Journal of Korea Multimedia Society Vol. 19, No. 7, July 2016(pp. 1154-1158)
http://dx.doi.org/10.9717/kmms.2016.19.7.1154

# Design and Implementation of Image-Pyramid

Bongkyu Lee†

## ABSTRACT

This paper presents a System-On-a-chip for embedded image processing applications that need Gaussian Pyramid structure. The system is fully implemented into Field-Programmable Gate Array (FPGA) based on the prototyping platform. The SoC consists of embedded processor core and a hardware accelerator for Gaussian Pyramid construction. The performance of the implementation is benchmarked against software implementations on different platforms.

**Key words:** Image Pyramid, Multi-resolution, Visual System, Gaussian Pyramid

## 1. INTRODUCTION

The Scale Invariant Feature Transform, or SIFT algorithm, has rapidly been adopted in the machine vision community as the "best-in-class" standard for feature detection and matching. Feature detection has a variety of applications in image processing across many domains, from object recognition and tracking in robotics to create photo mosaics in consumer photography applications [1]. However, since SIFT algorithm involves an intensive time-consuming module, the Gaussian pyramid [2], current SIFT software implementations typically involve the use of a high power, general-purpose processor to achieve less-than-real-time performance. However, for many embedded applications, this is an unacceptable solution. A computing power of embedded processors is much less powerful than that of desktop or laptop computers. In addition, almost of current embedded processors do not have hardware supporting floating point computation in order to reduce power consumption and chip size [3]. With a hardware implementation of the Gaussian pyramid, the full power of the SIFT algorithm could be available to mobile sensing platforms without the overhead of a separate, high power processor [1].

A few hardware implementations [1, 4, 5, 6, 7] were proposed for constructing image pyramids. In [1], they implemented the hardware for generating a variation of Gaussian pyramid. However, the implemented hardware accepts fixed sized images so that it can be applied to various applications. In [4], they implemented image pyramid generation unit (IPGU) as a part of face detection hardware. However, since it only performed size reductions without convolution operations, the aliasing problem can be occurred. In [5], they implemented the hardware that generates 2-levels pyramid. In [6], they implemented pipeline schemed hardware for construction of the Gaussian pyramid. In [7], they designed the 3-levels pyramid generation hardware embedded into their stereo matching hardware system.

Although many implementations were proposed, they have common problems to be used for embedding various applications. For embedding into various devices, the hardware implementation has a

※ Corresponding Author : Bongkyu Lee, Address: (690-756) Jejudaehak-ro 120, Jeju-si, Jeju-Do, TEL : +82-64-754-3593, FAX : +82-64-725-2579, E-mail : bklee@jejunu.ac.kr
Receipt date : May 9, 2016, Revision date : June 4, 2016
Approval date : July 11, 2016
† Dept. of Computer&Statistics, Jeju National University
※ This research was supported by the 2015 scientific promotion program funded by Jeju National University

component for connecting to processors, memory systems and even external devices embedded into applications. Second, it is needed to accept various sized images, since cameras attached to mobile sensing devices have several resolutions, such as 320×240, 640×480 or 160×120. Finally it is also needed that the Gaussian pyramid supports different levels of pyramids according to the applications.

In this paper, we describe the design and implementation of a Gaussian pyramid processor for various embedded image processing applications. The implementation steps are as follow. First, we designed and implemented the prototyping platform based on a Field Programmable Gate Array (FPGA). The platform was used for testing and debugging the target SoC architecture in the register transfer level. We evaluated the performance of the implemented SoC in the register transfer level using the platform.

The remainder of the paper is as follows. In Section II, we review the computation process of Gaussian pyramid. Section III explains the proposed Gaussian pyramid processor. The emulation result is presented in Section IV, and final discussions are given in Section V.

