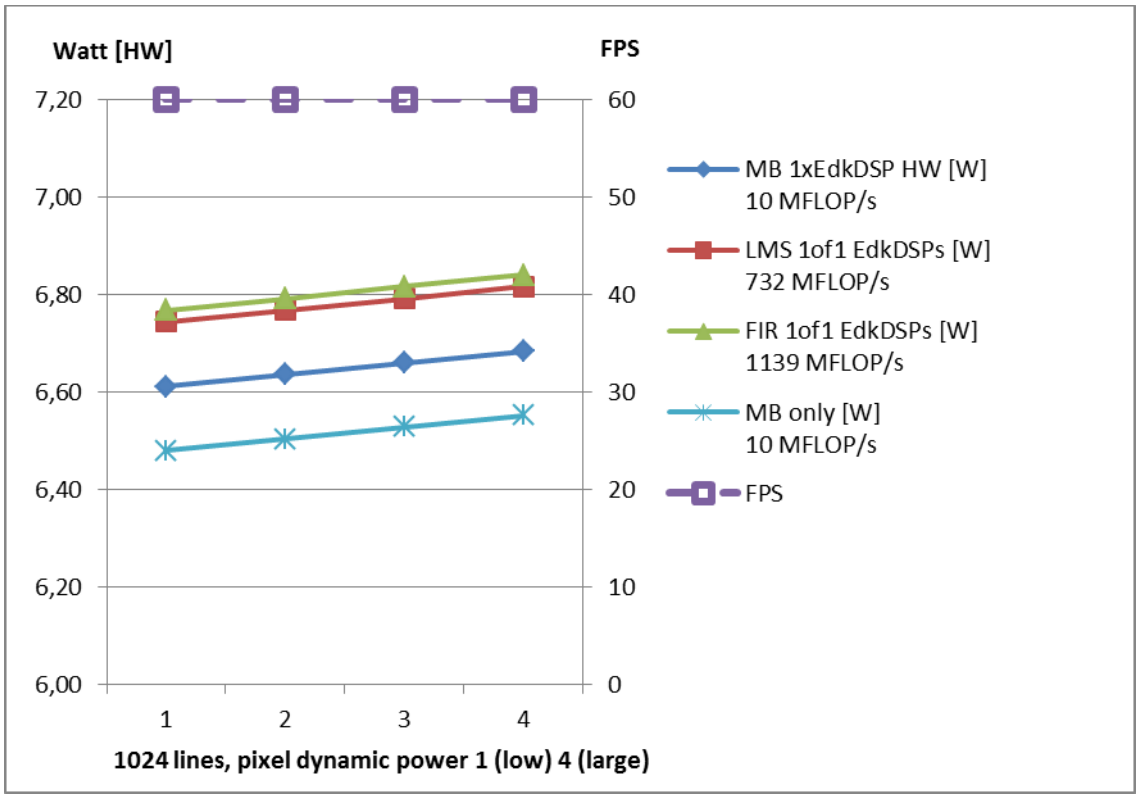


Figure 14: Project sh03 - Edge detection in 3x HW accelerator. 1x EdkDSP accelerator.

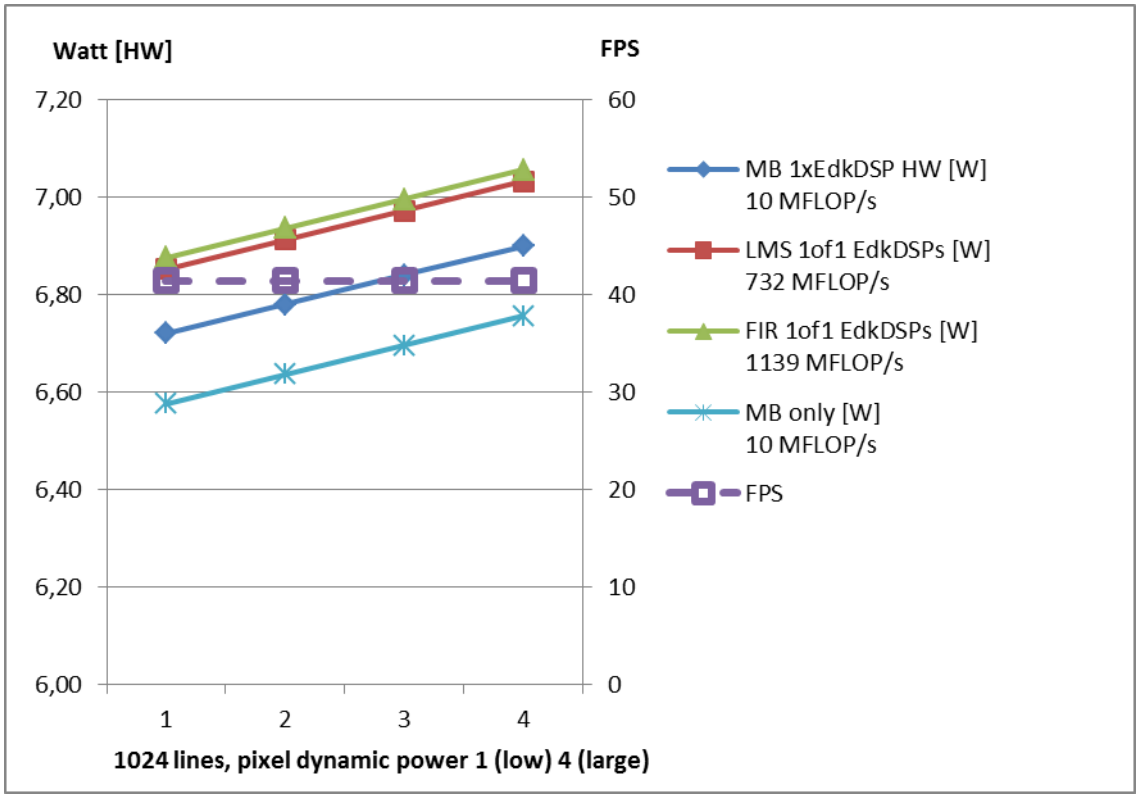


Figure 15: Project md01 - Motion detection in 8 chained HW accelerators. 1x EdkDSP accelerator.

2. Installation of the Evaluation Package

2.1 Import of SW projects in Xilinx SDK 2015.4

Unzip the evaluation package to directory of your choice.
The directory **C:\VM_07** will be used in this application note.
C:\VM_07\t20i2pm4_V54_IMPORT

Create empty directory for Xilinx SDK workspace.
C:\VM_07\t20i2pm4

Start Xilinx SDK 2015.4 and select the directory for the SDK 2015.4 workspace. See Figure 16. Select **C:\VM_07\t20i2pm4**

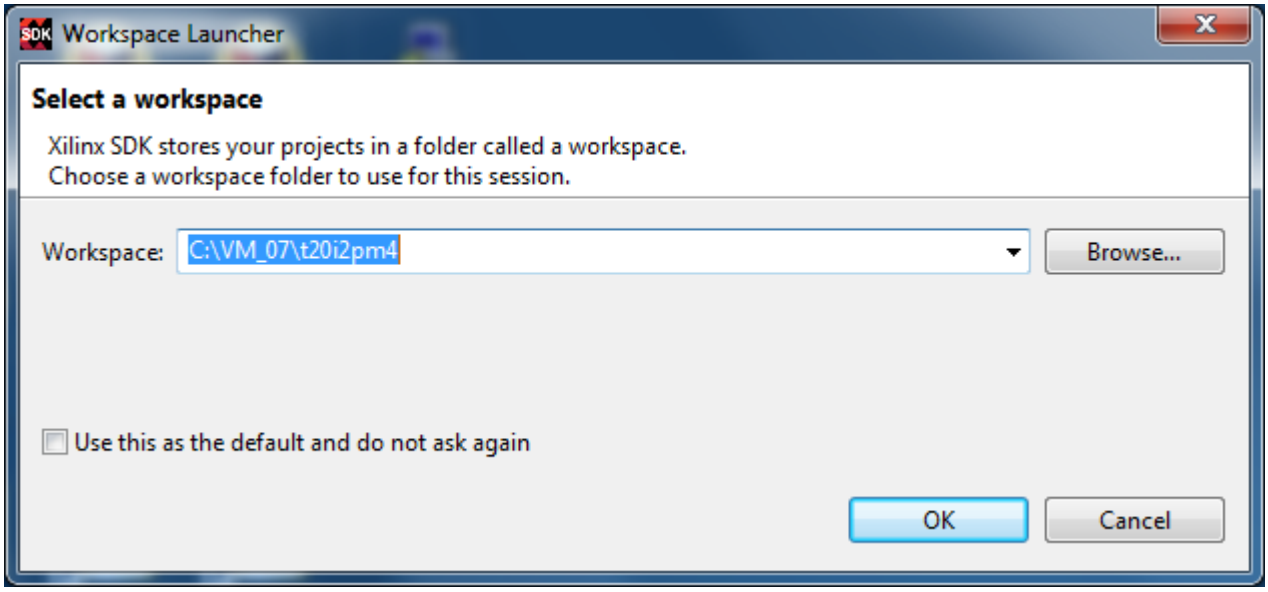


Figure 16: Select the SDK Workspace.

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

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

Click on Next button. See Figure 17.

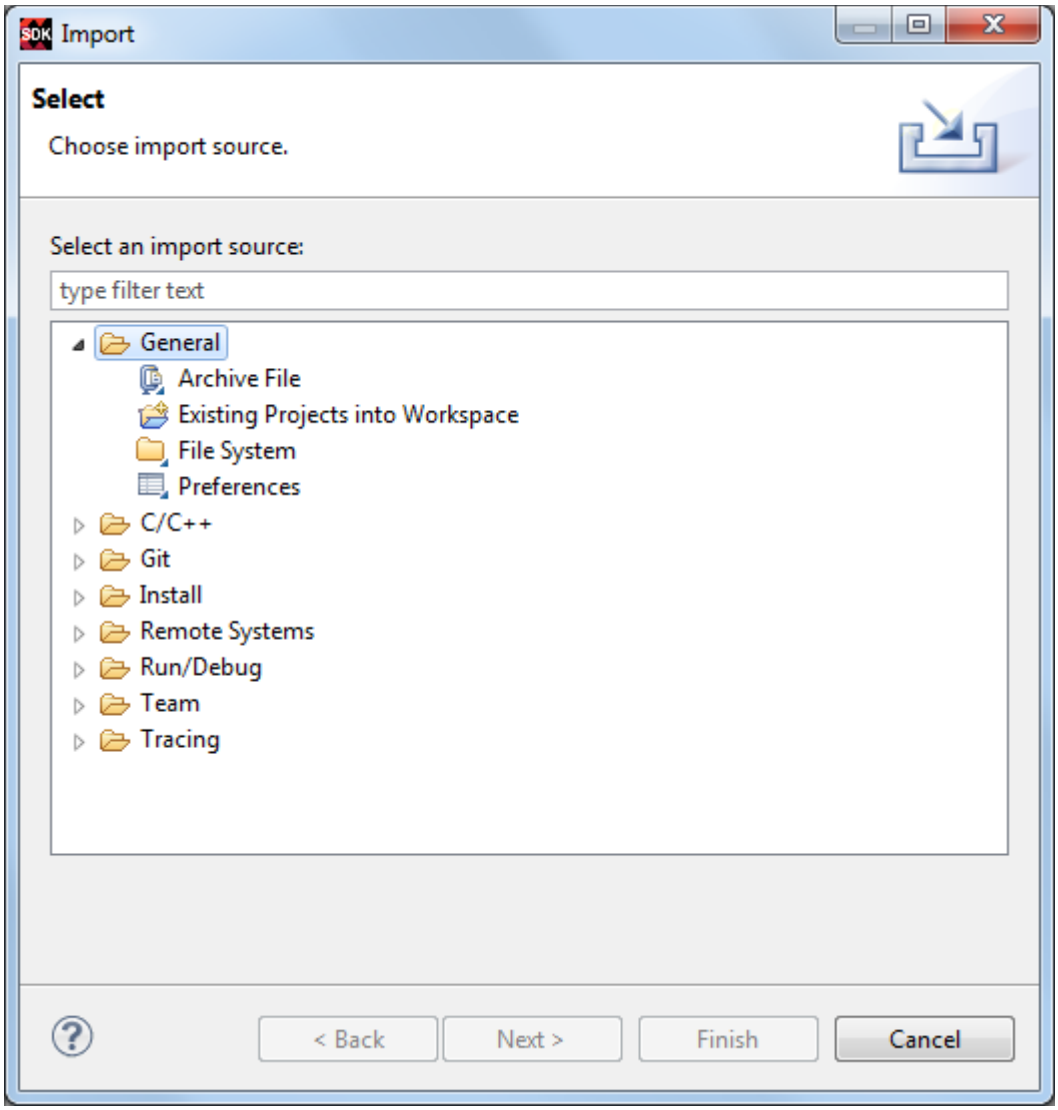


Figure 17: Import Existing Projects into Workspace.

Type the directory with projects to be imported. See Figure 18.

C:\VM_07\t20i2pm4_V54_IMPORT

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

Click on Finish button. See Figure 18.

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.

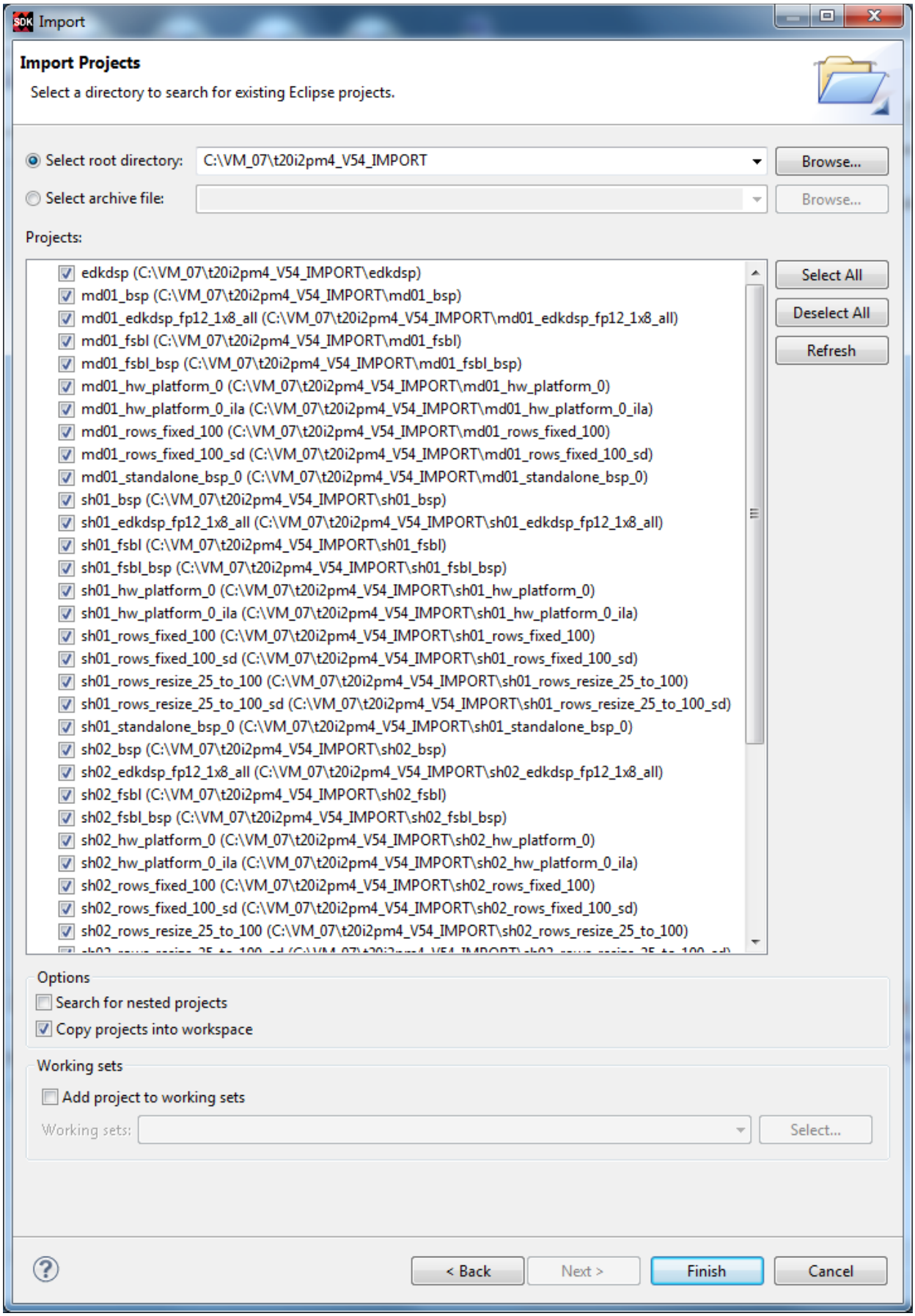


Figure 18: Select "Copy projects into workspace" and finish the import of all projects.

