APEX Programming Overview

ABSTRACT:

This document presents the ways to take advantage of the two APEX Vision Accelerator cores present on S32V234 chips.

KEYWORDS:

APEX cores, APEX-CV, ACF

REVISION HISTORY

VERSION	DATE	AUTHOR	COMMENT
1.0	Aug. 29, 2018	Stephane Francois	Initial version

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

The APEX core is a programmable high-performance energy efficient Vision Accelerator core. There are two independent APEX cores featured in S32V234 ADAS chips.

VisionSDK offers scalable approaches to leverage the APEX cores into applications.

1.1 Scope

This document describes how APEX cores can be used and programmed using Vision SDK.

1.2 References

The documents referred across the plan can be found in the following table:

NAME	VERSION	LOCATION
[1] APEX-CV BASE DOCUMENTATION	UG-10328-01-XX	S32V234 SDK
[2] APEX-CV PRO DOCUMENTATION	UG-10328-02-XX	S32V234 SDK
[3] APU-2 C PROGRAMMING GUIDE	UG-10301-00-XX	S32V234 SDK
[4] ACF USER GUIDE	UG-10267-03-XX	S32V234 SDK

Table 1 References Table

1.3 Definitions, Acronyms, and Abbreviations

TERM/ACRONYM	DEFINITION
APU-2	Array Processor Unit
ACF	APEX Core Framework
S32V234	S32V234 SoC
SDK	System Development Kit
ARM	Family of RISC architectures

Table 2 Acronyms Table

1.4 Document Location

The document is located *s32v234_sdk/docs/apex/*

2 APEX Core

The APEX core is a programmable high-performance energy efficient Vision Accelerator core. It is a massively parallel hybrid processor well suited for the processing of large amount of data.

Each APEX core comprises two (2) Array Processor Units (APU), advanced Direct Memory Access (DMA) engines, and other hardware blocks. For more information consult the S32V234 Reference Manual: https://www.nxp.com/docs/en/reference-manual/S32V234RM.pdf

An APU is a scalar-vector hybrid processor with thirty-two (32) 16-bit Computational Units (CU) with their local dedicated Computational Memory (CMEM) for vector, tightly coupled to a 32-bit Scalar RISC processor.

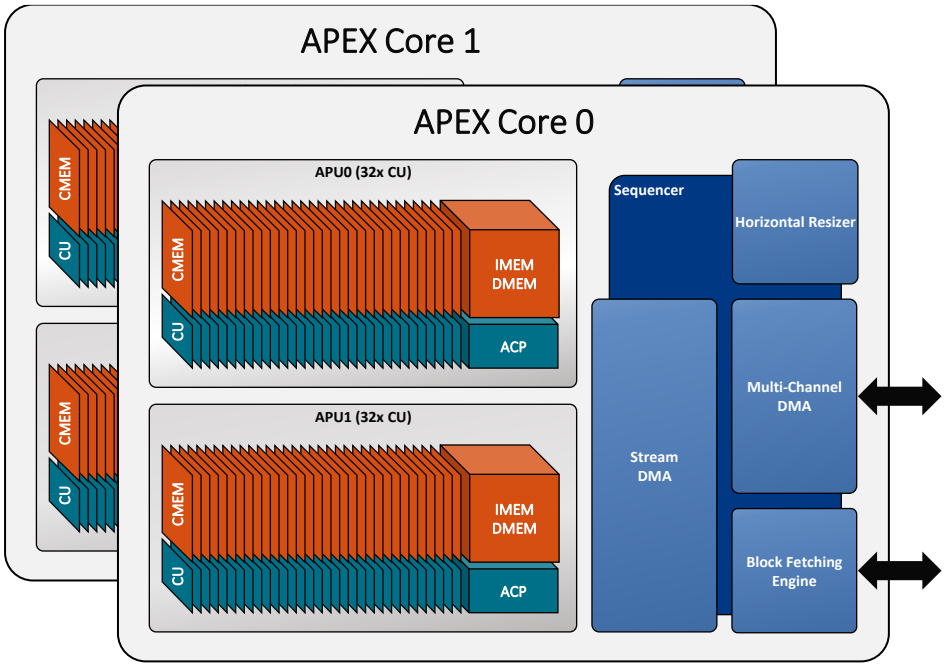

Figure 1: APEX Core Block Diagram

The APEX core offers flexible configuration at runtime. By default, the APEX is configured as a single APU with 64-CU. It is also possible, when using APEX Core Framework library, to change the configuration at runtime and have two independent APUs with 32-CU each.

3 APEX Core Framework (ACF)

ACF is a collection of software libraries and tools providing an easy way to develop for and execute code on the APEX cores.

