Application Note

Full HD HDMI In-Out HW-Accelerated Demos for Zynq System-on-Module TE0720-03-2IF and TE0701-05 Carrier Board

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	20.07.2016	Jiří Kadlec	Release for SDK 2015.4
1	21.07.2016	Jiří Kadlec	Fixed multiple typos

Acknowledgements:

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

Table of contents

Table of figures

1. Summary

1.1 Objectives

This application describes use of an evaluation package with 3 edge detection and 3 motion detection video processing designs on 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 [6]. Algorithms have been compiled by Xilinx SDSoC 2015.4 system level compiler (based on the Xilinx HLS compiler) to Vivado 2015.4 projects, and compiled by Vivado 2015.4 [5] to bitstreams. The SW access functions controlling the HW accelerators have been exported to the Xilinx SDK 2015.4 [5] SW projects as static .a libraries for standalone ARM Cortex A9 applications. This application note also describes 4 edge detection algorithms defined in the SDSoC 2015.4 in form, which enables in the SDK 2015.4 the parallel execution of predefined video processing HW paths with C user code on ARM.

Main objectives of this application note are:

- To demonstrate how to install, compile, modify and use the enclosed SW projects in the SDK 2015.4 [5].
- To demonstrate the HW accelerated video processing algorithms and the speedup against SW versions.
- To demonstrate parallel execution of predefined video processing HW paths with C user code on ARM.

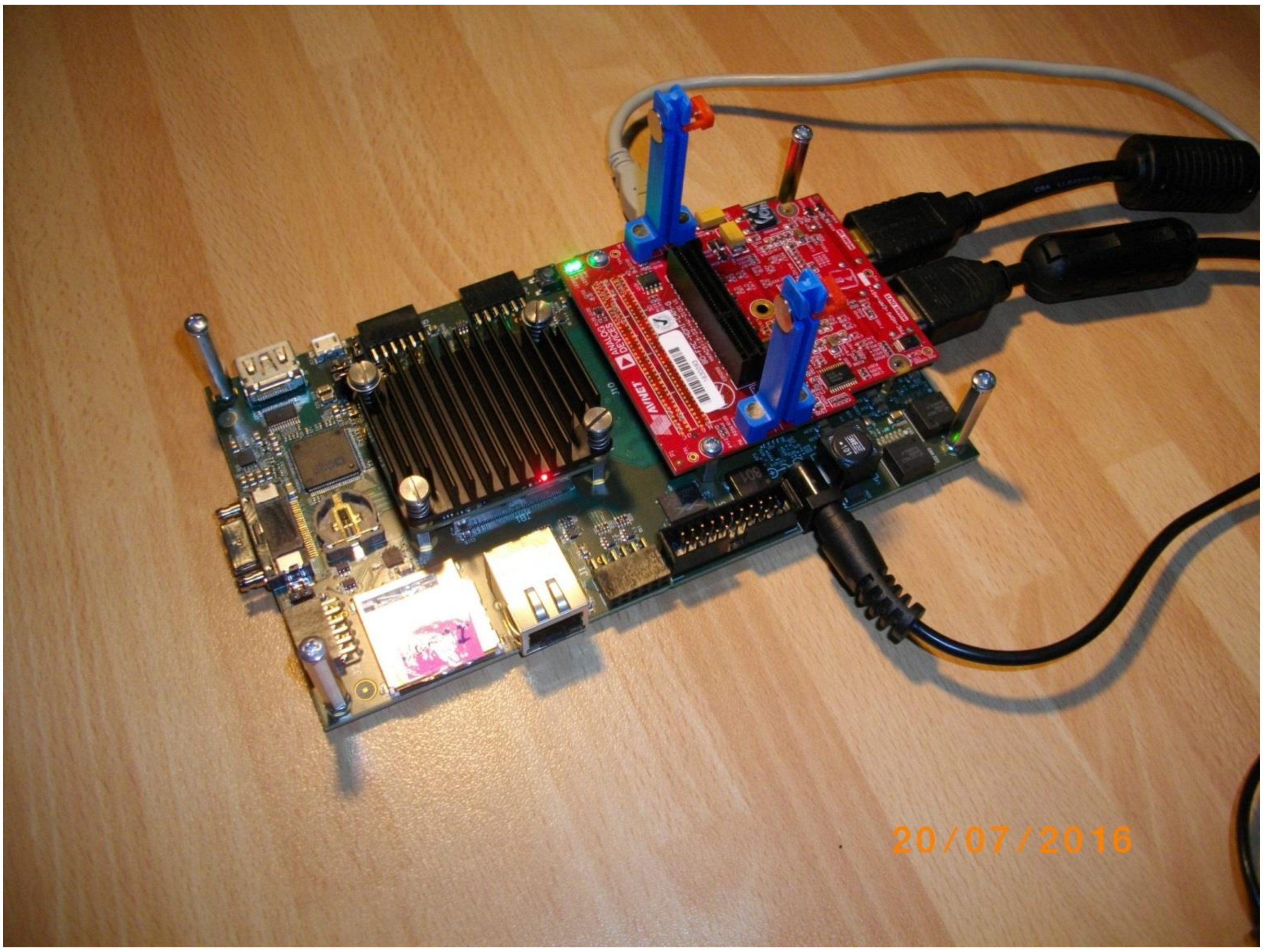


Figure 1: TE0701-05 platform with Zynq XC7Z020-2I device and HDMII-HDMIO support.

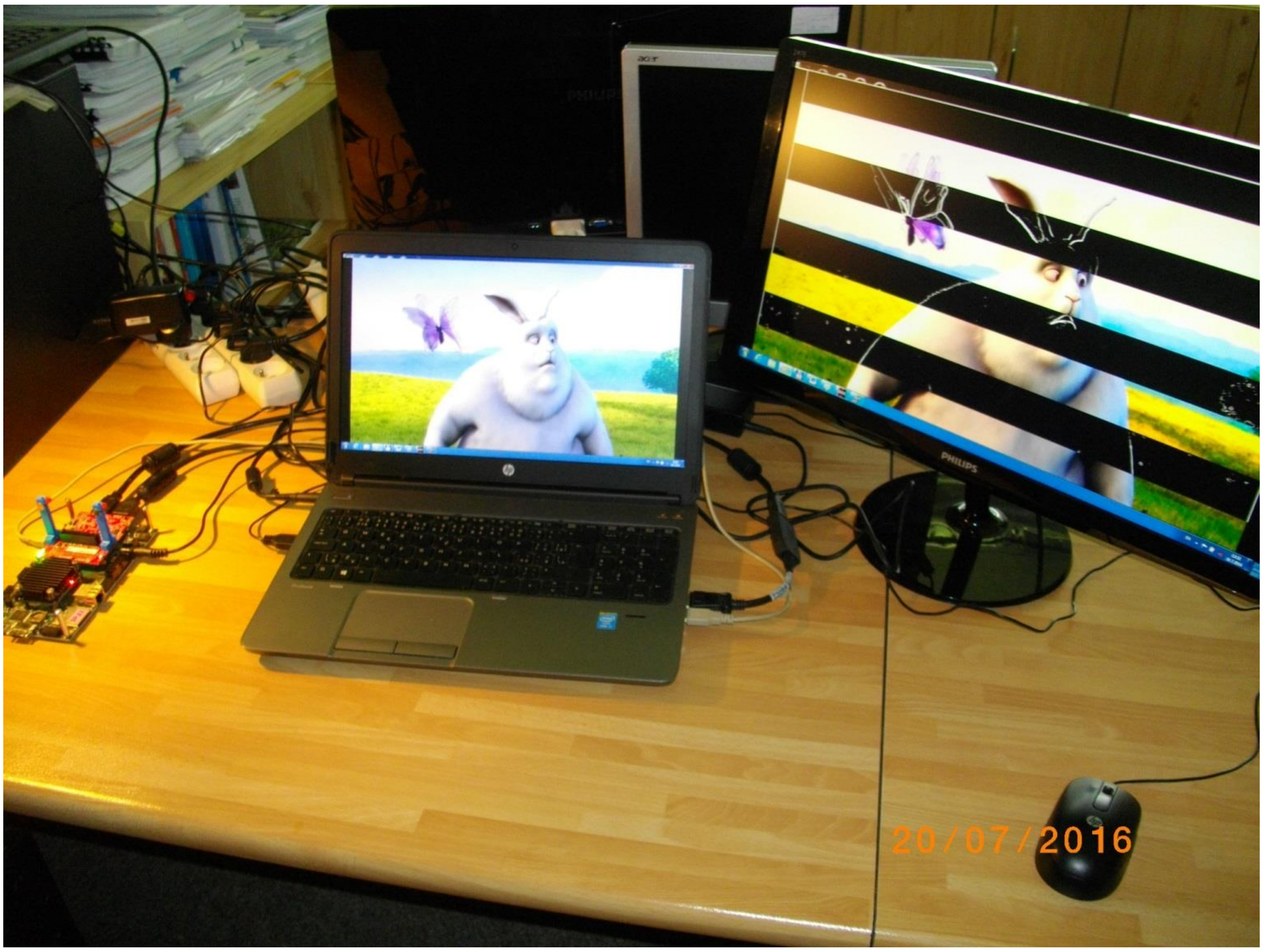


Figure 2: HW accelerated edge detection in Full HD with 4 HW paths and variable area (so04 demo).

Demo running in *Figure 2* is executing **so04_rows_resize_25_to_100** project with 4 HW data paths and time variable size (number of micro-lines) processed by the edge detection filters. Demo is controlled from user-defined C code running on ARM. The ARM processor can also perform user-defined, synchronous computation in parallel to the HW data paths. See section 2.4 for details.

Common setup for all included 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.
- Compiled demos can boot from the SD card directly after the power ON.

- Video data are provided by a Full HD HDMI source with resolution 1920x1080p60 (laptop).
- 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 IP core is sending data to the Full HD display (1920x1080p60).

1.2 Introduction to the demos

Edge detection
The edge detection algorithm is producing B/W Full HD 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 information, if the pixel is part of an edge or not. All computation is performed in fixed point.

Demos **sh01**, **sh02** and **sh03** provide accelerated HW computation of edge detection with 1, 2 or 3 parallel HW data paths. HW demos are using 1, 2 or 3 DMA HW channels from the DDR3 to 1, 2 or 3 as an input to Sobel filters. Another 1, 2 or 3 DMA HW channels support output from the Sobel filters to the DDR3. Zynq PL resources and the accelerations reached for these HW designs are summarised in sections 1.3, 1.4 and 1.5.

Demos **so01**, **so02**, **so03** and **so04** perform also edge detection. Used programming model enables write the user-defined, synchronous, parallel computation or ARM and to execute it in parallel with HW data paths. All other HW and performance related parameters of **so01**, **so02** and **so03** demos are practically identical to the **sh01**, **sh02** and **sh03** demos. That is why we do not repeat the diagrams. The Zynq PL resources and accelerations reached for the **so04** demo are summarised in sections 1.6.

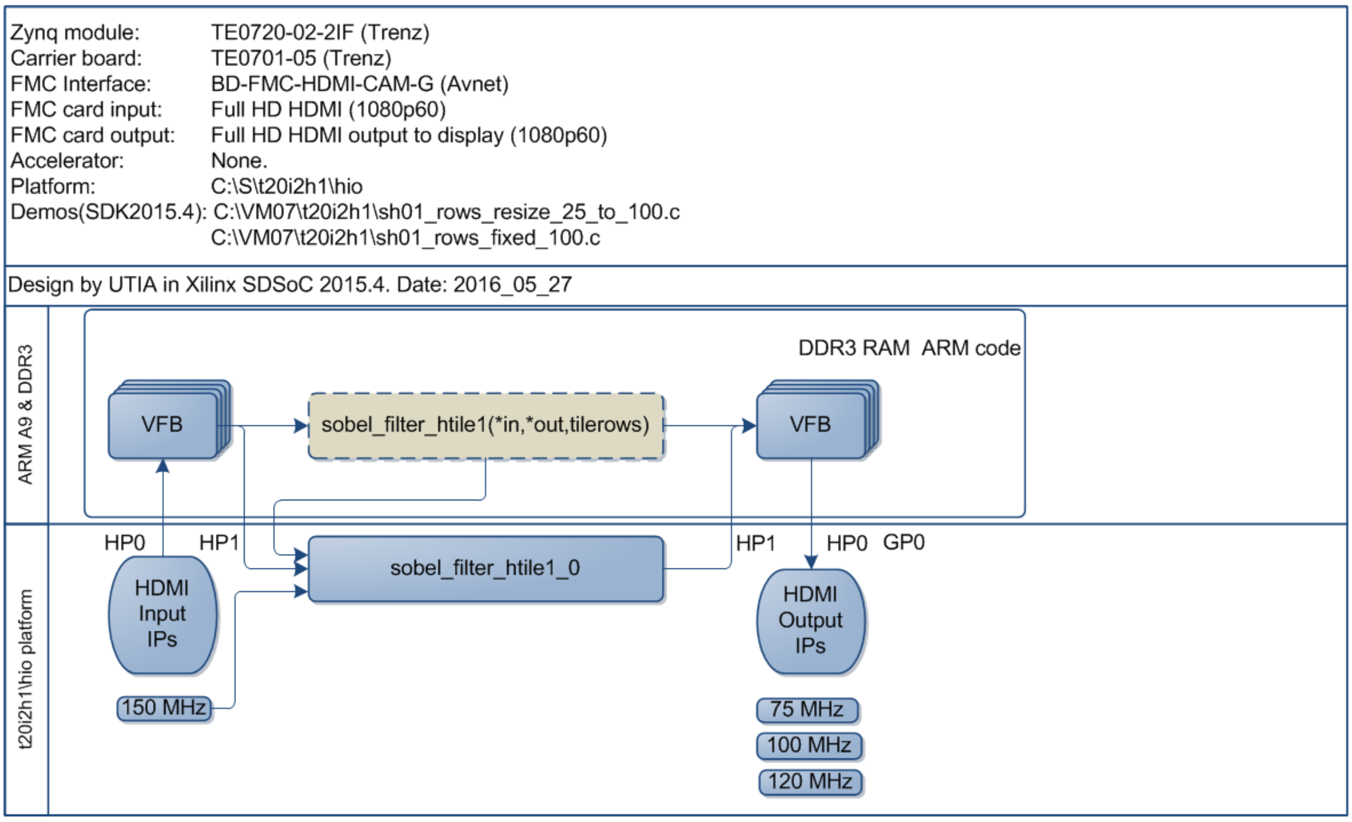
Motion detection
The motion detection algorithm detects and performs visualisation of **moving edges**. The moving edges are identified by two Sobel filters performing FIR filtering (similar to the above described edge detection) on pixels with identical coordinates, but from two subsequent video frames. The difference of these two filtered signals is filtered by a Median filter. The resulting signal is used for the nonlinear binary decision about the pixel. If the pixel is part of a moving edge, it is assigned red colour and merged with the original colour video signal. Resulting output Full HD video signal is unchanged, with the exception of red colour marked moving edges. See *Figure 2*. The fast moving edges in the face of the rabbit are marked by red pixels.