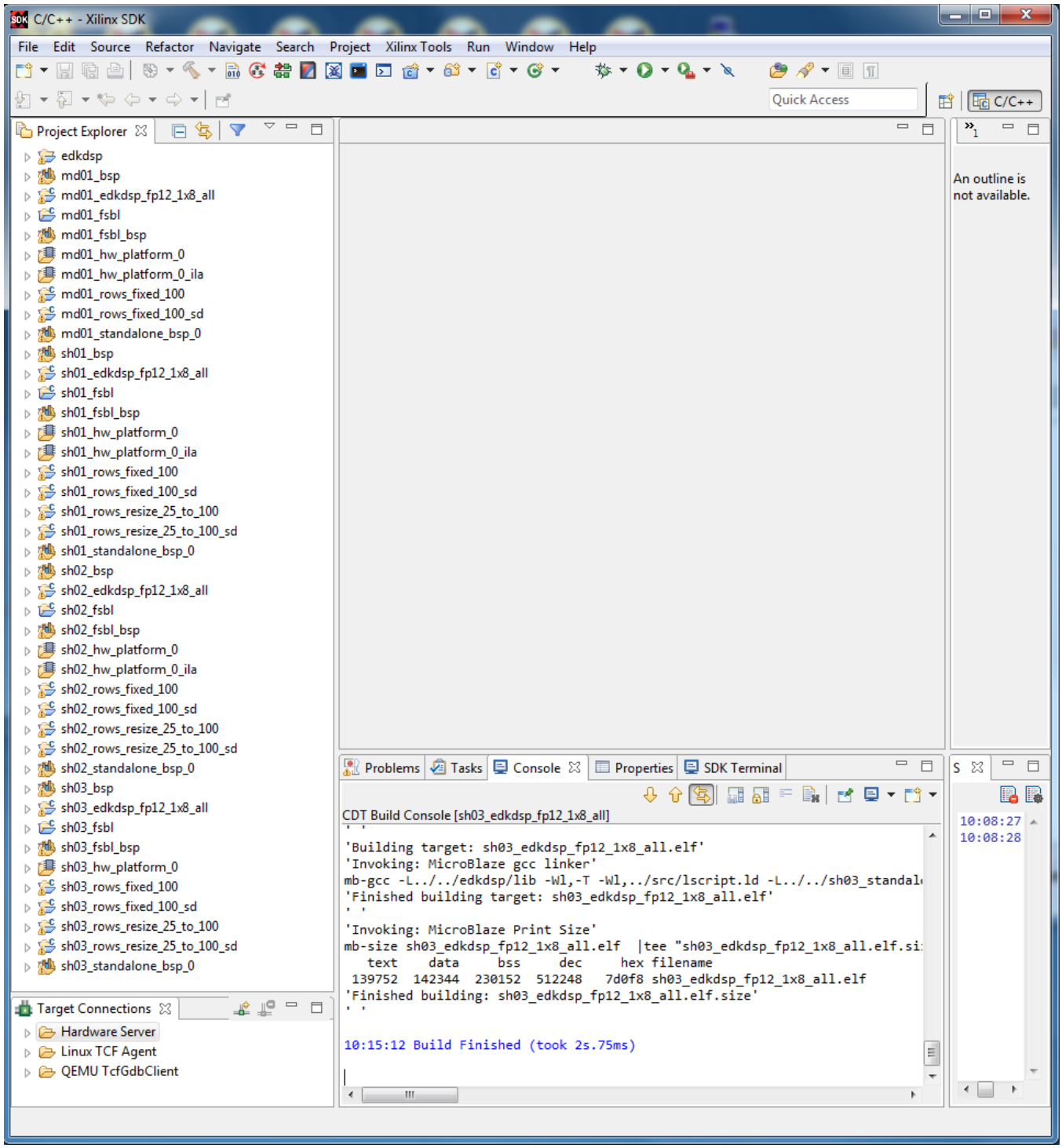


Figure 19: All projects are compiled in debug mode.

SDK 2015.4 compiles SW of all imported demos in debug mode.

2.2 HW setup

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

TE0720-03-2IF; Part: XC7Z020-2CLG484I; 1 GByte DDR; Industrial Grade (-40°C to +85°C) [1].
Heatsink for TE0720, spring-loaded embedded [2].
TE0701-06 Carrier Board for Trenz Electronic 7 Series [3].
AES-FMC-HDMI-CAM-G FMC card with HDMI I/O and CAM interface [4].
AES-CAM-ON-P1300C-G PYTHON-1300 Color Image Sensor Camera Module [5].
PmodRS232: Serial converter & interface [6].

See the latest technical reference manual (TRM) for the description of the TE0701-06 carrier board [3].

Set the TE0701-06 carrier board switches and jumpers for the TE0720-03-2IF module as follows:

- Set switch S3 as described in Figure 20
  TE701-06 switch S3: 1=OFF 2=OFF 3=ON 4=OFF

- Set jumpers to generate FMC_VADJ=2.5V as described in Figure 21
  TE701-06: TE701-06: (VIOTA=2.5V and VADJ=2.5V) J17: connect 2-3; J16: open; J21: connect 2-3

- Set switch S4 as described in Figure 22
  TE701-06 switch S4: (use FMC_VADJ=2.5V power source): 1=OFF 2=ON 3=ON 4=OFF

After this setup, the TE0720-03-2IF Zynq device works with IO-bank supply-voltages VIOTA=2.5V and VADJ=2.5V.

It is highly recommended to set switches of the TE0701-06 carrier board and measure the PL IO-bank supply-voltage before mounting of the module on TE0701-06. See locations and switch positions in Figure 20, Figure 21 and Figure 22 and read the TE0701-06 TRM [3].

In described configuration, the carrier board TE0701-06 connects these power supplies to the PMOD headers:

Header J1: VOUT = 3.3V
Header J5: VIOTA = 2.5V This header is used for the MicroBlaze serial console Pmod RS232 module.
Header J6: FMC_VADJ = 2.5V

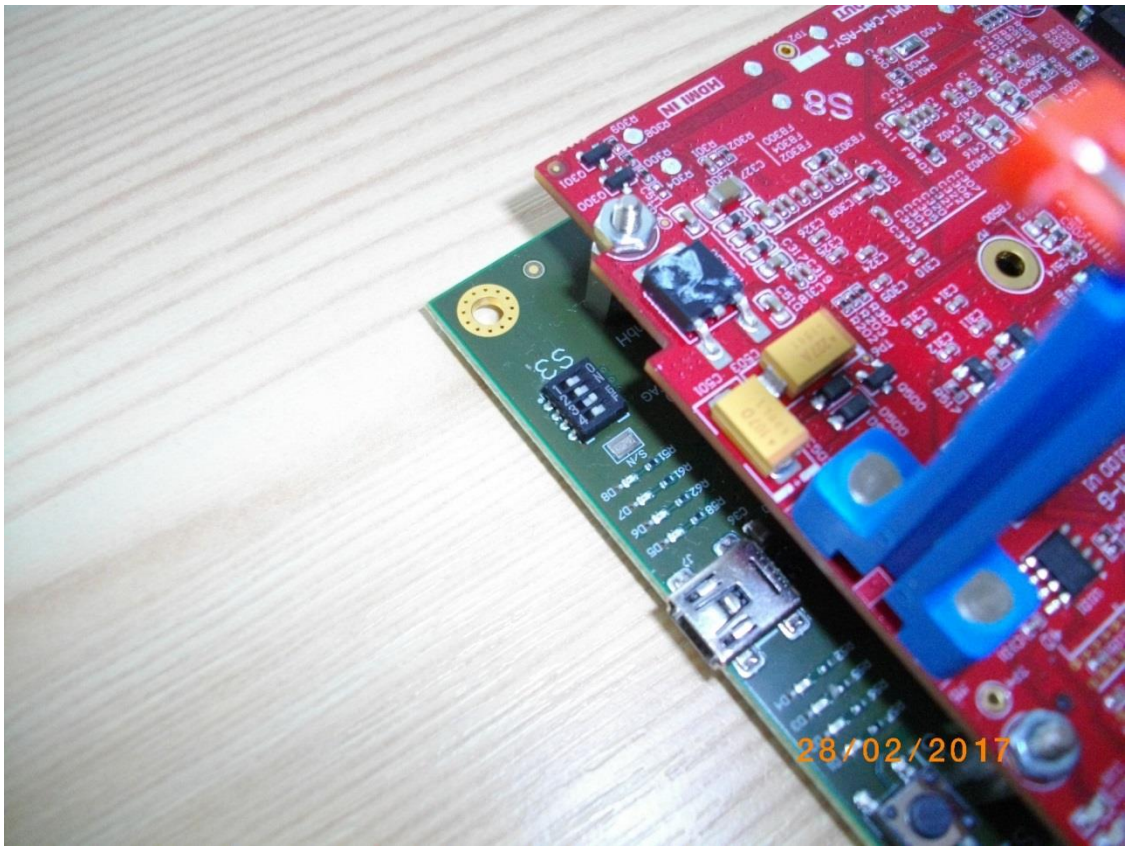


Figure 20: TE701-06 switch S3: 1=OFF 2=OFF 3=ON 4=OFF.

Figure 21: TE701-06: (VIOTA=2.5V and VADJ=2.5V) J17: connect 2-3; J16: open; J21: connect 2-3.

Figure 22: TE701-06 switch S4: (use FMC_VADJ=2.5V power source): 1=OFF 2=ON 3=ON 4=OFF.

Note:
In this setup of the TE0701-06 carrier board [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 with the power supply 2.5 V. Therefore, no special action is needed to fix it. If you decide to provide the correct range of the power supply for the PmodRS232 serial converter (3.3 V), you can make this simple HW adaptation: Disconnect the 2.5V wire and replace it with the 3.3 V for the PMODRS232 module.

This can be done by a custom made cable adapter. It disconnects the 2.5V power supply from J5 header and provides instead the 3.3V power supply. This supply can be taken from the PMOD J1 adapter pin 12.

See Figure 23, Figure 24 and Figure 25.

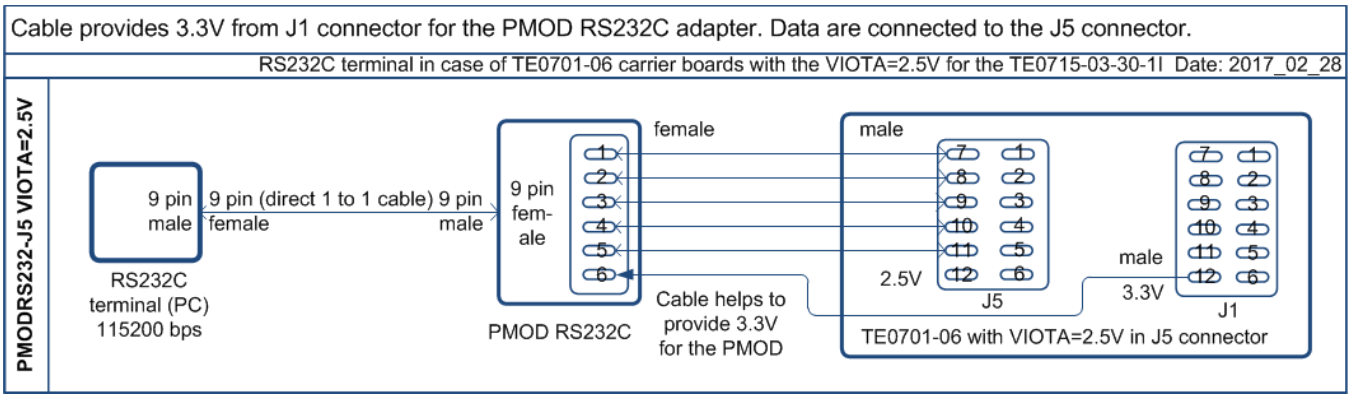


Figure 23: PMODRS232 cable connection to the TE701-06 set to with VIOTA=2.5 V.

All provided designs can work also without the RS232 terminal.

The terminal is used for a non-blocking output to PC terminal with the communication speed set to 115200 bps. The RS232 terminal output helps with debug and visualization of progress of MicroBlaze applications.

If the terminal is not present, consider the observation of the MicroBlaze application by stepping through the program in the SDK 2015.4 debug.

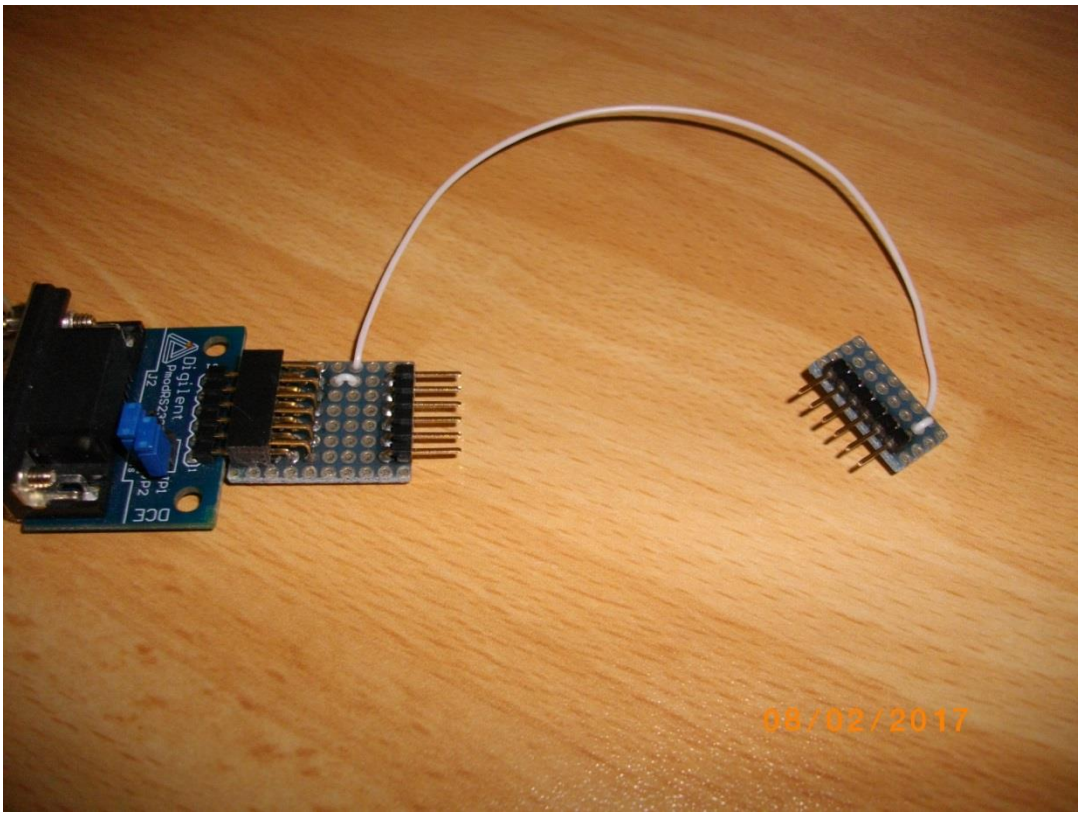


Figure 24: TE701-06: external 3.3V for the PMODRS232 at header J5 from header J1.

