freescale semiconductor	AMCU Software	
VISION	Revision <3.7>S32V234, OPENCV, GHS INTEGRITY RTOS AMCU_SW	Page 1 of 68

Vision SDK User Guide

ABSTRACT:

The document describes main components of the Vision SDK. Firstly, the list of prerequisites is discussed along with necessary utility libraries build. In the second part, the examples of use on S32V234 are presented.

KEYWORDS:

OpenCV, GHS INTEGRITY RTOS

APPROVED:

AUTHOR	SIGN-OFF SIGNATURE #1	SIGN-OFF SIGNATURE #2
Rostislav Hulik		

Revision History

VERSION	DATE	AUTHOR	CHANGE DESCRIPTION
1.0	18-December-13	Rostislav Hulik	First draft
1.1	07-April-14	Igor Aleksandrowicz	New demos added
2.0	28-April-14	Rostislav Hulik	OS build update, build contents update, new demos added
2.1	30-September-14	Samuel Mudrik	SDK Content
2.2	1-October-14	Rostislav Hulik	New demos and HDMI support added
2.3	11-November-14	Rostislav Hulik	Env. variables setup change
2.4	19-January-15	Rostislav Hulik	Content descriptions and new demos added.
2.5	02-March-15	Rostislav Hulik	Add Safety demo added
2.6	24-April-15	Anca Dima	Changed s32v234_sdk to s32v234_sdk
3.0	12-May-15	Rostislav Hulik & Tomas Babinec	Separated dedicated ZYNQ part into standalone document, updated this one. Added ISP demo description.
3.1	15-July-15	Anca Dima & Rostislav Hulik	First S32V234 release Removed ZC702 builds Added new demos
3.2	17-July-15	Roman Kubica & Tomas Babinec	ARM NEON Gaussian filter demo added ISP demos added
3.3	21-August-15	Rostislav Hulik	Linux support added
3.4	8-October-15	Rostislav Hulik & Roman Kubica	INTEGRITY support added ARM NEON demos added
3.5	6-November-15	Tomas Babinec	Added ISP demo description for Linux
3.6	7-December-15	Stefan Schwarz, Rostislav Hulik, Tomas Babinec	General update.
3.7	28-January-16	Tomas Babinec	General update.

Table of Contents

1 Introduction

In this document, the S32V234 Vision SDK is described. The SDK supports Standalone (OS less), Linux OS and INTEGRITY RTOS runtime environment.

The first part of this document covers installation and build of required prerequisites. This includes necessary libraries and any collateral parts of the SW.

The second part then describes the ready to use examples which are part of this SDK. These examples aim to provide an understanding of the basic functionality, such as using OpenCV with the SDK.

1.1 Purpose

The purpose of this document is to present the S32V234 Vision SDK and help users to bring up examples quickly.

1.2 Audience Description

This document is intended to S32V234 Vision SDK users.

1.3 References

Id	***Title***	***Location***
[1]	*GHS INTEGRITY RTOS*	http://www.ghs.com/products/rtos/integrity.html
[2]	*OpenCV library*	http://opencv.org/

Table 1 References Table

1.4 Definitions, Acronyms, and Abbreviations

Term/Acronym	***Description***
ACF	*APEX Core Framework*
APEX2	*A parallel image processing accelerator HW block, designed by CGV, part of S32V234 SoC.*
APEX COMPILER	*A set of tools (Synopsys IP Programmer for APEX2) that allow compilation of code for APEX2 subsystem.*
ARM	*Family of RISC architectures*

CGV	*Abbreviation for Cognivue company.*
DCU	*Display Controller Unit*
FDMA	*Fast DMA HW block.*
GHS	*Greenhills*
ISP	*Image Signal Processor subsystem of the S32V234 SoC.*
MIPI CSI	*MIPI Alliance standard for Camera Serial Interface.*
OpenCL	*Open Computing Language*
OpenCV	*Open Source Computer Vision*
RTOS	*Real Time Operating System*
SDK	*System Development Kit*
Sequencer	*M0+ core and its firmware controlling the ISP subsystem.*
SRAM	*Static RAM.4MB memory area used for fast data buffering during ISP processing.*

Table 2 Acronyms Table

1.5 Document Location

s32v234_sdk/docs/VisionSDK_UserGuide.pdf

2 SDK Content

2.1 Content Description

The package was created in order to present the Vision SDK to the user. The vast majority of user applications are demos, which demonstrates all possible uses of the system development kit.

This publication is targeted mainly at the vision processing pipeline of S32V234, which is aimed for fast, massively parallel image operations (APEX) and Image Signal Processing of the camera input (ISP). For APEX, two ways of programming are available:

1. APEX Core Framework (ACF)
2. OpenCL

The publication does not aim to describe either way of programming, but to enable user for using provided demos and compile all necessary prerequisites.

However, it’s necessary to mention that all provided software is also available in the sdk itself as out-of-the-box package. The user is not forced to build all described components.

2.2 Documentation Description

The Section 3, SW prerequisites build is described. The paragraph describes a build environment setup, necessary environment variables and OpenCV build. The OpenCV however is also provided in pre-built form.

The last Section 4 then describes provided demo applications.

2.3 Directory Structure

Note: the vSDK installer is targeted at Windows development host only.

After full installation, this is the content of the installation directory:

- **Cygwin [dir]** – Cygwin installation
- **Cygwin-packages [dir]** – offline packages used by Cygwin in the installation phase
- **s32v234_sdk [dir]** – the SDK itself
    - **build [dir]** – build toolchains used by the SDK
        - **cmake [dir]**
        - **mvc [dir]**
        - **nbuild [dir]**
    - **demos [dir]** – demo applications (chapter 4)
    - **docs [dir]** – documentation of the SDK
        - **acf [dir]**– CogniVue's ACF documentation
        - **html [dir]** – Doxygen-generated HTML
            - kernels [dir] – documentation of processing kernels
        - PDF user guides
    - **eigen [dir]** – Eigen library (source code)
    - **ffmpeg [dir]** – FFmpeg library (build scripts and pre-compiled binaries)
    - **fftw [dir]** – FFTW library build
    - **gprotobuf** – Protocol buffers tools
    - **isp** – All ISP related runtime SW components
        - **firmware [dir]** – sequencer SW
        - **graphs [dir]** – ISP processing graphs
        - **image_data [dir]** – test images
        - **inc [dir]** – header files required for ISP sequencer
        - **kernels [dir]** – IPUx kernels
    - **kernels [dir]** – APEX processing kernels which may be used in user applications
        - **apu [dir]** – APEX kernels implemented in "Vector C" language
        - **cl [dir]** – OpenCL kernels
    - **libs [dir]** – SDK software libraries

    - **apu [dir]** – software libraries for APEX
    - **arm [dir]** – software libraries for ARM CPU
    - **x86 [dir]** – software libraries for PC
- **ocv [dir]** – OpenCV library (build scripts and pre-compiled binaries)
- **os [dir]** - Environment build directory (OS, u-boot). Includes pre-compiled SD card contents.
    - **boot [dir]** - ???
    - **debug [dir]** – debugger scripts (mostly Lauterbach)
    - **integrity [dir]** – INTEGRITY OS related MULTI projects (for kernel and modules)
    - **linux [dir]** – Linux/platform specific components
    - build_content.tar.gz [file] – archive with pre-built Linux kernel, rtfs, device tree and u-boot to populate SD Card
- **platform [dir]** – Platform/device specific header files
- **tbb_lib [dir]** – Threading Building Blocks library for Windows (only used for APEX emulation)

- **tools [dir]** – support tools
  - **emu [dir]** –emulation libraries
    - **acf [dir]** –ACF emulation library
    - **apu [dir]** – APEX emulation library

3 SW prerequisites

This chapter describes the installation of necessary packages to run the examples. Note that not all examples need the presence of all packages described here. For more information about necessary installations, please refer to the section 5.

3.1 Build Environment

3.1.1OS Less and Linux OS Cross-Compilation

The applications for standalone run and for Linux OS can be cross-compiled from either Linux or Windows OS. Under Windows OS, a Cygwin bash is recommended along with GNU gcc/g++ compilers. **The package is part of the Vision SDK** and can be installed from the Vision SDK installer. Cygwin will automatically set up itself into PATH environment variable.

NOTE: it is recommended to use the Cygwin installation coming with the vSDK installer to avoid any incompatibilities.

When manual installation is needed, step by step guide is provided.