Demos **md01** and **md02** provide accelerated HW computation with 1 or 2 parallel HW data paths. HW demos are using 2 or 4 DMA HW channels for reading from **two** subsequent video frame buffers (located in the DDR3) to 1 or 2 video processing chains of HW accelerators performing the motion detection.
Another 1 or 2 DMA HW channels perform parallel write of results to the DDR3. Zynq PL resources and accelerations reached for these HW designs are summarised in sections 1.7 and 1.8.

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. ARM NEON HW accelerator is not used.

The performance of SW version of algorithms can be evaluated by booting demos from the enclosed BOOT.bin files. Files have been generated in SDSoC 2015.4 environment. The SDSoC 2015.4 source code and the SDSoC 2015.4 platform for the TE0720-03-2IF on TE0701-05 carrier board are not included. The SD card BOOT.bin files are enclosed.

1.3 Project sh01: Edge detection with single HW accelerator



Zynq module:	TE0720-02-2IF (Trenz)
Carrier board:	TE0701-05 (Trenz)
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)
FMC card input:	Full HD HDMI (1080p60)
FMC card output:	Full HD HDMI output to display (1080p60)
Accelerator:	None.
Platform:	C:\S\t20i2h1\hio
Demos(SDK2015.4):	C:\VM07\t20i2h1\sh01_rows_resize_25_to_100.c
	C:\VM07\t20i2h1\sh01_rows_fixed_100.c

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_05_27

Figure 3: Project sh01 - Edge detection with single HW accelerator.

TE0720-03-2IF Sobel 1x

Acceleration by HW: 5.59 x

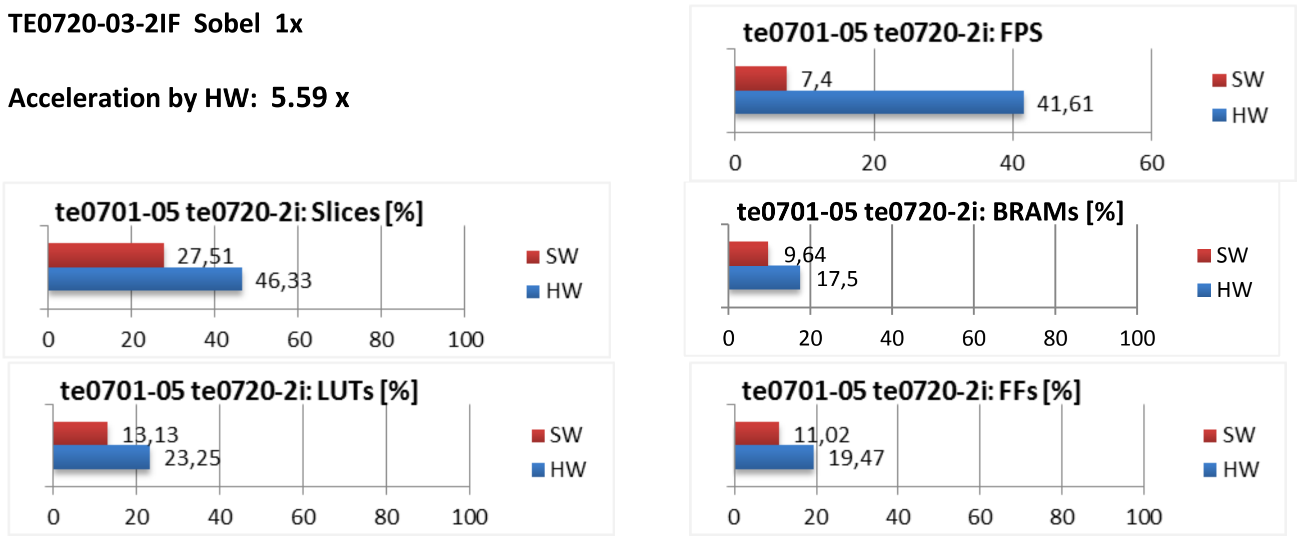
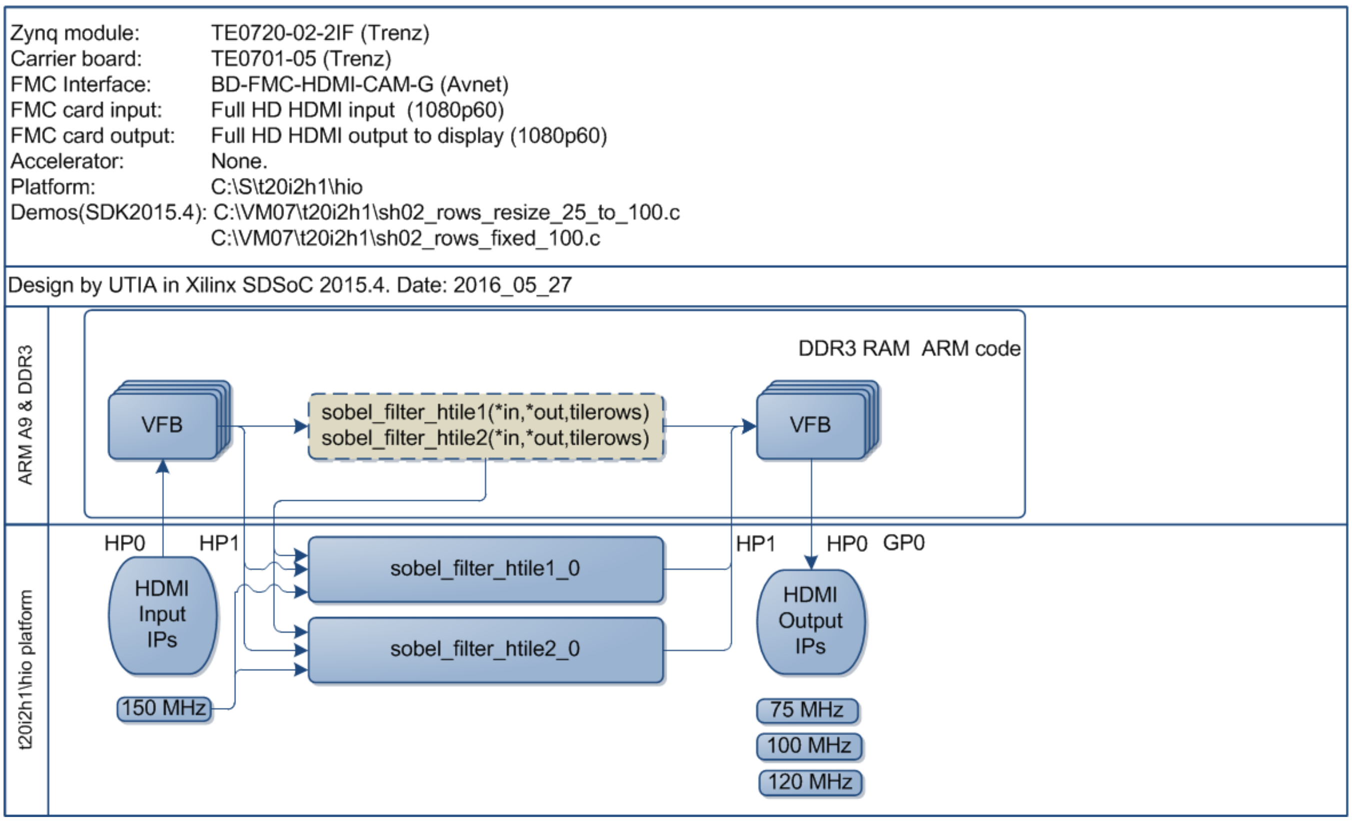


Figure 4: Project sh01 - Acceleration and HW resources used.

1.4 Project sh02: Edge detection with two HW accelerators



Zynq module:	TE0720-02-2IF (Trenz)
Carrier board:	TE0701-05 (Trenz)
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)
FMC card input:	Full HD HDMI input (1080p60)
FMC card output:	Full HD HDMI output to display (1080p60)
Accelerator:	None.
Platform:	C:\S\t20i2h1\hio
Demos(SDK2015.4):	C:\VM07\t20i2h1\sh02_rows_resize_25_to_100.c
	C:\VM07\t20i2h1\sh02_rows_fixed_100.c

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_05_27

Figure 5: Project sh02 - Edge detection with two HW accelerators.

TE0720-03-2IF Sobel 2x

Acceleration by HW: 8.11 x

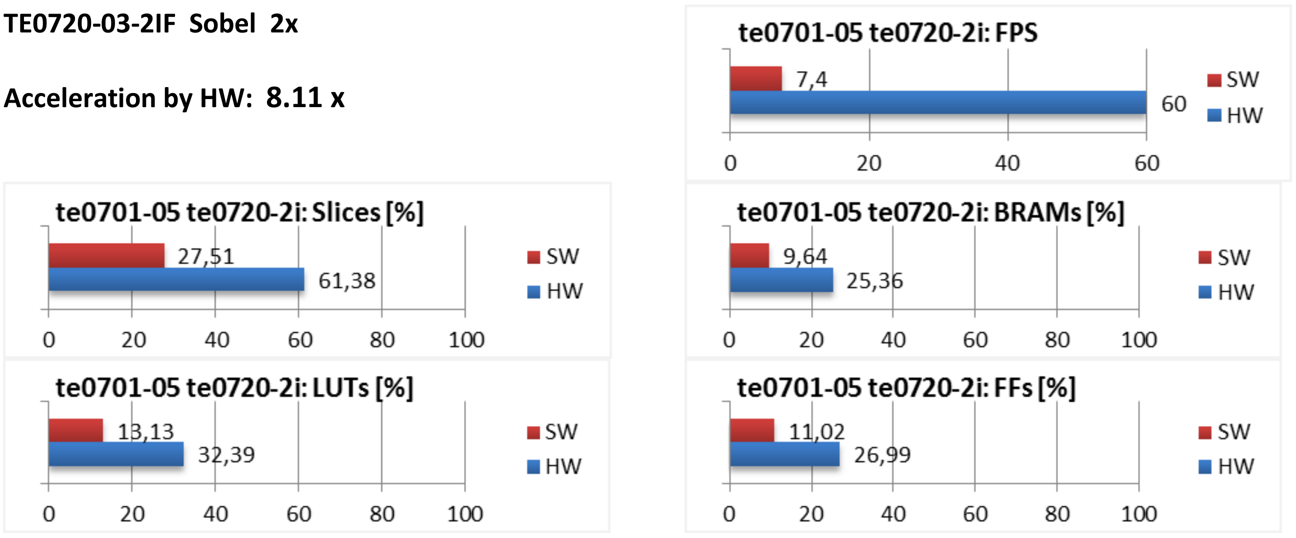
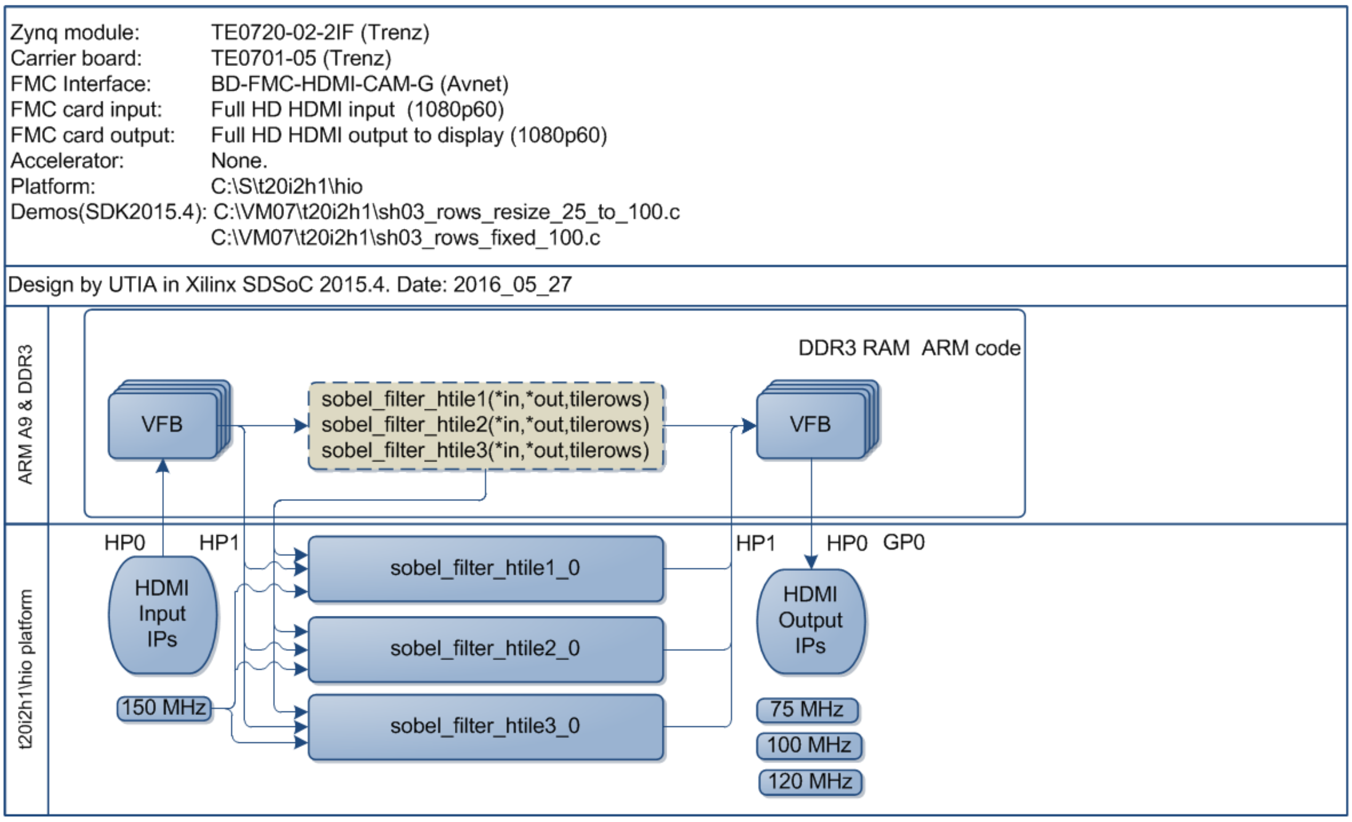


Figure 6: Project sh02 – Acceleration and HW resources used.

