APU Math Library User Guide

ABSTRACT:
This document presents the information related to the APU Math Library.
KEYWORDS:
Product documentation, user guide
APPROVED:

Revision History

VERSION	DATE	AUTHOR	CHANGE DESCRIPTION
1.0	Oct. 16, 2018	Pham Vu Trung Kien	Initial version

Table of Contents

APU Math Library **1**
1 Introduction **4**
1.1 Purpose 4
1.2 Audience Description 4
1.3 Document location 4
1.4 References 4
1.5 Definitions, Acronyms and Abbreviations 4
2 Fixed-Point Arithmetic Theory **5**
2.1 Principles of Fixed-Point Arithmetic 5
3 APU Math Library Implementation **7**
3.1 Data Types 7
3.2 Data Types Manipulation Examples 7
3.3 APU Math Library Access 7
4 Demo application **8**
4.1 Demo flow chart 8
4.2 Demo struct 8
4.3 Demo design 9
4.4 Demo input 10

1 Introduction

The APU Math Library offers multiple arithmetic operations with an implementation in fixed-point suitable for use in custom APU code.

1.1 Purpose

Provide background information on the fixed-point arithmetic and how it is used in APU code.

1.2 Audience Description

This document is intended for external use.

1.3 Document location

This document is located in Vision SDK at: vsdk/s32v234_sdk/docs/apex/apu

1.4 References

Id	***Title***	***Location***
1	*The APEX-CV Base Library*	*vsdk/s32v234_sdk/docs/apex/apex_cv/UG-10328-01-11_APEX-CV_Base_Library.pdf*
2	*The APEX-CV Pro Library*	*vsdk/s32v234_sdk/docs/apex/apex_cv/UG-10328-02-13_APEX-CV_Pro_Library.pdf*
3	*The APU Math Library Reference Manual*	*vsdk/s32v234_sdk/docs/apex/apu/rm_apu_math_lib.pdf*

Table 1: References

1.5 Definitions, Acronyms and Abbreviations

Term/Acronym	***Description***
Math lib	*APU math library, using fixed point arithmetic*

Table 2: Acronyms

2 Fixed-Point Arithmetic Theory

Fixed-point arithmetic allows to manipulate fractional numbers on processors without a Floating Point Unit. Fixed-point arithmetic provides fast and energy efficient computation as it is based on integers.

2.1 Principles of Fixed-Point Arithmetic

In computer base-2 arithmetic, fractional quantities can be expressed using a pair of integers (n, e) where n represents the mantissa and e the exponent. The pair represents the fraction: $n2^{-e}$.

The exponent e can be considered as the number of digits you have to move into n before placing the binary point.

For fixed-point numbers, the exponent e is usually denoted by the letter q; in this document we will use the letters p, q and r for exponent. The subsections below will show how to perform basic arithmetic operations on two fixed-point numbers, a = $n2^{-p}$ and b = $m2^{-q}$, expressing the answer in the form c= $k2^{-r}$.

2.1.1 Change of exponent

The simplest operation to be performed on a fixed-point number is to change the exponent. To change the exponent from p to r (to perform the operation c = a) the mantissa k can be calculated from n by a simple shift.

Since:

$$n2^{-p} = n2^{r-p} \text{ x } 2^{-r}$$

You have the formula:

when (r ≥ p)

$$k = n \ll (r - p)$$

and when (p > r)

$$k = n \gg (p - r)$$

2.1.2 Addition and subtraction

To perform the operation c = a + b, first convert a and b to have the same exponent r as c and then add the mantissas.

$$n * 2^{-r} + m * 2^{-r} = (n + m) * 2^{-r}$$

Subtraction is similar

$$n * 2^{-r} - m * 2^{-r} = (n - m) * 2^{-r}$$

2.1.3 Multiplication

The product c = a * b can be performed using a single integer multiplication. From the equation:

$$\text{ab} = n2^{-p} \times m2^{-q} = (\text{nm})\, 2^{-(p+q)}$$

It follows that the product n*m is the mantissa of the answer with exponent p+q. To convert the answer to have exponent r, perform shifts as described above.

For example, if p + q >= r:

$$k = (n * m) \gg (p + q - r)$$

2.1.4 Division

Divisions, c = $\frac{a}{b}$, can also be performed using a single integer division. The equation is:

$$\frac{a}{b} = \frac{n2^{-p}}{m2^{-q}} = \left(\frac{n}{m}\right) * 2^{\,q-p} = \left(\frac{n}{m}\right) * 2^{\,(r+q-p)} * 2^{\,-r}$$

In order not to lose precision, the multiplication by $2^{(r+q-p)}$ must be performed before the division by m.

For example, assuming that r + q ≥ p, perform the calculation:

$$k = (n \ll (r + q - p))/m$$

2.1.5 Square root

The equation for square root is:

$$\sqrt{a} = \sqrt{n2^{-p}} = \sqrt{n2^{(2r-p)}} * 2^{-r}$$

In other words, to perform c = $\sqrt{a}$, set k = isqr (n << (2r-p)) where *isqr* is an integer square root function.

