Application Note

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

# Full HD HDMI In-Out HW-Accelerated Demos for Zynq System-on-Module TE0715-03-30-1I and Sundance EMC2-DP-V2 Platform

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

Revision history:

| Rev. | Date | Author | Description |
|---|---|---|---|
| 1 | 15.07.2016 | Jiří Kadlec | Sundance EMC2-DP-V2, Xilinx SDK 2015.4 |
| 2 | 18.07.2016 | Jiří Kadlec | SD cards with compiled SDSoC SW projects |
| 3 | 20.07.2016 | Jiří Kadlec | SoM is fixed to TE0715-03-30-1I, Typos fixed. |
| 4 | 21.07.2016 | Jiří Kadlec | Typos fixed. |

Acknowledgements:

This work has been partially supported by the ARTEMIS JU project EMC2 “Embedded Multi-Core Systems for Mixed Criticality Applications in Dynamic and Changeable Real-Time Environments”, project number ARTEMIS JU 621429 and 7H14005 (Ministry of Education Youth and Sports of the Czech Republic). See **http://www.emc2-project.eu/**.

# 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 Sundance EMC2-DP-V2 platform [3] with industrial grade Zynq XC7Z030-1I device on System on Module TE0715-03-30-1I [1]. All demonstrated video processing algorithms have been developed, debugged and tested in Xilinx SDSoC 2015.4 environment [6] for the Sundance EMC2-DP-V2 platform [3]. 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. Application note also describes 3 edge detection algorithms coded in format supporting user defined ARM C code which is executing in parallel with the video processing HW paths.

**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] for the Sundance EMC2-DP-V2 platform [3].
- To demonstrate on the EMC2-DP-V2 platform the HW accelerated video processing algorithms and the speedup against reference SW versions running on ARM.
- To demonstrate parallel processing of predefined video processing HW paths executing in parallel with user defined C code on ARM.

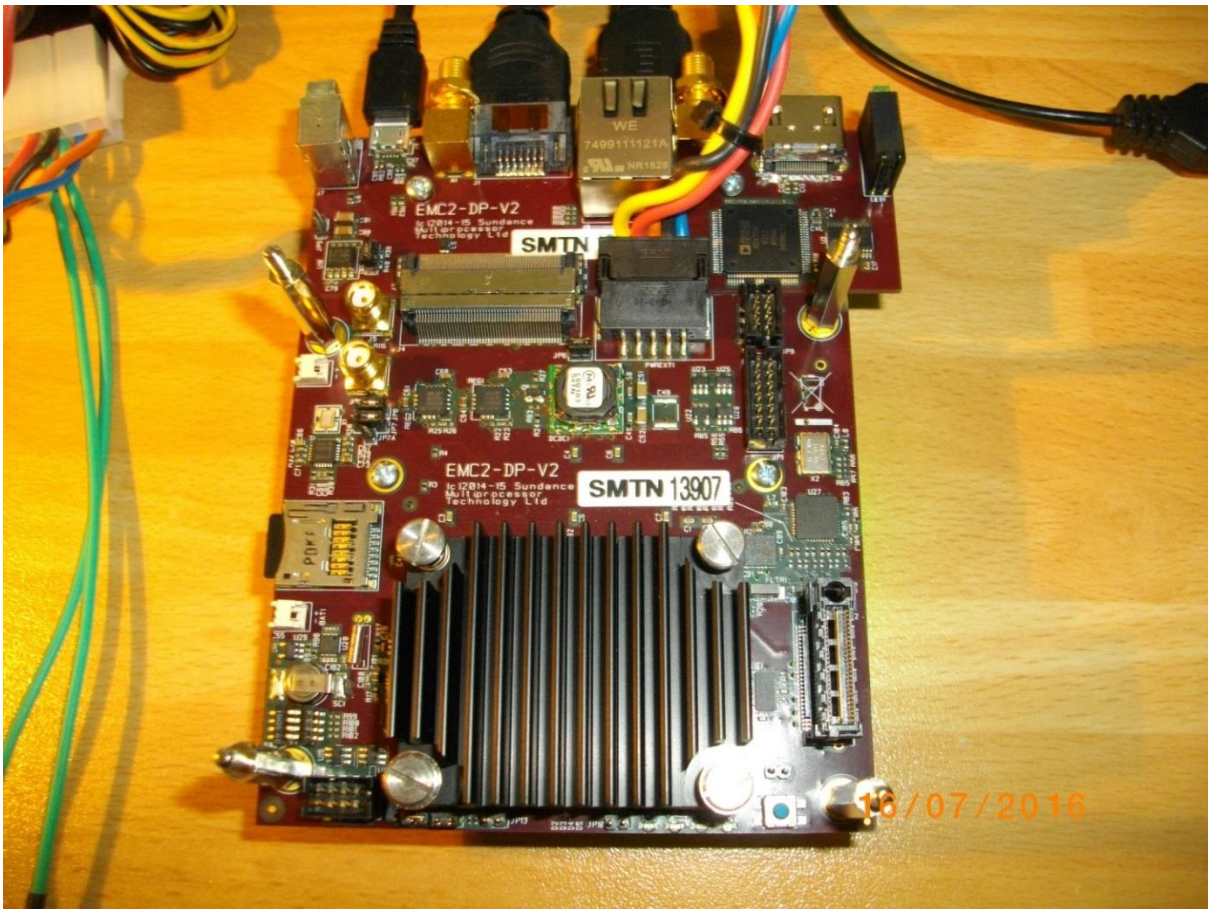


*Figure 1: Sundance EMC2-DP-V2 platform with Zynq XC7Z030-1I device and HDMII-HDMIO support.*

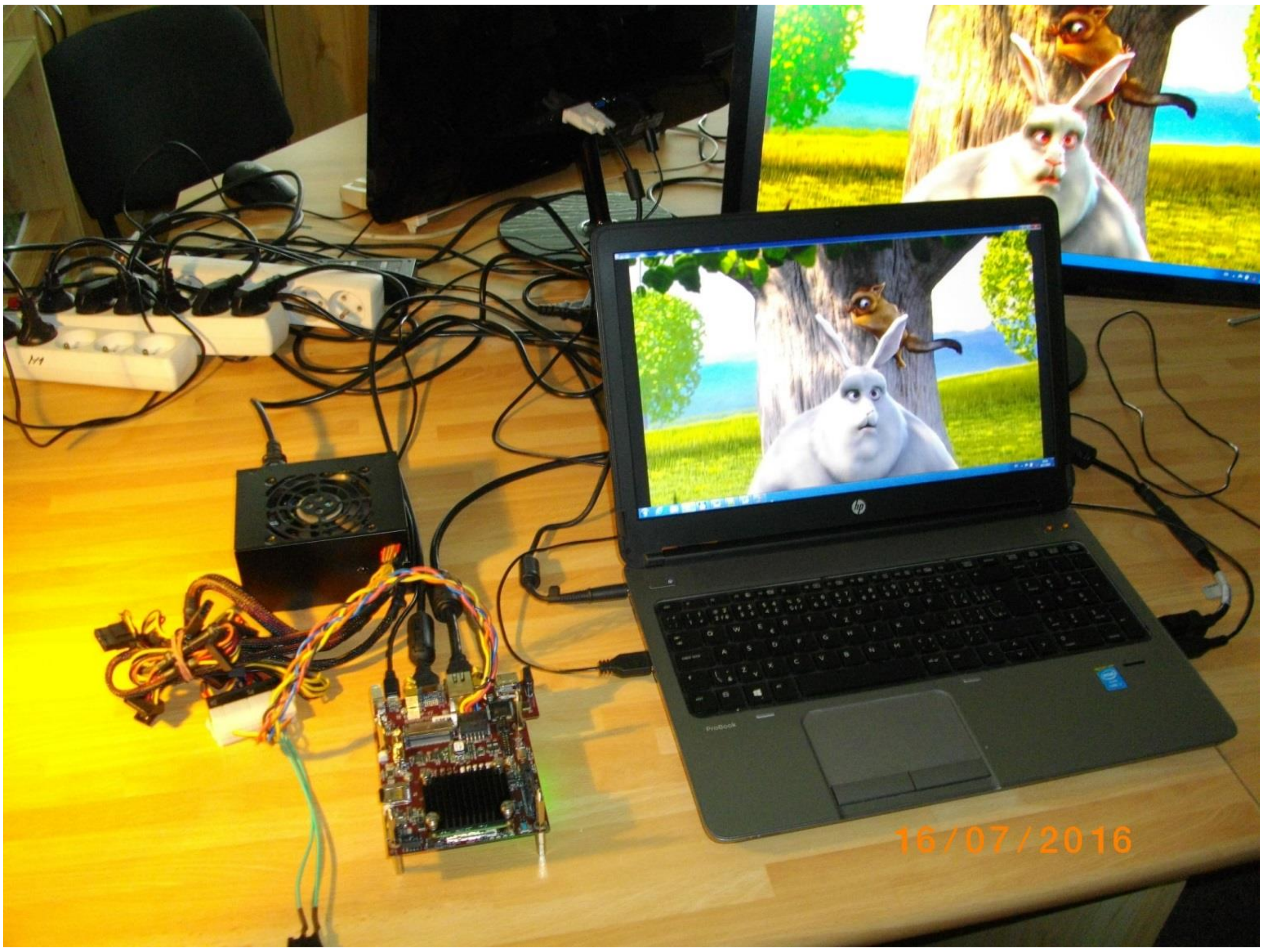


*Figure 2: HW accelerated motion detection in Full HD with 60 FPS video processing speed.*

Notice the fast moving edges in the face of the rabbit. Edges are marked by red pixels in the output display.

**Common setup for all included demos:**

- ARM Cortex A9 processor of Xilinx Zynq device XC7Z030-1I executes standalone C application programs performing initialisation and synchronisation of the HW accelerated video processing chain.
- Enclosed C programs can be modified by the user and recompiled in Xilinx SDK 2015.4.
- Compiled demos boot from the micro SD card directly after the power ON.

- Raw 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 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 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. 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.