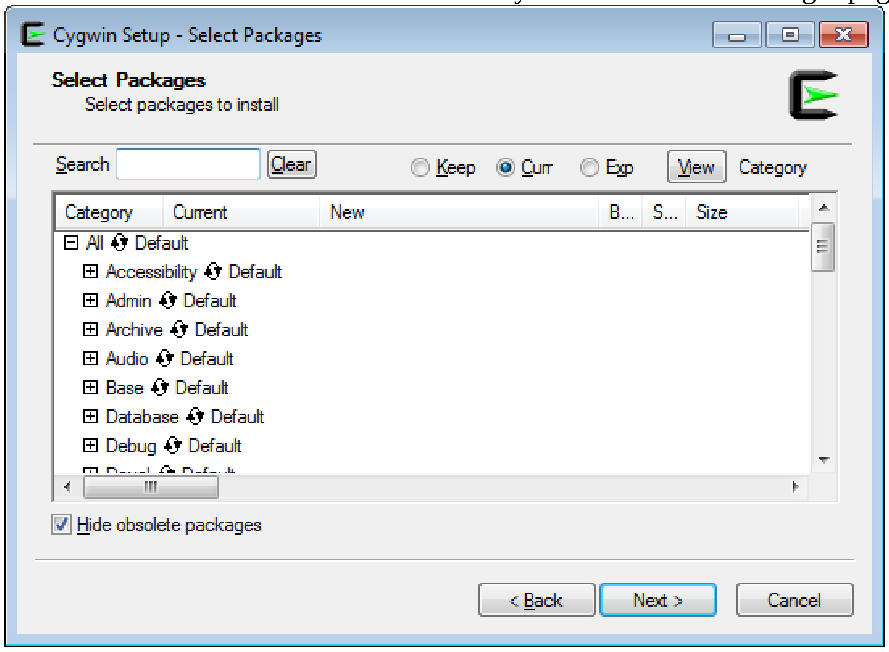
3.1.1.1Cygwin installation

1. **Download the installer**

   Current version of the installer from the official site http://cygwin.com/install.html is recommended.

2. **Run the installer**

   Proceed with the standard installation until you see the Select Packages page.



Cygwin Setup - Select Packages

Select Packages
Select packages to install

Search [ ] Clear   ○ Keep ● Curr ○ Exp   View   Category

Category	Current	New	B...	S...	Size
⊟ All ♲ Default					
⊞ Accessibility ♲ Default					
⊞ Admin ♲ Default					
⊞ Archive ♲ Default					
⊞ Audio ♲ Default					
⊞ Base ♲ Default					
⊞ Database ♲ Default					
⊞ Debug ♲ Default					

☑ Hide obsolete packages

< Back   Next >   Cancel

3. **Install additional utilities**

   Select these packages:

   a. Devel/make
   b. Devel/gcc-g++
      IMPORTANT! You have to install G++ 4.8, as some parts of the SDK are built using this version.
   c. Perl/perl

4. **Finish installation**

5. **(Optional) register Cygwin into System Path**

   If you wish to use default Windows command line instead of Cygwin bash, add the Cygwin/bin folder to the PATH variable.

3.1.2INTEGRITY RTOS Cross-Compilation

The build for INTEGRITY RTOS can be done in the same manner as for Linux OS with the aid of Cygwin combined with Green Hills MULTI compiler for 64 bit ARM, or through MULTI project files.

If Cygwin Makefile build is needed, please follow the instructions in 3.1.1 (in case Cygwin needs to be installed manually. Moreover, the user needs to install Green Hills MULTI SDK and add the compiler path to the PATH environment variable.

If MULTI project is needed, the installation of Green Hills MULTI SDK is imperative.

3.2 OpenCV

OpenCV library provides the API for handling image I/O in examples and provides large, community developed library of vision processing algorithms.

The SDK release contains prebuild binaries for supported platforms, so there is no need for re-build of the OpenCV. If a customized build is required please follow the steps in subsection bellow. The OpenCV library build is optimized to be executed on 64bit Linux machine. In Cygwin environment on Windows the build will fail because of Linux style paths being used with Windows compiler binaries. Please use Linux OS environment on host PC for building the OpenCV (the cross-compiled libraries are useable also for Windows cross-compilation – if the same compiler version is used).

3.2.1Build for OS Less application

1. **Check the compiler options for OCV**

   Open corresponding s32v234_sdk/build/cmake/toolchains/aarch64-sa-gnueabi-gcc.cmake file and check following lines:

   ```
   SET(CMAKE_C_COMPILER   arm-compiler-executable-gcc)
   SET(CMAKE_CXX_COMPILER arm-compiler-executable-g++)
   ```

   The compilers must be visible by the script, so they must be placed in the PATH directory, or the absolute path must be provided in these commands.

2. **Download the OpenCV 2.4.10 sources**

   http://opencv.org/downloads.html

3. **Extract the sources to the s32v234_sdk/../opencv_src**

4. **Apply the Standalone patch**

   a. Switch into /s32v234_sdk/../opencv_src directory

   b. Apply available patch by executing:

   ```
   patch -p1 -i s32v234_sdk/ocv/standalone_2.4.10.patch
   ```

5. **Build the OpenCV**

   a. Open corresponding /s32v234_sdk/ocv/cmake-v234ce-gnu-sa directory

   b. `sh build.sh`

3.2.2Build for Linux OS application

1. **Check the compiler options for OCV**

   Open corresponding s32v234_sdk/build/cmake/toolchains/aarch64-linux-gnu-gcc.cmake file and check following lines:

   ```
   SET(CMAKE_C_COMPILER   arm-compiler-executable-gcc)
   SET(CMAKE_CXX_COMPILER arm-compiler-executable-g++)
   ```

   The compilers must be visible by the script, so they must be placed in the PATH directory, or the absolute path must be provided in these commands.

2. **Set the environment variable**

   a. For this step, the root file system must be available on the host machine. It's possible to do that by mounting the uramdisk.image.gz into host machine (see section 3.2.2. – 2.)

   b. For examples which are using OpenCV is recommended to use NFS mounted rootfs due to the size of the OpenCV libraries (ramdisk file is prepared for maximum of 16MB)

   c. `Export LINUX_ROOT_PATH=/path/to/crosscompile/rootfs/`

3. **Download the OpenCV 2.4.10 sources**

   http://opencv.org/downloads.html

4. **Extract the sources to the s32v234_sdk/../opencv_src**

5. **Build the OpenCV**

   a. Open corresponding /s32v234_sdk/ocv/cmake-v234ce-gnu-linux directory

   b. `sh build.sh`

3.2.3Build for INTEGRITY RTOS application

1. **Check the compiler options for OCV**

   Open corresponding s32v234_sdk/build/cmake/toolchains/aarch64-integrity-ghs.cmake file and check following lines:

   ```
   SET(CMAKE_C_COMPILER "${MULTI_COMPILER_PATH}/ccintarm64")
   SET(CMAKE_CXX_COMPILER "${MULTI_COMPILER_PATH}/cxintarm64")
   ```

   The compilers must be visible by the script, so they must be placed in the PATH directory, or the absolute path must be provided in these commands.

2. **Set the environment variable**
   a. For this step, the INTEGRITY RTOS installation must be available on the host machine.
   b. `Export INTEGRITY_ROOT_PATH =/path/to/integrity/installation`
   c. `Export MULTI_ROOT_PATH=/path/to/multi/installation`
   d. `Export MULTI_COMPILER_ROOT_PATH =/path/to/multi/compiler/executables`
   e.

3. **Download the OpenCV 2.4.10 sources**

   http://opencv.org/downloads.html

4. **Extract the sources to the s32v234_sdk/../opencv_src**
5. **Apply the INTEGRITY patch**
   a. Switch into /s32v234_sdk/../opencv_src directory
   b. Apply available patch by executing:

   ```
   patch -p1 -i s32v234_sdk/ocv/integrity_2.4.10.patch
   ```

6. **Build the OpenCV**
   a. Open corresponding /s32v234_sdk/ocv/cmake-v234ce-ghs-integrity directory
   b. `sh build.sh`

3.3 Kernel modules in Linux OS

To run an application using the S32V234 vision processing pipeline including camera I/F, ISP and APEX2 accelerators a number of kernel (*.ko) modules are required. Those are available pre-built in the vSDK (please refer to the chapter 2.3). In case re-building those kernel modules is required please follow following instructions.

In Linux OS, the drivers need to be build and installed as a kernel modules into running OS. User can build and install the drivers using following steps:

3.3.1General Preparation

Export necessary environment variables:

- `export ARCH=arm`
  - Target architecture
- `export CROSS_COMPILE=aarch64-linux-gnu-`
  - Compiler prefix specification
- `export PATH=/path/to/your/compiler/binaries:$PATH`
  - Path to the compiler binaries
- `export PATH=/path/to/your/CHESS/compiler/binaries:$PATH`
  - Path to the target APEX compiler binaries
- `export LINUX_S32V234_DIR=/linux/sources/directory/for/zc702/`
  - optional, only for s32v234 Linux build
  - Path to the Linux sources (same kernel as is running on the board)
- `export PATH=/path/to/your/compiler/binaries:$PATH`
  - Path to the compiler binaries

3.3.2Building the Kernel Modules

Following kernel modules can be built:

csi.ko:	s32v234_sdk/libs/arm/isp/csi/kernel/build-v234ce-gnu-linux-d
sram.ko:	s32v234_sdk/libs/arm/isp/sram/kernel/build-v234ce-gnu-linux-d
sequencer.ko:	s32v234_sdk/libs/arm/isp/sequencer/kernel/build-v234ce-gnu-linux-d
fdma.ko:	s32v234_sdk/libs/arm/isp/fdma/kernel/build-v234ce-gnu-linux-d
oal_cma.ko:	s32v234_sdk/libs/arm/utils/oal/kernel/build-v234ce-gnu-linux-d
oal_cma.ko:	s32v234_sdk/libs/arm/apex/apex/kernel/build-v234ce-gnu-linux-d

1. **Building kernel modules**

```
cd <kernel_build_dir>
make
```

2. **Install the module on target board**

```
insmod sram.ko
insmod csi.ko
insmod sequencer.ko
insmod fdma.ko
insmod oal.ko
insmod apex2.ko
```

NOTE: the included rootfs s32v234_sdk/os/build_content.tar.gz already contains a startup script which will insert the kernel modules.

