Parallel-Processing VLSI Architecture for Intelligent Image Processing

(高度画像処理のための並列処理 **VLSI** アーキテクチャ)

Kiyoto ITO

Supervisor: Professor Tadashi SHIBATA

Department of Frontier Informatics,
The University of Tokyo

December 2006

Abstract

With the rapid progress in multimedia and information technology, real-time image processing becomes more and more important in various applications. In such applications, intelligent image processing which takes expensive computational cost has become essential even at a pixel level. Therefore, software solutions running on general-purpose microprocessors cannot attain real-time response capability even when the state-of-the-art VLSI technologies are employed. As a result, a new VLSI design paradigm is in demand in order to realize efficient execution of image processing for real-time applications.

The purpose of this thesis is to develop parallel-processing VLSI architectures for real-time intelligent image processing. In order to realize efficient execution of image processing on two-dimensional image data, a single-clock-cycle readout scheme for block-of-pixel data has been developed employing a *logic-in-memory* architecture. The concept has been extended to apply the CMOS image sensor, thus achieving the parallel readout from a two-dimensional pixel array. The *smart image sensor* architecture has also been presented employing time-domain technique, which allows us to build a compact pixel processing element having programmability.

Firstly, a VLSI image filtering processor capable of performing various kernel convolution processing in a single clock cycle has been presented based on the logic-in-memory architecture. In order to eliminate redundant memory accesses and complicated memory address control for partial image acquisition, pixel data are rearranged in the buffer memory according to the quaternary-tile pixel-mapping scheme developed in the present work. As a result, a fast and low-power operation has been achieved. In addition, the kernel size is variable up to 5×5 pixels, thus making the processor compatible to a number of advanced image-processing algorithms. The concept has been verified by a test chip fabricated in a 0.18-μm 5-metal CMOS technology. Without pipelining, the processor operates at 50MHz under a 1.8-V power supply, which outperforms the software processing running on a 2.2GHz MPU.

A computational digital-pixel-sensor (DPS) VLSI with on-chip image processing circuits has been developed. Block-readout architecture is employed in memory configuration at each pixel in order to resolve a interconnection bottleneck between a DPS array and processing units. The data from sensor array are read out in block and processed on the chip. As a result, this architecture enhances the performance of bit-serial digital signal processing, and pixel-parallel seamless scan of filtering operations has been successfully realized. A proof-of-concept chip including 64×48 DPS array and rank-order-filtering circuits has been designed and fabricated, and the concept has been verified by measurements.

For high-speed execution of motion-related image-processing algorithms, a time-domain gradient-detection architecture of analog motion sensors has been developed. In this architecture, a compact gradient detector circuit calculates temporal and spatial gradients of photodiode signals at each pixel location, converting the gradients to pulse widths, i.e. to time-domain signals. The time-domain signals are then converted to digital format using on-chip binary counters, thus allowing us to carry out various algorithms by external digital imaging systems. A proof-of-concept chip of 31×31 pixels has been designed and fabricated using 0.35-μm CMOS technology, and the optical flow estimation at the rate of 400 frames/sec with 3.3V supply has been experimentally demonstrated.

A mixed-signal focal-plane image processor for real-time spatiotemporal convolution has been developed based on time-domain computation technique. This allows us to build a compact pixel processing element having programmability. The concept was verified by the first version of prototype chips fabricated in a 0.18-μm CMOS technology, demonstrating over 78,000 convolutions/s with 1.0V supply. In addition, a enhanced version of the focal-plane image processor is also designed, where digital arithmetic operations are realized by simple logic circuits based on the time-domain technique.

The architectural concepts developed in this work are general and applicable to other image processing VLSI's. Since block-readout access scheme for two-dimensional pixel data is essential in a number of image processing algorithms, the single-clock-cycle access method developed in this work can significantly improved the *latency* of the system. Also, the proposed smart image sensor architectures enable us to realize more compact and flexible configuration in designing processing elements. Therefore, merging these two architectures would make contributions to enhance performances or efficiency in various intelligent image processing systems.

Acknowledgments

With the utmost gratitude I would like to thank my advisor Professor Tadashi Shibata for his loyal support and enduring guidance over many years. His enthusiasm for teaching and research, his constant support and fruitful discussions on my research led me to become a full-fledged member of society and brought my research to success. I feel very fortunate to have taken him as my supervisor, and the precious experiences in my study days will be irreplaceable assets in my life.

I would like to acknowledge my dissertation committee: Prof. Kunihiro Asada, Prof. Shin-ichi Takagi, Prof. Kiyoharu Aizawa, Prof. Makoto Ikeda for their preview of the thesis and their extremely valuable comments, discussions to my research. Their constructive suggestions was indispensable for making my dissertation study successful.

I would like to thank Prof. Yoshio Mita for meaningful discussion on my research and for his motivating engineering lessons. His insightful teaching has greatly expanded my interest in and enriched my knowledge of MEMS.

I express my appreciation to Mr. Toru Murai for his advice; Ms. Tomi Takei, Ms. Michiyo Abe, Ms. Miei Komiyama, Ms. Kimiko Mori, and Ms. Motoko Inagaki for their supports.

I am very thankful to my colleagues and friends; Dr. Masakazu Yagi, Dr. Gu Qian Rong, Dr. Toshihiko Yamasaki, Dr. Makoto Ogawa, and Dr. Daisuke Kobayashi for their invaluable support and advice; Teruyasu Taguchi, Hideo Yamasaki, Yasufumi Suzuki, and Hitoshi Hayakawa for being dependable and supportive; Yusuke Nakashita, Tomoyuki Nakayama, Tomohiro Fukuda, Benjamas Tongprasit, Kunio Kawahara, Satoshi Kametani, and Ken Takata for their input and enthusiasm. Portions of this dissertation have been made possible with their assistance and backing.

Finally, I am immensely grateful to my parents, my elder sister, and my grandparents for their very prudent unconditional support.

This work was supported in part by the Ministry of Education, Science, Sports, and Culture under Grant-in-Aid for Scientific Research #17206030. The VLSI chips in this study have been fabricated in the chip fabrication program of VDEC, the University of Tokyo with the collaboration by Rohm Corp. and Toppan Printing Corp. and by Hitachi Ltd. and Dai Nippon Printing Corp., respectively.