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 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**, **md02** and **md03** provide accelerated HW computation with 1, 2 or 3 parallel HW data paths. HW demos are using 2, 4 or 6 DMA HW channels for reading from **two** sub sequent video frame buffers located both in the DDR3 to 1, 2 or 3 video processing chains of accelerators performing the motion detection.
Another 1, 2 or 3 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 1.6, 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.

Performance of SW version of algorithms can be evaluated by booting of the Sundance EMC2-DP-V2 platform from 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 Sundance EMC2-DP-V2 platform are not included. Micro SD card BOOT.bin files are enclosed.

## 1.3 Project sh01: Edge detection with single HW accelerator

| | |
|---|---|
| Zynq module: | TE0715-03-30-1I (Trenz) |
| Carrier board: | EMC2-DP2 (Sundance) |
| 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: | Sobel edge detection with single data path |
| Platform: | C:\S\s30i1h1\hio |
| Demos(SDK2015.4): | C:\VM07\s30i1h1\sh01_rows_resize_25_to_100.c |
| | C:\VM07\s30i1h1\sh01_rows_fixed_100.c |

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_07_15

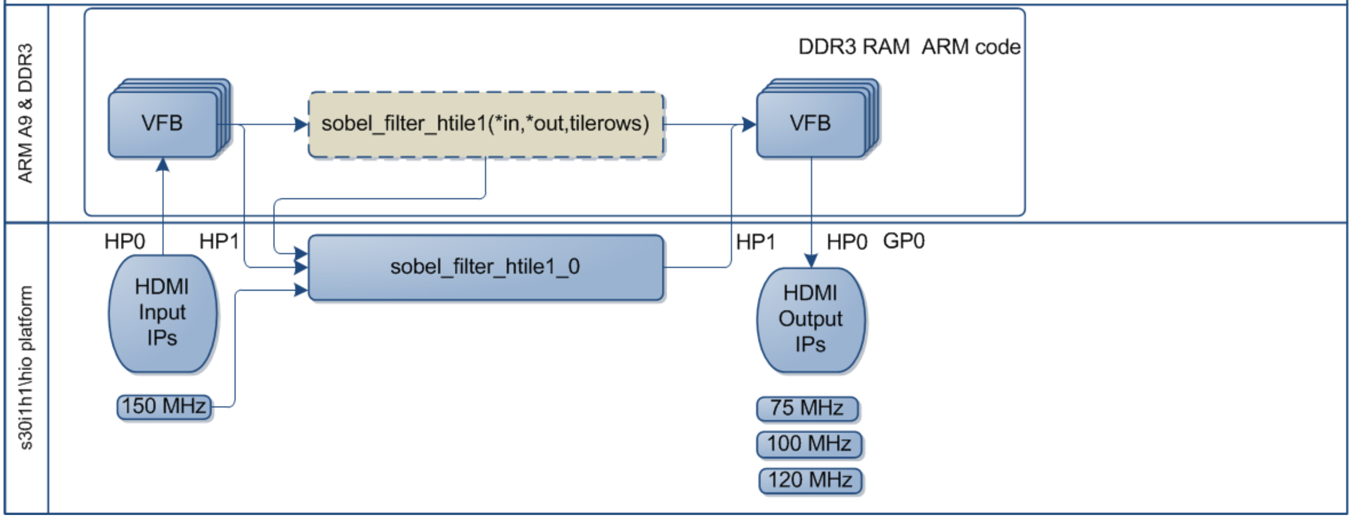
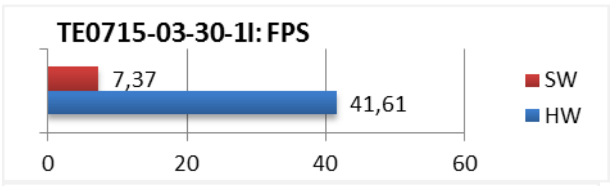


*Figure 3: Project sh01 - Edge detection with single HW accelerator.*

**TE0715-03-30-1I Sobel 1x**

**Acceleration by HW: 5.64 x**



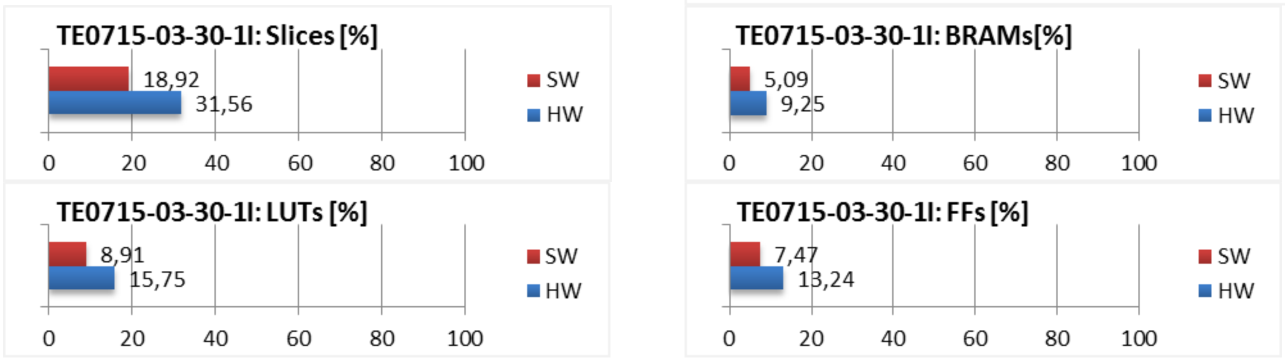
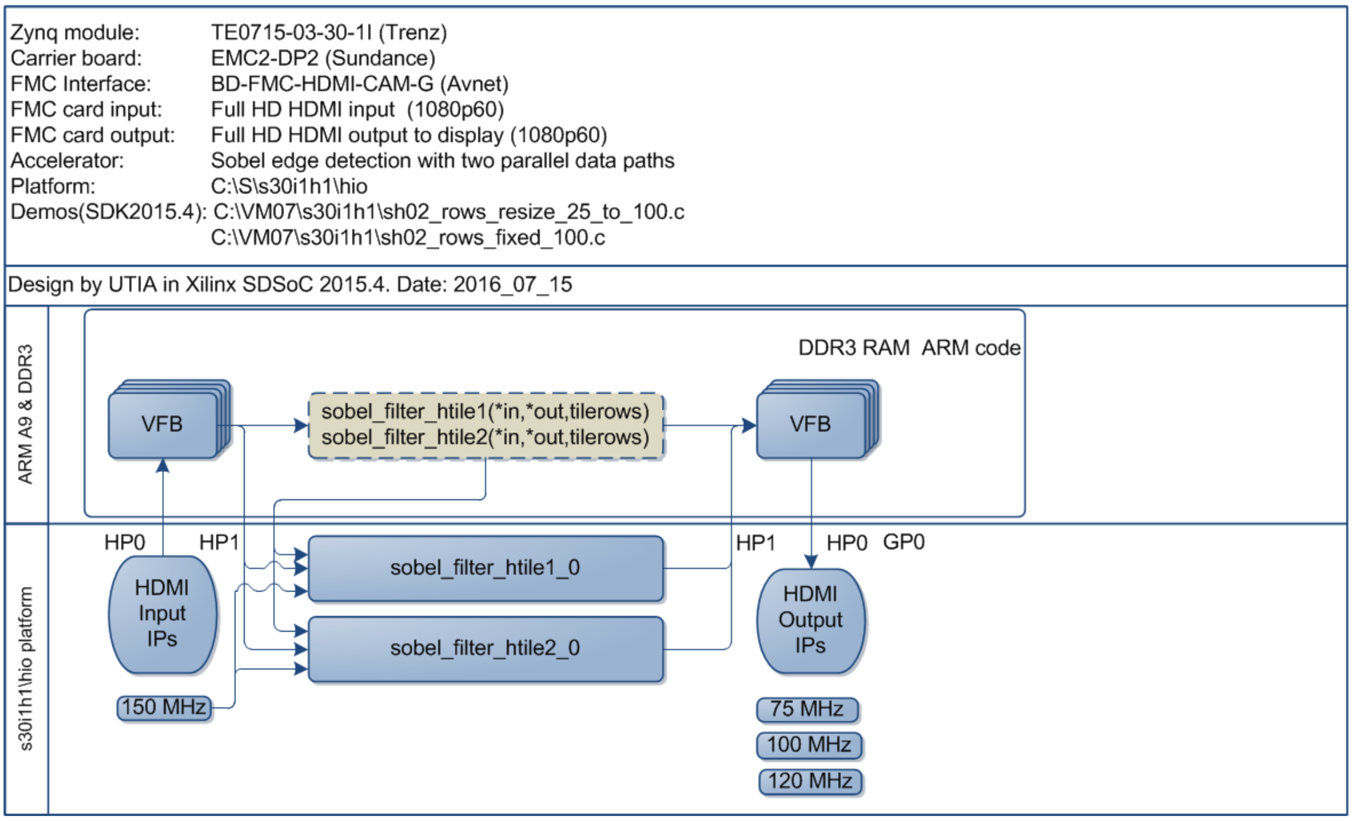


*Figure 4: Project sh01 - Acceleration and HW resources used.*

## 1.4 Project sh02: Edge detection with two HW accelerators



| | |
|---|---|
| Zynq module: | TE0715-03-30-1I (Trenz) |
| Carrier board: | EMC2-DP2 (Sundance) |
| 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: | Sobel edge detection with two parallel data paths |
| Platform: | C:\S\s30i1h1\hio |
| Demos(SDK2015.4): | C:\VM07\s30i1h1\sh02_rows_resize_25_to_100.c |
| | C:\VM07\s30i1h1\sh02_rows_fixed_100.c |