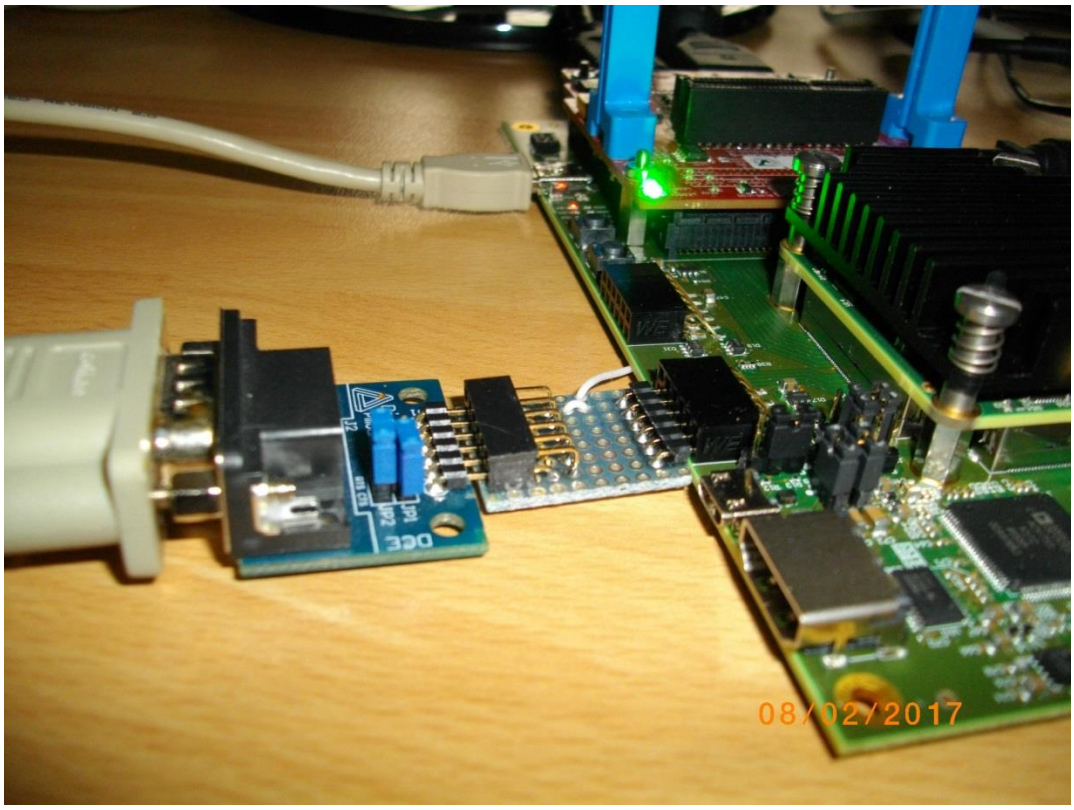


Figure 25: TE701-06 PMODRS232 connection.

2.3 Test demos

To test demos follow these steps:

- Insert the Python 1300 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 26. 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.

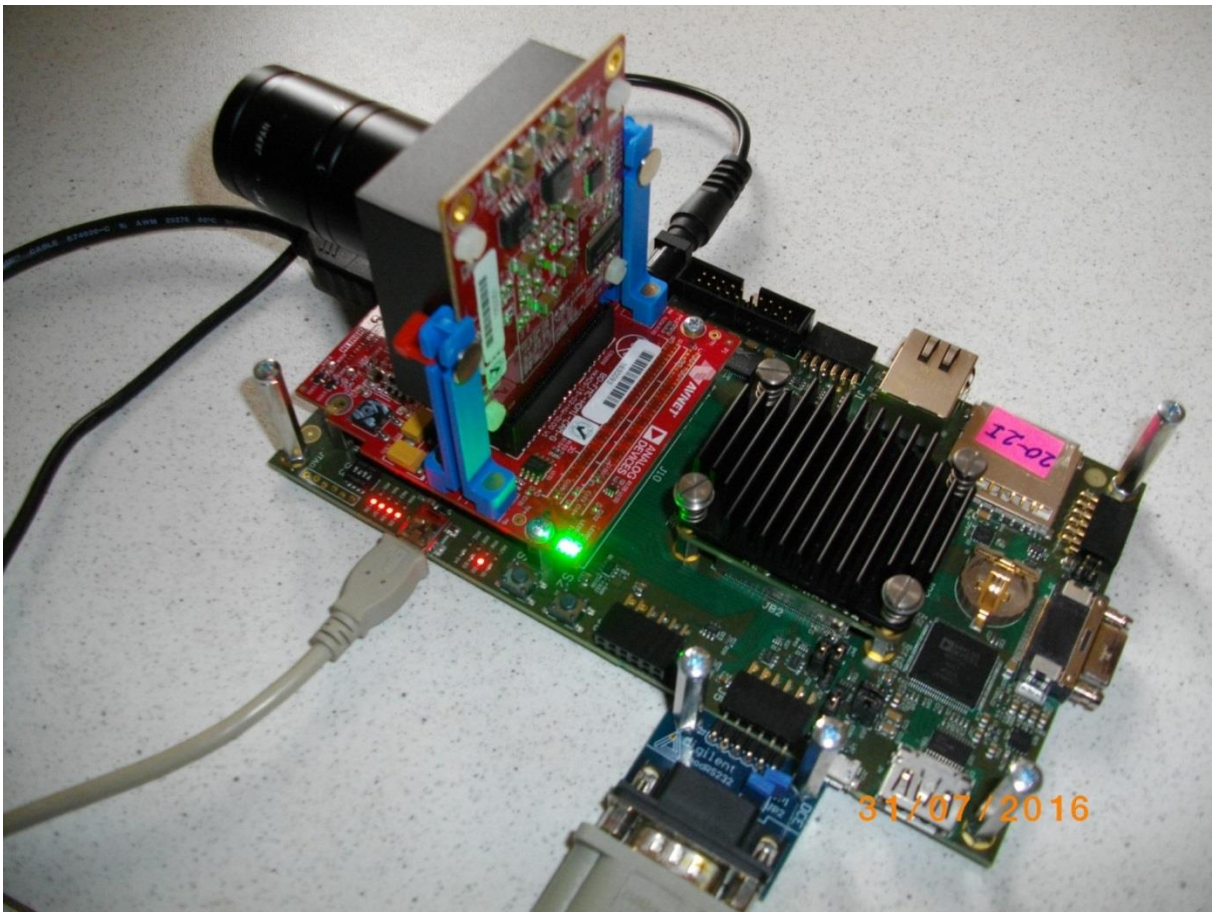


Figure 26: USB for ARM terminal and JTAG. RS232C Pmod for MicroBlaze.

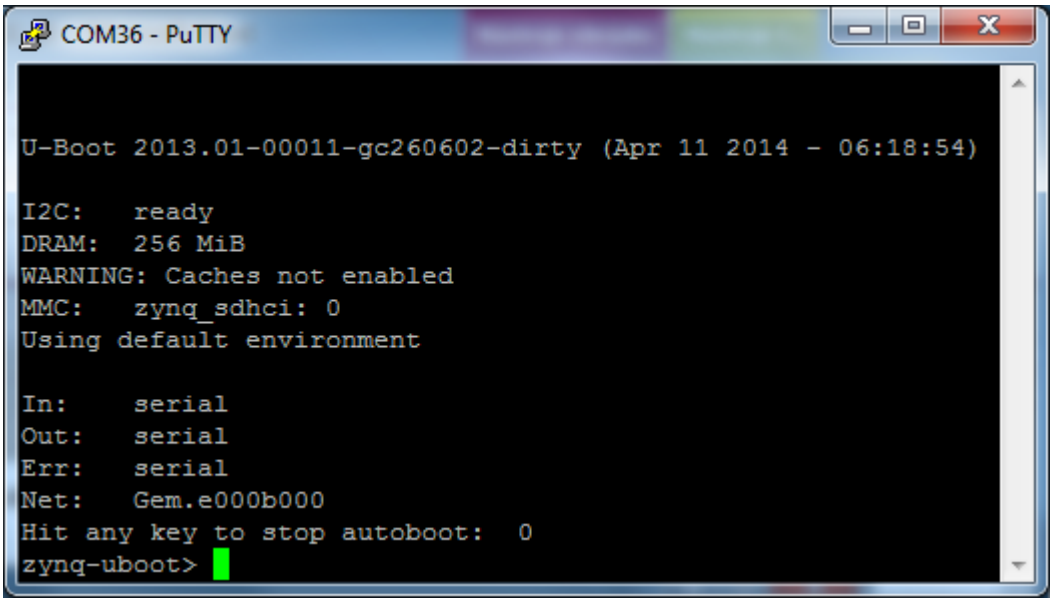


Figure 27: Serial console. Reset board and stop autoboot by any key.

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\t20i2pm4\sh02_hw_platform_0**

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

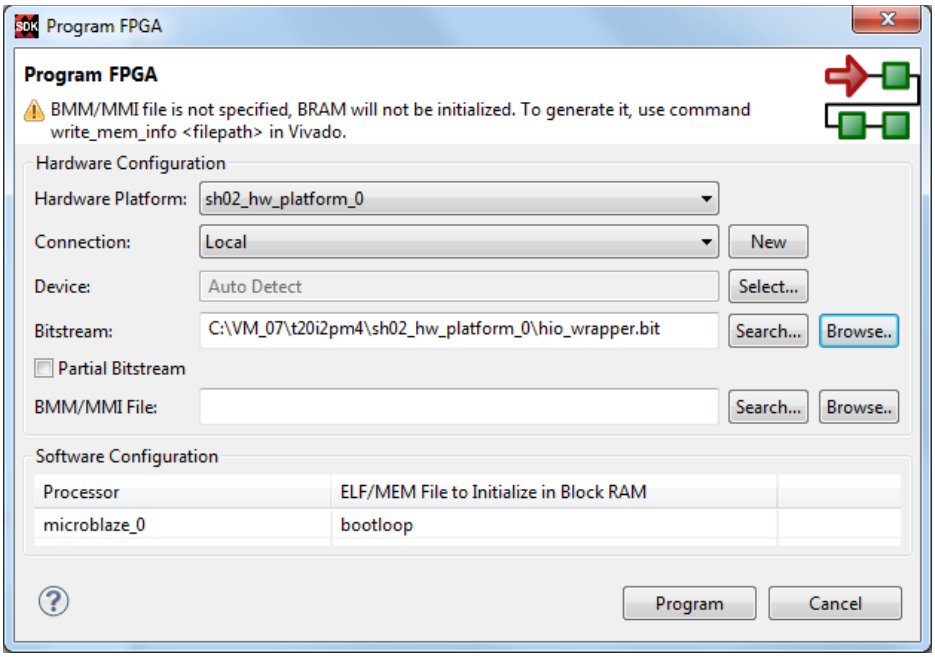


Figure 28: Download bitstream to the PL part of Zynq.

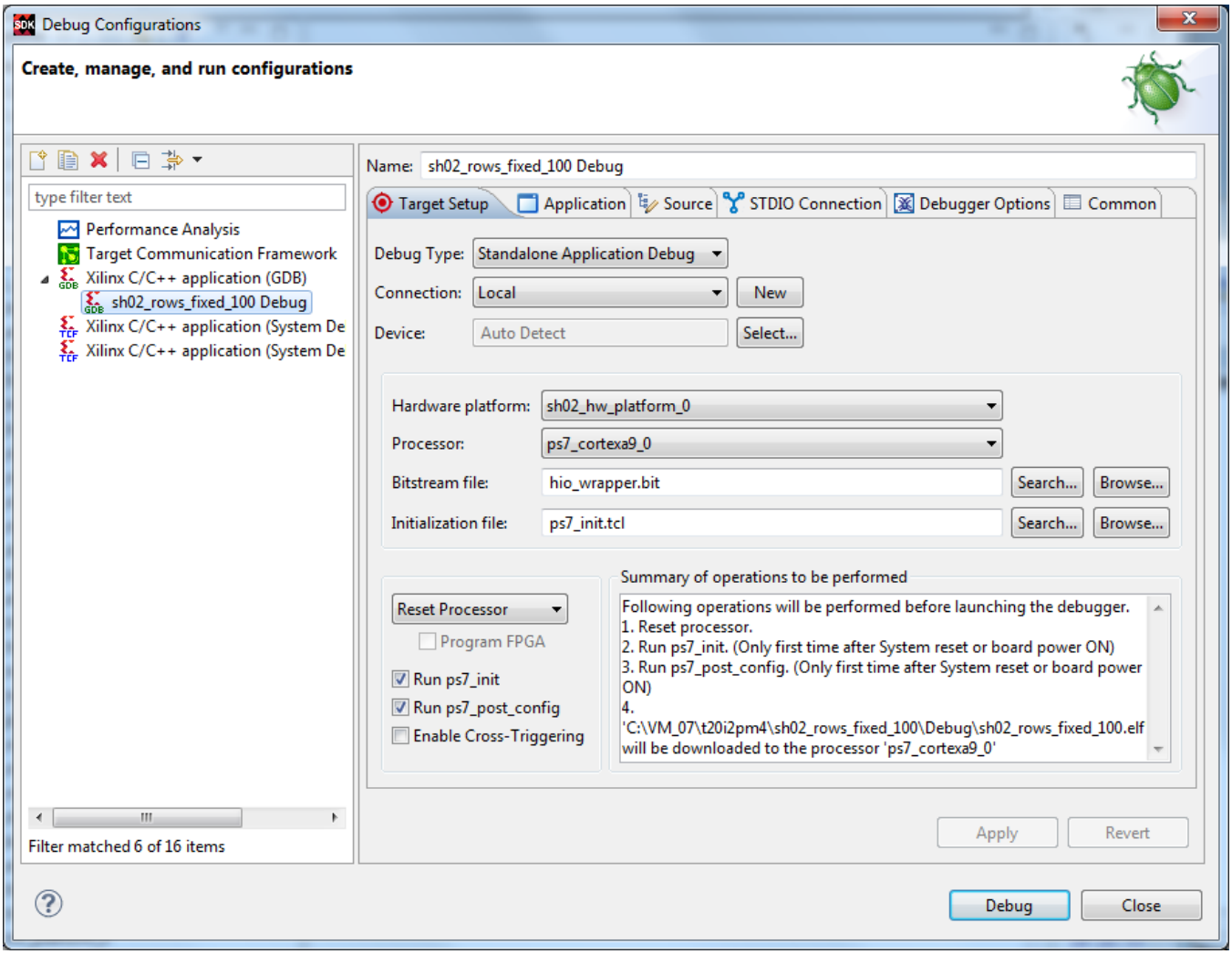


Figure 29: Select demo application for debug.