Kiyoto Ito
The University of Tokyo

Contents

List of Figures

List of Tables

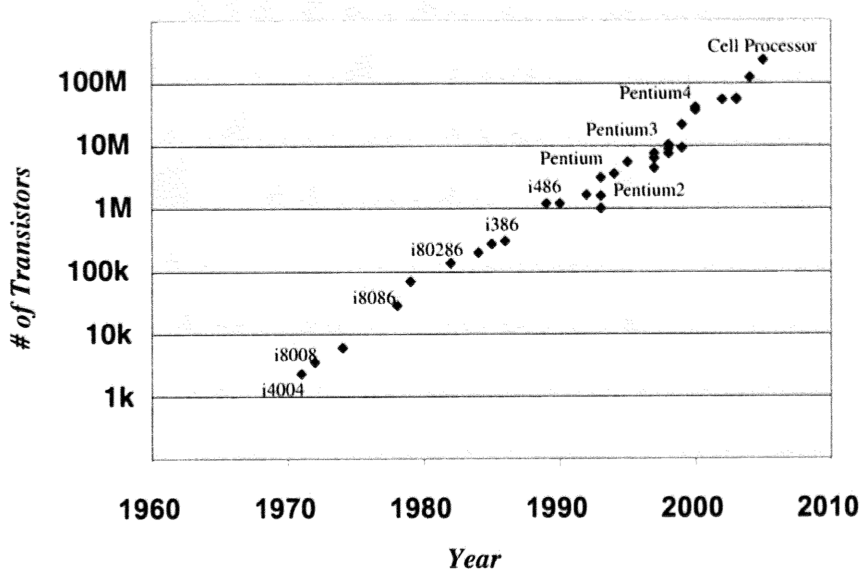
Chapter 1

Introduction

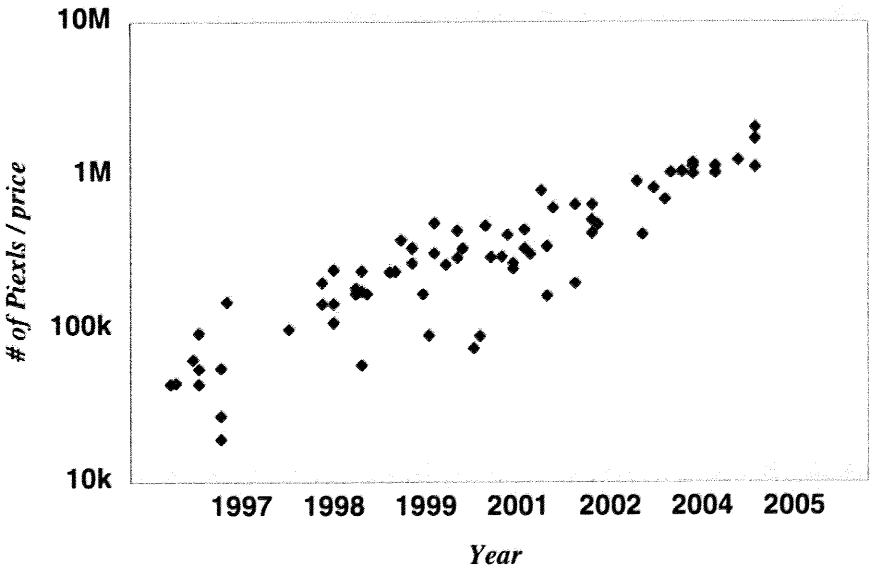
1.1 Background

In the era of multimedia and information, a huge amount of visual information is available in our lives. Therefore, real-time image processing becomes more important in various applications. However, image data in the real world are massive in quantity, and typical image processing is computationally very expensive. Especially, in applications such as image recognition, robotics control, security surveillance systems, etc, advanced image-processing algorithms are introduced, where intelligent image processing taking expensive computational cost is utilized even at a pixel level. In those cases, the processing would be intolerably time-consuming for the state-of-the-art computer technologies. As a result, software solutions running on general-purpose microprocessors cannot attain real-time response capability except for specific purposes.

It is important that this performance insufficiency of general-purpose microprocessors would not be improved easily. Fig. 1.1 summarizes the advances in silicon technology. As shown in Fig. 1.1(a), the number of transistors integrated into one VLSI chip has been increasing exponentially as predicted by well-known Moore's Law [1]. On the other hand, Fig. 1.1(b) displays the number of pixels per price in commercial digital still camera, which indicates image acquisition technology of silicon devices is also growing exponentially according to the Moore's Law. Namely,



(a)



(b)

Figure 1.1: (a) Number of transistors in microprocessors. In the figure, generations of typical MPU's are shown as examples. (b) Number of Pixels per price in commercial digital still cameras.

both processing units and image acquisition devices have been growing in the same rate. While the performance of microprocessors has been improved, the performance of the imaging system is not changed because the amount of visual information obtained by image acquisition devices has also been increasing. Namely, the performance requirement would not be realized by the classical concept of the scaling law. As a result, new VLSI design paradigm is in demand in order to realize efficient execution of image processing.

1.2 Typical Image Processing System

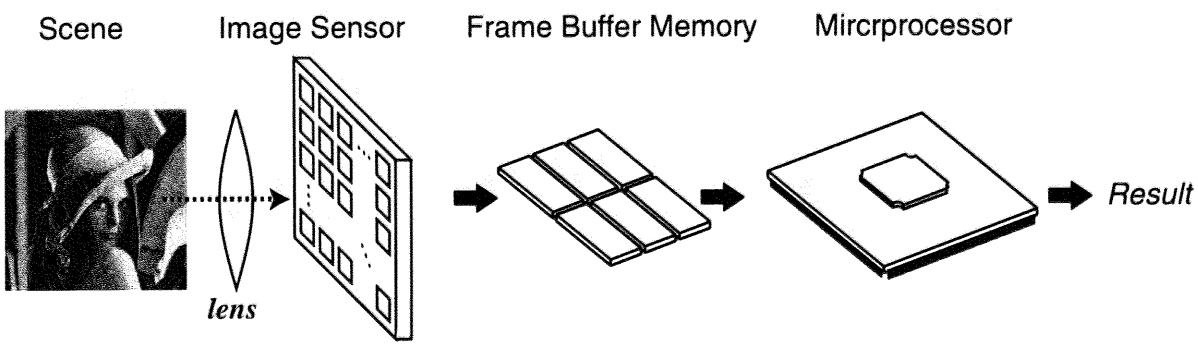


Figure 1.2: Configuration of typical image processing system.