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_07_15

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

**TE0715-03-30-1I Sobel 2x**

**Acceleration by HW: 8.14 x**

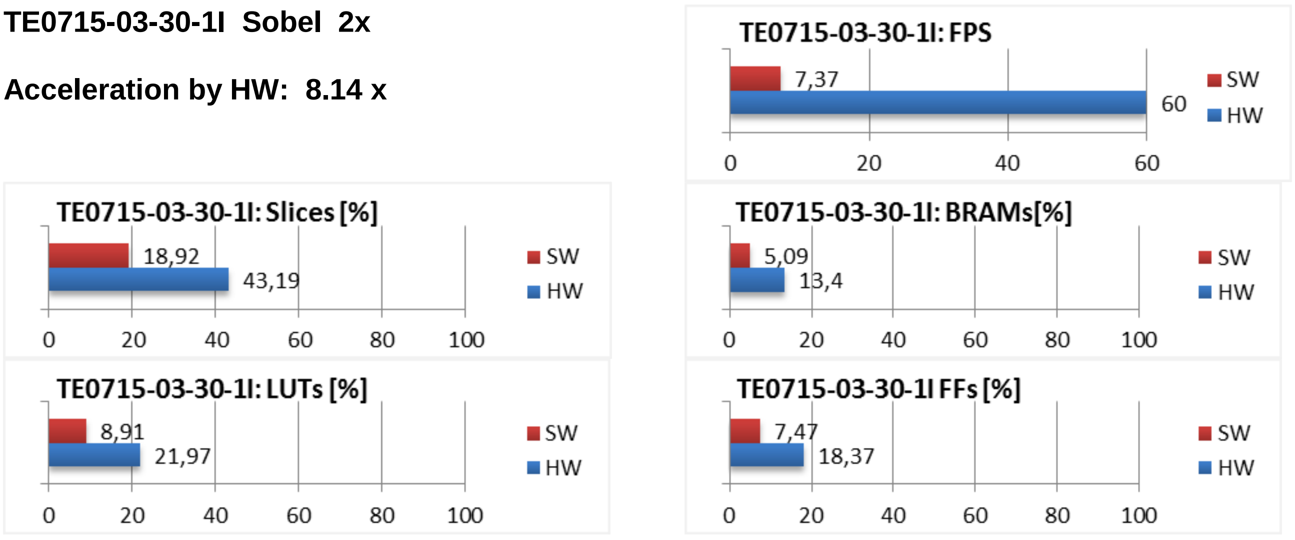
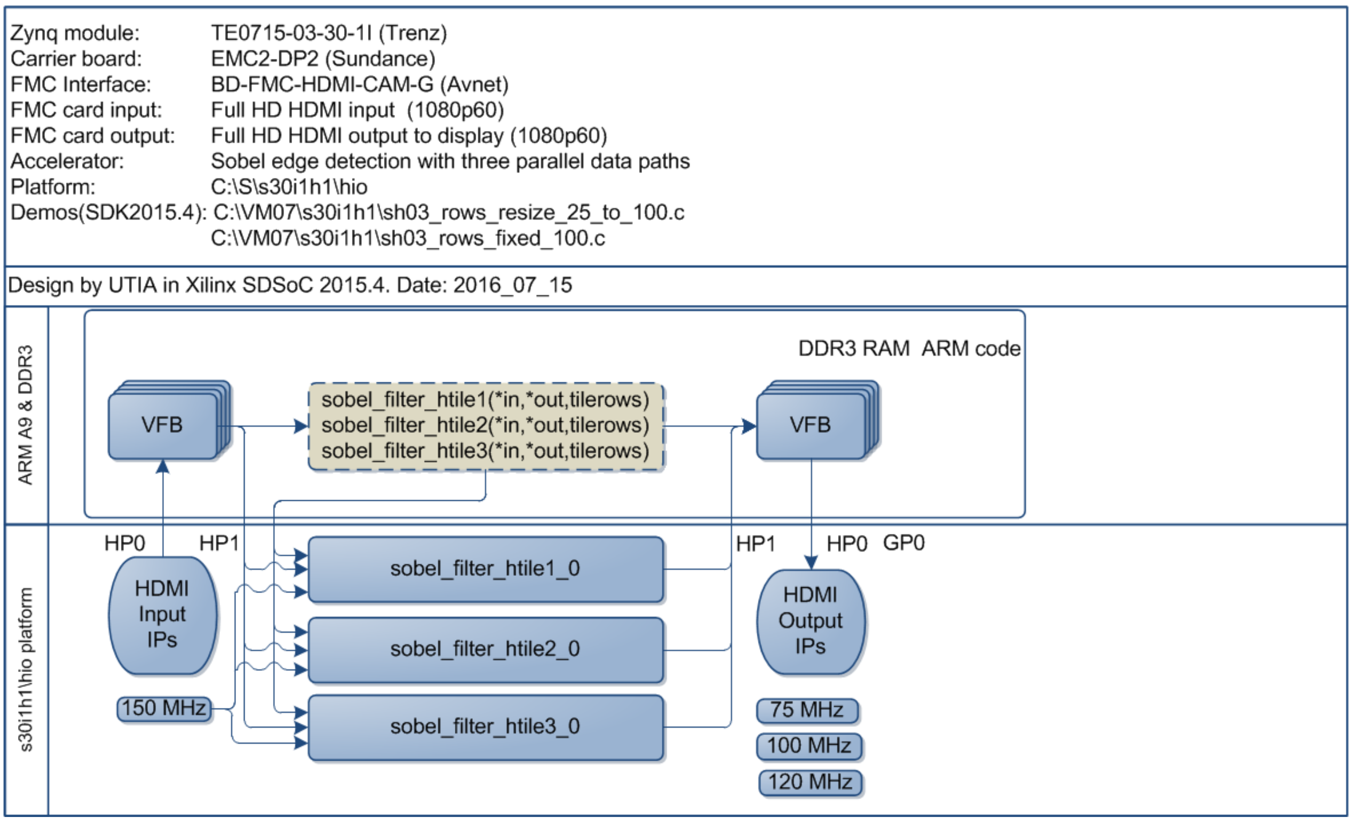


Figure 6: Project sh02 – Acceleration and HW resources used.

## 1.5 Project sh03: Edge detection with three HW accelerators



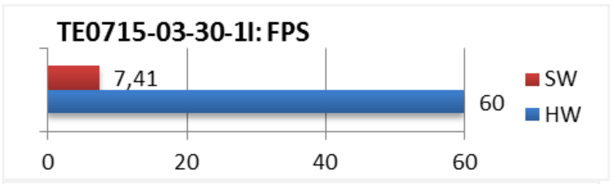
| | |
|---|---|
| Zynq module: | TE0715-03-30-1I (Trenz) |
| Carrier board: | EMC2-DP2 (Sundance) |
| FMC Interface: | BD-FMC-HDMI-CAM-G (Avnet) |
| FMC card input: | Full HD HDMI input (1080p60) |
| FMC card output: | Full HD HDMI output to display (1080p60) |
| Accelerator: | Sobel edge detection with three parallel data paths |
| Platform: | C:\S\s30i1h1\hio |
| Demos(SDK2015.4): | C:\VM07\s30i1h1\sh03_rows_resize_25_to_100.c |
| | C:\VM07\s30i1h1\sh03_rows_fixed_100.c |

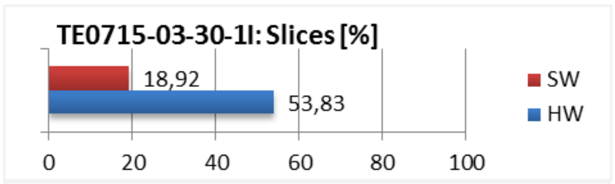
Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_07_15

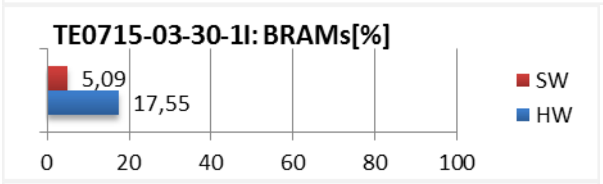
*Figure 7: Project sh03 - Edge detection with three HW accelerators.*

