Application Note

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

Full HD Video Processing in HW with three EdkDSP 8xSIMD Accelerators for TE0715-04-30-3E SoM on 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	15.02.2017	Jiří Kadlec	Evaluation package for Xilinx SDK 2015.4 with max clock and 3x (8xSIMD) EdkDSP. Pre-release to the Artemis Almarvi partners.

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

1. Summary

This application note describes demos of HW accelerated Full HD HDMI video processing and HW accelerated floating point filters computed in (8xSIMD) EdkDSP accelerators on the largest Zynq platform with the Kintex PL fabric supported by the free Xilinx Vivado 2015.4 and SDK 2015.4 tool chain:

- 3 edge detection video processing designs (sh01, sh02, sh03)
  - These demos document 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 of accelerated data paths and ARM Cortex A9 standalone C 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) in this case.

Each Full HD demo includes also the HW accelerated computation of two DSP filters. These single precision floating point filters are computed on one of the three (8xSIMD) EdkDSP run-time reprogrammable single precision floating point accelerators with these properties:

- C programs can be compiled for single MicroBlaze processor and for one of the three EdkDSP accelerators. Compiled C (and ASM) code can be executed by the accelerators, without the need to re-compile the design in Vivado 2015.4 [8].
- C programs for the MicroBlaze processor and for the three (8xSIMD) EdkDSP accelerators can be edited in the same SDK 2015.4 environment used for ARM Cortex A9 programming and debug.
- The three (8xSIMD) EdkDSP accelerators can run different programs in parallel and perform run-time change of tasks, task migration.
- Design is supporting the run-time re-programming of each of the (8xSIMD) EdkDSP acceleratorrs, under the control of the user-defined MicroBlaze C program.
- The MicroBlaze processor executes its program and utilizes data located in the top 256 MBytes of the 1Gbyte DDR3 memory. This region is also accessible by ARM processor. ARM initiates and controls the content of program and data executed by the MicroBlaze.
- ARM and MicroBlaze programs use HW mutex for synchronization.

1.1 Full HD Platform with HDMI I/O and three EdkDSP Accelerators

This application note describes HW platform performing integration of three runtime reprogrammable (8xSIMD) EdkDSP floating point accelerators. These accelerators work in parallel with an edge detection or motion detection video processing algorithms. Source of video data is an HDMI input with resolution 1920x1080p60 (Full HD). The platform is composed from these HW building blocks (boards):

- Zynq XC7Z030-3E device on System on Module TE0715-04-30-3E from Trenz-electronic [1].
- Carrier board TE0701-06 with FMC connector from Trenz-electronic [3], [4].
- AES-FMC-HDMI-CAM-G FMC HDMI I/O extension board from Avnet [5].
- PMODRS232: Serial converter & interface [6].

All implemented Full HD 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 Xilinx Vivado 2015.4 [8] to bitstreams for Zynq XC7Z030-3E device. Created SW access functions controlling the HW accelerators have been exported from the Xilinx SDSoC 2015.4 environment to the Xilinx SDK 2015.4 [8] SW projects as static libraries for the standalone ARM Cortex A9 processor C programs.

Figure 1: TE0701-06 carrier TE0715-04-30-3E running HDMII-HDMIO in HW and 3x EdkDSP accelerators.

Architecture

The Xilinx Zynq device XC7Z030-3E has two ARM Cortex A9 processors (operating at 1.0 GHz). Memory controller of Zynq device provides DDR3 memory access ports for ARM processors as well as to the reprogrammable logic. The Zynq device provides the programmable logic (PL) used for:

1. Three UTIA EdkDSP (8xSIMD) floating point processors (operating at 200 MHz) connected to Xilinx MicroBlaze 32bit processor (operating at 142.8 MHz).
2. Input chain of video processing Full HD data to input video frame buffers. The input video DMA (VDMA) controller is operating at 150 MHz.
3. Video processing HW accelerators and data movers defined in Xilinx SDSoC 2015.4 environment. These accelerators are controlled from the ARM Cortex A9 C programs compiled in SDK 2015.4 C projects. These accelerators are operating at 200 MHz.
4. Chain of output video processing IPs connects output frame buffers to the Full HD display by HDMI cable. The output VDMA controller is operating at 150 MHz.

- Three EdkDSP is 8xSIMD floating point accelerators are reprogrammable in runtime by change of firmware for the build-in PicoBlaze6 8bit controllers. These controllers are serving as schedulers of vector operations performed in the EdkDSP is 8xSIMD floating point data paths.
- These schedulers are programmed by simple C programs compiled by UTIA C compiler and assembler. These compilers respect the minimal resources of the PicoBlaze6 controllers.
- The three EdkDSP 8xSIMD floating point accelerators are controlled by single 32bit MicroBlaze processor. The MicroBlaze processor executes larger C programs from the DDR3 memory. Algorithms can benefit from execution of selected operations on three EdkDSP coprocessors. The EdkDSP coprocessors are connected to the MicroBlaze by local dual ported memories.
- MicroBlaze C program can take benefit of the potential overlap of data communication from DDR3 to the EdkDSP dual-ported memories (managed by the MicroBlaze processor) and the parallel computations performed in the three EdkDSP accelerators and controlled locally by the three PicoBlaze6 sequencers.
- All designs include also the video processing chain of Full HD I/O IPs controlled by the ARM processor via the Axi-lite control bus operating at 142.8 MHz.
- ARM Cortex A9 processor performs the global initialization and synchronisation of the video processing chain. The Arm program and the FPGA image is downloaded to the board from the Xilinx SDK 2015.4 via USB JTAG to the 1GB DDR3 located on the Zynq system on module.
- System can be also started directly from the SD card with help of the ARM FSBL loader. ARM processor performs the initialization of program for the MicroBlaze processor. This MicroBlaze program contains also the initial firmware for the three EdkDSP accelerators. The ARM processor also initiates the HDMI video input and video output IPs.

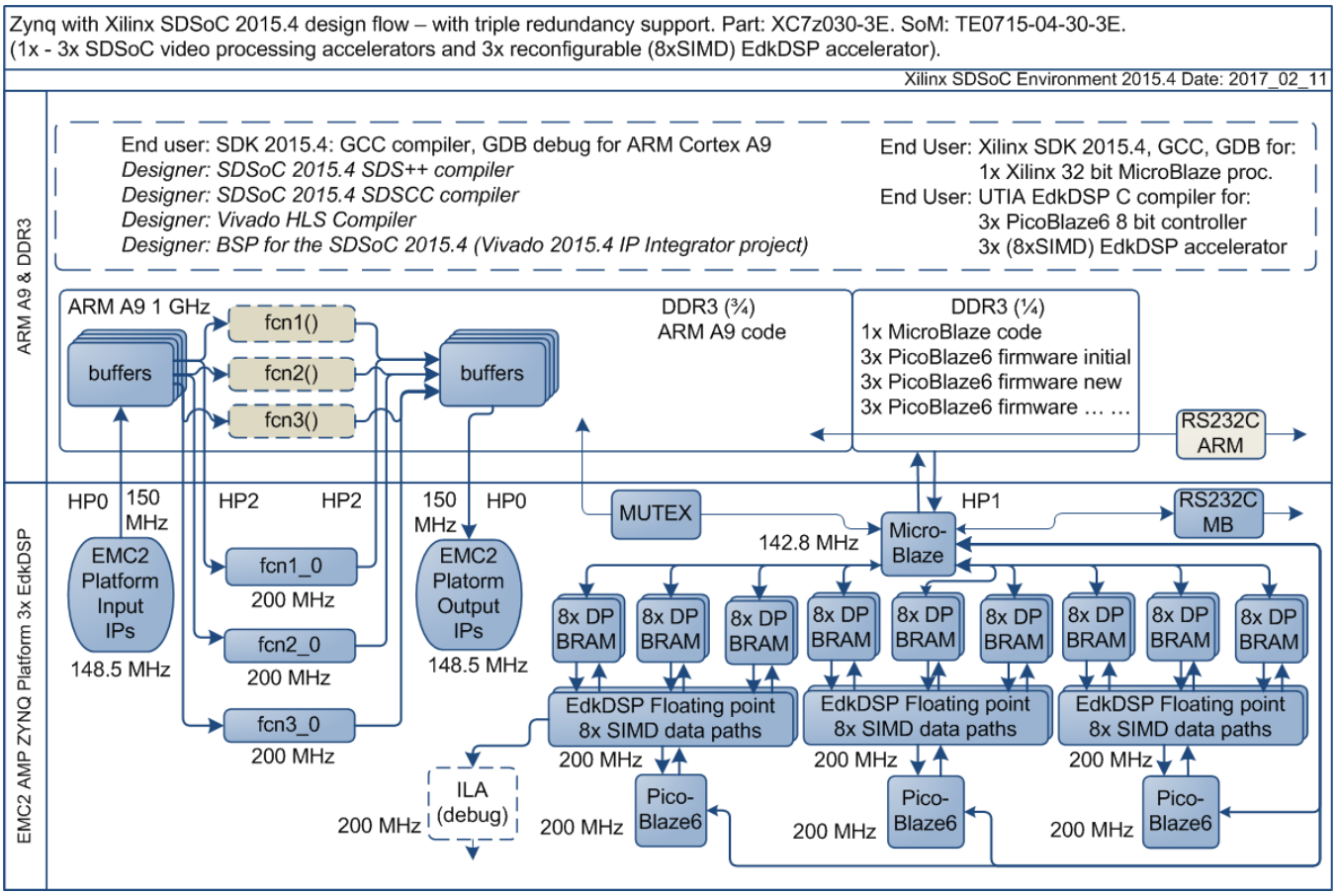


Figure 2: Full HD HDMII-HDMIO platform with three (8xSIMD) EdkDSP accelerators.

Configurations of video processing Accelertors and EdkDSP Accelerators:

Next sections describe used resources and the acceleration for these configurations:

- The MicroBlaze with the three (8xSIMD) EdkDSP are present in HW. The MicroBlaze computes floating point FIR filter on one of the three EdkDSP accelerators and the video accelerator chain computes selected video processing algorithm in HW.
- The MicroBlaze with the three (8xSIMD) EdkDSP are present in HW. The MicroBlaze computes floating point LMS adaptive filter on one of the three EdkDSP acceleratos and the video accelerator chain computes selected video processing algorithm in HW.
- The MicroBlaze with the three (8xSIMD) EdkDSP are present in HW. The MicroBlaze computes in SW (only with its internal HW floating point unit) the FIR or LMS filter in parallel to the dedicated video processing accelerator HW chain. None of the three EdkDSP accelerators is used.
- MicroBlaze is present in HW. It computes in SW (only with its internal HW floating point unit) the FIR or LMS filter in parallel to the dedicated video processing accelerator HW chain. The EdkDSP accelerators are not present in the PL logic.

Video processing algorithms have been implemented in SW by ARM Cortex A9 processor, first. The ARM C/C++ code was compiled with -O3 optimisation (but without use of the NEON accelerator) in the SDSoC 2015.4 environment [9]. The related HW resources include the MicroBlaze with the three (8xSIMD) EdkDSP present in the PL part of Zynq device and only the basic HDMI input and output HW support.

The evaluation designs with HW accelerators have been created from the selected C/C++ functions in SDSoC 2015.4 environment. New HW design have been generated and exported into the final set of SDK 2015.4 projects. The resulting demos are included in the evaluation package.

General setup of all demos:

- ARM Cortex A9 processor of Xilinx Zynq device XC7Z030-1I executes standalone C application programs performing initialisation and synchronisation of the HW accelerated video processing chains.
- Enclosed C programs for ARM, MicroBlaze and PicoBlaze6 sequencers can be modified by the user and recompiled in single Xilinx SDK 2015.4 development framework.
- Video signal input with resolution 1920x1080p60.
- 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 data to the Full HD display with HDMI.

1.2 Demos

Main objectives of demos:

- To demonstrate how to install, compile, modify and use the enclosed SW projects in the SDK 2015.4.
- To demonstrate the HW accelerated video processing algorithms and the acceleration in comparison to the original ARM Cortex A9 SW versions of video processing algorithms.
- To demonstrate parallel execution of predefined video processing HW paths with C user code on ARM.
- To demonstrate HW accelerated video processing algorithms and the accelerated floating point FIR/LMS filters computed by the 8xSIMD EdkDSP run-time re-programmable floating point accelerator.
- To evaluate power consumption of several system configurations.

Edge detection

The edge detection algorithm detects 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 computation is 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 accelerated HW 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 are using 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.

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. This difference signal is finally filtered by the 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 is using 2 DMA HW channels for reading from two sub sequent video frame buffers located both in the DDR3 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.

1.3 Measurements of acceleration and resources used in included projects

The acceleration results have been measured as a ratio of the frame 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. HW support for the HDMI I/O data movement by the dedicated VDMA HW channels was used in all cases.

1.4 Project sh01: Edge detection with single HW accelerator and 3x EdkDSP

The accelerated data path runs at 200 MHz and includes HW version of the function:

- sobel_filter_htile1

with one data-mover controlling one input and one output AXI DMA channel to the DDR3.

Acceleration by HW: 5.23 x

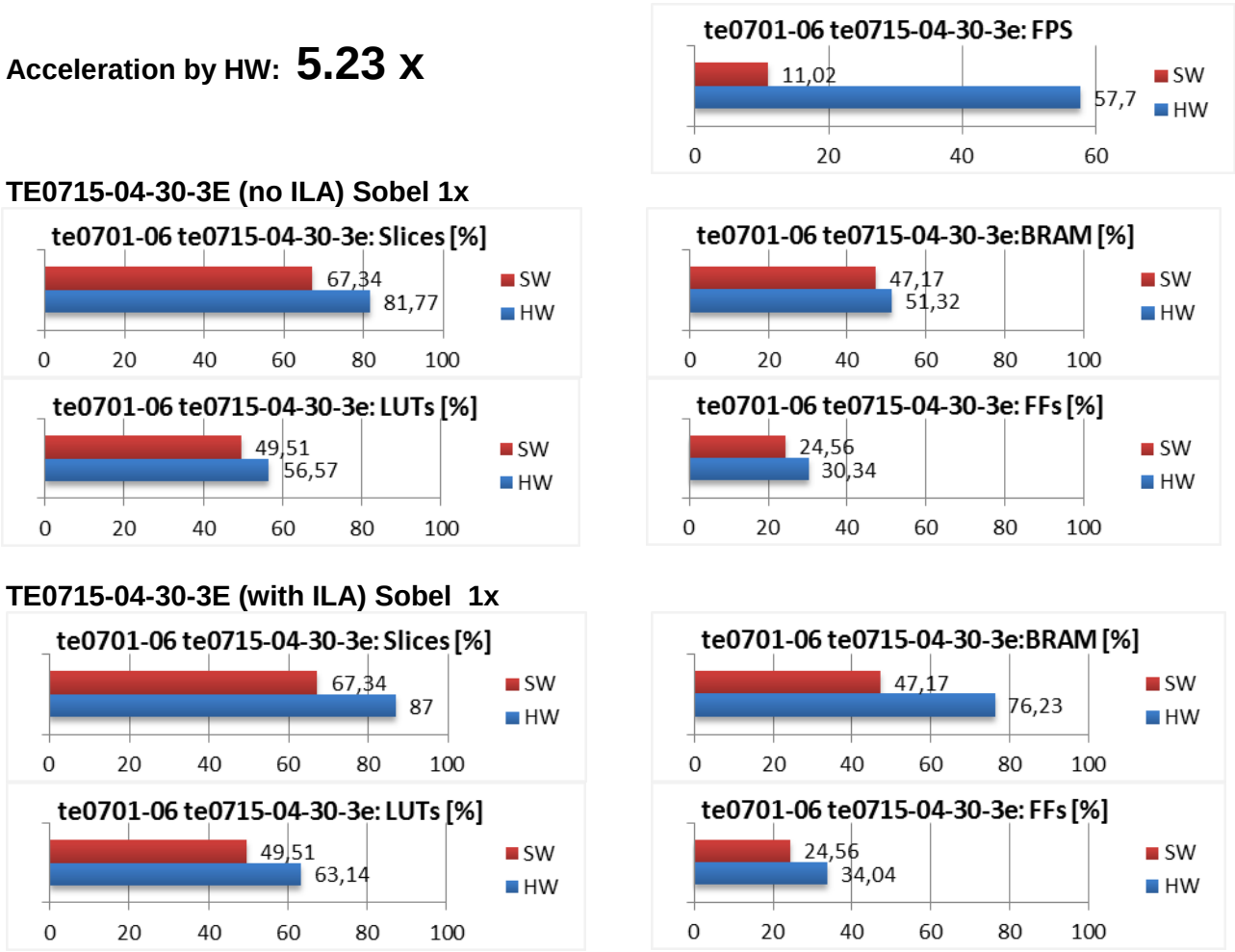


Figure 3: Project sh01 - Acceleration and HW resources used.

Power consumption

Power consumption of complete working system is measured on 12V power supply rail. Figure 4 and Figure 5 display power consumption for:

- ARM A9 + video I/O and HW accelerators + MicroBlaze
- ARM A9 + video I/O and HW accelerators + MicroBlaze + 3x EdkDSP HW instantiated
- ARM A9 + video I/O and HW accelerators + MicroBlaze + 3x EdkDSP HW filters are present. One instantiated EdkDSP accelerators is computing the LMS or FIR filter in floating point:
  - One EdkDSP HW accelerator computing LMS Filter: 474 mW/GFLOP/s
  - One EdkDSP HW accelerator computing FIR Filter: 327 mW/GFLOP/s
  - One EdkDSP HW accelerator computing LMS Filter with ILA: 515 mW/GFLOP/s
  - One EdkDSP HW accelerator computing FIR Filter with ILA: 354 mW/GFLOP/s

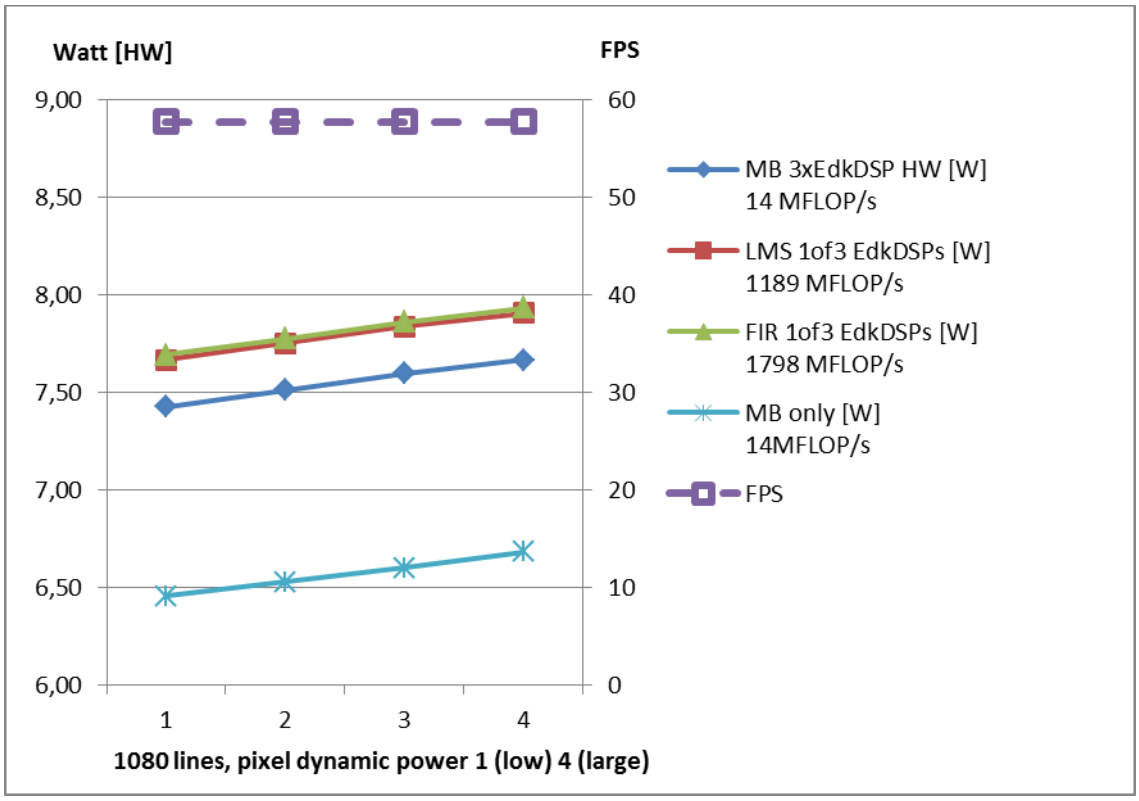


Figure 4: Power consumption of sh01 demo without ILA

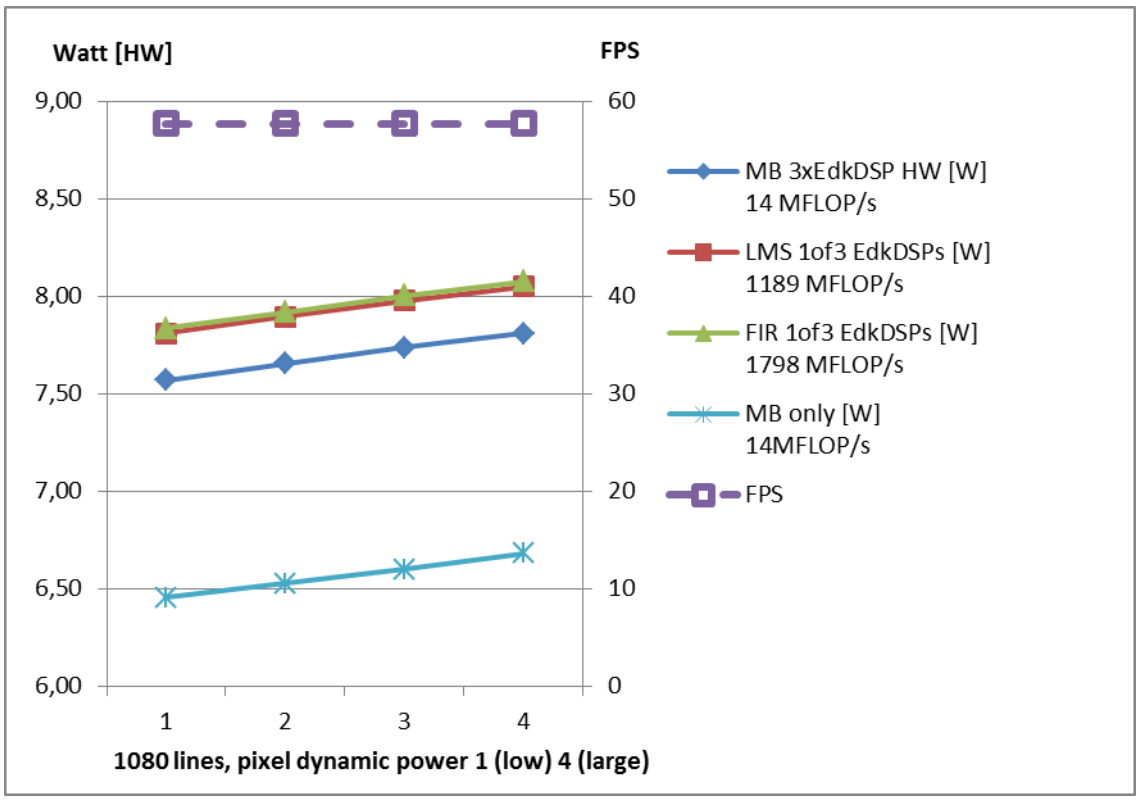


Figure 5: Power consumption of sh01 demo with ILA

1.5 Project sh02: Edge detection with two HW accelerators and 3x EdkDSP

The two parallel accelerated data paths run at 200 MHz and include HW version of these functions:

- sobel_filter_htile1
- sobel_filter_htile2

with two data-movers controlling two input and two output AXI DMA channels to the DDR3.