1.5 Project sh03: Edge detection with three HW accelerators



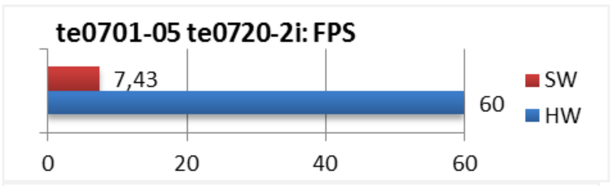
Zynq module:	TE0720-02-2IF (Trenz)
Carrier board:	TE0701-05 (Trenz)
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)
FMC card input:	Full HD HDMI input (1080p60)
FMC card output:	Full HD HDMI output to display (1080p60)
Accelerator:	None.
Platform:	C:\S\t20i2h1\hio
Demos(SDK2015.4):	C:\VM07\t20i2h1\sh03_rows_resize_25_to_100.c
	C:\VM07\t20i2h1\sh03_rows_fixed_100.c

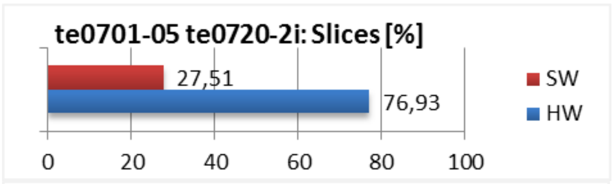
Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_05_27

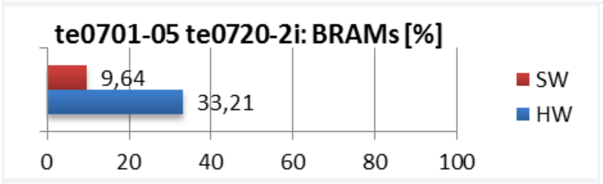
Figure 7: Project sh03 - Edge detection with three HW accelerators.

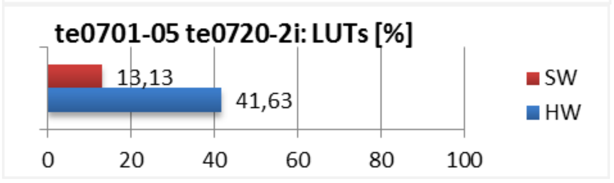
TE0720-03-2IF Sobel 3x

Acceleration by HW: 8.08 x









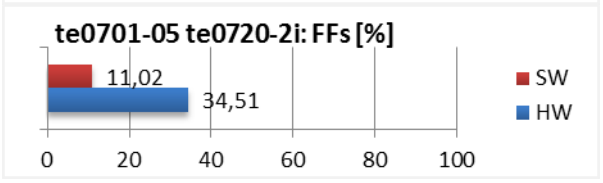
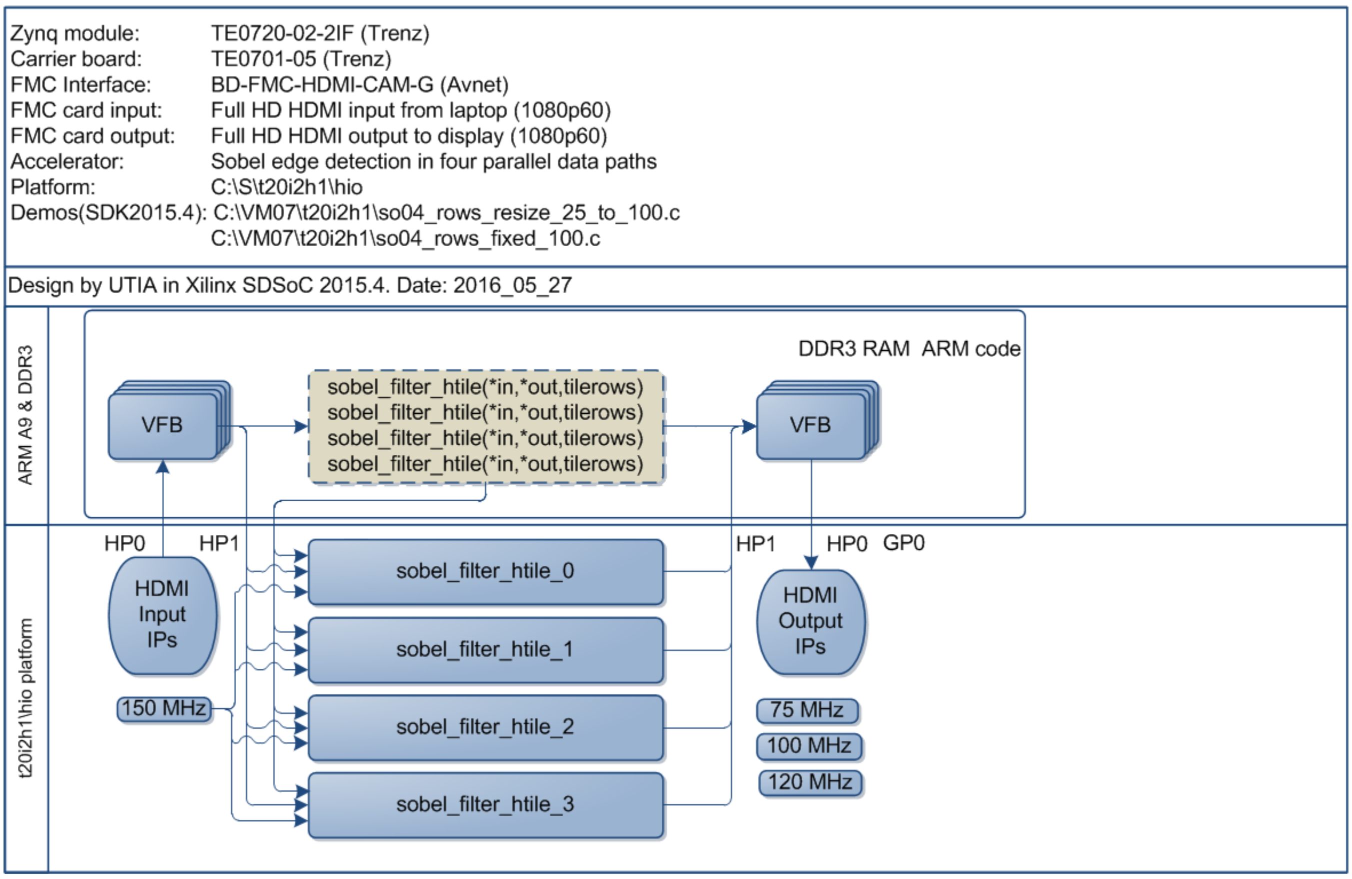


Figure 8: Project sh03 - Acceleration and HW resources used.

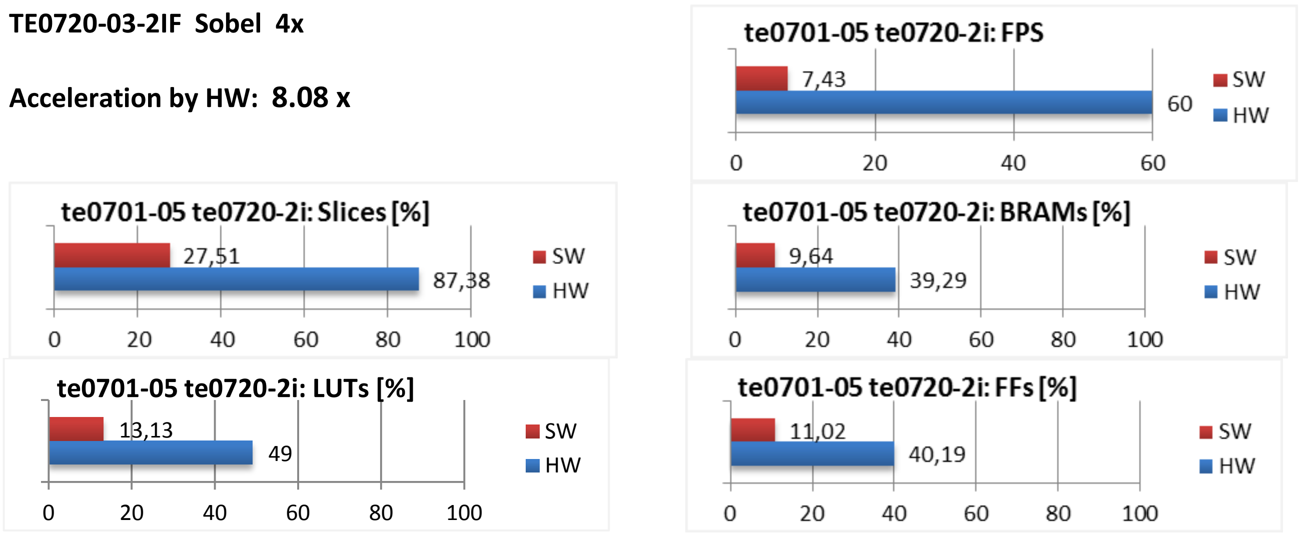
1.6 Project so04: Edge detection with four HW accelerators



Zynq module:	TE0720-02-2IF (Trenz)
Carrier board:	TE0701-05 (Trenz)
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)
FMC card input:	Full HD HDMI input from laptop (1080p60)
FMC card output:	Full HD HDMI output to display (1080p60)
Accelerator:	Sobel edge detection in four parallel data paths
Platform:	C:\S\t20i2h1\hio
Demos(SDK2015.4):	C:\VM07\t20i2h1\so04_rows_resize_25_to_100.c
	C:\VM07\t20i2h1\so04_rows_fixed_100.c

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_05_27

Figure 9: Project so04 – Alternative programming style - Edge detection with four HW accelerators.



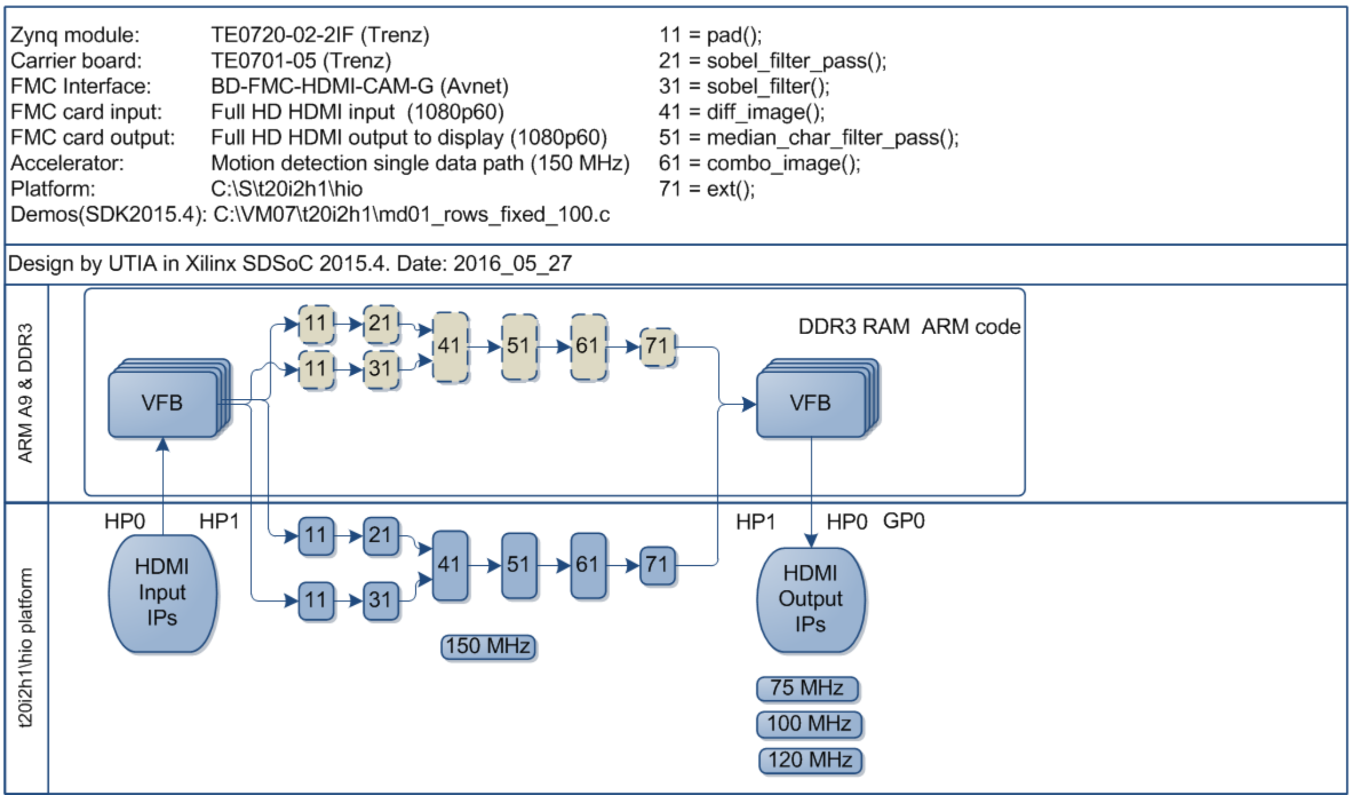
TE0720-03-2IF Sobel 4x

Acceleration by HW: 8.08 x

Figure 10: Project so04 - Acceleration and HW resources used.