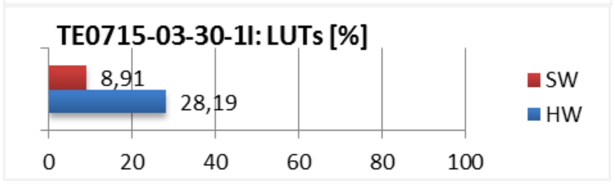
**TE0715-03-30-1I Sobel 3x**

**Acceleration by HW: 8.09 x**









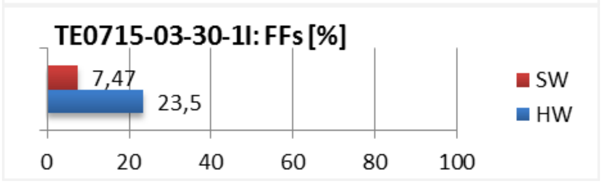
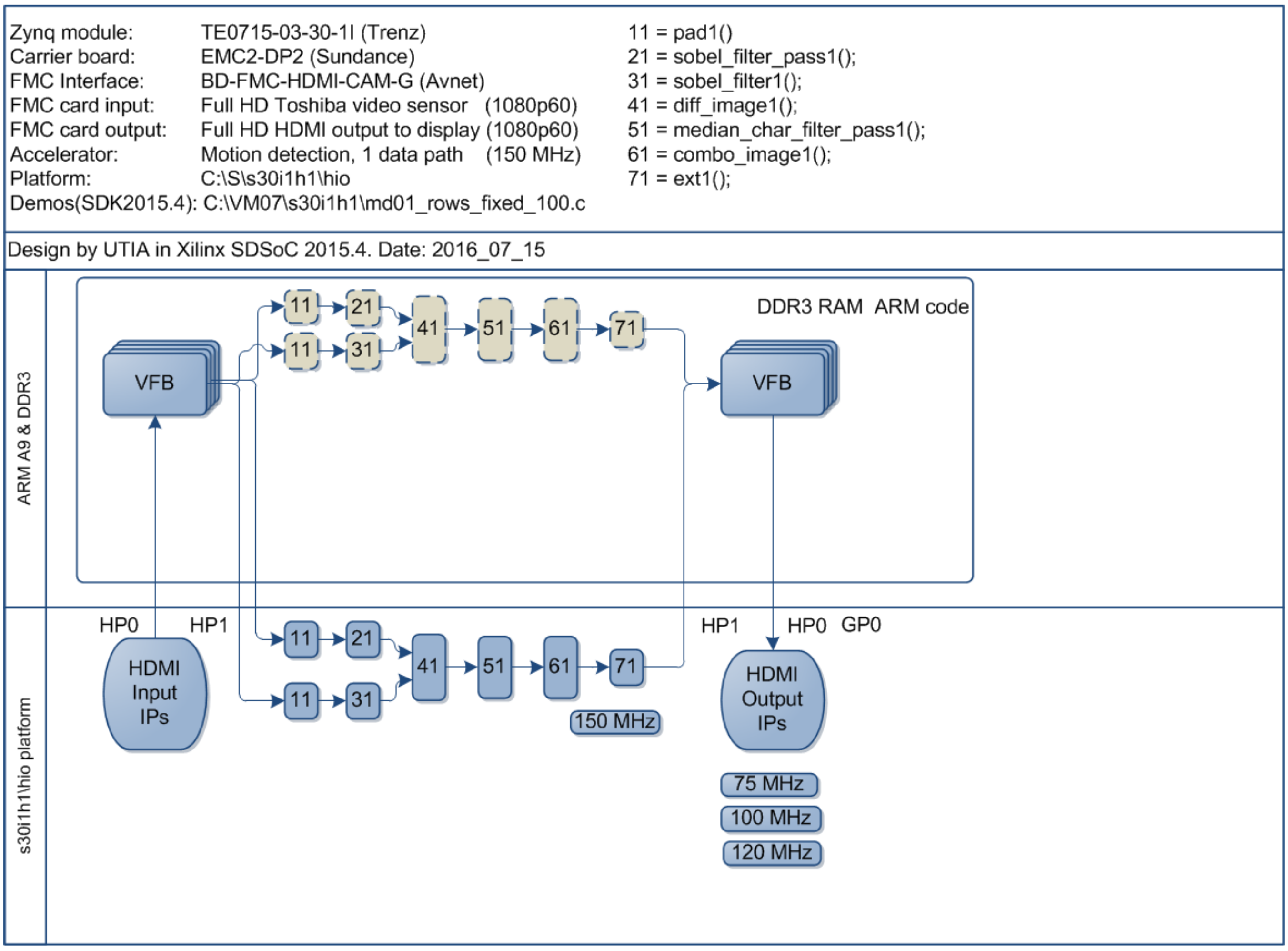


*Figure 8: Project sh03 - Acceleration and HW resources used.*

## 1.6 Project md01: Motion detection with single chain of HW accelerators



| | | |
|---|---|---|
| Zynq module: | TE0715-03-30-1I (Trenz) | 11 = pad1() |
| Carrier board: | EMC2-DP2 (Sundance) | 21 = sobel_filter_pass1(); |
| FMC Interface: | BD-FMC-HDMI-CAM-G (Avnet) | 31 = sobel_filter1(); |
| FMC card input: | Full HD Toshiba video sensor (1080p60) | 41 = diff_image1(); |
| FMC card output: | Full HD HDMI output to display (1080p60) | 51 = median_char_filter_pass1(); |
| Accelerator: | Motion detection, 1 data path (150 MHz) | 61 = combo_image1(); |
| Platform: | C:\S\s30i1h1\hio | 71 = ext1(); |
| Demos(SDK2015.4): | C:\VM07\s30i1h1\md01_rows_fixed_100.c | |

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_07_15

*Figure 9: Project md01 - Motion detection with single HW accelerator data path.*

**TE0715-03-30-1I Motion Detection 1x**

**Acceleration by HW: 31.2 x**

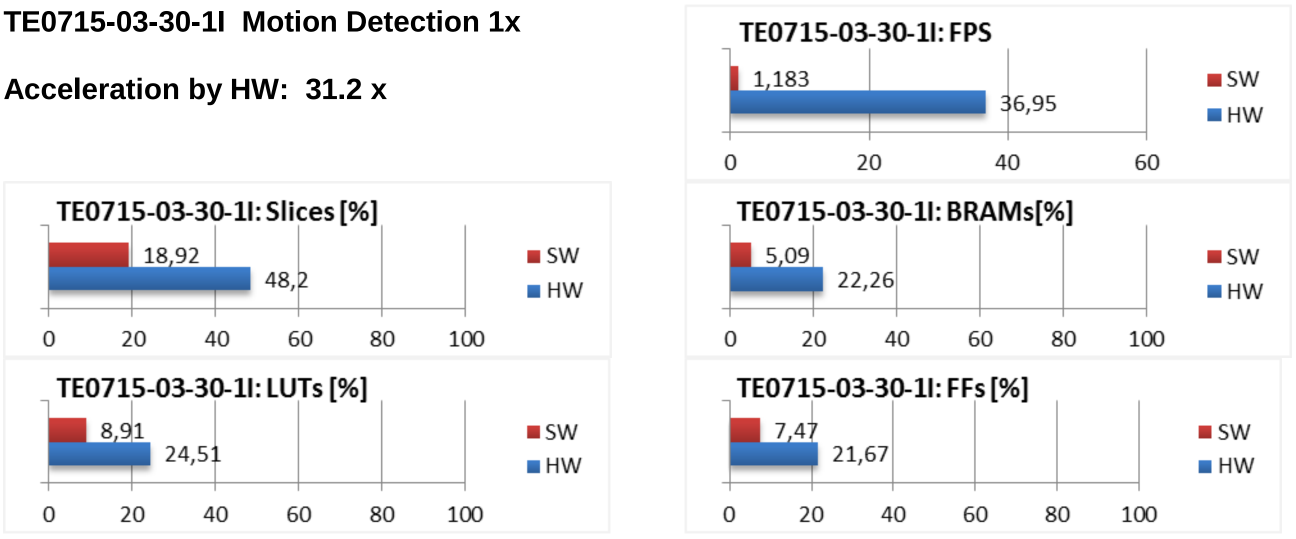


*Figure 10: Project md01 - Acceleration and HW resources used.*

## *1.7 Project md02: Motion detection with two chains of HW accelerators*

| | | |
|---|---|---|
| Zynq module: | TE0715-03-30-1I (Trenz) | 1[1,2] = pad[1,2]() |
| Carrier board: | EMC2-DP2 (Sundance) | 2[1,2] = sobel_filter_pass[1,2](); |
| FMC Interface: | BD-FMC-HDMI-CAM-G (Avnet) | 3[1,2] = sobel_filter[1,2](); |
| FMC card input: | Full HD Toshiba video sensor (1080p60) | 4[1,2] = diff_image[1,2](); |
| FMC card output: | Full HD HDMI output to display (1080p60) | 5[1,2] = median_char_filter_pass[1,2,3](); |
| Accelerator: | Motion detection, 2 data paths (150 MHz) | 6[1,2] = combo_image[1,2](); |
| Platform: | C:\S\s30i1h1\hio | 7[1,2] = ext[1,2](); |
| Demos(SDK2015.4): | C:\VM07\s30i1h1\md02_rows_fixed_100.c | |

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_07_15

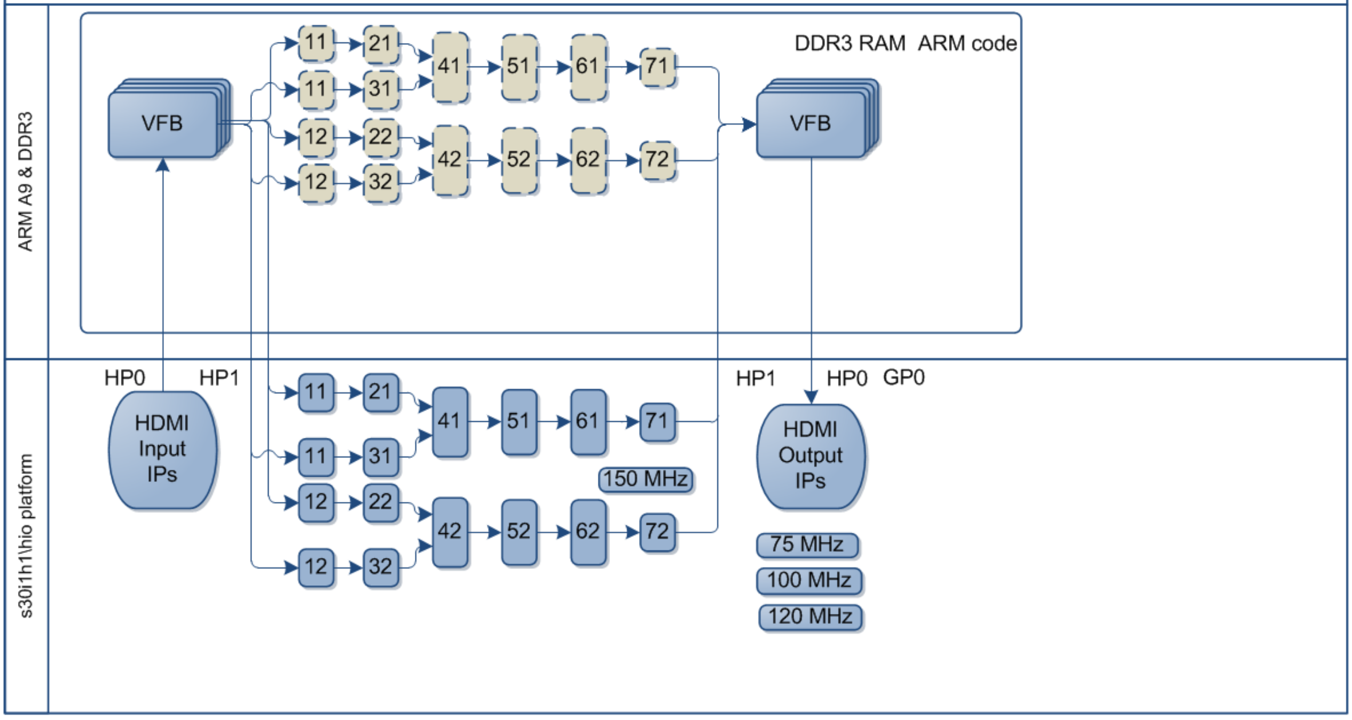
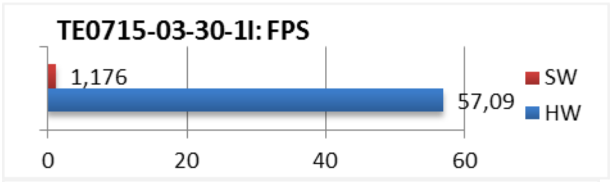
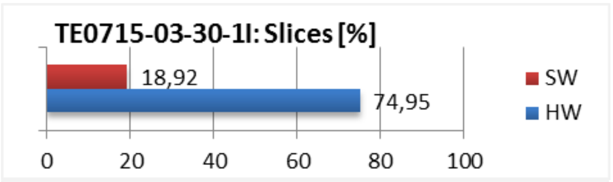


