The 7th Annual IEEE International Conference on
Cyber Technology in Automation, Control and Intelligent Systems
July 31-August 4, 2017, Hawaii, USA

Implementation of Sobel Edge Detection on FPGA based on OpenCL

Baoshan You [1], Weihua Sheng[2], Hongwei Ma [1], Ye Gu[3] ， Yinglin Qin [1]

1. School of Computer Science and Technology ,Shandong Jianzhu University, Jinan,250101, Shandong ,China,youbaoshan@sdjzu.edu.cn

2. School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078, USA, weihua.sheng@okstate.edu

3. Shenzhen Academy of Robotics, Shenzhen, 518057, China, ye.gu@szarobots.com

Abstract*—In this paper, Sobel edge detection is implemented on an FPGA using a DE1-SoC development board from TERASIC. OpenCL as a new scheme is used for implementation. The development process and implementation details are described. The results show that OpenCL is an efficient method for FPGA development compared to traditional Hardware Description Language (HDL). We apply the OpenCL programming to FPGA-based Sobel edge detection, which gains a significant performance increase compared to that of software methods. The work of this paper can be extended to more complicated algorithms.

***Keywords*—OpenCL; Sobel; FPGA;Embedded System**

I. Introduction

Recent years have seen the progress of Artificial Intelligence, especially in computer vision, natural language processing and speech recognition. Commercial AI products have emerged, such as Siri from Apple, Cortana from Microsoft, Google Now from Google and so on. Deep learning has been a key technology in these AI applications, which requires incredible amounts of data and computing power [1]. It is an even bigger challenge to implement AI algorithms in embedded systems. The hard options for algorithm implementation include central processing units (CPUs), graphic processing units (GPUs), digital signal processors (DSPs), and field-programmable gate arrays (FPGAs). Among them, FPGA is a very competitive solution which can accelerate algorithms with very limited power consumption. In this context, FPGA is a very ideal choice, with their inherent capability to facilitate the launching of many concurrent processes at a low power profile.

However, FPGA has its own disadvantages when it comes to implementing an application. Programming FPGAs for complex algorithms is not straightforward. Traditional applications are developed in FPGA with low level Hardware Description Languages (HDL). HDL is a low level and error-prone language for complex algorithms, which renders it inefficient and significantly affects the time to market for an application [2]. Fortunately, current design tools for FPGAs, such as those from Altera and Xilinx, have made them more compatible with the high-level software practices. Especially the OpenCL programming model has made FPGAs more accessible to software engineers and system architects. Traditional software codes can be easily ported to FPGA systems using the OpenCL programming model without the low level understanding of FPGAs. Since the FPGA architectures are configurable, this also means that researchers can explore model-level optimization beyond what is possible on fixed architectures such as GPUs. In addition, FPGAs can provide more computational power per watt of energy consumption, which is particularly important for battery-powered embedded systems and data centers that require enormous amounts of energy.

This paper demonstrates the use of OpenCL to program an FPGA that implements a Sobel edge detection algorithm in an embedded system. The DE1-SoC FPGA development board from TERASIC is adopted. This paper demonstrates the flow of development of a typical image processing algorithm on FPGA using OpenCL.

A. Edge detection

Edge detection is a method that detects the locations where the image brightness sharply changes, or the discontinuities in an image. Most of the shape information of an image is reflected in edges. Edges are one of the most important features in image analysis and computer vision. They play a very important role in many applications of image processing [3][4]. Conventional edge detection methods examine all the image pixels for sudden changes by comparing pixels with their neighbors [5][6]. This is often done by detecting the maximal value of gradient such as the Prewitt Operator, Sobel Operator, Robinson Compass Masks, Krisch Compass Masks, Laplacian Operator and so on, all of which are classical linear filters or smoothing filters. Sobel Operator is adopted in this paper.