Fig. 1.2 depicts a simplified block diagram of typical imaging-system architecture, which consists of an image sensor, a frame buffer memory, and a microprocessor. First, the scene is focused on the image sensor using the imaging optics such as a lens system. The image sensor is composed of a two-dimensional array of pixels, and it converts the light intensity at its surface into electrical signals of pixel data. The analog pixel data (i.e., the electrical signals) are read out from the image sensor and stored into the frame buffer memory. In this stage, most of imaging systems introduce analog-to-digital conversion (ADC), thus storing the pixel data as a digitized value. Finally, the microprocessor downloads the pixel data from the frame buffer and produces the result of processing.

Each component of the imaging system plays a role in determining its overall performance.

However, a major bottleneck of the system is data transmission between every component. In order to accomplish the whole processing, pixel data obtained by the image sensor must be transfered to the microprocessor via frame buffer memory. This process often requires a huge number of clock cycles. A popular approach to improve the performance is a pipeline technique. Namely, whole processing is divided into a series of small operations, and each operation is executed in a time-sliced fashion. This approach is efficient in such applications as video-compression because it increases the *throughput* of the system in processing data streams. However, the *latency* through the pipeline is typically several frames, thus resulting that such a system is unsuitable for many time-critical applications where the system must be react in real time against the processing result.

1.3 Parallel Processing VLSI Architecture

1.3.1 Overview

In order to improve the *latency* of the system, introducing a parallel processing architecture is efficient. In image processing, especially in image processing at an early stage, identical operations are performed at each pixel site. Therefore, implementing pixel parallel processing on VLSI chips is quite natural approach to improve the performance. Although a number of parallel architectures have been developed aiming at efficient execution of image processing, the emphasis of the thesis is focused on a *single-instruction and multiple-data* (SIMD) architecture. It is widely employed in various VLSI's thanks to its high parallelism in the processing and its simple structure. In the SIMD architecture, a number of processing elements (PE's) are laid out as an array. All PE's simultaneously execute the operation upon pixel data according to a broadcasted instruction, thus realizing parallel processing. For further high parallelism in the SIMD architecture, merging several components into a single VLSI is effective. One of those architectures is a *logic-in-memory* architecture [2], which integrates a number of PE's into frame buffer memories, thus taking the advantage of inherently high memory bandwidth between the PE's and memories. Another approach is a *smart image sensor* [3], where a PE is implemented with a photodetector at each pixel site, thus achieving massively parallel operation on a focal plane.

Logic-in-Memory

The idea of merging logic and memory on a single chip was proposed as a correction strategy for the wide performance gap between the microprocessor and its main memory [2,4,5]. The primary objective of these *logic-in-memory* systems is to utilize the high data bandwidth and inherent parallelism that is available inside the memory module. This not only improves system performance, but also improves power consumption and in some cases reduces system cost. There exists some archetypes of the logic-in-memory SIMD processor designs in literatures; a linear processor array (LPA) [6–8], a square processor array (SPA) [9–11], pyramid structure [12], and hypercube structure [13].

Smart Image Sensor

A smart image sensor integrates image sensing device and processing circuitry into a single chip, which is also refereed to as a computational image sensor, a functional image sensor, or a vision chip. The smart-image-sensor architecture offers massively parallel operation on focal plane. However, the boundary to design processing element is strictly limited because a large pixel element reduces the resolution of obtained images. Therefore, more compact design for each pixel circuitry is important in developing the smart image sensors.

Since analog circuitry makes possible to implement much simpler and power-efficient configuration, integration of photodetecting elements and analog processing circuits into one pixel has been discussed from late 1980's [3,14–16]. This configuration is often refereed to as "silicon retina," and has inspired many engineers to develop retinomorphic vision systems that also imitate these parallel processing capabilities [17–26]. On the other hand, Ishikawa et. al. have proposed a digital SIMD vision chip aiming at high-speed object tracking [27–30], in which 1b general purpose ALU is integrated into each pixel.

1.3.2 Challenges in Designing Parallel Processing VLSI

Missing One Dimension

One of the challenges to develop a parallel architecture is a configuration of the PE array. Visual information obtained from the real world is two-dimensional data. However, when we carry out some operation on the 2D information, one more dimension is necessary. Namely, there is another direction for the data processing as shown in Fig. 1.3. Therefore, it can be said that image processing requires a three-dimensional structure for processing, while only two dimensions are allowed for VLSI designers. This one missing dimension is crucial [7, 31, 32].

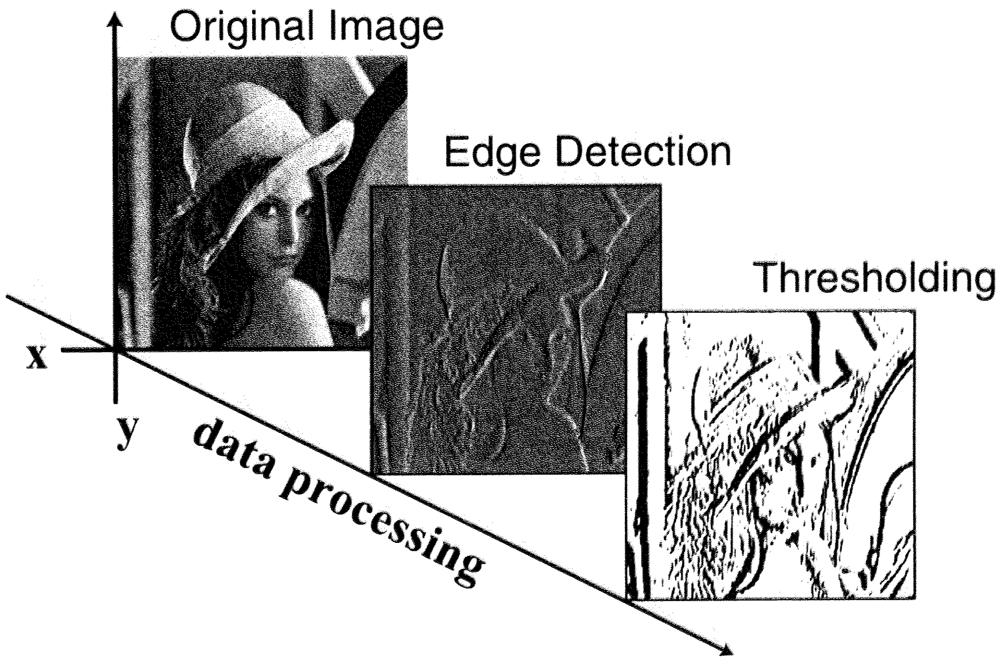


Figure 1.3: Three-dimensional structure of image processing.