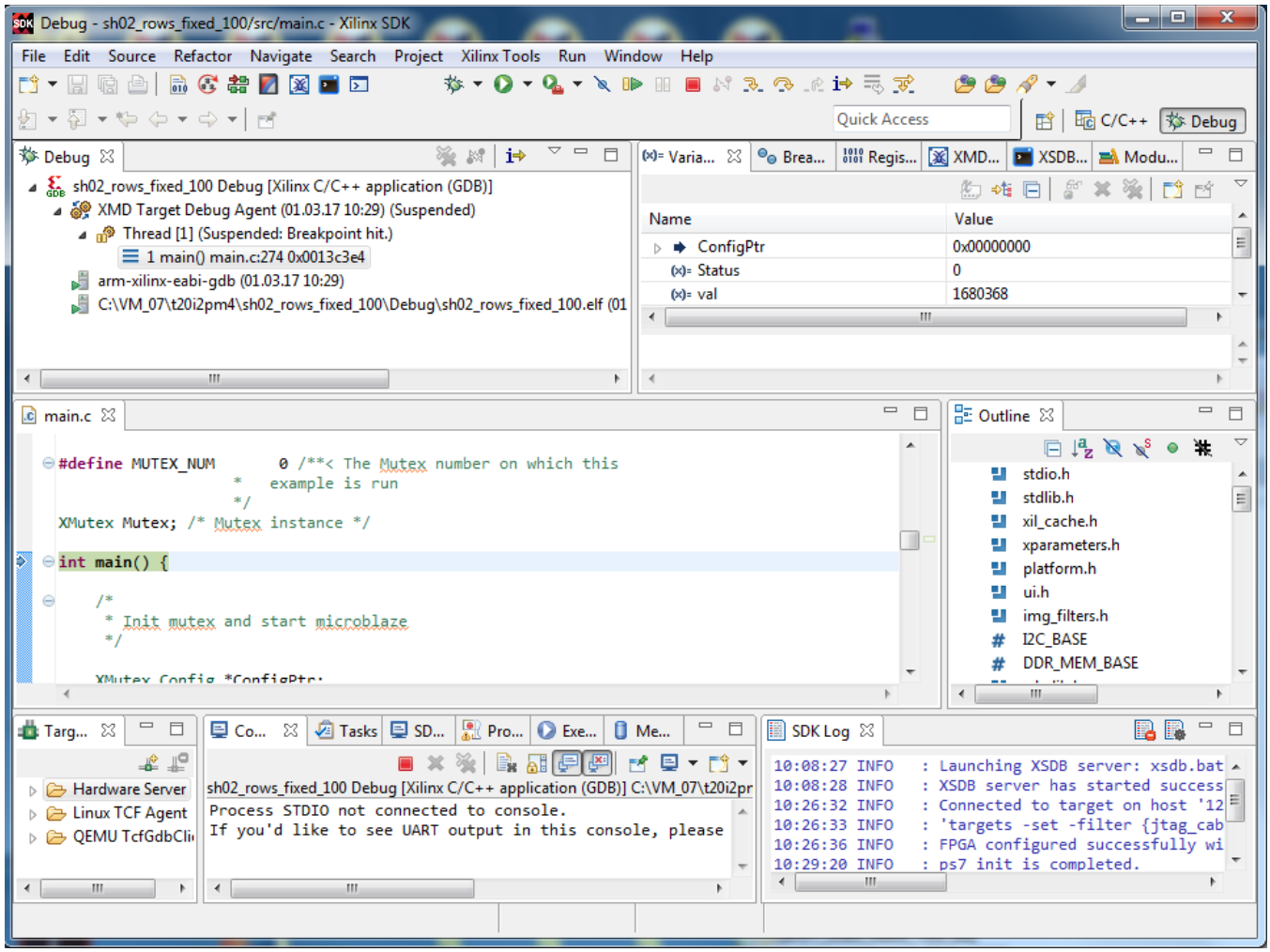


Figure 30: Demo app is booted to ARM and the debugger is waiting on the first executable line.

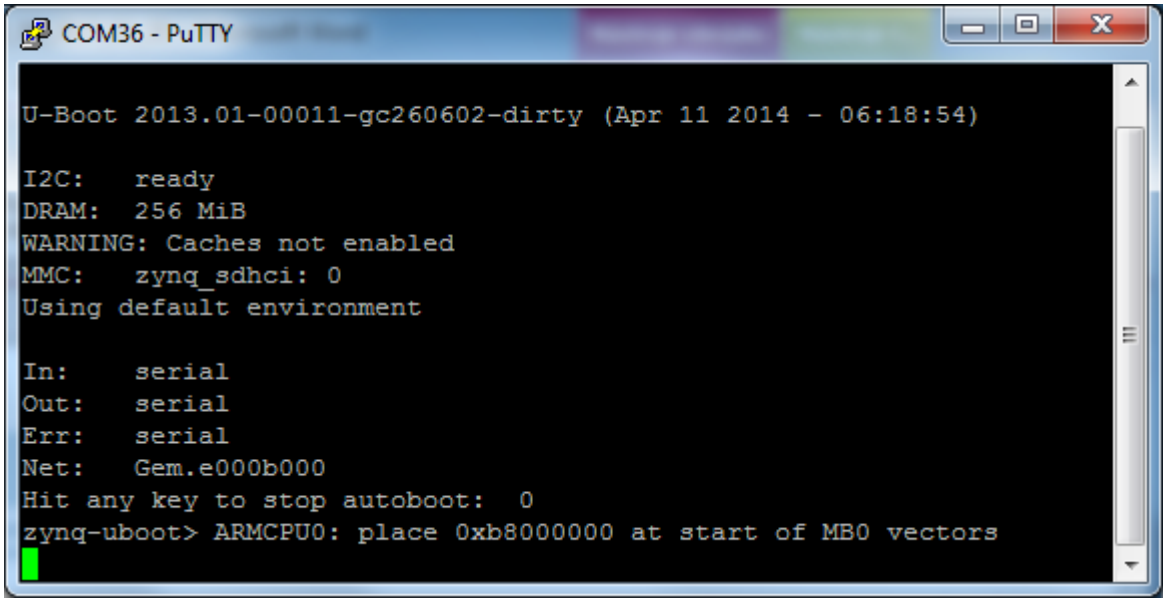


Figure 31: ARM is waiting on HW Mutex for the MicroBlaze start.

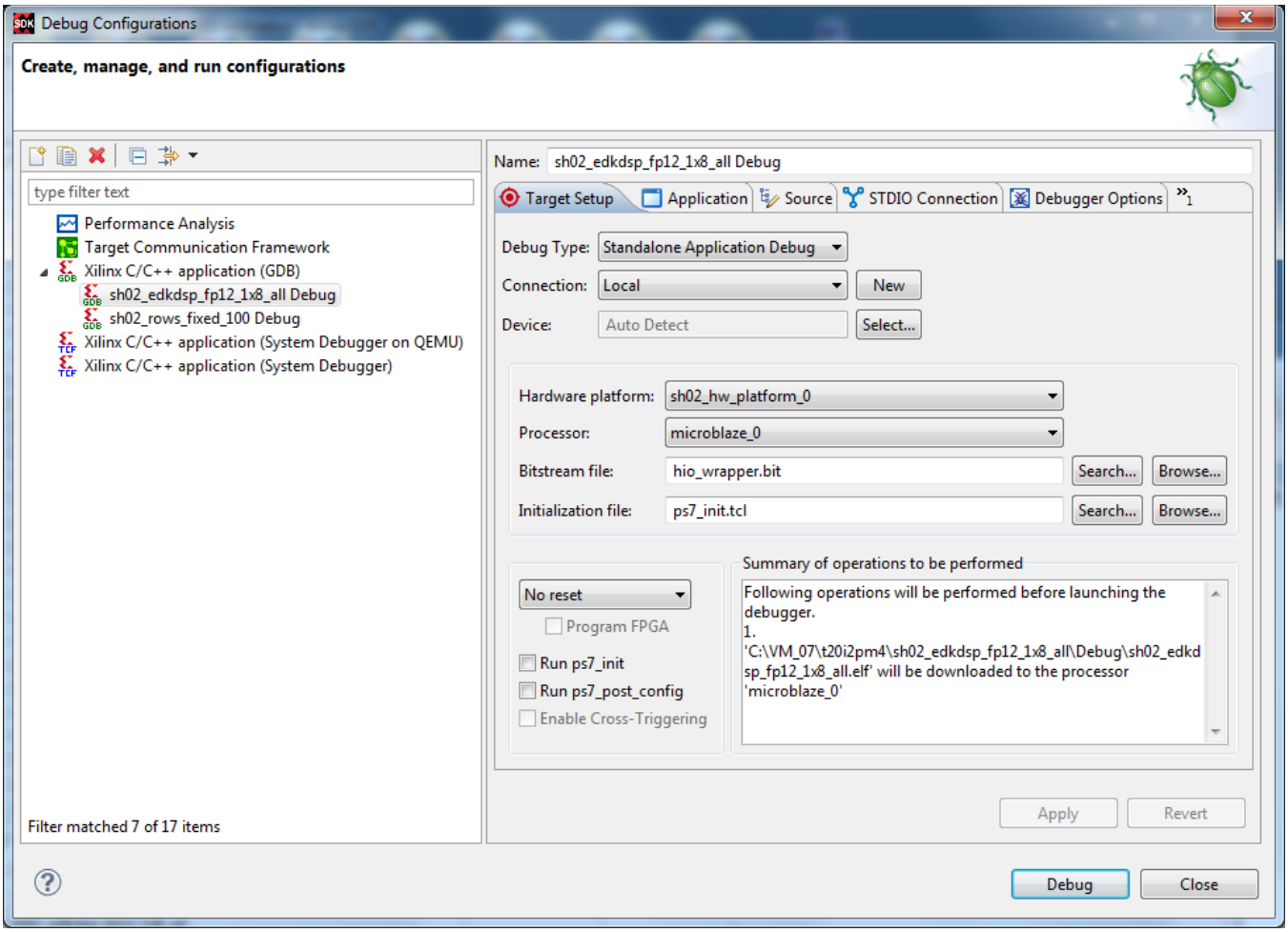


Figure 32: Select the MicroBlaze application (with the EdkDSP accelerator code) for debug.

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 32.

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 33.

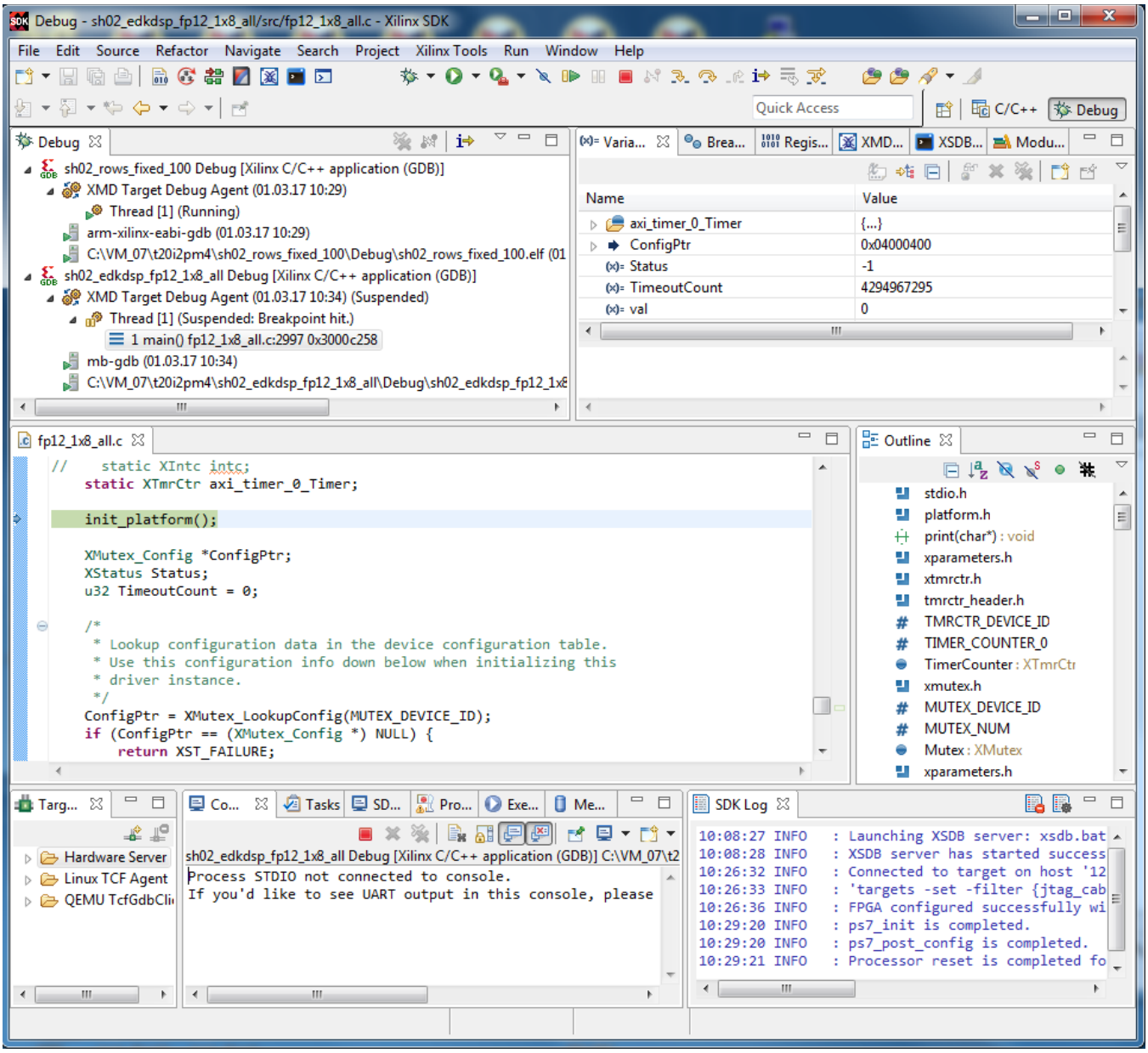


Figure 33: MicroBlaze application is loaded and debugger stops on the first instruction.

- The ARM SW thread is running.
- The MicroBlaze SW thread is currently suspended at breakpoint hit. See Figure 33.

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 Python 1300 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 35, 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.

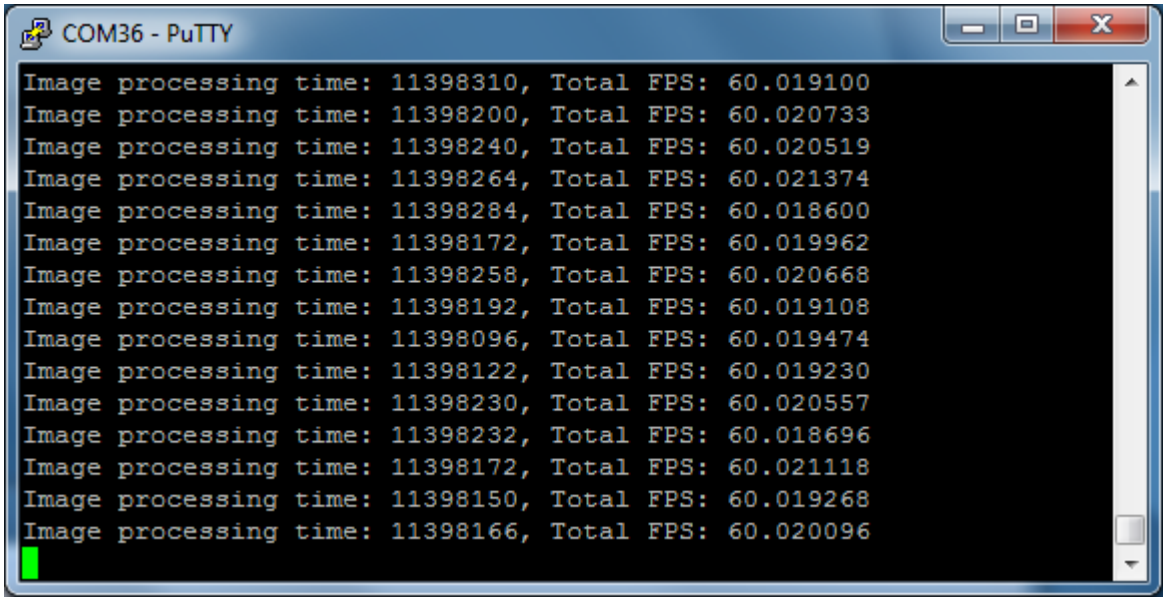


Figure 34: ARM is running in debug mode. It indicates the number of frames per second.