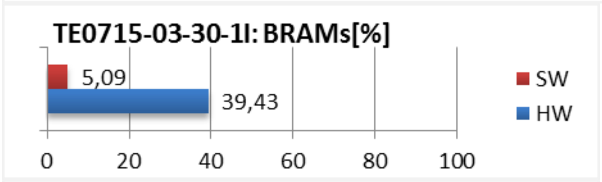
*Figure 11: Project md02 - Motion detection with two HW accelerator data paths.*

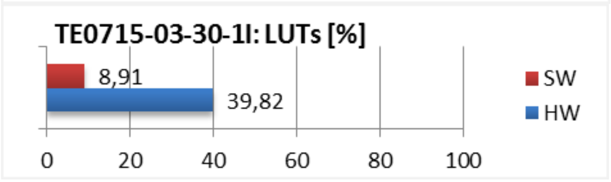
**TE0715-03-30-1I Motion Detection 2x**

**Acceleration by HW: 48.5 x**









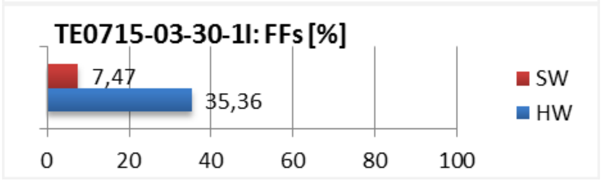
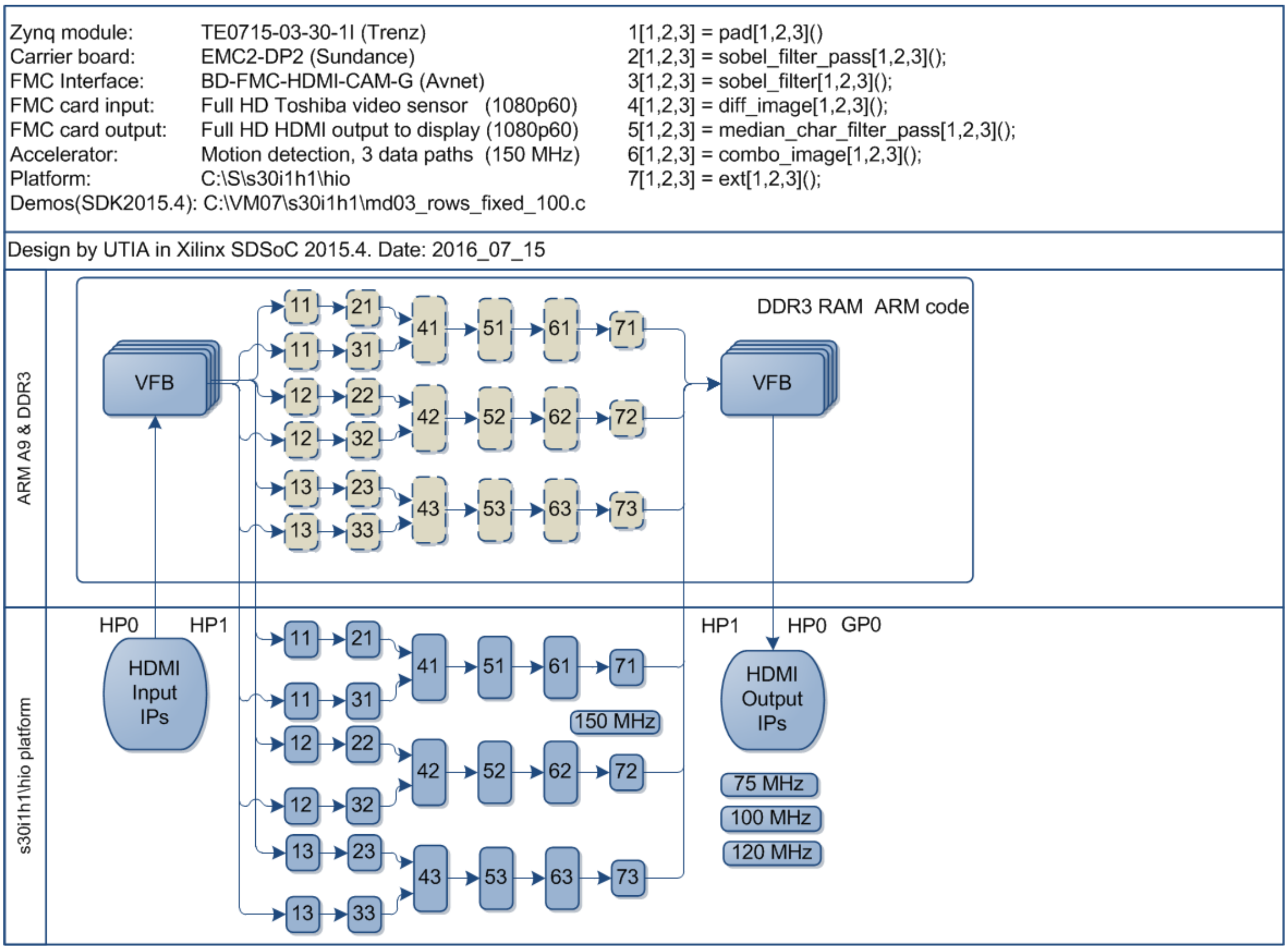


*Figure 12: Project md02 - Acceleration and HW resources used.*

## 1.8 Project md03: Motion detection with three chains of HW accelerators



| | | |
|---|---|---|
| Zynq module: | TE0715-03-30-1I (Trenz) | 1[1,2,3] = pad[1,2,3]() |
| Carrier board: | EMC2-DP2 (Sundance) | 2[1,2,3] = sobel_filter_pass[1,2,3](); |
| FMC Interface: | BD-FMC-HDMI-CAM-G (Avnet) | 3[1,2,3] = sobel_filter[1,2,3](); |
| FMC card input: | Full HD Toshiba video sensor (1080p60) | 4[1,2,3] = diff_image[1,2,3](); |
| FMC card output: | Full HD HDMI output to display (1080p60) | 5[1,2,3] = median_char_filter_pass[1,2,3](); |
| Accelerator: | Motion detection, 3 data paths (150 MHz) | 6[1,2,3] = combo_image[1,2,3](); |
| Platform: | C:\S\s30i1h1\hio | 7[1,2,3] = ext[1,2,3](); |
| Demos(SDK2015.4): | C:\VM07\s30i1h1\md03_rows_fixed_100.c | |

Design by UTIA in Xilinx SDSoC 2015.4. Date: 2016_07_15

Figure 13: Project md03 - Motion detection with tree HW accelerator data paths.

**TE0715-03-30-1I Motion Detection 3x**

**Acceleration by HW: 51.06 x**

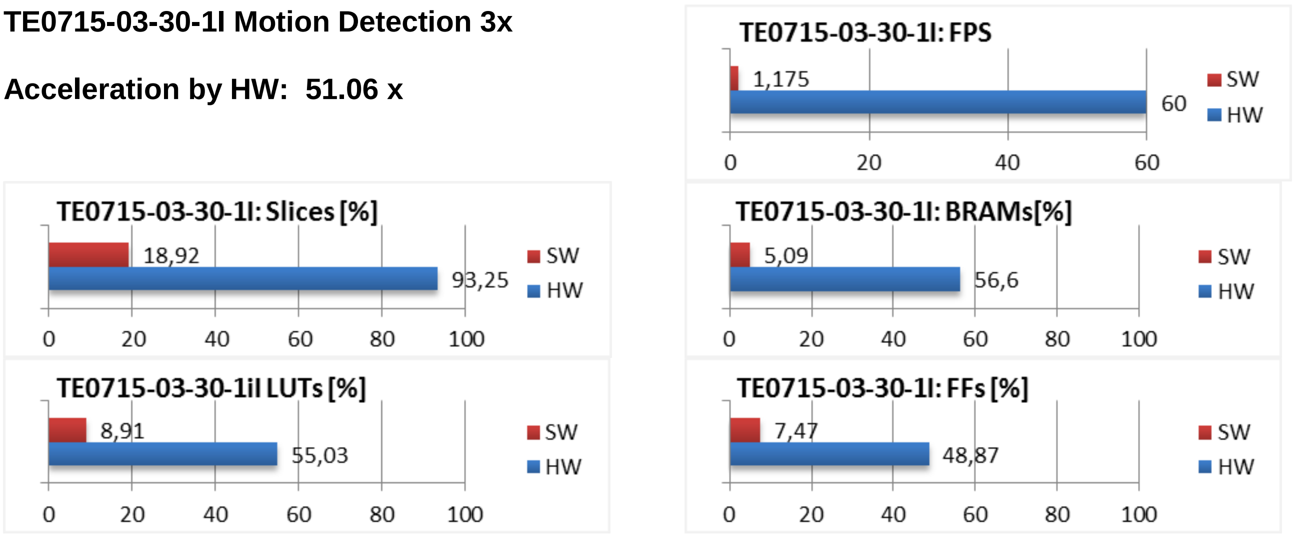


Figure 14: Project md03 - Acceleration and HW resources used.