3 APU Math Library Implementation

3.1 Data Types

APU Math Library uses specially defined data types to clarify the range of the mantissa and the exponent. The data types express the signness and the number of bits forming the integer and the fractional part organized as follow: vfxp_<sign><m>q<n>

Where:

sign: u - unsigned/ s - signed

m: number of bits allocated for the integer part

n: number of bits allocated for the fractional part

Example: vfxp_u1q15

Note: the data types are aliases to standard datatype such as uint16_t, conversion between type needs to be handled by the programmer as no automatic conversion will be done.

3.2 Data Types Manipulation Examples

Let's consider the following example.

Floating-point Notation:

float fA = 1.08434f

float fB = 1.07326f

float fSum = A + B = 2.1576f

Fixed-point Notation:

Using vfxp_u1q15, find representation of A and B (Q = 15):

vfxp_u1q15 aF = (vfxp_u1q15)round(A * 2 ^ Q) = 35532

vfxp_u1q15 bF = (vfxp_u1q15)round(B * 2 ^ Q) = 35169

vfxp_u1q15 sum = aF + bF = 70701

Fixed-point to Floating-point:

vfxp_u1q15 sumF = (float)sum / 2 ^ Q = 2.15762f

3.3 APU Math Library Access

APU Math Library is a header file including inline functions.

File: <vfxp_math_inline.h>

Path: <vsdk/s32v234_sdk/kernels/apu/vfxp_math_lib/include/>

4 Demo application

To illustrate the use of the APU Math Lib, an example is included in VSDK, at vsdk/s32v234_sdk/demos/apex/apex_math_lib

4.1 Demo flow chart

The APU Math Lib is accessible from custom APU Kernel code.

In this demo, a custom kernel is created. This kernel is used in conjunction with APEX-CV Lib kernels, for the computation of sobel_x and sobel_y. All the kernels are in a custom ACF Graph. The host application code will execute the processing code on APEX core.

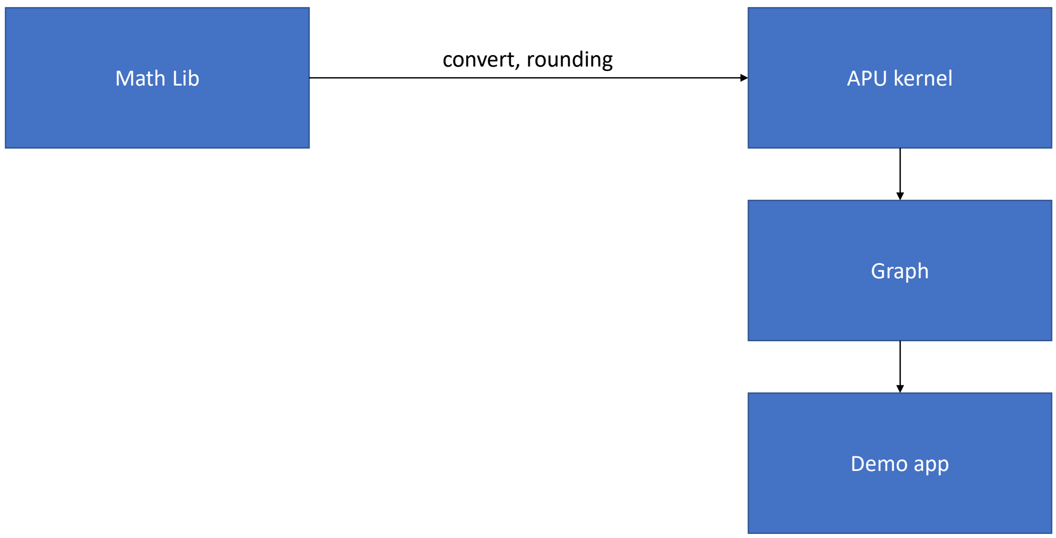

Figure 1: Demo Flow Chart

4.2 Demo File Structure

```
SDK_ROOT
 |___ demos
    |_____apex
        |_____ apex_math_lib
            |_____ build-v234ce-gnu-linux-d
            |_____ build-v234ce-gnu-linux-o
            |_____ build-v234ce-gnu-sa-d
```

|______ build-v234ce-gnu-sa-o
|______ graphs
|______ kernels
|______ src
|______ BUILD.mk

4.3 Demo Processing Details

The Graph as illustrated below, uses three kernels. Sobel_X and Sobel_Y from APEX-CV library, and a 'math_demo' custom kernel. This kernel computes for each pixel a value using the ComputePixelVal function which illustrate the use of few functions from the APU Math library such as sqrt, atan2 and exp.