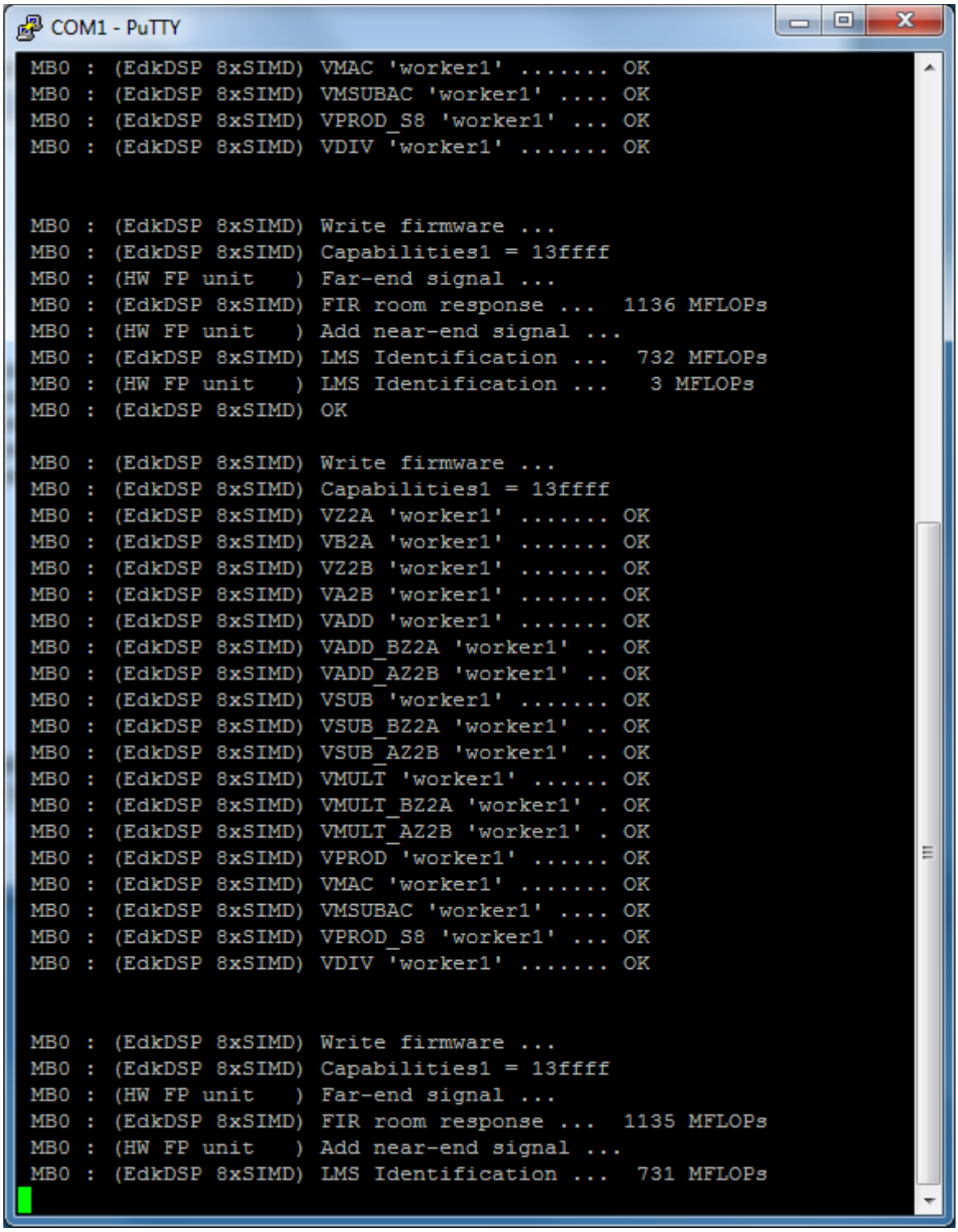


Figure 35: MicroBlaze is running in debug mode. It indicates MFLOPs.

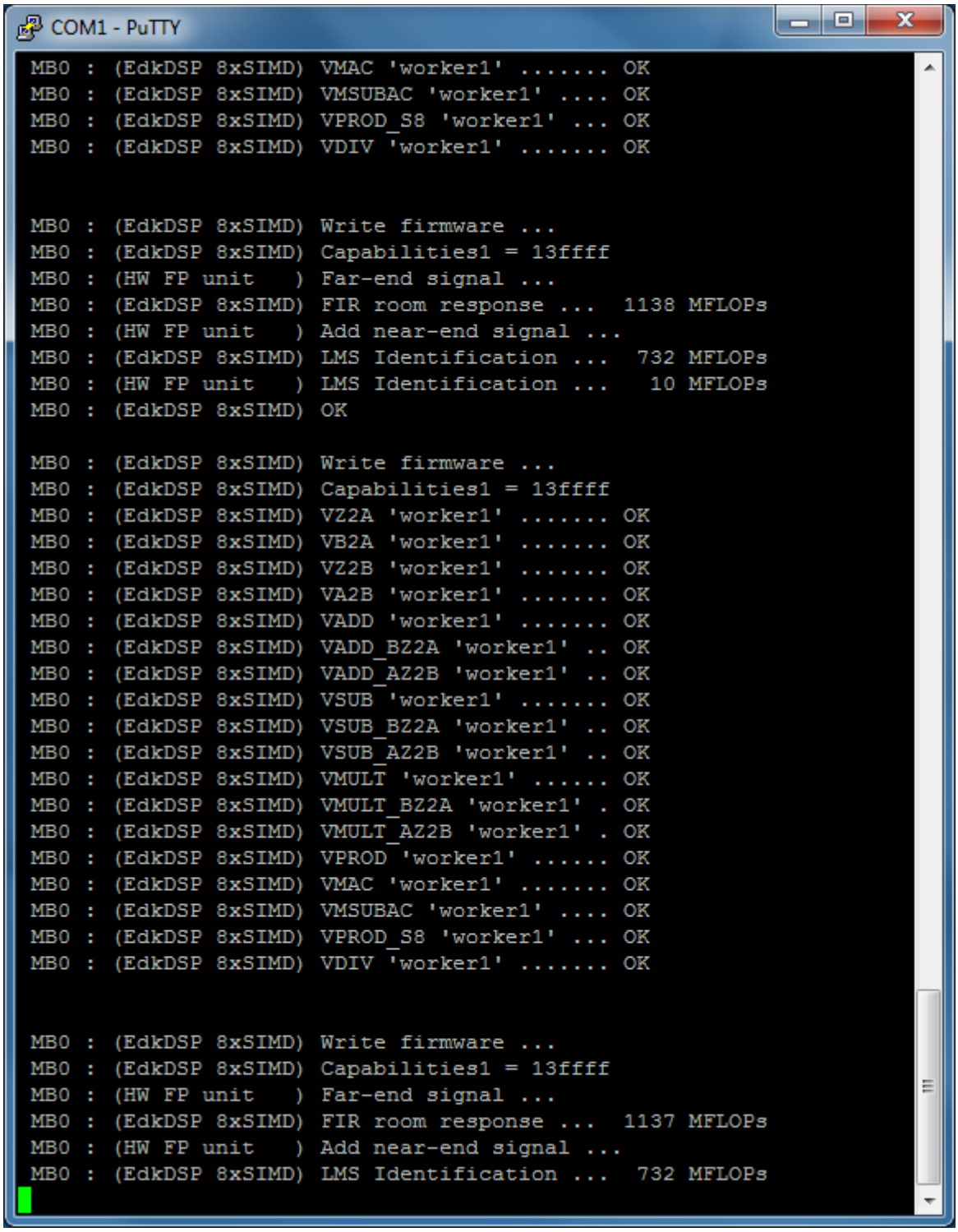


Figure 36: MicroBlaze is running in release mode. It indicates MFLOPs.

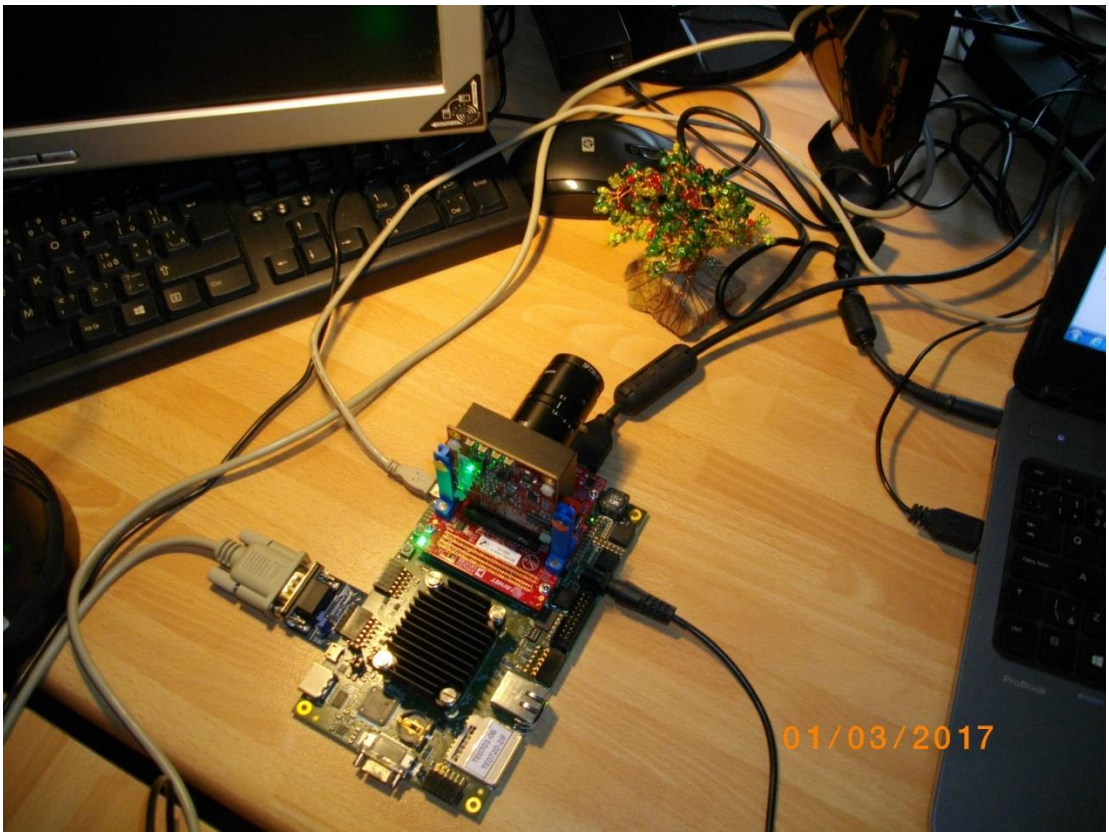


Figure 37: Accelerated edge detection Python 1300 sensor and Zynq with EdkDSP.

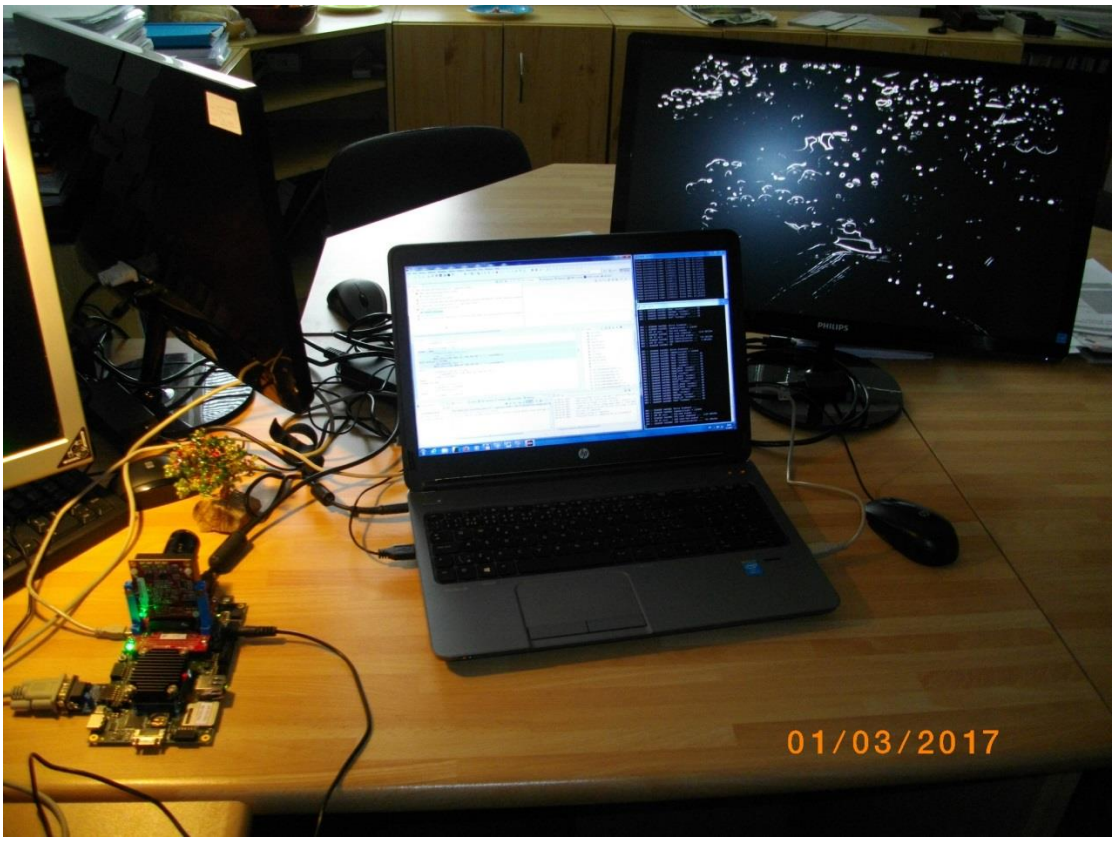


Figure 38: Edge detection (sh02 - Sobel filters, 2 variable HW data paths) output on HDMI monitor.

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 cases of programming models are described.

- User defined synchronisation with parallel HW data paths.
- Internal synchronisation with parallel HW data paths.

User defined synchronisation with parallel HW data paths (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\t20i2pm4\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 39: ARM C function with an internal synchronisation for chain of HW accelerators in a single data path.

Arm C code calls two functions (see *Figure 39* and *Figure 8*):

_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 40 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 40 and Figure 12.*

C:\VM_07\t20i2pm4\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 40: Listing of ARM C function with fixed data width interfacing HW pipeline of 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

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.

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 with MicroBlaze SW)
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.

2.6 EdkDSP C compiler

This section briefly describes how to use the UTIA EdkDSP C compiler. It cross-compiles (on PC) simple C programs for the PicoBlaze6 controller.

The evaluation package includes also precompiled firmware files for the PicoBlaze6 controller. These files can be used for the first evaluations of the EdkDSP accelerator before installation of the EdkDSP C cross compiler to user PC.

The UTIA EdkDSP C compiler is part of this evaluation package in form of Ubuntu binaries. The “VMware player” software with compatible Ubuntu image is needed to run the UTIA EdkDSP C compiler on Windows 7 PC.

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

Install VMware Workstation 12 Player [7] on Win 7 64 bit PC.