4 Examples and demos

The example applications and demos serve to demonstrate the APEX functionality on the S32V234 based boards. In this chapter, each demo is described along with its necessary prerequisites and run commands.

4.1 Demo Overview

Following table gives an overview of available demos and which platform (OS) is currently supported.

Focus	Name	Description	Bare Metal	Linux	INTEGRITY
Generic	hello	Hello World example how to build a simple ARM only program	x	x	x
	qspi_readwirte	QSPI ReadWrite Application: The application demonstrated SD card boot and QSPI read-write interface during program run	x		
APEX	apex_add	APEX Add Two Images: simplest APEX usage model which creates a graph calling the addition kernel and computing the addition in parallel on the APEX device	x	x	
	apex_downsample_upsample	APEX Downsample/Upsample:example on how to compute on APEX with image size change from input to output	x	x	
	apex_emulation_test	APEX Emulation Test: is a collection of some of the APEX algorithms listed here (gauss filtering, FAST9, histogram computation, integral image, indirect inputs, general filtering), which demonstrates different ways of handling code	x	x	

		organization for either simplicity or readability. It has also an attached VS project for Windows emulation.			
	apex_face_detection_cv	APEX Face Detection: complex algorithm which uses feature detection and classifiers on APEX to detect human heads in images.	x	x	
	apex_fast9	APEX Fast9 Corner Detection: FAST9 corner detection algorithm implementation on APEX	x	x	
	apex_gauss5x5_cv	: APEX Gauss 5x5 Filter: Simple filtering demo which exemplifies how to cope with spatial dependencies on APEX	x	x	
	apex_histogram_cv	APEX Histogram Computing: computes in parallel the histogram of an image with demonstration of static data usage on APEX	x	x	
	apex_indirect_input_cv	APEX Indirect Inputs: Shows how to randomly access image patches on APEX, with an example of image rotation	x	x	
	apex_integral_image_cv	APEX Integral Image: computes the integral image (Summed Area Table – SAT) of an image, which is used an optimization for applying subsequent box filters and others.	x	x	
	apex_irq	demonstrates the interrupt use for signaling the end of APEX execution instead of polling mechanism.	x		

	apex_orb_cv	APEX ORB Matching: complex algorithm for computing the disparity match of two images using the Oriented BRIEF algorithm	x	x	
	apex_roi_cv	example on how to use the Region Of Interest feature of the APEX, by envisioning a certain region of an input image.	x	x	
APEX & ISP	apex_isp_ldw_cv	APEX Lane Departure Warning: demonstrates the implementation on APEX of the complex Lane Detection algorithm.	x	x	
	apex_isp_face_detection_cv	APEX Face Detection Algorithm: demonstrates the implementation of APEX driven face detection algorithm.	x		
ISP	isp_csi_dcu	ISP pass-through demo: shows how to grab camera data using ISP and show them on the internal display.	x	x	
	isp_mipi_raw	ISP pass-through demo: shows how to grab camera data using ISP and show them on the internal display	x	x	
	isp_mipi_raw	ISP Mipi raw data: displays the raw data from camera sensor to display.	x	x	
	isp_chroma_key	ISP Chroma key: presents ISP ability to modify the camera data by mapping specific color ranges to blue or green.	x	x	
	isp_gpr	ISP general purpose register of IPU: demonstrates possibility to parametrize IPU processing at runtime	x	x	

		using general purpose registers.			
APEX OpenCL	apexcl_bitonic_sort	Bitonic Sort: Sorts on APEX a vector of numbers with the AMD Bitonic sort OpenCL algorithm		x	
	apexcl_floyd_warshall	APEX CL Floyd Warshall: Applies on APEX the AMD Floyd Warshall OpenCL algorithm for searching for shortest path in a graph		x	
	apexcl_median	APEX CL Median Filter: Applies with APEX an OpenCL median filter kernel onto an image		x	
	apexcl_memory_model	APEX CL Memory Model: demonstrates accesses to various memories in the system.		x	
APEX CV	apex_cv	ApexCV Base testing demo: Tests all ApexCV Base functions.	x		
	apex_cv_pro_affine	ApexCV Pro Affine transformations demonstration and test.	x		
	apex_cv_pro_blockmatching	ApexCV Pro Blockmatching demonstration and test.	x		
	apex_cv_pro_brief	ApexCV Pro BRIEF demonstration and test.	x		
	apex_cv_pro_canny	ApexCV Pro Canny edge detector demonstration and test.	x		
	apex_cv_pro_gftt	ApexCV Pro Good Features To Track demonstration and test.	x		
	apex_cv_pro_harris	ApexCV Pro Harris corner detector demonstration and test.	x		

	apex_cv_pro_hog	ApexCV Pro Histogram of Gaussians demonstration and test.	x		
	apex_cv_pro_hough	ApexCV Pro Hough Transform demonstration and test.	x		
	apex_cv_pro_image_pyramids	ApexCV Pro Image Pyramid demonstration and test.	x		
	apex_cv_pro_remap	ApexCV Pro image remapping demonstration and test.	x		
	apex_cv_pro_resize	ApexCV Pro image resize demonstration and test.	x		
NEON	neon_gauss3x3_cv	NEON Gaussian filter: Demonstrates simple ARM NEON SIMD filtering kernel.	x		
	neon_eigen_add	Eigen library addition: Presents simple matrix addition which uses Eigen library accelerated by ARM NEON	x		
	neon_fftw_cv	FFTW library Fourier transform: Computes forward and backward Fourier transform realized through FFTW library accelerated by ARM NEON	x		

4.2 Building the demos

4.2.1OS Less Makefile Cross-Compilation

For Standalone application, the Linaro build framework is available to ease up the build process. For any board, following syntax will build the application:

1. **<u>Change to the application directory</u>**
   a. `cd s32v234_sdk/demos/name_of_demo/build-v234ce-gnu-sa-d`
2. **<u>If building with APEX COMPILER, run "make clean"</u>**
   a. This option will disable permanently build without APEX compiler (deletes the header files)
3. **<u>Run "make allsub" command</u>**
   a. If APEX compiler is not present, "**make APEX_PREBUILD=1 allsub**" will use pre-build APEX kernels.
   b. WARNING, "**make clean**" will remove all targets as well as pre-build APEX kernels, please issue "**make APEX_PREBUILD=1 clean**" if you wish to preserve those headers.

4.2.2Linux OS Makefile Cross-Compilation

On Linux OS, the Linaro build framework is available to ease up the build process. For any board, using following syntax will build the application for Linux OS:

1. **<u>Change to the application directory</u>**
   a. `cd s32v234_sdk/demos/name_of_demo/ build-v234ce-gnu-linux-d`
2. **<u>If building with APEX COMPILER, run "make clean"</u>**
   a. This option will disable permanently build without APEX compiler (deletes the header files)
3. **<u>Run "make allsub" command</u>**
   a. If APEX compiler is not present, "**make APEX_PREBUILD=1 allsub**" will use pre-build APEX kernels.
   b. WARNING, "**make clean**" will remove all targets as well as pre-build APEX kernels, please issue "**make APEX_PREBUILD=1 clean**" if you wish to preserve those headers.

This algorithm will build the demo along with all its dependencies. The only parts which are not building by this approach are the kernel modules. To build the kernel modules (APEX driver, Framebuffer driver, OAL Driver), it's necessary to execute similar approach:

1. **<u>Change to the driver directory</u>**
    a. `cd s32v234_sdk/libs/arm/apex/kernel/build-v234ce-gnu-linux-d`
2. **<u>run "sh build.sh" command</u>**

4.2.3INTEGRITY RTOS Makefile cross-compilation

On INTEGRITY RTOS, the MULTI build framework is available to ease up the build process. For any board, using following syntax will build the application for Linux OS:

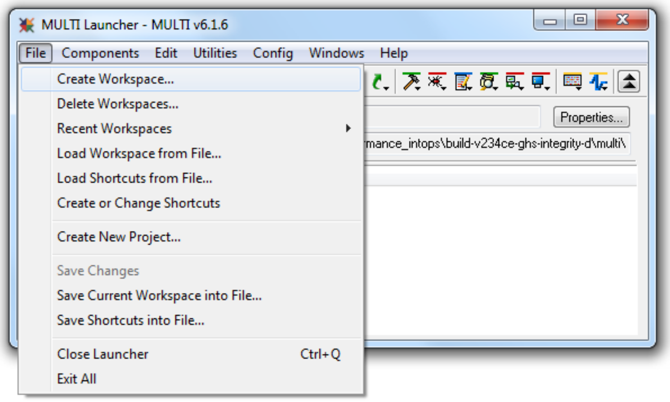
1. **<u>Change to the application directory</u>**
    a. `cd s32v234_sdk/demos/name_of_demo/`
       `build-v234ce-ghs-integrity-d`
2. **<u>If building with APEX COMPILER, run "make clean"</u>**
    a. This option will disable permanently build without APEX compiler (deletes the header files)
3. **<u>Run "make allsub" command</u>**
    a. If APEX compiler is not present, "**make APEX_PREBUILD=1 allsub**" will use pre-build APEX kernels.

