Application Note

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

Python 1300 Sensor Video Processing in HW with EdkDSP 8xSIMD Accelerator for TE0720-03-2IF SoM on TE0701-05 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	31.07.2016	Jiří Kadlec	Evaluation package for Xilinx SDK 2015.4
2	05.08.2016	Jiří Kadlec	Demos solve horizontal borders. ARM code can be executed in parallel with HW data paths Number of HW designs is reduced to 4: md01, sh01, sh02 and sh03. Top level C code is identical for sh01, sh02 and sh03. Pointers to the parallel processed regions of a frame are defined in function img_process().

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

1.1 Objectives

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

- 3 edge detection video processing designs (sh01, sh02, sh03) with separate HW accelerated data paths.
  - These demos 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 effective, synchronised parallel execution with ARM C user code.

- 1 motion detection video processing.
  - This demonstrated the pipelined parallel execution of HW video processing accelerators.
  - HW accelerators work with fixed number of processed micro-lines (1024 micro-lines).

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

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

All demos are designed for the Trenz TE0701-05 platform [3] with industrial grade Zynq XC7Z020-2I device on System on Module TE0720-03-2I [1].

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

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

Created SW access functions controlling the HW accelerators have been exported from the Xilinx SDSoC 2015.4 projects to the Xilinx SDK 2015.4 [8] SW C projects as static .a libraries for standalone ARM Cortex A9 processor.

Main objectives of this application note:

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

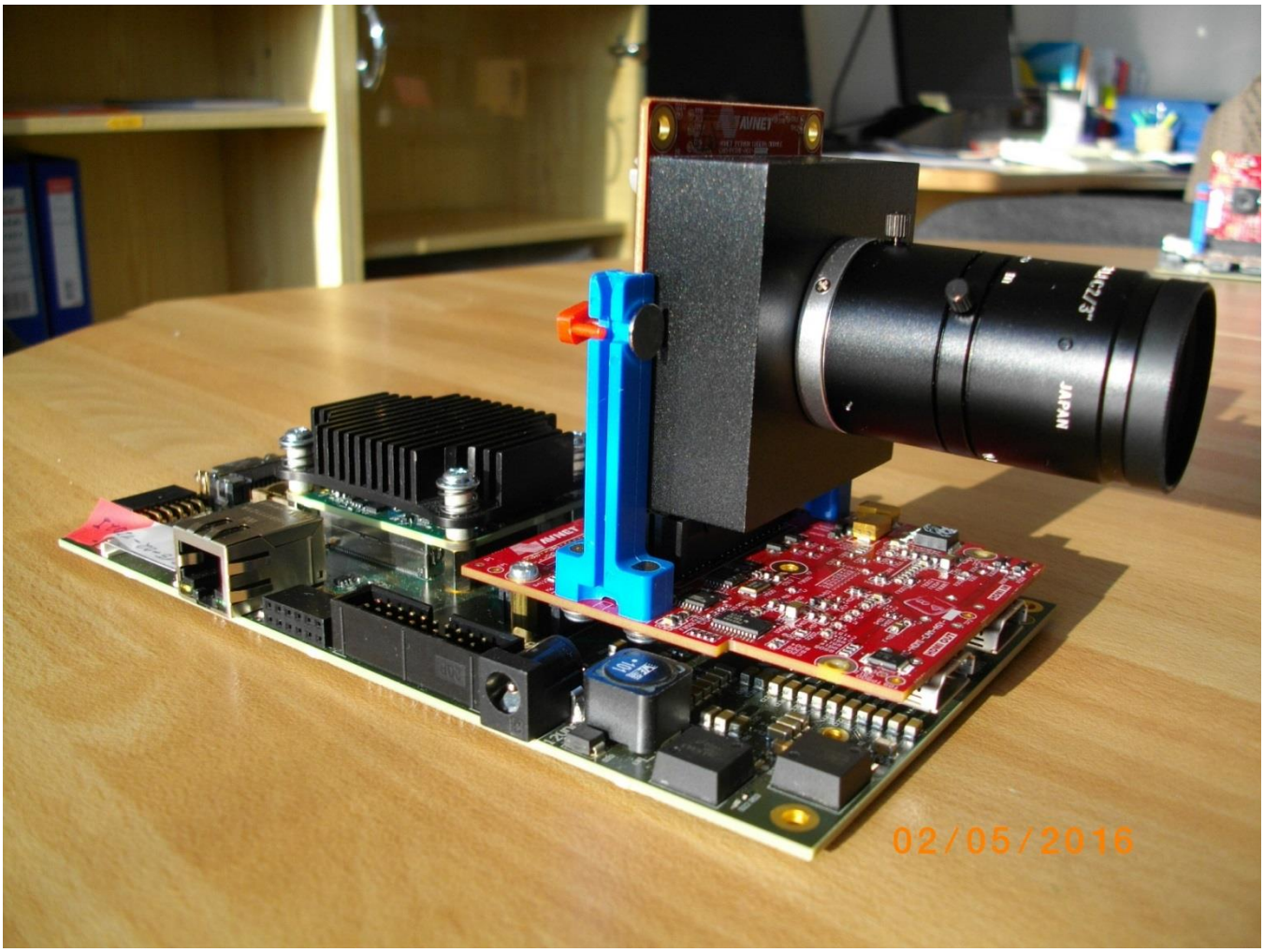


Figure 1: Python 1300 evaluation platform with video processing in HW and EdkDSP accelerator.

Common setup for demos:

- ARM Cortex A9 processor of Xilinx Zynq device XC7Z020-2I executes standalone C application programs performing initialisation and synchronisation of the HW accelerated video processing chains.
- Enclosed C programs can be modified by the user and recompiled in Xilinx SDK 2015.4.
- Python 1300 sensor with resolution 1280x1024p60 provides raw colour video data.
- Data are processed in HW into the YCrCb 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 HDMI display with fixed resolution 1280x1024p60.

1.2 Python 1300 platform with EdkDSP Accelerator

This application note describes HW platform performing integration of the runtime reprogrammable 8xSIMD EdkDSP floating point accelerator with edge detection and motion detection video processing of raw data from the Python 1300 colour video sensor with fixed resolution 1280x1024p60.

- The Xilinx Zynq device xc7z020-2I has two ARM Cortex A9 processors (operating at 666 MHz). Memory controller provides interface to DDR3 memory access ports. The Zynq device provides also the programmable logic area used for:

- UTIA EdkDSP (8xSIMD) floating point processor (operating at 120 MHz) connected to Xilinx MicroBlaze 32bit processor (operating at 100 MHz).
- Input chain of video processing IPs is connecting Python 1300 video sensor to input video frame buffers. The input video DMA (VDMA) controller is operating at 150 MHz.
- Area reserved for HLS HW accelerators and data movers defined in Xilinx SDSoC 2015.4 environment. These accelerators can be controlled from ARM Cortex A9 C programs compiled in SDK 2015.4 C projects. These HLS accelerators are operating at 150 MHz.
- Chain of output video processing IPs is connecting output frame buffers to the display connected by HDMI cable. The output VDMA controller is operating at 150 MHz.

- UTIA EdkDSP is 8xSIMD floating point accelerator reprogrammable in runtime by change of firmware of build in PicoBlaze6 8bit controller. This is serving as a scheduler of vector operations performed in the EdkDSP is 8xSIMD floating point processor data paths. This scheduler is programmed by simple C programs compiled by simple C compiler and assembler, respecting the minimal resources of the PicoBlaze6 controller.
- UTIA EdkDSP is 8xSIMD floating point accelerator is controlled by the 32bit MicroBlaze processor. The MicroBlaze processor is executing C programs from the DDR3 memory. It executes complex C algorithms. Algorithms can benefit from execution of selected operations effectively on the EdkDSP coprocessor connected to the MicroBlaze by local dual ported memories. MicroBlaze C programs can take benefit of overlap of data communication from DDR3 to the EdkDSP dual-ported memories with parallel computations in the EdkDSP accelerator.
- Platform includes also the video processing chain of IPs controlled by ARM Cortex A9 processor.
- ARM Cortex A9 processor of Xilinx Zynq is performing initialisation and synchronisation of the video processing chain. 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. ARM processor initiates the IP cores in the programmable logic (PL) part of the Zynq. It also initiates the Python 1300 video sensor and the video output to a monitor with fixed 1280x1024p60 resolution.

Details of the video processing video chain:

- Raw video data are provided by the Python 1300 video sensor with the resolution 1280x1024p60.
- Data are processed into the YCrCb 16 bit per pixel format and stored by Video DMA (VDMA) to input video frame buffers (VFBs) defined in the DDR3.
- HW DMA controller(s) send data from/to the VFBs to the processing accelerators. Clock is 150 MHz.

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

1. MicroBlaze with EdkDSP coprocessor is computing Floating point FIR filter (in parallel to the dedicated video processing accelerator chain).
2. MicroBlaze with EdkDSP coprocessor is computing Floating point LMS adaptive filter (in parallel to the dedicated video processing accelerator chain).
3. MicroBlaze is computing in SW (only with its Floating point unit) FIR or LMS filter (in parallel to the dedicated video processing accelerator chain) but EdkDSP accelerator is not used.
4. MicroBlaze and EdkDSP is not present in the PL logic and only the dedicated video processing accelerator chain is processing the video from the Python 1300 video sensor.

SW figures indicate the energy/pixel consumed by the complete system in case of computation in ARM. C/C++ code was compiled with -O3 optimisation (but without NEON) in the SDSoC 2015.4 environment [9]. No HLS accelerators present. The evaluation designs with HLS accelerators have been created from these C/C++ functions in SDSoC 2015.4 [9].

1.3 Introduction to the demos

Edge detection

The edge detection algorithm is producing B/W video stream. Edges in each frame are marked as white and remaining part of the figure is set as black.

The edges are detected by a Sobel filter. Each pixel is filtered by a 3x3 2D FIR filter. A nonlinear decision on the output of the filter provides decision if the pixel is part of an edge or not. All 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. Zynq PL resources and the accelerations reached for these HW designs are summarised in sections 1.3, 1.4 and 1.5.

Motion detection

The motion detection algorithm detects and performs visualisation of moving edges. The moving edges are identified by two Sobel filters performing FIR filtering (similar to the above described edge detection) on pixels with identical coordinates but from two subsequent video frames. A difference of these filtered results is computed a noise in that signal is filtered by a Median filter.

Resulting signal is used for the nonlinear binary decision if the analysed pixel is part of a moving edge or not. If the pixel is part of a moving edge, it is assigned red colour and merged with the original colour video signal. Resulting output video signal is unchanged, with the exception of red colour marked moving edges.

Demo **md01** provides accelerated HW computation with one parallel HW data path. HW demo 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. Zynq PL resources and accelerations reached for these HW designs are summarised in section 1.6.