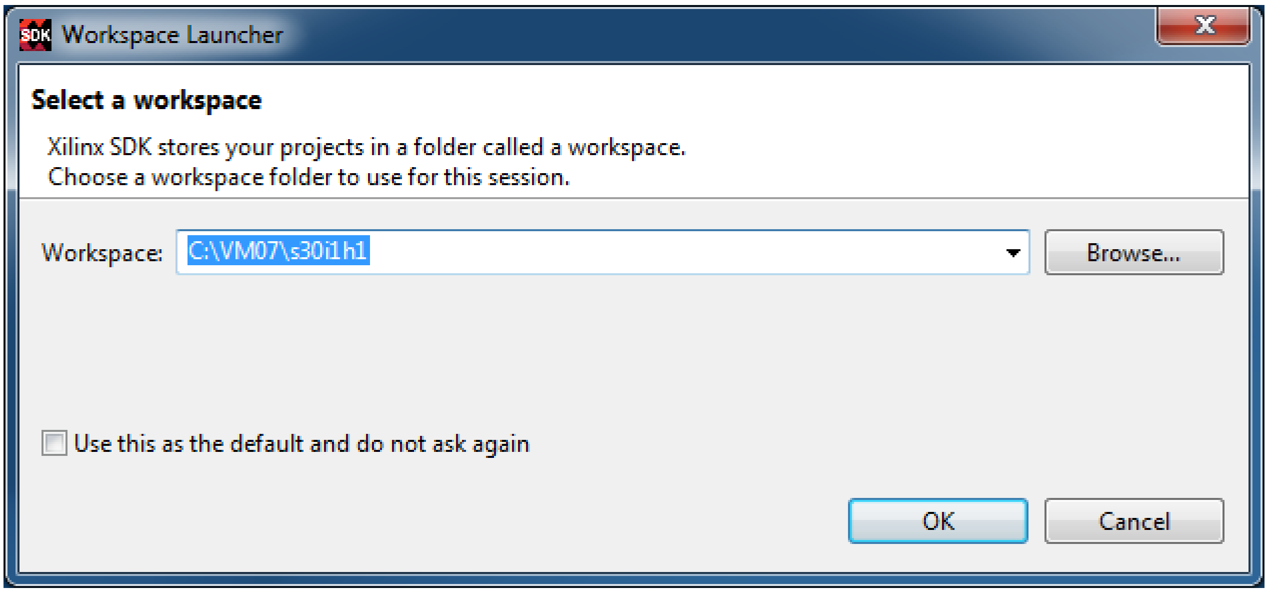
# 2. Installation of evaluation package

## 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\s30i1h1_V54_IMPORT**

Create empty directory for Xilinx SDK workspace.
**C:\VM_07\s30i1h1**

Start Xilinx SDK 2015.4 and select the directory for the SDK 2015.4 workspace. See Figure 15. Select **C:\VM_07\s30i1h1**


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\s30i1h1

☐ Use this as the default and do not ask again

*Figure 15: 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 16.

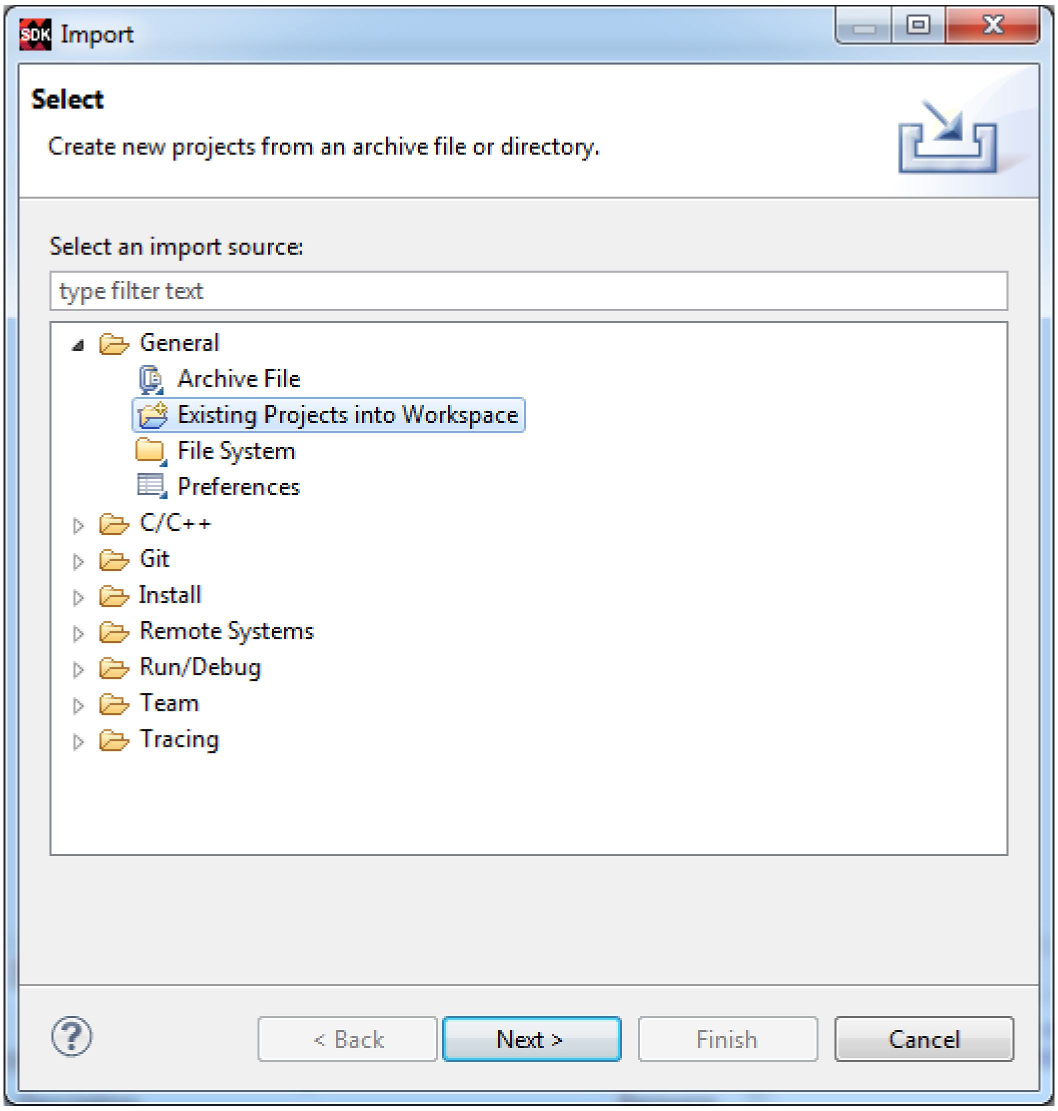


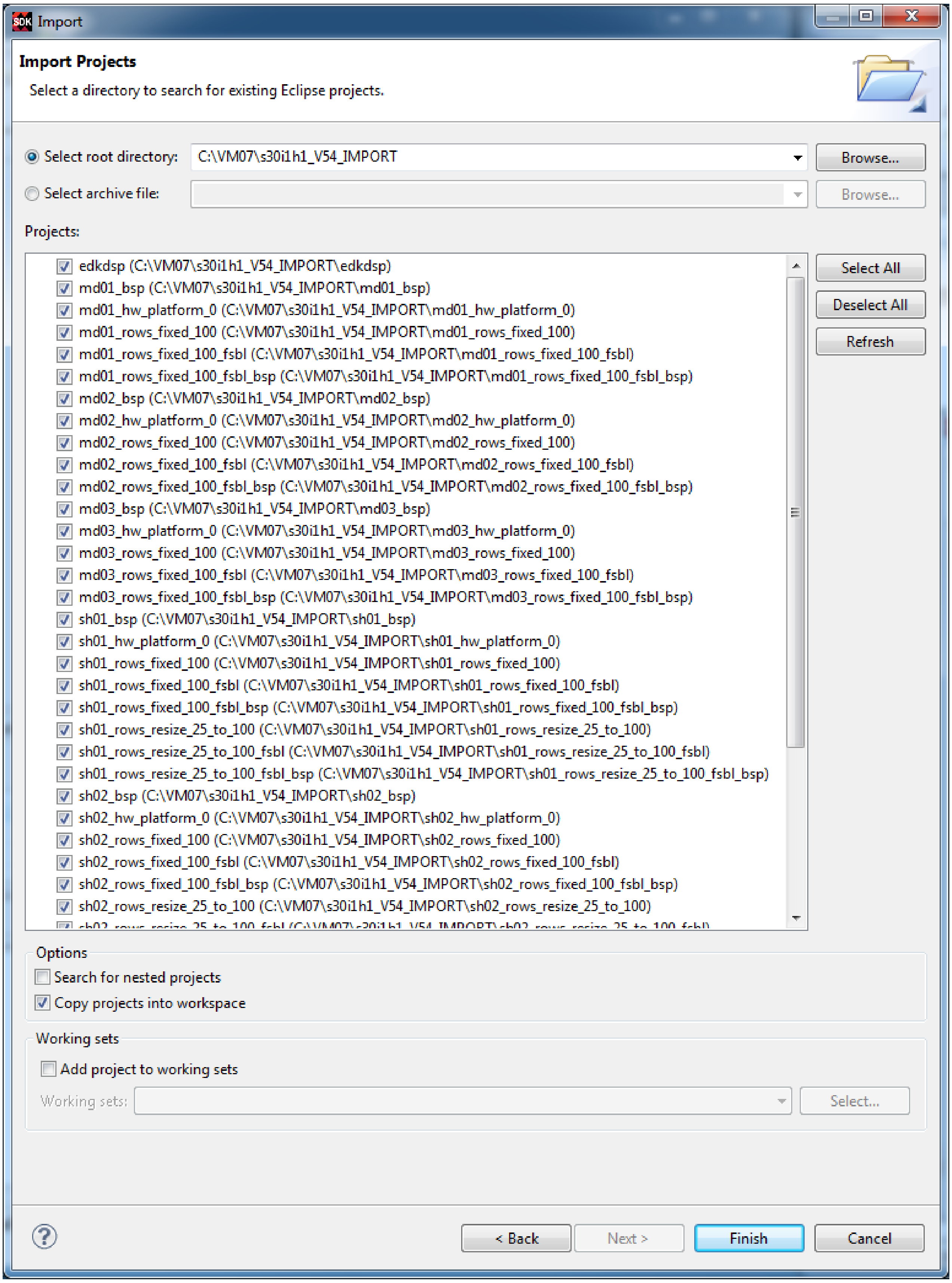
*Figure 16: Import existing projects into workspace.*

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

**C:\VM_07\s30i1h1_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 SW projects can take several minutes to finish. It should finish without errors.