ARM can execute user defined C code in parallel with HW data paths. See section 2.4 for more details.

1.7 Project md01: Motion detection with single chain of HW accelerators



Zynq module:	TE0720-02-2IF (Trenz)	11 = pad();
Carrier board:	TE0701-05 (Trenz)	21 = sobel_filter_pass();
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)	31 = sobel_filter();
FMC card input:	Full HD HDMI input (1080p60)	41 = diff_image();
FMC card output:	Full HD HDMI output to display (1080p60)	51 = median_char_filter_pass();
Accelerator:	Motion detection single data path (150 MHz)	61 = combo_image();
Platform:	C:\S\t20i2h1\hio	71 = ext();

Demos(SDK2015.4): C:\VM07\t20i2h1\md01_rows_fixed_100.c

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_05_27

Figure 11: Project md01 - Motion detection with single HW accelerator data path.

TE0720-03-2IF Motion Detection 1x

Acceleration by HW: 31.31 x

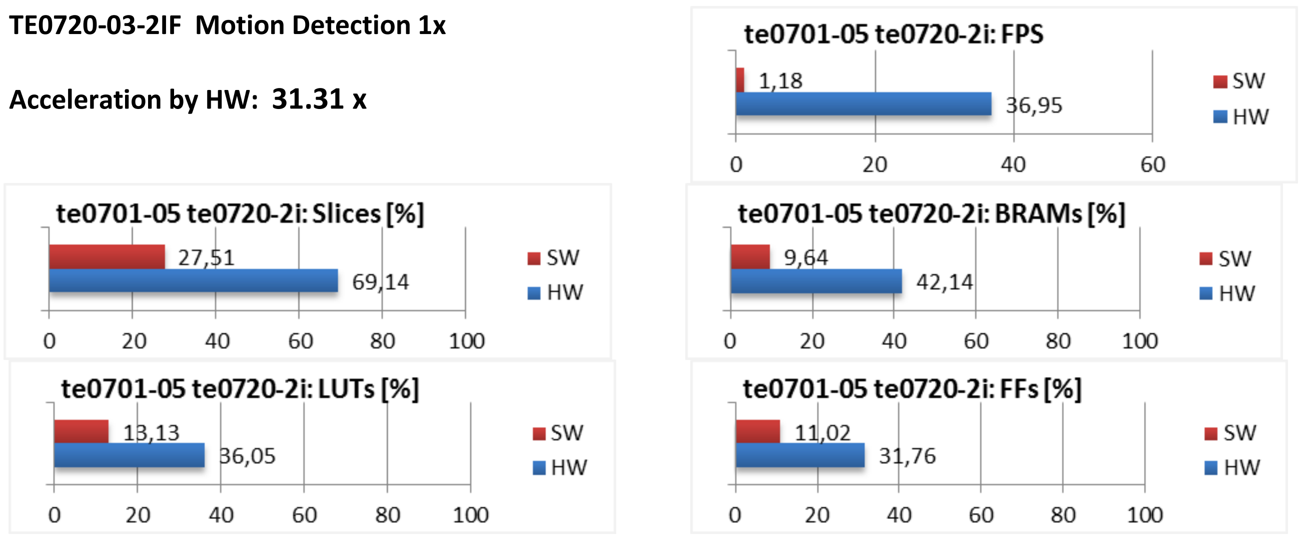
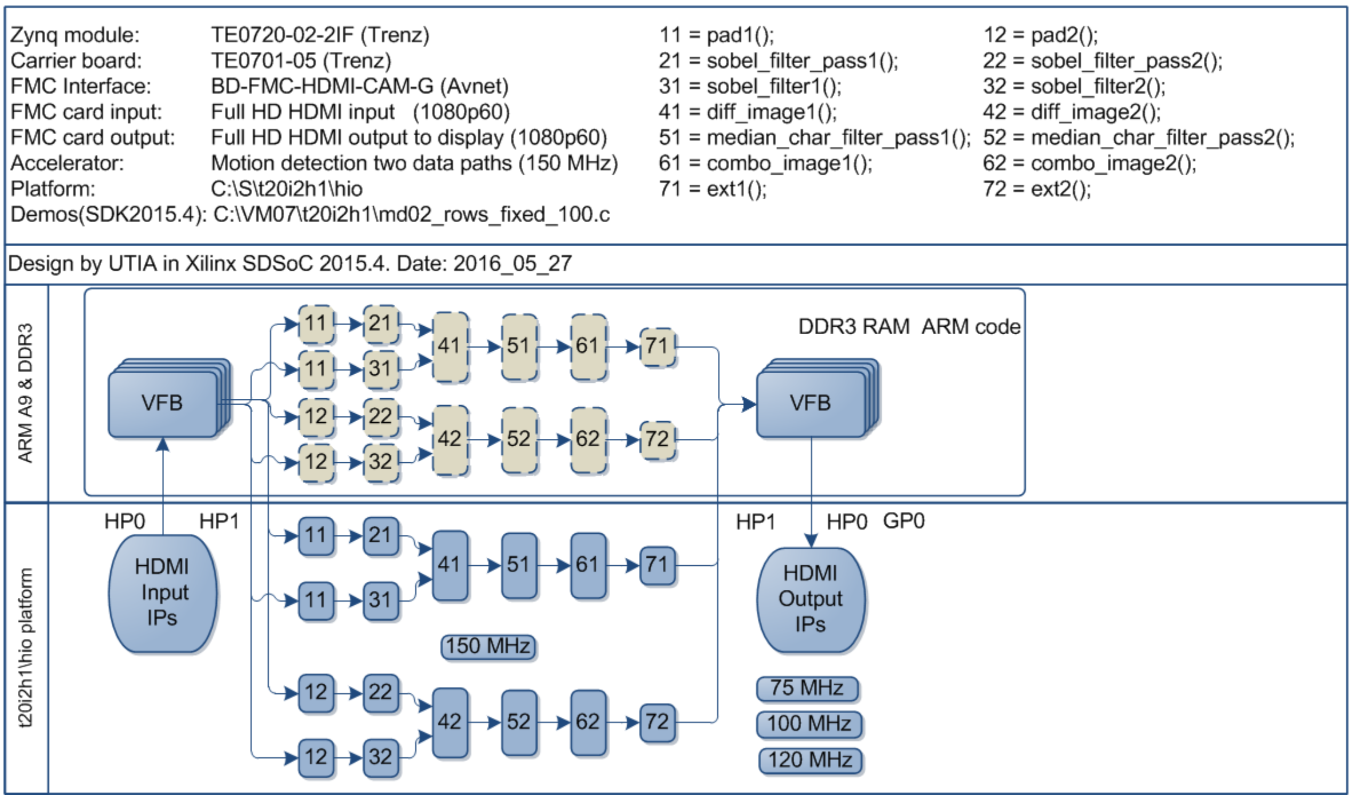


Figure 12: Project md01 - Acceleration and HW resources used

1.8 Project md02: Motion detection with two chains of HW accelerators



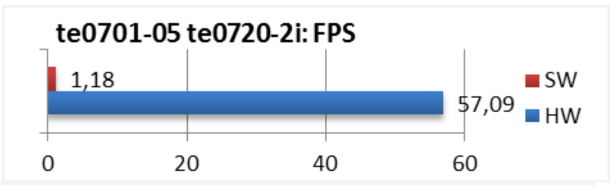
Zynq module:	TE0720-02-2IF (Trenz)	11 = pad1();	12 = pad2();
Carrier board:	TE0701-05 (Trenz)	21 = sobel_filter_pass1();	22 = sobel_filter_pass2();
FMC Interface:	BD-FMC-HDMI-CAM-G (Avnet)	31 = sobel_filter1();	32 = sobel_filter2();
FMC card input:	Full HD HDMI input (1080p60)	41 = diff_image1();	42 = diff_image2();
FMC card output:	Full HD HDMI output to display (1080p60)	51 = median_char_filter_pass1();	52 = median_char_filter_pass2();
Accelerator:	Motion detection two data paths (150 MHz)	61 = combo_image1();	62 = combo_image2();
Platform:	C:\S\t20i2h1\hio	71 = ext1();	72 = ext2();
Demos(SDK2015.4):	C:\VM07\t20i2h1\md02_rows_fixed_100.c		

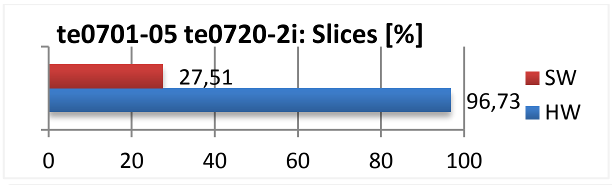
Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_05_27

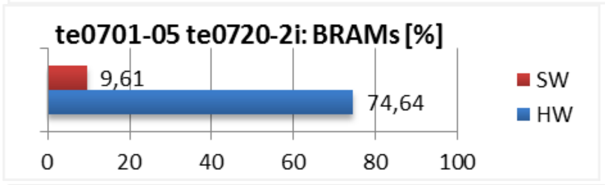
Figure 13: Project md02 - Motion detection with two HW accelerator data paths.

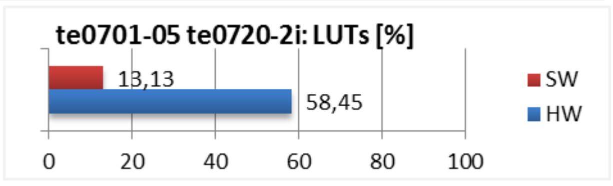
TE0720-03-2IF Motion Detection 2x

Acceleration by HW: 48.38 x









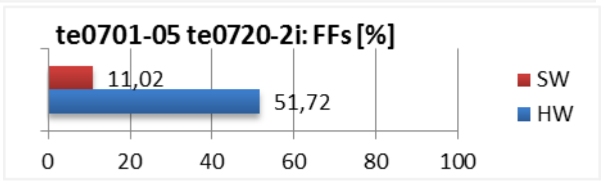


Figure 14: Project md02 - Acceleration and HW resources used.

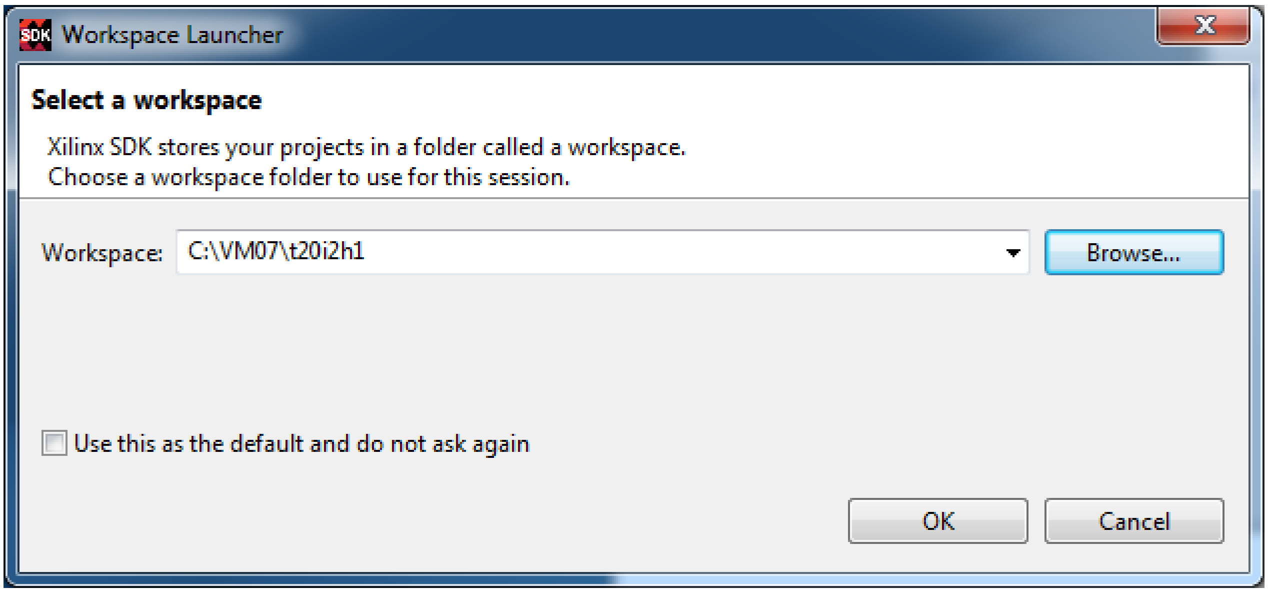
2. Installation of evaluation package

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\t20i2h1_V54_IMPORT

Create empty directory for Xilinx SDK workspace.
C:\VM_07\t20i2h1

Start Xilinx SDK 2015.4 and select the directory for the SDK 2015.4 workspace. See Figure 15. Select **C:\VM_07\t20i2h1**


Workspace Launcher

Select a workspace

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

Workspace: C:\VM07\t20i2h1

☐ Use this as the default and do not ask again

Figure 15: Select the SDK 2015.4 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 16.