WARNING, "**make clean**" will remove all targets as well as pre-build APEX kernels, please issue "**make APEX_PREBUILD=1 clean**" if you wish to preserve those headers.

4.2.4INTEGRITY RTOS MULTI Project cross-compilation

On INTEGRITY RTOS, MULTI Project files are also available. In current version, the project does not include APEX build. If the rebuild of APEX files is necessary, please follow the step 4.3.3 to rebuild the APEX program.

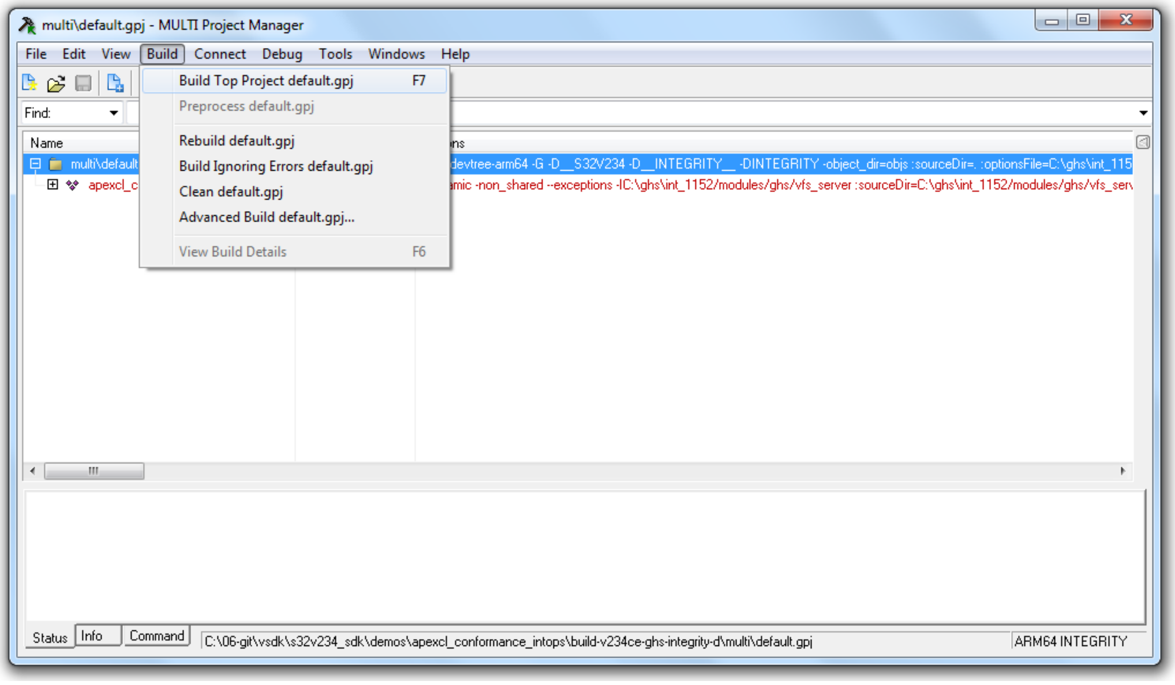
1. **<u>Open MULTI Launcher</u>**
2. **<u>Create a new workspace for a demo</u>**


MULTI Launcher - MULTI v6.1.6
File
Components
Edit
Utilities
Config
Windows
Help
Create Workspace...
Delete Workspaces...
Recent Workspaces
Load Workspace from File...
Load Shortcuts from File...
Create or Change Shortcuts
Create New Project...
Save Changes
Save Current Workspace into File...
Save Shortcuts into File...
Close Launcher Ctrl+Q
Exit All
Properties...
mance_intops\build-v234ce-ghs-integrity-d\multi\

a. Choose existing project

b. Choose demo project file:
`s32v234_sdk/demos/name_of_demo/build-v234ce-ghs-integrity-d/multi/default.gpj`

3. Open the workspace and build the application



devtree-arm64 -G -D__S32V234 -D__INTEGRITY__ -DINTEGRITY -object_dir=objs :sourceDir=. :optionsFile=C:\ghs\int_115
mic -non_shared --exceptions -IC:\ghs\int_1152/modules/ghs/vfs_server :sourceDir=C:\ghs\int_1152/modules/ghs/vfs_serv

C:\06-git\vsdk\s32v234_sdk\demos\apexcl_conformance_intops\build-v234ce-ghs-integrity-d\multi\default.gpj

4.3 General Examples

4.3.1Hello World Application

Hello world application is the simplest application in the SDK. The run itself consists only of one printf showing a text on console. The used can however use it in order to get familiar with the build and SD card boot.

Name of the application directory: **helloworld**

Available builds:

- S32V234 Standalone
- S32V234 INTEGRITY + MULTI Project

4.3.1.1Prerequisites

4.3.1.1.1 OS Less application

There are no specific requirements to run this application.

4.3.1.1.2 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.3.1.2Running the example

4.3.1.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading helloworld.cmm script.

OR

Load the helloworld.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=helloworld.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

4.3.1.2.2 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

4.3.2QSPI Read Write

The application demonstrates the QSPI driver use. After loading the application, it will write and reads from the QSPI flash. The results are then displayed on console. It can be loaded via SD card to demonstrate functionality of both systems.

Name of the application directory: **qspi_readwrite**

Available builds:

- S32V234 Standalone

4.3.2.1Prerequisites

4.3.2.1.1 OS Less application

There are no specific requirements to run this application.

4.3.2.2Running the example

4.3.2.2.1 OS Less application

Load the qspi_readwrite.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if= qspi_readwrite.bin of=/dev/sdb bs=512 seek=0
conv=fsync
```

4.3.2.2.2 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

4.4 APEX Related Examples

4.4.1APEX Add Two Images

The add example is prepared to demonstrate one-kernel APEX graph in use. The application generates two random grayscale 8-bit images (matrices) and adds them using APEX HW and SW algorithm. The parallel use of both APEX cores is also demonstrated, because the program adds two images on both cores simultaneously creating two outputs. The resulting two 16-bit grayscale images are compared and result is printed to the user.

Name of the application directory: **apex_add**

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 Standalone
- S32V234 INTEGRITY + MULTI Project

4.4.1.1Prerequisites

4.4.1.1.1 OS Less application

There are no specific requirements to run this application.

4.4.1.1.2 Linux OS

The apex2.ko and oal_cma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod apex2.ko
insmod oal_cma.ko
```

4.4.1.1.3 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.4.1.2Running the example

4.4.1.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_add.cmm script.

OR

Load the apex_add.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

  ```
  sudo dd if=apex_add.bin of=/dev/sdb bs=512 seek=0 conv=fsync
  ```

4.4.1.2.2 Linux OS

Application starts as follows:

```
./apex_add.elf
```

4.4.1.2.3 Windows OS

Please load the Visual Studio Project from ./build-deskwin32/mvc/add_project.sln, compile and run it.

4.4.1.2.4 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

4.4.2APEX Downsample/Upsample

The application demonstrates data rate change when the element k is different for the connected ports. The demo loads an input image and performs a linear downsampling and upsampling of the image. The output consists of two images, one of them is an enlarged version of the input, the other one is a shrinked version displayed on S32V234 EVB display.

Name of the application directory: **apex_downsample_upsample**

Available builds:

- S32V234 Linux
- S32V234 Standalone
- S32V234 INTEGRITY + MULTI Project

4.4.2.1 Prerequisites

4.4.2.1.1 OS Less application

There are no specific requirements to run this application.

4.4.2.1.2 Linux OS

The apex2.ko and oal_cma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod apex2.ko
insmod oal_cma.ko
```

OpenCV shared libraries must be present in the root file system.

A 256x256 8-bit grayscale input image named in_grey_256x256.png must be present in the working directory.

4.4.2.1.3 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.4.2.2Running the example

4.4.2.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_downsample_upsample.cmm script.

OR

Load the apex_downsample_upsample.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_downsample_upsample.bin of=/dev/sdb bs=512 seek=0
conv=fsync
```

4.4.2.2.2 Linux OS

Application starts as follows:

```
./apex_downsample_upsample.elf
```

4.4.2.2.3 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

Example outputs of the downsample/upsample demo

4.4.3APEX Emulation Test

The application demonstrates the use of emulation library. It works as test of several subsequent ACF graphs defined in the application, however if built for host desktop machine, the application uses emulation library. If built for ZC702, an ACF implementation is used. By this approach, it’s possible to demonstrate use of the kernels in both emulated and real environment.