Measurements of acceleration

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: EdkDSP accelerator with edge detection in single HLS accelerator

Zynq module:	TE0720-03-2IF (Trenz)
Carrier board:	TE0701-05 (Trenz)
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)
FMC card input:	Python 1300 video sensor (color 1280x1024p60)
FMC card output:	HDMI output to display (color 1280x1024p60)
Accelerator:	8xSIMD EdkDSP (120 MHz) with MicroBlaze (100 MHz)
Accelerator:	Edge detection single data path (150 MHz)
Platform:	C:\S\t20i2pm1\hio\
Demos (SDK 2015.4):	C:\VM07\t20i2pm1\sh01_rows_resize_25_to_100\ C:\VM07\t20i2pm1\sh01_rows_fixed_100\

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_07_29

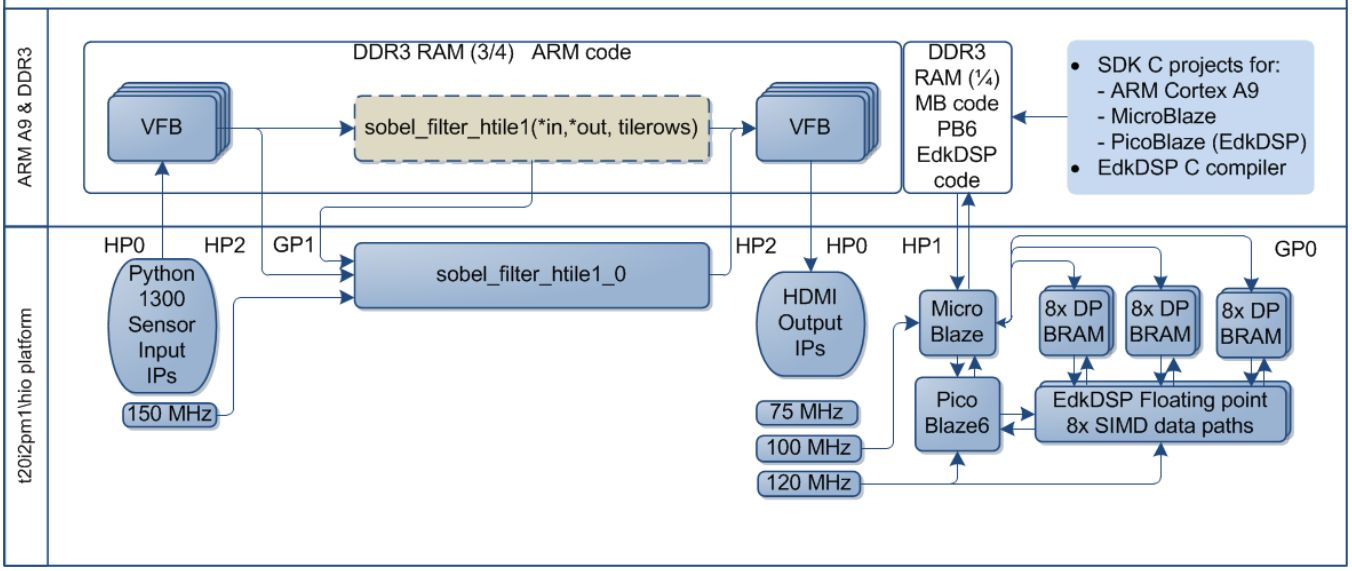
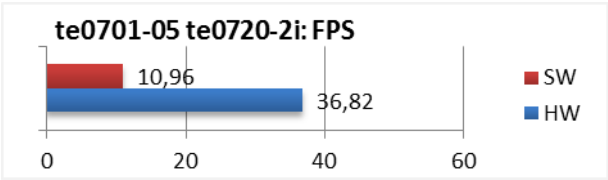
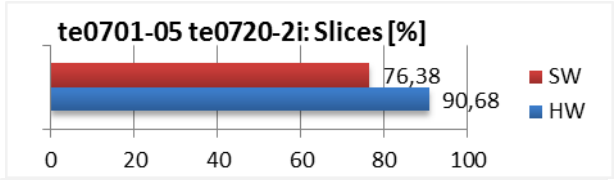
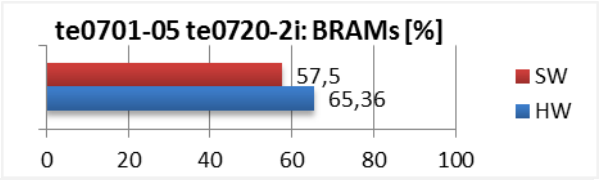


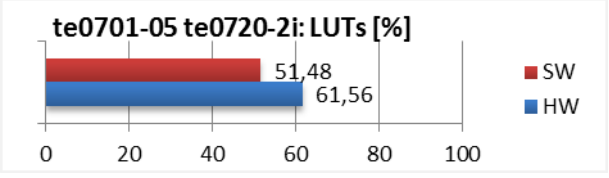
Figure 2: Project sh01 - Edge detection with single HW accelerator and EdkDSP accelerator.

Energy per pixel (nJ/p = nano Joule/pixel) **Reduced:**
EdkDSP FIR SW: 451.08 nJ/p HW: 135.02 nJ/p **3.34 x**
EdkDSP LMS SW: 449.41 nJ/p HW: 135.52 nJ/p **3.32 x**
Filter by MB SW: 439.39 nJ/p HW: 131.54 nJ/p **3.34 x**
Without MB SW: 416.83 nJ/p HW: 124.08 nJ/p **3.36 x**









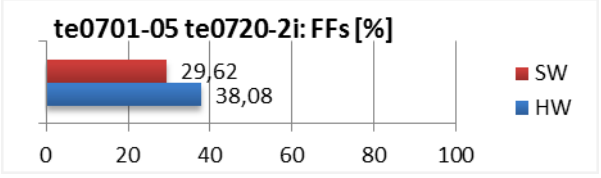


Figure 3: Project sh01 - Energy per frame reduction and used HW resources.

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

Zynq module:	TE0720-03-2IF (Trenz)
Carrier board:	TE0701-05 (Trenz)
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)
FMC card input:	Python 1300 video sensor (color 1280x1024p60)
FMC card output:	HDMI output to display (color 1280x1024p60)
Accelerator:	8xSIMD EdkDSP (120 MHz) with MicroBlaze (100 MHz)
Accelerator:	Edge detection two data paths (150 MHz)
Platform:	C:\S\t20i2pm1\hio\
Demos (SDK 2015.4):	C:\VM07\t20i2pm1\sh02_rows_resize_25_to_100\ C:\VM07\t20i2pm1\sh02_rows_fixed_100\

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_07_29

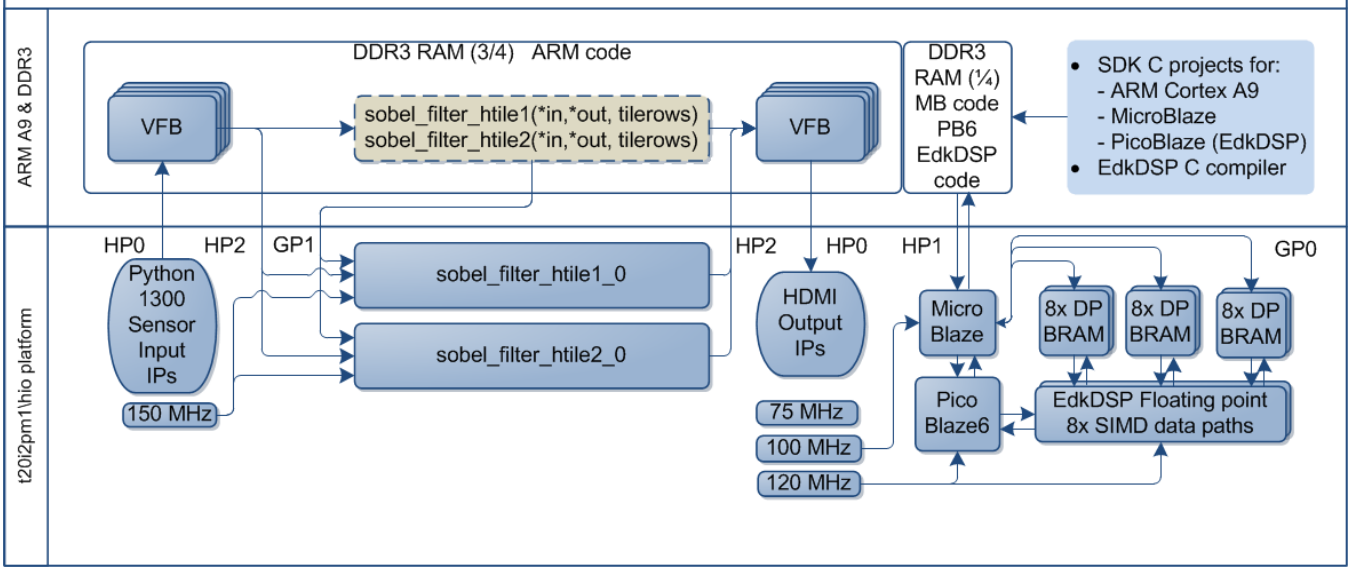


Figure 4: Project sh02 - Edge detection with two HW accelerators and EdkDSP accelerator.

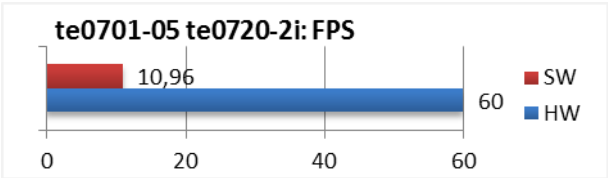
Energy per pixel (nJ/p = nano Joule/pixel) Reduced:

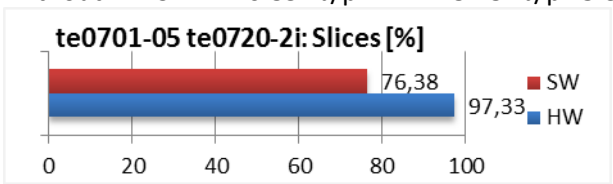
EdkDSP FIR SW: 451.08 nJ/p HW: 84.69 nJ/p **5.33 x**

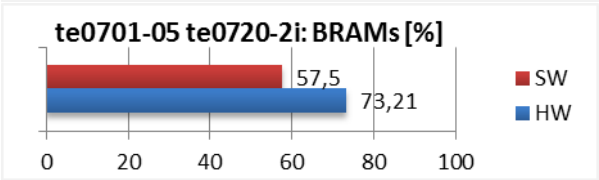
EdkDSP LMS SW: 449.41 nJ/p HW: 84.38 nJ/p **5.33 x**

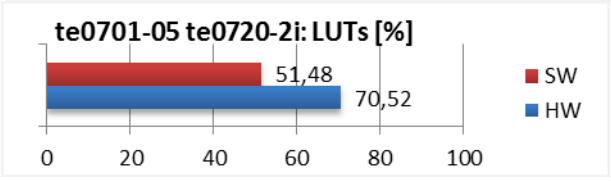
Filter by MB SW: 439.39 nJ/p HW: 82.55 nJ/p **5.32 x**

Without MB SW: 416.83 nJ/p HW: 78.28 nJ/p **5.32 x**









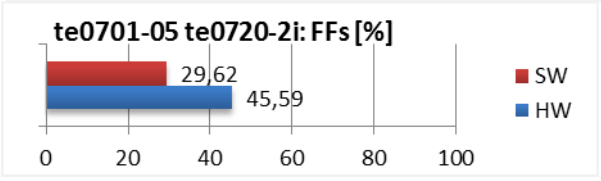


Figure 5: Project sh02 - Energy per frame reduction and used HW resources.