The APU requires a dedicated compiler to generate compatible executable. Vision SDK installer, as well as S32DS for Vision, come with a version of NXP APU Compiler which is free of charge.

An alternate set of tools, called APU-2 Tools, is available at extra cost. APU-2 Tools features a compiler, cycle accurate instruction set simulators and on-chip debugger software capability. APU-2 Tools is recommended for advanced APEX programming development. Contact your sales representative for more details.

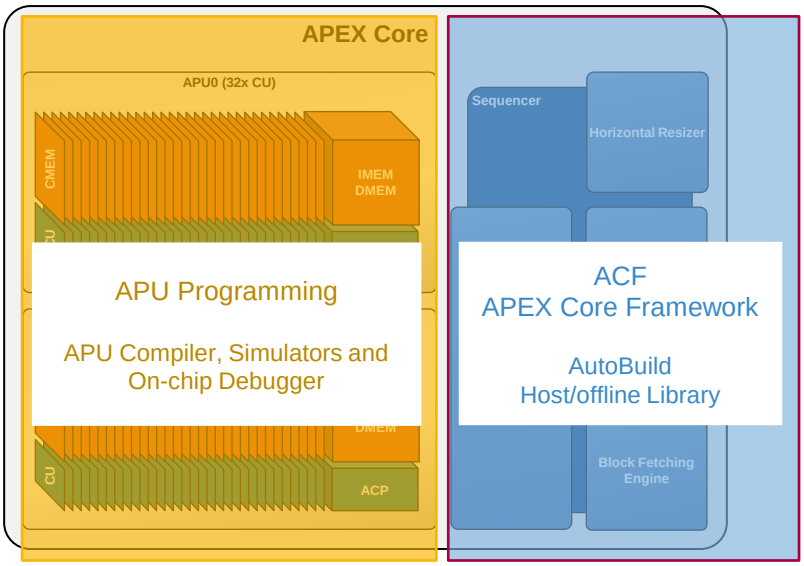

Figure 2: APU Compiler and ACF

ACF is abstracting the APEX hardware to ease the programming of the functionality independent of the hardware. This also allows for a trans-APEX core generation compatibility. ACF removes the painful task of coding DMA transfers thus gaining of development time and performance by chosing automatically optimum hardware resources usage.

Easy programming, vision SDK offers a single interface to build applications, the invocation of the APU compiler and ACF tools is done automatically by the build system.

4 Vision SDK

There are different ways within Vision SDK to develop code which will leverage the APEX core massive computation capacity: APEX-CV library, custom ACF Graphs, custom ACF Kernels.

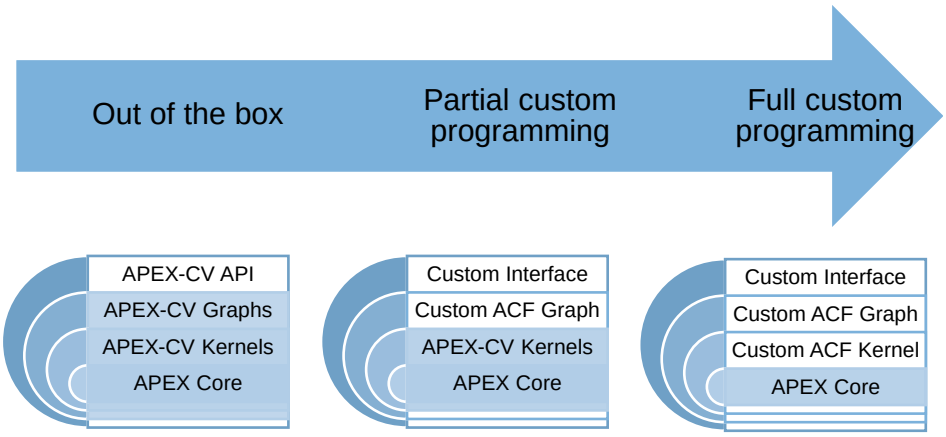

Figure 3: Scalable Development Scheme

APEX-CV Library is the easiest way to leverage the APEX core. The Computer Vision library offers a large number of algorithms easily accessible from the ARM through an intuitive Application Programming Interface (API). Consult the APEX-CV library documentation[1][2] for the list of algorithms and corresponding API. Consult section 5 APEX-CV Library [below] for further details.

Custom ACF Graphs allow for consolidating a group of operations and reduce transfer time to main memory. Custom ACF Graphs are a first step in optimization or required when using Custom ACF Kernels. Consult section 6 Custom ACF Graphs [below] for further details.

Custom ACF Kernels provide the way to execute fully customized code on the APU. Custom ACF Kernels offer access to full programming and possibilities for maximum optimization. Consult section 7 Custom ACF Kernels [below] for further details.