Acceleration by HW: 5.56 x

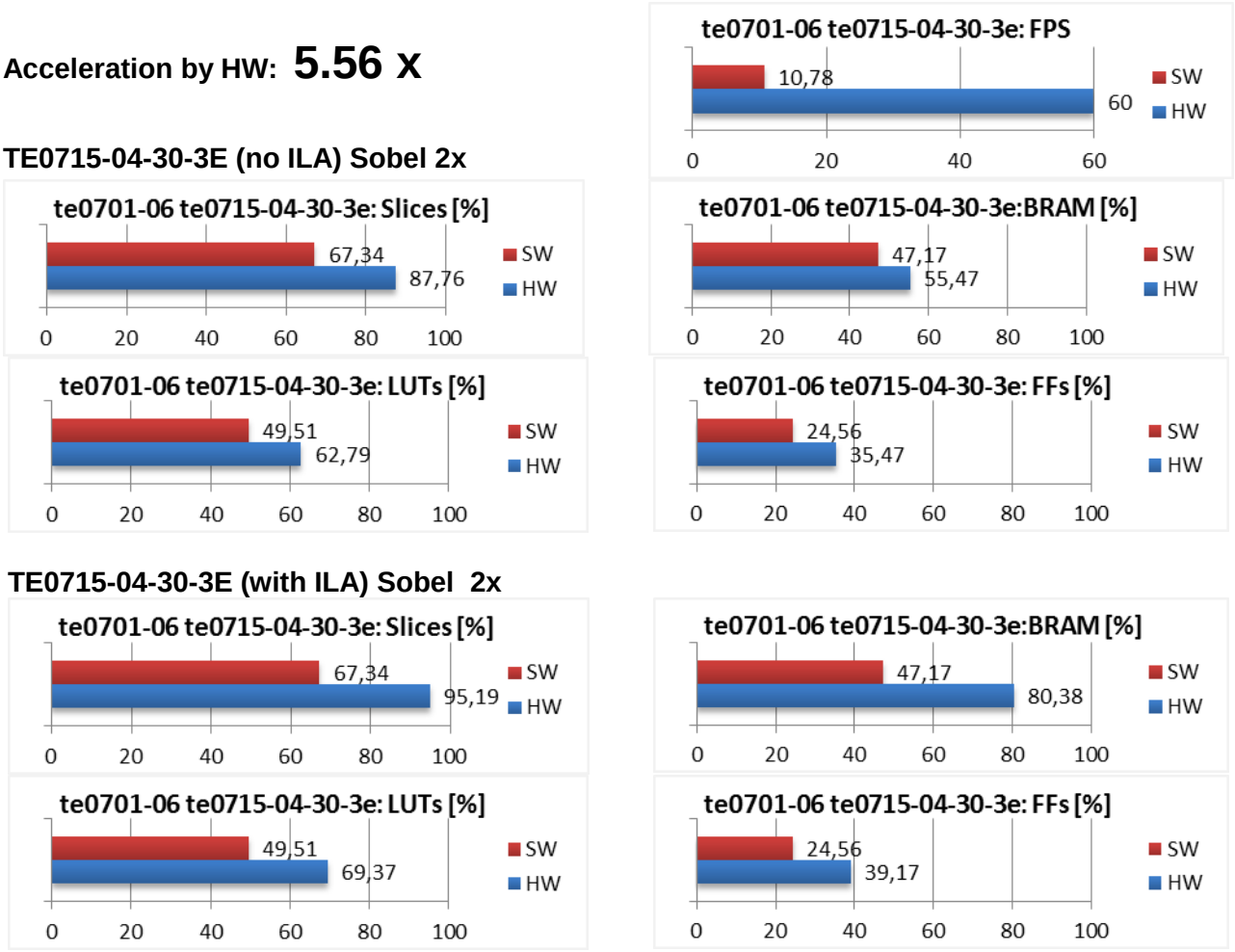


Figure 6: Project sh02 – Acceleration and HW resources used.

Power consumption

Figure 7 and Figure 8 display power consumption for:

- ARM A9 + video I/O and HW accelerators + MicroBlaze
- ARM A9 + video I/O and HW accelerators + MicroBlaze + 3x EdkDSP HW instantiated
- ARM A9 + video I/O and HW accelerators + MicroBlaze + 3x EdkDSP HW filters are present. One instantiated EdkDSP accelerators is computing the LMS or FIR filter in floating point:
  - One EdkDSP HW accelerator computing LMS Filter: 474 mW/GFLOP/s
  - One EdkDSP HW accelerator computing FIR Filter: 327 mW/GFLOP/s
  - One EdkDSP HW accelerator computing LMS Filter with ILA: 521 mW/GFLOP/s
  - One EdkDSP HW accelerator computing FIR Filter with ILA: 358 mW/GFLOP/s

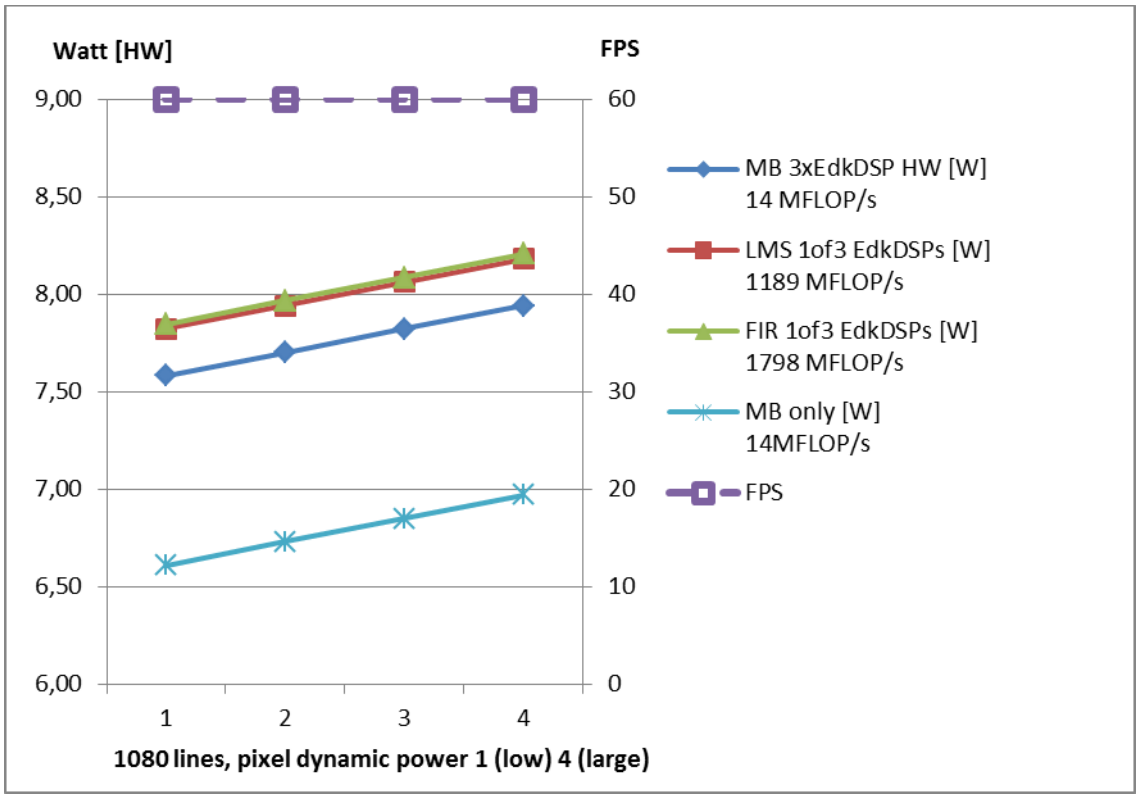


Figure 7: Power consumption of sh02 demo without ILA

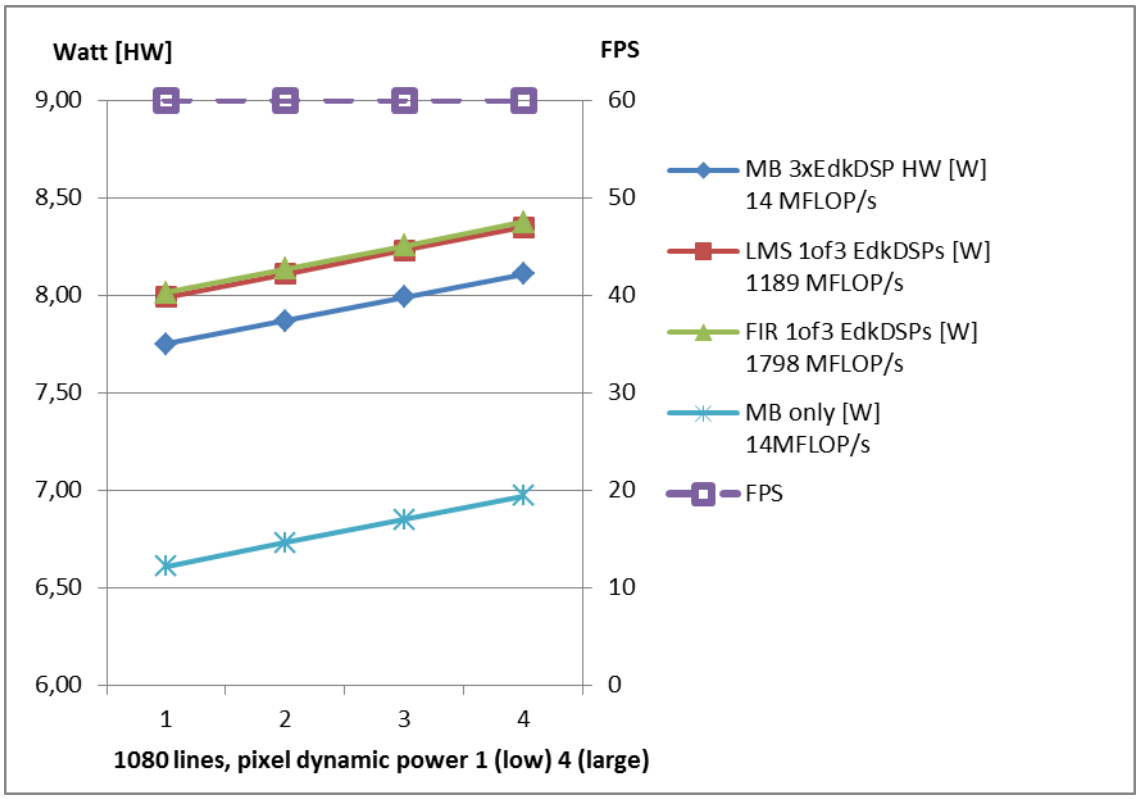


Figure 8: Power consumption of sh02 demo with ILA

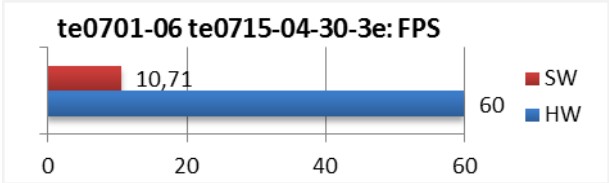
1.6 Project sh03: Edge detection with three HW accelerators and 3x EdkDSP

The three parallel accelerated data paths run at 200 MHz and include HW version of these functions:

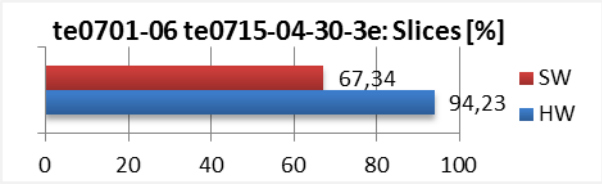
- sobel_filter_htile1
- sobel_filter_htile2
- sobel_filter_htile3

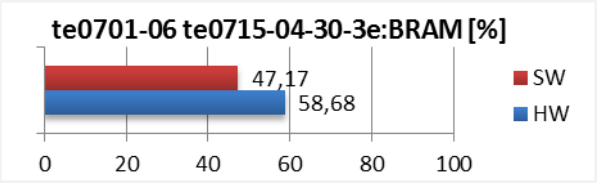
with three data-movers controlling three input and three output AXI DMA channels to the DDR3.

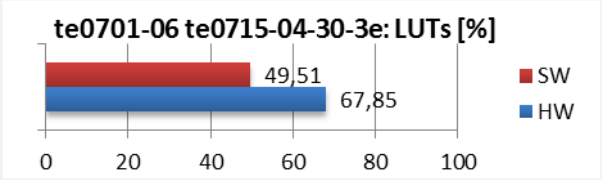
Acceleration by HW: 5.60 x

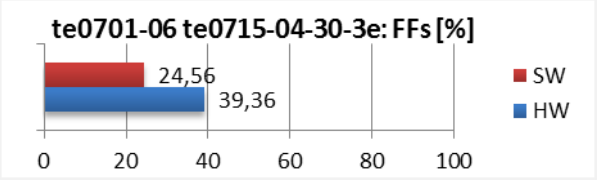


TE0715-04-30-3E (no ILA) Sobel 3x

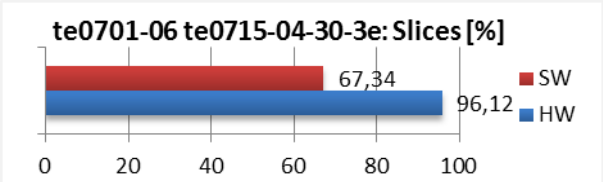


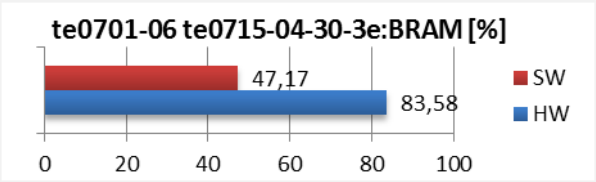


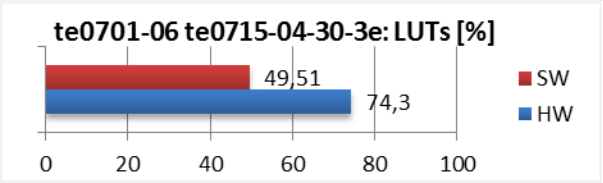




TE0715-04-30-3E (with ILA) Sobel 3x







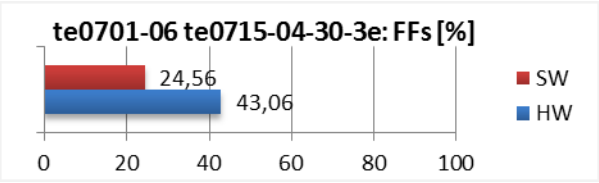


Figure 9: Project sh03 - Acceleration and HW resources used.

Power consumption

Figure 10 and Figure 11 display power consumption for:

- ARM A9 + video I/O and HW accelerators + MicroBlaze
- ARM A9 + video I/O and HW accelerators + MicroBlaze + 3x EdkDSP HW instantiated
- ARM A9 + video I/O and HW accelerators + MicroBlaze + 3x EdkDSP HW filters are present. One instantiated EdkDSP accelerators is computing the LMS or FIR filter in floating point:
  - One EdkDSP HW accelerator computing LMS Filter: 491 mW/GFLOP/s
  - One EdkDSP HW accelerator computing FIR Filter: 338 mW/GFLOP/s
  - One EdkDSP HW accelerator computing LMS Filter with ILA: 535 mW/GFLOP/s
  - One EdkDSP HW accelerator computing FIR Filter with ILA: 367 mW/GFLOP/s

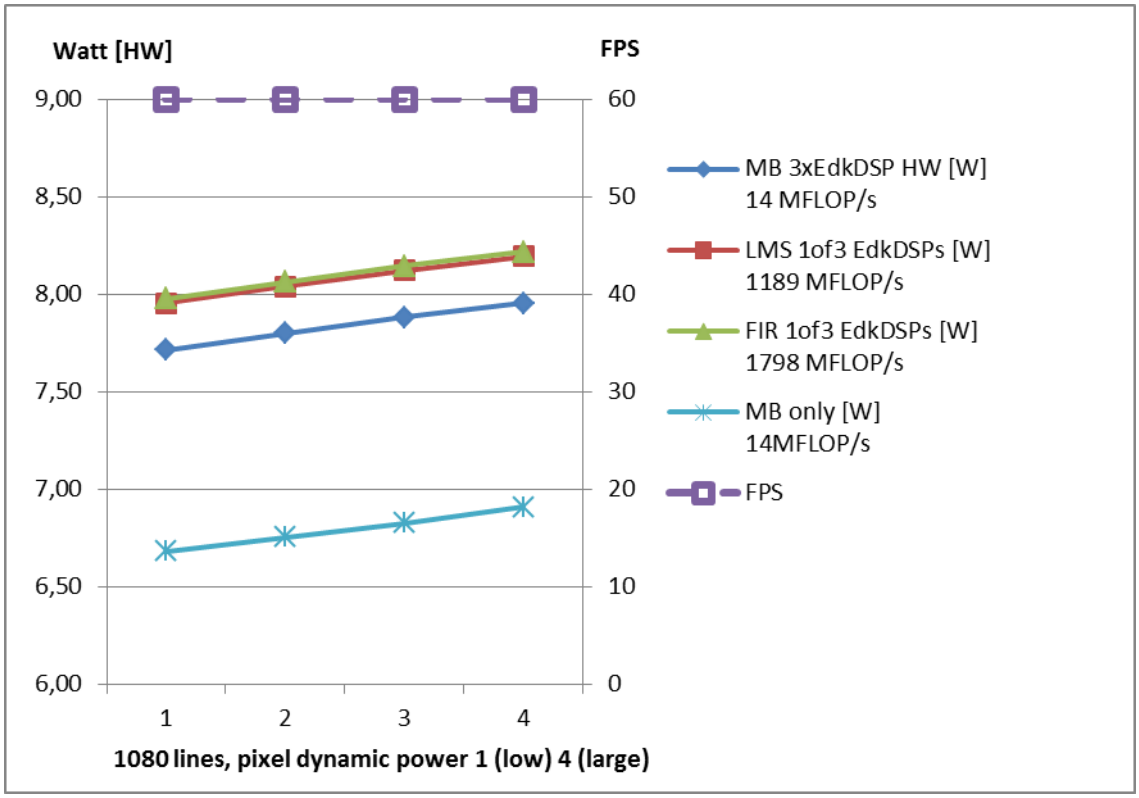


Figure 10: Power consumption of sh03 demo without ILA

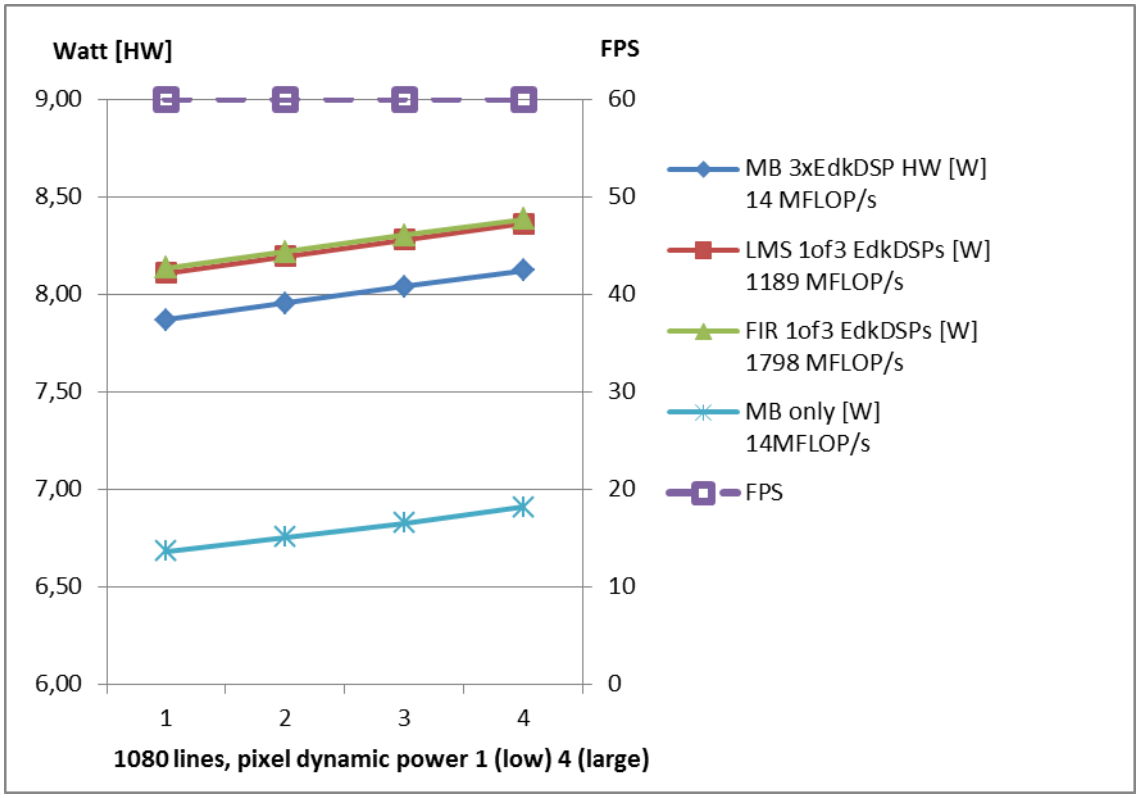


Figure 11: Power consumption of sh03 demo with ILA

1.7 Project md01: Motion detection with chain of HW accelerators and 3x EdkDSP

Motion detection is implemented as single accelerated data path run at 200 MHz. It processes data from two subsequent video frames and it includes chain of HW versions of these functions:

(1) pad (two instances) (2) sobel_filter_pass sobel_filter (3) diff_image
(4) median_char_filter_pass (5) combo_image (6) ext

four data-movers controlling the 200 MHz input AXI DMA channels from the DDR3 and one data-mover controlling the 200 MHz output AXI DMA channel to the DDR3.

