APU-2 Sample Code Overview

UG-10301-02-04

Revision History

Version	Details of Change	Author	Date
01	Initial release	Warren Hulme	Mar. 31, 2013
02	Update for APU2 Tools R2.1	Warren Hulme	Apr. 17, 2013
03	Update for APU2 Tools R2.2	Warren Hulme	July 3, 2013
04	Update for APU2 Tools J-2014.09	Lee, Ki-ju	Feb. 11, 2015

Table of Contents

1 General Document Information 5
1.1 Overview 5
1.2 Acronyms 5
1.3 References 5
1.4 Scope 5

2 System Overview Example 6
2.1 Example Code 6
2.1.1 Overview 6
2.1.2 Algorithms 7
2.1.3 Compile and execute 7
2.1.4 Profiling 7

Table of Tables

Table 1-1: Acronyms 5
Table 2-1: Profile Results 8
Table 2-2: Profile Breakdown 8

1 General Document Information

1.1 Overview

The purpose of this document is to provide guidance in compiling, running, and profiling the sample code, using the APU-2 C Tools, notably the CHESS Development Environment.

1.2 Acronyms

The following acronyms are used in this document, or in documents related to this release.

Acronym	Definition
APU	Array Processing Unit
FLT	Filter
UTIL	Utilities

Table 1-1: Acronyms

1.3 References

[1] "APU2 Engagement Package Release Notes", CogniVue document UG-10301-03

[2] "APU2 Tool User Guide", CogniVue document UG-10301-01

[3] "OpenCV Image Filtering" http://docs.opencv.org/modules/imgproc/doc/filtering.html#sobel

1.4 Scope

This document applies to the sample code provided with the APU-2 Engagement Package. See the APU-2 Engagement Package Release Notes for relevant version information.

2 System Overview Example

For more information on Sobel filters, see [3].

See section 2.1 for description of example code. The example code provides all 4 Sobel filters in two different versions – a generic filter implementation (reference), and an optimized implementation.

The Sobel filters used in the example code are:

- 3x3 in x, apu_flt_sobel_3x3_x
- 3x3 in y, apu_flt_sobel_3x3_y
- 5x5 in x, apu_flt_sobel_5x5_x
- 5x5 in y, apu_flt_sobel_5x5_y

2.1 Example Code

2.1.1 Overview

The example code in the release package runs 4 different Sobel filters on a test image tile, and compares the results with the pre-calculated OpenCV results.

The 4 Sobel filters implemented are:

- 3x3 horizontal (X) Sobel filter
- 3x3 vertical (Y) Sobel filter
- 5x5 horizontal (X) Sobel filter
- 5x5 vertical (Y) Sobel filter

Each filter is implemented and run twice. The first implementation is a generic filter implementation, while the second represents an implementation optimized for the specific filter coefficients.

The installation directory contains the following files:

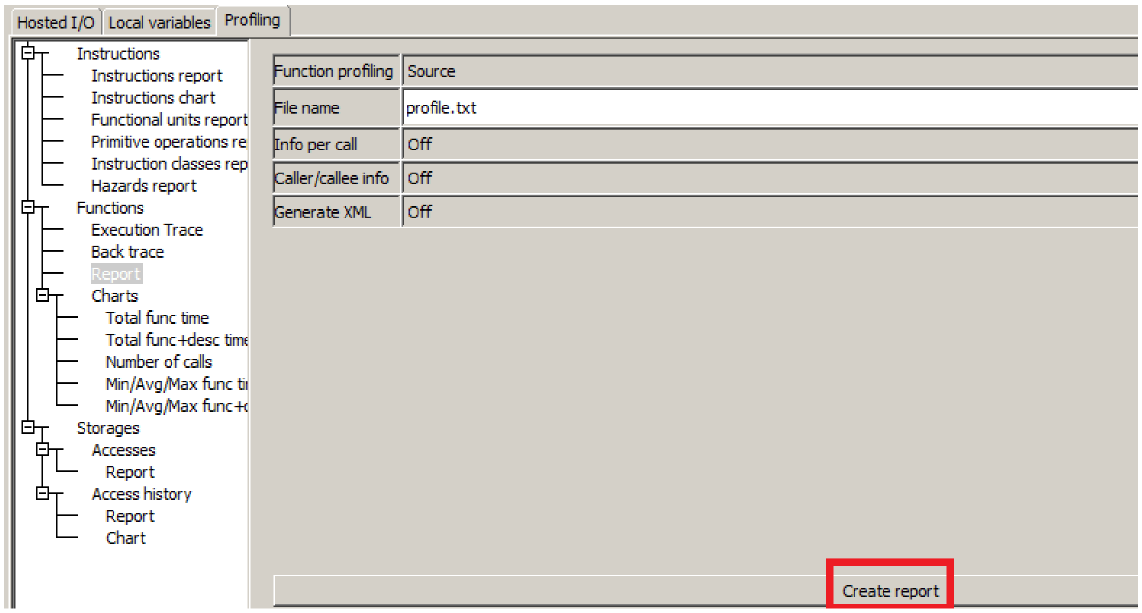
- apu_lib.h – contains prototypes for the filter, color conversion, histogram, integral image implementations.
- apu_filter_generic.cpp – contains source code for generic filter implementations.
- apu_filter_sobel.cpp – contains optimized implementations of specific Sobel filters.
- apu_test_util.cpp – contains methods for padding CMEM, and for comparing the output with reference results.
- apu_test_util.h – contains prototypes for padding, comparison and transfer methods.
- example.cpp – contains the top level code for invoking the tests, and outputting the results.
- example.prx – Chess project file.
- example_data.h – contains the input and reference data sets.
- example_data.s – contains the hard-coded source buffer contents.

Note that the doc directory contains generated documentation for the source code.

2.1.2 Algorithms

2.1.3 Compile and execute

a. Go to "File →Open → Project" in menu of ChessDE
b. Select a project \samples\example\example.prx
c. Go to and Select "Compile → Rebuild" or Press "Shift + F7"
d. After finishing compilation, Press F5 to start simulation
e. Go to and Select "View → Profiling"
f. Press F6 to run the example project.
g. After finishing the execution, Select " Profile tap → Report" then Press Create report



Function profiling	Source
File name	profile.txt
Info per call	Off
Caller/callee info	Off
Generate XML	Off

h. The results of profiling would be save as \samples\example\profile.txt.

See section 4 of "APU-2 Engagement Package Release Notes" for details on the compiler warnings related to chess_prepare_for_pipelining.

2.1.4 Profiling

Input/Output resolution : 8x4

The profiling values for the code, running in APU2 12.1R2, are as follows:

```
Cycles % of total Function
 18232      5.71%      apu_filter_fir2 void_apu_filter_fir2___Pvec08u___s
  2072      0.65%      apu_flt_sobel_5x5_y void_apu_flt_sobel_5x5_y___Pve
  1941      0.61%      apu_flt_sobel_5x5_x void_apu_flt_sobel_5x5_x___Pve
   710      0.22%      apu_flt_sobel_3x3_x void_apu_flt_sobel_3x3_x___Pve
```

```
       676     0.21%      apu_flt_sobel_3x3_y void_apu_flt_sobel_3x3_y___Pve
```

	Generic Filter Implementation	Optimized Implementation
Sobel 3x3 in X		434
Sobel 3x3 in Y		438
Sobel 5x5 in X		1494
Sobel 5x5 in Y		1382
Total Cycles:	**13860**	**3748**

Table 2-1: Profile Results

Note that profiling of the generic apu_filter_fir2() call only shows the total number of cycles spent in the function, which represents the total time spent in the function across all 4 calls. In order to extract the specific performance for each filter, the other calls in test_filter_generic() must be commented out, so that only the relevant filter is passed to the generic function. (Alternately, each separate call to test_filter_generic() can be placed in a wrapper function.) In this way, the following numbers may be extracted.

	Generic Filter Implementation	Optimized Implementation
Sobel 3x3 in X	2057	434
Sobel 3x3 in Y	2057	438
Sobel 5x5 in X	4873	1494
Sobel 5x5 in Y	4873	1382
Total Cycles:	**13860**	**3748**

Table 2-2: Profile Breakdown