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

Zynq module:	TE0720-03-2IF (Trenz)
Carrier board:	TE0701-05 (Trenz)
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)
FMC card input:	Python 1300 video sensor (color 1280x1024p60)
FMC card output:	HDMI output to display (color 1280x1024p60)
Accelerator:	8xSIMD EdkDSP (120 MHz) with MicroBlaze (100 MHz)
Accelerator:	Edge detection three data paths (150 MHz)
Platform:	C:\S\t20i2pm1\hio\
Demos (SDK 2015.4):	C:\VM07\t20i2pm1\sh03_rows_resize_25_to_100\ C:\VM07\t20i2pm1\sh03_rows_fixed_100\

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_07_29

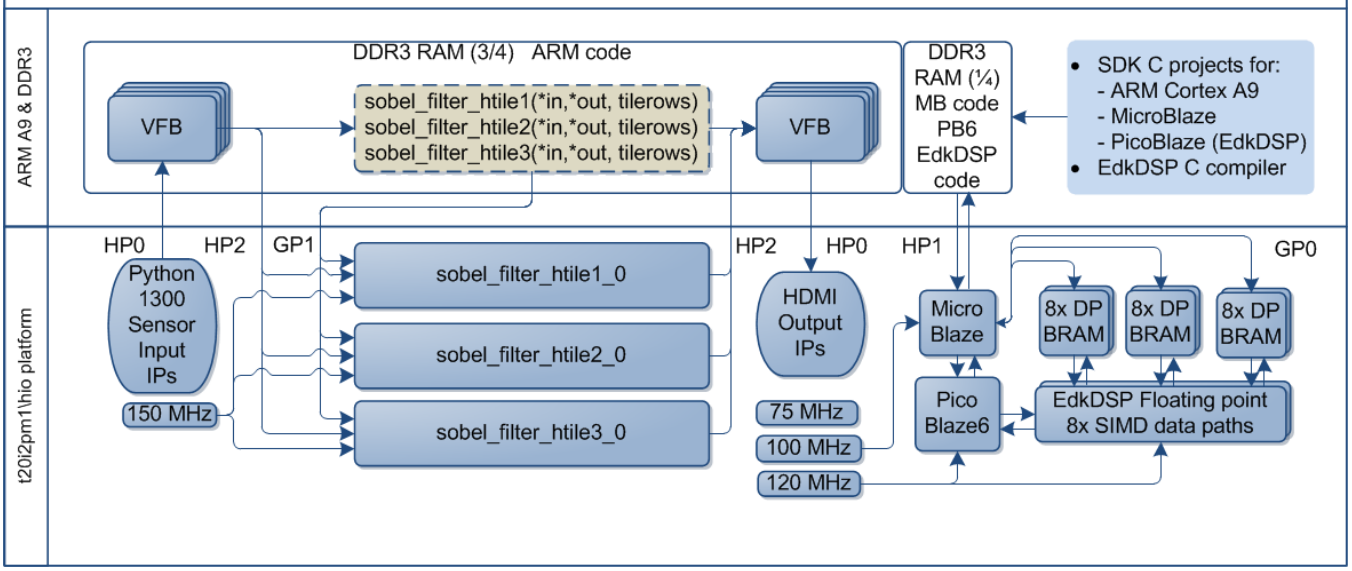
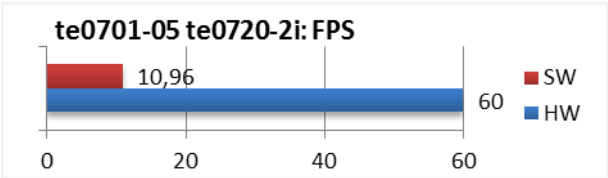
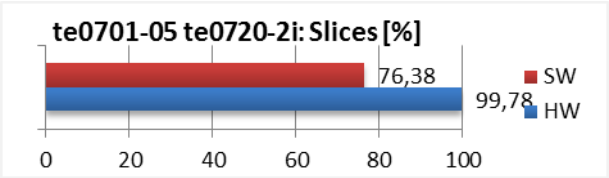
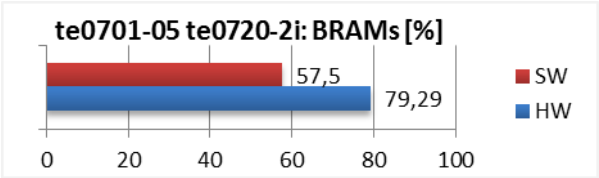


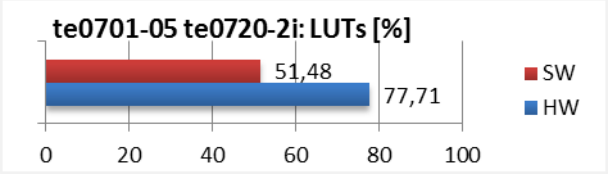
Figure 6: Project sh03 - Edge detection with three HW accelerators and EdkDSP accelerator.

Energy per pixel (nJ/p = nano Joule/pixel)			**Reduced:**
EdkDSP FIR	SW: 451.08 nJ/p	HW: 85.14 nJ/p	**5.30 x**
EdkDSP LMS	SW: 449.41 nJ/p	HW: 84.84 nJ/p	**5.30 x**
Filter by MB	SW: 439.39 nJ/p	HW: 83.01 nJ/p	**5.29 x**
Without MB	SW: 416.83 nJ/p	HW: 79.04 nJ/p	**5.27 x**









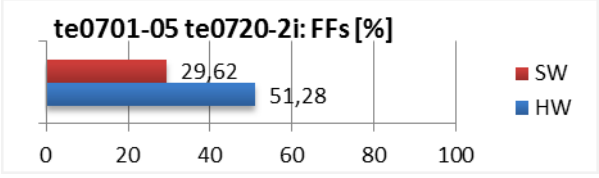


Figure 7: Project sh03 - Energy per frame reduction and used HW resources.

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

Zynq module:	TE0720-03-2IF (Trenz)	1 = pad();
Carrier board:	TE0701-05 (Trenz)	2 = sobel_filter_pass();
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)	3 = sobel_filter();
FMC card input:	Python 1300 video sensor (color 1280x1024p60)	4 = diff_image();
FMC card output:	HDMI output to display (color 1280x1024p60)	5 = median_char_filter_pass();
Accelerator:	8xSIMD EdkDSP (120MHz) with MicroBlaze (100MHz)	6 = combo_image();
Accelerator:	Motion detection single data path (150 MHz)	7 = ext();
Platform:	C:\S\t20i2pm1\hio\	
Demos(SDK 2015.4):	C:\VM07\t20i2pm1\md01_rows_fixed_100\	

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_07_29

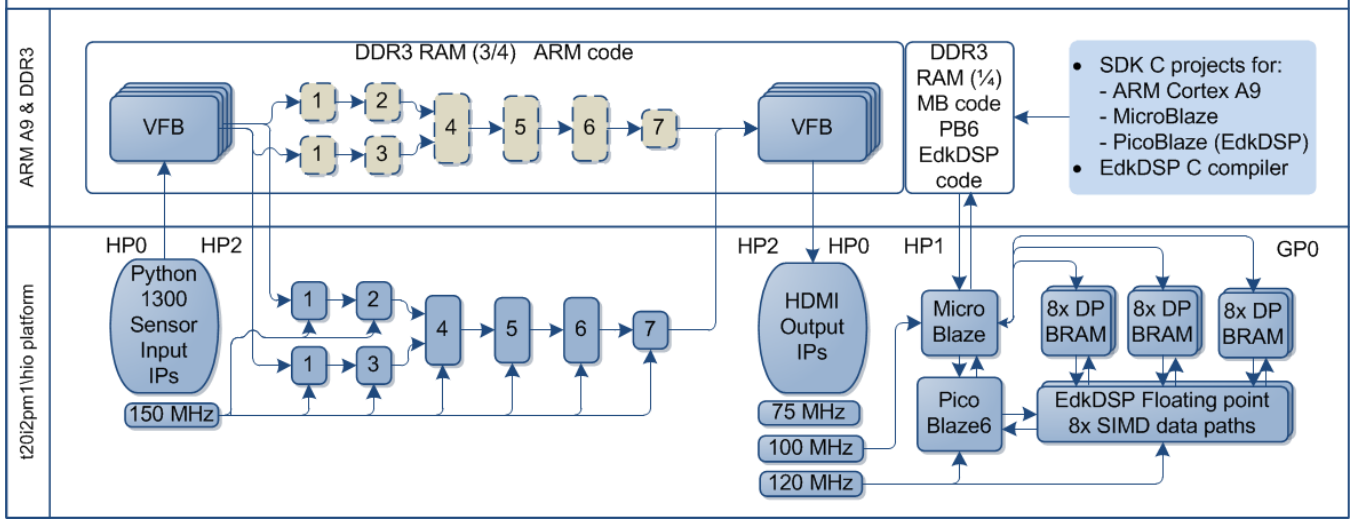


Figure 8: Project md01 - Motion detection with single HW accelerator and EdkDSP accelerator.

Energy per pixel (nJ/p = nano Joule/pixel) **Reduced:**
EdkDSP FIR SW: 2706.8 nJ/p HW: 134.4 nJ/p **20.14 x**
EdkDSP LMS SW: 2696.8 nJ/p HW: 134.0 nJ/p **20.13 x**
Filter by MB SW: 2637.1 nJ/p HW: 131.1 nJ/p **20.12 x**
Without MB SW: 2527.7 nJ/p HW: 124.1 nJ/p **20.37 x**

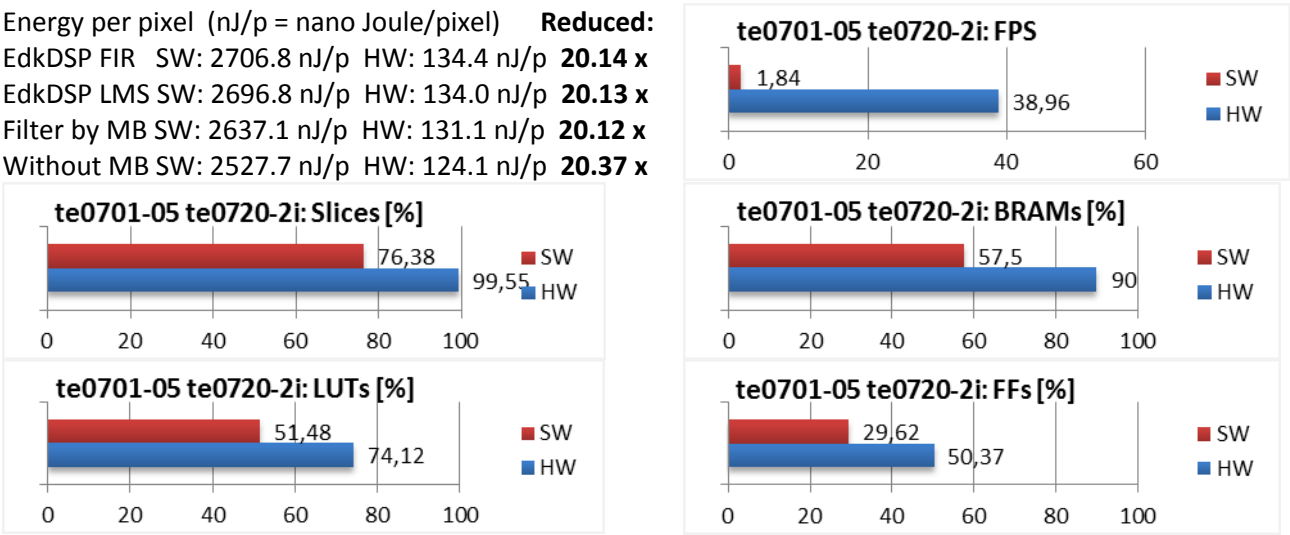


Figure 9: Project md01 - Energy per frame reduction and HW resources.

2. Installation of 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\t20i2pm2_V54_IMPORT

Create empty directory for Xilinx SDK workspace.
C:\VM_07\t20i2pm2

Start Xilinx SDK 2015.4 and select the directory for the SDK 2015.4 workspace. See Figure 10. Select **C:\VM_07\t20i2pm2**

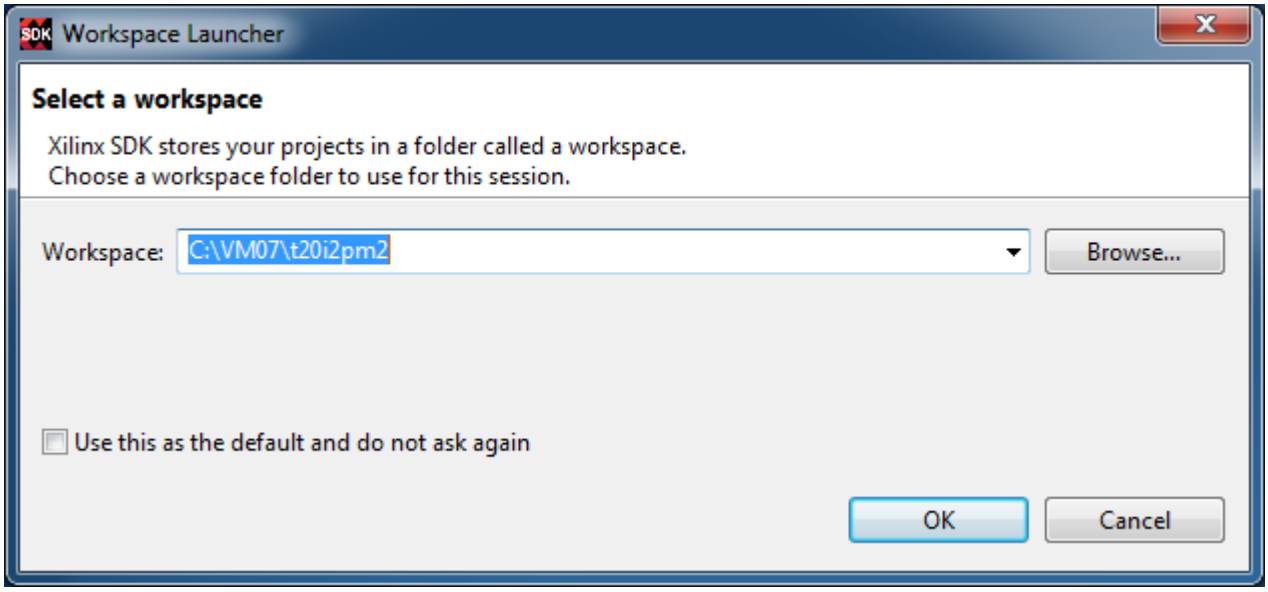


Figure 10: Select the SDK Workspace

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

File -> Import -> General -> Existing Projects into Workspace
Click on Next button. See Figure 11.

