The parallelization of binarization using a GP-GPU

Seong Hyeon Han¹, Suk Won Yoo²*

¹Department of Computer Engineering, Seoekyeong University, Seoul, Korea ²Department of Computer Science, Seokyeong University, Seoul, Korea {cromcruach, swyoo*}@skuniv.ac.kr

Abstract

In this paper, we propose the optimized binarization in the GP-GPU. Because the binarization is easily parallelized, we propose two ways of binary operations that utilize GP-GPU. The first method was to divide data load, subtraction and conversion, data store. The second method was processed collectively. The second method was 2.52 times faster than the first method. After synthesizing the GP-GPU to the FPGA, the GP-GPU on the binarization were compared with the binarization on the ODROID XU. The binarization on the GP-GPU was 1.89 times faster than the binarization on the ODROID XU.

Keywords: Binarization, SIMT, Processor, YUV Color Space, GPU Computing

1. Introduction

There is a large range of recent researches on GPU Computing utilizing GPU(Graphic Processing Unit) for general purpose of operations. The reason that the operation speed of GPU is faster than that of CPU is because GPU has SIMT(Single Instruction Multiple Thread) structure[1][2]. The reason that the operation speed of GPU is faster than that of CPU is because GPU has SIMT(Single Instruction Multiple Thread) structure. SIMT is a special structure that executes a command through several threads. Utilization of SIMT structure can perform quickly same operation with different data like repetitive statements.

Binarization in digital image processing is an operation that converts intensity values greater than or equal to a given thresholding value to white color and also less intensity values to black in gray image. Binarization in digital image processing is an operation that converts intensity values greater than or equal to a given thresholding value to white color and also less intensity values to black in gray image. Utilization of GP-GPU can decrease operation time because binarization operation is executed repeatedly.

2. Binarization

2.1 YUV 4:2:0 planar format

Input image used in this thesis has YUV 4:2:0 planar format and VGA resolution. Y area of YUV color space represents brightness of light, U area does color difference of red color, and V area does that of blue color[3]. Y area has data size of resolution size, and also each of U and V area has data size of half of horizontal resolution and half of vertical resolution. Each area has values in the range of 0 and 255.
Therefore, values can be stored in data structure of one byte. Fig. 1 shows data size of YUV 4:2:0 planar format and also Y, U, and V areas used to represent pixels on display.

![Figure 1. YUV 4:2:0 planar format](image)

### 2.2 Binarization

Binarization used in this thesis calculates intensity difference of two adjacent pixels in Y area and then converts it to 255 if it is greater than or equal to a given thresholding value or does it to 0 if otherwise. This operation can be represented by the following pseudo-code[4].

```
For index = 0 to Width x Height
    If ( | Y[index+2] - Y[index] | > Threshold )
        Y[index] = 255
    Else
        Y[index] = 0
```

![Figure 2. Pseudo-code of Binarization](image)

Fig. 3 shows sample input image, gray scaled image(Y area), and binarized image respectively, and Fig. 4 does a portion of data of gray scaled image and binarized image.

![Figure 3. Sample image, gray scaled Image, binarized image](image)
3. Binarization on GP-GPU

3.1 Memory Structure of GP-GPU

Memory of GP-GPU in this thesis is composed of code memory, shared memory, and thread memory in Fig. 5.
Code memory area is the command memory of GP-GPU, and it stores machine code that is the compiled version of application code.

Thread memory area is the memory area that each thread has, and it stores local variables and global variables declared in application code. Size of thread memory area depends on the number of thread. This GP-GPU has maximum 256 threads. Let’s suppose that there are 256 number of threads and each thread has 4 KByte area. Then, size of thread memory becomes 1 MBytes.

Shared memory area is the domain that all threads can be accessible to, and it is input/output path of data. Thread ID is stored in special register of SP(Scalar Processor). Address used in memory operation is calculated by addition of base address and thread ID as offset address. Fig. 6 shows memory operation of GP-GPU.

![Figure 6. Memory operation of GP-GPU](image)

3.2 Parallelization of binarization using GP-GPU

Binarization using GP-GPU is composed of data load, subtraction, conversion, and data store in Fig. 7.

![Figure 7. Process of binarization](image)

Data load and data store steps load data using address obtained from thread ID in shared memory. Subtraction step can access in two ways in Fig. 8. First way is that one thread performs data load operation and subtraction operation for 1200 number of pixels and then does conversion operation and data store operation for 1200 number of pixels. Second way is that one thread performs data load operation, subtraction operation, conversion operation and data store operation for one pixel and then repeat these operations 1200 number of times.