In a VLSI chip, we need to consume at least one dimension to implement processing elements (PE's). When we lay out PE's in a line, the direction for data processing is available in a vertical direction against the PE stripe. Therefore, many functions can be flexibly integrated by stacking PE stripes along the direction. However, the image processing is carried out line-by-line with scanning each row across the target 2D image data. As a result, the number of clock cycle is required until the processing is finished. On the other hand, if we try to make a high-density 2D array of PE's on a die, the functionality of a PE must be traded off with the total number of PE's on the chip. In addition, the image processing is limited among only nearest neighbor pixels due to the complexity of interconnection. If the pixel area required for the processing is larger than 3×3, the circuit configuration would be prohibitively complex, which is known as "interconnect explosion" [33]. In this manner, The "missing one dimension" in silicon technology is a real obstacle in building parallel image-processing VLSI.

Boundary between Analog and Digital

Where to place the boundary between analog processing and digital processing is another issue in designing parallel processing VLSI, especially in designing smart image sensors [34]. Analog implementations offer a compact and power-efficient configuration of the PE. Therefore, a number of PE's can be implemented in a single chip, thus making possible to realize massively parallel processing on every pixel. Integration an image sensing device with a processing element is another advantage of analog implementation. However, the drawback of the analog circuitry is that it is difficult to change functions in hardwired analog circuits, thus the target application were limited for only specific purposes.

On the other hand, digital implementations offer a large flexibility and programmability in accommodating the system to various algorithms. One of the drawbacks in digital implementation is a volume of the circuitry. Namely, a scale of the PE becomes relatively large, thus reducing the parallelism of the processing. In addition, a large scale circuit often increases total power consumptions.

1.4 Research Objectives and Thesis Organization

A scope of the thesis is the study of parallel processing VLSI architecture for intelligent image processing. In order to solve the "missing-one-dimension" issue, a new logic-in-memory architecture has been presented and the concept has been extended to apply the smart image sensor architecture. Then, new smart image sensor architectures has been developed to realize the compactness of analog circuits and the flexibility of digital circuits.

In this work, spatio-temporal image filtering is focused because it is the most fundamental function in image processing, and it is essential for various intelligent image processing algorithms. For instance, the spatial convolution filtering plays an important role in edge detection, feature extraction, and so forth. For another example, the result of temporal convolution, i.e. intensity derivatives at each pixel, is an important information in motion-related algorithms. Although operations performed at each pixel are simple, image filtering is computationally very expensive because it requires a huge number of pixel-level operations. Therefore, it is important in various intelligent image processing to develop a not only high-speed but also low-power VLSI architecture for image filtering.

The organization of the thesis is as in the following. In Chapter 2, a VLSI image filtering processor capable of performing various kernel convolution processing in a single clock cycle is presented. Chapter 3 proposes a new architecture of a computational digital-pixel-sensor chip which allows the parallel readout of block-of-pixel data for image filtering. In Chapter 4, an architecture to realize high-speed spatiotemporal edge detection for analog motion sensor chips is presented. In Chapter 5, a mixed-signal focal-plane image processor for real-time spatiotemporal convolution has been developed based on time-domain computation technique. Finally, Chapter 6 concludes the thesis.

In Appendix A, a time-domain winner-take-all circuit is presented as another experience of the architecture employing time-domain technique.

Chapter 2

Variable-Kernel Flash-Convolution Image Filtering Processor

2.1 Introduction

Image filtering is a fundamental operation in image-processing algorithms, playing essential roles in primary image processing such as noise reduction, edge enhancement, feature extraction, and so forth. In image filtering, an image window of a certain size is defined as a kernel for the processing. In the kernel, weight values are assigned to every pixel locations, and the convolution is taken between the weight matrix and the pixel data within the kernel window. The result is relocated as the pixel data in the filtered image. The kernel processing is conducted at each pixel site in an input image and is repeated pixel-by-pixel by scanning the entire image with the kernel window. Therefore, the processing is computationally very expensive, in particular as the kernel size increases. However, in certain advanced image-processing algorithms, like that in image recognition using directional edge-based feature representations [35, 36], a kernel size as large as 5×5 pixels is extensively used. A very fast execution of kernel processing with varying kernel sizes is in demand in real-time image processing applications.

The problem of using typical MPU's for image filtering processing is that it requires a number of redundant memory accesses and complicated memory address control for partial image data acquisition to carry out the convolution operation. The physical addresses of pixel data in the memory module must be calculated each time in the processing and the same pixel data are downloaded to the MPU a number of times at different kernel locations. Therefore, the processing is not only time consuming but also power hungry.

A number of hardware accelerators have been developed aiming at efficient execution of image processing. Among those, a *single-instruction and multiple-data* (SIMD) architecture is employed in order to exploit its high parallelism in the processing. Moreover, VLSI chips that integrate SIMD processing elements (PE's) into memory modules take the advantage of inherently high memory bandwidth between the PE's and memories, thus achieving high-speed and low-power execution of image processing. Such design scheme is sometimes referred to as the *logic-in-memory* architecture [2, 37].