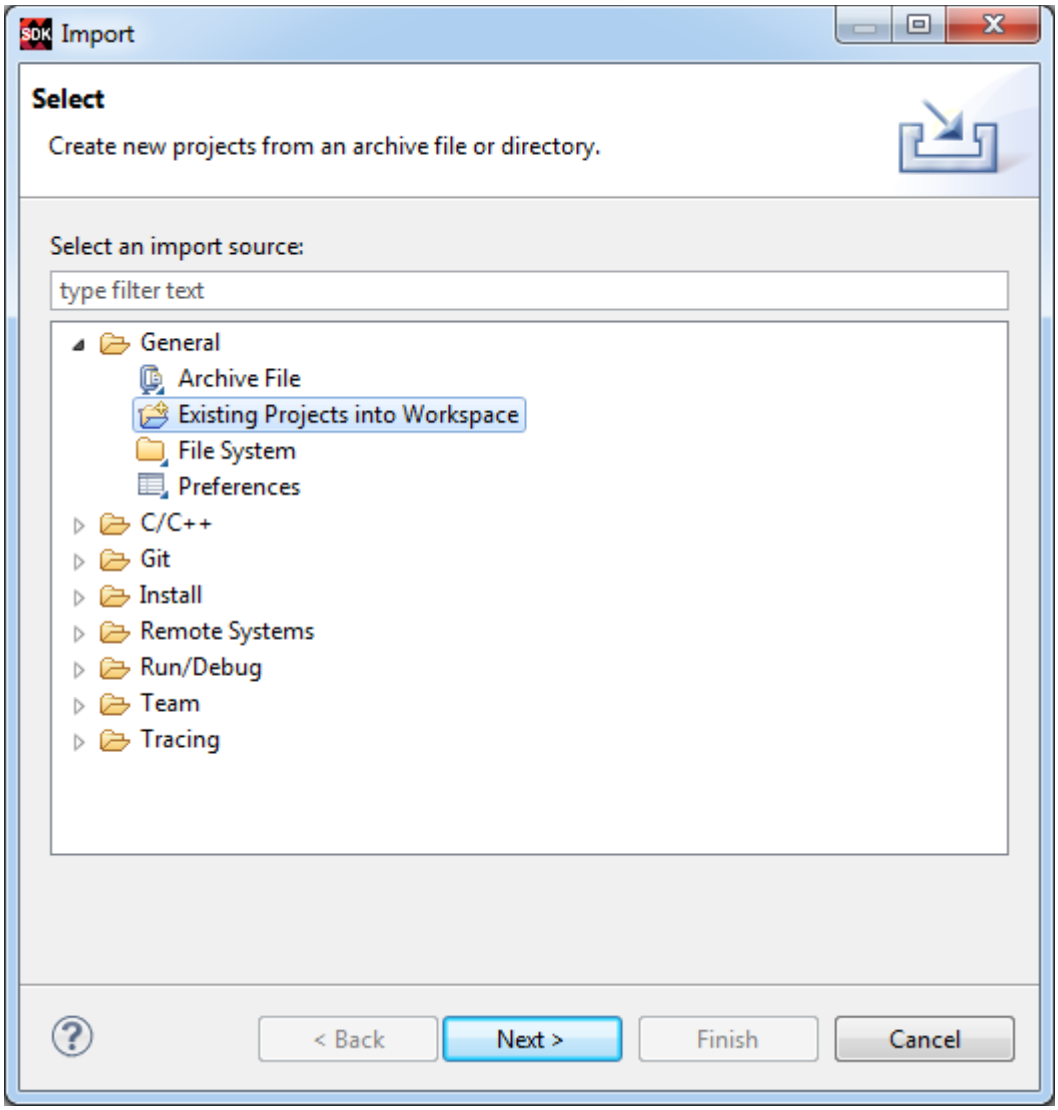


Figure 11: Import Existing Projects into Workspace

Type the directory with projects to be imported. See Figure 12.

C:\VM_07\t20i2pm2_V54_IMPORT

Set the "**Copy projects into workspace**" check box.
Click on Finish button. See Figure 12.

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.

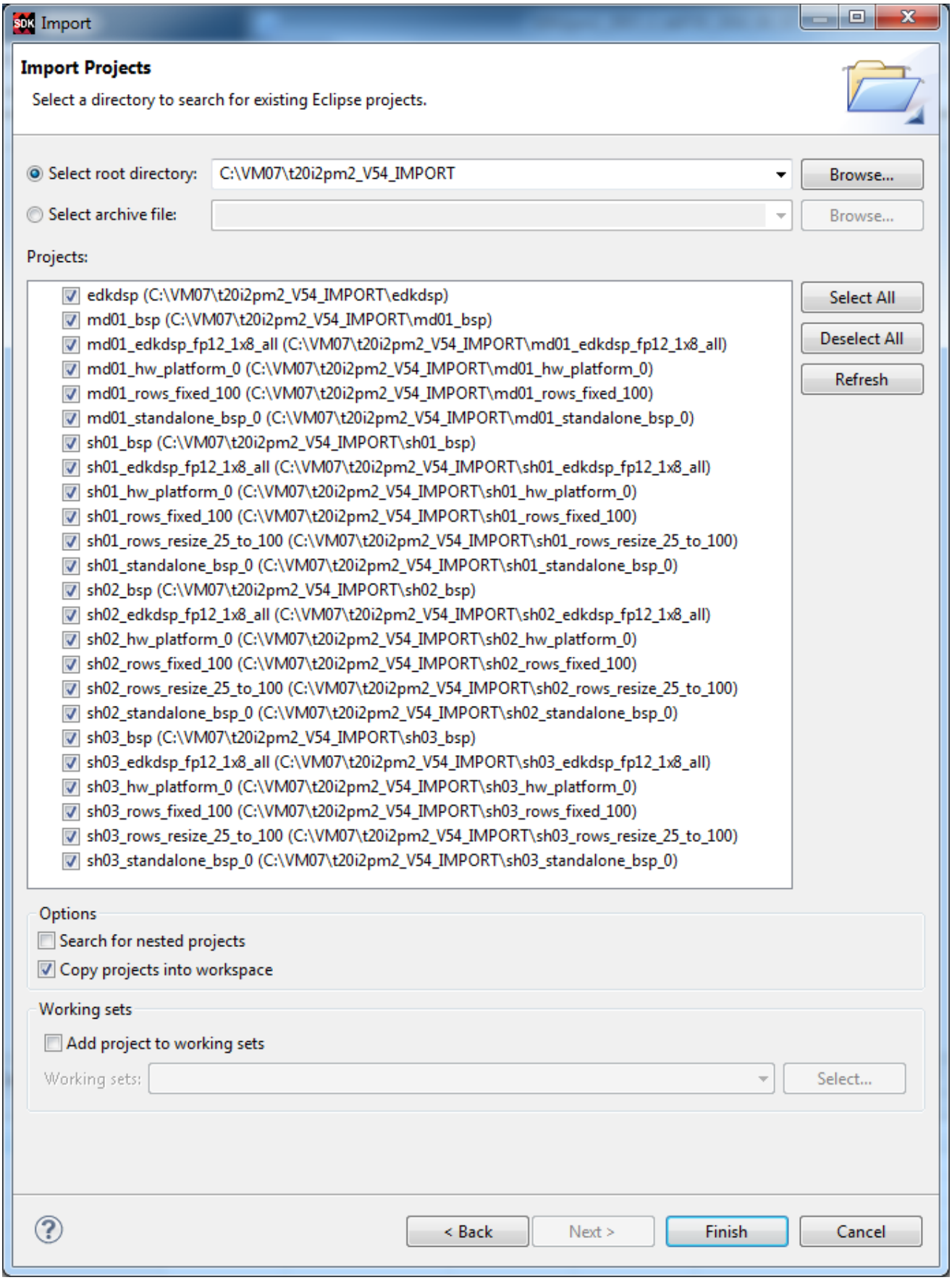


Figure 12: Select "Copy projects into workspace" and finish the import of all projects.

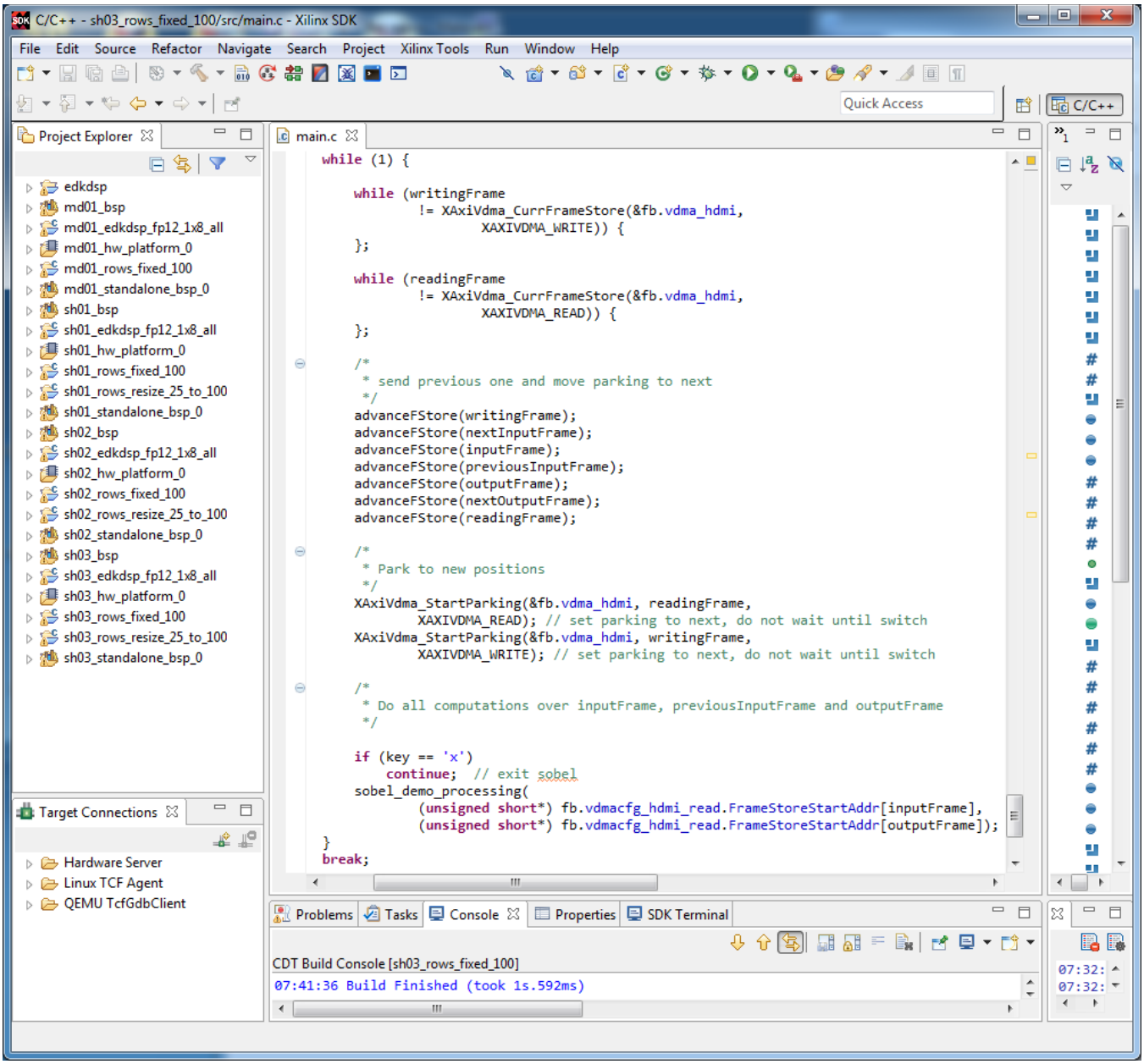


Figure 13: All projects are compiled in debug mode.