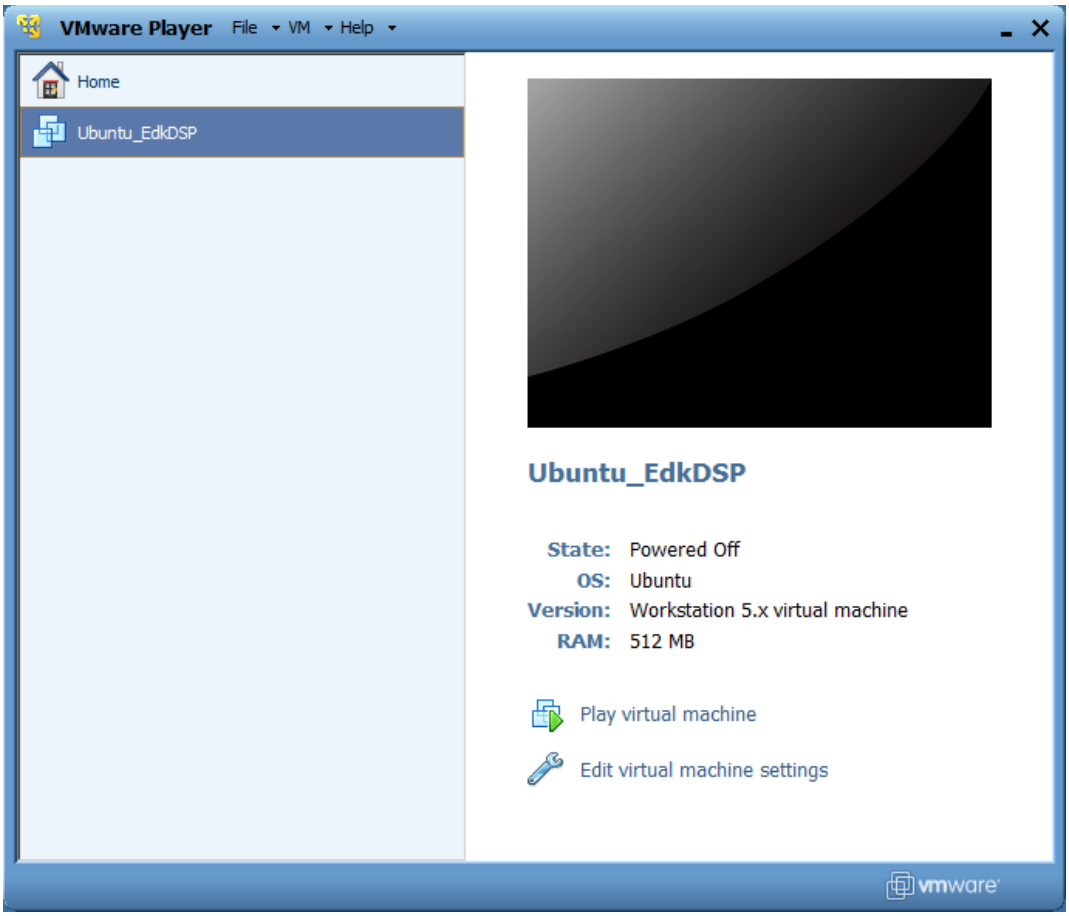


Figure 41: Select the Ubuntu_EdkDSP image in the VMware Player and click "Play".

Open the VMware Workstation 12 Player and select the "**Ubuntu_EdkDSP**" image. The Ubuntu will start.
Login as:

User: **devel**
Pswd: **devuser**

The PC directory **C:\VM_07** needs to be shared by Windows 7 with Ubuntu OS. In Windows 7, set the directory **C:\VM_07** and its subdirectories as shared with the **__vmware_user__** for Read and Write.

In Ubuntu, open terminal and mount the PC directory **C:\VM_07** to Ubuntu by typing:
cd bin
samba_07.sh

The Windows 7 **C:/VM_07** directory is mounted to the Ubuntu OS as: **/mnt/cdrive**

In Ubuntu terminal, change the directory to:

/mnt/cdrive/t20i2pm4/edkdsp

The EdkDSP C compiler utilities have to be on the Ubuntu PATH. This is done by sourcing the **settings.sh** script in this directory.

Type in Ubuntu terminal (See *Figure 42*):

source settings.sh

In Ubuntu terminal, change the directory to the example directory: **cd a**

/mnt/cdrive/t20i2pm4/edkdsp/a$

Provided C source code examples can be compiled by script **ca_fp11.sh** with parameter **a**.
Type in the Ubuntu terminal:

ca_fp11.sh a

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

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

Figure 43 presents C source code for the firmware for computation of the LMS in the (8xSIMD) EdkDSP platform. The FIR filter source code is presented in Figure *44*.

To use the compiled headers in the SDK project, copy and paste

edkdsp/a/fill_FA1101P0_program_store.h
edkdsp/a/fill_FA1101P1_program_store.h
edkdsp/a/fill_FA1124P0_program_store.h
edkdsp/a/fill_FA1124P0_program_store.h

to the SDK project directory (in case of **sh02_edkdsp_fp12_1x8_all**) to:

C:\VM_07\t20i2pm4\sh02_edkdsp_fp12_1x8_all\src

Recompile the MicroBlaze project "**sh02_edkdsp_fp12_1x8_all**". The compiled firmware for the (8xSIMD) EdkDSP will be used by the MicroBlaze C code of the demo as data for the runtime (re)configurations of the (8xSIMD) EdkDSP accelerator 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 in all included demos.

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

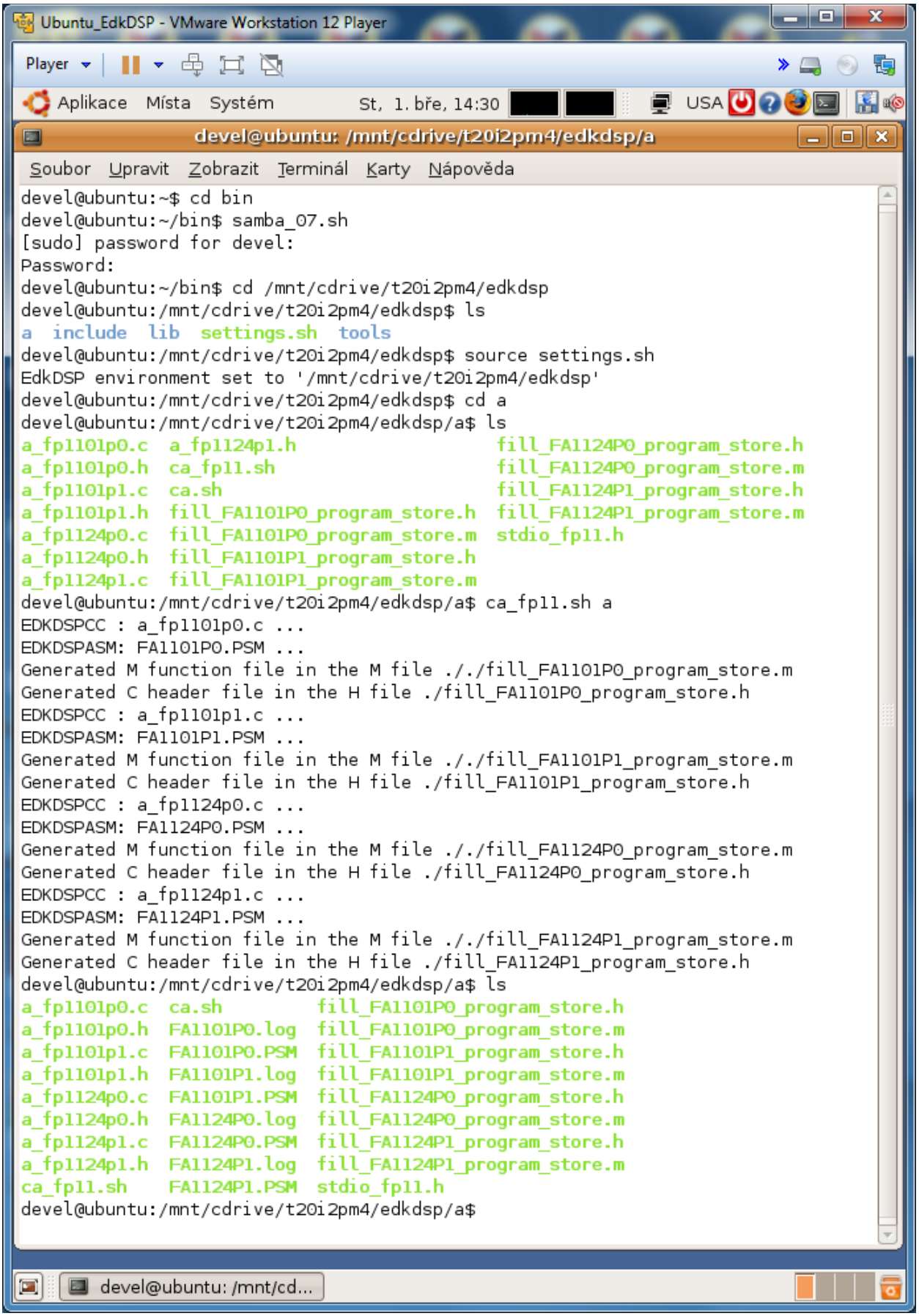


Figure 42: Compilation of EdkDSP firmware in Ubuntu.

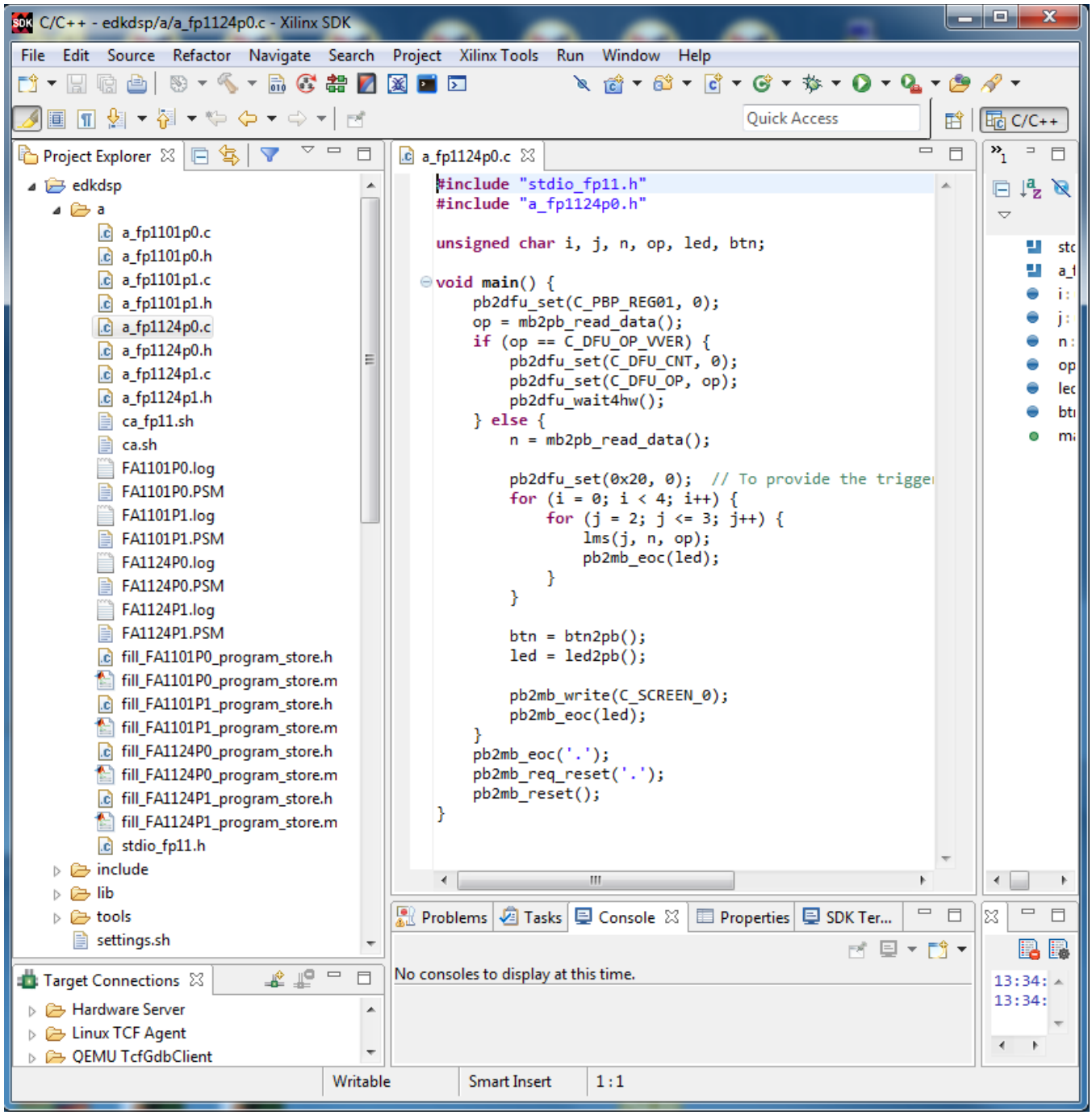


Figure 43: C listing of the LMS filter firmware for the EdkDSP.

Note:

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.

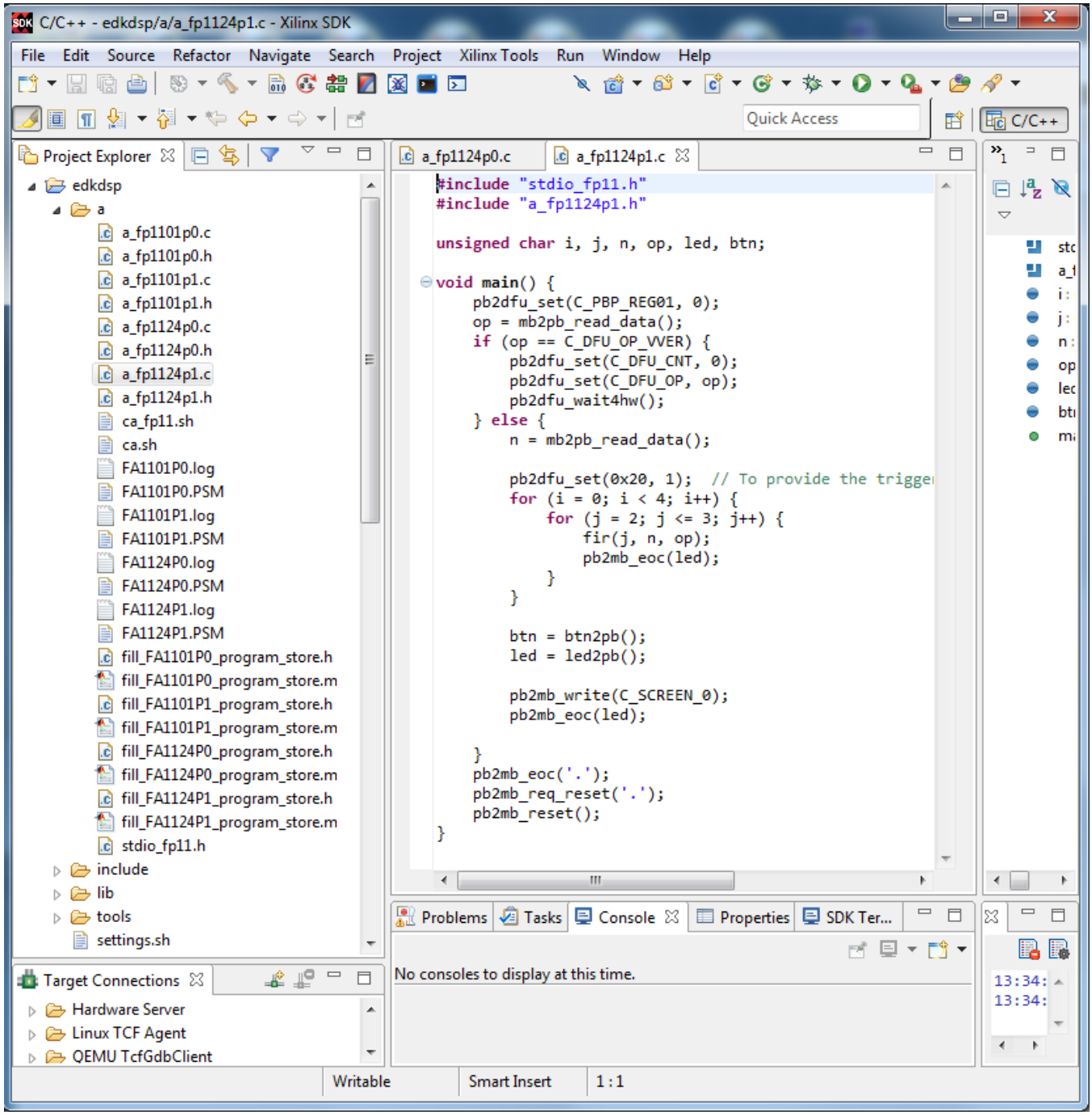


Figure 44: C listing of the FIR filter firmware for the EdkDSP.

2.7 Debug of EdkDSP accelerator firmware with In-circuit Logic Analyser (ILA)

This application note includes evaluation version of designs with ARM Cortex A9 processor and MicroBlaze processor controlling single (8xSIMD) EdkDSP accelerator. First of these accelerators is 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 present in the directory:

C:\VM_07\ t20i2pm4\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.