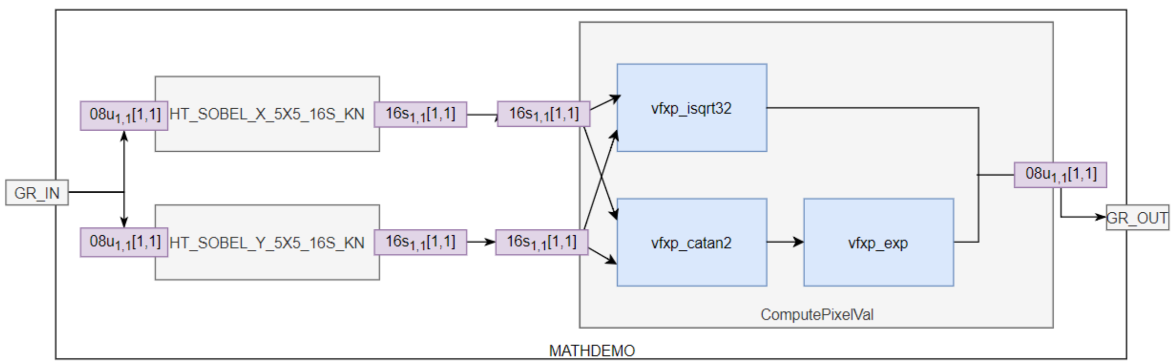

Figure 2: Demo ACF Graph

- Use APEX-CV Sobel to compute G_x and G_y. The Sobel Operators G_x, G_y are defined as:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

- Create new 'Math Demo' kernel combines the outputs of the Sobel kernels to give gradient magnitude (using sqrt function: vfxp_isqrt32) and compute angle value depend factor (using atan2 function: vfxp_catan2 and exp function: vfxp_exp):

$$M = \sqrt{{G_x}^2 + {G_y}^2}$$

A = vfxp_catan2(G_y, G_x)

B = AngleSelect(A) //group of operations giving emphasis on +/-45°

C = vfxp_exp (B) //further emphasis on the selected angles

O = M * C //combines magnitude and selected angle

4.4 Demo Input

The demo will use as default the image illustrated below, but it is possible to use any other image by indicating path and filename at the command line.

- Input file: star_512x512.png
- Input size: 512 x 512

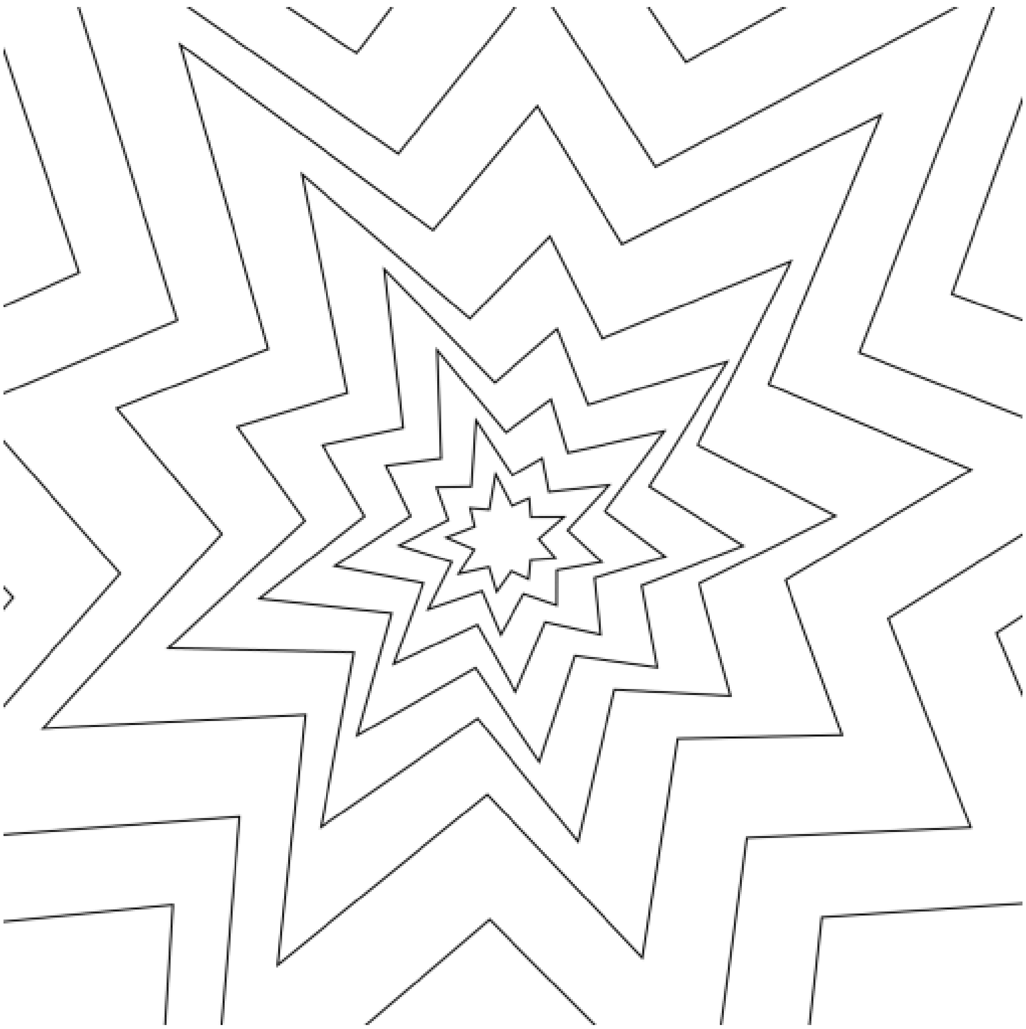

Figure 3: star_512x512.png