Name of the application directory: **apex_emulation_test**

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 Standalone
- S32V234 INTEGRITY + MULTI Project

4.4.3.1Prerequisites

4.4.3.1.1 OS Less application

There are no specific requirements to run this application.

4.4.3.1.2 Linux OS

The apex2.ko and oal_cma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod apex2.ko
insmod oal_cma.ko
```

OpenCV shared libraries must be present in the root file system.

4.4.3.1.3 INTEGRITY RTOS application

There are no specific requirements to run this application.

Running the example

4.4.3.1.4 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_emulation_test.cmm script.

OR

Load the apex_emulation_test.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
  sudo dd if=apex_emulation_test.bin of=/dev/sdb bs=512 seek=0
conv=fsync
```

4.4.3.1.5 Linux OS

Application starts as follows:

```
./apex_emulation_test.elf
```

- Do it by executing: `export LD_LIBRARY_PATH=/mnt/lib`

4.4.3.1.6 Windows OS

Please load into Visual Studio, the project from ./build-deskwin32/mvc/apex_emulation_test.sln, compile and run it.

4.4.3.1.7 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

4.4.4APEX Face Detection using LBP Cascades

This application demonstrates the use of APEX LBP classifier. The demo loads an RGB input image, transforms it to grayscale, executes the classifier, marks the faces as green rectangles and outputs the resulting RGB image. The classifier is implemented both in ACF and on the host processor. The applications outputs 3 images: faces found by APEX, faces found by the OpenCV library and faces found by a custom host CPU fixed-point implementation.

Name of the application directory: **apex_face_detection_cv**

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 Standalone
- S32V234 INTEGRITY + MULTI Project

4.4.4.1Prerequisites

4.4.4.1.1 OS Less application

There are no specific requirements to run this application.

4.4.4.1.2 Linux OS

The apex2.ko and oal_cma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod apex2.ko
insmod oal_cma.ko
```

OpenCV shared libraries must be present in the root file system.

A 384x320 RGB input image named input_faces.png and LBP cascade classificator XML named lbpcascade_frontalface.xml must be present in the working directory.

4.4.4.1.3 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.4.4.2Running the example

4.4.4.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_face_detection_cv_test.cmm script.

OR

Load the apex_face_detection_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
  sudo  dd  if=face_detection_cv.bin  of=/dev/sdb  bs=512  seek=0
conv=fsync
```

4.4.4.2.2 Linux OS

Application starts as follows:

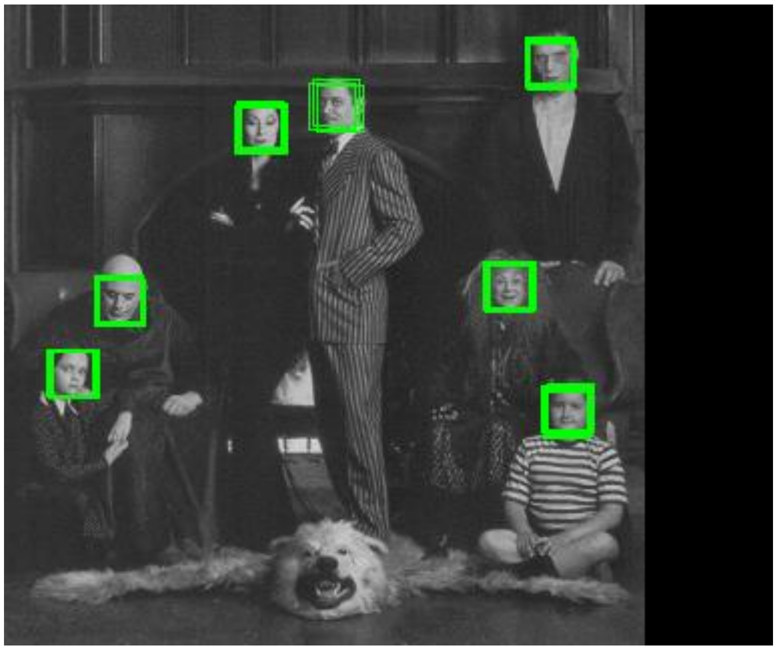
```
./apex_face_detection_lbp.elf
```

4.4.4.2.3 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

4.4.4.2.4 Windows OS

Please load into Visual Studio, the project from ./build-deskwin32/mvc/apex_emulation_test.sln, compile and run it



Example output of the LBP face detection algorithm

4.4.5APEX FAST9 Corner Detector

This application demonstrates the use of multi-channel ACF input. The demo loads the RGB input image, transforms it to grayscale, executes the FAST9 algorithm, marks the corners in green output channel and outputs the resulting RGB image (EVB Display). All the phases are implemented in ACF.

Name of the application directory: **apex_fast9_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone
- S32V234 INTEGRITY + MULTI Project

4.4.5.1Prerequisites

4.4.5.1.1 OS Less application

There are no specific requirements to run this application.

4.4.5.1.2 Linux OS

The apex2.ko and oal_cma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod apex2.ko
insmod oal_cma.ko
```

OpenCV shared libraries must be present in the root file system.

A 256x256 RGB input image named in_color_256x256.png must be present in the working directory.

4.4.5.1.3 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.4.5.2Running the example

4.4.5.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_fast9_cv.cmm script.

OR

Load the apex_fast9_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_fast9_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

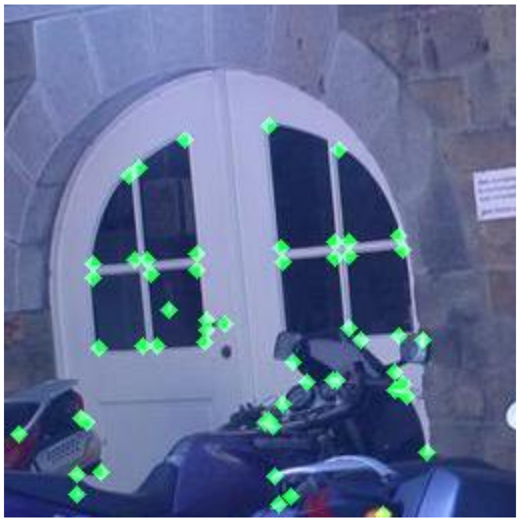
4.4.5.2.2 Linux OS

Application starts as follows:

```
./apex_fast9_color_cv.elf
```

4.4.5.2.3 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.



Example output of the FAST9 RGB algorithm

4.4.6APEX Gauss 5x5 Filter

The application demonstrates another simple one-kernel algorithm on the input image. The demo loads an image and executes the 3x3 Gaussian blur APEX kernel on it. The output is the blurred image displayed on the EVB.

Name of the application directory: **apex_gauss5x5_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone
- S32V234 INTEGRITY + MULTI Project

4.4.6.1Prerequisites

4.4.6.1.1 OS Less application

There are no specific requirements to run this application.

4.4.6.1.2 Linux OS

The apex2.ko and oal_cma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod apex2.ko
insmod oal_cma.ko
```

OpenCV shared libraries must be present in the root file system.

A 256x256 8-bit grayscale input image named in_grey_256x256.png must be present in the working directory.

4.4.6.1.3 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.4.6.2Running the example

4.4.6.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_gauss5x5_cv.cmm script.

OR

Load the apex_gauss5x5_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_gauss5x5_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

4.4.6.2.2 Linux OS

Application starts as follows:

```
./apex_gauss5x5_cv.elf
```

4.4.6.2.3 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

4.4.7APEX Histogram Computation

The application demonstrates a non-image ACF output. The demo loads an input image, computes its histogram and compares it to the reference histogram. It outputs the histogram as an image.

Name of the application directory: **apex_histogram_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone
- S32V234 INTEGRITY + MULTI Project

4.4.7.1Prerequisites

4.4.7.1.1 OS Less application

There are no specific requirements to run this application.

4.4.7.1.2 Linux OS

The apex2.ko and oal_cma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod apex2.ko
insmod oal_cma.ko
```

OpenCV shared libraries must be present in the root file system.

A 256x256 8-bit grayscale input image named in_grey_256x256.png must be present in the working directory.

4.4.7.1.3 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.4.7.2Running the example

4.4.7.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_histogram_cv.cmm script.

OR

Load the apex_histogram_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
  sudo  dd  if=apex_histogram_cv.bin  of=/dev/sdb  bs=512  seek=0
conv=fsync
```

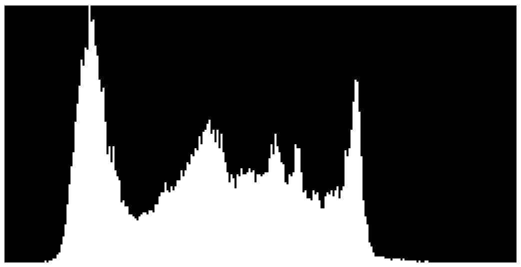
4.4.7.2.2 Linux OS

Application starts as follows:

```
./apex_histogram_cv.elf
```

4.4.7.2.3 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.