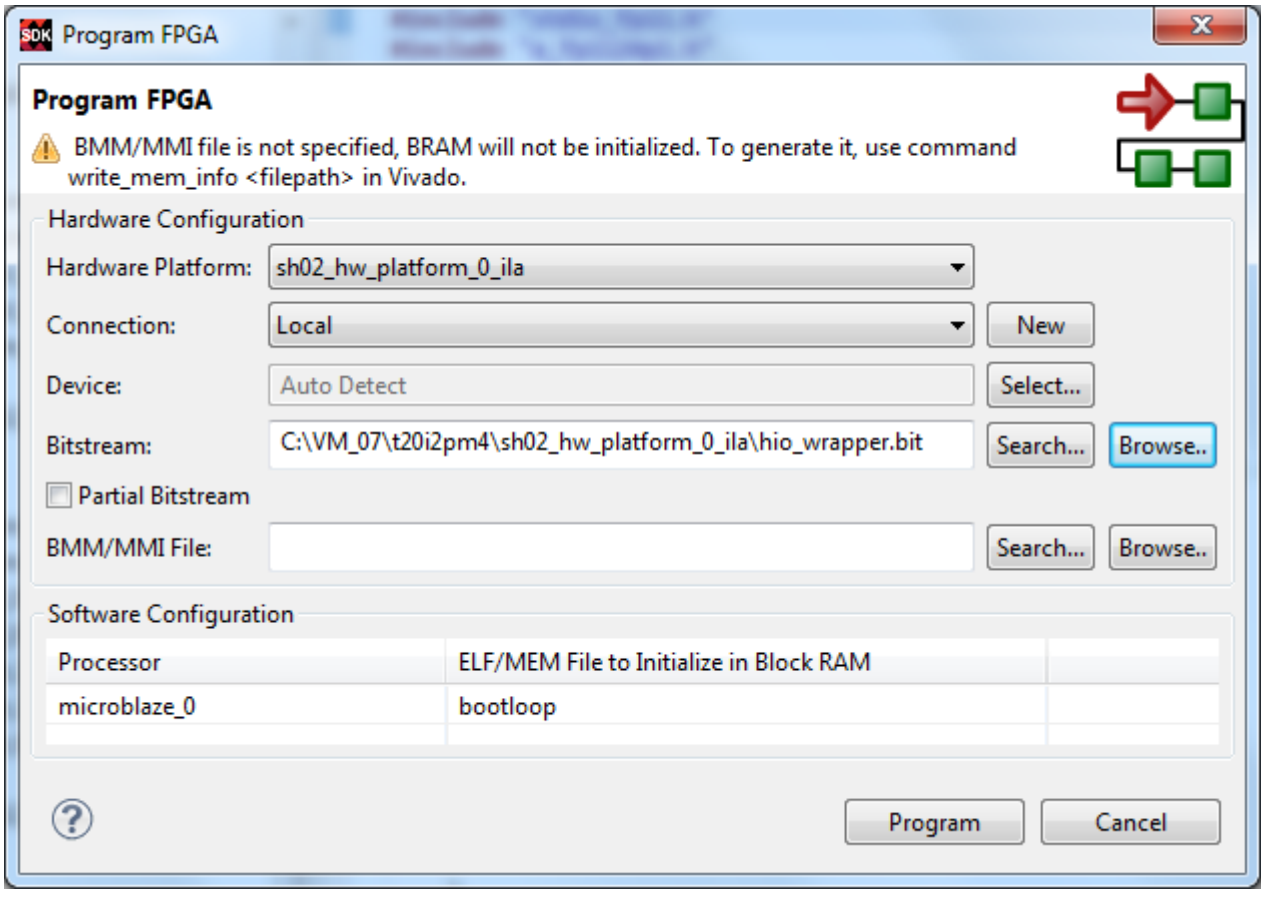


Figure 45: Change the default hw_platform_0 to the hw_platform_0_ila.

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. Prepared debug ILA environment provides synchronised time records of addresses and schedule of executed floating point vector operations.

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 at synchronising points defined by programmer.

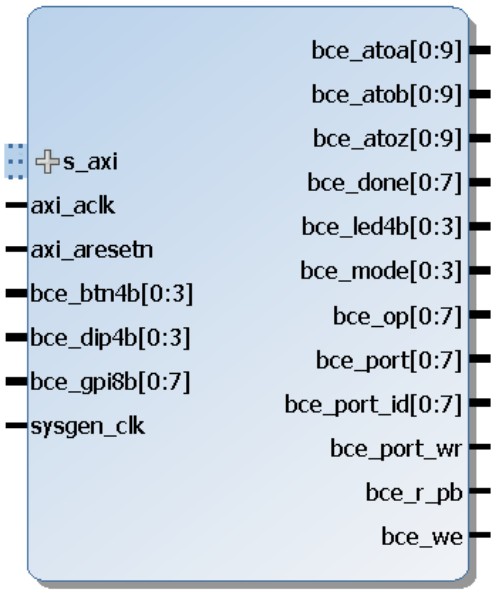


Figure 46: Debug ports of the (8xSIMD) EdkDSP floating point accelerator IP core.

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 46*):

- 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.

All vector operations of the (8xSIMD) EdkDSP **bce_fp12_1x8_40** accelerator can be monitored at the **bce_op[0:7]** debug port. These 8xSIMD vector operations are 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

2.8 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 *47*.

Figure 47: Vivado Lab Edition 2015.4.

Select: **Open Target** See Figure *48*. 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 46) 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/t20i2pm4/sh02_hw_platform_0_ila/ debug_nets.ltx
See Figure *49*.

This will add the names and parameters of probes (see Figure 46) 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).

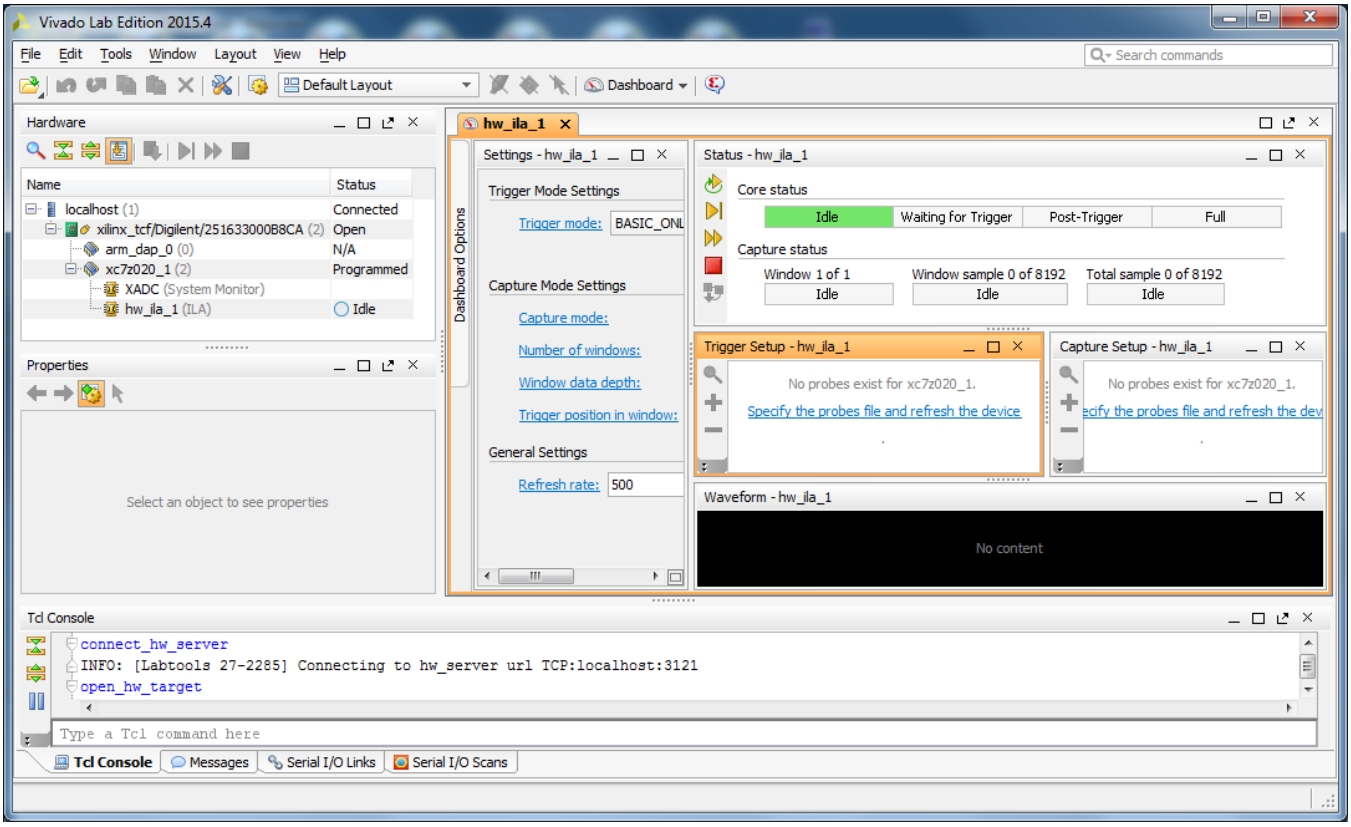


Figure 48: Select Open Target.

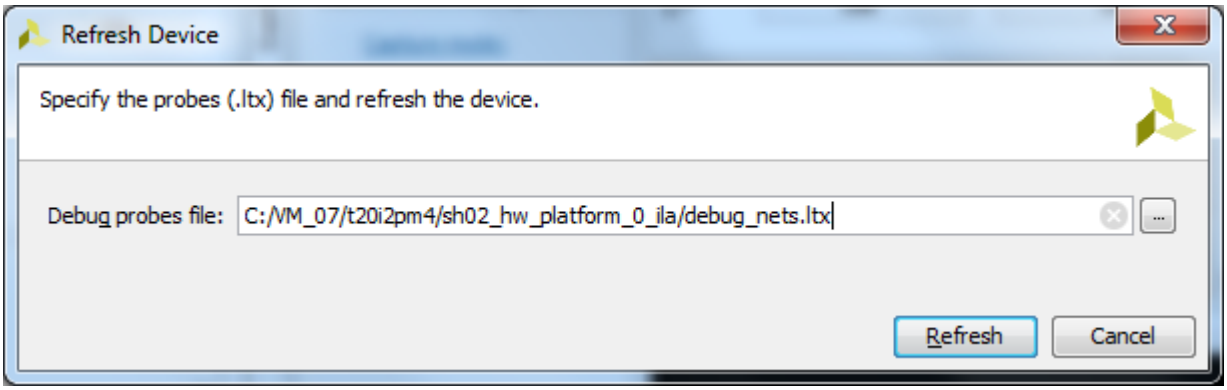


Figure 49: Select file with definition of probes present in HW.

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

In SDK, open the **edkdsp_cc/a/a_fp1124p1.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_FA1124P1_program_store.h** contains firmware resulting from compilation of C source code **a_fp1124p1.c** (see *Figure 44*).

…

```
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 44*. The red trigger corresponds to the event. See Figure *50*.

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.

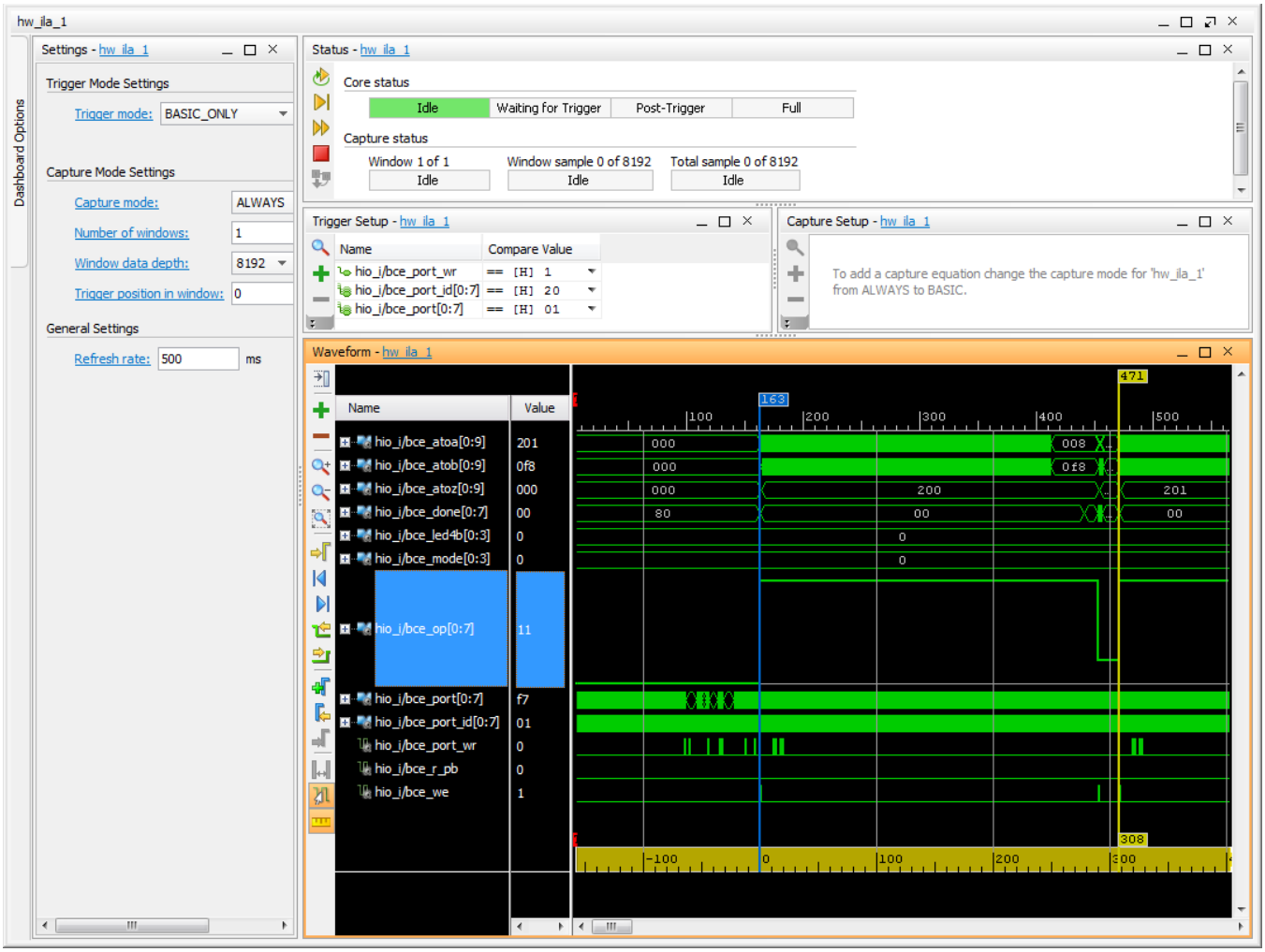


Figure 50: FIR filter waveforms after the trigger in Vivado Lab Edition 2015.4.

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_cc/a/a_fp1124p0.c** code implementing the LMS filter on the identical EdkDSP HW. See *Figure 43*.

```
…
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 43*). 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 *51*.