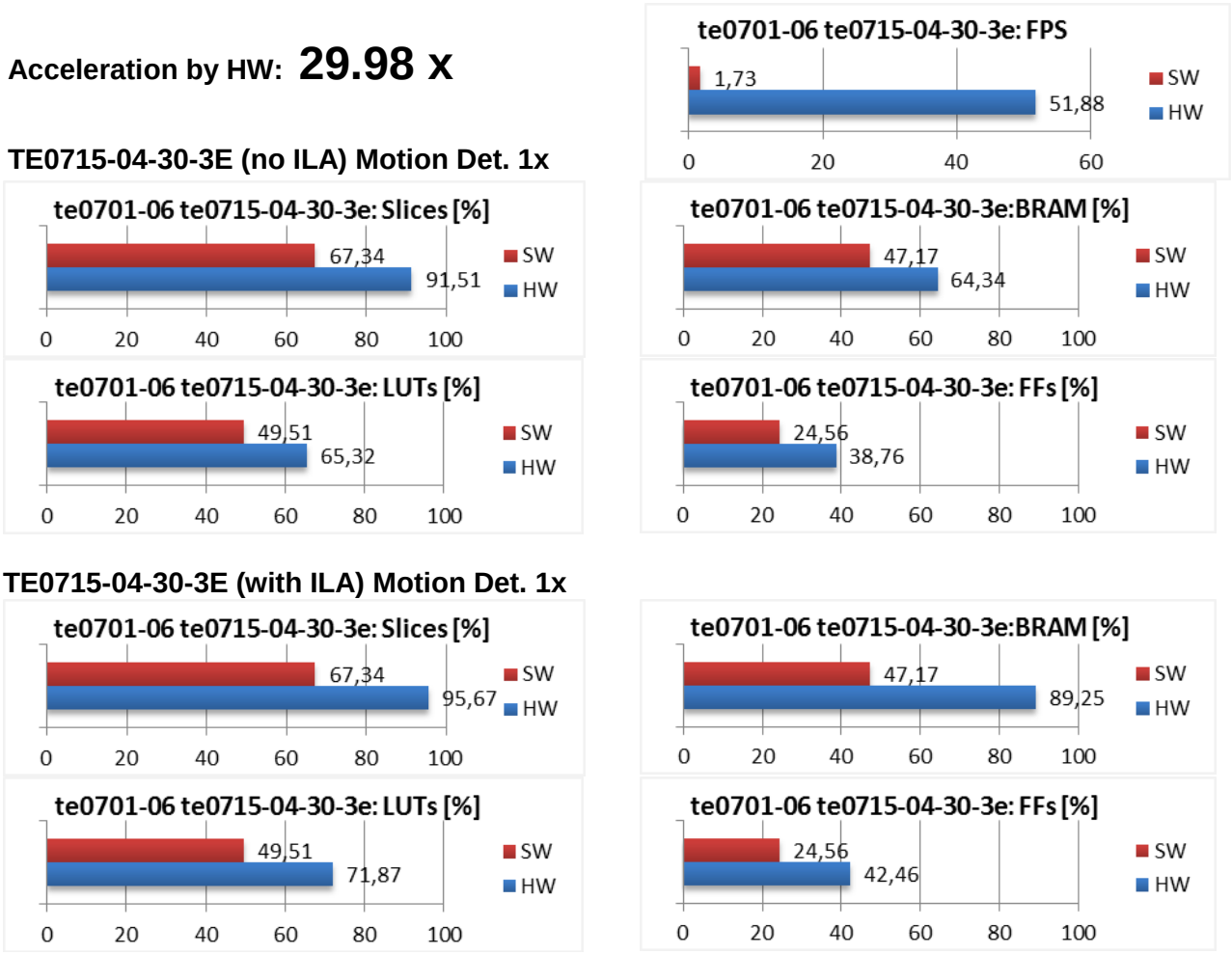


Figure 12: Project md01 - Acceleration and HW resources used

Power consumption

Figure 13 and Figure 14 display power consumption for:

- ARM A9 + video I/O and HW accelerators + MicroBlaze
- ARM A9 + video I/O and HW accelerators + MicroBlaze + 3x EdkDSP HW instantiated
- ARM A9 + video I/O and HW accelerators + MicroBlaze + 3x EdkDSP HW filters are present. One instantiated EdkDSP accelerators is computing the LMS or FIR filter in floating point:
    - One EdkDSP HW accelerator computing LMS Filter: 437 mW/GFLOP/s
    - One EdkDSP HW accelerator computing FIR Filter: 303 mW/GFLOP/s
    - One EdkDSP HW accelerator computing LMS Filter with ILA: 495 mW/GFLOP/s
    - One EdkDSP HW accelerator computing FIR Filter with ILA: 340 mW/GFLOP/s

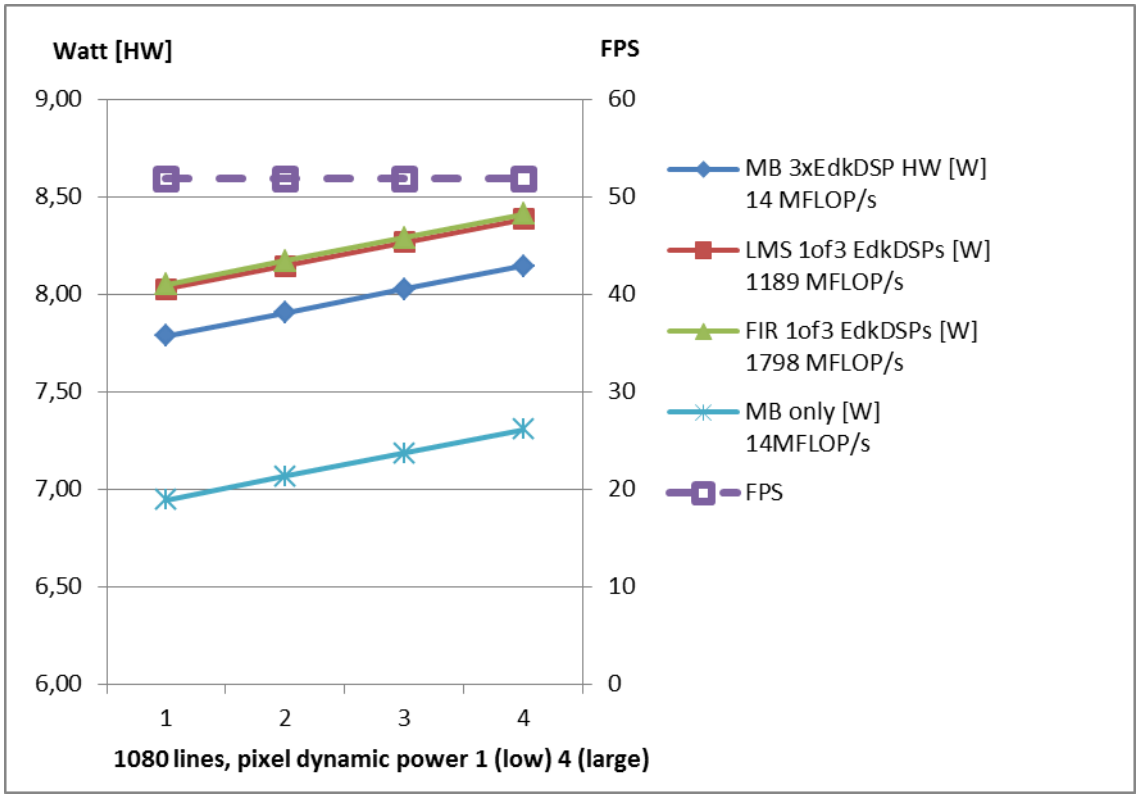


Figure 13: Power consumption of md01 demo without ILA

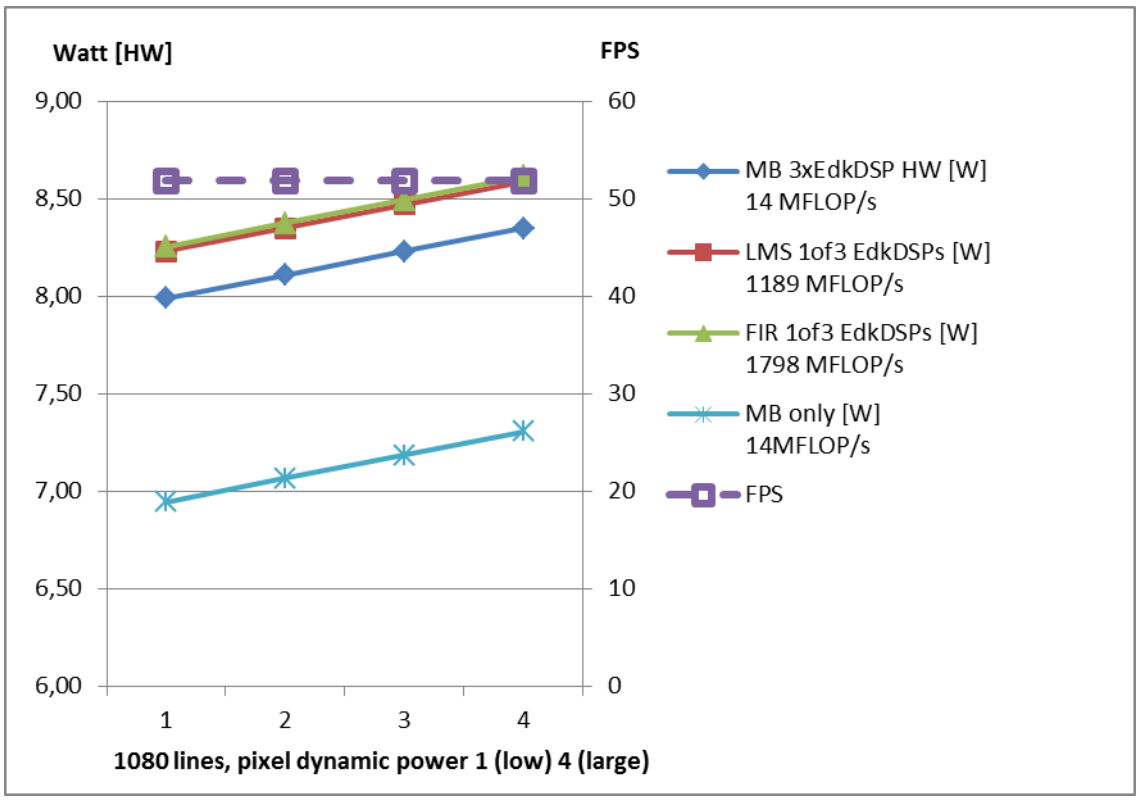


Figure 14: Power consumption of md01 demo with ILA

1.8 Floating point performance

This chapter summarises measured sustained single precision floating point performance of the system in parallel to the accelerated video processing:

1798 MFLOP/s on the 200 MHz (8xSIMD) EdkDSP (FIR filter in floating point on single 8xSIMD accelerator).
144 MFLOP/s on the 142.8 MHz MicroBlaze processor (with the MB single precision floating point unit in HW).

The controllers inside each (8xSIMD) EdkDSP accelerators are reprogrammed by the firmware compiled from C code with the use of the UTIA EDKDSP C compiler. Each accelerator can be programmed with two firmware programs. Designs can swap firmware in the runtime in only few clock cycles. The alternative firmware can be downloaded to the (8xSIMD) EdkDSP accelerator controllers in parallel with the execution of the current firmware.

This is demonstrated by swap of the firmware for the FIR filter (room response) to the firmware for adaptive LMS identification of the filter coefficients in the acoustic noise cancellation demo. This also demonstrates the mechanism and support for the move from one task to another task on the same accelerator.

Each of the three (8xSIMD) EdkDSP accelerators can deliver single-precision floating point results, which are bit-exact identical to the reference software implementation running on the MicroBlaze with the Xilinx HW single precision floating point unit.

1.9 Summary

The 28nm Kintex-based programmable logic part of the Zynq XC7Z030-1I device is capable of implementation in **three** UTIA (8xSIMD) EdkDSP floating point accelerators together with the Full HD video processing chain for the real-time video processing.

The combination of single 32bit MicroBlaze with three instances of the (8xSIMD) EdkDSP single precision floating point accelerators brings additional capability to compute floating point operations (single precision) with the performance **1798 MFLOP/s** (in case of FIR filter) on single (8xSIMD) EdkDSP accelerator at the expense of relatively moderate increase of total power consumption of the system.

Instantiation of the 142.8 MHz MicroBlaze processor with three instances of the 200 MHz (8xSIMD) EdkDSP accelerators enables to work with the triple redundancy and, in parallel, execute HW accelerated video processing algorithms. The optional in-circuit logic analyser (ILA) is capable of triggering and visualizing up to 32k of data samples at 150MHz clock rate for the first of the three (8xSIMD) EdkDSP accelerators. This is very useful for debugging of sequences vector operations and addresses generated by the sequencer of the EdkDSP accelerator.

Designs debugged and developed in the high level SDSoC 2015.4 environment [9] are exported for the end-user in form of SDK 2015.4 [8] projects. The released evaluation package with SDK 2015.4 projects provide sufficient freedom for the end-user to make certain SW adaptations and customisations of the final application without the need to understand app low level details of accelerator IP cores, of the Vivado 2015.4 project. The initial SDSoC 2015.4 board support package is not needed in the released precompiled SDK 2015.4 projects. The SDSoC 2015.4 license is also not needed to run and modify them in the SDK 2015.4 SW projects.

1.10 HW setup

HW setup usies commercially accessible components [1]-[6].

TE0715-04-30-3E	Part XC7Z030-3SBG485E; 1GByte DDR; Industrial Grade [1].
Heatsink for TE0715	Spring-loaded embedded [2].
TE0701-06 Carrier Board	Trenz Electronic 7 Series [3], [4].
AES-FMC-HDMI-CAM-G	FMC card with HDMI I/O and CAM interface [5].
PMODRS232	Serial converter & interface [6] serves as the RS232C ASCII terminal for the MicroBlaze.

See the TE0701-06 technical reference manual [4] for the description of the TE0701-06 carrier board [3].

Set the TE0701-06 carrier board switches and jumpers as follows:

- Set switch S3 as described in Figure 15
  (1=OFF; 2=OFF; 3=ON; 4=OFF)

- Set jumpers to generate VADJ=1.8V as described in Figure 16
  (J17: open; J16: connect 1-2; J21: connect 2-3)

- Set switch S3 as described in Figure 17
  (set VADJ=1.8V) by the selection: 1=ON; 2=OFF; 3=ON; 4=OFF;

After this setup, the TE0715-04-30-3E Zynq device PL IO-bank supply-voltages are set to the 1.8V. Higher voltage for the I/O banks is not possible for this device. It would cause a damage of the device.

It is highly recommended to set app switches of the TE0701-06 carrier board and measure the PL IO-bank supply-voltage before mounting of the module on TE0701-06. It helps to avoid failures and damages to the functionality of the mounted module if the voltage for the HPF banks would be higher than the 1.8V.

See locations and switch positions in Figure 15, Figure 16 and Figure 17 and read the TE0701-06 TRM [4].

Figure 15: TE701-06 setting of switch S3: 1=OFF 2=OFF 3=ON 4=OFF

Figure 16: TE701-06: (use VADJ=1.8V) J17: open; J16: connect 1-2; J21: connect 2-3

Figure 17: TE701-06 setting of switch S4 (VADJ=1.8V): 1=ON 2=OFF 3=ON 4=OFF

The MicroBlaze uses the RS232 terminal for its output. It uses the J5 PMOD connector and the PMODRS232 module [9]. The TE0701-06 carrier board propagates the selected VADJ = 1.8V to the **J5** connector. This is not compatible with the 3.0V-5.0V power supply requirement of the PMODRS232 module. This problem can be fixed by disconnecting the 1.8V wire and replacing it with the 3.3V for the PMODRS232 module. This can be done by a custom made cable disconnecting the 1.8V power supply and providing instead the 3.3V power supply, which can be taken from the PMOD **J1** connector pin 12. See Figure 18, Figure 19 and Figure 20.

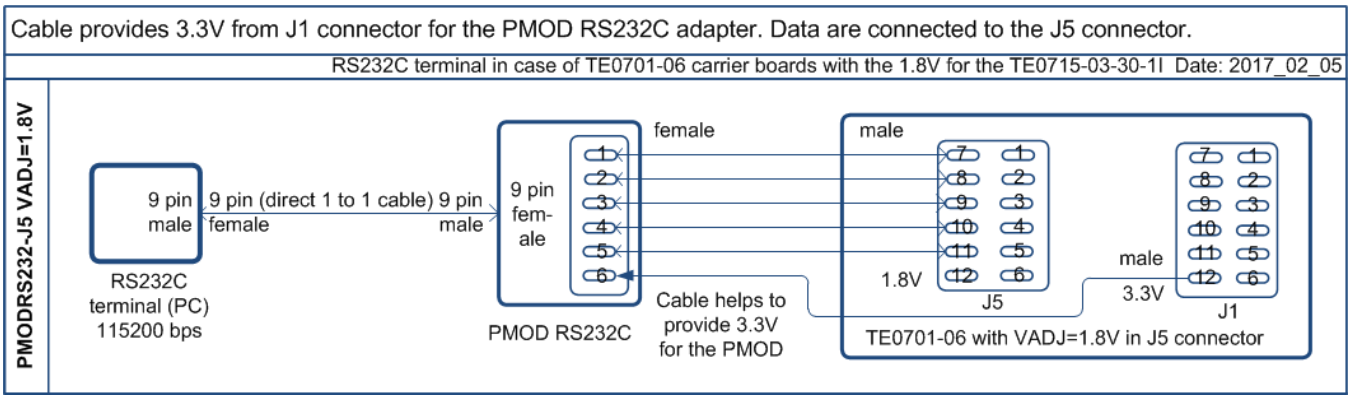


Figure 18: PMODRS232 cable connection to the TE701-06 set to with VADJ=1.8V

All provided designs will work also without the RS232 terminal because it is used only for a non-blocking text output with the communication speed set to 115200 bps. The RS232 terminal helps with debug and visualization of progress of the MicroBlaze application. If the terminal is not present, consider the observation of the MicroBlaze application by stepping through the program in the SDK 2015.4 debugger. See Figure 19 and Figure 20.

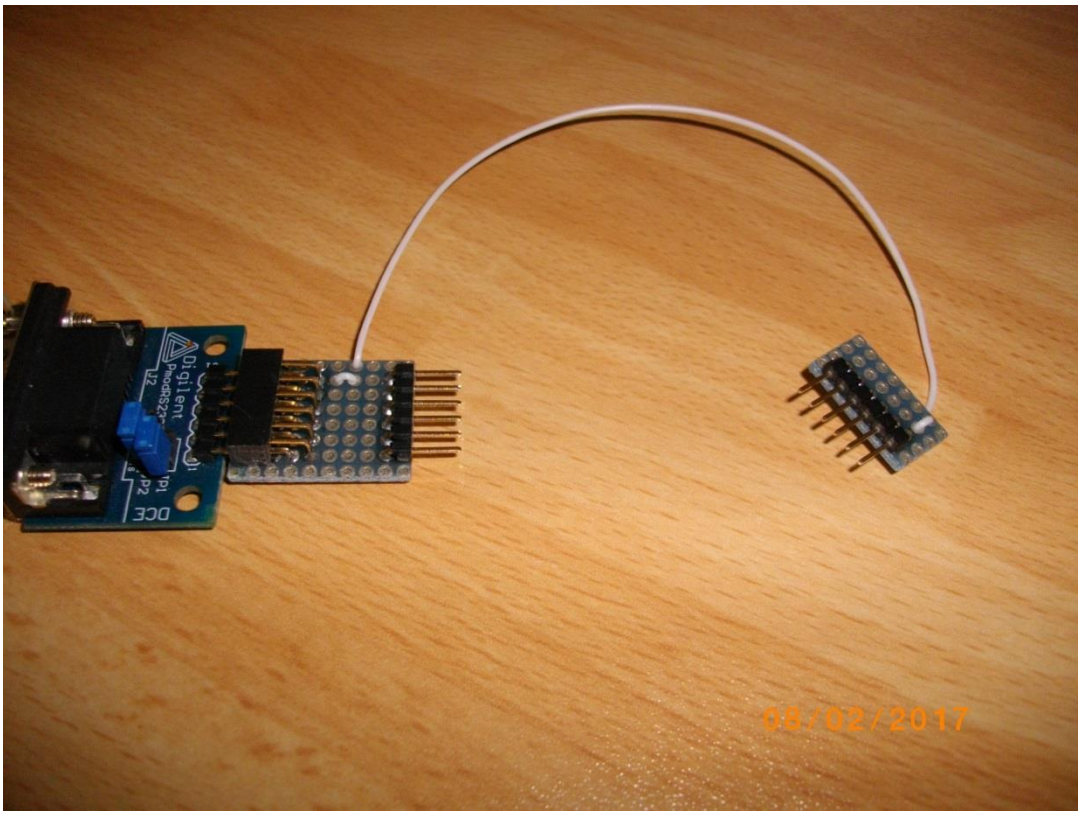


Figure 19: TE701-06: external 3.3V for the PMODRS232 from J1

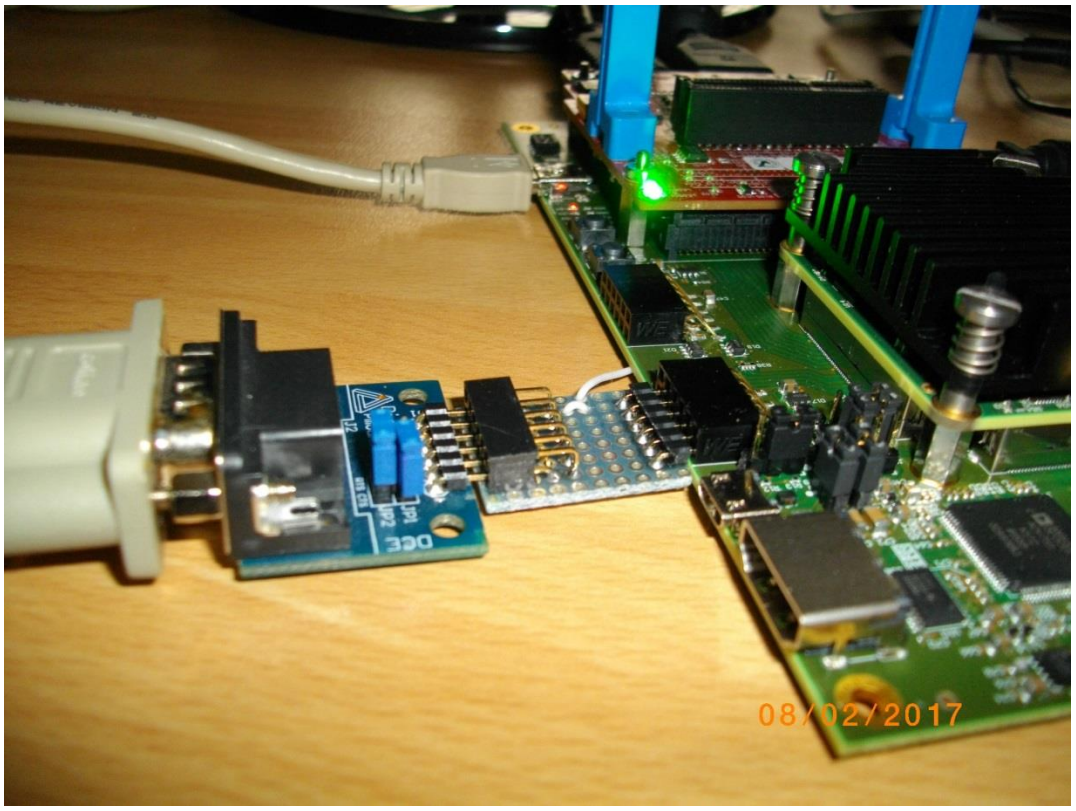


Figure 20: TE701-06 PMODRS232 connection

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\t30e3hm4_V54_IMPORT

Create empty directory for Xilinx SDK workspace.
C:\VM_07\t30e3hm4

Start Xilinx SDK 2015.4 and select the directory for the SDK 2015.4 workspace. See Figure 21. Select **C:\VM_07\t30e3hm4**

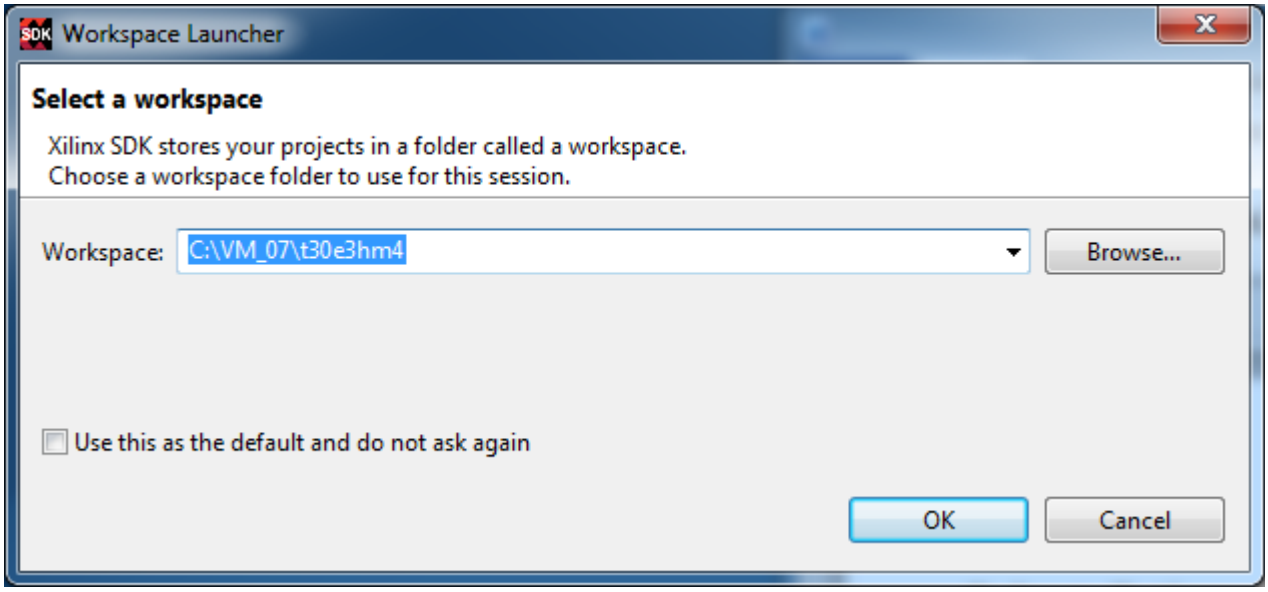


Figure 21: Select the SDK Workspace

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

File -> Import -> General -> Existing Projects into Workspace
Click on Next button. See Figure 22.