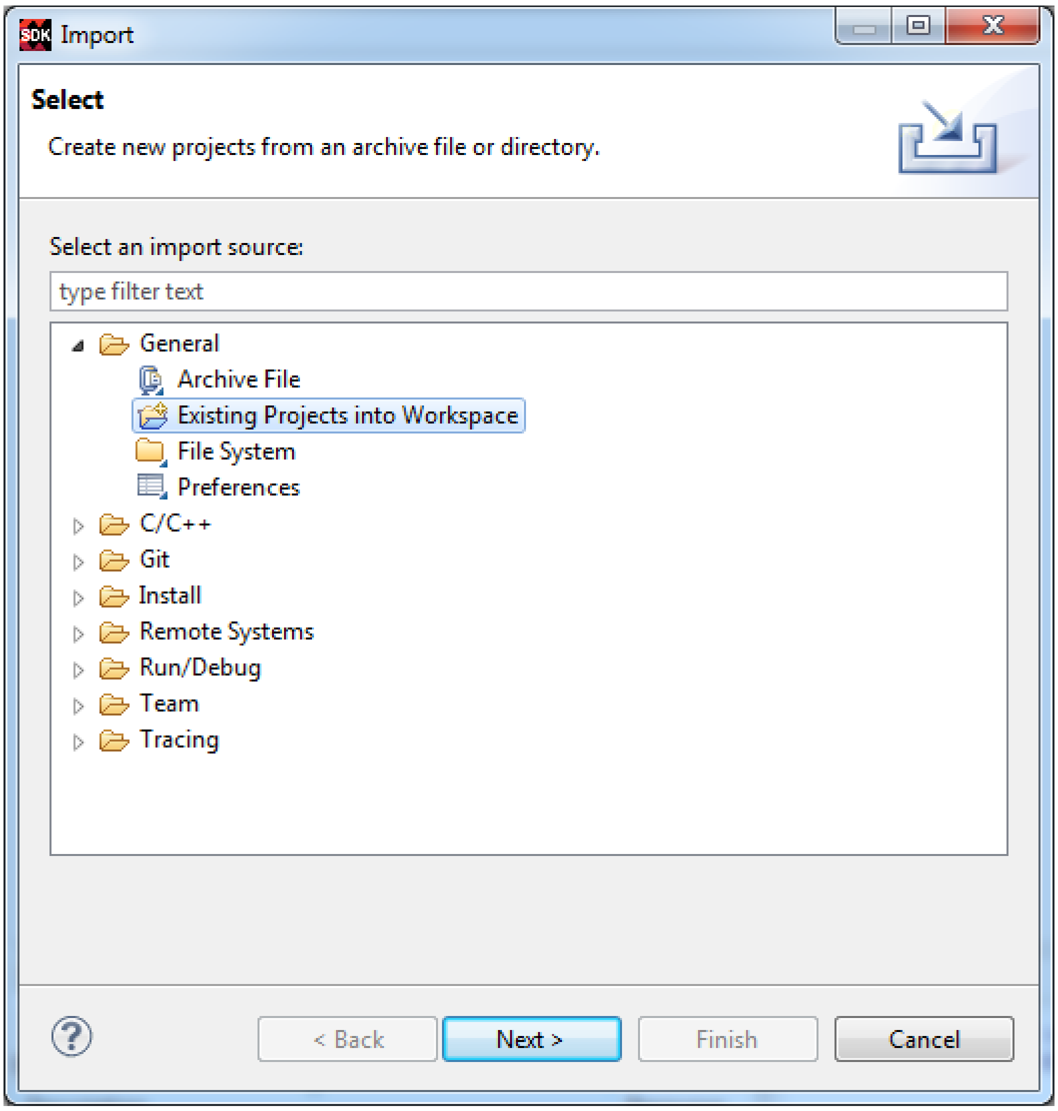


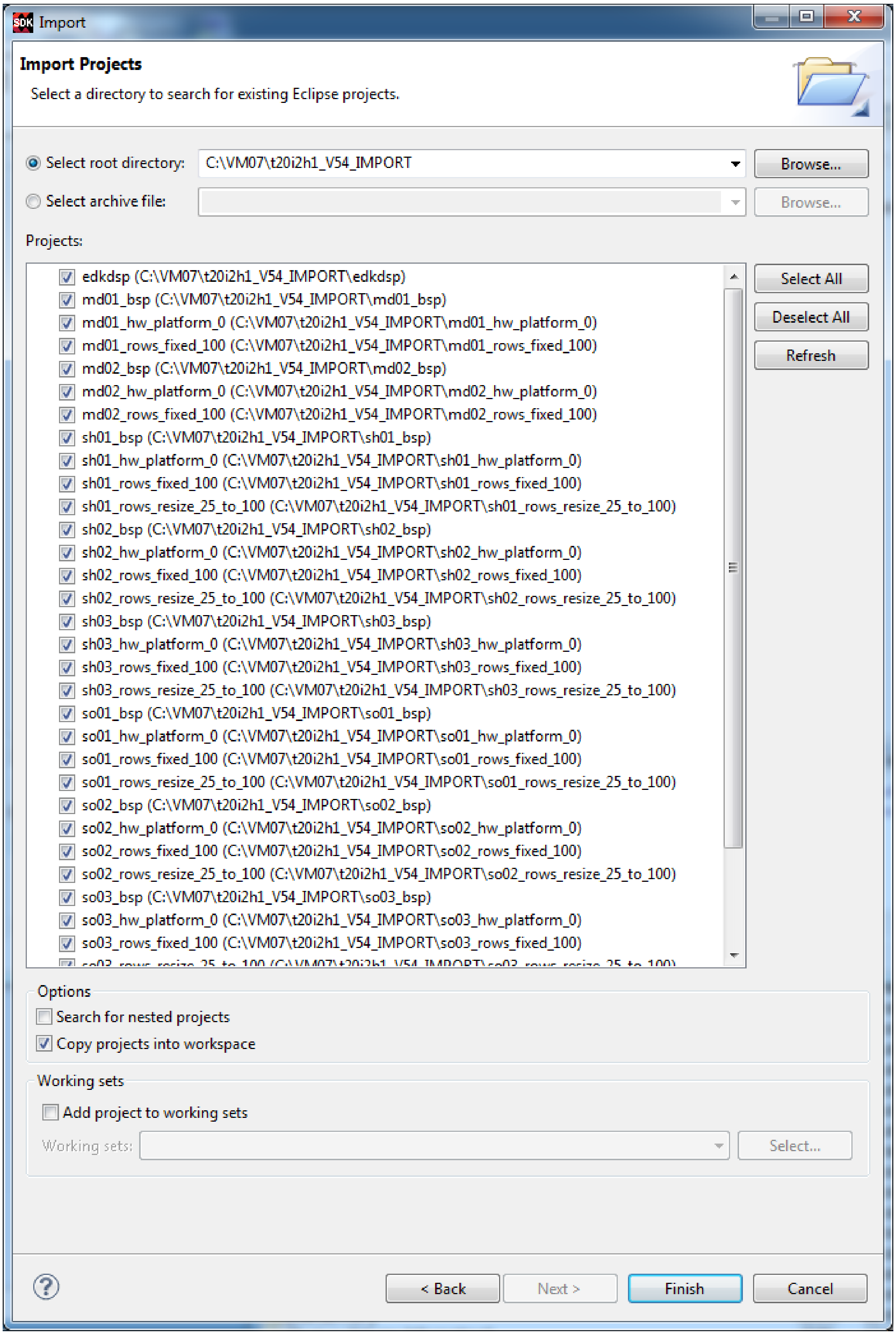
Figure 16: Import existing projects into workspace.

Type directory with projects to be imported. See Figure 17.

C:\VM_07\t20i2h1_V54_IMPORT

Set the "**Copy projects into workspace**" check box.
Click on Finish button. See Figure 17.

Projects are imported. Compilation starts automatically. This first compilation of all SDK 2015.4 SW projects will take several minutes to finish. It should finish without errors.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

(●) Select root directory: C:\VM07\t20i2h1_V54_IMPORT [Browse...]

( ) Select archive file: [Browse...]

Projects:

- [✓] edkdsp (C:\VM07\t20i2h1_V54_IMPORT\edkdsp)
- [✓] md01_bsp (C:\VM07\t20i2h1_V54_IMPORT\md01_bsp)
- [✓] md01_hw_platform_0 (C:\VM07\t20i2h1_V54_IMPORT\md01_hw_platform_0)
- [✓] md01_rows_fixed_100 (C:\VM07\t20i2h1_V54_IMPORT\md01_rows_fixed_100)
- [✓] md02_bsp (C:\VM07\t20i2h1_V54_IMPORT\md02_bsp)
- [✓] md02_hw_platform_0 (C:\VM07\t20i2h1_V54_IMPORT\md02_hw_platform_0)
- [✓] md02_rows_fixed_100 (C:\VM07\t20i2h1_V54_IMPORT\md02_rows_fixed_100)
- [✓] sh01_bsp (C:\VM07\t20i2h1_V54_IMPORT\sh01_bsp)
- [✓] sh01_hw_platform_0 (C:\VM07\t20i2h1_V54_IMPORT\sh01_hw_platform_0)
- [✓] sh01_rows_fixed_100 (C:\VM07\t20i2h1_V54_IMPORT\sh01_rows_fixed_100)
- [✓] sh01_rows_resize_25_to_100 (C:\VM07\t20i2h1_V54_IMPORT\sh01_rows_resize_25_to_100)
- [✓] sh02_bsp (C:\VM07\t20i2h1_V54_IMPORT\sh02_bsp)
- [✓] sh02_hw_platform_0 (C:\VM07\t20i2h1_V54_IMPORT\sh02_hw_platform_0)
- [✓] sh02_rows_fixed_100 (C:\VM07\t20i2h1_V54_IMPORT\sh02_rows_fixed_100)
- [✓] sh02_rows_resize_25_to_100 (C:\VM07\t20i2h1_V54_IMPORT\sh02_rows_resize_25_to_100)
- [✓] sh03_bsp (C:\VM07\t20i2h1_V54_IMPORT\sh03_bsp)
- [✓] sh03_hw_platform_0 (C:\VM07\t20i2h1_V54_IMPORT\sh03_hw_platform_0)
- [✓] sh03_rows_fixed_100 (C:\VM07\t20i2h1_V54_IMPORT\sh03_rows_fixed_100)
- [✓] sh03_rows_resize_25_to_100 (C:\VM07\t20i2h1_V54_IMPORT\sh03_rows_resize_25_to_100)
- [✓] so01_bsp (C:\VM07\t20i2h1_V54_IMPORT\so01_bsp)
- [✓] so01_hw_platform_0 (C:\VM07\t20i2h1_V54_IMPORT\so01_hw_platform_0)
- [✓] so01_rows_fixed_100 (C:\VM07\t20i2h1_V54_IMPORT\so01_rows_fixed_100)
- [✓] so01_rows_resize_25_to_100 (C:\VM07\t20i2h1_V54_IMPORT\so01_rows_resize_25_to_100)
- [✓] so02_bsp (C:\VM07\t20i2h1_V54_IMPORT\so02_bsp)
- [✓] so02_hw_platform_0 (C:\VM07\t20i2h1_V54_IMPORT\so02_hw_platform_0)
- [✓] so02_rows_fixed_100 (C:\VM07\t20i2h1_V54_IMPORT\so02_rows_fixed_100)
- [✓] so02_rows_resize_25_to_100 (C:\VM07\t20i2h1_V54_IMPORT\so02_rows_resize_25_to_100)
- [✓] so03_bsp (C:\VM07\t20i2h1_V54_IMPORT\so03_bsp)
- [✓] so03_hw_platform_0 (C:\VM07\t20i2h1_V54_IMPORT\so03_hw_platform_0)
- [✓] so03_rows_fixed_100 (C:\VM07\t20i2h1_V54_IMPORT\so03_rows_fixed_100)

Options

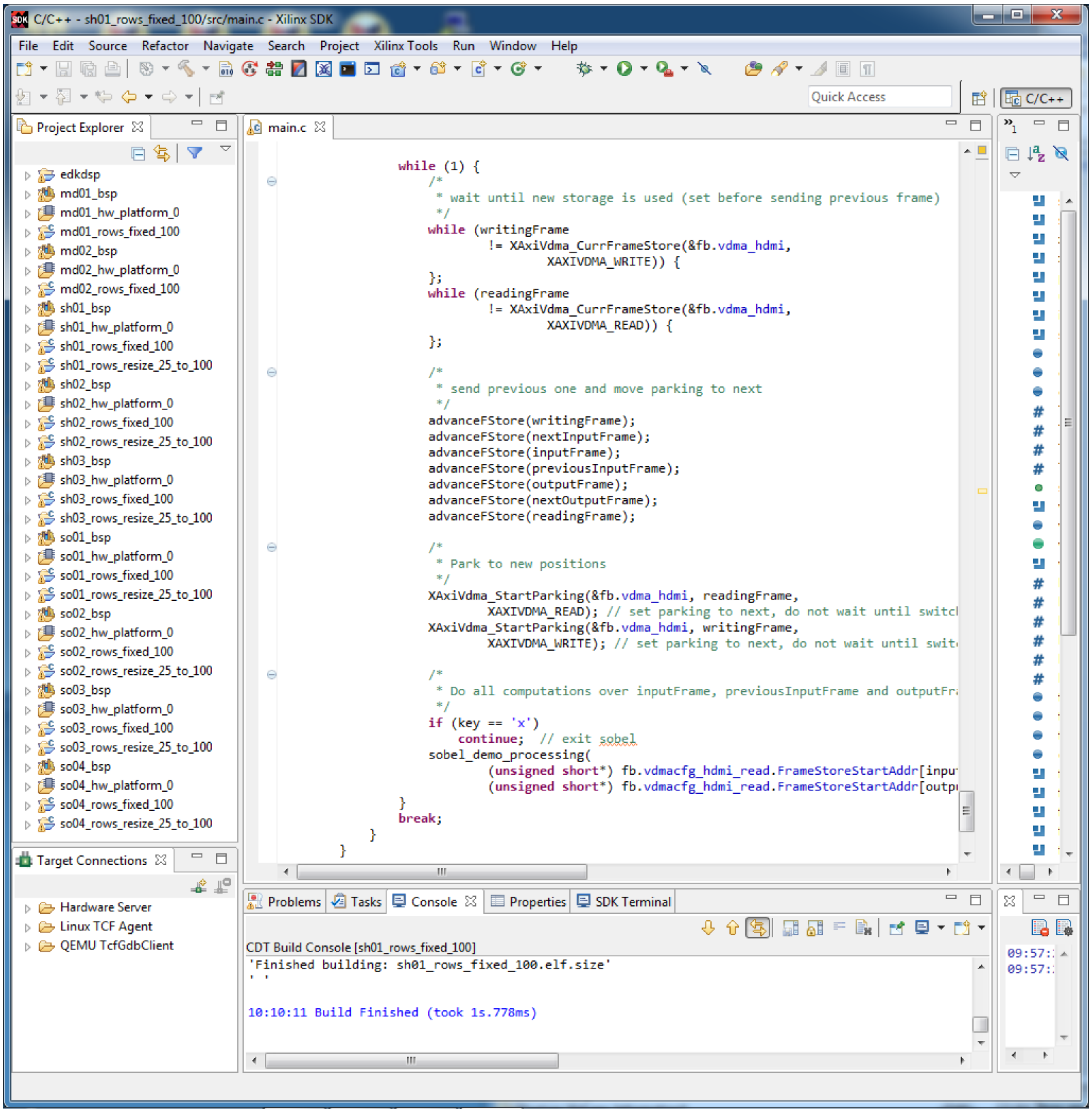
- [ ] Search for nested projects
- [✓] Copy projects into workspace