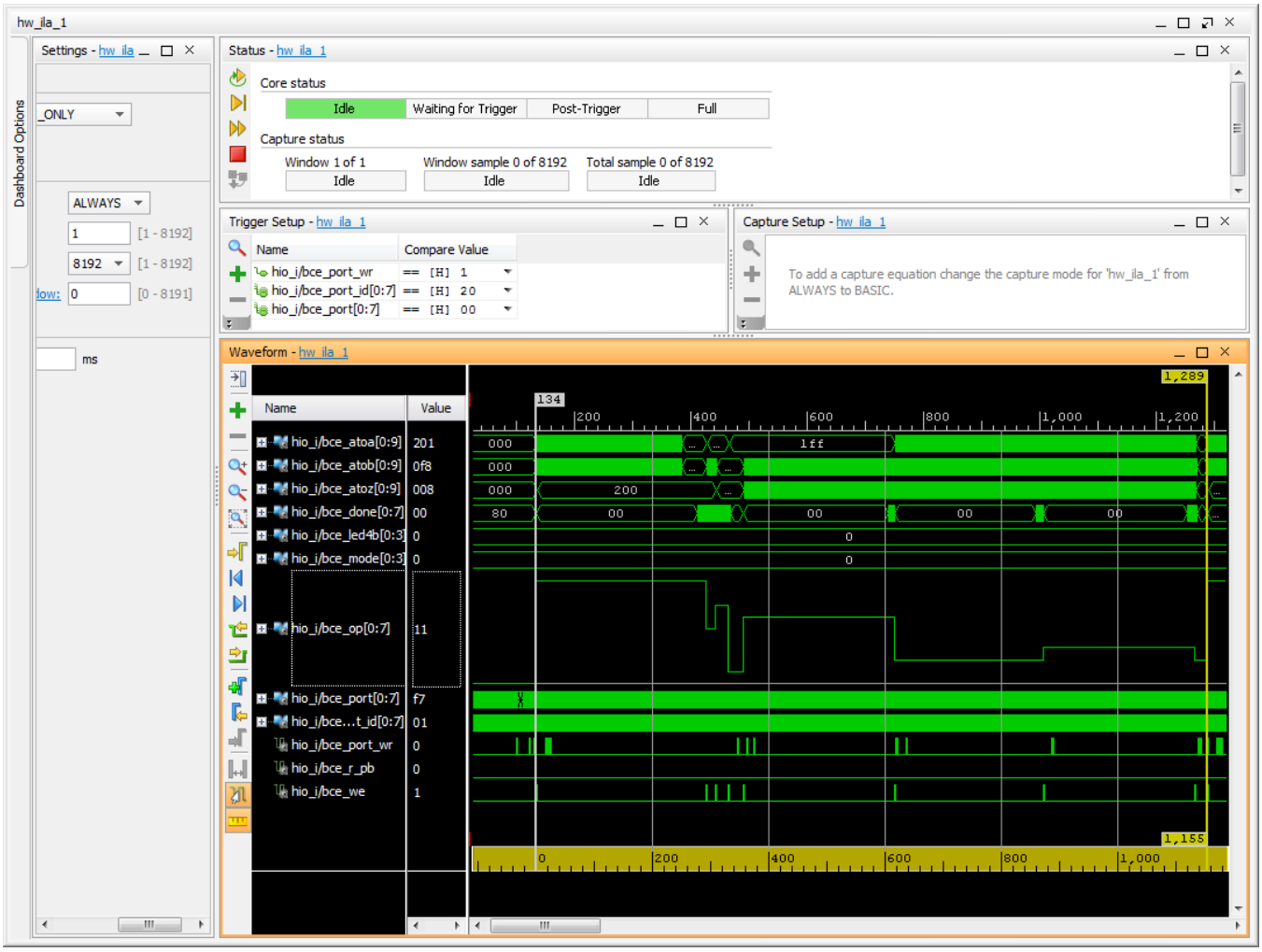


Figure 51: LMS filter waveforms after the trigger in Vivado Lab Edition 2015.4.

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 43*, 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.

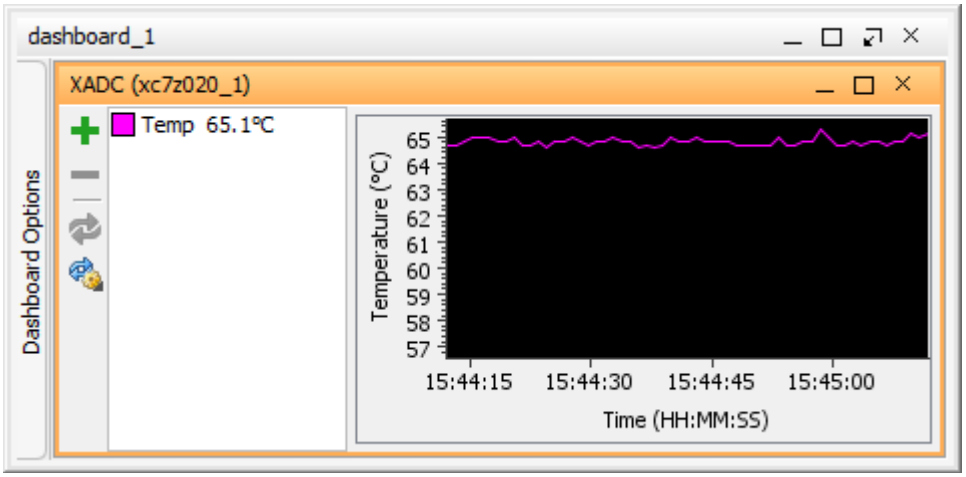


Figure 52: Separate dashboard with display of temperature and voltage in Vivado Lab Edition 2015.4.

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 *52*. 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 edge detection together with floating point computation of FIR and LMS filter and the ILA debug facility on *Figure 53*.

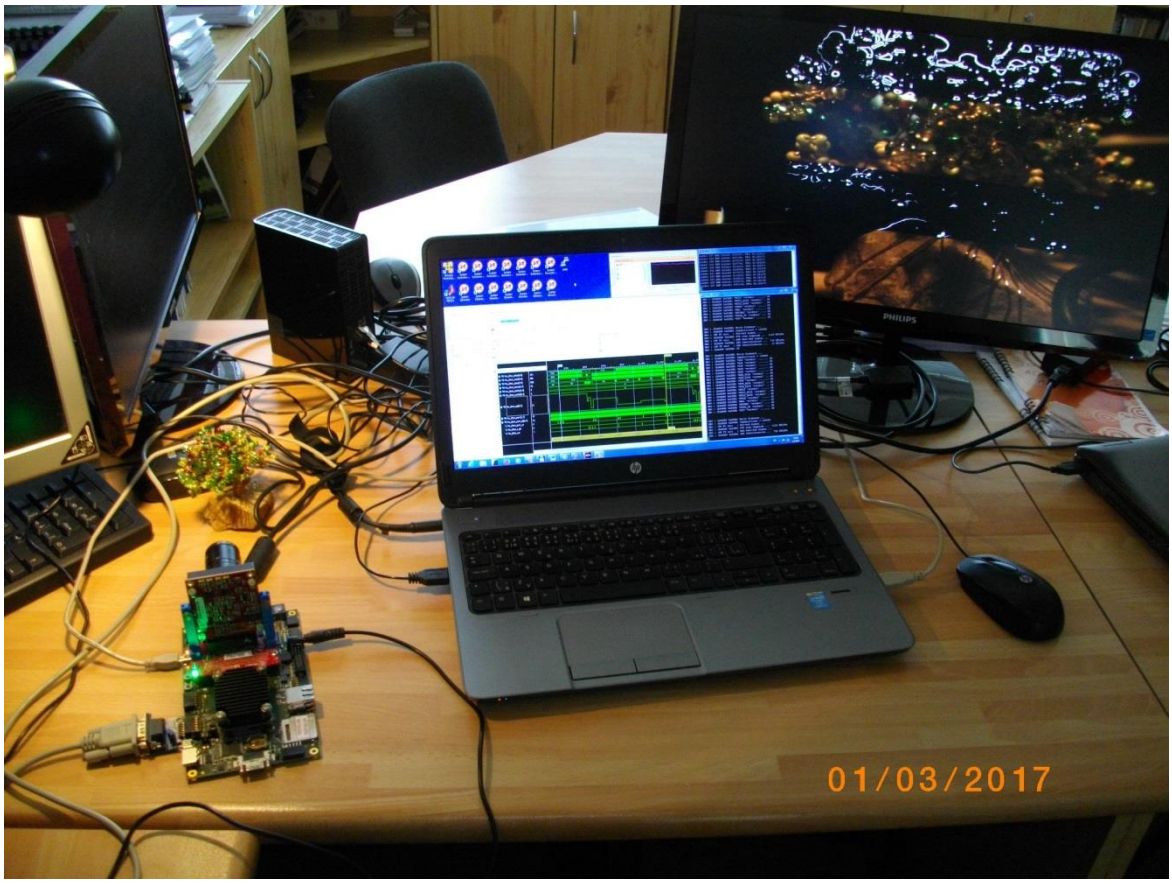


Figure 53: Accelerated edge detection algorithm and ILA debug of the accelerated LMS filter.

Figure 54: 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 is capable of implementation:
  - (8xSIMD) EdkDSP floating point accelerator in parallel with
  - HW accelerated video processing chain for the Python 1300 color sensor with fixed resolution 1280x1024p60.

- 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 brings additional capability to compute (in single precision floating point) with performance in the range of around one GFLOP/s (1.139 GFLOP/s in case of FIR filter) at the expense of relatively moderate increase of total power consumption of the system:

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

- MicroBlaze soft core with single (8xSIMD) EdkDSP accelerator takes significant part of Zynq PL resources. This limits the maximal number of parallel HW video processing chains and therefore limits the achievable reduction of the energy per pixel.

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 SW adaptations and customisations of the final application in the C source code. The run-time re-programming of the 8xSIMD EdkDSP accelerator in C and ASM is also supported.

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 Python 1300 and the EdkDSP platform.

Application note explained integration of the Python 1300 sensor with the HW accelerated video processing algorithms and single run-time reprogrammable 8xSIMD EdkDSP floating point accelerator.

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

[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

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

[5] AES-CAM-ON-P1300C-G PYTHON-1300 Color Image Sensor Camera Module
AVNET: Home / Products / Kits and Tools / Development Kits / AES-CAM-ON-P1300C-G
https://products.avnet.com/shop/en/ema/kits-and-tools/development-kits/aes-cam-on-p1300c-g-3074457345628965500/

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

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

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 accelerators working in parallel with the HW-accelerated edge detection and motion detection algorithms for video processing on the Trenz Electronic TE0720-03-2IF module [1] located on the Trenz Electronic TE0701-06 carrier [3] with the FMC card [4], PYTHON-1300 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 TE0720-03-2IF module [1] on Trenz Electronic TE701-06 carrier board [3].

The evaluation package [11] includes SDK 2015.4 SW projects with C source code for the 1 GHz ARM Cortex A9 processor (32bit) in standalone mode, C source code for MicroBlaze and C/ASM 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
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 this evaluation package.

The evaluation package [11] includes these binary applications for Ubuntu:

edkdsppp	EdkDSP C pre-processor binary for Ubuntu in VMware Workstation 12 Player.
edkdspcc	EdkDSP C compiler binary for Ubuntu in VMware Workstation 12 Player.
edkdspasm	EdkDSP ASM compiler binary for Ubuntu in VMware Workstation 12 Player.

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 EdkDSP accelerators 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 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].

The evaluation package [11] also includes compiled versions of this firmware in form of header files .h. These compiled firmware files can be used for initial test of the UTIA EdkDSP accelerators on the Trenz Electronic TE0720-03-2IF module [1] on the Trenz Electronic TE0701-06 carrier board [3] without the need to install the UTIA compiler binaries and the Ubuntu image under the VMware Workstation 12 Player [7].

On email request to kadlec@utia.cas.cz , UTIA will send DVD with the Ubuntu image with pre-installed compiler binary files free of charge. The image can be played in the VMware Workstation 12 Player [7].

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 Artemis Almarvi Project Partners

This evaluation package includes **Vivado 2015.4 projects** for the Trenz Electronic TE0720-03-2IF module [1] located on the Trenz Electronic TE0701-06 carrier [3] with the FMC card [4], PYTHON-1300 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 Artemis Almarvi 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 Almarvi 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 TE0720-03-2IF module [1] located on the Trenz Electronic TE0701-06 carrier [3] with the FMC card [4], PYTHON-1300 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 Artemis Almarvi project**
> **(Without VAT) 0,00 Eur**

After receiving confirmation from the Almarvi 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 ALMARVI partners [10] includes the Vivado 2015.4 design projects which can be modified and recompiled by the Almarvi 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 (8xSIMS) EdkDSP accelerator has an HW limit on number of vector operations.

ALMARVI 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 XC7Z020-2I 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 ALMARVI 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 ALMARVI partners includes SDK 2015.4 SW projects with C source code for the 1 GHz 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
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 this evaluation package.

The evaluation package for ALMARVI partners includes SDK 2015.4 SW projects with source code for MicroBlaze processor and ARM 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 evaluation package for ALMARVI partners includes these binary applications for Ubuntu:

edkdsppp	EdkDSP C pre-processor binary for Ubuntu in VMware Workstation 12 Player.
edkdspcc	EdkDSP C compiler binary for Ubuntu in VMware Workstation 12 Player.
edkdspasm	EdkDSP ASM compiler binary for Ubuntu in VMware Workstation 12 Player.

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 EdkDSP accelerators 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 ALMARVI partners 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].

The evaluation package for ALMARVI partners also includes compiled versions of this firmware in form of header files .h. These compiled firmware files can be used for initial test of the UTIA EdkDSP accelerators on the Trenz Electronic TE0720-03-2IF module [1] on the Trenz Electronic TE0701-06 carrier board [3] without the need to install the UTIA compiler binaries and the Ubuntu image under the VMware Workstation 12 Player [7].

On email request to kadlec@utia.cas.cz , UTIA will send DVD with the Ubuntu image with pre-installed compiler binary files free of charge. The image can be played in the VMware Workstation 12 Player [7].

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 IPs with no HW Limit on Number of Vector Operations

This release package includes **Vivado 2015.4 projects** for the Trenz Electronic TE0720-03-2IF module [1] located on the Trenz Electronic TE0701-06 carrier [3] with the FMC card [4], the PYTHON-1300 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 TE0720-03-2IF module [1] located on the Trenz Electronic TE0701-06 carrier [3] with the FMC card [4], the PYTHON-1300 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 1 GHz ARM Cortex A9 processor (32bit) in standalone mode, C source code for MicroBlaze and C source code for the EdkDSP PicoBlaze6 controller.

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 binary applications for Ubuntu:

edkdsppp	EdkDSP C pre-processor binary for Ubuntu in VMware Workstation 12 Player.
edkdspcc	EdkDSP C compiler binary for Ubuntu in VMware Workstation 12 Player.
edkdspasm	EdkDSP ASM compiler binary for Ubuntu in VMware Workstation 12 Player.

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 EdkDSP accelerators 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].

The release package also includes compiled versions of this firmware in form of header files .h. These compiled firmware files can be used for initial test of the UTIA EdkDSP accelerators on the Trenz Electronic TE0720-03-2IF module [1] on the Trenz Electronic TE0701-06 carrier board [3] without the need to install the UTIA compiler binaries and the Ubuntu image under the VMware Workstation 12 Player [7].

On email request to kadlec@utia.cas.cz , UTIA will send DVD with the Ubuntu image with pre-installed compiler binary files free of charge. The image can be played in the VMware Workstation 12 Player [7].

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

This disclaimer is not a license and does not grant any rights to the materials distributed herewith. Except as otherwise provided in a valid license issued to you by UTIA AV CR v.v.i., and to the maximum extent permitted by applicable law:

(1) THIS APPLICATION NOTE AND RELATED MATERIALS LISTED IN THIS PACKAGE CONTENT ARE MADE AVAILABLE "AS IS" AND WITH ALL FAULTS, AND UTIA AV CR V.V.I. HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and
(2) UTIA AV CR v.v.i. shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under or in connection with these materials, including for any direct, or any indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or UTIA AV CR v.v.i. had been advised of the possibility of the same.

Critical Applications:
UTIA AV CR v.v.i. products are not designed or intended to be fail-safe, or for use in any application requiring fail-safe performance, such as life-support or safety devices or systems, Class III medical devices, nuclear facilities, applications related to the deployment of airbags, or any other applications that could lead to death, personal injury, or severe property or environmental damage (individually and collectively, "Critical Applications"). Customer assumes the sole risk and liability of any use of UTIA AV CR v.v.i. products in Critical Applications, subject only to applicable laws and regulations governing limitations on product liability.