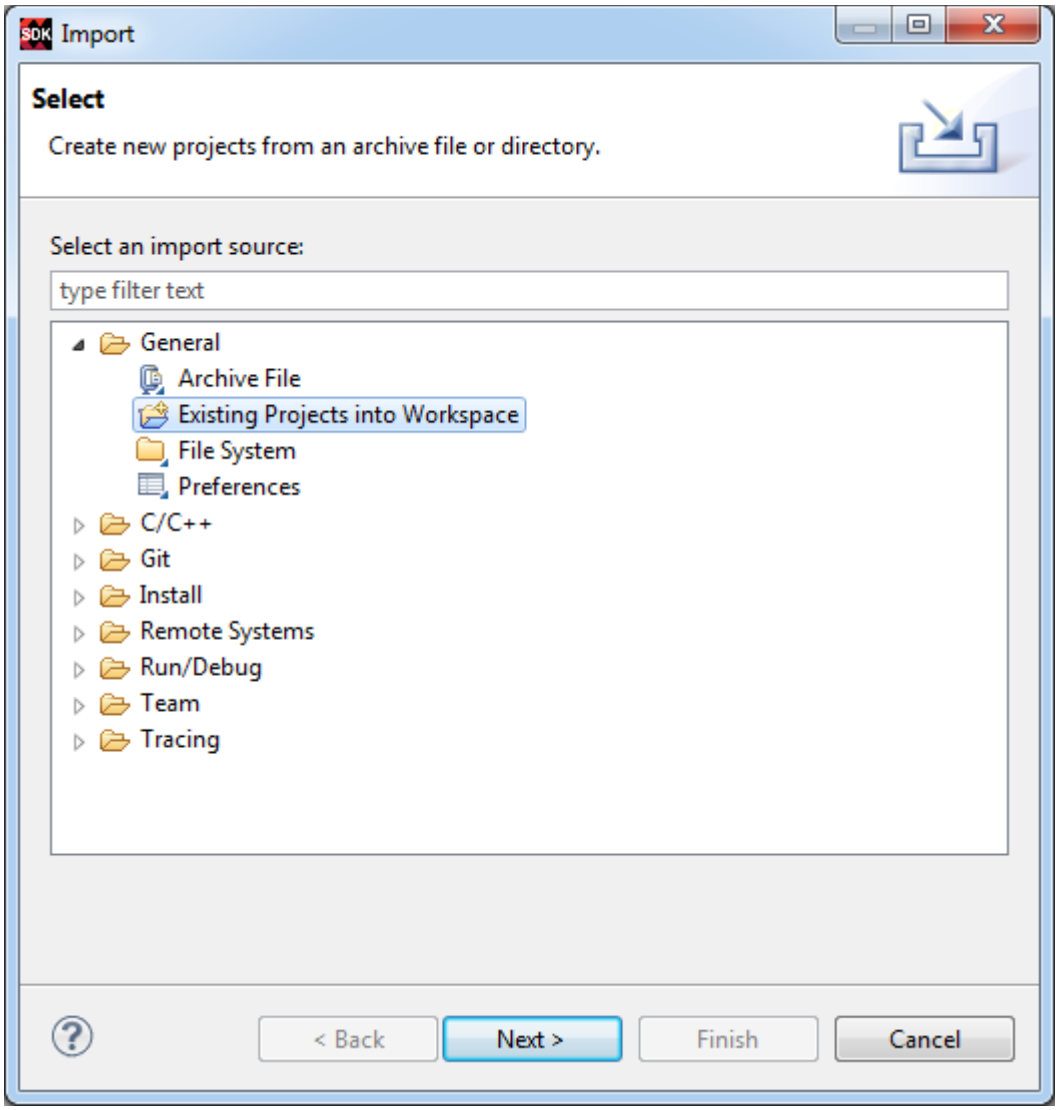


Figure 22: Import Existing Projects into Workspace

Type the directory with projects to be imported. See Figure 23.

C:\VM_07\t30e3hm4_V54_IMPORT

Set the "**Copy projects into workspace**" check box.
Click on Finish button. See Figure 23.

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.

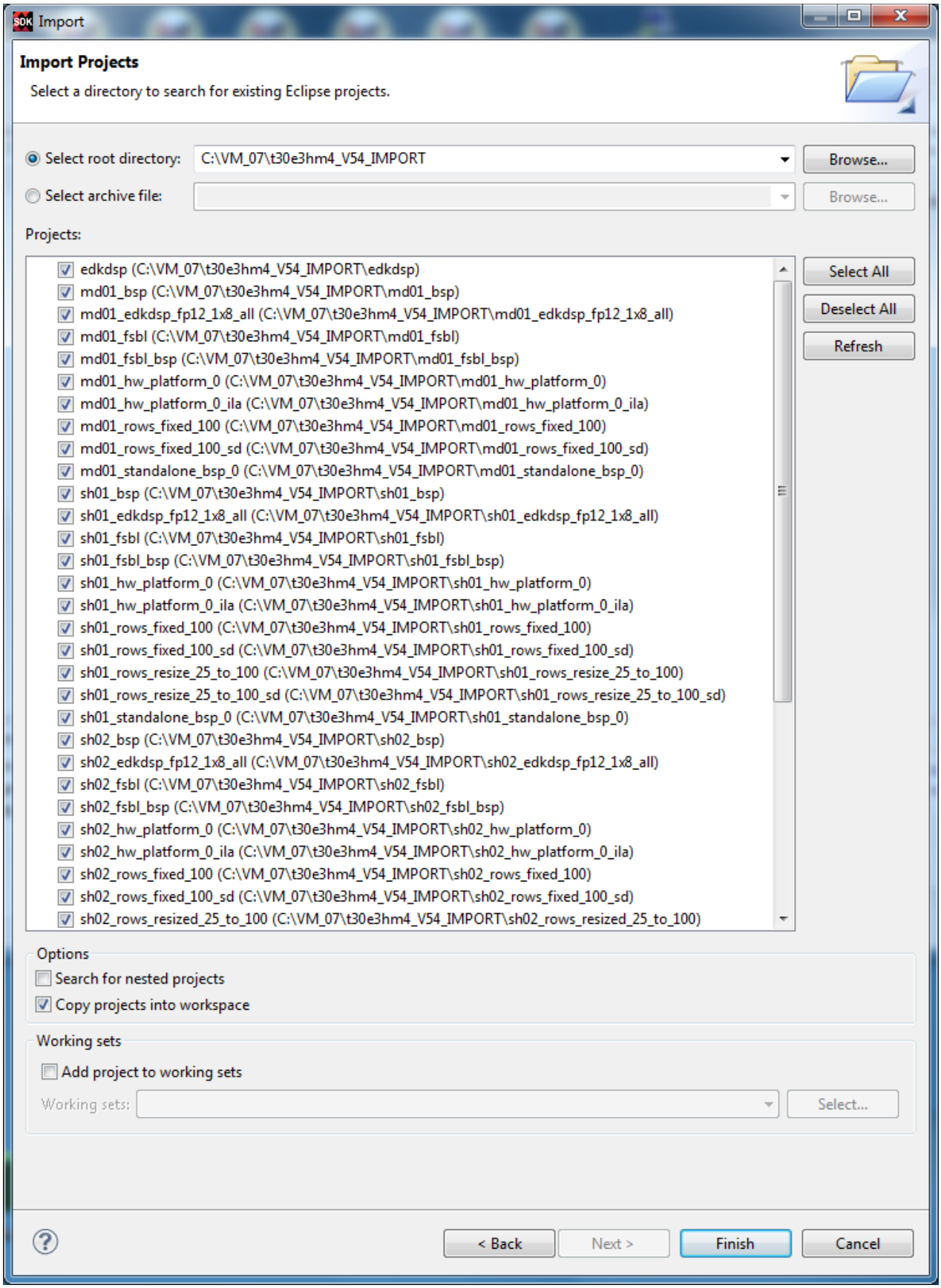


Figure 23: Select "Copy projects into workspace" and finish the import of all projects.

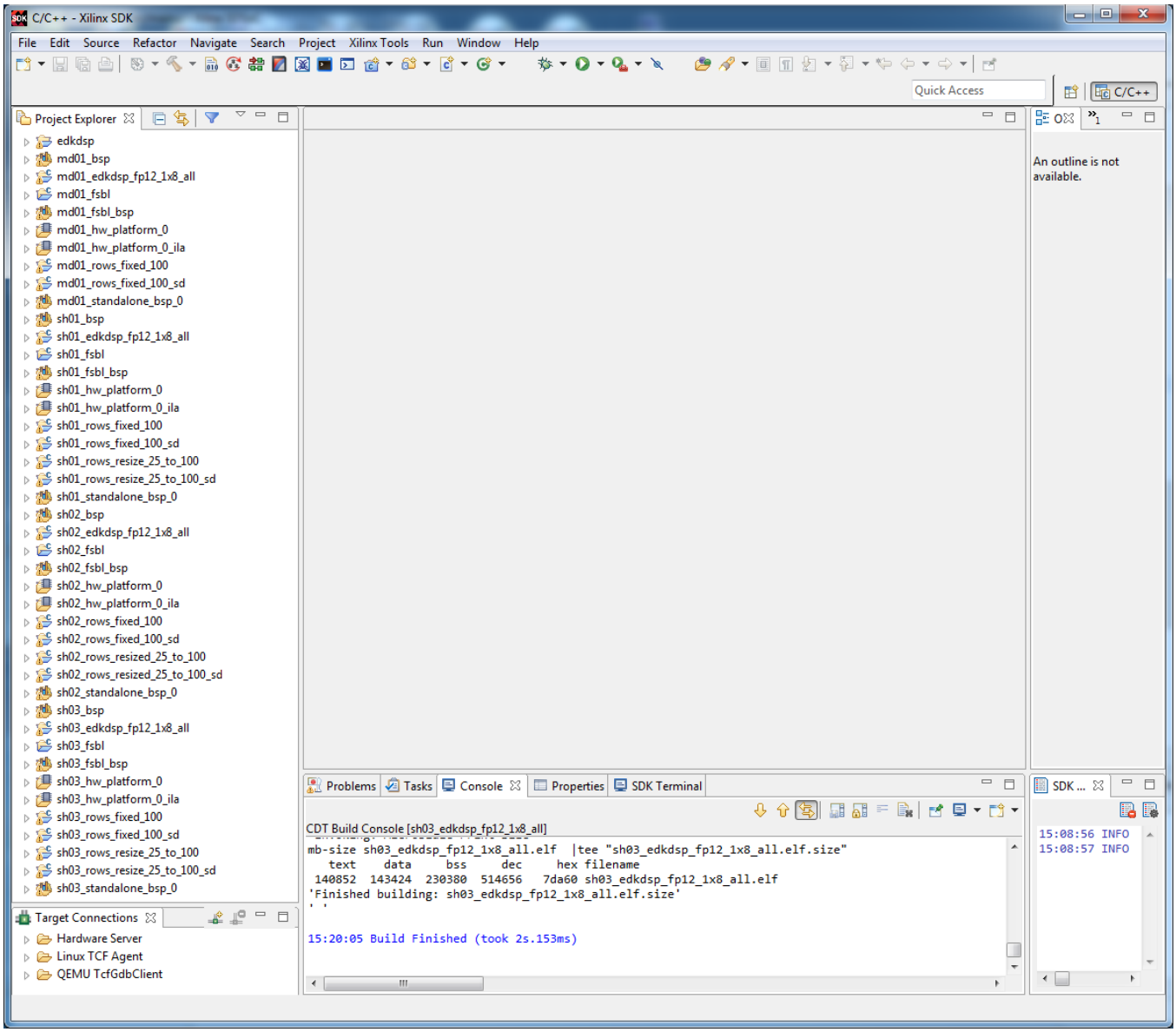


Figure 24: All projects are compiled in debug mode.

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

2.2 Test demos

To test demos follow these steps:

- Connect HDMI (or DVI) source by HDMI cable to the HDMI IN connector of the AES-FMC-HDMI-CAM-G.
- Connect HDMI (or DVI) monitor by HDMI cable to the HDMI OUT on the AES-FMC-HDMI-CAM-G board.
- 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 25. 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).

(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. (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. It is writing messages to the serial terminal. On request, “Hit any key to stop autoboot” type 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 the lost connection of the PC terminal.

Figure 25: USB for ARM terminal and JTAG. RS232C Pmod for MicroBlaze

Download bitstream to the board. Demo **sh03_rows_resize_25_to_100** will be used as an example.
The **bitstream.bit** for demo **sh03** is located in the directory: **C:\VM_07\t30e3hm4\sh03_hw_platform_0**

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

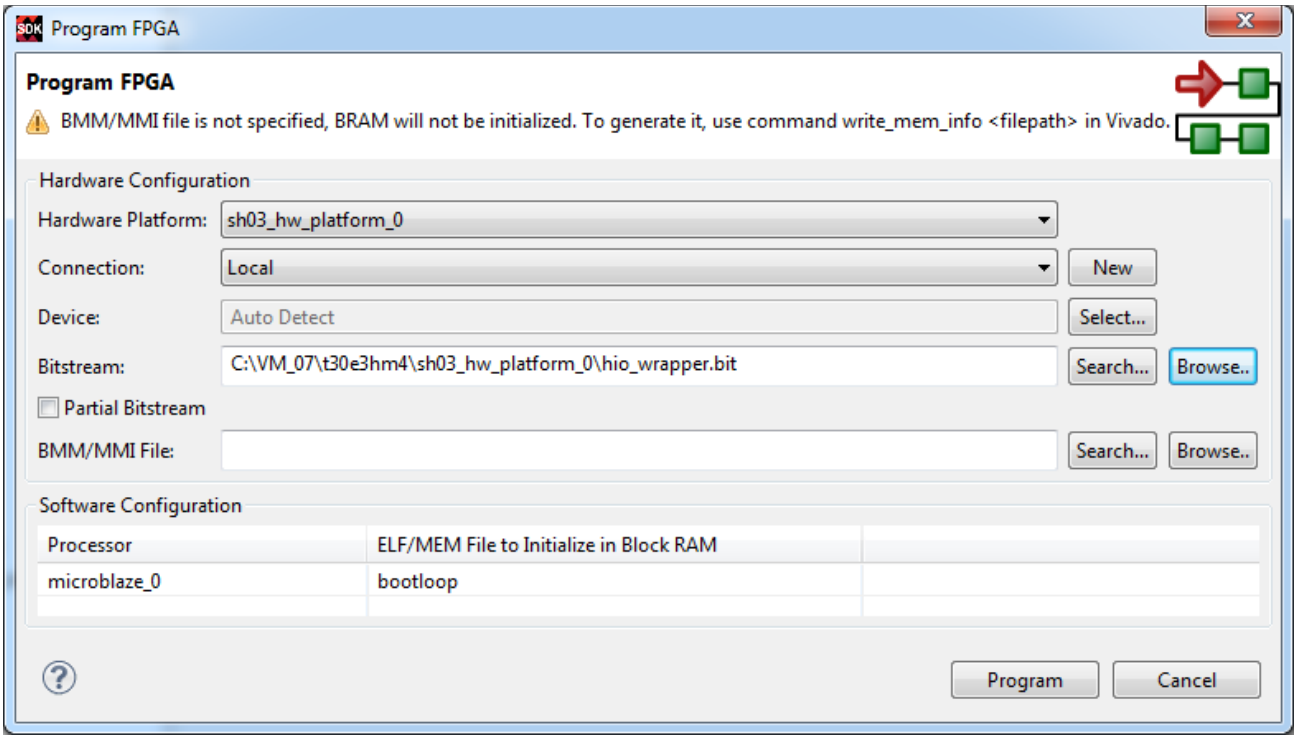


Figure 26: Download bitstream to the PL part of Zynq.

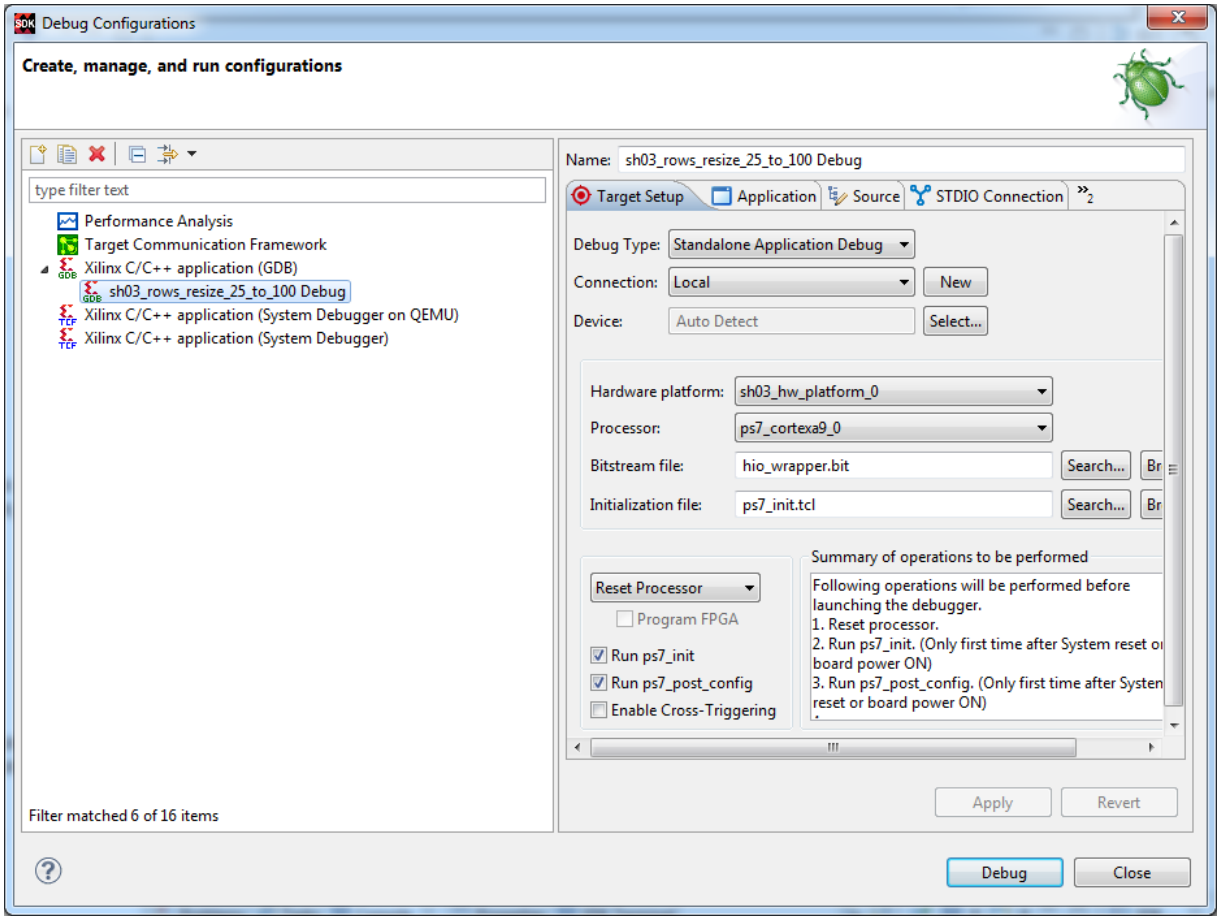


Figure 27: Select demo application for debug.

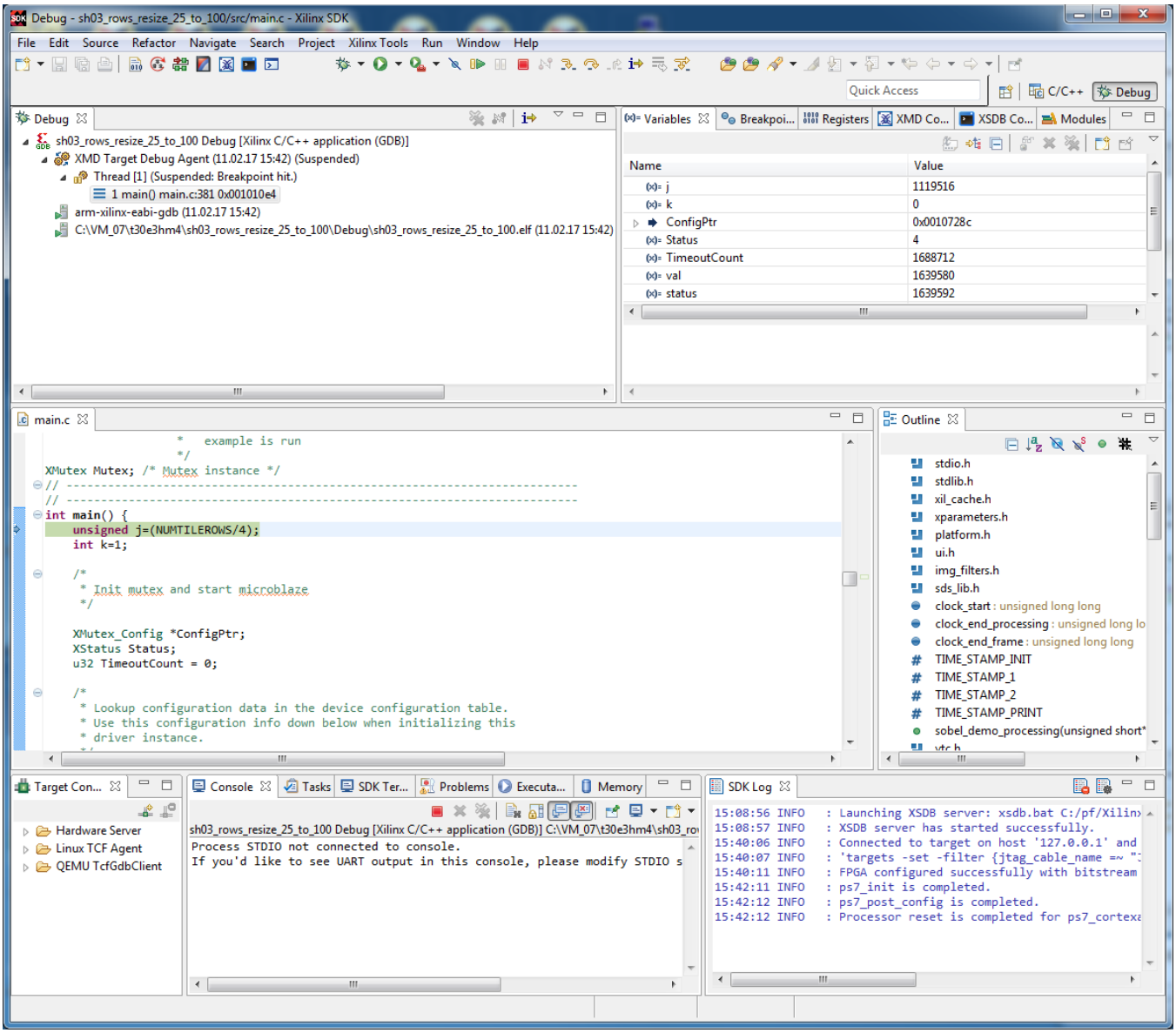


Figure 28: Demo app is booted to ARM and the debugger is waiting on the first executable line.