Import

**Import Projects**

Select a directory to search for existing Eclipse projects.

(●) Select root directory: C:\VM07\s30i1h1_V54_IMPORT [Browse...]

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

Projects:

- [✓] edkdsp (C:\VM07\s30i1h1_V54_IMPORT\edkdsp)
- [✓] md01_bsp (C:\VM07\s30i1h1_V54_IMPORT\md01_bsp)
- [✓] md01_hw_platform_0 (C:\VM07\s30i1h1_V54_IMPORT\md01_hw_platform_0)
- [✓] md01_rows_fixed_100 (C:\VM07\s30i1h1_V54_IMPORT\md01_rows_fixed_100)
- [✓] md01_rows_fixed_100_fsbl (C:\VM07\s30i1h1_V54_IMPORT\md01_rows_fixed_100_fsbl)
- [✓] md01_rows_fixed_100_fsbl_bsp (C:\VM07\s30i1h1_V54_IMPORT\md01_rows_fixed_100_fsbl_bsp)
- [✓] md02_bsp (C:\VM07\s30i1h1_V54_IMPORT\md02_bsp)
- [✓] md02_hw_platform_0 (C:\VM07\s30i1h1_V54_IMPORT\md02_hw_platform_0)
- [✓] md02_rows_fixed_100 (C:\VM07\s30i1h1_V54_IMPORT\md02_rows_fixed_100)
- [✓] md02_rows_fixed_100_fsbl (C:\VM07\s30i1h1_V54_IMPORT\md02_rows_fixed_100_fsbl)
- [✓] md02_rows_fixed_100_fsbl_bsp (C:\VM07\s30i1h1_V54_IMPORT\md02_rows_fixed_100_fsbl_bsp)
- [✓] md03_bsp (C:\VM07\s30i1h1_V54_IMPORT\md03_bsp)
- [✓] md03_hw_platform_0 (C:\VM07\s30i1h1_V54_IMPORT\md03_hw_platform_0)
- [✓] md03_rows_fixed_100 (C:\VM07\s30i1h1_V54_IMPORT\md03_rows_fixed_100)
- [✓] md03_rows_fixed_100_fsbl (C:\VM07\s30i1h1_V54_IMPORT\md03_rows_fixed_100_fsbl)
- [✓] md03_rows_fixed_100_fsbl_bsp (C:\VM07\s30i1h1_V54_IMPORT\md03_rows_fixed_100_fsbl_bsp)
- [✓] sh01_bsp (C:\VM07\s30i1h1_V54_IMPORT\sh01_bsp)
- [✓] sh01_hw_platform_0 (C:\VM07\s30i1h1_V54_IMPORT\sh01_hw_platform_0)
- [✓] sh01_rows_fixed_100 (C:\VM07\s30i1h1_V54_IMPORT\sh01_rows_fixed_100)
- [✓] sh01_rows_fixed_100_fsbl (C:\VM07\s30i1h1_V54_IMPORT\sh01_rows_fixed_100_fsbl)
- [✓] sh01_rows_fixed_100_fsbl_bsp (C:\VM07\s30i1h1_V54_IMPORT\sh01_rows_fixed_100_fsbl_bsp)
- [✓] sh01_rows_resize_25_to_100 (C:\VM07\s30i1h1_V54_IMPORT\sh01_rows_resize_25_to_100)
- [✓] sh01_rows_resize_25_to_100_fsbl (C:\VM07\s30i1h1_V54_IMPORT\sh01_rows_resize_25_to_100_fsbl)
- [✓] sh01_rows_resize_25_to_100_fsbl_bsp (C:\VM07\s30i1h1_V54_IMPORT\sh01_rows_resize_25_to_100_fsbl_bsp)
- [✓] sh02_bsp (C:\VM07\s30i1h1_V54_IMPORT\sh02_bsp)
- [✓] sh02_hw_platform_0 (C:\VM07\s30i1h1_V54_IMPORT\sh02_hw_platform_0)
- [✓] sh02_rows_fixed_100 (C:\VM07\s30i1h1_V54_IMPORT\sh02_rows_fixed_100)
- [✓] sh02_rows_fixed_100_fsbl (C:\VM07\s30i1h1_V54_IMPORT\sh02_rows_fixed_100_fsbl)
- [✓] sh02_rows_fixed_100_fsbl_bsp (C:\VM07\s30i1h1_V54_IMPORT\sh02_rows_fixed_100_fsbl_bsp)
- [✓] sh02_rows_resize_25_to_100 (C:\VM07\s30i1h1_V54_IMPORT\sh02_rows_resize_25_to_100)
- [✓] sh02_rows_resize_25_to_100_fsbl (C:\VM07\s30i1h1_V54_IMPORT\sh02_rows_resize_25_to_100_fsbl)

[Select All] [Deselect All] [Refresh]

Options

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

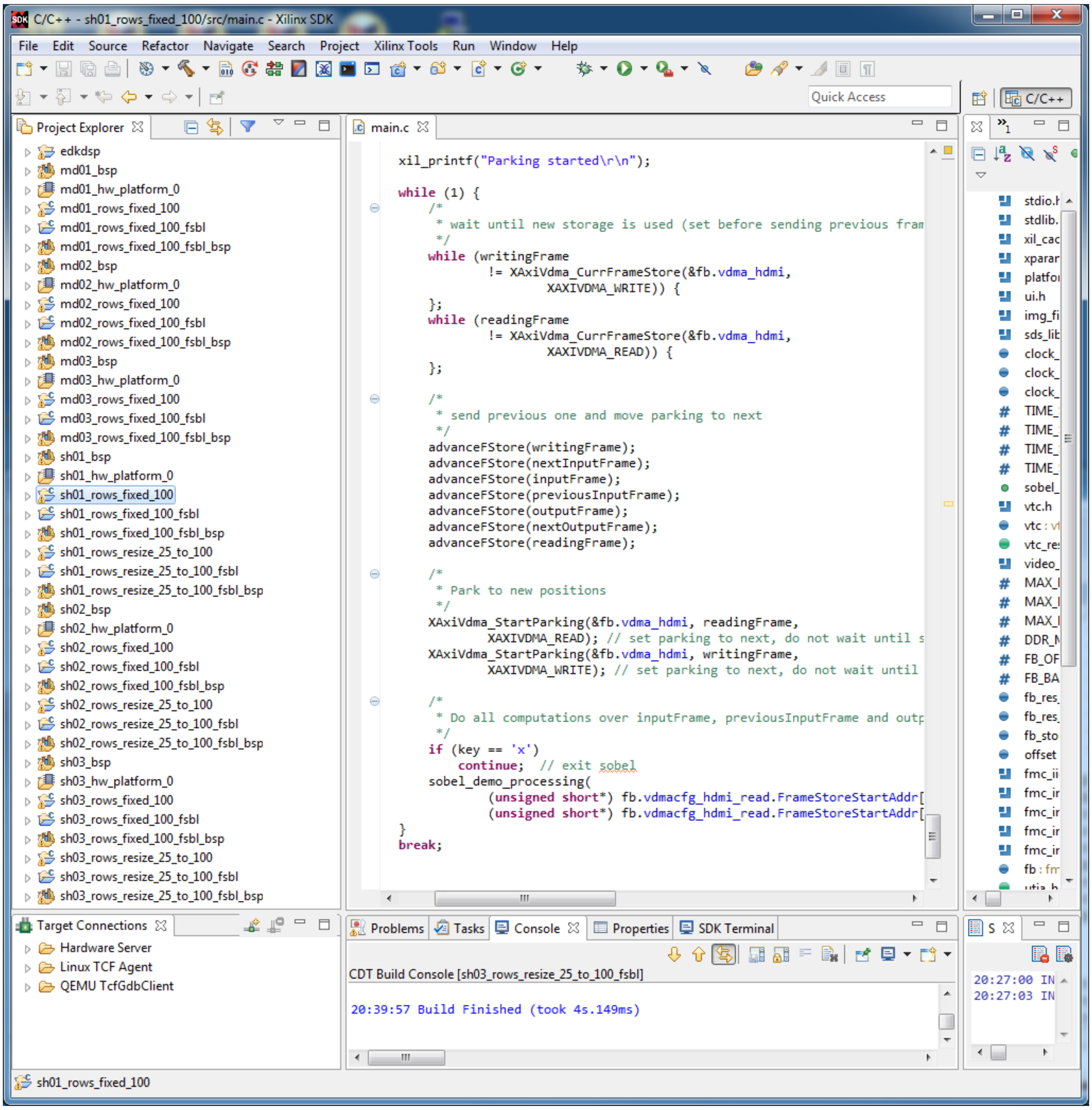
Working sets

- [ ] Add project to working sets

Working sets: [Select...]