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

2.2 HW setup

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

TE0720-03-2IF; Part: XC7Z020-2CLG484I; 1 GByte DDR; Industrial Grade [1].
Heatsink for TE0720, spring-loaded embedded [2].
TE0701-05 Carrier Board for Trenz Electronic 7 Series [3].
AES-FMC-HDMI-CAM-G FMC card with HDMI I/O and CAM interface [4].
AES-CAM-ON-P1300C-G PYTHON-1300 color image sensor [5].
PmodRS232: Serial converter & interface[6].

HW Options:

TE0720-03-2IF	can be replaced by	**TE0720-02-2IF**	(Same Price, both boards from Trenz) [1].
TE0701-05	can be replaced by	**TE0701-04**	(Same Price, both boards from Trenz) [3].

Trenz TE0701-04 or TE0701-05 carriers require modifications to run the FMC Imageon carrier AES-FMC-HDMI-CAM-G with Zynq TE0720-03-2IF system on module. The modification is related to the swapped polarity of the differential clock signal for the FMC board. Evaluation HW systems with carriers TE0701-04 or TE0701-05 provided by UTIA have these modifications already done.

UTIA can implement these HW modifications for the original Trenz TE0701-04 and TE0701-05 carriers. This requires written e-mail request to kadlec@utia.cas.cz . Request will be first confirmed by UTIA. The interested party has to cover the cost of shipment of the carrier board to/from UTIA. Modification can be done in 5 working days and it is offered free of charge.

2.3 Test demos

To test demos follow these steps:

- Insert the Python 1300 video sensor to the connector on the Imageon board.
- Connect HDMI (or DVI) monitor by HDMI cable to the HDMI OUT on the Imageon FMC card.
- Switch the monitor ON.
- Connect the carrier board by USB-to-microUSB cable to PC to support JTAG serial link and the standard serial terminal.
- Connect the PmodRS232 Serial converter & interface module to the carrier board as indicated in Figure 14. 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

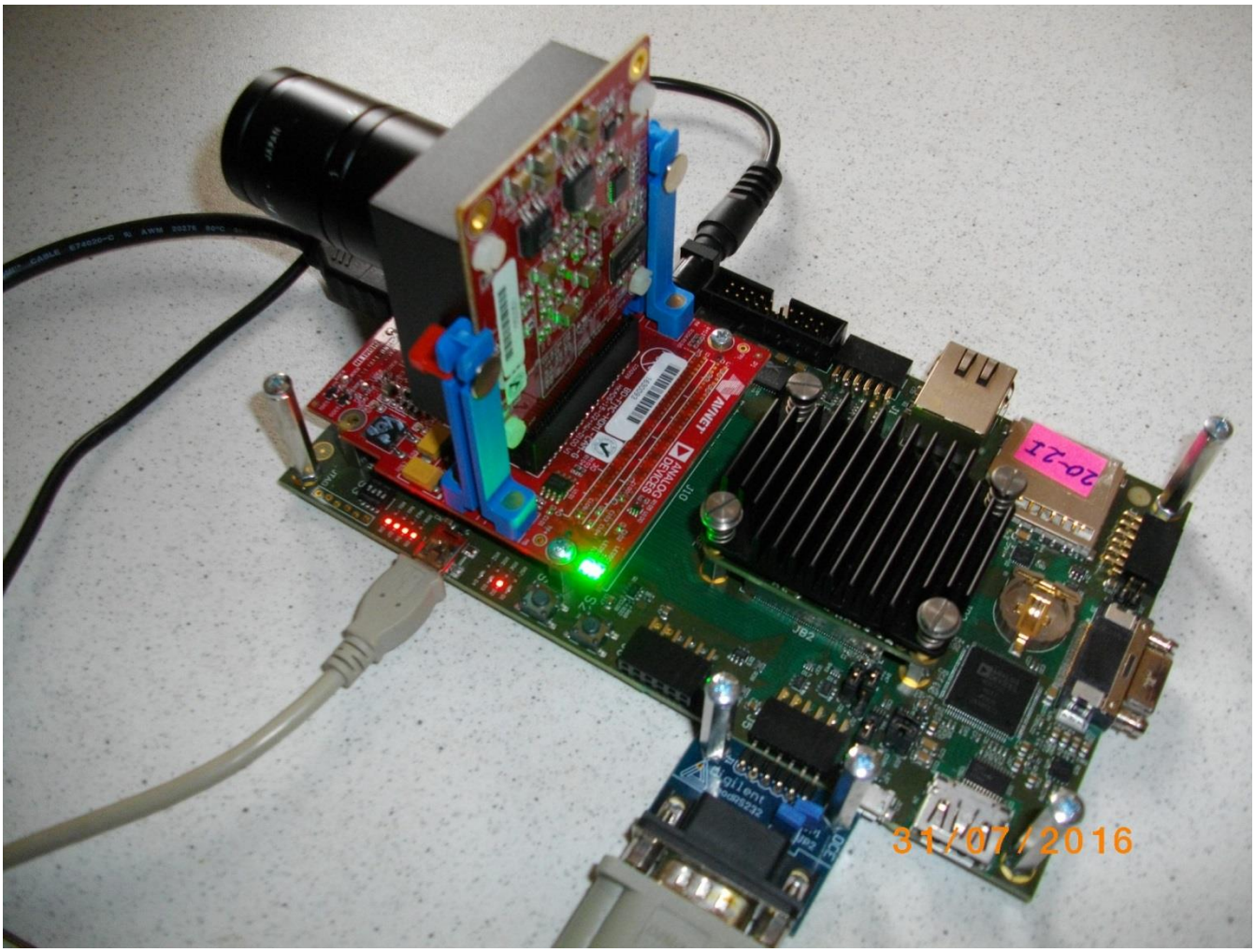


Figure 14: USB for ARM terminal and JTAG. RS232C Pmod for MicroBlaze

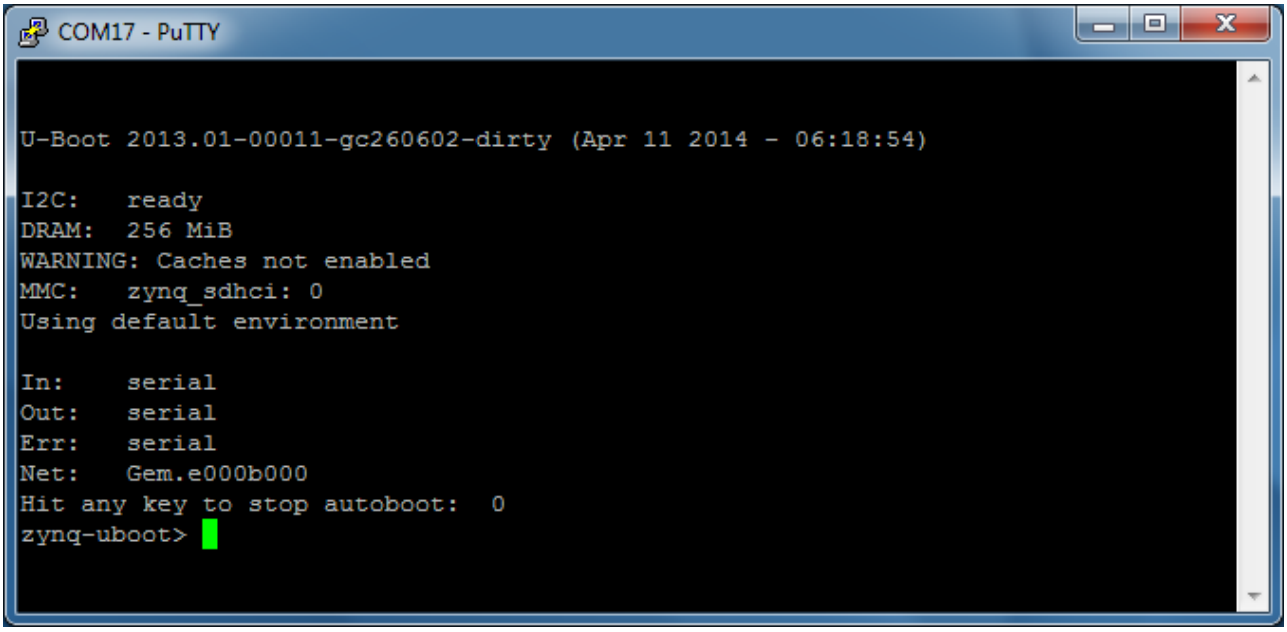


Figure 15: Serial console. Reset board and stop autoboot by any key.

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\t20i2pm2\sh03_hw_platform_0**

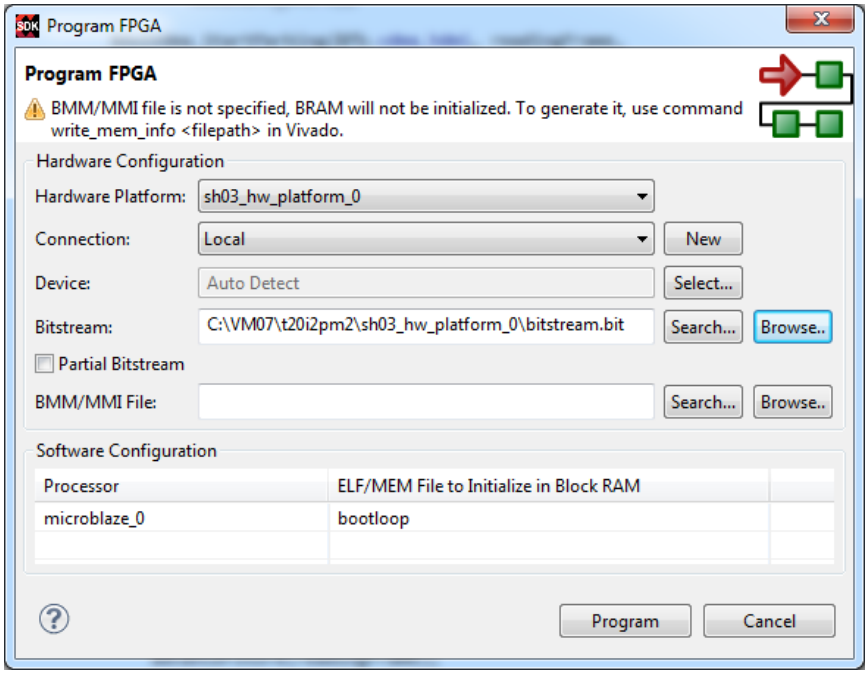


Figure 16: Download bitstream to the PL part of Zynq.