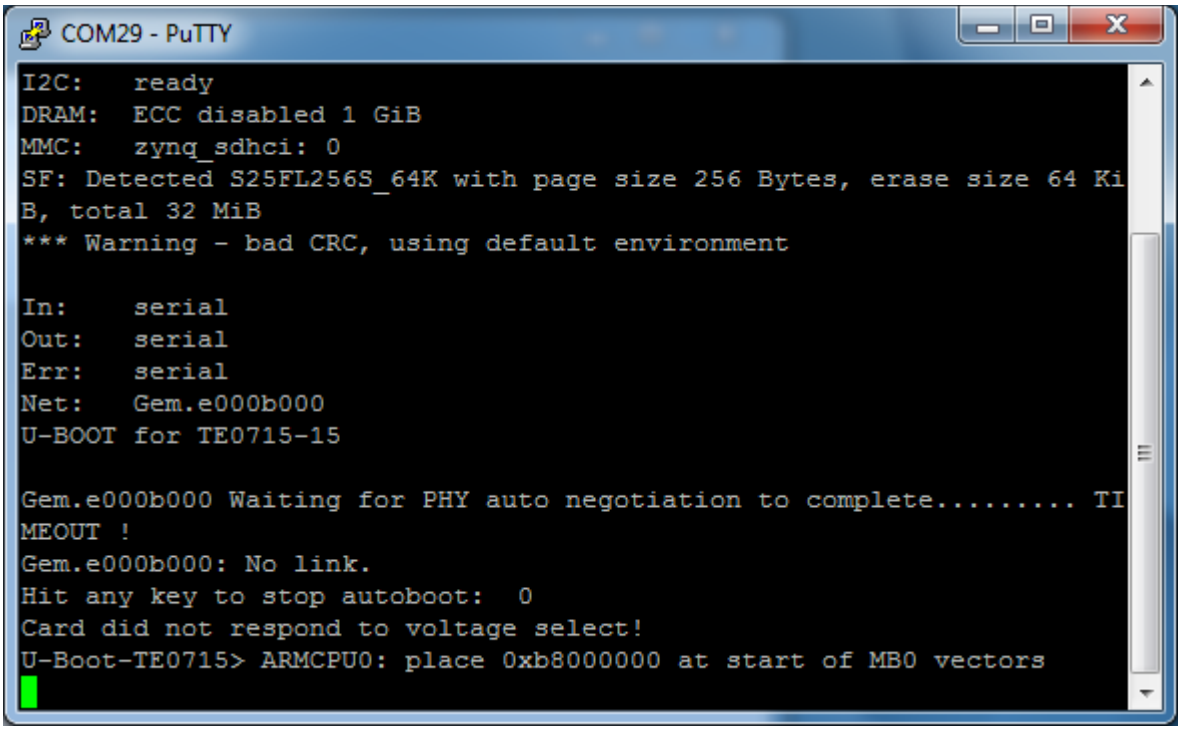


Figure 29: ARM is waiting on HW Mutex for the MicroBlaze start.

The debug perspective is opened and **Debug\sh03_rows_resize_25_to_100.elf** can be debugged or started on the ARM core. See Figure 28 Start the Resume button (F8) of the program from the debugger. It starts to run, with output to the terminal window. The timer is started with 3ns resolution. CPU0: on terminal is indicating the output from the Core_0 of the dual core Cortex A9 of the ZYNQ. The ARM processor is running and waiting in a pooling loop for handshake with MicroBlaze. See Figure 29.

The ARM application **Debug\sh03_rows_resize_25_to_100.elf** has prepared the initial waiting loop code for the MicroBlaze processor at the address 0x30000000 in the DDR3. The MicroBlaze has been released from reset by the ARM application. MicroBlaze is running the initial loop code at the address 0x30000000 now.

The MicroBlaze application **Debug\sh03_edkdsp_fp12_1x8_all.elf** will be loaded to the DDR3 memory in next steps.

The SDK has to connect to the running MicroBlaze via JTAG. The MicroBlaze processor will be stopped under the jtag control. The **Debug\sh03_edkdsp_fp12_1x8_all.elf** code will be downloaded to DDR3 and the MicroBlaze will be started again by JTAG from the second debugger instance.

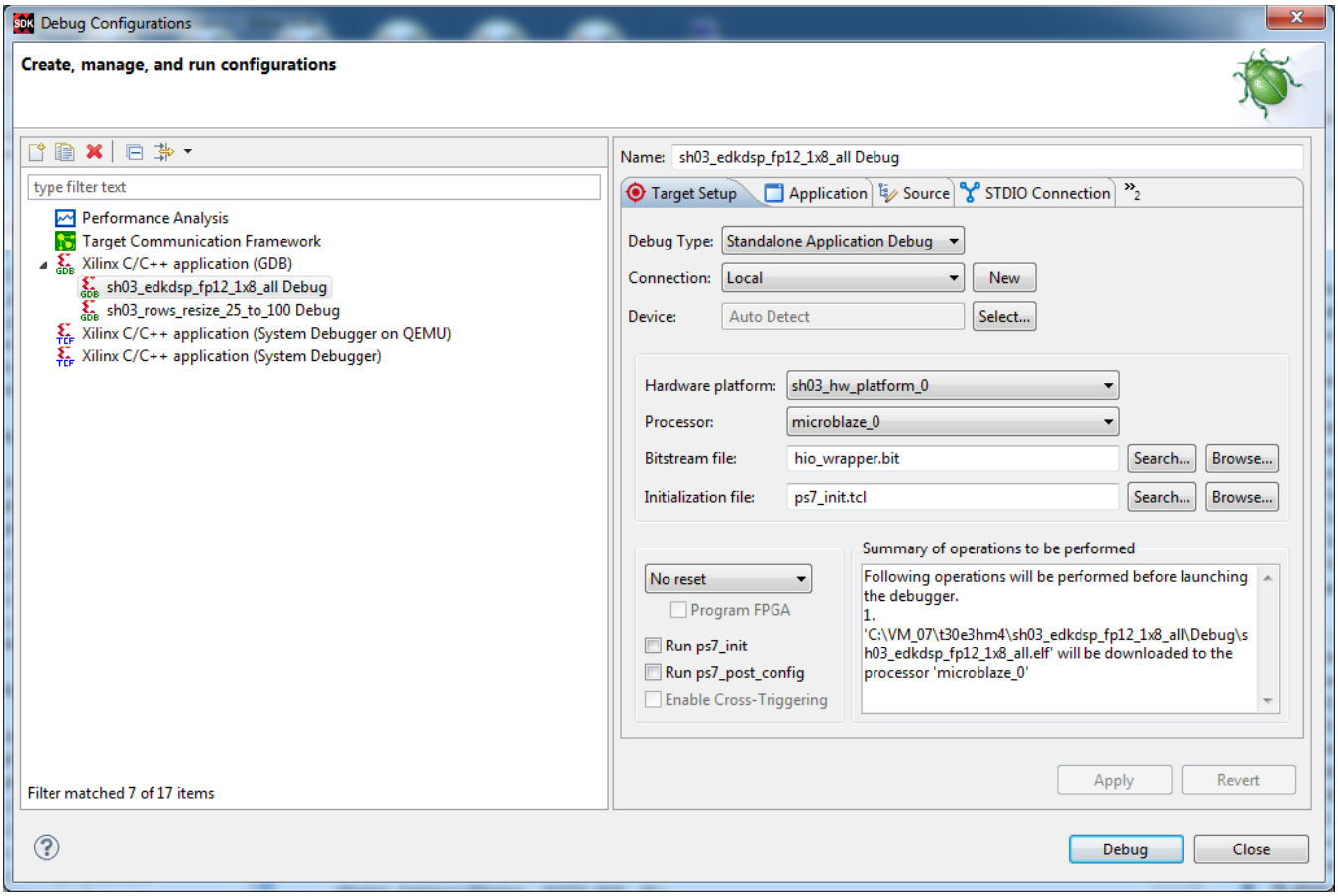


Figure 30: Select the MicroBlaze application (with the EdkDSP accelerator code) for debug.

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.

Click on “Debug” to download the **Debug\sh03_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 31.

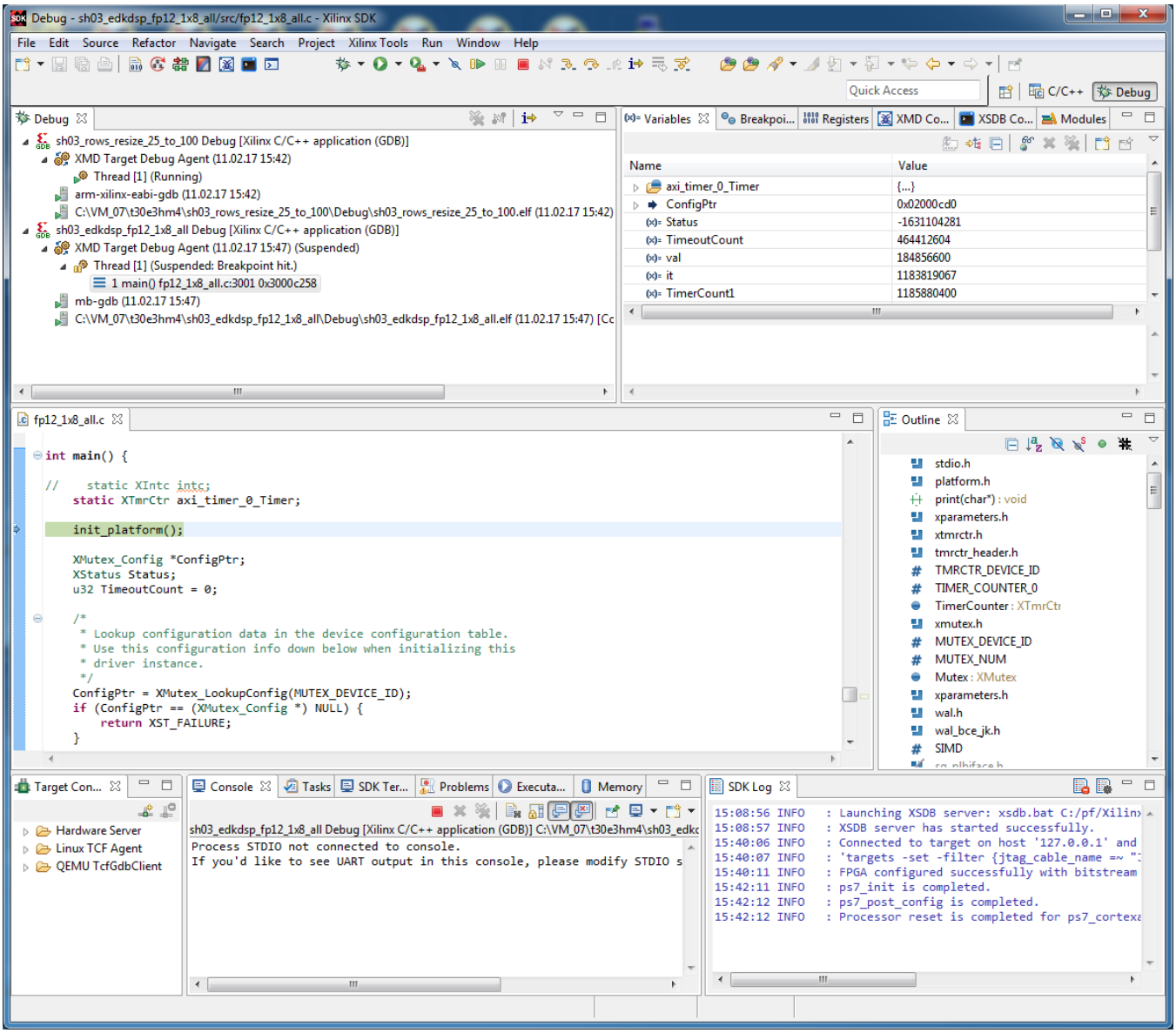


Figure 31: MicroBlaze application is loaded and debugger stops on the first instruction.

- ARM thread is running.
- MicroBlaze thread is currently suspended at breakpoint hit. See Figure 31.

Click on the |> icon to start the execution of MicroBlaze. The SW hands hake between ARM and MicroBlaze supported by the HW Mutex IP is completed at this point. Both processors start to run. ARM initiates the Full HD HDMI Video processing IP cores. It controls in SW status of VDMA units and sets correct pointers to the active video frame buffers. Video processing is performed by HW accelerators. Data are moved from video frame buffers to HW accelerators and back to output video frame buffers by HW data mover IPs. All HW IPs are configured by the ARM SW via the Axi-Lite interface.

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

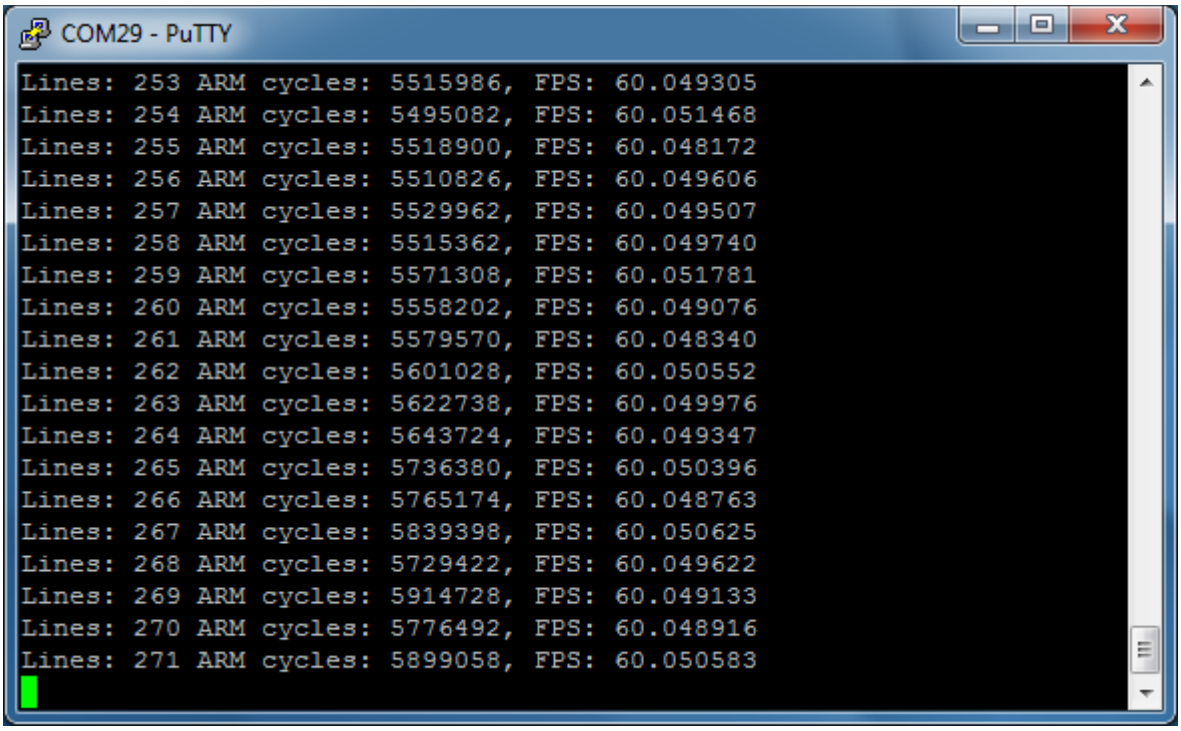


Figure 32: ARM is running. It indicates the number of frames per second.

The MicroBlaze processor executes in parallel with the ARM CPU program from DDR3. It sets up the firmware and also data for the (8xSIMD) EdkDSP floating point accelerators while these accelerators are in reset stage.

MicroBlaze program runs test of all the basic floating point operations which are supported by the EdkSPP accelerators and verifies, if the (8xSIMD) EdkDSP results are bit-exact identical with the reference MicroBlaze results.

In the next stage MicroBlaze reprograms the (8xSIMD) EdkDSP to perform first FIR filter, working on predefined data received from the DDR3 memory. As next stage, with another firmware, the (8xSIMD) EdkDSP accelerates an LMS adaptive filter, working again on predefined I/O acoustic data received from the DDR3 memory.

The demo application – the acoustic data processing with (2000 coefficient FIR filter) and (2000 coefficient LMS identification of filter coefficients) - is computed in single precision floating point in the first 8xSIMD EdkDSP accelerator (with support from MicroBlaze).

Finally, the same demo application (2000 coefficient FIR filter) and (2000 coefficient LMS identification of filter coefficients) is also computed in single precision floating point on MicroBlaze with the HW floating point unit to verify, that the (8xSIMD) EdkDSP results are identical to the MicroBlaze results.

The performance of the combination of MicroBlaze with EdkDSP accelerator is measured by HW timer instantiated as MicroBlaze AXI-Lite IP core. See Figure 33.

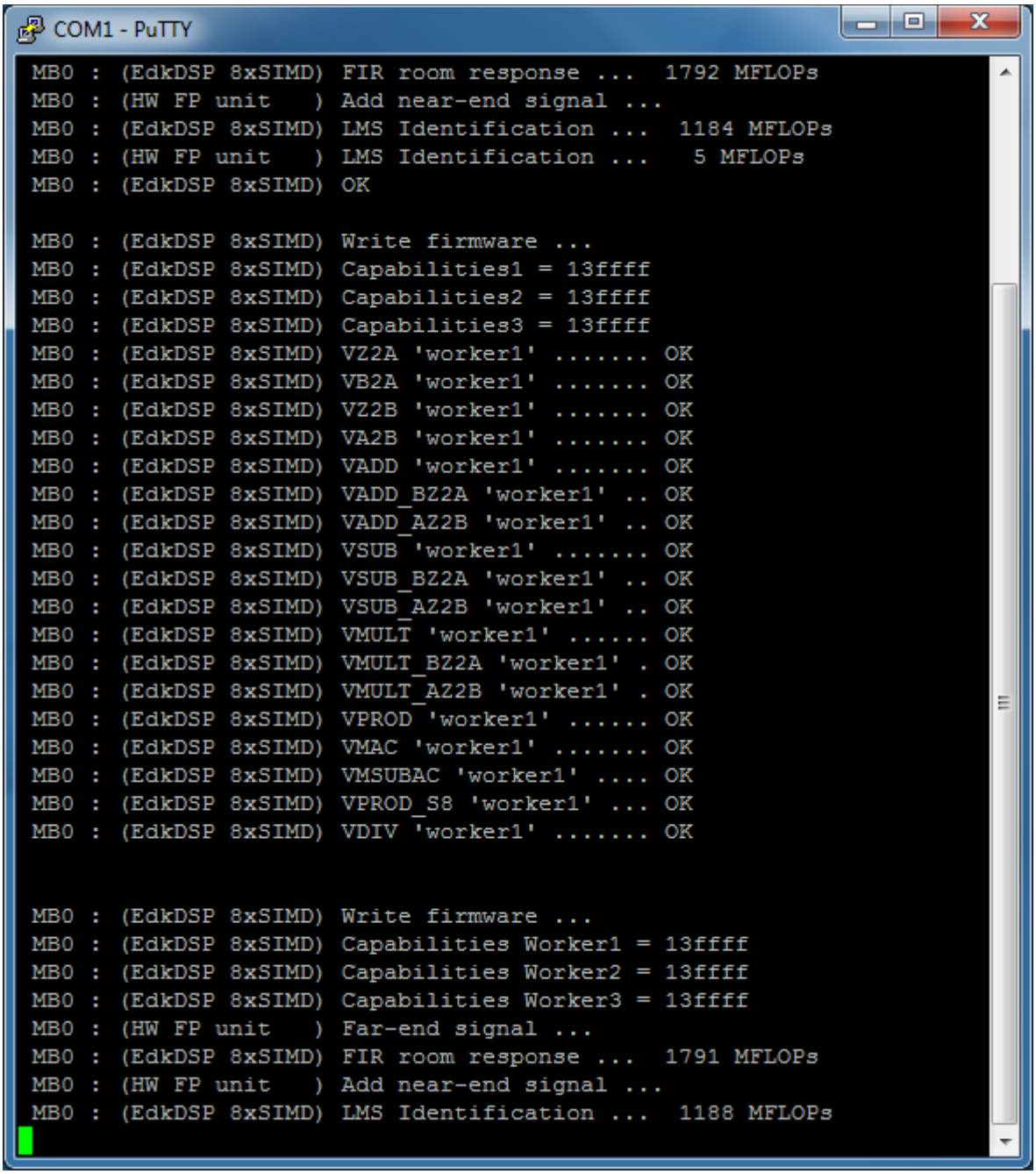


Figure 33: MicroBlaze is running. Debug version. It indicates MFLOPs.

The (8xSIMD) EdkDSP accelerators are named worker1 … worker3. All 3 workers have identical capabilities.

Each of the two parallel running processors (ARM and MicroBlaze) can be stopped/resumed/terminated from the debugger.

Finally, terminate the debug session by this sequence of commands.

1. Stop MicroBlaze.
2. Stop Arm.
3. Terminate MicroBlaze.
4. Terminate Arm.
5. Close the debug perspective.

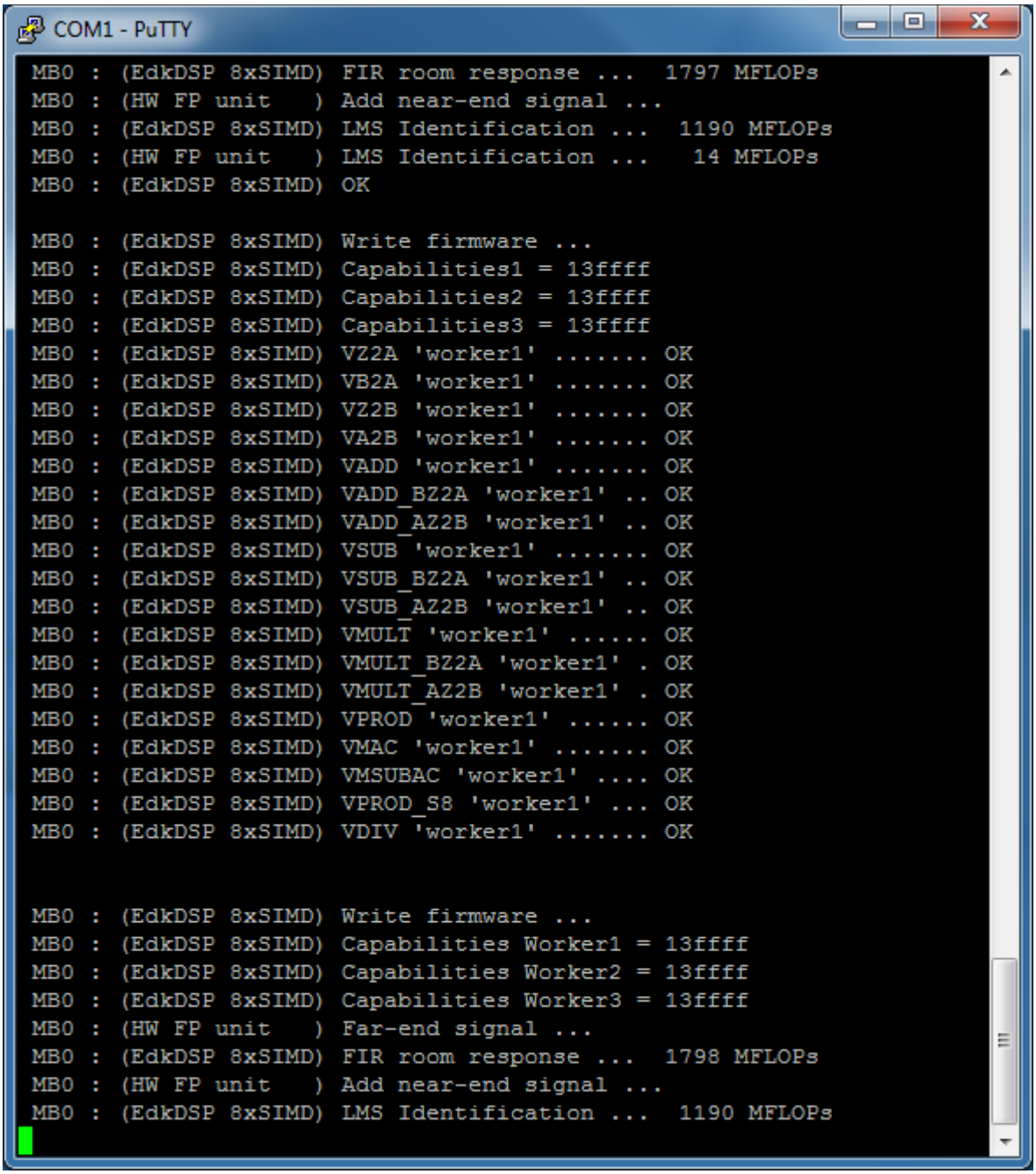


Figure 34: MicroBlaze is running. Release version. It indicates MFLOPs.