Example output of the histogram demo

4.4.8APEX Indirect Inputs Example

The application demonstrates indirect ACF inputs. The demo loads an input image and performs a 180-degree rotation of the image. It outputs the rotated image displayed on EVB.

Name of the application directory: **apex_indirect_input_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone
- S32V234 INTEGRITY + MULTI Project

4.4.8.1Prerequisites

4.4.8.1.1 OS Less application

There are no specific requirements to run this application.

4.4.8.1.2 Linux OS

The apex2.ko and oal_cma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod apex2.ko
insmod oal_cma.ko
```

OpenCV shared libraries must be present in the root file system.

A 256x256 8-bit grayscale input image named in_grey_256x256.png must be present in the working directory.

4.4.8.1.3 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.4.8.2Running the example

4.4.8.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_indirect_input_cv.cmm script.

OR

Load the apex_indirect_input_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo  dd  if=apex_indirect_input_cv.bin  of=/dev/sdb  bs=512  seek=0
conv=fsync
```

4.4.8.2.2 Linux OS

Application starts as follows:

```
./apex_downsample_upsample_cv.elf
```

4.4.8.2.3 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

Example output of the rotate 180 demo

4.4.9APEX Integral Image

The application demonstrates static data usage. The demo loads an input image, computes the integral image (i.e. sat = summed area table) and performs box filtering using this table. The output consists of two images, namely the summed area table and the final filtered result image displayed on EVB.

Name of the application directory: **apex_integral_image_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone
- S32V234 INTEGRITY + MULTI Project

4.4.9.1Prerequisites

4.4.9.1.1 OS Less application

There are no specific requirements to run this application.

4.4.9.1.2 Linux OS

The apex2.ko and oal_cma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod apex2.ko
insmod oal_cma.ko
```

OpenCV shared libraries must be present in the root file system.

A 256x256 8-bit grayscale input image named in_grey_256x256.png must be present in the working directory.

4.4.9.1.3 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.4.9.2Running the example

4.4.9.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_integral_image_cv.cmm script.

OR

Load the apex_integral_image.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo  dd  if=apex_integral_image.bin  of=/dev/sdb  bs=512  seek=0
conv=fsync
```

4.4.9.2.2 Linux OS

Application starts as follows:

```
./apex_sat_cv.elf
```

4.4.9.2.3 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

Example outputs of the SAT demo

4.4.10 APEX IRQ Example

The IRQ example demonstrates the interrupt use for signaling the end of APEX execution instead of polling mechanism. It's behavior is same as the Add example, just the program is not stopped by "Wait()" function, but the IRQ handler is set up to be called immediately after APEX execution stops. The resulting two 16-bit grayscale image is compared and result is printed to the user.

Name of the application directory: **apex_irq**

Available builds:

- S32V234 Standalone

4.4.10.1 Prerequisites

4.4.10.1.1 OS Less application

There are no specific requirements to run this application.

4.4.10.2 Running the example

4.4.10.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_irq.cmm script.

OR

Load the apex_irq.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_irq.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

4.4.11 APEX Lane Departure Warning System

This application implements a lane departure warning system. It reads the input from camera and produces the output displayed on EVB with detected lane using inner contour search on APEX. The result is visually depicted.

Name of the application directory: **apex_isp_ldw_cv**

Available builds:

- S32V234 Standalone

4.4.11.1 Prerequisites

4.4.11.1.1 OS Less application

There are no specific requirements to run this application.

OR

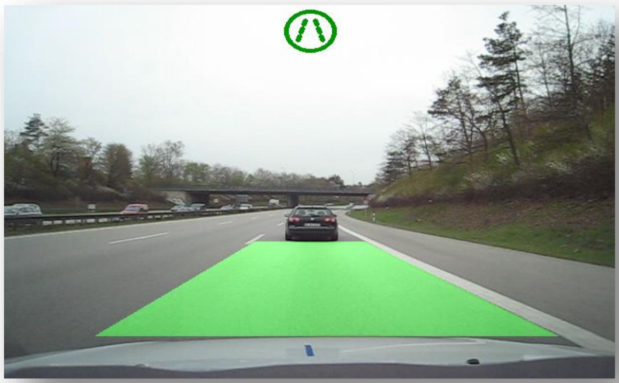
Load the apex_isp_ldw_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:
- `sudo dd if=apex_isp_ldw_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync`

4.4.11.2 Running the example

4.4.11.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_isp_ldw_cv.cmm script.



Example output of the Lane Departure Warning algorithm

4.4.11.2.2 Linux OS

The oal_cma.ko, sram.ko, csi.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod apex2.ko
insmod oal_cma.ko
insmod oal_cma.ko
insmod sram.ko
insmod csi.ko
insmod seq.ko
insmod fdma.ko
```

4.4.12 APEX Face Detection Demo

This application implements a face detection demo with camera input and display output used for demonstration purposes. It reads the input from camera and produces the output displayed on EVB with detected faces using LBP search on APEX. The result is visually depicted.

Name of the application directory: **apex_isp_face_detection_cv**

Available builds:

- S32V234 Standalone

4.4.12.1 Prerequisites

4.4.12.1.1 OS Less application

- There are no specific requirements to run this application.

4.4.12.2 Running the example

4.4.12.2.1 OS Less application

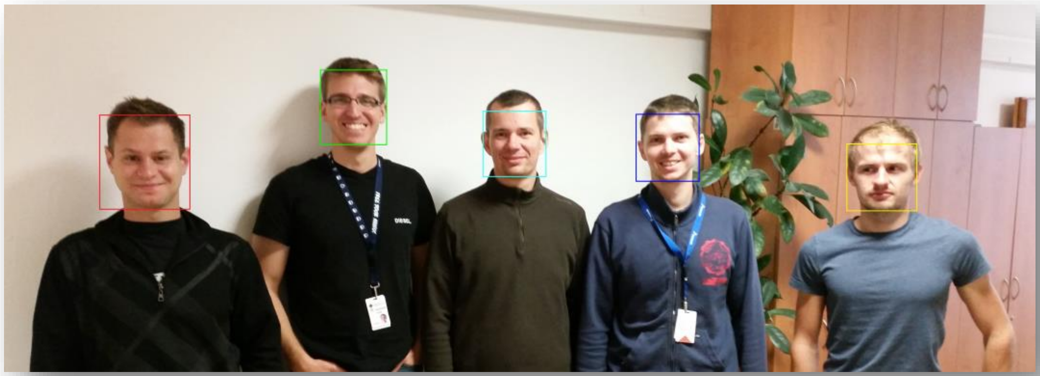
Application is loaded via Lauterbach Debugger by loading apex_isp_face_detection_cv.cmm script.

OR

Load the apex_isp_face_detection_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_isp_face_detection_cv.bin of=/dev/sdb bs=512 seek=0
conv=fsync
```



Example output of the Face Detection algorithm

4.4.13 APEX ORB Matching

The application demonstrates more complicated algorithm – an ORB based image matching based on FAST9 points of interest. The demo loads two input images, detects the FAST9 points and matches both images. The output is one image with highlighted matches. Matching phase can be performed both by the host CPU and the APEX-2. The result is displayed on EVB.

Name of the application directory: **apex_orb_cv**

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 Standalone
- S32V234 INTEGRITY + MULTI Project

4.4.13.1 Prerequisites

4.4.13.1.1 OS Less application

There are no specific requirements to run this application.

4.4.13.1.2 Linux OS

The apex2.ko and oal_cma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod apex2.ko
insmod oal_cma.ko
```

OpenCV shared libraries must be present in the root file system.

Two 768x576 8-bit grayscale input images named f0.png and f1.png must be present in the working directory.

4.4.13.1.3 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.4.13.2 Running the example

4.4.13.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_orb_cv.cmm script.

OR

Load the apex_orb_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_orb_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

4.4.13.2.2 Linux OS

Application starts as follows:

Host matching:

```
./apex_orb_cv.elf
```

APEX-2 matching:

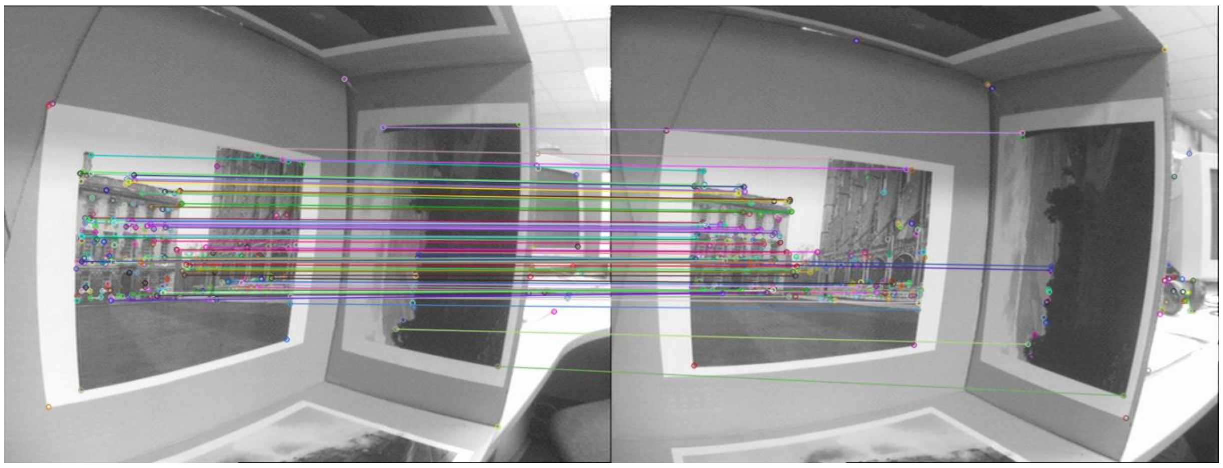
```
./apex_orb_cv.elf -am
```

4.4.13.2.3 Windows OS

Please load into Visual Studio, the project from ./build-deskwin32/mvc/apex_emulation_test.sln, compile and run it

4.4.13.2.4 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.



Example output of the ORB algorithm

4.4.14 APEX ROI

The application demonstrates another simple one-kernel algorithm on the input image, but this version is aimed to demonstrate the Region of Interest settings in the image. The demo loads an image and executes in parallel two 5x5 Gaussian blur APEX kernel on it. The parallel implementation is due to demonstration of two ROI use cases. The output is the blurred image displayed on EVB.

If ROI needs to be specified, there are two possibilities to do so:

- Specify the ROI in input images
    - The DataDescriptor::SetROI(…) function sets the ROI in the image. This function does not alter any of image size indicators, just sets the ROI definition. When the image is used in ConnectIO(…) function, only the ROI part is passed into graph.
    - If the user wants to get a local pointer to the ROI region, a DataDescriptor instance can be passed as a parameter for another DataDescriptor constructor. When done so, a new DataDescriptor have pointers, width, height etc. set accordingly.
    - Warning, the SetFreeOnExit() must be used only once for all derived structures, the data are common.
- Specify the ROI when attaching the images into graph
    - It's also possible to attach the image into process by ConnectIO_ROI() function, where user specifies the region only when attaching the image. The DataDescriptor ROI specification is intact.

The demo is thoroughly commented to present both solutions.

Name of the application directory: **apex_roi_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone
- S32V234 INTEGRITY + MULTI Project

4.4.14.1 Prerequisites

4.4.14.1.1 OS Less application

There are no specific requirements to run this application.

4.4.14.1.2 Linux OS

The apex2.ko and oal_cma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod apex2.ko
insmod oal_cma.ko
```

OpenCV shared libraries must be present in the root file system.

A 256x256 8-bit grayscale input image named in_grey_256x256.png must be present in the working directory.

4.4.14.1.3 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.4.14.2 Running the example

4.4.14.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_roi_cv.cmm script.

OR

Load the apex_roi_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_roi_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

4.4.14.2.2 Linux OS

Application starts as follows:

```
./apex_gauss5x5_cv_roi.elf
```

4.4.14.2.3 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

4.5 ISP Related Examples

4.5.1ISP CSI to DCU

The application demonstrates simple Sony camera frame input combined with image display using DCU. The demo invokes functionality of SDI library, SRAM, FDMA, Sequencer and DCU drivers. The RAW 12 bit camera data is debayered and resized by the Sequencer and through the user application code provided to the DCU driver to be displayed.

Name of the application directory: **isp_csi_dcu**

Available builds:

- S32V234 Standalone

4.5.1.1Prerequisites

Sony camera attached to Mipi CSI0 interface.

4.5.1.2Running the example

4.5.1.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading isp_csi_dcu.cmm script.

OR

Load the isp_csi_dcu.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=isp_csi_dcu.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

4.5.1.2.2 Linux OS

The oal_cma.ko, sram.ko, csi.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

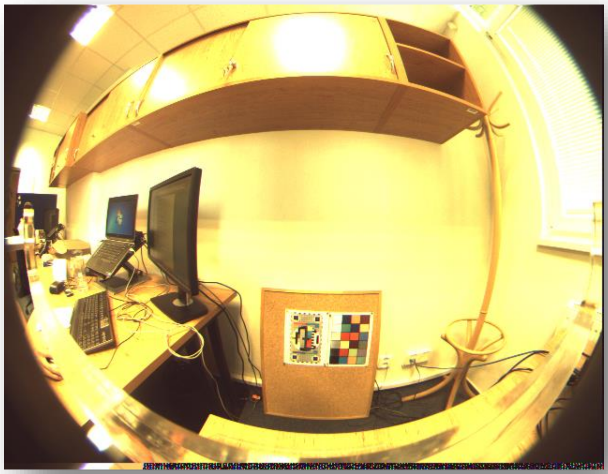
Do it by executing:

```
insmod oal_cma.ko
insmod sram.ko
insmod csi.ko
insmod seq.ko
insmod fdma.ko
```

4.5.1.3Expected input and output

Sony camera attached to MIPI CSI0 interface serves as a source of data. The preprocessed images are being displayed on internal LVDS display.

Example output of the CSI to DCU demo



4.5.2ISP Mipi RAW

The application demonstrates how the raw (unprocessed) data from Sony camera look like. The demo invokes functionality of SDI library and SRAM, FDMA, Sequencer and DCU driver. The RAW 12 bit camera data is being copied without additional processing to DDR buffers which are being displayed through the DCU. Each 12bit pixel data are swapped to 16 bits (MSB aligned). This creates considerable noise in the displayed image.

Name of the application directory: **isp_mipi_ddr**

Available builds:

- S32V234 Standalone

4.5.2.1Prerequisites

Sony camera attached to Mipi CSI0 interface.

4.5.2.2Running the example

4.5.2.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading isp_mipi_raw.cmm script.

OR

Load the isp_mipi_raw.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=isp_mipi_raw.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

4.5.2.2.2 Linux OS

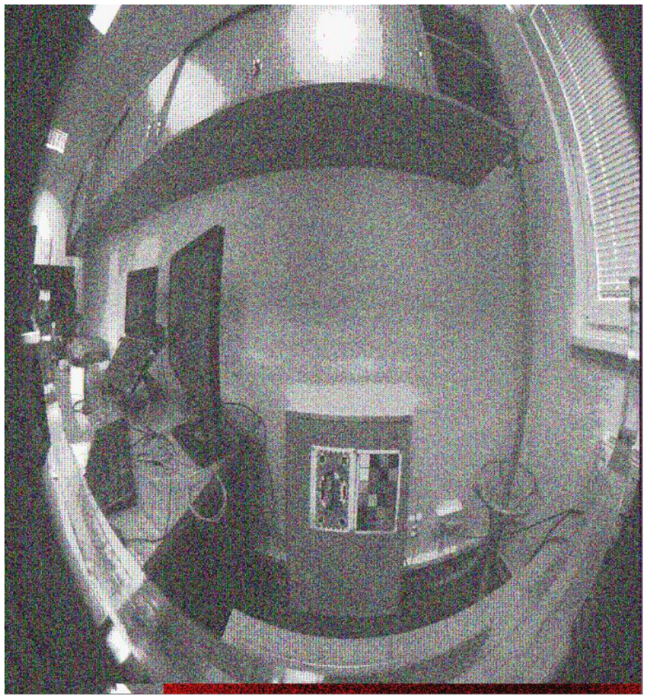
The oal_cma.ko, sram.ko, csi.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod sram.ko
insmod csi.ko
insmod seq.ko
insmod fdma.ko
```

4.5.2.3Expected input and output

Sony camera attached to MIPI CSI0 interface serves as the source of data. The unprocessed (noisy, grayscale, unproportioned) images are being displayed on internal LVDS display.



Example output of the Mipi RAW demo

4.5.3ISP Chroma Key

The application demonstrates preprocessing of Sony camera raw data by ISP. The demo invokes functionality of SDI library and SRAM, FDMA, Sequencer and DCU driver. The RAW 12 bit camera data is processed by the Sequencer (includes mapping of specific color ranges to blue or green) and through the user application code provided to the DCU driver to be displayed.

Name of the application directory: **isp_chroma_key**

Available builds:

- S32V234 Standalone

4.5.3.1Prerequisites

Sony camera attached to Mipi CSI0 interface.

4.5.3.2Running the example

4.5.3.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading isp_chroma_key.cmm script.

OR

Load the isp_chroma_key.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=isp_chroma_key.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

4.5.3.2.2 Linux OS

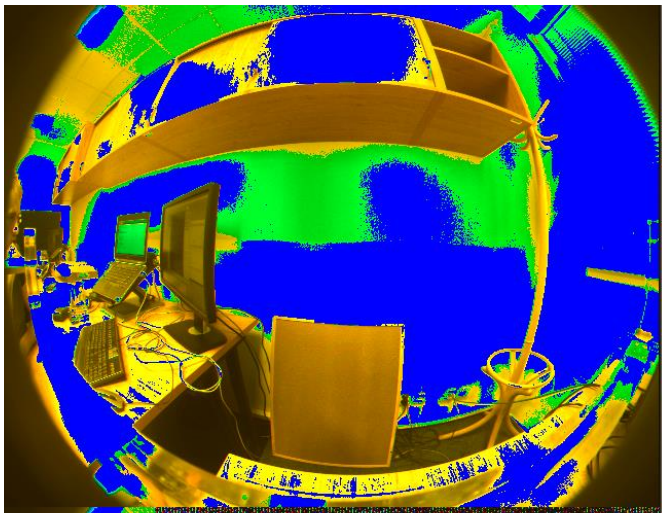
The oal_cma.ko, sram.ko, csi.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod sram.ko
insmod csi.ko
insmod seq.ko
insmod fdma.ko
```

4.5.3.3Expected input and output

Sony camera attached to MIPI CSI0 interface serves as a source of data. The preprocessed images are being displayed on internal LVDS display.