First way should store calculation result of subtraction operation. The calculation result is stored in 1 Byte data structure because Y area has values in the range of 0 and 255. However, result of subtraction operation of two values should be stored in 2 Bytes data structure because the result is in the range of –255 and 510.
GP-GPU has 256 number of threads, and VGA resolution has 307,200 number of pixels. If it is assumed that all threads are used, then one thread should process 1200 number of pixels. GP-GPU has 256 number of threads, and VGA resolution has 307,200 number of pixels. If it is assumed that all threads are used, then one thread should process 1200 number of pixels.

Second way can reduce memory usage because it stores result of conversion operation in shared memory directly and also it uses only one 2 Byte data structure.

First way takes longer run time because all threads should wait until data load and subtraction operations are finished. However, second way takes shorter run time because all threads do not need to wait until these operations are finished.

![Figure 8. Two ways of binarization on GP-GPU](image)

Binarization using GP-GPU can be represented by the following pseudo-code.

For $i = 0$ to 1200

$\text{index} = i \times 256 + \text{Thread ID}$

$\text{subtraction} = Y[\text{index} + 2] - Y[\text{index}]$

If ($|\text{subtraction}| > \text{Threshold}$)

$Y[\text{index}] = 255$

Else

$Y[\text{index}] = 0$

![Figure 9. Pseudo-code of binarization using GP-GPU](image)

4. Design and Verification of GP-GPU

4.1 Verification of Simulation Design

Let’s suppose that those two ways in Fig. 8 are called A and B respectively. Simulations for experiments of A and B are performed by GP-GPU at 100Mhz using Test Drive Profiling Master[5]. Table 1 shows comparison of simulation time and memory usages for A and B.

| A     | 157,604 | 8,300 | 2,400 | 307,200 |
|-------|---------|-------|-------|---------|
| B     | 62,32-  | 5,490 | 2     | 307,200 |

It shows that simulation time of B is approximately 2.52 times faster than that of A. Memory usage of B except shared memory is approximately 51% of memory usage of A.
4.2 FPGA Design Verification

After performing composition using Xilinx Vivado Design Suite 2016.1[6] by the B way previously mentioned in 4.1 Verification of Simulation Design, experiment is implemented using Xilinx FPGA board, VC-707 XC7VX485T FPGA board[7]. FPGA is controlled by Test Drive Profiling Master. FPGA works at 100MHz. Fig. 10 shows initial status of GP-GPU in the experiment.

![Figure 10. Initial state of GP-GPU](image)

ODROID XU[8] board equipped with ARM Cortex-A15 Quad Core and ARM Cortex-A7 Quad Core is used as comparison group of the experiment. ODROID XU and GP-GPU are compared in terms of CPU clock, run time, and memory usage using pseudo-code in 3.2 Parallelization of binarization using GP-GPU.

### Table 2. Comparative experiment on ODROID XU and GP-GPU

|         | Clock (MHz) | Run time (us) | Memory usage (Bytes) |
|---------|-------------|---------------|----------------------|
| ODROID XU | 1,700       | 57,000        | 614,464              |
| GP-GPU  | 100         | 30,008        | 312,692              |

Binarization using GP-GPU is 1.89 times faster than that using ODROID XU. Memory usage for binarization using GP-GPU is approximately 51% smaller than that for binarization using ODROID XU.

5. Conclusion

Binarization using GP-GPU has been experimented in two ways. Faster binarization between these two ways has been executed on GP-GPU composed with FPGA. After that, it has been compared with ODROID XU equipped with ARM Cortex-A15 Quad Core and ARM Cortex-A7 Quad Core. As a result, GP-GPU has 1.89 times faster run time and uses 51% smaller memory usage than ODROID XU does.
The parallelization of binarization using a GP-GPU

References

[1] DoHyun Kim, ChiYong Kim, Design of a SIMT architecture GP-GPU using Tile based on graphic pipeline structure, Journal of IEEE. (2016), Vol.20, No.1, pp.75-81
[2] YunSeop Hwang, HeeKyeong Jeon, Kwanho Lee, Kwangyeob Lee, Implementation of the SIMT based image signal processor for the image processing, Journal of IEEE. (2016), Vol.20, No.1, pp.89-93
[3] Seon-gye Hwang, Image processing by Visual C++, Hanbit Media, Seoul (2011), Vol. 11, pp.601-602
[4] Seon-gye Hwang, Image processing by Visual C++, Hanbit Media, Seoul (2011), Vol. 13, pp.640-643
[5] https://sourceforge.net/projects/test-drive/
[6] http://www.xilinx.com/support/download.html
[7] http://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html#productTable)
[8] http://www.hardkernel.com/main/products/prdt_info.php?g_code=G137510300620