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

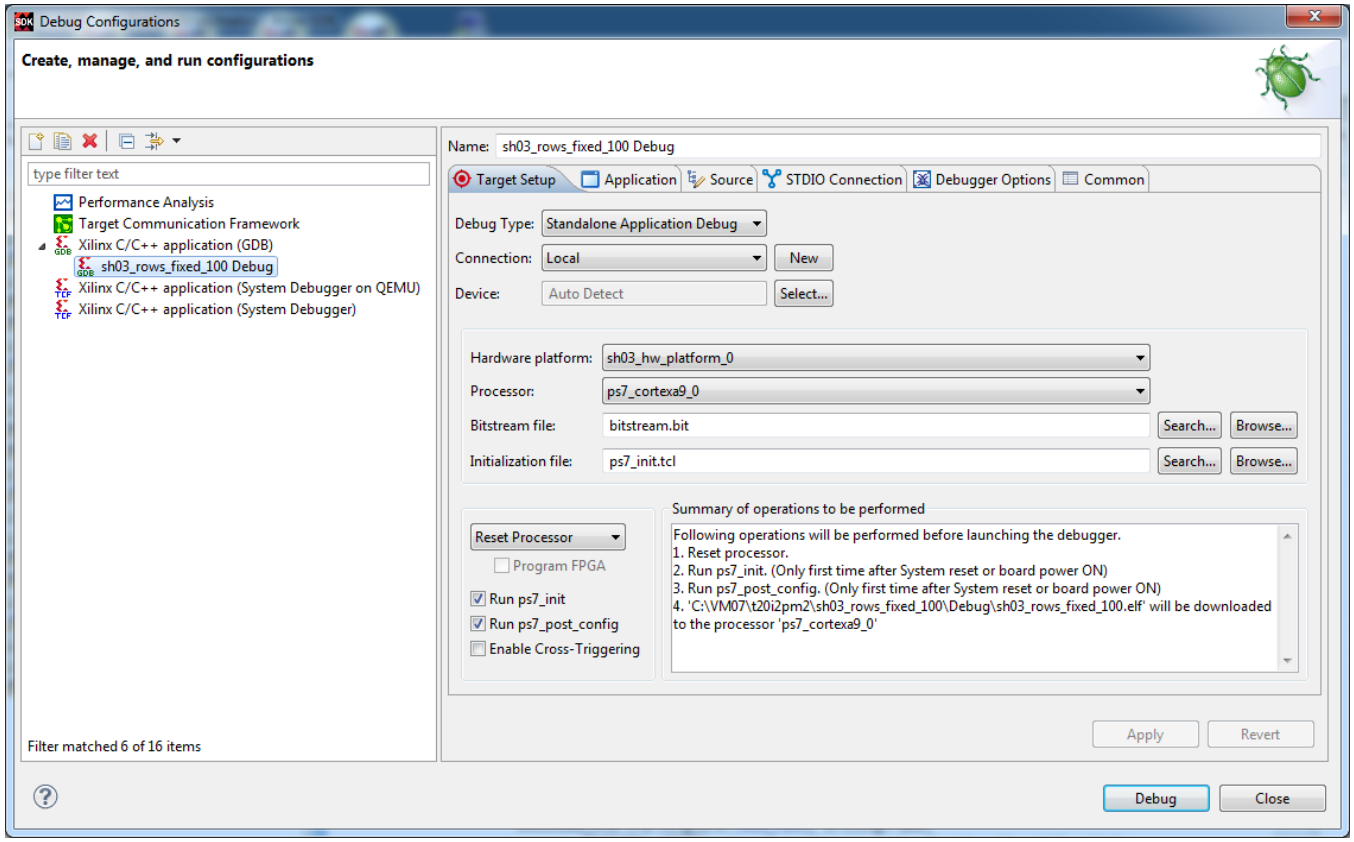


Figure 17: Select demo application for debug.

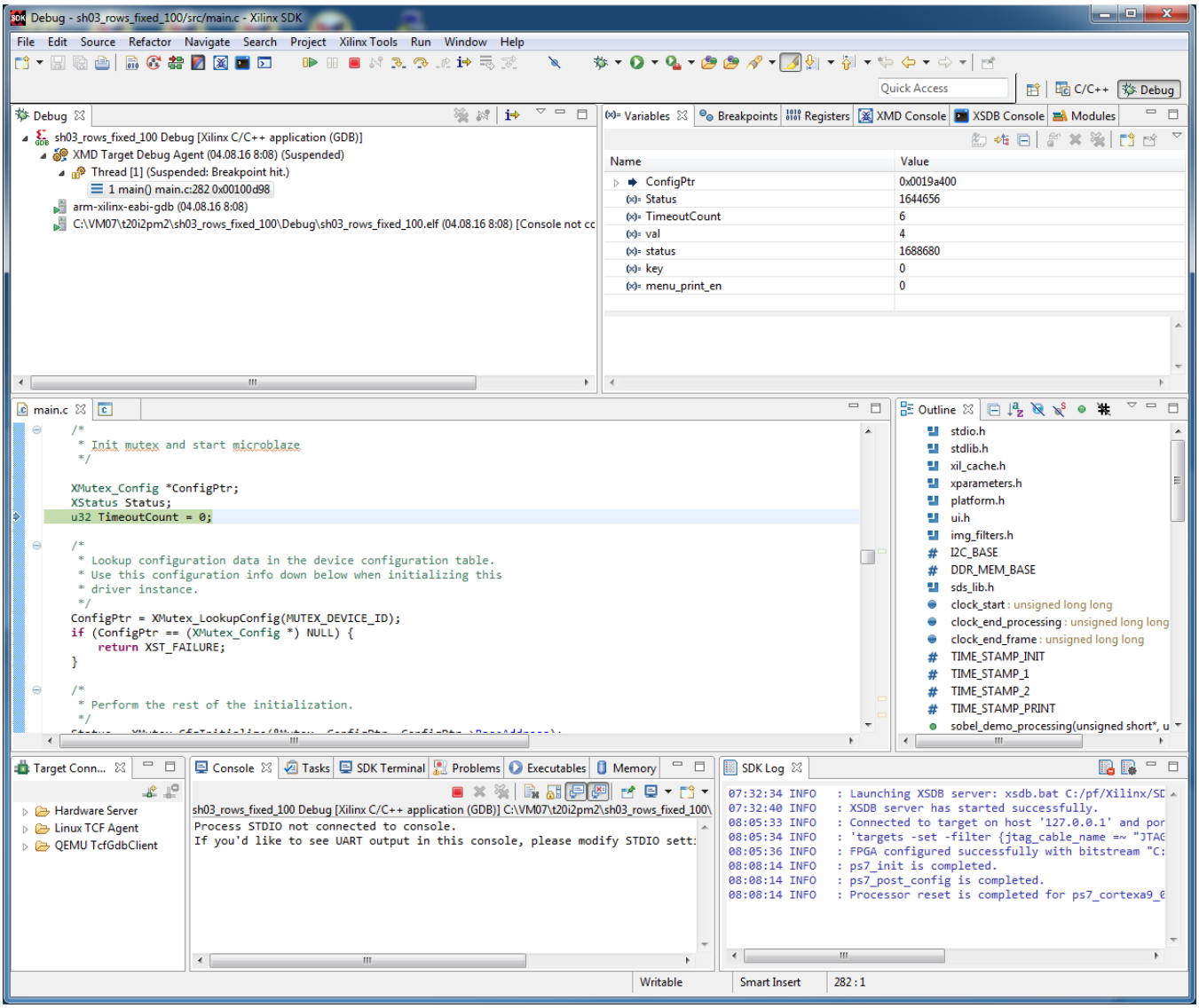


Figure 18: Demo app is booted to ARM and the debugger is waiting on the first executable line.

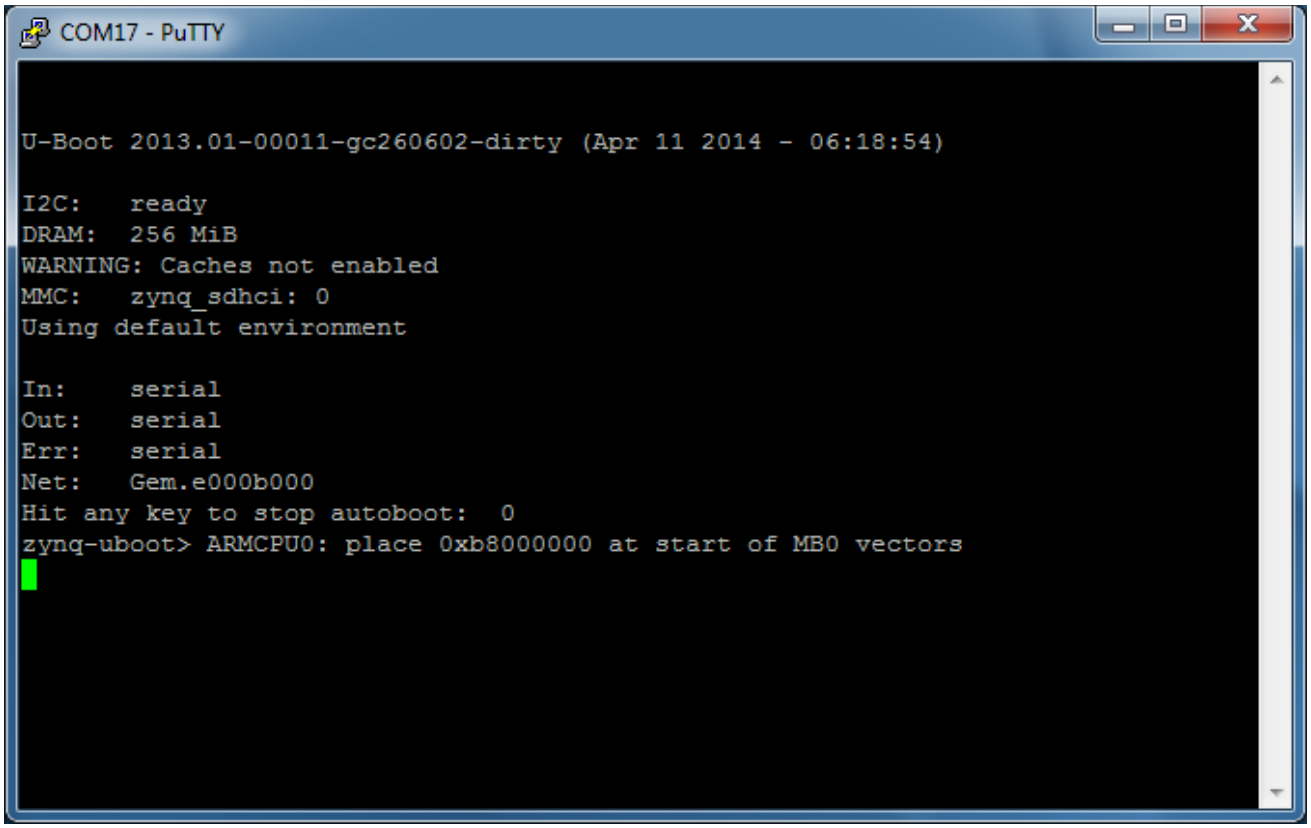


Figure 19: ARM is waiting on HW Mutex for the MicroBlaze start.