## 2. THE GAUSSIAN PYRAMID

Gaussian pyramid consists of reduced copies of an image, where the size of an image decreases in a factor $s$ from one level to the next, and a low-pass filter is convolved with level for avoiding aliasing problems [8]. Gaussian pyramid construction can be described as follows. Suppose the image is represented initially by the array $g_0$ which contains $C$ column and $R$ rows of pixel. Pyramid level 1 contains image $g_1$ which is reduced and low-pass filtered version of $g_0$. The level-to-level computation is performed by the function REDUCE [8].

$$g_k = REDUCE(g_{k-1}) \tag{1}$$

which means, for levels $0 < l < N$ and pixels $(i,j), 0 \leq i < C_l, 0 \leq j < R_l$,

$$g_l(i,j) = \sum_{m=-\frac{M}{2}}^{m=\frac{M}{2}} \sum_{n=-\frac{N}{2}}^{n=\frac{N}{2}} w(m,n) g_{l-1}(2i+m, 2j+n) \tag{2}$$

$N$ refers to the number of levels in the pyramid, $M$ is the length of the kernel window and $w$ is the pattern of weights which resemble the Gaussian probability density functions, while $C_l$ and $R_l$ are dimensions of the $i^{th}$ level. The shape of a graphical representation of this process is given in Fig. 1.

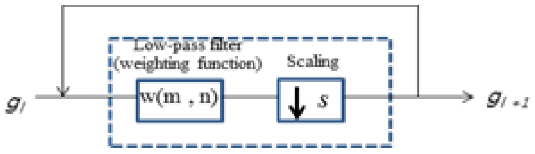


Fig. 1. Iterative pyramid generation.

## 3. THE GAUSSIAN PYRAMID PROCESSOR

### 3.1 The prototyping emulation platform

SoC design is the trend for the development of embedded systems [9]. To design and verification of SoC, FPGA-based emulation platform has become popular in co-verification and rapid prototyping [10]. Mapping the entire design of the target SoC into an FPGA gives an accurate and fast representation.

For the prototyping platform based on a FPGA, the basic components, - including CPU, several system buses and associated interconnection blocks, - were selected according to the design steps. We used LENO2 (32-bit RISC processor) and its related Floating Point Unit (FPU) for base processors. Register transfer level modules of processors were implemented into the FPGA (XILINX X2CV8000). The operational clock rate of processors was 30MHZ. For acquisition of images, a MICRON MT9V112 image sensor was connected to the emulation platform using an I$^2$C bus which was implemented in the FPGA.

The emulation platform has the SDRAM-based memory (128 Mbytes, SAMSUNG 16bits × 2) and the Flash-based storage (8 Mbytes, INTEL STRATA 16bits × 2). SDRAM-based memory unit is used for storing images captured by the image sensor, while Flash-based storage is used for storing codes (program) and data. Fig. 2 shows the implemented emulation platform with compact size of 112×129 mm. We designed small interactive bootstrap program for operating the emulation platform. Fig. 3 shows the flow of the bootstrap program. By using this program, the overhead of operational software was greatly reduced.

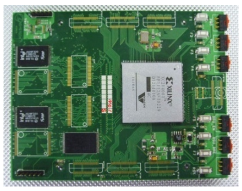


Fig. 2. The prototyping emulation platform.

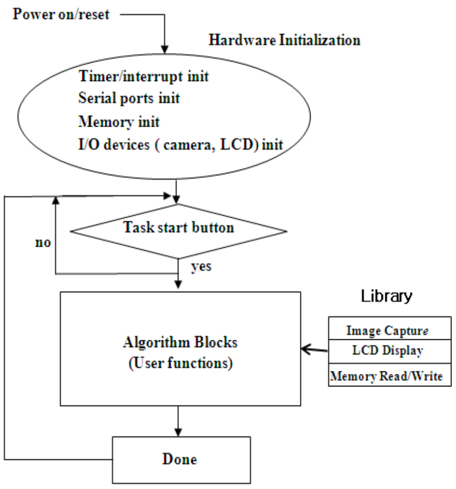


Fig. 3. The monitor program.

### 3.2 Pyramid reduction filter block

Our goal in implementing Gaussian pyramid processor is to create an architecture that is flexible enough to be implementable in hardware using FPGA [10, 11]. Fig. 4 shows the top level block description of the Gaussian pyramid processor. The pyramid reduction filer block is the heart of the Gaussian pyramid processor. In addition to the pyramid filer unit, additional control units, Direct Memory Access (DMA) unit, DMA FIFO buffers and working memories were implemented for the target Gaussian pyramid processor.