B. OpenCL

The OpenCL standard is the first open, royalty-free, unified programming model for accelerating algorithms on heterogeneous systems. OpenCL allows the use of a C-based programming language for developing codes across different platforms, such as CPU, GPU, DSP, and FPGA [7].

As a programming model for software engineers and a methodology for system architects, OpenCL is based on standard ANSI C (C99) with extensions to extract parallelism. OpenCL also includes an application program interface (API) for the host to communicate with the hardware accelerator, traditionally over PCI Express, or one kernel to communicate with another without host interaction. ARM-based System-on-Chip (SoC) is supported by OpenCL as well.

In the OpenCL model, the user schedules tasks to command queues. There is at least one command queue for each device. The OpenCL run-time then divides the data-parallel tasks into pieces and sends them to the processing elements in the device. This is the method for a host to communicate with any hardware accelerator. In this paper, the Intel FPGA SDK for OpenCL is adopted which conforms to the OpenCL 1.0 standard.

II. Implementation of Image Edge Detection Using Sobel Algorithm

This paper presents an image edge detection system based on OpenCL. An embedded DE1-SoC development board [8] from Terrasic is used. The DE1-SoC board is a robust hardware design platform built around the Altera System-on-Chip (SoC) FPGA, which combines a dual-core Cortex-A9 microcontroller with industry-leading programmable logic for ultimate design flexibility. Users can leverage the power of tremendous re-configurability paired with a high-performance, low-power processor system. Altera's SoC integrates an ARM-based hard processor system (HPS) consisting of processor, peripherals and memory interfaces tied seamlessly with the FPGA fabric using a high-bandwidth interconnect backbone [8]. OpenCL is fully supported by the DE1-SoC board.

In the OpenCL model, the system is divided into two parts. One is the host computer system, the other is the hardware accelerator. The host computer system offloads data-parallel tasks to hardware accelerator. The hardware accelerator returns the computation result to the host. Within the DE1-SoC board, the host is the Cortex-A9 embedded core and the hardware accelerator is the FPGA. Fig. 1 shows the relation between the two parts. According to the two-part OpenCL scheme, the design also has two parts. One part is the host program and the other is the kernel program which runs in the Accelerator part.

This paper demonstrates image edge detection with the Soble algorithm. The Soble algorithm involves convolution calculation which needs a significant amount of computing power, so this part is implemented within FPGA as a Kernel program. The rest is implemented as a host program.

A. Kernel program implementation.

The OpenCL standard is divided into two parts. A device side language and the host side language or API.

On the device side, the language is commonly referred to as the Kernel Code or OpenCL C. This is the portion that is based on the C99 standard with certain restrictions. The kernels are mapped to a wide range of accelerators dependent on the hardware.

The Kernel program is implemented within FPGA. The Sobel operator is used in edge detection. It has been researched for parallelism [9]. Due to the size of the image, it is slow for a traditional processor. This work aims to speed up the edge detection by using the Kernal program on the FPGA [10].

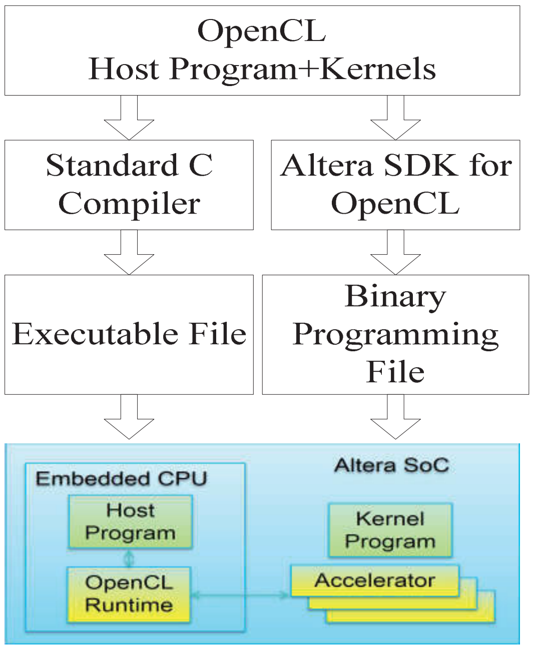


Fig. 1. The Embedded Host and the Accelerator

The Sobel operator is used to perform a 2-D spatial gradient measurement on an image to emphasize the regions of high-spatial frequency, which correspond to edges. Technically, it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. Employing this Sobel operator at any pixel of an image will produce the corresponding grayscale vector and its normal vector.