Working sets

- [ ] Add project to working sets

Working sets:

Figure 17: Select "Copy projects into workspace" and finish the import of all projects.



```
            while (1) {
                /*
                 * wait until new storage is used (set before sending previous frame)
                 */
                while (writingFrame
                        != XAxiVdma_CurrFrameStore(&fb.vdma_hdmi,
                                XAXIVDMA_WRITE)) {
                };
                while (readingFrame
                        != XAxiVdma_CurrFrameStore(&fb.vdma_hdmi,
                                XAXIVDMA_READ)) {
                };

                /*
                 * send previous one and move parking to next
                 */
                advanceFStore(writingFrame);
                advanceFStore(nextInputFrame);
                advanceFStore(inputFrame);
                advanceFStore(previousInputFrame);
                advanceFStore(outputFrame);
                advanceFStore(nextOutputFrame);
                advanceFStore(readingFrame);

                /*
                 * Park to new positions
                 */
                XAxiVdma_StartParking(&fb.vdma_hdmi, readingFrame,
                        XAXIVDMA_READ); // set parking to next, do not wait until switcl
                XAxiVdma_StartParking(&fb.vdma_hdmi, writingFrame,
                        XAXIVDMA_WRITE); // set parking to next, do not wait until swit

                /*
                 * Do all computations over inputFrame, previousInputFrame and outputFr
                 */
                if (key == 'x')
                    continue;  // exit sobel
                sobel_demo_processing(
                        (unsigned short*) fb.vdmacfg_hdmi_read.FrameStoreStartAddr[inpu
                        (unsigned short*) fb.vdmacfg_hdmi_read.FrameStoreStartAddr[outp
            }
            break;
        }
    }
```

CDT Build Console [sh01_rows_fixed_100]

```
'Finished building: sh01_rows_fixed_100.elf.size'
' '

10:10:11 Build Finished (took 1s.778ms)
```

Figure 18: All projects are compiled in debug mode.

The SDK 2015.4 environment compiles all imported demos in debug mode by default.

2.2 HW setup

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

TE0720-03-2IF; Part: XC7Z020-2CLG484I; 1 GByte DDR; Industrial Grade;	Price: €269,00 [1]
Heatsink for TE0720, spring-loaded embedded;	Price: €19.00 [2]
TE0701-05 Carrier Board for Trenz Electronic 7 Series;	Price: €249.00 [3]
AES-FMC-HDMI-CAM-G FMC card with HDMI I/O and CAM interface	Price: $250.00 [4]

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].

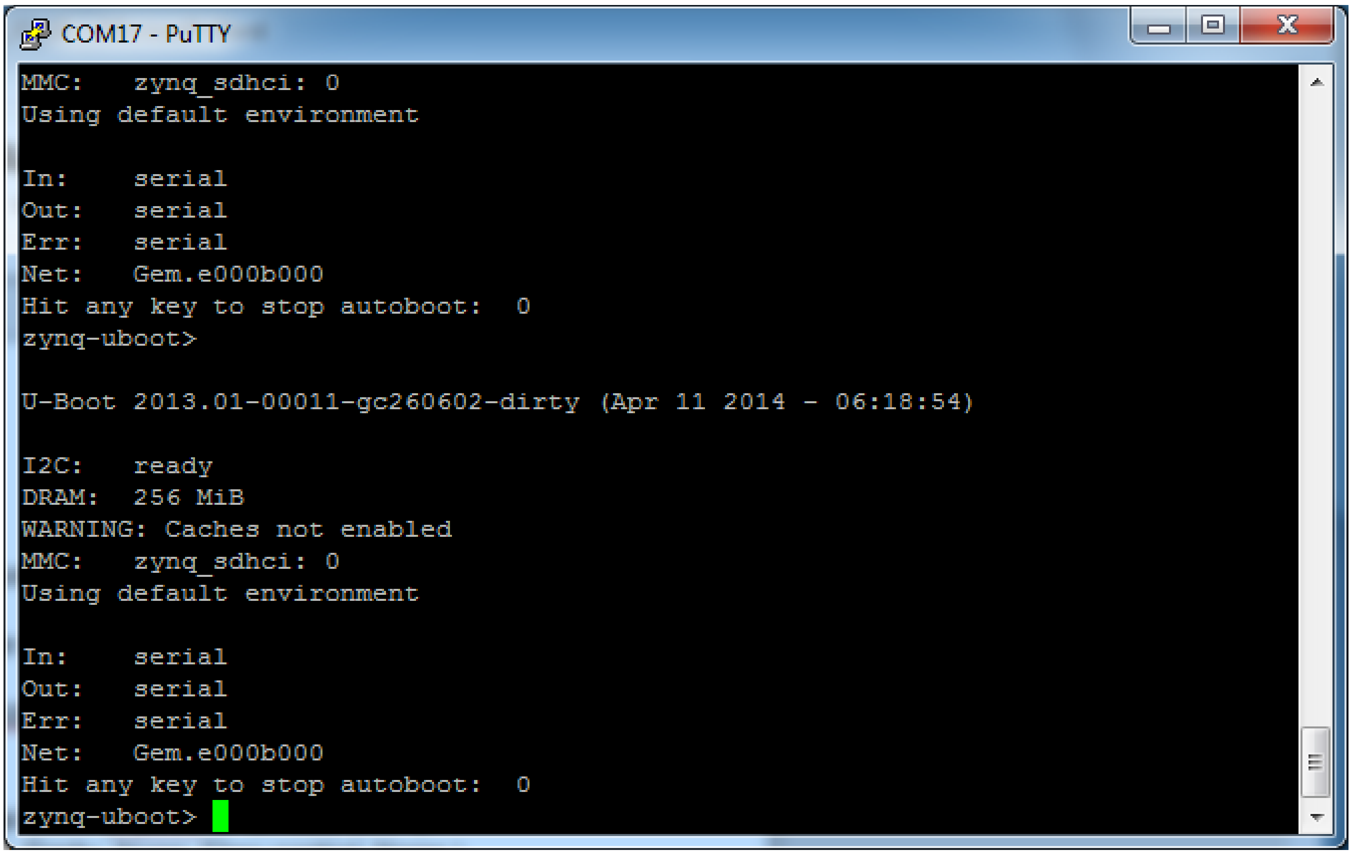
Trenz TE0701-04 or TE0701-05 carriers require modifications to run the FMC carrier AES-FMC-HDMI-CAM-G with Zynq TE0720-03-2IF system on module. The modification is related to the swapped polarity of the differential clock signal for the FMC board. Evaluation HW systems with carriers TE0701-04 or TE0701-05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

- Connect source of the Full HD HDMI signal (usually PC or laptop) to the HDMI IN connector on the AES-FMC-HDMI-CAM-G FMC card.
- Connect Full HD HDMI (or DVI) monitor by HDMI cable to the HDMI OUT on the AES-FMC-HDMI-CAM-G FMC card.
- Switch the monitor ON.
- Connect the carrier board by USB-to-miniUSB cable to PC to support JTAG serial link and the standard serial terminal.
- Connect power supply (DC 12V).
- Open and configure the standard serial terminal client (PuTTY or similar) on PC.
  (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.
- Close the serial terminal client SW on the PC before you switch-off the power for the TE0701-05 carrier board.

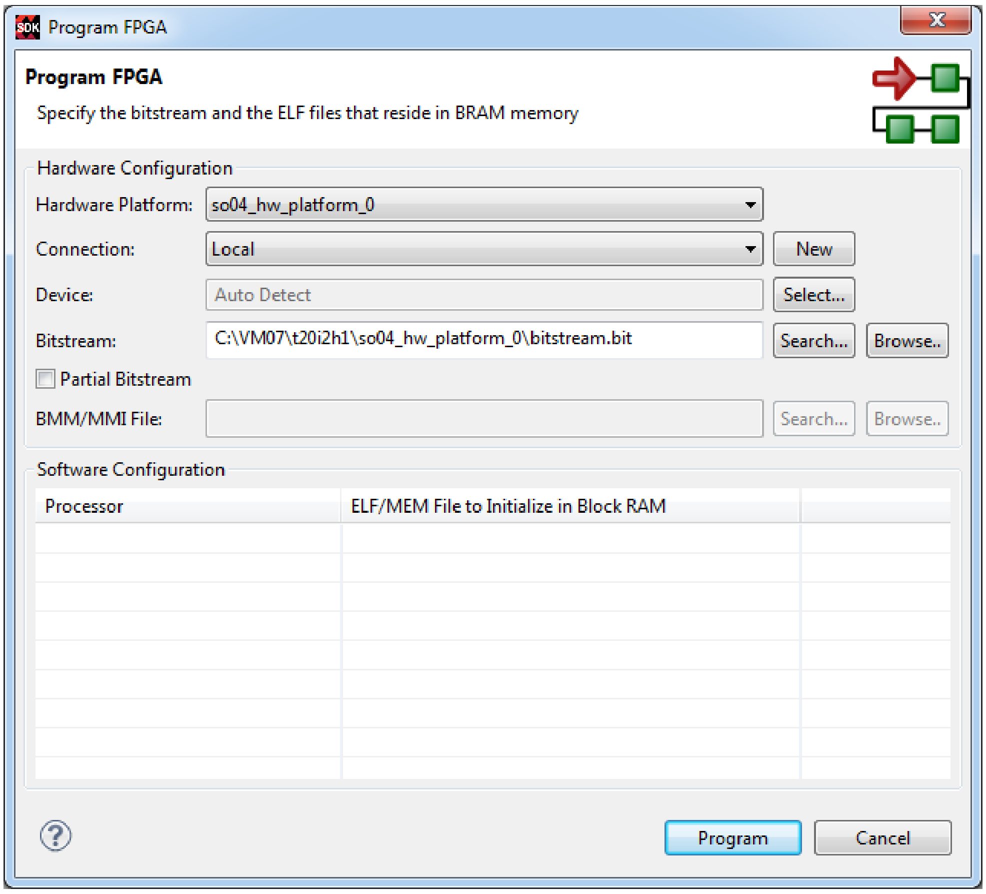


```
MMC:   zynq_sdhci: 0
Using default environment

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

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

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

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

Figure 19: Serial console. Reset board and stop autoboot.

Download bitstream to the board. Demo **so04_rows_resize_25_to_100** will be used as an example. The **bitstream.bit** for the demo is located in the directory:

C:\VM_07\t20i2h1\so04_hw_platform_0



Program FPGA

Program FPGA

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

Hardware Configuration

Hardware Platform: so04_hw_platform_0

Connection: Local

Device: Auto Detect

Bitstream: C:\VM07\t20i2h1\so04_hw_platform_0\bitstream.bit

☐ Partial Bitstream

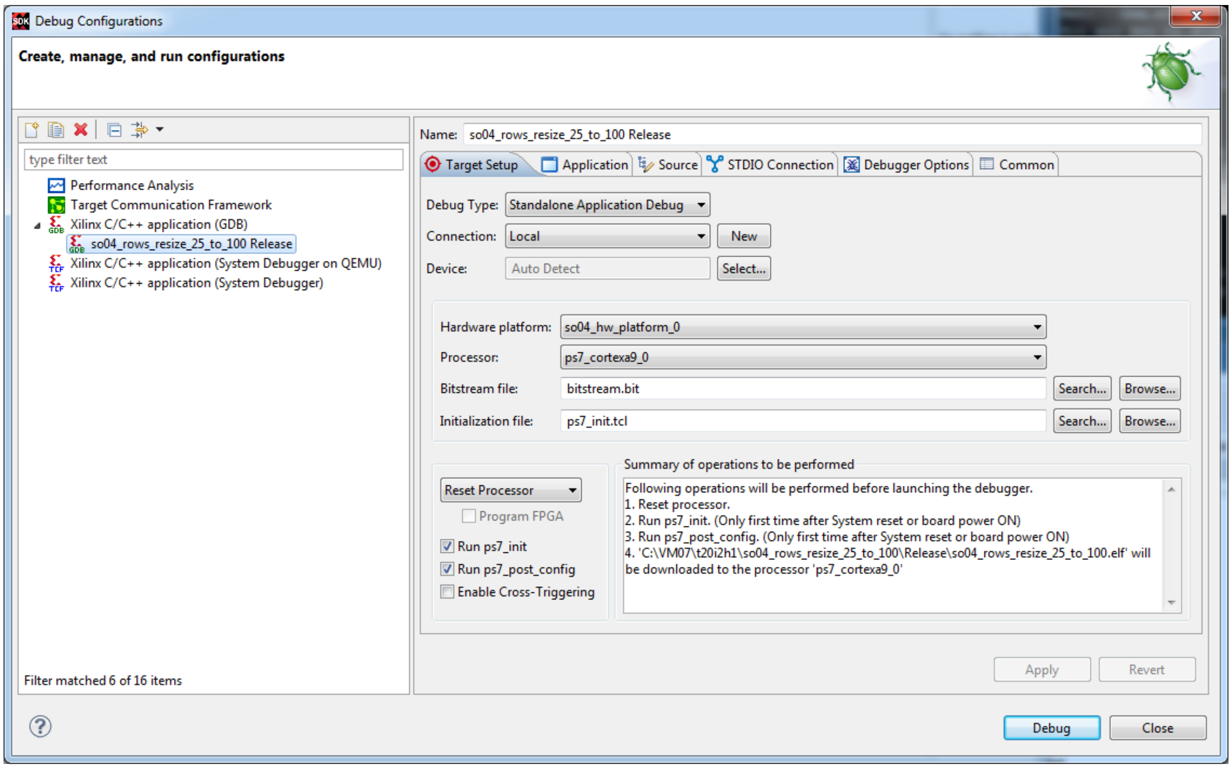
BMM/MMI File:

Software Configuration

Processor	ELF/MEM File to Initialize in Block RAM

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

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



Debug Configurations

Create, manage, and run configurations

type filter text

- Performance Analysis
- Target Communication Framework
- Xilinx C/C++ application (GDB)
  - so04_rows_resize_25_to_100 Release
- Xilinx C/C++ application (System Debugger on QEMU)
- Xilinx C/C++ application (System Debugger)

Filter matched 6 of 16 items

Name: so04_rows_resize_25_to_100 Release

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

Debug Type: Standalone Application Debug

Connection: Local

Device: Auto Detect

Hardware platform: so04_hw_platform_0

Processor: ps7_cortexa9_0

Bitstream file: bitstream.bit

Initialization file: ps7_init.tcl

Reset Processor

☐ Program FPGA

☑ Run ps7_init

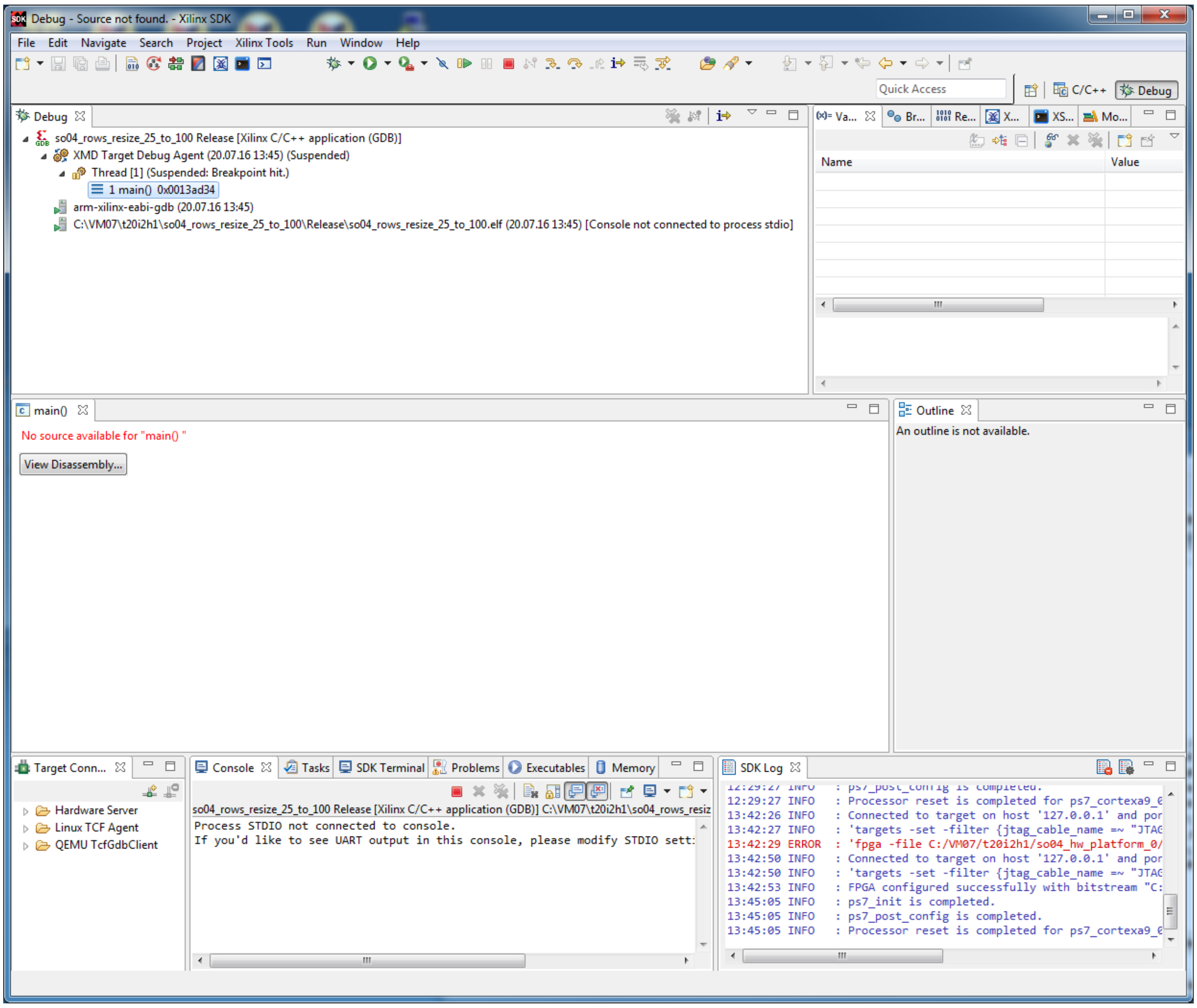
☑ Run ps7_post_config

☐ Enable Cross-Triggering

Summary of operations to be performed