- All evaluation demos can be also compiled into release versions with optimisation set to –O2 or -O3. These optimisations can be set independently for the ARM and for the MicroBlaze processor in the SDK 2015.4 project. See MicroBlaze terminal (in Figure 34) as an example output from the release version of the project.
- Demos like sh01_rows_fixed_100 work on complete video frame (with single HW accelerator data path).
- Demos like sh01_rows_resize_25_to_100 work with identical bitstream and HW video-processing accelerators, but the ARM SW is setting dynamically the number of lines to be processed for each new frame. In the demo, ARM scales the number of horizontal lines from ¼ of the frame to the complete frame. The HW data movers are instructed about the number of lines to be processed. Demo SW running on ARM is writing this information to the AXI-lite configuration registers of the data mover IP cores before start of processing of each frame.
- Please notice, that part of the frame which is not processed is propagated to the HDMI output via the cyclic structure of the 8 video frame buffers. See Figure 35 for an example of three parallel variable data paths.

- 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. See Figure 35.
- Demo md01_rows_fixed_100 works with one HW video processing chain with fixed set of processed lines.

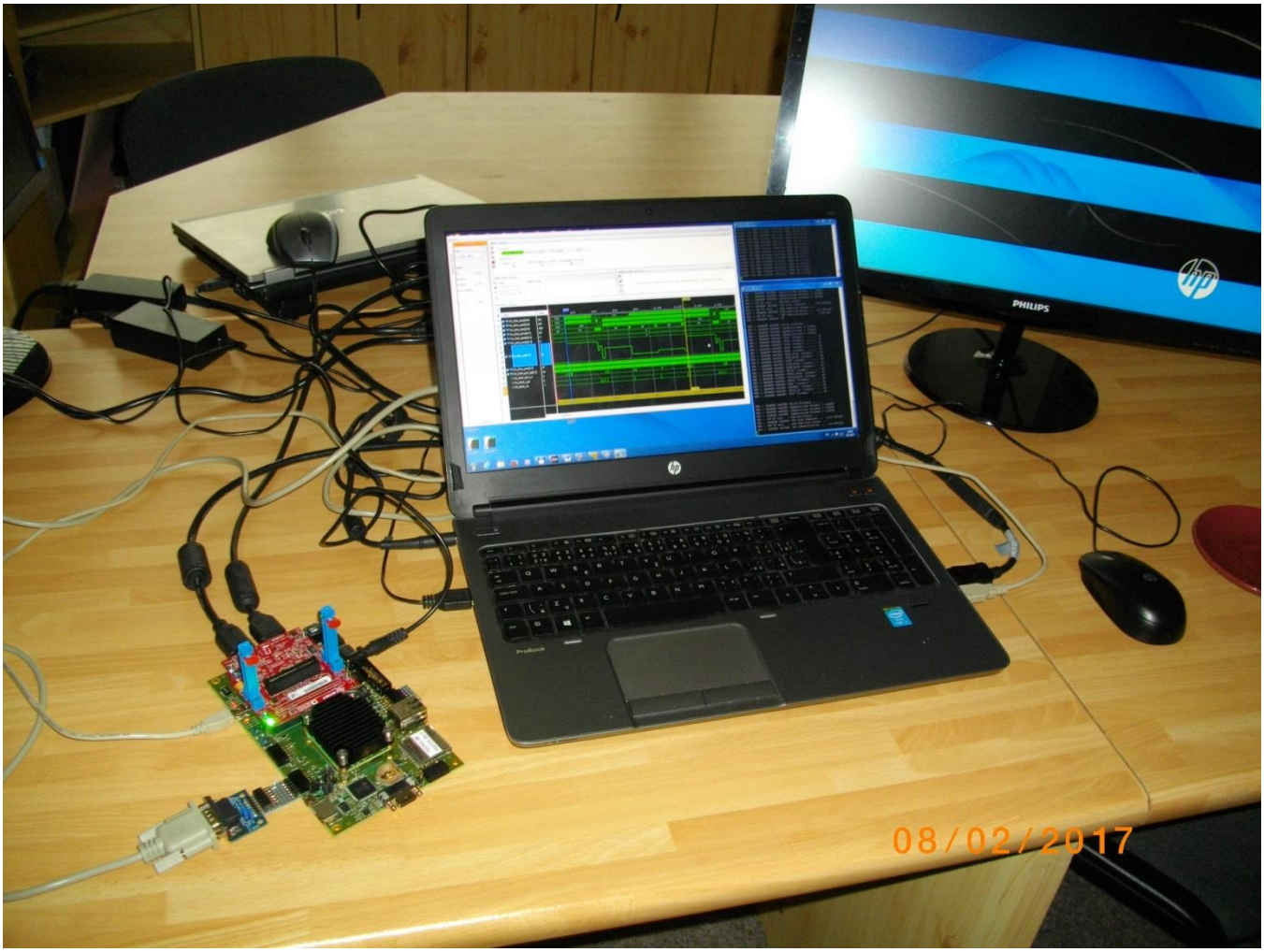


Figure 35: HW accelerated edge detection video processing, in parallel with the EdkDSP accelerator.

2.3 Synchronisation of ARM C/C++ code with 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 **sh03_rows_resize_25_to_100** project as an example. Three HW data paths perform edge detection in parallel on 3 separate areas of a DDR3 video frame. ARM C code is calling function (see Table 1.):

C:\VM_07\t30e3hm4 \sh03_rows_resize_25_to_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(3)
      _p0_sobel_filter_htile3_0(fb_in + 2*(NUMTILEROWS-1)*NUMPADCOLS,
                  fb_out + 2*(NUMTILEROWS-1)*NUMPADCOLS, NUMTILEROWS);

#pragma SDS async(2)
      _p0_sobel_filter_htile2_0(fb_in + (NUMTILEROWS-1)*NUMPADCOLS,
                  fb_out + (NUMTILEROWS-1)*NUMPADCOLS, (NUMTILEROWS+1));

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

// Parallel ARM code here

      sds_wait(3);
      sds_wait(2);
      sds_wait(1);
}
```

Figure 36: Listing of ARM C function using the internal synchronisation with parallel HW data paths.

The three functions:
_p0_sobel_filter_htile3_0() // Not blocking, Starts HW path 3
_p0_sobel_filter_htile2_0() // Not blocking, Starts HW path 2
_p0_sobel_filter_htile1_0() // Not blocking, Starts HW path 1

are corresponding to the three HW video acceleration data paths. These functions are independent. Each of functions only starts its HW data path. All three functions are not blocking. All three functions have been defined in the original SDSoC 2015.4 project with the #pragma SDS async and exported in the libsh03.a static library. The synchronisation point (similar to a barrier in case of SW threads) is implemented separately by three calls to the functions sds_wait(3); sds_wait(2); 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. This parallel processing is implemented in a single SW thread.

The video processing speed will be unaffected, if the time needed for the ARM code segment is shorter than the time needed for the parallel, HW controlled data paths.

Internal synchronisation with parallel HW data paths

Table 2 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 Table 2 is fixed. It cannot be changed. It is related to the md01 HW pipeline.

C:\VM_07\t30e3hm4\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;
unsigned short  yc_out_tmp1[NUMROWS*NUMCOLS], yc_out_tmp2[NUMROWS*NUMCOLS];
unsigned short  yc_out_tmp3[NUMROWS*NUMCOLS], yc_out_tmp4[NUMROWS*NUMCOLS];
unsigned char   sobel_curr[NUMROWS*NUMCOLS], sobel_prev[NUMROWS*NUMCOLS];
unsigned char   motion_image_tmp1[NUMROWS*NUMCOLS];
unsigned char   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;

    _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 37: Listing of ARM C function with fixed data width interfacing HW pipeline of accelerators.

2.4 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.

2.5 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 your PC.

The UTIA EdkDSP C compiler is included in 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.

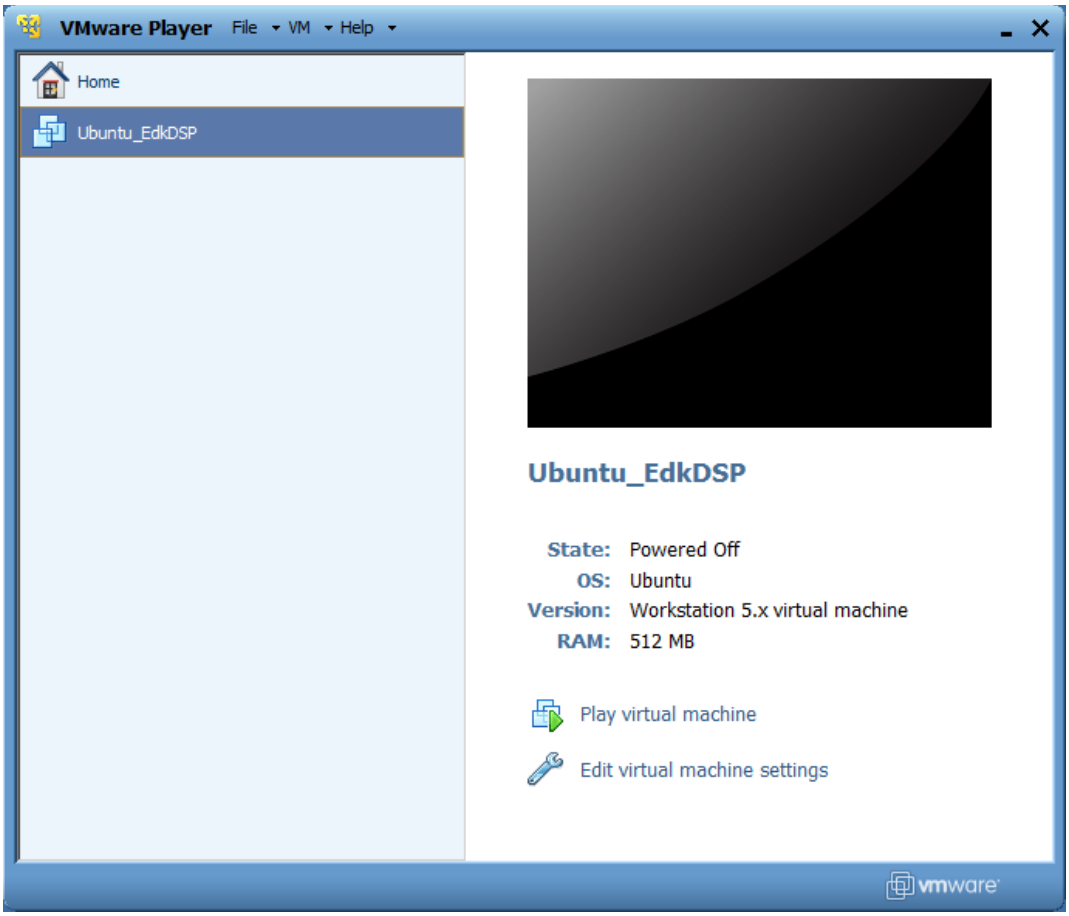


Figure 38: Select the Ubuntu_EdkDSP image in the VMware Player and click "Play".

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/t30e3hm4/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:

source settings.sh

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

/mnt/cdrive/t30e3hm4/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 40 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 41.

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 **sh03_edkdsp_fp12_1x8_all**) to:

C:\VM_07\t30e3hm4\sh03_edkdsp_fp12_1x8_all\src

Recompile the MicroBlaze project "**sh03_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 three instances of the (8xSIMD) EdkDSP floating point accelerator IP cores **bce_fp12_1x8_40**.

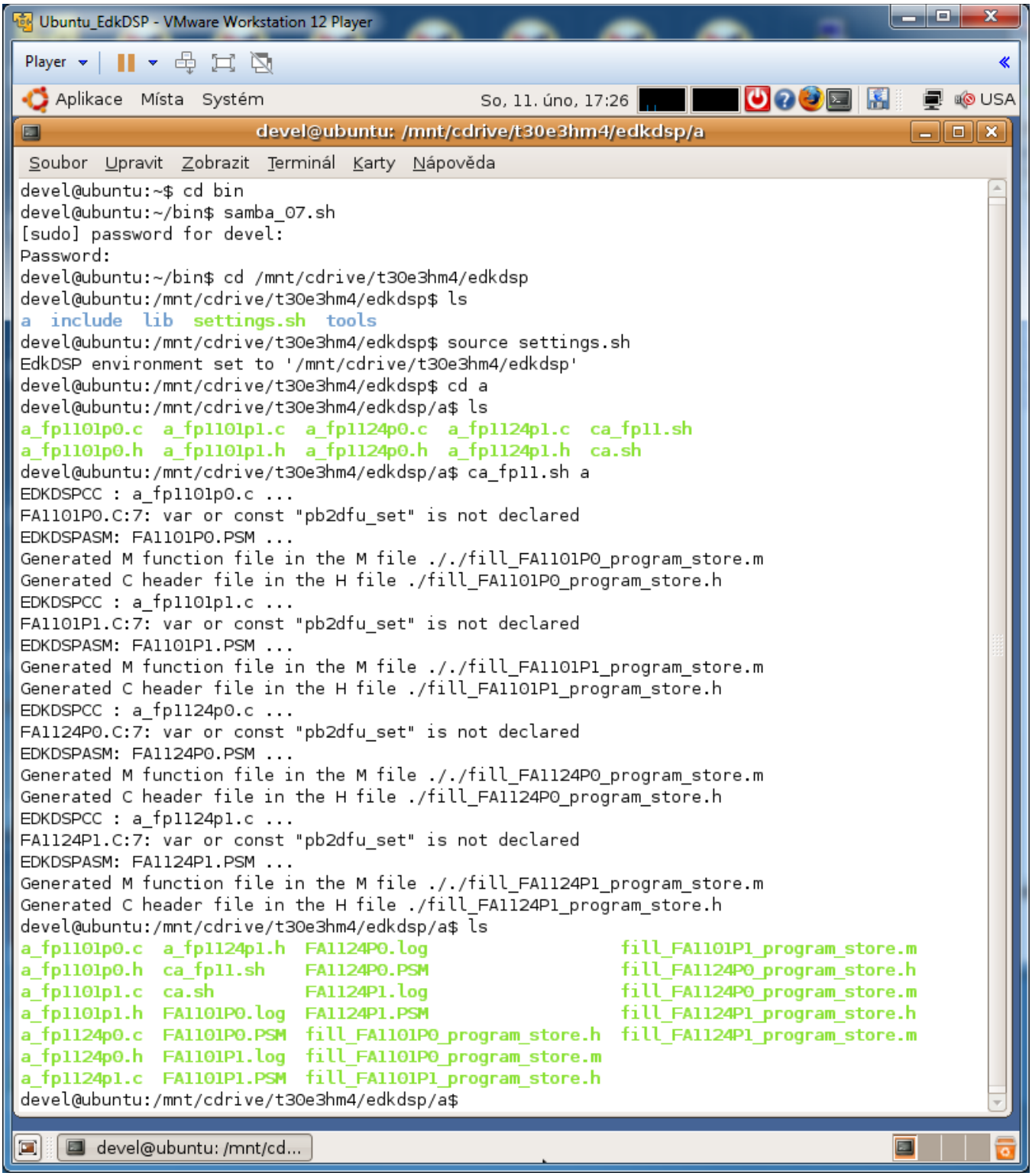


Figure 39: Compilation of EdkDSP firmware in Ubuntu.

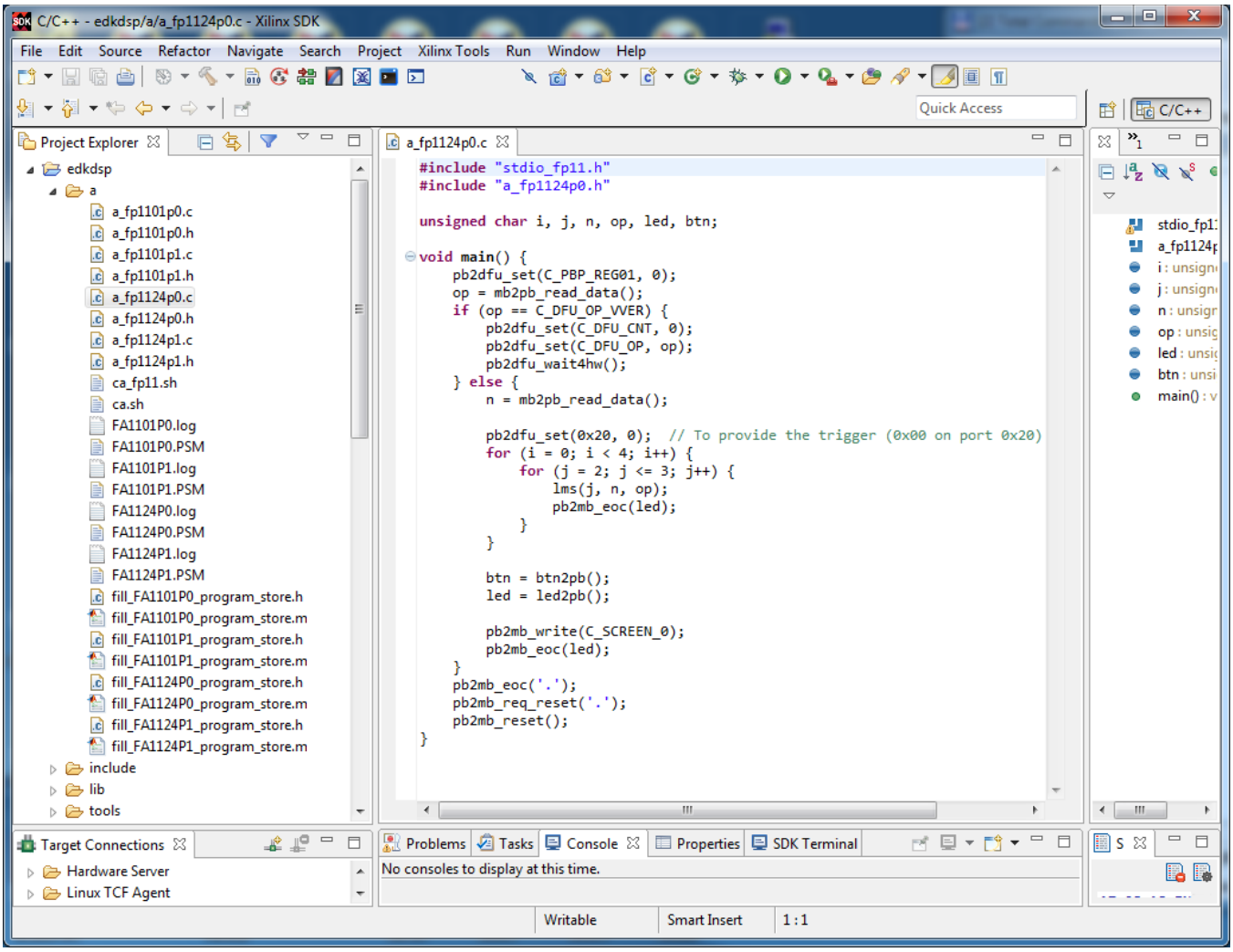


Figure 40: C listing of the LMS filter firmware for the EdkDSP.

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 10. This accelerator is optimized for applications like floating point matrix multiplication with one row and column dimensions <= 10.
- **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 (in the range from 11 to 250).
- **bce_fp12_1x8_40** support 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 versions IP cores are suitable for applications like adaptive normalised NLMS filters, Square-root-free versions of adaptive RLS QR filters and Adaptive RLS LATTICE filters.

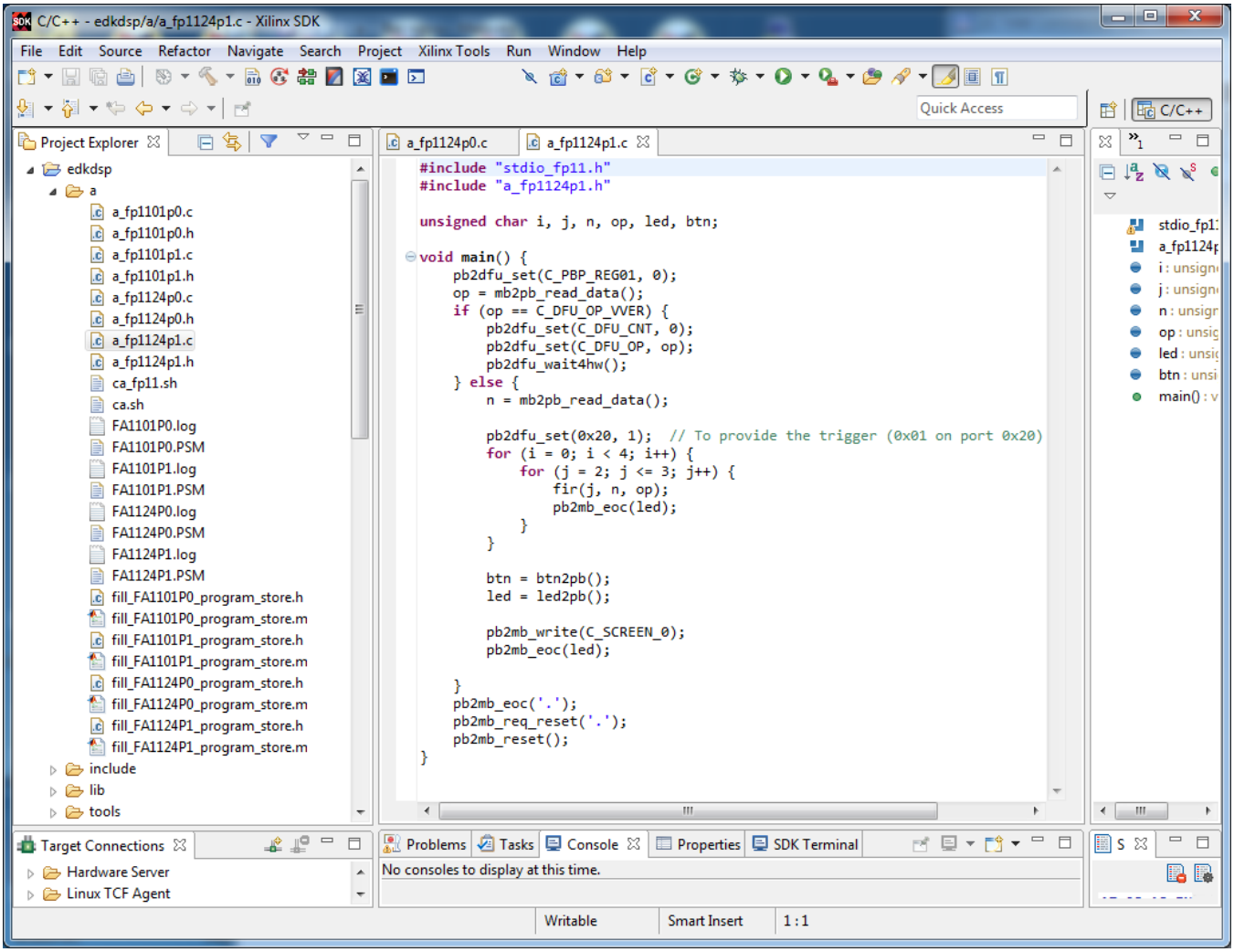


Figure 41: C listing of the FIR filter firmware for the EdkDSP.

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

This application note includes evaluation version of designs design with ARM Cortex A9 processor, MicroBlaze processor controlling three (8xSIMD) EdkDSP accelerators. 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 three (8xSIMD) EdkDSP accelerators and ILA support is present in the directory:

C:\VM_07\ t30e3hm4\sh03_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 sh03 demo:
In SDK, select: **Xilinx Tools -> Program FPGA** select **"sh03_hw_platform_0_ila" (**instead of **"sh03_hw_platform_0").** Click on the "**Program**" button.

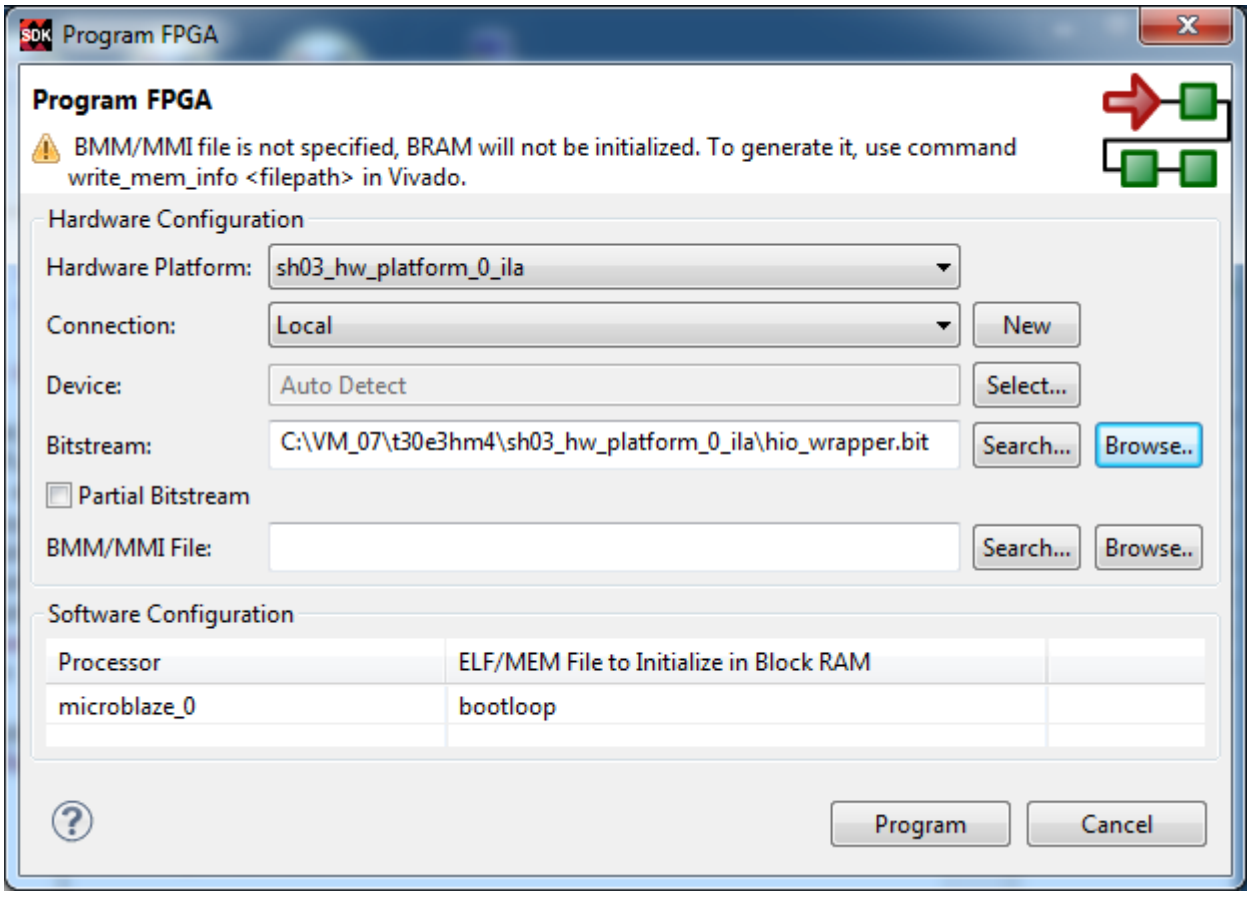


Figure 42: Change the default hw_platform_0 to the hw_platform_0_ila.

The implemented In-Circuit Logic Analyser (ILA) stores 32k 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.

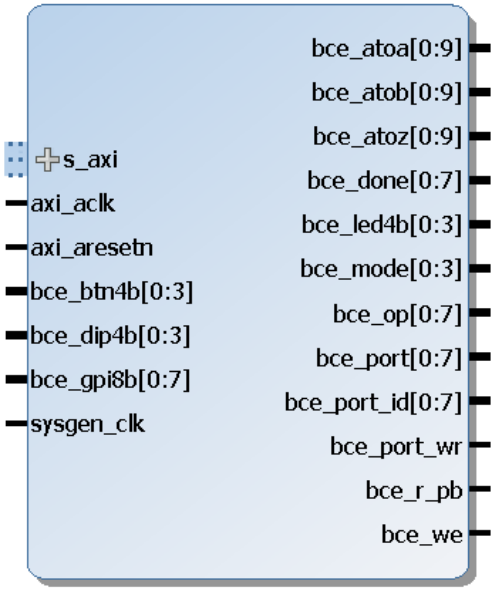


Figure 43: Debug ports of the (8xSIMD) EdkDSP floating point accelerator IP core.

Debug ports of the (8xSIMD) EdkDSP accelerator

All debug ports are stored with depth of 32k samples with the sample frequency 150 MHz (see Figure 43):

- 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_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 am[i] <= zm[j]; m=1..8
WAL_BCE_JK_VB2A	**= 2**	8xSIMD copy am[i] <= bm[j]; m=1..8
WAL_BCE_JK_VZ2B	**= 3**	8xSIMD copy bm[i] <= zm[j]; m=1..8
WAL_BCE_JK_VA2B	**= 4**	8xSIMD copy bm[i] <= am[j]; m=1..8
WAL_BCE_JK_VADD	**= 5**	8xSIMD add zm[i] <= am[j] + bm[k]]; m=1..8
WAL_BCE_JK_VADD_BZ2A	**= 6**	8xSIMD add am[i] <= bm[j] + zm[k]]; m=1..8
WAL_BCE_JK_VADD_AZ2B	**= 7**	8xSIMD add bm[i] <= a[j] + z[k]]; m=1..8
WAL_BCE_JK_VSUB	**= 8**	8xSIMD sub zm[i] <= am[j] - bm[k]; m=1..8
WAL_BCE_JK_VSUB_BZ2A	**= 9**	8xSIMD sub am[i] <= bm[j] - zm[k]; m=1..8
WAL_BCE_JK_VSUB_AZ2B	**= 10**	8xSIMD sub bm[i] <= am[j] - zm[k]; m=1..8
WAL_BCE_JK_VMULT	**= 11**	8xSIMD mult zm[i] <= am[j] * bm[k]; m=1..8
WAL_BCE_JK_VMULT_BZ2A	**= 12**	8xSIMD mult am[i] <= bm[j] * zm[k]; m=1..8
WAL_BCE_JK_VMULT_AZ2B	**= 13**	8xSIMD mult bm[i] <= am[j] * zm[k]; m=1..8
WAL_BCE_JK_VPROD	**= 14**	8xSIMD vector products: zm[i] <= am'[j..j+nn]*bm[k..k+nn]; m=1..8; nn range 1..255
WAL_BCE_JK_VMAC	**= 15**	8xSIMD vector MACs: zm[i..i+nn] <= zm[i..i+nn] + am[j..j+nn] * bm[k..jk+nn]; nn range 1..13
WAL_BCE_JK_VMSUBAC	**= 16**	8xSIMD vector MSUBACs zm[i..i+nn] <= zm[i..i+nn] - am[j..j+nn] * bm[k..jk+nn]; nn range 1..13
WAL_BCE_JK_VPROD_S8	**= 17**	8xSIMD vector product (extended) zm[i] <= ((a1'[j..j+nn]*b1[k..k+nn]+a2'[j..j+nn]*b2[k..k+nn]) + (a3'[j..j+nn]*b3[k..k+nn]+a4'[j..j+nn]*b4[k..k+nn])) + ((a5'[j..j+nn]*b5[k..k+nn]+a6'[j..j+nn]*b6[k..k+nn]) + (a7'[j..j+nn]*b7[k..k+nn]+a8'[j..j+nn]*b8[k..k+nn])); m=1..8; nn range 1..255
WAL_BCE_JK_VDIV	**= 20**	vector division (extended) zm[i] <= a1[j] / b1[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 three (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 44.

Figure 44: Vivado Lab Edition 2015.4

Select: **Open Target** See Figure 45. 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 43) 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/t30e3hm4/sh03_hw_platform_0_ila/ debug_nets.ltx
See Figure 46.

This will add the names and parameters of probes (see Figure 43) to the ILA Waveform window. See Figure 48.

Use **+** to select probes used for triggering, and select the condition for trigger for each probe and their combination (use AND as default).

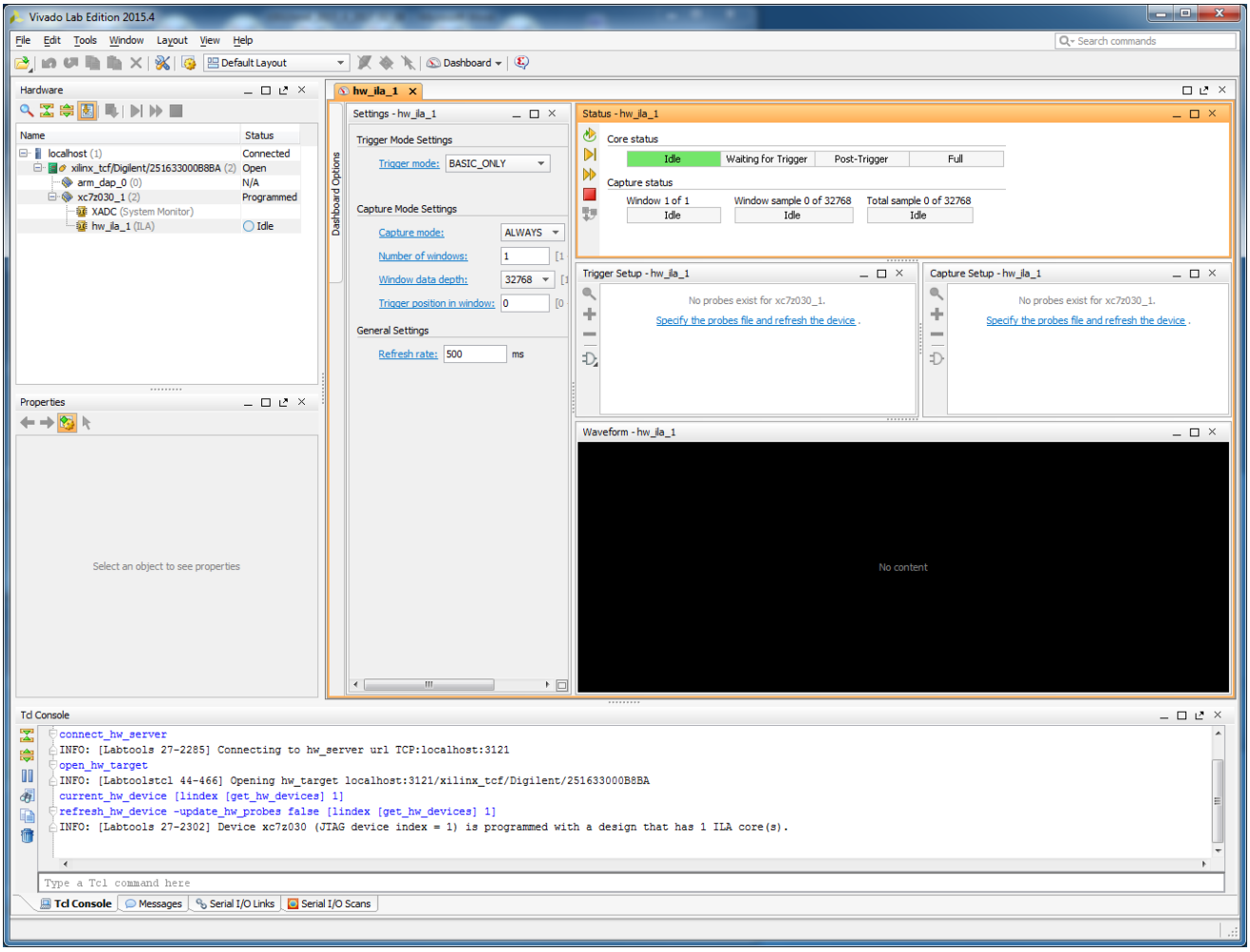


Figure 45: Select Open Target

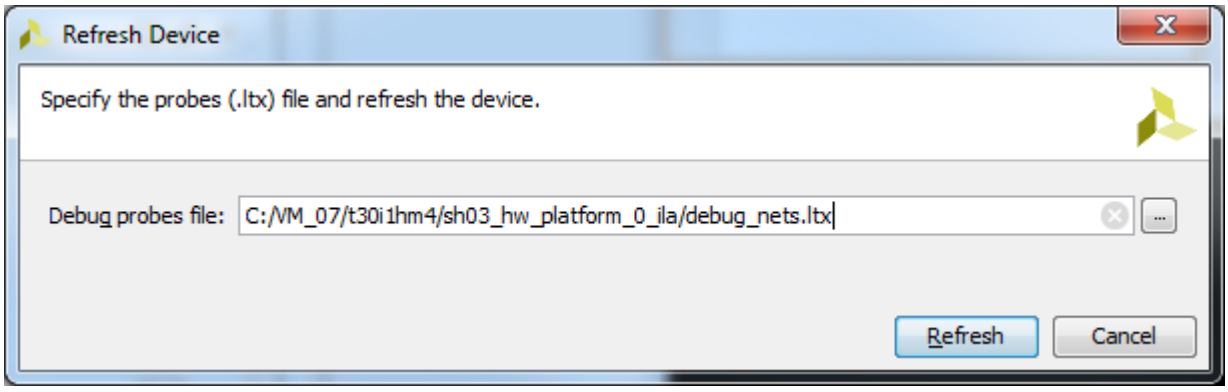


Figure 46: Select file with definition of probes present in HW.

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 modified C code **FIR** 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 41).