The operator consists of a pair of 3x3 convolution kernels as shown in Fig.2 below.

-1	0	+1
-2	0	+2
-1	0	+1

x filter

+1	+2	+1
0	0	0
-1	-2	-1

y filter

Fig. 2. Filters of Sobel Edge Detection

The two filters compute both the vertical edges (the left one) and horizontal edges (the right one). The filters are convolved with the entire input image. The middle pixels are weighted more heavily than the outer pixels, since they are closer to the pixel in the center of grid and thus have a higher impact on the edge gradient for that pixel.

The detailed calculation is shown below in Equation (1), (2) and (3).

$$Gx=[f(x+1,y-1)+2*f(x+1,y)+f(x+1,y+1)]-[f(x-1,y-1)+2*f(x-1,y)+f(x-1,y+1)] \quad (1)$$

$$Gy=[f(x-1,y-1)+2f(x,y-1)+f(x+1,y-1)]-[f(x-1,y+1)+2*f(x,y+1)+f(x+1,y+1)] \quad (2)$$

$$G = sqrt(Gx\hat{}2+Gy\hat{}2) \quad (3)$$

According to equation (1), X filter of Fig.2 is used as the convolution kernel to get the convolution value of Gx. The detailed process is illustrated in Fig.3. The Center element of the kernel is placed over the source pixel. The convolution result +4 is calculated as in Fig.3 with a weighted sum of itself and the nearby pixels. Gy can be obtained similar to Gx.

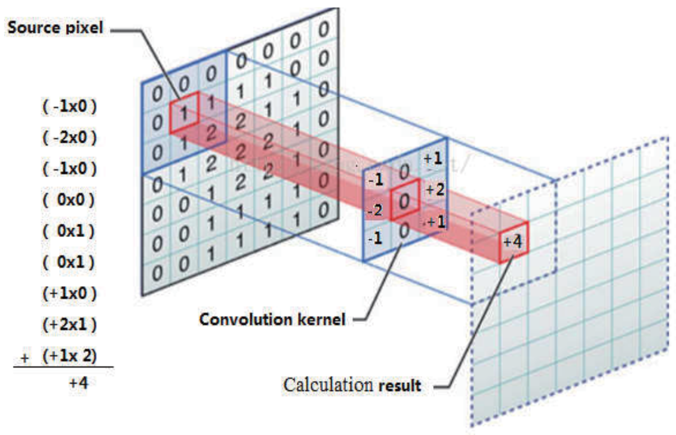


Fig. 3. Gx convolution illustration

In order to improve the efficiency, this paper uses equation (4) as an approximation of the above Equation (3). If G is larger than a certain threshold, we then consider this pixel as an edge pixel.