Although there exists some archetypes in the logic-in-memory SIMD processor designs [7, 38, 39], two types are mainly employed in image processing: a linear processor array (LPA) and a square processor array (SPA). Due to the simplicity in layouts, the LPA architecture [6, 8, 39] is widely employed to accelerate various parallel algorithms. However, the linear-array structure is not compatible to the two-dimensional nature of image data placement, and several clock cycles are required to readout all pixel data necessary for the kernel convolution from memory modules. On the other hand, SPA processors are composed as two-dimensional arrays of PE's (processing elements) compatible to the image data arrangement, thus enabling massively pixel-parallel operation. Therefore, SPA processors have been extensively discussed during the past few decades [9–11]. One of the problems in the SPA approach is its complexity in the configuration. PE's are allocated in a shape of two-dimensional array, reflecting the physical location of pixel data. In order to achieve a single-clock-cycle convolution in this configuration, each PE must be connected to other PE's by multiple interconnects for data acquisition. Such interconnection is possible if the data transmission is limited only within the nearest neighbors. If the data transmission among nearest neighbors and next nearest neighbors are required for kernel sizes larger than 3×3, the interconnects structure would be prohibitively complex, which is known as "interconnect explosion" [33]. Therefore, a number of clock cycles are required to cumulate all necessary pixel data for the kernel processing at each pixel site. In addition, it is power consuming as well.

The purpose of this chapter is to present a VLSI image filtering processor which is capable of conducting any basic kernel-convolution within a single clock cycle for varying kernel sizes. A quaternary tile mapping (QTM) scheme has been developed in order to eliminate a large number of redundant memory accesses and complicated memory address control. In addition, radix-2 signed digit data representation is employed in the global data distribution for SIMD control in order to reduce not only the wiring area but also the power dissipations. The concept has been verified by a test chip fabricated in a 0.18-μm 5-metal CMOS technology. Without pipelining, the processor operates at 50MHz under a 1.8-V power supply, which outperforms the software processing running on a 2.2GHz MPU.

This chapter is organized as follows. In Section 2.2, the system architecture and the QTM (quaternary tile mapping) scheme is described. The structure of the major building blocks are also given in Section 2.2. The prototype chip implementation and the experimental results are presented in Section 2.3. Finally, the summary of this chapter is given in Section 2.4.

2.2 System Organization

2.2.1 Convolution Operation

Convolution operation is defined as in the following:

$$C(x,y) = \sum_{i,j\in K} W_{i,j} I(x+i, y+j) \tag{2.1}$$

where $I(x,y)$ is the pixel intensity at the location (x,y), $W_{i,j}$ the weight value of the kernel K, and

$C(x,y)$ the result of convolution at (x,y). In the present work, pixel values are represented by 8-bit binary numbers (8 bit-per-pixel(bpp)), and the convolution results $C(x,y)$'s are represented by two-byte (16-bit) numbers in order to guarantee the computation accuracy. The weight values in the kernel are given in three digits: +1,-1 and 0. Therefore, the convolution of Eq. 2.1 reduces to the summation of positive numbers and negative numbers as in the following:

$$C(x,y) = \sum_{W_{i,j}=1} I(x+i,y+j) - \sum_{W_{i,j}=-1} I(x+i,y+j). \tag{2.2}$$

If negative numbers are represented by 2's complement, a single accumulator can carry out the calculation. However, each number must be represented with the maximum bit length of 16b, resulting in an increased volume of the accumulator circuitry.

In order to reduce the size of the accumulator, the addition of multiple numbers and the sign maintenance are separated. At first, the summation is carried out using 8b numbers where negative numbers are represented by 1's complement. In this case, sign bit extension from a 8-b number to a 16-b number goes as follows:

$$\begin{aligned} \bar{\bar{I}}_{(16)} &= \bar{\bar{I}}_{(8)} + 2^{16} - 2^{8} \\ &= \bar{I}_{(8)} + 2^{16} - 2^{8} + 1 \end{aligned} \tag{2.3}$$

where $\bar{\bar{I}}$ and $\bar{I}$ are the 2's complement representation of I and the 1's compliment of I, respectively. The small numbers in the parentheses at the bottom right of I indicate the bit precision in the binary representation. Therefore, substituting (2.3) into (2.2) yields

$$\begin{aligned} C_{(16)}(x,y) = & \sum_{W_{i,j}=1} I_{(8)}(x+i,y+j) \\ & + \sum_{W_{i,j}=-1} \bar{I}_{(8)}(x+i,y+j) \\ & - N(2^{8}-1). \end{aligned} \tag{2.4}$$

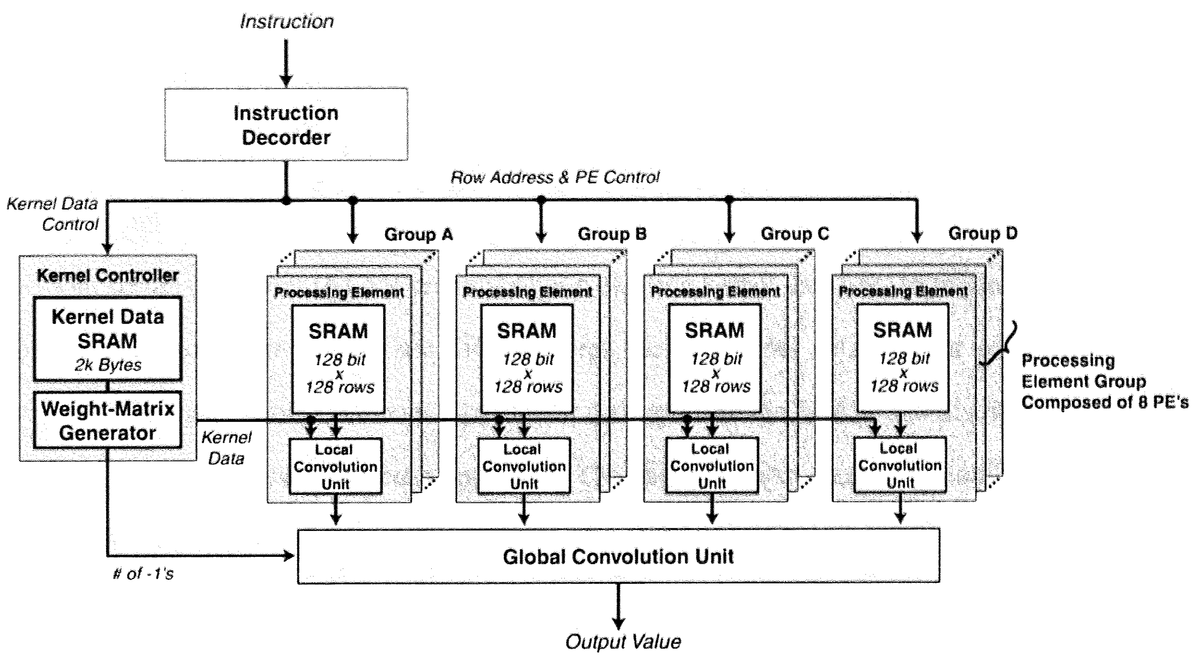


Figure 2.1: Simplified block diagram of image filtering processor.