Example output of the Chroma Key demo

4.5.4ISP general purpose registers (GPR)

The application demonstrates use of IPUx general purpose registers to modify the ISP preprocessing parameters online during runtime. The demo invokes functionality of SDI library and SRAM, FDMA, Sequencer and DCU driver. The RAW 12 bit camera data is processed by the Sequencer (includes color marking of over/under exposed areas) and through the user application code provided to the DCU driver to be displayed.

Name of the application directory: **isp_gpr**

Available builds:

- S32V234 Standalone

4.5.4.1Prerequisites

Sony camera attached to Mipi CSI0 interface.

4.5.4.2Running the example

4.5.4.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading isp_gpr.cmm script.

OR

Load the isp_gpr.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=isp_gpr.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

4.5.4.2.2 Linux OS

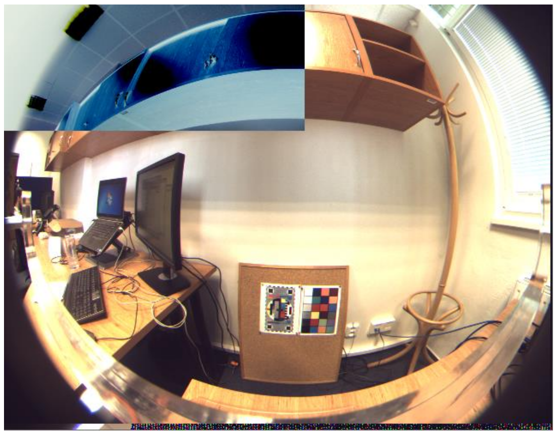
The oal_cma.ko, sram.ko, csi.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod sram.ko
insmod csi.ko
insmod seq.ko
insmod fdma.ko
```

4.5.4.3Expected input and output

Sony camera attached to MIPI CSI0 interface serves as a source of data. The preprocessed images are being displayed on internal LVDS display. A flying/resizing rectangular region with inverted colors is being displayed inside the camera stream. By jumper adjustment the output can be redirected to HDMI connected monitor.



Example output of the GPR demo

4.6 APEX CL Related Examples

4.6.1APEX CL Bitonic Sort

The bitonic sort is an automatic test (no input, no output) using AMD Bitonic sort OpenCL algorithm. The algorithm executes OpenCL code on APEX + SW checker function and outputs the comparison result via UART.

Name of the application directory: **apexcl_bitonic_sort**

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 INTEGRITY + MULTI Project

4.6.1.1Prerequisites

4.6.1.1.1 Linux OS

The apex2.ko kernel module must be loaded prior to application launch.

Do it by executing: `insmod apex2.ko`

4.6.1.1.2 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.6.1.2Running the example

4.6.1.2.1 Linux OS

Application starts as follows:

```
./apexcl_bitonic_sort.elf
```

4.6.1.2.2 Windows OS

Please load the Visual Studio Project from ./build-deskwin32/mvc/apex_opencl_project.sln, compile and run it.

4.6.1.2.3 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

4.6.1.3Prerequisites

4.6.1.3.1 Linux OS

The apex2.ko kernel module must be loaded prior to application launch.

Do it by executing: `insmod apex2.ko`

4.6.1.3.2 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.6.1.4Running the example

4.6.1.4.1 Linux OS

Application starts as follows:

```
./apexcl_conformance_intops.elf
```

4.6.1.4.2 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

4.6.2APEX CL Floyd Warshall pass

The Floyd Warshall is an automatic test (no input, no output) using AMD Floyd Warshall OpenCL algorithm for searching for shortest path in a graph. The algorithm executes OpenCL code on APEX + SW checker function and outputs the comparison result via UART.

Name of the application directory: **apexcl_floyd_warshall**

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 INTEGRITY + MULTI Project

4.6.2.1Prerequisites

4.6.2.1.1 Linux OS

The apex2.ko kernel module must be loaded prior to application launch.

Do it by executing: `insmod apex2.ko`

4.6.2.1.2 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.6.2.2Running the example

4.6.2.2.1 Linux OS

Application starts as follows:

```
./apexcl_floyd_warshall.elf
```

4.6.2.2.2 Windows OS

Please load the Visual Studio Project from ./build-deskwin32/mvc/apex_opencl_project.sln, compile and run it.

4.6.2.2.3 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

4.6.3APEX CL Median Filter

The median filtering sort is an automatic test (no input, no output) using AMD Median filter OpenCL algorithm. The algorithm executes OpenCL code on APEX + SW checker function and outputs the comparison result via UART.

Name of the application directory: **apexcl_median**

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 INTEGRITY + MULTI Project

4.6.3.1Prerequisites

4.6.3.1.1 Linux OS

The apex2.ko kernel module must be loaded prior to application launch.

Do it by executing: `insmod apex2.ko`

4.6.3.1.2 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.6.3.2Running the example

4.6.3.2.1 Linux OS

Application starts as follows:

```
./apexcl_median.elf
```

4.6.3.2.2 Windows OS

Please load the Visual Studio Project from ./build-deskwin32/mvc/apex_opencl_project.sln, compile and run it.

4.6.3.2.3 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

4.6.4APEX CL Memory model

The memory model is an automatic test (no input, no output) using AMD Memory model OpenCL algorithm for demonstrating accesses to various memories in the system. The algorithm executes OpenCL code on APEX + SW checker function and outputs the comparison result via UART.

Name of the application directory: **apexcl_memory_model**

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 INTEGRITY + MULTI Project

4.6.4.1Prerequisites

4.6.4.1.1 Linux OS

The apex2.ko kernel module must be loaded prior to application launch.

Do it by executing: `insmod apex2.ko`

4.6.4.1.2 INTEGRITY RTOS application

There are no specific requirements to run this application.

4.6.4.2Running the example

4.6.4.2.1 Linux OS

Application starts as follows:

```
./apexcl_memory_model.elf
```

4.6.4.2.2 Windows OS

Please load the Visual Studio Project from ./build-deskwin32/mvc/apex_opencl_project.sln, compile and run it.

4.6.4.2.3 INTEGRITY RTOS

Connect the board via MULTI Connect utility and load the application executable.

4.7 ARM NEON Related Examples

4.7.1ARM NEON Gauss 3x3 Filter

The application demonstrates simple one-kernel algorithm on the input image. The demo loads an image and executes the 3x3 Gaussian blur ARM NEON kernel on it. The output is the blurred image displayed on the EVB.

Name of the application directory: **neon_gauss3x3_cv**

Available builds:

- S32V234 Standalone

4.7.1.1Prerequisites

4.7.1.1.1 OS Less application

There are no specific requirements to run this application.

4.7.1.2Running the example

4.7.1.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading neon_gauss3x3_cv.cmm script.

OR

Load the neon_gauss3x3_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=neon_gauss3x3_cv.bin of=/dev/sdb bs=512 seek=0
conv=fsync
```

4.7.2ARM NEON Eigen matrix add

The Eigen add application computes simple matrix addition using Eigen library accelerated by ARM NEON.

Name of the application directory: **neon_eigen_add**

Available builds:

- S32V234 Standalone

4.7.2.1Prerequisites

4.7.2.1.1 OS Less application

There are no specific requirements to run this application.

4.7.2.2Running the example

4.7.2.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading neon_eigen_add.cmm script.

OR

Load the neon_eigen_add.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=neon_eigen_add.bin of=/dev/sdb bs=512 seek=0
conv=fsync
```

4.7.3ARM NEON FFTW library Fourier transform

The application uses FFTW library to compute Fourier transformation on the input image. The demo loads an image and executes forward and afterwards backward Fourier transformation on it. As result, original image and image after backward transformation are displayed on the EVB.

Name of the application directory: **neon_fftw_cv**

Available builds:

- S32V234 Standalone

4.7.3.1Prerequisites

4.7.3.1.1 OS Less application

There are no specific requirements to run this application.

4.7.3.2Running the example

4.7.3.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading neon_fftw_cv.cmm script.

OR

Load the neon_fftw_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=neon_fftw_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5 Known problems and TODOs

- Several demos are not fully operable at Standalone and Linux OS due to subsystems in construction.
- In case the pure Linux BSP 5.1 U-boot is used (not the one from VSDK) isp_chroma_key demo signals error due to low Sequencer clock frequency.