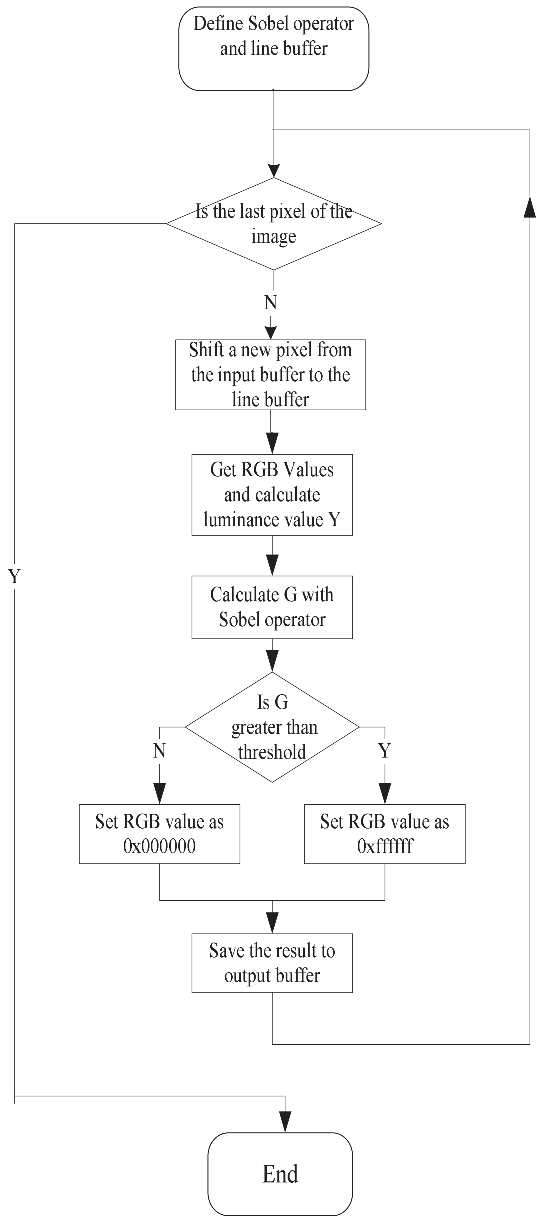
$$|G| = |Gx| + |Gy| \quad (4)$$

An input RGB (8 bits per component) image is used as the input. First, the R, G and B values are obtained from the RGB image. Then the Y, U and V values are calculated with a conversion formula. In the paper, the luminance value Y can be calculated from the YUV/RGB conversion formula (5). To reduce the resource usage, the floating point math operation is avoided. The approximation conversion formula (6) is used to calculate the luminance value Y. To implement image edge detection, the Sobel operator is used on each pixel's luminance value Y of the whole image.

$$Y = 0.299*R + 0.587*G + 0.114*B \quad (5)$$

$$Y = ((\ 66 * R + 129 * G + \ 25 * B + 128) >> 8) + \ 16 \quad (6)$$

The single work-item OpenCL kernel is used to detect the edges of the input image. A monochrome image output is the result of edges detection. The kernel uses a sliding-window line buffer to efficiently compute the convolution of a pixel's luminance value. The result of the convolution is compared against an input threshold value that can be controlled by the user. The kernel program is developed with C language. Fig.4 is the specific flow chart for the kernel program.



Define Sobel operator and line buffer
Is the last pixel of the image
N
Y
Shift a new pixel from the input buffer to the line buffer
Get RGB Values and calculate luminance value Y
Calculate G with Sobel operator
Is G greater than threshold
N
Y
Set RGB value as 0x000000
Set RGB value as 0xffffff
Save the result to output buffer
End

Fig. 4. Kernel Sobel Edge Detection program flow

The Intel FPGA Software Development Kit (SDK) for OpenCL 16.1 is used to compile the kernel file. The FPGA configuration file (.aocx) is produced during the compiling process. With the use of the Sobel operator which consists of a pair of 3x3 convolution kernels, the line buffer in Fig.4 should be equal or greater than two rows and 3 pixels to meet the convolution calculation requirement. The kernel program uses rows [2 * COLS + 3] as the line buffer. COLS is the number of pixel in one row. For example, the image resolution is 640×480. So, the row number is 480, COLS is 640. G in Fig.4 is the value obtained from Equation (4).

B. Host program implementation

The host Cortex-A9 embedded core runs the Ubuntu operating system. To develop the host program for DE1-SoC, the Intel SoC FPGA Embedded Development Suite (SoC EDS) is used.

According to the OpenCL specification, there are four models: Platform model, Execution model, Memory model and Programming model [11].