```
…
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);
        }
}
…
```

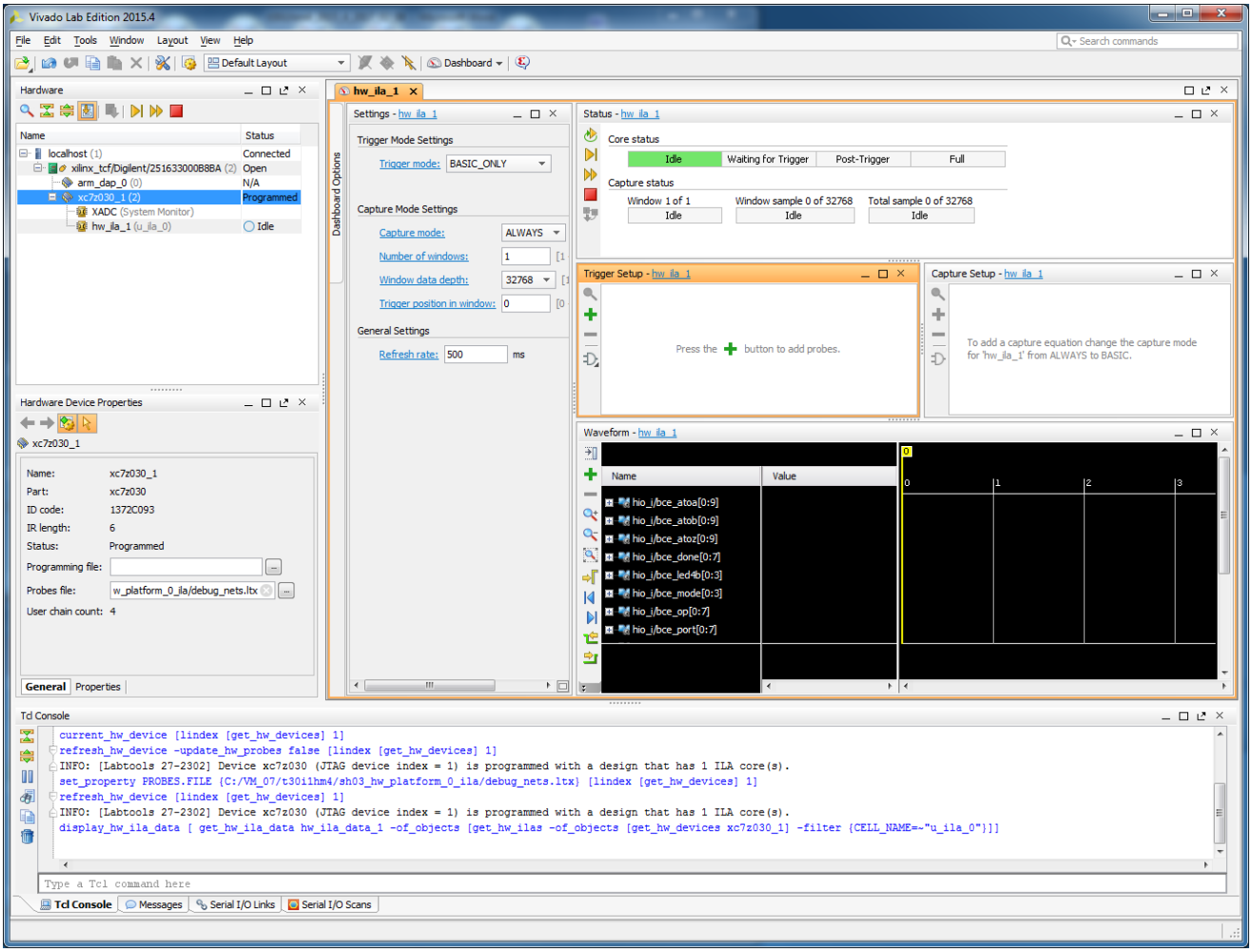


Figure 47: Compilation results of EdkDSP CC compiler. FIR filter code with probe trigger call.

In Vivado Lab Edition, in the ILA configuration page, change the trigger condition to (See Figure 48):

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

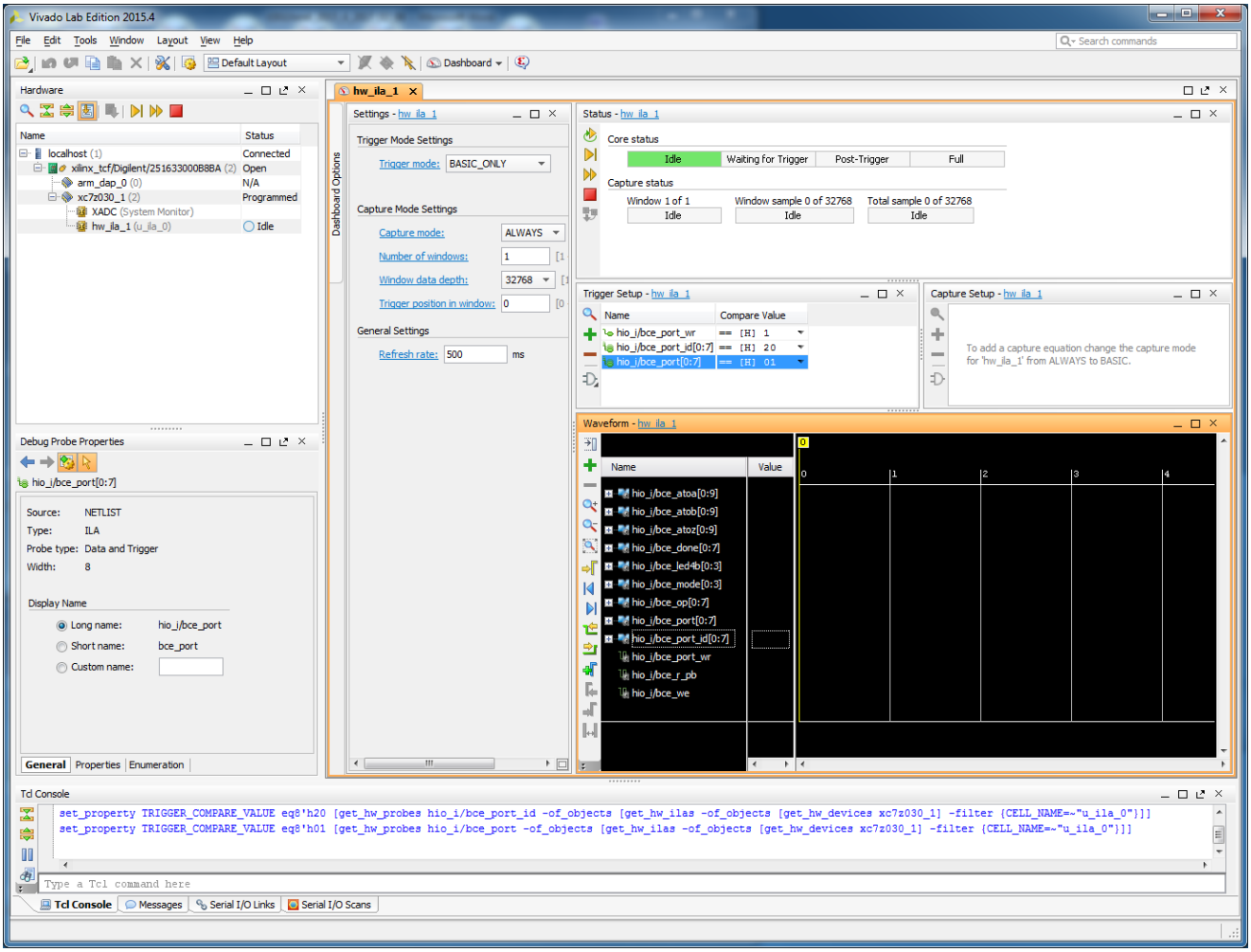


Figure 48: Trigger conditions. Selection in Vivado Lab Edition 2015.4.

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 32K samples of all debug signals with the sampling rate 150 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 32k clock cycles of the FIR filter computation as defined by the SW displayed in Figure 41. The red trigger is corresponding to the event. See Figure 49.

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 (200 MHz = 5.0 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.

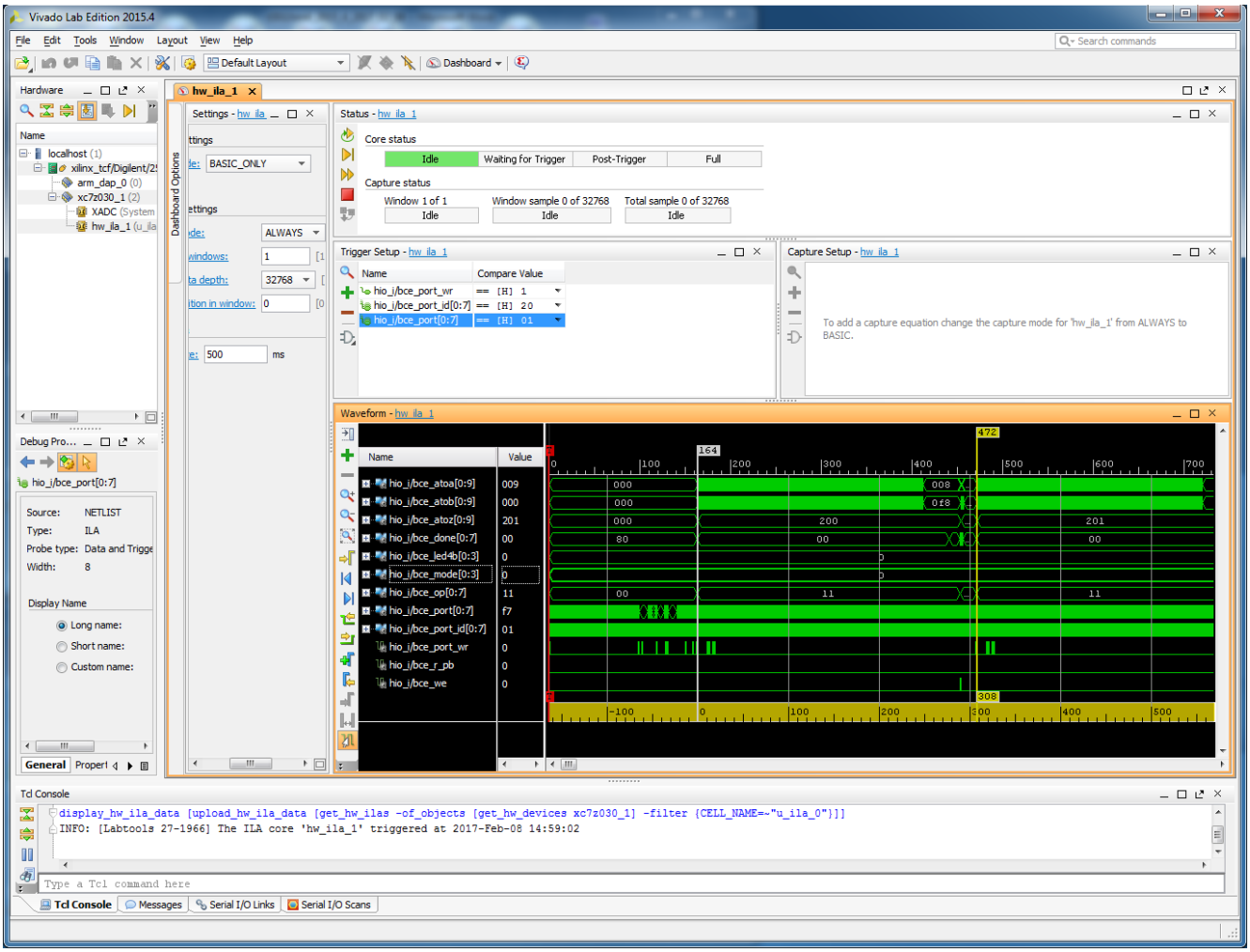


Figure 49: FIR filter waveforms after the trigger in Vivado Lab Edition 2015.4.

While the system is running, we can modify the trigger condition to capture the initial phase of the LMS filter running 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 40.