[< Back] [Next >] [Finish] [Cancel]

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


C/C++ - sh01_rows_fixed_100/src/main.c - Xilinx SDK
File Edit Source Refactor Navigate Search Project Xilinx Tools Run Window Help
Quick Access
C/C++
Project Explorer
edkdsp
md01_bsp
md01_hw_platform_0
md01_rows_fixed_100
md01_rows_fixed_100_fsbl
md01_rows_fixed_100_fsbl_bsp
md02_bsp
md02_hw_platform_0
md02_rows_fixed_100
md02_rows_fixed_100_fsbl
md02_rows_fixed_100_fsbl_bsp
md03_bsp
md03_hw_platform_0
md03_rows_fixed_100
md03_rows_fixed_100_fsbl
md03_rows_fixed_100_fsbl_bsp
sh01_bsp
sh01_hw_platform_0
sh01_rows_fixed_100
sh01_rows_fixed_100_fsbl
sh01_rows_fixed_100_fsbl_bsp
sh01_rows_resize_25_to_100
sh01_rows_resize_25_to_100_fsbl
sh01_rows_resize_25_to_100_fsbl_bsp
sh02_bsp
sh02_hw_platform_0
sh02_rows_fixed_100
sh02_rows_fixed_100_fsbl
sh02_rows_fixed_100_fsbl_bsp
sh02_rows_resize_25_to_100
sh02_rows_resize_25_to_100_fsbl
sh02_rows_resize_25_to_100_fsbl_bsp
sh03_bsp
sh03_hw_platform_0
sh03_rows_fixed_100
sh03_rows_fixed_100_fsbl
sh03_rows_fixed_100_fsbl_bsp
sh03_rows_resize_25_to_100
sh03_rows_resize_25_to_100_fsbl
sh03_rows_resize_25_to_100_fsbl_bsp
main.c
Target Connections
Hardware Server
Linux TCF Agent
QEMU TcfGdbClient
Problems Tasks Console Properties SDK Terminal
sh01_rows_fixed_100

```
    xil_printf("Parking started\r\n");

    while (1) {
        /*
         * wait until new storage is used (set before sending previous fram
         */
        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 s
        XAxiVdma_StartParking(&fb.vdma_hdmi, writingFrame,
                XAXIVDMA_WRITE); // set parking to next, do not wait until

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

```
CDT Build Console [sh03_rows_resize_25_to_100_fsbl]

20:39:57 Build Finished (took 4s.149ms)
```

*Figure 18: All projects are compiled in debug mode.*

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

## 2.2 Precompiled SW projects

Files for the micro SD card (SW implementation of video processing algorithms on ARM from 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\md01\BOOT.bin
SD_cards\SW\md02\BOOT.bin
SD_cards\SW\md03\BOOT.bin
```

## 2.3 Precompiled HW projects

Files for the micro SD card (generated in SDK 2015.4 for HW accelerated projects) can be found in:

```
C:\VM07\s30i1h1\sh01_rows_fixed_100_fsbl\bootimage\BOOT.bin
C:\VM07\s30i1h1\sh01_rows_resize_25_to_100_fsbl_fsbl\bootimage\BOOT.bin
C:\VM07\s30i1h1\sh02_rows_fixed_100_fsbl\bootimage\BOOT.bin
C:\VM07\s30i1h1\sh02_rows_resize_25_to_100_fsbl_fsbl\bootimage\BOOT.bin
C:\VM07\s30i1h1\sh03_rows_fixed_100_fsbl\bootimage\BOOT.bin
C:\VM07\s30i1h1\sh03_rows_resize_25_to_100_fsbl_fsbl\bootimage\BOOT.bin

C:\VM07\s30i1h1\so01_rows_fixed_100_fsbl\bootimage\BOOT.bin
C:\VM07\s30i1h1\so01_rows_resize_25_to_100_fsbl_fsbl\bootimage\BOOT.bin
C:\VM07\s30i1h1\so02_rows_fixed_100_fsbl\bootimage\BOOT.bin
C:\VM07\s30i1h1\so02_rows_resize_25_to_100_fsbl_fsbl\bootimage\BOOT.bin
C:\VM07\s30i1h1\so03_rows_fixed_100_fsbl\bootimage\BOOT.bin
C:\VM07\s30i1h1\so03_rows_resize_25_to_100_fsbl_fsbl\bootimage\BOOT.bin

C:\VM07\s30i1h1\md01_rows_fixed_100_fsbl\bootimage\BOOT.bin
C:\VM07\s30i1h1\md02_rows_fixed_100_fsbl\bootimage\BOOT.bin
C:\VM07\s30i1h1\md03_rows_fixed_100_fsbl\bootimage\BOOT.bin
```

## 2.4 Recompilation of evaluation HW projects from SW source code in SDK 2015.4

- C code of included evaluation projects can be modified in the SDK 2015.4.
- Evaluation projects can be recompiled in the SDK 2015.4 for Debug or Release configuration.
- Result of the compilation is binary executable elf file for ARM. This file can be packed together with the first stage boot loader executable and bitstream to the updated BOOT.bin file.

The recompiled BOOT.bin is replacing the corresponding initial precompiled BOOT.bin file.

## 2.5 HW setup

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

**TE0715-03-30-1I**, Part: xc7z030sbg485-1I; 1 GByte DDR3; Industrial grade; [1]
**Heatsink for TE0715**, spring-loaded embedded; [2]
**EMC2-DP-V2 Carrier Board,** EMC²-DP PC/104 OneBank Carrier for SoC Modules [3]
**AES-FMC-HDMI-CAM-G,** FMC card with HDMI I/O and CAM interface [4]

EMC2-DP-V2 Carrier Board [3] requires one modification to run the demos on AES-FMC-HDMI-CAM-G with Zynq TE0715-03-30-1I system on module. The modification is related to the swapped polarity of the differential clock signal for the AES-FMC-HDMI-CAM-G FMC board on the Zynq TE0715-03-30-1I SoM module [1].

UTIA can implement these necessary HW modifications for the original EMC2-DP-V2 Carrier Board [3] from Sundance. 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 the shipment of the original EMC2-DP-V2 carrier board. Modification can be done in 5 working days and it is offered free of charge.

## 2.6 Test demos

To test demos follow these steps:

**Initial setup:**

- Start with EMC2-DB-V2 with switch-OFF power supply.
- 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 (or DVI) monitor by HDMI cable to the HDMI OUT on the AES-FMC-HDMI-CAM-G FMC card.
- Switch the HDMI monitor ON.
- Connect the carrier board by USB-to-microUSB cable to PC to support serial terminal.

**For each demo:**

- On PC, copy BOOT.bin file to the top directory of the micro SD card.
- Enter the SD card to the switched off EMC2-DB-V2.
- Switch-ON the power supply for the EMC2-DB-V2 board.
  - The first stage boot loader boots the bitstream and the application to the ARM Cotex A9 processor.
  - The source of 1920x1080p60 video signal (PC or Laptop) connects to the EMC2-DB-V2 board like to a Full HD HDMI monitor.
  - The EMC2-DB-V2 board connects to the Full HD HDMI output monitor. This initial process takes few seconds.
  - The processed video signal is displayed with fixed resolution 1920x1080p60 on the display.
- At this stage it is possible to open and configure the standard serial terminal client (PuTTY or similar) on your PC to see information printed from the currently running application. All applications use identical terminal setting:
  - Speed: 115200 bit/sec; Data bits: 8; Stop bits: 1; Parity: None; Flow control: None.
- On your PC, terminate the terminal client before switch-OFF of the EMC2-DB-V2 board.
- Switch-OFF power supply of EMC2-DB-V2 board to stop the demo.

**Additional notes about included demos:**

- All evaluation demos have first stage boot loader (FSBL) projects used for initialisation of the Zynq ARM processor, download of the bitstream and download of the application.
- The FSBL projects use the standard project template as provided by the SDK 2015.4.
- Edge detection demo sh01_rows_fixed_100 works on complete frame with single HW accelerator data path.
- 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.
- Edge detection demo sh02_rows_fixed_100 and sh02_rows_resize_25_to_100 work with 2 data paths.
- Edge detection demo 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.
- Motion detection demo md01_rows_fixed_100, md02_rows_fixed_100 and md03_rows_fixed_100 work with one, two and three HW video processing chains. These HW chains have only fixed set of processed lines (1x 100%, 2x 50% and 3x 33.3% of the Full HD frame).
- Demos md01 or md02 are linked with static libraries libmd01.a, or libmd02.a.

## *2.7 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 data paths

**Internal synchronisation with parallel HW data paths**

Consider **sh03_rows_fixed_100** project as an example. Three HW data paths will 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\s30i1h1\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 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 are defined in precompiled 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 data paths**

Consider **so03_rows_fixed_100** project as an example of this 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\s30i1h1\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 and returns to the main. All three functions are not blocking in this case. All three functions are defined in precompiled libso03.a static library.

The synchronisation point (similar to a barrier in case of SW threads) is implemented separately by three calls to functions **`sds_wait(1);`** **`sds_wait(2);`** **`sds_wait(3);`**. Functions are blocking and each of the functions terminates when the corresponding HW accelerated data path is done.

ARM processor can be programmed with user C code and execute it in parallel to the HW data paths in this case.

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

- Edge detection demo so01_rows_fixed_100 works on complete frame with single HW accelerator data path.
- Edge detection demo 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.

- Edge detection demo so02_rows_fixed_100 and so02_rows_resize_25_to_100 work with 2 data paths.
- Edge detection demo so03_rows_fixed_100 and so03_rows_resize_25_to_100 work with 3 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 library libso01.a, libso02.a or libso03.a.
- Block diagrams, area and acceleration results of these edge detection demos are practically identical to the corresponding demos described in sections 1.3, 1.4, 1.5.

# 3. References

[1] Xilinx Zynq Z-7030 SoC Micromodule XC7Z030-1SBG485I (ind. temp. range)
https://shop.trenz-electronic.de/en/TE0715-03-30-1I-Xilinx-Zynq-Z-7030-SoC-Micromodule-XC7Z030-1SBG485I-ind.-temp.-range

[2] Heatsink for TE0715, spring-loaded embedded;
https://shop.trenz-electronic.de/en/26923-Heatsink-for-TE0715-spring-loaded-embedded

[3] EMC²-DP PC/104 OneBank Carrier for SoC Modules
http://www.sundance.technology/som-cariers/pc104-boards/emc2-dp/

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

[5] Vivado HLx Web Install Client - 2015.4 Full Product Installation
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 - 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 |
| | |
| **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 |
| **libmd03.a** | SDSoC 2015.4 static library for HW accelerator in project md03 |

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.