The platform model defines the roles of the host and the devices. There is a single host that coordinates execution on one or more devices. In the OpenCL execution model, a host defines an abstract container, called a context which is configured on the host and enables it to pass commands and data to control the device. The context manages several resources, including the device. Kernels are functions that are executed on the devices. The kernels are extracted from program objects which are either compiled in run time or precompiled as in the case of Altera's implementation. Think of the program as a dynamic library. Memory objects encapsulate data. They are only valid in one context and allows OpenCL to make runtime movements to and from specific devices. The host can request action by the device through the command queue. Each command queue can only be associated with one device. If a host needs to perform an action such as executing kernels or memory transfers, it will submit commands to the proper command queue. Data storage and movement on the device are also explicitly specified.

To run an OpenCL program on the device from the host, there are seven steps which are divided into two different layers, the platform layer and the runtime layer. In the platform layer, we need first query and select the vendor specific platform. Then, within the platform select the proper devices, and finally create a context that manages those devices.

In the run time layer, we need to create a command queue for each device, write data to the device using memory objects, launch the kernels in a parallel fashion, and finally read result from the device[11][12]. According to the steps mentioned above, the host program is designed as in Fig.5. It mainly focuses on steps of the OpenCL, the other parts of the code are omitted. The key parts of the host OpenCL program are the last several steps in Fig.5. The host dispatches the calculation task to FPGA with the flowing steps. First, send image data to be calculated to the FPGA device, then launch the kernel in FPGA to calculate the data, finally, read back the result to the host. The details are showen in Fig.5.

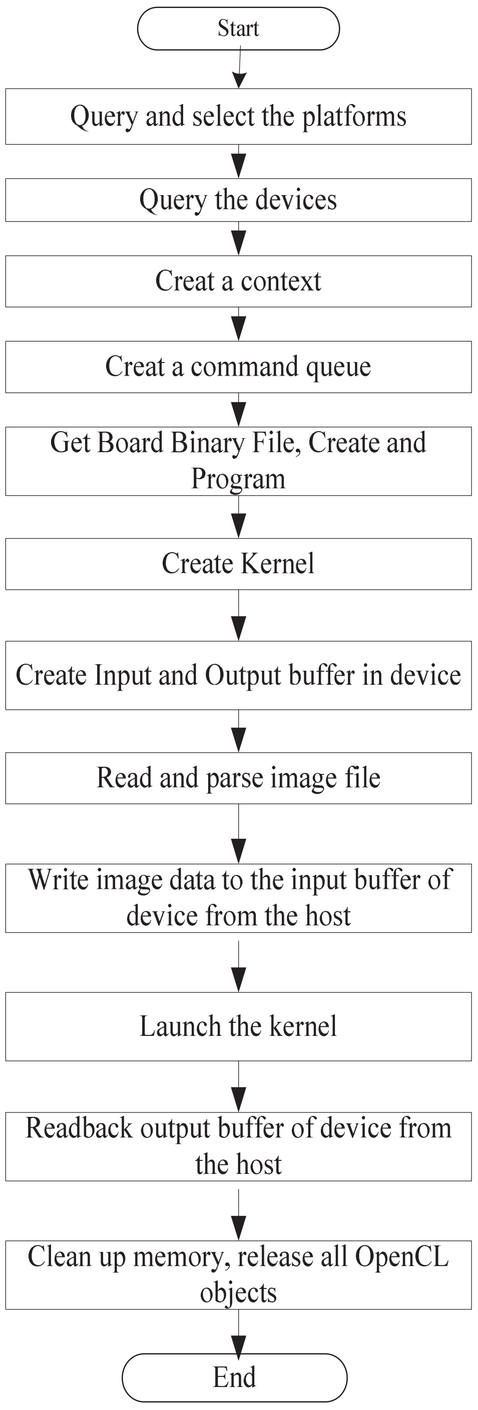

Start
Query and select the platforms
Query the devices
Creat a context
Creat a command queue
Get Board Binary File, Create and Program
Create Kernel
Create Input and Output buffer in device
Read and parse image file
Write image data to the input buffer of device from the host
Launch the kernel
Readback output buffer of device from the host
Clean up memory, release all OpenCL objects
End