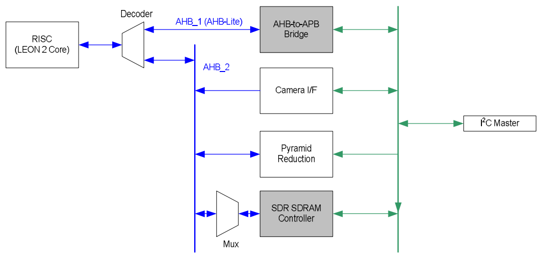


Fig. 4. The block diagram of the GPP.

Fig. 5 shows the register transfer level architecture and the data-flow path of the pyramid reduction filter block. To take advantage of the property of separation for the two-dimensional convolution, the pyramid reduction filter was implemented by two one-dimensional filters, horizontal and vertical filer. A 3-taps filter was used for convolution (low-pass filtering). Fig. 6 shows the internal architecture of the implemented pyramid reduction filter.

The source DMA block retrieves each of 16 pixels of stored image into input-FIFO- buffer using DMA. Using pixels in the input-FIFO-buffer, the pyramid reduction filter block performs first the horizontal convolution and decimation, and saves the result pixels into the line memory 'A'. Then the vertical convolution is performed with pixels in the line memory 'A' and saves the result into the line memory 'B'. Pixels in the line memory 'B" is finally decimated vertically and transferred into the output-FIFO - buffer. Final result is transmitted back to SRAM-based memory via DMA. The source

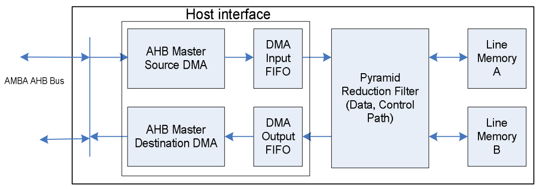
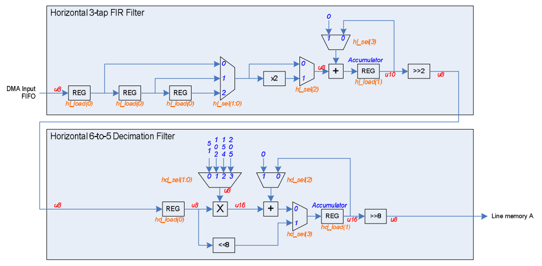
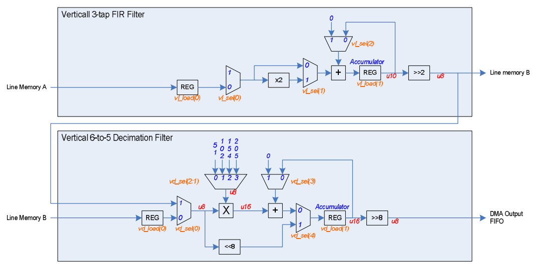


Fig. 5. Architectural overview of the processor.


Horizontal 3-tap FIR Filter
Horizontal 6-to-5 Decimation Filter
DMA Input FIFO
Accumulator
Line memory A

(a) Filtering and Decimation through horizontal direction



(b) Filtering and Decimation through vertical direction

Fig. 6. The pyramid reduction filter.

DMA block retrieves an input image from the external memory and stores it into DMA input FIFO. The block sends the signal to the reduction filter in order to start the task of the pyramid reduction block. The destination DMA block stores the generated image pyramid into the external memory when it receives the signal from the pyramid reduction filter. At each time, source/destination DMA block retrieves/saves 16 pixels.

## 4. SIMULATIONS AND EVALUATIONS

The implemented Gaussian pyramid processor were fully synthesized by VHDL model [12] and transferred into the FPGA (XILINX X2CV8000). The operational clock rate of the FPGA was 30 MHz. Image data were collected by external camera connected to USB interface of the emulation platform. An image captured by the camera was stored into the external SRAM-based memory. The Gaussian pyramid processor retrieves an image from SRAM-based memory using DMA. Then the Gaussian processor completes the task and save the image pyramid into the SRAM-based memory via DMA. Fig. 7 shows examples of the result stored in the SRAM after the processing of the Gaussian pyramid processor.

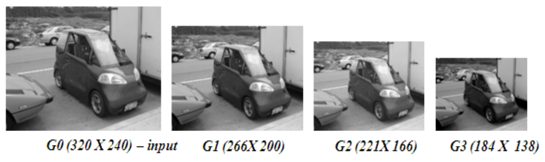


Fig. 7. The result of the Gaussian pyramid processor.