```
Following operations will be performed before launching the debugger.
1. Reset processor.
2. Run ps7_init. (Only first time after System reset or board power ON)
3. Run ps7_post_config. (Only first time after System reset or board power ON)
4. 'C:\VM07\t20i2h1\so04_rows_resize_25_to_100\Release\so04_rows_resize_25_to_100.elf' will
be downloaded to the processor 'ps7_cortexa9_0'
```

Figure 21: Select demo application for debug.



```
so04_rows_resize_25_to_100 Release [Xilinx C/C++ application (GDB)]
  XMD Target Debug Agent (20.07.16 13:45) (Suspended)
    Thread [1] (Suspended: Breakpoint hit.)
      1 main() 0x0013ad34
  arm-xilinx-eabi-gdb (20.07.16 13:45)
  C:\VM07\t20i2h1\so04_rows_resize_25_to_100\Release\so04_rows_resize_25_to_100.elf (20.07.16 13:45) [Console not connected to process stdio]
```

```
No source available for "main() "
```

```
An outline is not available.
```

```
so04_rows_resize_25_to_100 Release [Xilinx C/C++ application (GDB)] C:\VM07\t20i2h1\so04_rows_resiz
Process STDIO not connected to console.
If you'd like to see UART output in this console, please modify STDIO sett:
```

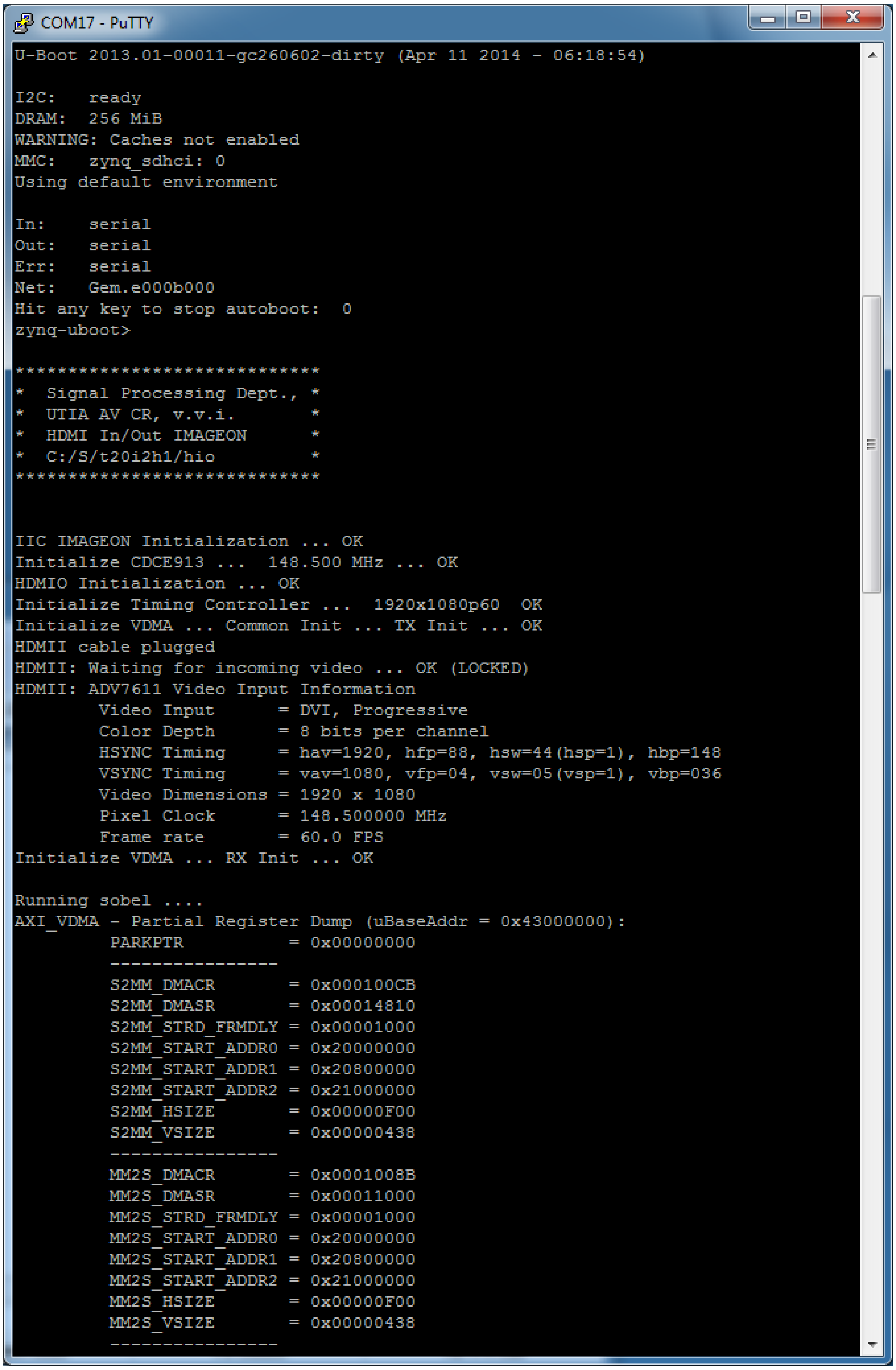
```
12:29:27 INFO   : ps7_post_config is completed.
12:29:27 INFO   : Processor reset is completed for ps7_cortexa9_0
13:42:26 INFO   : Connected to target on host '127.0.0.1' and por
13:42:27 INFO   : 'targets -set -filter {jtag_cable_name =~ "JTAG
13:42:29 ERROR  : 'fpga -file C:/VM07/t20i2h1/so04_hw_platform_0/
13:42:50 INFO   : Connected to target on host '127.0.0.1' and por
13:42:50 INFO   : 'targets -set -filter {jtag_cable_name =~ "JTAG
13:42:53 INFO   : FPGA configured successfully with bitstream "C:
13:45:05 INFO   : ps7_init is completed.
13:45:05 INFO   : ps7_post_config is completed.
13:45:05 INFO   : Processor reset is completed for ps7_cortexa9_0
```

Figure 22: Debug stops at first executable line of ARM Cortex A9 code.

Start the **so04** demo ARM C code from the debugger.

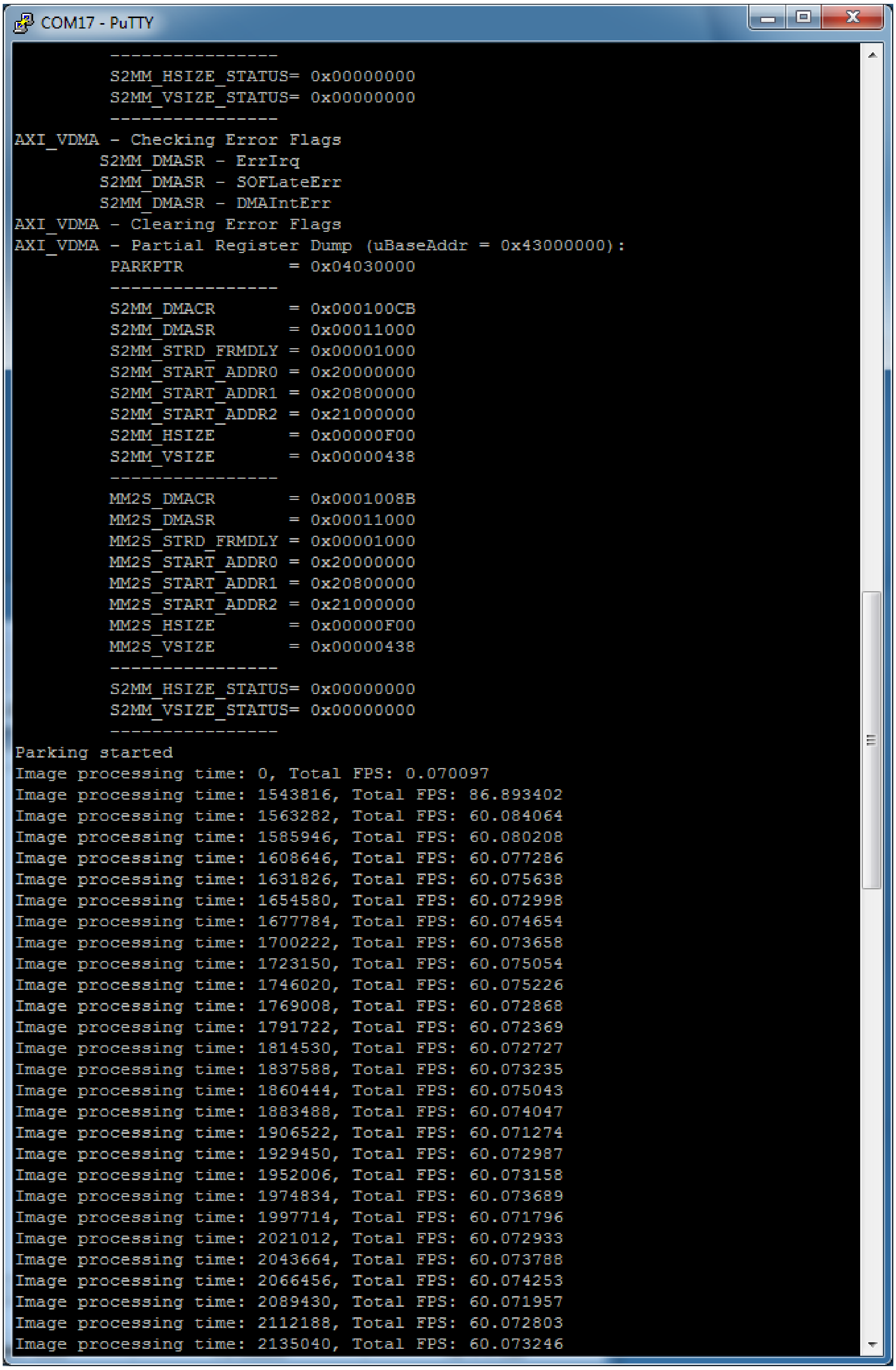
The demo performs initialisation and starts HW accelerated computation in four parallel data paths. See Figure 23 and Figure 24 for the listing. The image processing time is measured in number of clock cycles of the ARM 666 MHz clock. IT is increasing as the SW programmable number of processed micro-lines is increasing. It is increased by 1 in each new frame.

The output to the display is presented in *Figure 2*.



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

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

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

*****************************
*  Signal Processing Dept., *
*  UTIA AV CR, v.v.i.       *
*  HDMI In/Out IMAGEON      *
*  C:/S/t20i2h1/hio         *
*****************************


IIC IMAGEON Initialization ... OK
Initialize CDCE913 ...  148.500 MHz ... OK
HDMIO Initialization ... OK
Initialize Timing Controller ...  1920x1080p60  OK
Initialize VDMA ... Common Init ... TX Init ... OK
HDMII cable plugged
HDMII: Waiting for incoming video ... OK (LOCKED)
HDMII: ADV7611 Video Input Information
        Video Input      = DVI, Progressive
        Color Depth      = 8 bits per channel
        HSYNC Timing     = hav=1920, hfp=88, hsw=44(hsp=1), hbp=148
        VSYNC Timing     = vav=1080, vfp=04, vsw=05(vsp=1), vbp=036
        Video Dimensions = 1920 x 1080
        Pixel Clock      = 148.500000 MHz
        Frame rate       = 60.0 FPS
Initialize VDMA ... RX Init ... OK

Running sobel ....
AXI_VDMA - Partial Register Dump (uBaseAddr = 0x43000000):
         PARKPTR          = 0x00000000
         ----------------
         S2MM_DMACR       = 0x000100CB
         S2MM_DMASR       = 0x00014810
         S2MM_STRD_FRMDLY = 0x00001000
         S2MM_START_ADDR0 = 0x20000000
         S2MM_START_ADDR1 = 0x20800000
         S2MM_START_ADDR2 = 0x21000000
         S2MM_HSIZE       = 0x00000F00
         S2MM_VSIZE       = 0x00000438
         ----------------
         MM2S_DMACR       = 0x0001008B
         MM2S_DMASR       = 0x00011000
         MM2S_STRD_FRMDLY = 0x00001000
         MM2S_START_ADDR0 = 0x20000000
         MM2S_START_ADDR1 = 0x20800000
         MM2S_START_ADDR2 = 0x21000000
         MM2S_HSIZE       = 0x00000F00
         MM2S_VSIZE       = 0x00000438
         ----------------
```