where $\bar{I}(x,y)$ represents 1's complement of $I(x,y)$, and N is the number of -1's in the convolution kernel and N times 2^{16} terms disappears due to the overflow. N times $(2^8 - 1)$ is the *sign adjustment term*, which converts the accumulation result of 1's complement numbers to the corresponding signed 2's compliment representation. As shown in the equation, the convolution operation is accomplished by multiple-data addition of 8bit numbers and bit-inverted numbers (1's complements), followed by the subtraction of the sign adjustment term. In this manner, the accumulation of positive and negative numbers is conducted using a minimal number of adder units.

2.2.2 System Architecture

Fig. 2.1 shows a simplified block diagram of the processor. It consists of an instruction decoder, four groups of eight processing elements (PE's), a kernel controller (KC) and a global convolution unit.

Each PE is composed of a 2k Bytes (128b$\times$128 rows) SRAM module and a local convolution unit (LCU). Image data for processing are distributed in the SRAM modules based on a quaternary tile mapping scheme as described in the next subsection. The LCU is placed in the close vicinity of SRAM sense amplifiers similar to the organization of *logic-in-memory* architecture [2, 37], thus the LCU can obtain all the data in one row of each SRAM modules in a single readout operation.

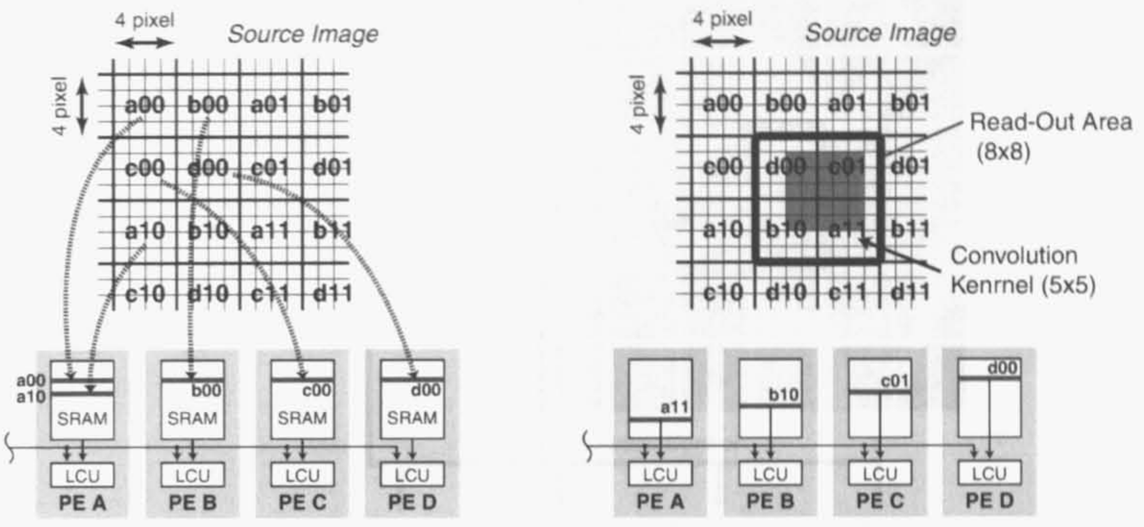
The kernel controller (KC) is comprised of a 2k-Byte kernel data SRAM module and a weight-matrix generator. The kernel data SRAM stores various kernel data sets for convolution. The weight-matrix generator downloads a particular data set from the SRAM and distributes the weight values to LCU's necessary for each operation. The generator also counts the number of -1's in the kernel data set, and transfer the information to the global convolution unit for the sign adjustment operation in Eq. 2.4.

One of eight PE's in each group is selectively activated to carry out a convolution operation, and therefore, four PE's work simultaneously. Finally, the global convolution unit accumulates the four outputs from the activated PE's and the number of -1's from the kernel controller according to Eq. 2.4, thus yielding the convolution result as an output.

2.2.3 Quaternary Tile Mapping Scheme

The quaternary tile mapping (QTM) scheme essential for efficient filtering operation is illustrated in Fig. 2.2. In the processor, as explained above, one of the eight PE's in each group is selectively activated and four PE's are always working simultaneously, which are denoted as PE A$\sim$D in the figure.

Fig. 2.2(a) illustrates the image data placement in the QTM scheme. All the pixel data in the source image is divided into 4×4-pixel areas. The data in each pixel area is stored in one row in the SRAM of a PE in such a way that adjacent areas are assigned to different PE groups, e.g., the area named "a00" is stored into the row "a00" of the PE in the group A (PE A), and the area "b00" into the row "b00" of the PE in the group B (PE B), etc. The number of memory cells in a single row of the SRAM (128b) is matched to the number of data bits in a 4×4-pixel area ($4\times4\times$8bpp), thus



(a) (b)

Figure 2.2: Quaternary tile mapping scheme: (a) Image data distribution to SRAM rows; (b) Data access scheme during convolution.

each row in the SRAM module stores the data in a particular pixel area.

In carrying out a convolution operation, the data in the four pixel areas are simultaneously accessed, as shown in Fig. 2.2(b). Namely, the data of the 8×8-pixel area are downloaded to the LCU's at one time. For kernel sizes up to 5×5, all the pixel data necessary for the kernel processing are available wherever the kernel is located within the 8×8 area. Also, the downloaded pixel data are temporarily memorized in the registers of the LCU's. Therefore, as long as the convolution kernel stays in the same 8×8 area, the data in the registers of the LCU's are re-used and it is not necessary to change the memory address and download a new set of pixel data each time the kernel location is shifted. Thus, the power dissipation due to redundant memory accesses has been eliminated.