5 APEX-CV Library

APEX-CV library is the easiest way to execute code which will run on the APEX vision accelerator core. In Vision SDK, APEX-CV library comprise a **Base** and **Pro** sub-libraries.

APEX-CV Base provide all the basic and simpler computer vision operations whereas APEX-CV Pro contains the most advanced functionality.

APEX-CV library is provided as an API and as a kernel library. APEX-CV Base can be easily used with the API, and also as a ACF Kernel library when creating Custom ACF graphs. APEX-CV Pro is mostly intended to by used via the API.

The API has been developed to provide ease of use and clarity without compromising on performance. Most classes provide a separate initialization and process methods, to initialize the object and to perform the computation respectively. To change the source and destination variables being used, the **reconnectIO** method shall be used. The Initialize() method is executed once, some dynamic allocations take place at this stage, then using the initialized object through the other methods doesn't trigger additional memory allocations. Memory is released when the object is released.

Most of the classes are supporting image size from 128-pixel to **2048**-pixel wide. Height is not a limiting factor for many of the classes, however some will be limited, please look at the class documentation.

APEX-CV is similar in some ways to **OpenCV**. Here are some key points:

- APEX-CV use a similar virtual image container UMat than OpenCV. This image container allows to refer to an image in memory without having a image in the ARM virtual space and potentially cached.
- It is possible to intermix APEX-CV and OpenCV function, it is recommended to get a Mat object for executing OpenCV functions, and then release the Mat object and use the UMat object for APEX-CV functions.
- A key difference with OpenCV is that APEX-CV function require the output buffer to be defined before calling them. This is mainly to control more tighly when and where memory allocation is taking place.

APEX-CV library is provided pre-compiled in Vision SDK, so the use of an APU compiler if not necessary.

There are **examples** provided in Vision SDK showing how to use APEX-CV classes, from basic use to full application with sensor input and display. Please refer to the folders: s32v234_sdk\demos\apexcv_base and s32v234_sdk\demos\apexcv_pro, as well as more advanced applications found in . s32v234_sdk\demos\apps.

APEX-CV classes have some support to target a specific APEX core on the chip. The method **SelectApexCore**() allows to set which core to use when the next method Process() is invoked. It is possible to change core for each call to Process(). Note that the number of CU is always 64 when using APEX-CV API. APEX-CV Process() method is blocking so to run multiple object in parallel, it is required to have the calls done in different threads.

Relevant Documentation: APEX-CV Base documentation, APEX-CV PRO documentation

6 Custom ACF Graphs

Using Custom ACF Graphs is a way to add new functionality and optimize code running on the APEX cores.

In a nutshell, an ACF Graph is a description of processing flow where the input and output are defined and referring to existing processing blocs, ACF Kernels. As APEX-CV Base also provides ACF Kernels, it is possible to create Custom ACF Graphs using any of these existing kernels.

It is highly encouraged to consult ACF User GUIDE to understand better the context and details about the APEX Core Framework.

To use Custom ACF Graphs, it is necessary to have an APU compiler. There two possible compilers for the APU: NXP APU Compiler (nxp), included with Vision SDK, and APU-2 Tools (tct). Consult 3 APEX Core Framework (ACF) for more details.

There are examples provided in Vision SDK showing how to use Custom ACF Graphs, from basic use to more complex. Please refer to the folder: s32v234_sdk\demos\apex. The required new binary encapsulating the desired functionality will be generated by ACF's Autobuild which is integrated into Vision SDK build system. This is achieved by adding a graphs folder to the project with its BUILD.mk file and APU target sub-folder, see code example mentioned above.

When using Custom ACF Graphs, it is possible to indicate which APEX core to use with its APU configuration. It is possible to use 64-CU or 32-CU, the latter allows to have two computation run concurrently on each APU of a single APEX core. See SelectApuConfiguration() documenation for more details. It is also possible to invoke **parallel execution** from the single thread by calling Start() for all ACF process to initiate processing and then call Wait(). See s32v234_sdk\demos\apex\apex_add as example.

Relevant Documentation: ACF User GUIDE

7 Custom ACF Kernels

Using Custom ACF Kernels provides full customization to add new functionality and optimize code running on the APEX cores.

By writing your own ACF Kernels you can add your own algorithm and leverage the processing power of the APEX cores.

It is highly encouraged to consult APU-2 C Programming Guide and ACF User GUIDE to understand better the context and details to implement Custom ACF Kernels.

There are existing ACF Kernels source code available in Vision SDK. Please refer to the folder: s32v234_sdk\kernels\apu\ where you'll find the kernels for APEX-CV Base and Pro, as well as additional kernels libraries.

Relevant Documentation: APU-2 C Programming Guide, ACF User GUIDE