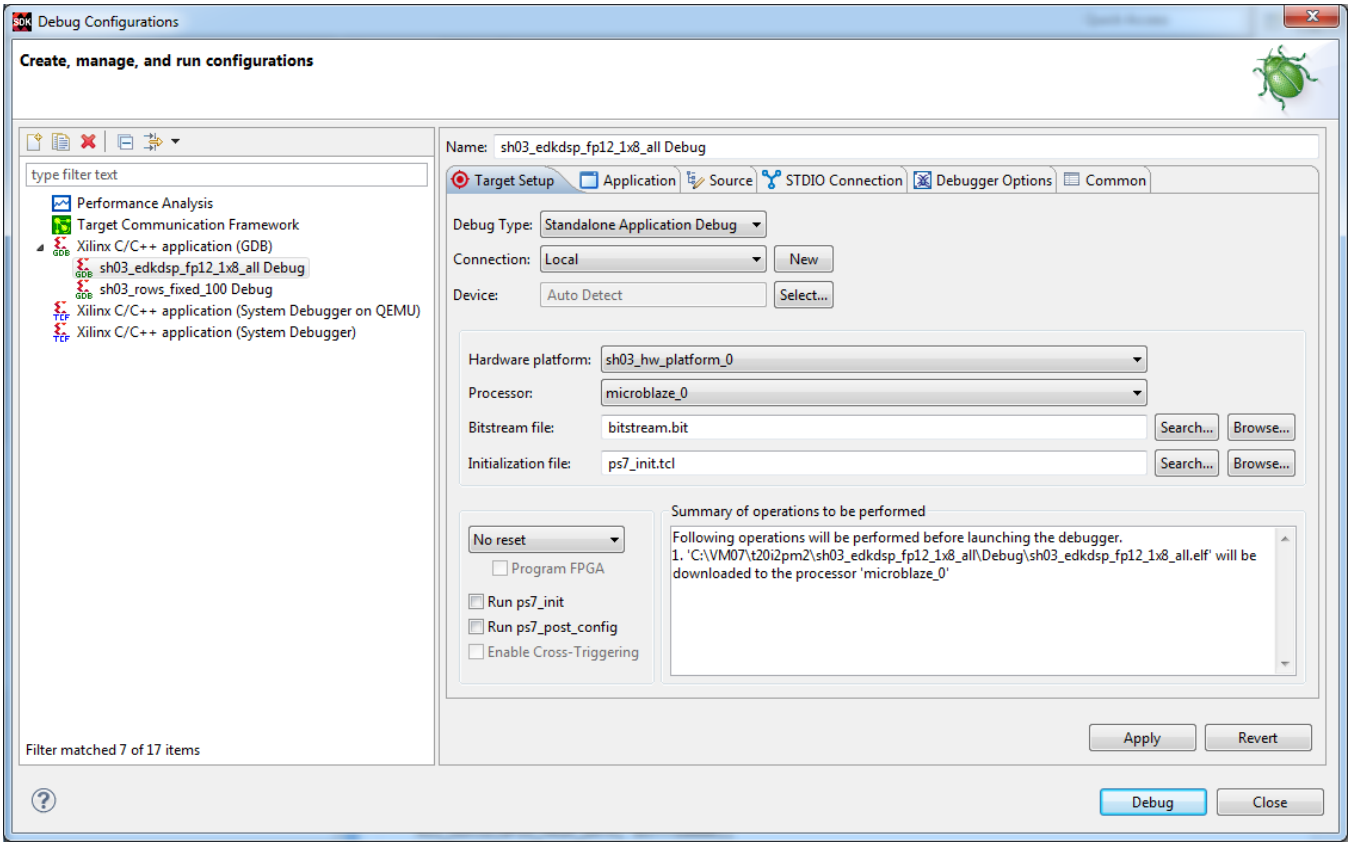


Figure 20: Select the MicroBlaze application (with the EdkDSP accelerator code) for debug.

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 **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 21.

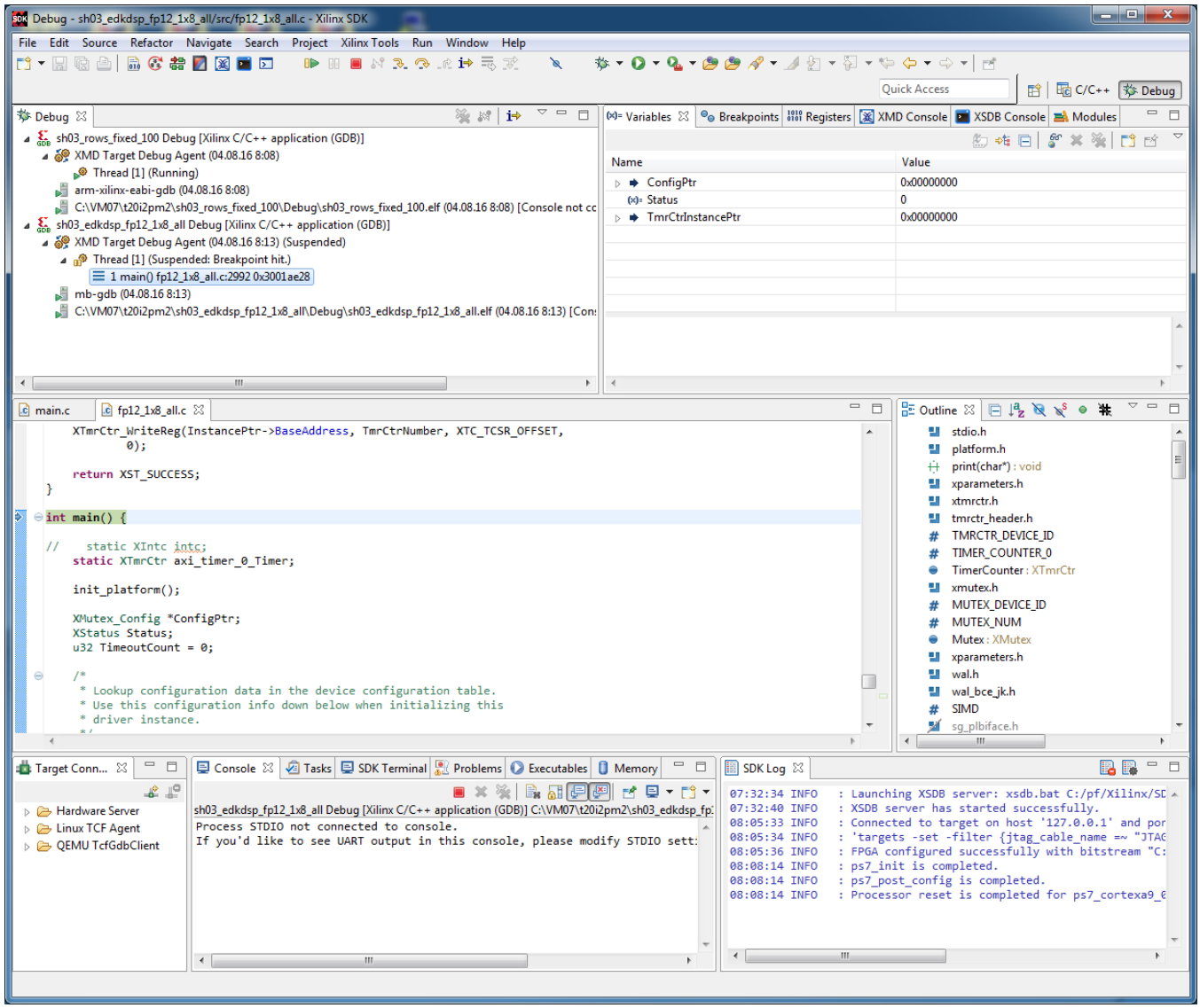


Figure 21: MicroBlaze application is loaded and debugger stops on the first instruction.

- ARM thread [1] is running.
- MicroBlaze thread [1] is currently suspended at breakpoint hit. See Figure 21.

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

ARM will initiate the Python 1300 video sensor and all 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 HLS IP cores in HW. Data are moved from video frame buffers to HW and back to output video frame buffers by HW data mover IPs. IPs are set-up by the ARM SW via the Axi-Lite interface.

Input HW data movers act as HW masters controlling the DMA engines moving data from DDR3 as input to the chain(s) of HLS IP cores. Output HW data movers act as HW masters controlling the DMA engines moving data from the output of chain(s) of HLS IP cores to DDR3 output video frame buffers. See Figure 22.

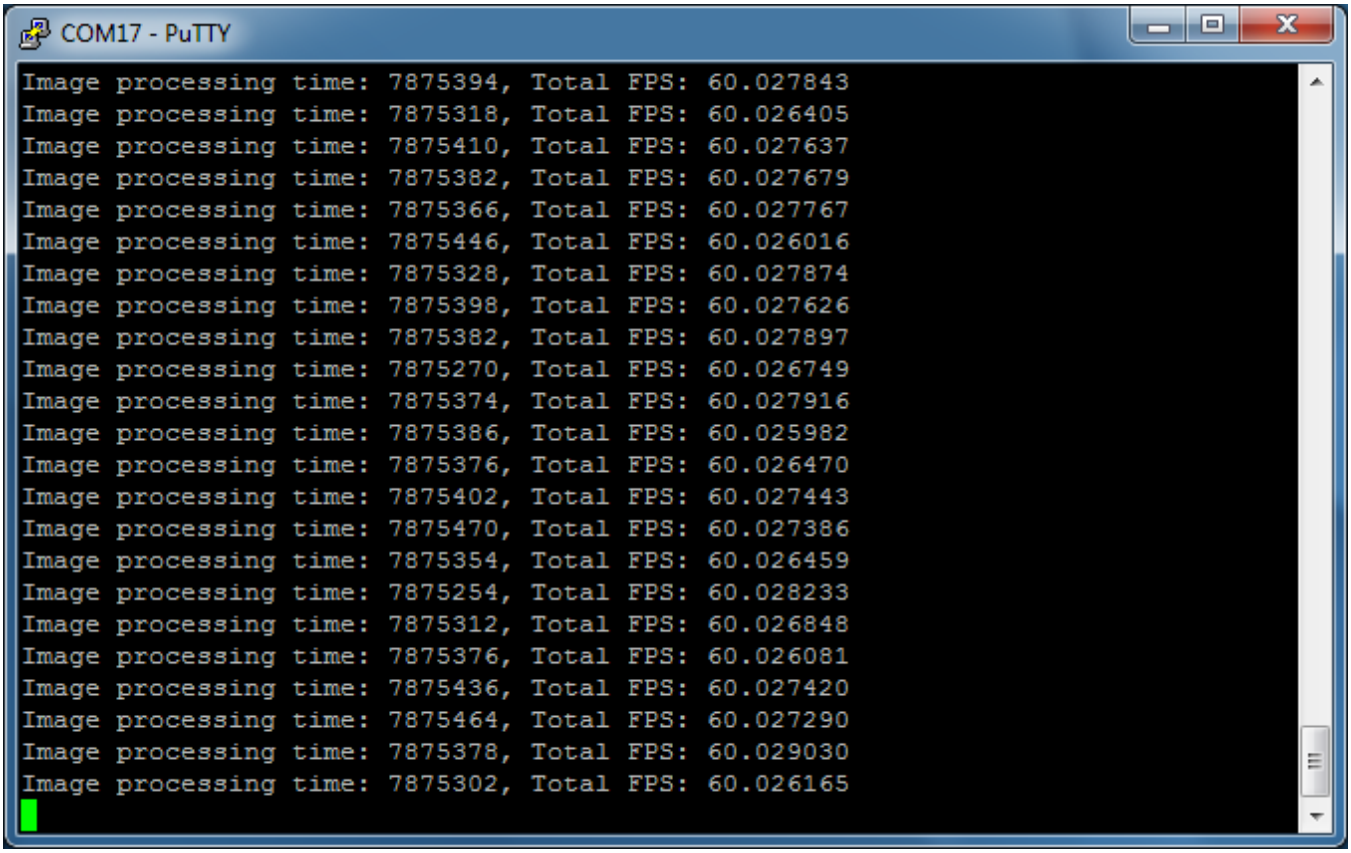


Figure 22: ARM is running. It indicates the number of frames per second.

The MicroBlaze processor executes in parallel program from DDR3 and communicated firmware and data to the (8xSIMD) EdkDSP floating point accelerator.

It is testing basic floating point operations and compares EdkDSP results with MicroBlaze floating point results.

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

The performance of the combination of MicroBlaze with EdkDSP accelerator is measured by HW timer instantiated as MicroBlaze AXI-Lite IP core. See Figure 23.