```
…
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 40). This C code is executed by the corresponding the PicoBlaze6 controller inside of the first EdkDSP accelerator. This will trigger capturing of new 32K samples of all debug signals with the sampling rate 200 MHz and provide detailed trace of the initial 32K samples of the LMS filter computation. See Figure 50 and Figure 51.

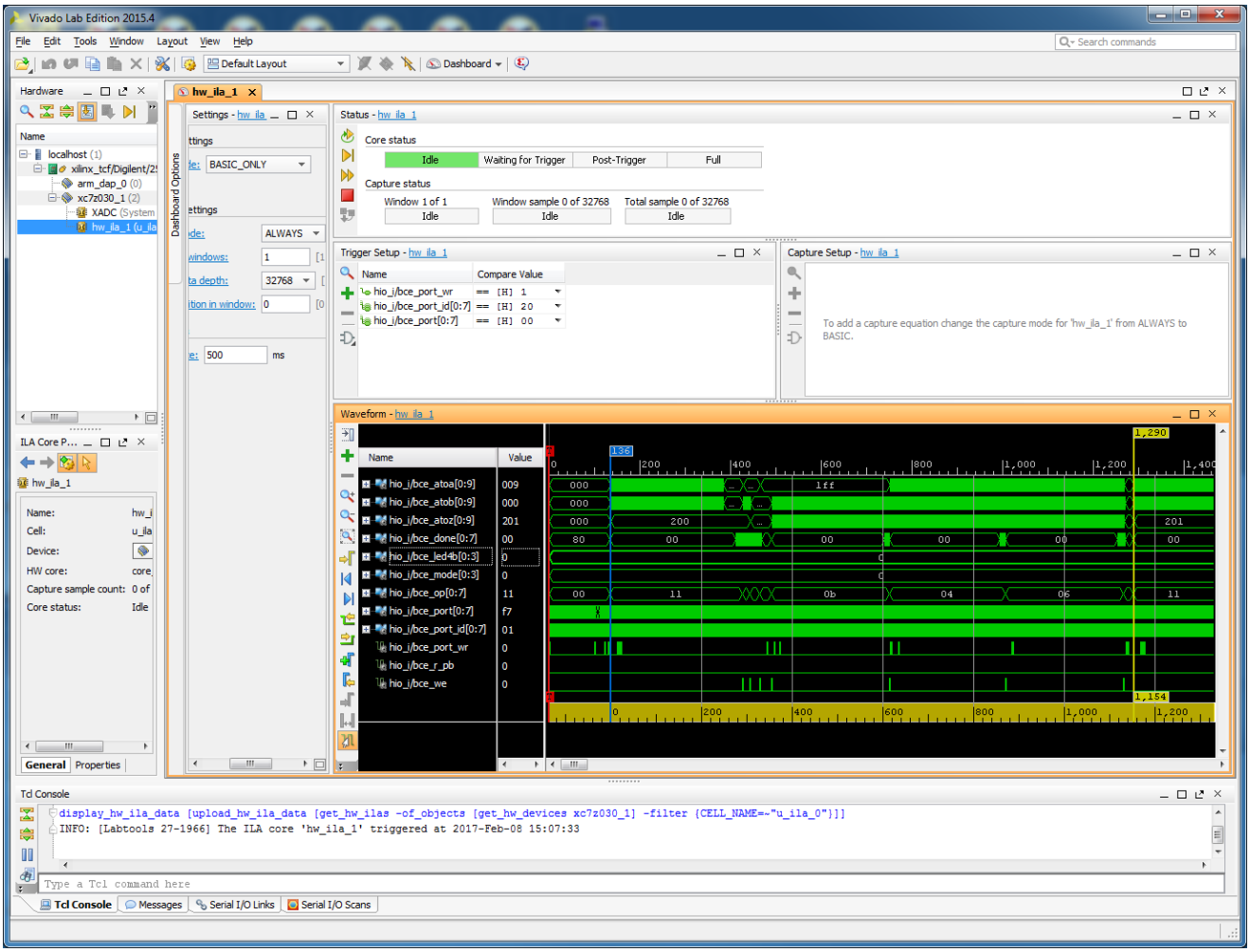


Figure 50: LMS filter waveforms after the trigger in Vivado Lab Edition 2015.4.

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 (200 MHz = 5.0 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 (see Figure 51).

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 40, Figure 51 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.

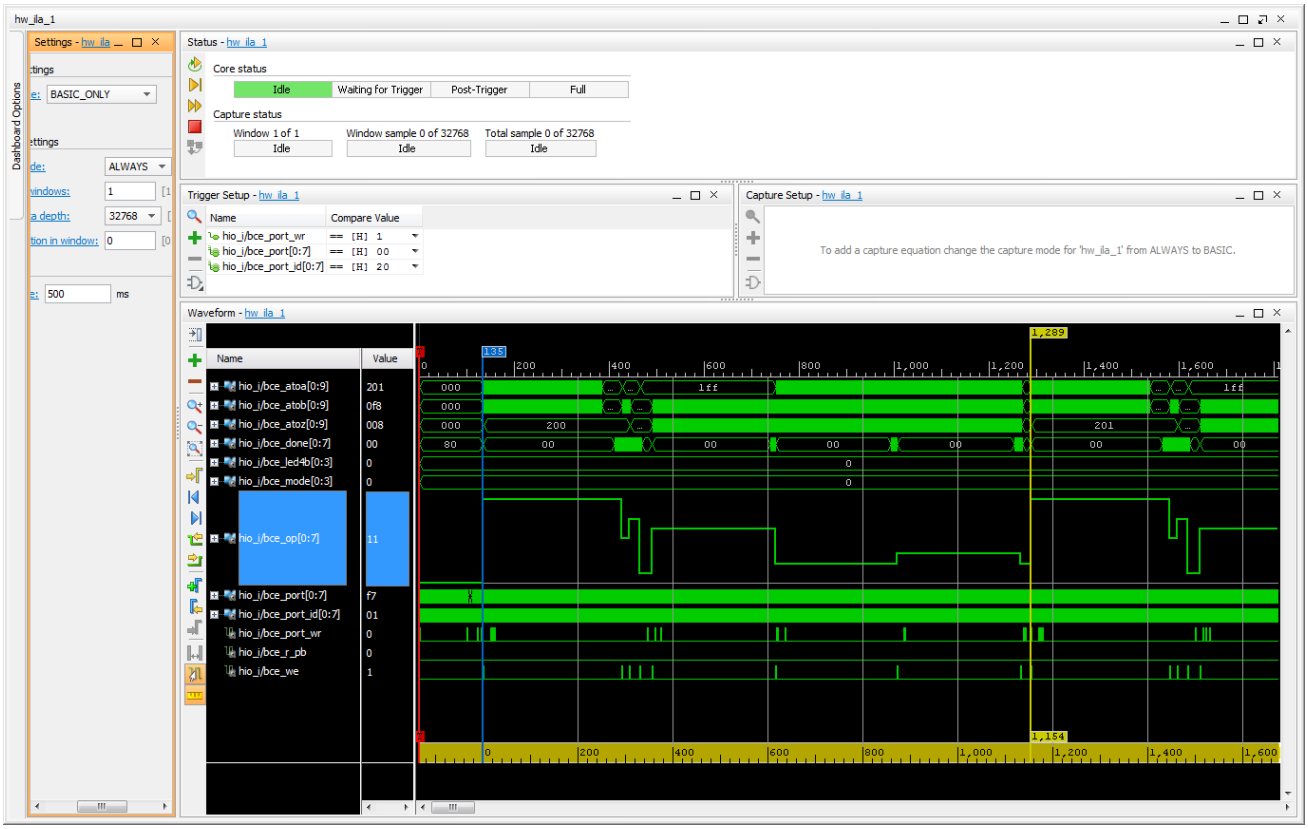


Figure 51: LMS filter sequence of operations defined in the PicoBlaze6 lms() function.

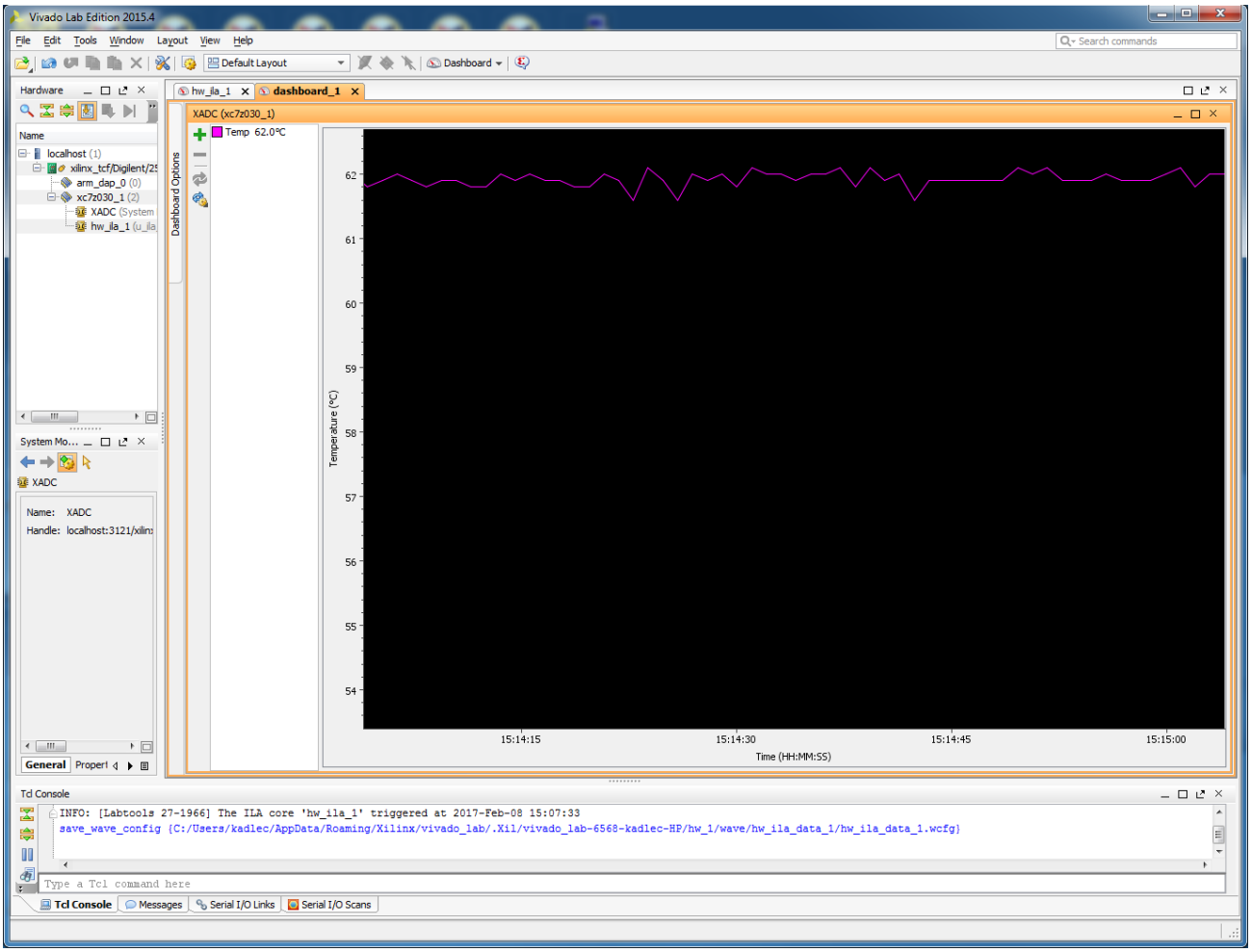


Figure 52: Separate dashboard with display of temperature and voltage in Vivado Lab Edition 2015.4.

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 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.

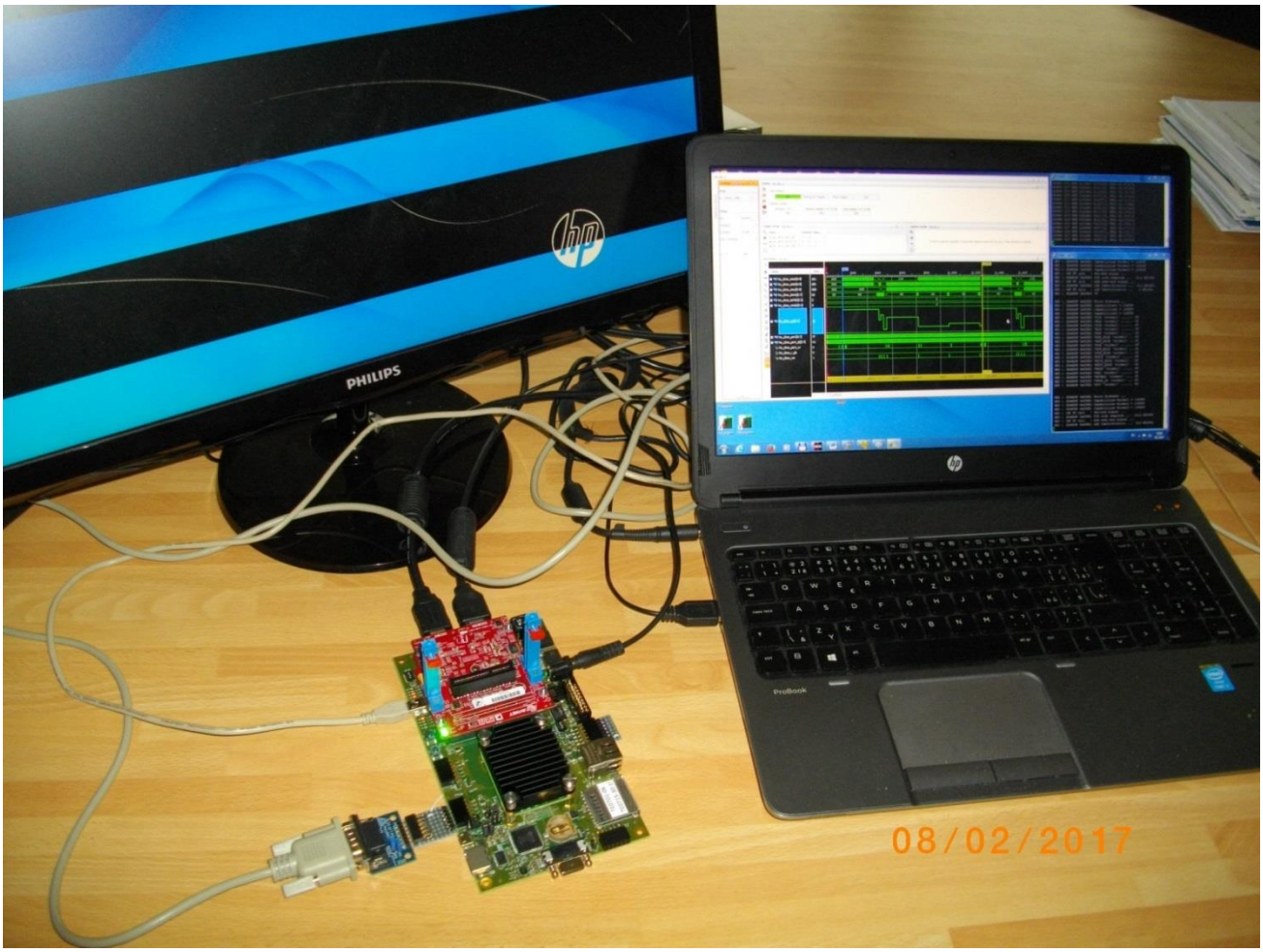


Figure 53: Accelerated video processing algorithm and ILA debug of the accelerated LMS filter.

1. Conclusions

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

- Programmable logic part of the Zynq XC7Z030-3E device is capable of implementation of the UTIA (8xSIMD) EdkDSP floating point accelerator together with the HW accelerated video processing chain for the Full HD HDMII-HDMIO video processing chain with fixed resolution 1920x1080p60.

- The total power consumption for the HW accelerated video processing (measured at the 12V DC power supply) is up to 8.7 W. Such power consumption requires heat sink for the Zynq part.

- The video processing is significantly faster due to the HW accelerators. Acceleration from 5.2x to 30x has been reached in comparison to 1GHz ARM Cortex A9 SW implementation.The acceleration of video processing in HW has been reached while the EdkDSP floating point accelerator computation are performed in the same time in the PL fabric of the Zynq device.

- Designs are provided with and without the ILA debug support.

- The combination of 32bit MicroBlaze with the three (8xSIMD) EdkDSP floating point accelerators brings additional capability to accelerate computation in single precision floating point with the floating point performance from 1.2 GFLOP/s to 1.8 GFLOP/s (1.798 GFLOP/s in case of FIR filter) in one of the three (8xSIMD) EdkDSP accelerators. This floating point performance comes at the expense of relatively moderate increase of the total power consumption of the system:

    - 6.95 W: ARM, md01 HW, MicroBlaze (+14 MFLOP/s) ……..(+6,95 W)
    - 7.79 W: ARM, md01 HW, MicroBlaze, 3x EdkDSP, 3x no use (+14 MFLOP/s)…….(+0.84 W)
    - 8.05 W: ARM, md01 HW, MicroBlaze, 3x EdkDSP, 2x no use, 1x use +1.8 GFLOP/s.(+0.26 W)

    - Power per one GFLOP/s for one (8xSIMD) EdkDSP accelerator (static+dynamic power, no ILA):

        - One EdkDSP HW accelerator (200 MHz) computing LMS Filter: 437 mW/GFLOP/s
        - One EdkDSP HW accelerator (200 MHz) computing FIR Filter: 303 mW/GFLOP/s

MicroBlaze soft core with 3x (8xSIMD) EdkDSP accelerators takes significant part of Zynq PL resources. This limits the maximal number of parallel HW video processing chains and therefore limits the achievable HW acceleration for the video processing in HW.

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 opportunity for the user to make the top level SW adaptations and customisations of the final application in the C source code. The run-time re-programming of the 8xSIMD EdkDSP accelerators in C and ASM is also supported. This user customisation is possible. Xilinx provides the SDK 2015.4 toolchain for free download [8]. User does not need the licensed access to the SDSoC 2015.4 board support package and to the SDSoC 2015.4 [9] license. This application note briefly explained the integration of the Full HD HDMII-HDMIO video processing I/O with the HW accelerated video processing algorithms and with three run-time reprogrammable 8xSIMD EdkDSP floating point accelerators on the commercially available modular hardware [1] – [6].

2. References

[1] TE0715-04-30-3E Xilinx Zynq Z-7030 SoC Micromodule with Xilinx XC7Z030-3SBG485E (ind. temp. range -40°C to +85°C)
https://shop.trenz-electronic.de/en/Products/Trenz-Electronic/TE07XX-Zynq-SoC/

[2] Heatsink for TE0715, 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] TE0701 TRM Revision: V35, 24-Jan-2017 09:32
https://www.trenz-electronic.de/fileadmin/docs/Trenz_Electronic/carrier_boards/TE0701/REV06/Documents/TRM-TE0701-06.PDF

[5] FMC Full HD HDMII-HDMIO extension board AES-FMC-HDMI-CAM-G
http://products.avnet.com/shop/en/ema/3074457345623664802

[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.
http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2015-4.html

[9] SDSoC - 2015.4 Full Product Installation.
http://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

3. 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 includes evaluation bitstreams with three (8xSIMD) EdkDSP accelerators working in parallel with the HW-accelerated edge detection and motion detection algorithms for the Full HD HDMII-HDMIO video processing on the Trenz TE0715-04-30-3E module [1] located on the Trenz TE0701-06 carrier [3] with the FMC card [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 three (8xSIMD) EdkDSP floating point accelerators for the Trenz TE0715-04-30-3E module [1] on Trenz 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 TE0715-04-30-3E module [1] on Trenz TE0701-06 carrier board [3].

The evaluation package also includes compiled versions of this firmware in form of header files .h. These compiled firmware files can be used for initial test of the UTIA EdkDSP accelerators on the Trenz TE0715-04-30-3E module [1] on the Trenz 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.

4. 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 TE0715-04-30-3E module [1] located on the Trenz TE0701-06 carrier [3] with the FMC card [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 TE0715-04-30-3E module [1] located on the Trenz TE0701-06 carrier [3] with the FMC card [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 [5] includes the Vivado 2015.4 design projects which can be modified and recompiled by the Almarvi project [10] 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 XC7Z030-3E 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 TE0715-04-30-3E module [1] on Trenz 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 TE0715-04-30-3E module [1] on Trenz 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 TE0715-04-30-3E module [1on the Trenz 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.

5. Vivado projects with the release version of the (8xSIMD) EdkDSP IP

This release package includes **Vivado 2015.4 projects** for the Trenz TE0715-04-30-3E module [1] located on the Trenz TE0701-06 carrier [3] with the FMC card [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 TE0715-04-30-3E module [1] located on the Trenz TE0701-06 carrier [3] with the FMC card [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 evaluation version of the AXI-lite interface
bce_fp12_1x8_40	Release netlist of the evaluation version of the floating point data path
bce_fp12_1x8_30	Release netlist of the evaluation version of the floating point data path
bce_fp12_1x8_20	Release netlist of the evaluation version of the floating point data path
bce_fp12_1x8_10	Release netlist of the evaluation version 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 XC7Z030-3E 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 TE0715-04-30-3E module [1] on Trenz 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 TE0715-04-30-3E module [1] on Trenz 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 TE0715-04-30-3E module [1] on the Trenz 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.