Fig. 5. The host program flow chart

III. SYSTME DEPLOYMENT

In the experiment, the Sobel Edge Detection system is implemented on the DE1-SoC development board. The Ubuntu operating system with minimal desktop LXDE is adopted. To deploy the system, we follow the following steps:

1: Compile the host code with Intel SoC FPGA Embedded Design Suite (SoC EDS).

2: Compile OpenCL Kernel code with Altera SDK for OpenCL Kernel Compiler. Usually the compile command is as follows:

Aoc device/sobel.cl --sw-dimm-partition -o bin/sobel.aocx

A .aocx file will be generated. To display the picture properly, the .aocx file cannot be used, which will be loaded automatically with the running of the host code. The loading of .aocx file will influence the display of picture. To solve the problem, the command "*quartus_cpf*" is used as follows:

quartus_cpf -c sobel.sof -o bitstream_compression=on sobel.rbf

With the "*quartus_cpf*" command, sobel.rbf is generated,which is used to configure the FPGA during booting from SD card with the DE1-SoC.

3: Build Linux_SD Card image file, which includes supporting of Altera SDK OpenCL 16.1 and LXDE.

4: Load Linux_SD Card image file to SD Card with utility "Win32DiskImager".

5: Copy sobel.rbf file to SD Card FAT partition.

6: Insert SD Card into DE1-SoC board and boot from SD Card with the Linux image.

7: Copy host file and .ppm image file from the host PC to the DE1-SoC Linux System by typing Linux "scp" command through Ethernet or USB storage.

8: Initial the OpenCL system with command "source ./init_opencl.sh" and run the host program.

IV. EXPERIENTIAL RESULTS

Several images with different resolutions were tested to get the compare the detection time between the hardware method and the software method. Fig.6 is the input and output image of Sobel edge detection. The result is shown in Table I and Fig.7. It can be seen that there is dramatic performance increase using FPGA (the hardware method). Especially with the increase in the size of the picture, the performance is increased significantly.

Meanwhile, Table I and Fig.7 show that the hardware implementation is not sensitive to the image size. With the performance above, real-time image processing can be achieved easily with OpenCL on FPGAs. The synthesis result is shown in Table II. It shows that only a small percentage of the FPGA resource is used. It means that there is room for further performance improvement.