Two software systems (written by C language) implemented on PC (Intel Xeon CPU, Linux) and Sun Enterprise 450 (UltraSparc II, Unix) were used for comparison. In the case of Gaussian Pyramid Processor, the required times for constructing 13-level pyramid with a 320X240 image was 0.16 seconds. A comparison of performances of the Gaussian pyramid processor and other systems is shown in Table 1. As a conclusion, the implemented Gaussian pyramid processor can be used for real-time image processing applications that require a pyramid algorithm.

## 5. DISCUSSION AND CONCLUSIONS

In this paper, we designed and implemented the Gaussian pyramid processor architecture for various smart devices. The implemented Gaussian pyramid processor was tested and verified in the register transfer level architecture using FPGA-based emulation platform. From experimental re-

Table 1. Performances of three implemented systems

| Method | Clock | FPU | Cycles (Mega cycles) | Time (sec) |
|---|---|---|---|---|
| THe GPP | 30Mhz | No | 4.8 | 0.16 |
| Software(Intel Xeon) | 3.06Ghz | Yes | 21.5 | 0.021 |
| Software (UltraSparc II) | 400Mhz | Yes | 89.1 | 0.223 |

sults, we proved that this processor can be effectively used to various devices which needs SIFT.

## REFERENCE

[ 1 ] B. Blair and C. Murphy, *Difference of Gaussian Scale-Space Pyramids for SIFT Feature Detection*, Complex Digital Systems Design, Final report, 2007.

[ 2 ] P.J. Burt, "Fast Filter Transforms for Image Processing," *Journal of Computer Graphics and Image Processing*, Vol. 16, No. 1, pp. 20-51, 1981.

[ 3 ] J. Yang, X. Chen, and W. Kunz, "A PDA-based Face Recognition System," *Proceedings of Winter Application Computer Vision*, pp. 457-460, 2002.

[ 4 ] T. G. Link, N. Vijaykrishnan, M. J. Irwin, and W. Wolf, "Embedded Hardware Face Detection," *Proceedings of the 17th International Conferenceon VLSI Design*, pp. 1221-1232, 2004.

[ 5 ] O. Sims and J. Irvine, "An FPGA Implementation of Pattern-selective Pyramidal Image Fusion," *Proceedings of 2006 International Conference of Field Programmable Logic and Application*, pp. 345-349, 2006.

[ 6 ] N. Petterson and L. Petterson, "Online Stereo Calibration using FPGAs," *IEEE Proceedings of Intelligent Vehicles Symposium*, pp. 780-784, 2005.

[ 7 ] A. Darabiha, W.J. MacLean, and J. Rose, "Reconfigurable Hardware Implementation of a Phase-correlation Stereo Algorithm," *Machine Vision and Applications*, Vol. 17, No. 2, pp. 116-132, 2006.

[ 8 ] P.J. Burt and E.H. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Transactions on Communications*, Vol. 31, No. 6, pp. 532-540, 1983.

[ 9 ] Zhen-jun Du and Min Li, "SoC Verification Based on WGL," *Journal of Korea Multimedia society,* Vol. 9, No. 12, pp. 1607-1616, 2006

[10] P.G.D. Valle, D. Atienza, G. Paci, and F. Poletti, "Application of FPGA Emulation to SoC Floorplan and Packaging Exploration," *Proceeding of XXII Conference on Design of Circuits and Integrated System*, pp. 236-240, 2003.

[11] M. Brogatti, F. Lertora, B. Foret, and L. Cali, "A Reconfigurable System Featuring Dynamically Extensible Embedded Microprocessor, FPGA, and Customizable I/O," *IEEE Journal of Solid- State Circuits*, Vol. 38, No. 6, pp. 521-529, 2003.

[12] R. McCready, "Real-Time Face Detection on a Configurable Hardware System", *Proceedings of International Symposium on FPGA*, pp. 23-26, 2000.

Bongkyu Lee

He received the Ph.D degree from the Department of computer engineering, Seoul National University in 1995. Since 1996, he has been a professor at the Jeju national university. His interesting research fields are in Neural Networks, image processing and Augmented Reality.