Such a tile mapping scheme was first introduced in the multimedia processor [40] in order to efficiently reuse the pixel data in the motion compensation operation. The pixel data were divided into 8×8-pixel areas and stored in nine DRAM banks separately. The data in four 8×8-pixel areas are downloaded to the processor core via data bus from four separated DRAM banks and are used

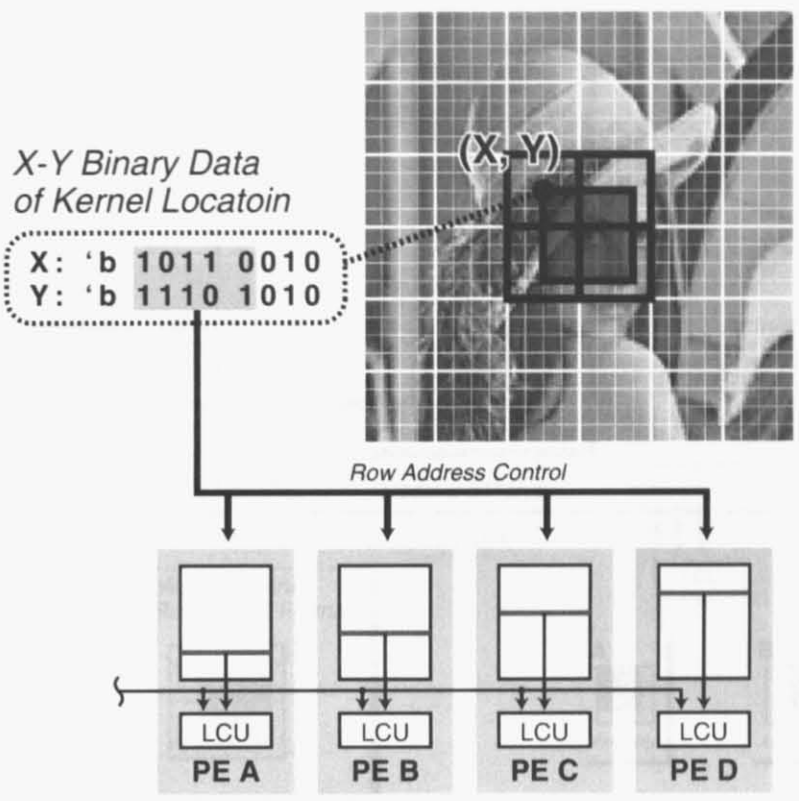


Figure 2.3: Row address control employing QTM scheme.

for motion vector generation. When the matching search area is shifted to a neighboring location, the data in two pixel areas can be reused, thus reducing the number of accesses to memory banks.

In the present work, on the other hand, every memory banks are directly connected to their respective processing units and a single memory access allows us to carry out the filtering operation for all locations in a 5×5-pixel area. As a result, it has become possible to carry out seamless scanning of 5×5-pixel kernel window over the entire target image with a minimal number of memory accesses.

The address control of the kernel location in the QTM scheme is illustrated in Fig. 2.3. When the X and Y address of the top-left corner of the kernel window are represented by 8b codes, their upper five bits do not change as long as the kernel is staying in the same 8×8 area. Therefore, the upper five bits in the 8b X and Y address codes are utilized to specify the row address of the SRAM

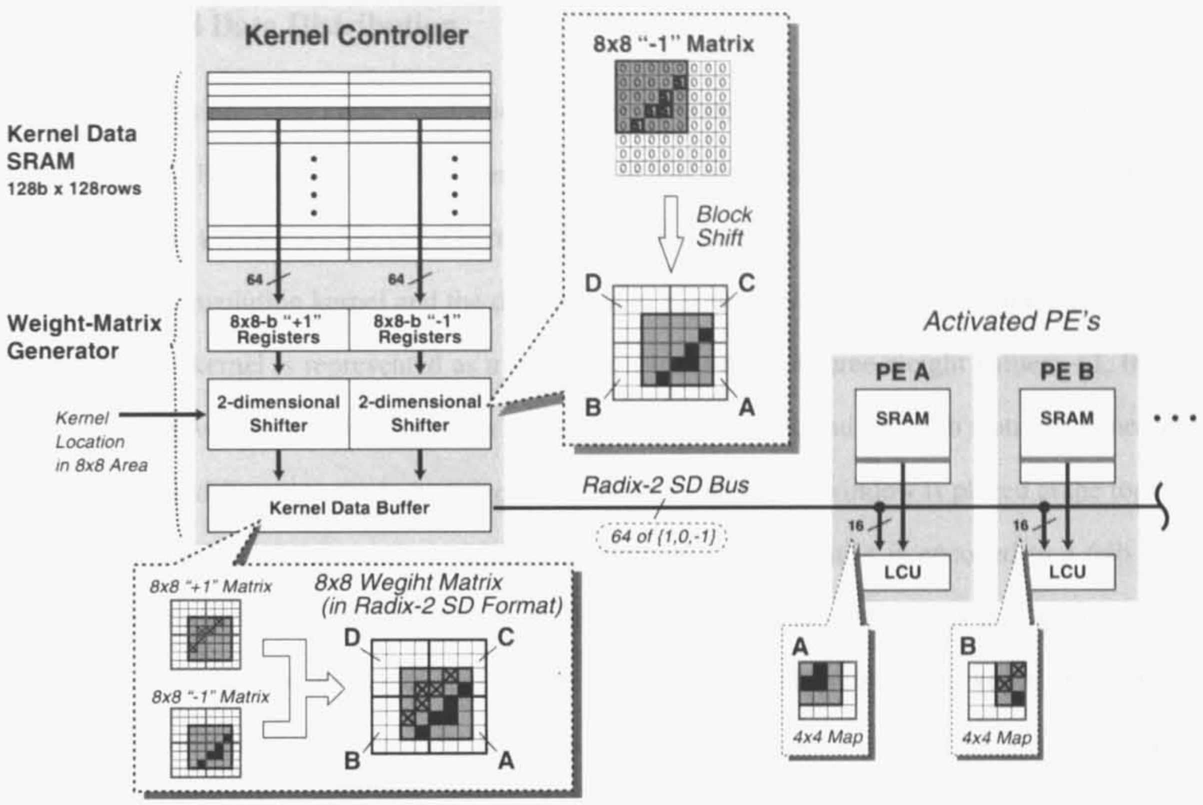


Figure 2.4: Kernel data operation in kernel controller.

in each PE for the kernel processing within the 8×8 area.