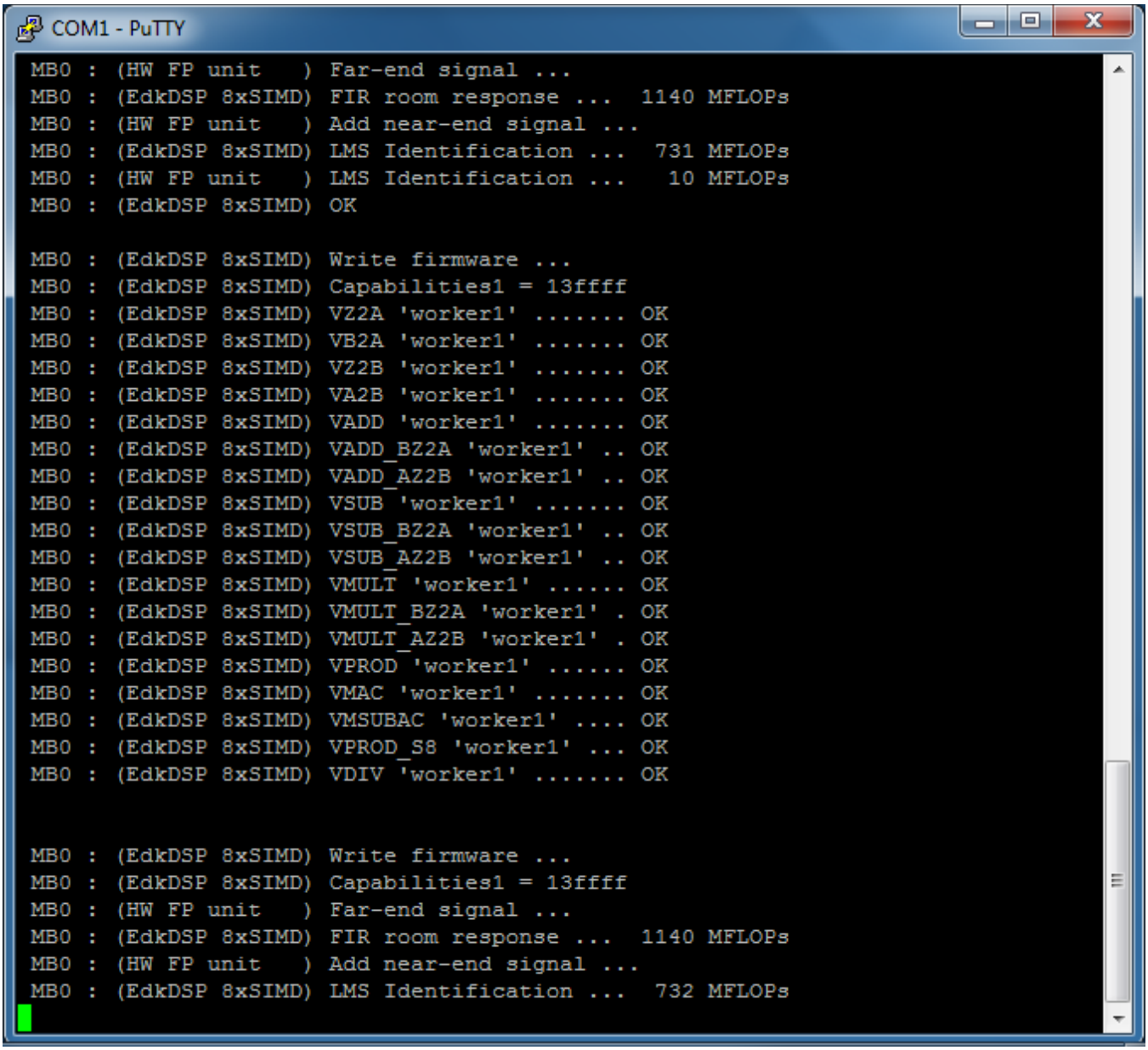


Figure 23: MicroBlaze is running. It indicates MFLOPs.

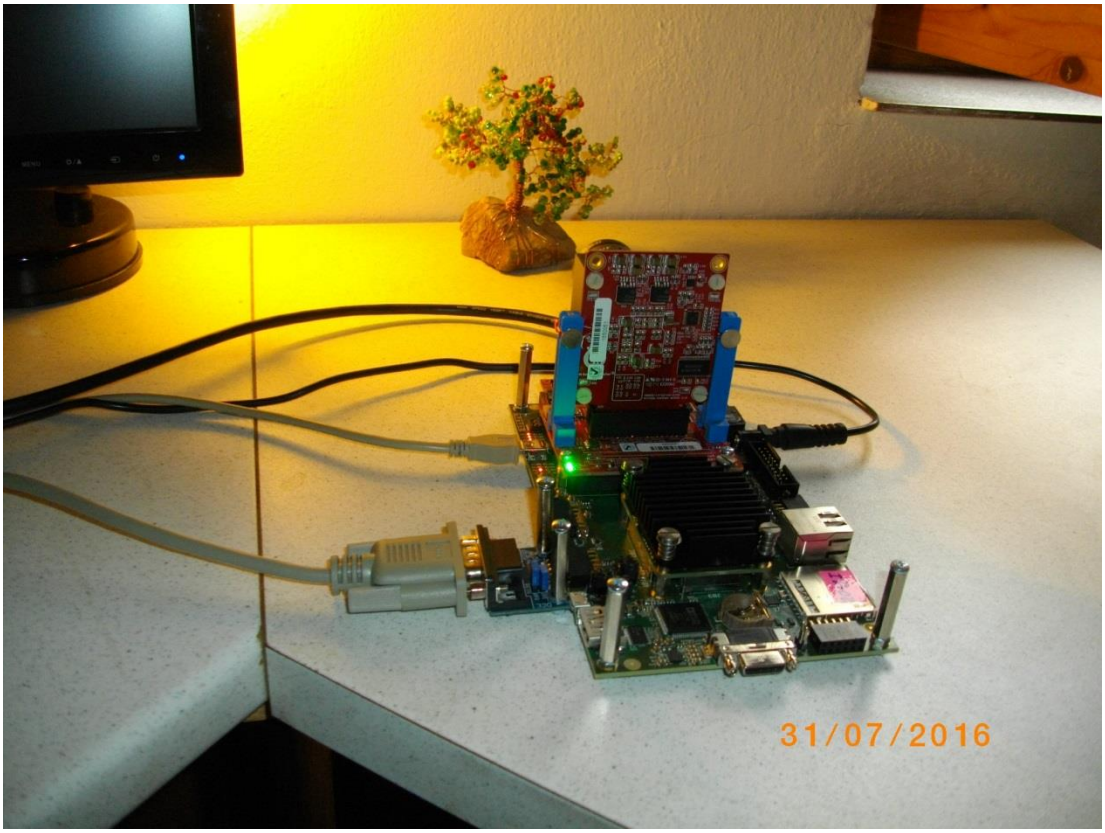


Figure 24: Accelerated edge detection Python 1300 sensor and Zynq with EdkDSP.

Figure 25: Edge detection (sh03 - Sobel filters, 3 variable HW paths) output on HDMI monitor.

- All evaluation demos can be also compiled into release versions with optimisation set to -O2 or -O3. These optimisations can be set for ARM and for MicroBlaze.
- Demos like sh01_rows_fixed_100 work on complete frame with single HW accelerator data path. Demos like sh01_rows_resize_25_to_100 work with identical HW, but the ARM SW scales dynamically the number of lines to be processed. This is scaling from ¼ of frame to the complete frame. Part of the frame which is not processed is automatically propagating the input video signal via the cyclic structure of 8 video frame buffers. The HW data movers are instructed about the number of lines to be processed. SW is writing this information to an AXI-lite configuration register of the data mover IP core.
- Demos sh02_rows_fixed_100 and sh02_rows_resize_25_to_100 work with 2 data paths.
- Demos sh03_rows_fixed_100 and sh03_rows_resize_25_to_100 work with 3 data paths.
- Demo md01_rows_fixed_100 works with one HW video processing chain with fixed set of processed lines.

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

This section describes synchronisation of ARM C code with parallel video processing HW accelerators.
Two cases or 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_fixed_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\t20i2pm2\sh03_rows_fixed_100\sobel\img_filters.c

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

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

#pragma SDS async(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);
}
```

Table 1: 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

correspond to the three HW video acceleration 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.

Internal synchronisation with parallel HW data paths

Table 2 is presenting 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\t20i2pm\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);
}
```

Table 2: Listing of ARM C function with fixed data width interfacing HW pipeline of accelerators.

2.5 EdkDSP C compiler

This section describes briefly how to use the UTIA EdkDSP C compiler. It cross-compiles (on PC) simple C programs for the PicoBlaze6 controller. The PicoBlaze6 controller acts as programmable finite state machine in the (8xSIMD) EdkDSP accelerator. It is setting the sequences of wide instructions for the 8xSIMD floating point data path of the EdkDSP accelerator.

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 [9] on Win 7 64 bit PC.

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/t20i2tm1/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/t20i2tm1/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**

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 **sh01_edkdsp_fp12_1x8_all**):

C:\VM_07\t20i2pm2\sh01_edkdsp_fp12_1x8_all\src

Recompile the MicroBlaze project "**sh01_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 the runtime change of firmware for computation of FIR and LMS filters in the EdkDSP accelerator in all included demos.

3. Conclusions

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

- Programmable logic part of the Zynq xc7z020-2I device is capable implement in parallel the UTIA (8xSIMD) EdkDSP floating point accelerator together with the HW accelerated video processing chain for the Python 1300 color sensor with fixed resolution 1280x1024p60.

- The total power consumption for the HW accelerated video processing (measured at the 12V DC power supply) is up to 6.86 W. This requires at least a passive heat sink.

- The total power consumption for the SW solution without HLS Video IPs (measured at the 12V DC power supply) is 6.48 W. This requires also passive heat sink.

- The energy per pixel is significantly reduced for the HW accelerated designs with HW accelerators. Energy per pixel reduction from 3x to 20x can be reached in comparison to ARM SW solution. The energy/pixel reduction 20x is reached for the HW motion detection running in parallel with the EdkDSP floating point accelerator.

- Main source of the energy per pixel saving is the increased frame rate of the video processing by HW accelerators.

- The combination of 32bit MicroBlaze with the (8xSIMD) EdkDSP floating point accelerator brings additional capability to compute in singleprecision floating point with performance in the range of cca.1 GFLOP/s (1.139 GFLOP/s in case of FIR filter) at the expense of relatively moderate increase of total power consumption of the system:

    - 6.34 W without MicroBlaze + (8xSIMD) EdkDSP - 0 GFLOP/s
    - 6.86 W with MicroBlaze + (8xSIMD) EdkDSP - 1.139 GFLOP/s (this is + 520 mW)

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

- The bill of material for the complete system [1]-[6] is approximately €1300.

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

Enclosed SDK 2015.4 projects provide space for the end-user to make SW adaptations and customisations of the final application in the C source code. The run-time re-programming of the 8xSIMD EdkDSP accelerator in C and ASM is also supported.

This user customisation is possible without detailed information about the used IP cores, Vivado 2015.4 projects and without the SDSoC 2015.4 board support package for the Python 1300/EdkDSP platform.

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

4. References

[1] TE0720-03-2IF; Part: XC7Z020-2CLG484I; 1 GByte DDR; Grade: Industrial; http://shop.trenz-electronic.de/en/TE0720-03-2IF-Xilinx-Zynq-module-XC7Z020-2CLG484I-ind.-temp.-range-1-Gbyte

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

[3] TE0701-05 Carrier Board for Trenz Electronic 7 Series; https://shop.trenz-electronic.de/en/TE0701-05-Carrier-Board-for-Trenz-Electronic-7-Series

[4] AES-FMC-HDMI-CAM-G Price: $250.00. http://products.avnet.com/shop/en/ema/3074457345623664802

[5] AES-CAM-ON-P1300C-G PYTHON-1300 color image sensor. https://products.avnet.com/shop/en/ema/development-kits/3074457345623664700

[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

5. Evaluation license

The **evaluation version of the package** can be downloaded from UTIA www pages [11] free of charge for evaluation of EdkDSP accelerator with HW accelerated edge detection and motion detection algorithms for the Python 1300 video sensor [5] on TE0720-03-2IF module [1] located on TE0701-05 carrier [3] with FMC card [4].

The evaluation package [11] includes SDK 2015.4 SW projects with C source code for 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 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 UTIA (8xSIMD) EdkDSP accelerators are compiled with HW limit on number of vector operations. The termination of the nonexclusive, non-transferable evaluation license is reported in advance by the demonstrator on the terminal. The evaluation package 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 TE0720-03-2IF Xilinx Zynq module [1] on Trenz TE701-05 Carrier Board board [3].

The evaluation 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 evaluation package includes demonstration firmware in C source code for the Xilinx PicoBlaze6 processor for the family of UTIA EdkDSP accelerators for the Xilinx TE0720-03-2IF module on TE0701-05 carrier board. The evaluation package also includes compiled versions of this firmware in form of header files .h. These compiled firmware files can be used for initial test of the UTIA EdkDSP accelerators on the Xilinx TE0720-03-2IF module on TE0701-05 carrier board 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 for the VMware Workstation 12 Player [7] free of charge.

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.