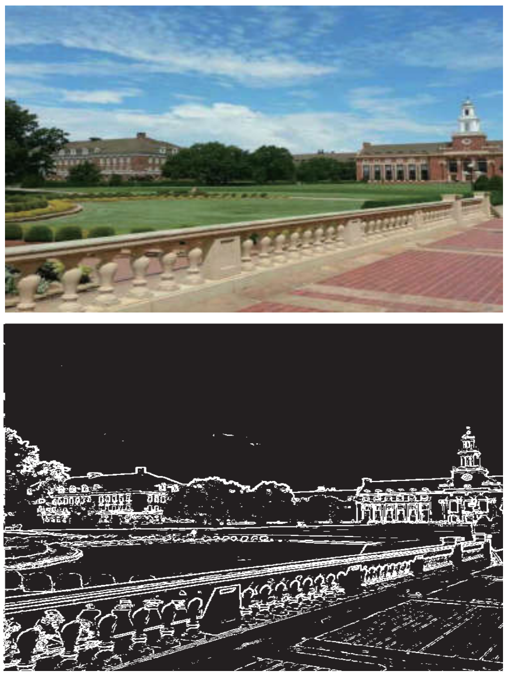


Fig. 6. Input image and output image

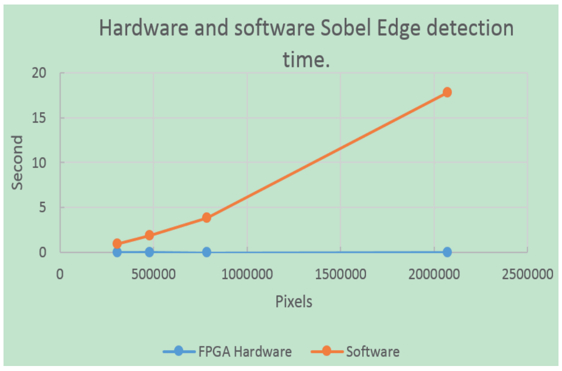


Fig. 7. Scatter chart of Sobel Edge Detection time

TABLE I. HARDWARE & SOFTWARE EDGE DETECTION TIME (SECOND)

File resolution	FPGA implementation	Software implementation
640x480	0.002	0.977
800x600	0.003	1.861
1024x768	0.005	3.843
1920x1080	0.015	17.817

TABLE II. FPGA RESOURCE USAGE

Synthesis parameters	FPGA resource usage	Percentage of FPGA resource usage
Logic utilization	5144	16%
Block memory bits	269814	7%
DSP Blocks	5	5%

V. CONCLUSION

In this paper, Sobel edge detection is implemented on FPGA using the DE1-SoC development board. OpenCL as a new scheme is used for the implementation of the algorithm on FPGA. The development process and details are described. The results show that OpenCL is an efficient method to FPGA development and the Sobel edge detection based FPGA gains a significant performance boost. It can be used in many complex computer vision systems. In the future, we will investigate the implementation of Deep Learning algorithms using OpenCL and FGPA.

ACKNOWLEDGMENT

This work is supported by China Scholarship Council(CSC) Grant, the National Natural Science Foundation of China (NSFC) Grants 61328302, 61222310 and the Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China (No. ICT170314) and the Shenzhen Overseas High Level Talent (Peacock Plan) Program (No. KQTD20140630154026047).

REFERENCES

[1] G Lacey, GW Taylor, S Areibi ,Deep Learning on FPGAs: Past, Present, and Future,2016, https://arxiv.org/abs/1602.04283.

[2] S. O. Ayat, M. Khalil-Hani and R. Bakhteri, "OpenCL-based hardware-software co-design methodology for image processing implementation on heterogeneous FPGA platform," *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, George Town, 2015, pp. 36-41.

[3] M. B. Ahmad and T. S. Choi, “Local threshold and boolean function based edge detection,” IEEE Transactions on Consumer Electronics, vol. 45, no. 3, pp. 674–679, 1999.

[4] T. A. Abbasi and M. U. Abbasi, “A novel FPGA-based architecture for Sobel edge detection operator,” International Journal ofElectronics, vol. 94, no. 9, pp. 889–896, 2007.

[5] Pratt, W. K. (2004). Digital Image Processing, John Wiley &Sons, Inc.

[6] Rafael C. Gonzalez, Richard E. Woods. Digital Image Processing(2nd Edition) Prentice Hall, 2nd edition (January 15, 2002)

[7] What Is OpenCL.https://www.altera.com/products/design-software/embedded-software-developers/opencl/overview.html

[8] DE1-SoC board.http://www.terasic.com.tw

[9] Yangli, Yangbing. “Study of FPGA based Parallel Processing of Sobel Operator” AI Modern Electronics Technique 2005.J.

[10] D. T.Saegusa, T.Maruyama, Y.Yamaguchi, “How fast is an FPGA in image processing?”, IEICE Technical Report, Vol.108.No.48,2008,pp.83-88

[11] The OpenCL Specification. Version 1.0. https://www.khronos.org.

[12] Intel FPGA SDK for OpenCL Cyclone V SoC Getting Started. Version 16.1.https://www.altera.com