Figure 23: Listing from so04 - first part.



```
         ----------------
         S2MM_HSIZE_STATUS= 0x00000000
         S2MM_VSIZE_STATUS= 0x00000000
         ----------------
AXI_VDMA - Checking Error Flags
        S2MM_DMASR - ErrIrq
        S2MM_DMASR - SOFLateErr
        S2MM_DMASR - DMAIntErr
AXI_VDMA - Clearing Error Flags
AXI_VDMA - Partial Register Dump (uBaseAddr = 0x43000000):
         PARKPTR           = 0x04030000
         ----------------
         S2MM_DMACR        = 0x000100CB
         S2MM_DMASR        = 0x00011000
         S2MM_STRD_FRMDLY  = 0x00001000
         S2MM_START_ADDR0  = 0x20000000
         S2MM_START_ADDR1  = 0x20800000
         S2MM_START_ADDR2  = 0x21000000
         S2MM_HSIZE        = 0x00000F00
         S2MM_VSIZE        = 0x00000438
         ----------------
         MM2S_DMACR        = 0x0001008B
         MM2S_DMASR        = 0x00011000
         MM2S_STRD_FRMDLY  = 0x00001000
         MM2S_START_ADDR0  = 0x20000000
         MM2S_START_ADDR1  = 0x20800000
         MM2S_START_ADDR2  = 0x21000000
         MM2S_HSIZE        = 0x00000F00
         MM2S_VSIZE        = 0x00000438
         ----------------
         S2MM_HSIZE_STATUS= 0x00000000
         S2MM_VSIZE_STATUS= 0x00000000
         ----------------
Parking started
Image processing time: 0, Total FPS: 0.070097
Image processing time: 1543816, Total FPS: 86.893402
Image processing time: 1563282, Total FPS: 60.084064
Image processing time: 1585946, Total FPS: 60.080208
Image processing time: 1608646, Total FPS: 60.077286
Image processing time: 1631826, Total FPS: 60.075638
Image processing time: 1654580, Total FPS: 60.072998
Image processing time: 1677784, Total FPS: 60.074654
Image processing time: 1700222, Total FPS: 60.073658
Image processing time: 1723150, Total FPS: 60.075054
Image processing time: 1746020, Total FPS: 60.075226
Image processing time: 1769008, Total FPS: 60.072868
Image processing time: 1791722, Total FPS: 60.072369
Image processing time: 1814530, Total FPS: 60.072727
Image processing time: 1837588, Total FPS: 60.073235
Image processing time: 1860444, Total FPS: 60.075043
Image processing time: 1883488, Total FPS: 60.074047
Image processing time: 1906522, Total FPS: 60.071274
Image processing time: 1929450, Total FPS: 60.072987
Image processing time: 1952006, Total FPS: 60.073158
Image processing time: 1974834, Total FPS: 60.073689
Image processing time: 1997714, Total FPS: 60.071796
Image processing time: 2021012, Total FPS: 60.072933
Image processing time: 2043664, Total FPS: 60.073788
Image processing time: 2066456, Total FPS: 60.074253
Image processing time: 2089430, Total FPS: 60.071957
Image processing time: 2112188, Total FPS: 60.072803
Image processing time: 2135040, Total FPS: 60.073246
```

Figure 24: Listing from so04 - next part.

Additional notes about included demos:

- The edge detection demo sh01_rows_fixed_100 works on complete frame with single HW accelerator data path.
- The edge detection demo sh01_rows_resize_25_to_100 works with identical HW as demo sh01_rows_fixed_100.
    - 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.
    - SW scales dynamically the number of micro-lines to be processed.
    - It is tuned 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 edge detection demos sh02_rows_fixed_100 and sh02_rows_resize_25_to_100 work with 2 data paths. The edge detection demos sh03_rows_fixed_100 and sh03_rows_resize_25_to_100 work with 3 data paths.
- Demos sh01, sh02 or sh03 are linked with static libraries libsh01.a, libsh02.a or libsh03.a.
- Demos so01, so02, so03 or so04 are linked with static libraries libso01.a, libso02, libso03.a or libso04.a. Demos use slightly different programming model to support execution of user defined, synchronous parallel code on ARM.
- Motion detection demos md01_rows_fixed_100 and md02_rows_fixed_100 work with one or two HW video processing chains. These HW chains have only fixed set of processed lines (1x 100% or 2x 50% of the Full HD frame).
- Demos md01 and md02 are linked with static libraries libmd01.a, or libmd02.a.

Files for the SD card (SW implementation of video processing algorithms on ARM in SDSoC 2015.4) can be found in:

SD_cards\SW\sh01\BOOT.bin
SD_cards\SW\sh02\BOOT.bin
SD_cards\SW\sh03\BOOT.bin

SD_cards\SW\so01\BOOT.bin
SD_cards\SW\so02\BOOT.bin
SD_cards\SW\so03\BOOT.bin
SD_cards\SW\so04\BOOT.bin

SD_cards\SW\md01\BOOT.bin
SD_cards\SW\md02\BOOT.bin

These files can be used for evaluation of the system performance in case of sequential SW computation on ARM Cortex A9 processor without HW acceleration. Projects have been compiled with maximal optimisation (-O3) without use of NEON.

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

This section describes synchronisation of ARM C code with Video processing accelerators.
Two cases or programming models are described.

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

Internal synchronisation with parallel HW data paths

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\t20i2h1\sh03_rows_fixed_100\sobel\img_filters.c

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

void img_process(unsigned short *in1, unsigned short *out1,
                 unsigned short *in2, unsigned short *out2,
                 unsigned short *in3, unsigned short *out3) {

     _p0_sobel_filter_htile1_0(in1, out1, NUMROWS);
     _p0_sobel_filter_htile2_0(in2, out2, NUMROWS);

//
//    Some user defined sequential C code for ARM can be inserted here.
//
//    ARM C code will run in parallel with HW path 1 and HW path 2.
//    However, start of the synchronising HW path 3 is delayed.
//
//    This programming style can be used,
//    if HW path 1 and HW path 2 are long and
//    HW path 3 together with the sequential ARM C code has together
//    similar total length(in terms of clock cycles)
//    as HW path 1 or HW path 2
//

     _p0_sobel_filter_htile3_0(in3, out3, NUMROWS);
}
```

Table 1: Listing of ARM C function using the internal synchronisation with parallel HW data paths.

The three called functions

```
_p0_sobel_filter_htile1_0()   // Not blocking, Starts HW path 1
_p0_sobel_filter_htile2_0()   // Not blocking, Starts HW path 2
_p0_sobel_filter_htile3_0()   // Blocking, Starts HW path 3 waits for 1,2,3.
```

correspond to the three HW video acceleration paths. These functions are NOT independent. Functions have to be called in the described fixed order. Each of functions starts its HW data path. However, only the third (last) function is blocking and waits internally for all 3 HW data paths to finish processing and return their results to their section of the output video frame buffer.

All three functions have been defined in the original SDSoC 2015.4 project and exported in the libsh03.a static library.

ARM processor is waiting inside of the last call function on this synchronisation point and it cannot be used easily for computation of user defined C code. One possible solution is described next.

User defined synchronisation with parallel HW data paths

Consider **so03_rows_fixed_100** project as an example of this alternative programming style. Three HW data paths will perform edge detection in parallel on 3 separate areas of a DDR3 video frame. ARM code is calling function (see Table 2):

C:\VM_07\t20i2h1\so03_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(1)
     _p0_sobel_filter_htile_2(fb_in,
                               fb_out,
                               NUMTILEROWS);
#pragma SDS async(2)
     _p0_sobel_filter_htile_1(fb_in + NUMTILEROWS*NUMPADCOLS,
                               fb_out + NUMTILEROWS*NUMPADCOLS,
                               NUMTILEROWS);
#pragma SDS async(3)
     _p0_sobel_filter_htile_0(fb_in + 2*NUMTILEROWS*NUMPADCOLS,
                               fb_out + 2*NUMTILEROWS*NUMPADCOLS,
                               NUMTILEROWS);
//
//    USER C code can be inserted here to run in parallel with HW paths.
//    For optimal use, the computing time of each of
//    the three HW paths should be similar as the computing time
//    of the sequential C code executed in parallel on ARM processor here.
//
     sds_wait(1);
     sds_wait(2);
     sds_wait(3);
}
```

Table 2: Listing of ARM C function with user-defined synchronisation of parallel HW data paths.

The three called functions

```
_p0_sobel_filter_htile_2()   // Not blocking, Starts HW path 1
_p0_sobel_filter_htile_1()   // Not blocking, Starts HW path 2
_p0_sobel_filter_htile_0()   // Not Blocking, Starts HW path 3
```

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 libso03.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(1); sds_wait(2); sds_wait(3);`**. 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 in this case.

Demos supporting synchronous execution of user C on ARM in parallel with accelerated HW:

- The edge detection demo so01_rows_fixed_100 works on complete frame with single HW accelerator data path. The edge detection demos so01_rows_resize_25_to_100 works with identical HW as demo so01_rows_fixed_100.
    - 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.
    - SW scales dynamically the number of micro-lines to be processed.
    - It is tuned 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 edge detection demos so02_rows_fixed_100 and so02_rows_resize_25_to_100 work with 2 data paths.
- The edge detection demos so03_rows_fixed_100 and so03_rows_resize_25_to_100 work with 3 data paths.
- The edge detection demos so04_rows_fixed_100 and so04_rows_resize_25_to_100 work with 4 data paths.
- All these demos support synchronised parallel execution of user defined C code on ARM while the HW data paths perform accelerated video processing.
- Demos are linked with static libraries libso01.a, libso02.a, libso03.a or libso04.a.
- Block diagrams, area and acceleration results of edge detection demos so01, so02 and so03 are practically identical to the corresponding demos described in sections 1.3, 1.4, 1.5.

3. 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; FMC card with HDMI I/O and CAM interface.
http://products.avnet.com/shop/en/ema/3074457345623664802

[5] 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

[6] 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

4. Evaluation license

The **evaluation version of the package** can be downloaded from UTIA www pages free of charge for evaluation of HW accelerated edge detection and motion detection algorithms.

The evaluation package includes SDK 2015.4 SW projects with C source code for ARM Cortex A9 processor (32bit) in standalone mode.

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
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
libso01.a	SDSoC 2015.4 static library for HW accelerator in project so01
libso02.a	SDSoC 2015.4 static library for HW accelerator in project so02
libso03.a	SDSoC 2015.4 static library for HW accelerator in project so03
libso04.a	SDSoC 2015.4 static library for HW accelerator in project so04
libmd01.a	SDSoC 2015.4 static library for HW accelerator in project md01
libmd02.a	SDSoC 2015.4 static library for HW accelerator in project md02

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

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.