In this implementation, the maximum kernel size is determined as 5×5 pixels in order to make it compatible to the image recognition algorithm proposed in [36]. When larger kernel sizes are required, it is only necessary to increase the number of memory cells in a single row of the SRAM so that it fits the data size in a single area of pixels. For example, the number of memory cells in a row is 512 (8×8×8bpp) when the maximum kernel size is 9×9. Namely, the number of bits in a row of the SRAM is determined so as to store all the data in the pixel area the size of which is one pixel smaller than the maximum kernel size.

2.2.4 Kernel Data Distribution

The processing in the kernel controller (KC) is illustrated in Fig. 2.4. The KC is composed of two parts: a 2k-Byte (128b$\times$128rows) kernel data SRAM module and a weight-matrix generator. In each row of the kernel data SRAM, a kernel data set for a particular filter is stored. The relationship between the convolution kernel and the data stored in a row of the SRAM is explained in Fig. 2.5. A convolution kernel is represented as a two-dimensional map of three weight values: +1, 0, and -1. Then, positive and negative digits are separated into two maps, and a 8×8-b matrix is generated for each map indifferent to the kernel size. In the matrix, the kernel window is placed at the top-left corner, and the rest of the matrix is filled with digit "0". Each matrix is encoded to a 64b code by concatenating every rows of data as shown in the figure. The two 64b codes thus generated are stored in a single row of the kernel data SRAM. In this manner, each row of the kernel data SRAM stores the information of the convolution kernel. Then, the weight matrix for the kernel-scanning operation is generated as in the following.

The weight-matrix generator consists of two sets of 8×8-b registers, two-dimensional shifters which are composed as switching matrixes, and a kernel data buffer (See Fig. 2.4). Before image filtering operation, a particular kernel data set is downloaded to the two 8×8-b registers, each of which stores separately the positive and negative digit matrixes. Then, each matrix is block shifted in both x and y directions according to the kernel window movement in the the 8×8-pixel area. The kernel data buffer combines two matrixes as a single 8×8 weight matrix, where three types of digits (+1, 0 and -1) are encoded into the radix-2 signed digit representation described in the next subsection. Finally, the weight matrix is divided into four 4×4 small maps, being distributed to respective LCU's by the kernel data buffers.

2.2.5 Radix-2 Signed Digit Representation

In order to reduce the amount of interconnects for distributing the kernel data from KC to LCU's, a radix-2 signed digit (SD) representation is employed and the three digits (+1, 0, -1) are transferred

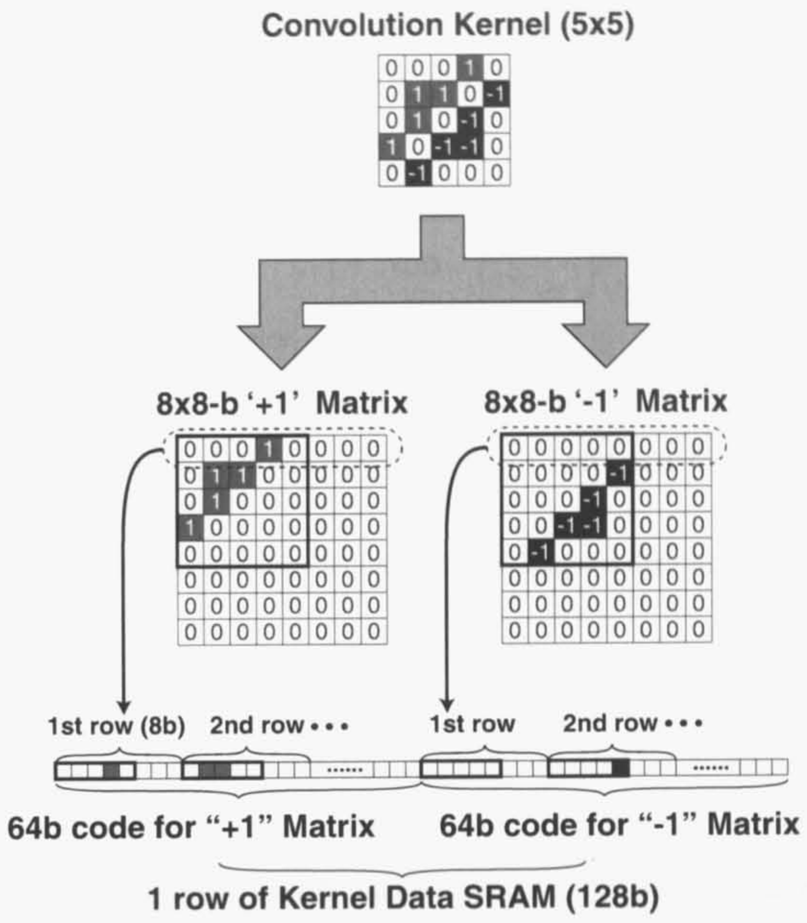


Figure 2.5: Row data configuration of kernel data SRAM

via single bus line. In addition to V_{DD} and V_{SS} assigned to "1" and "0", respectively, $V_{DD}/2$ is assigned to "-1" on the bus. The encoder and decoder units for the radix-2 SD bus are shown in the Fig. 2.6. The encoder unit consists of a standard CMOS tri-state buffer for V_{DD}/V_{SS} transmission and a $V_{DD}/2$-generator composed as a switched source follower. When *in_minus* is "0", the tri-state buffer is activated and the $V_{DD}/2$-generator is switched off. Therefore, the encoder transfers the signal equal to the *in_plus*. When *in_minus* is "1" and *in_plus* is "0", the tri-state buffer is disabled and $V_{DD}/2$-generator is switched on. Then, the output voltage of the encoder is determined by the resistance ratio between the driver transistor and the load transistor in the $V_{DD}/2$ generator, which is roughly $V_{DD}/2$ when a proper width/length ratio of the transistors is employed. On the other hand,

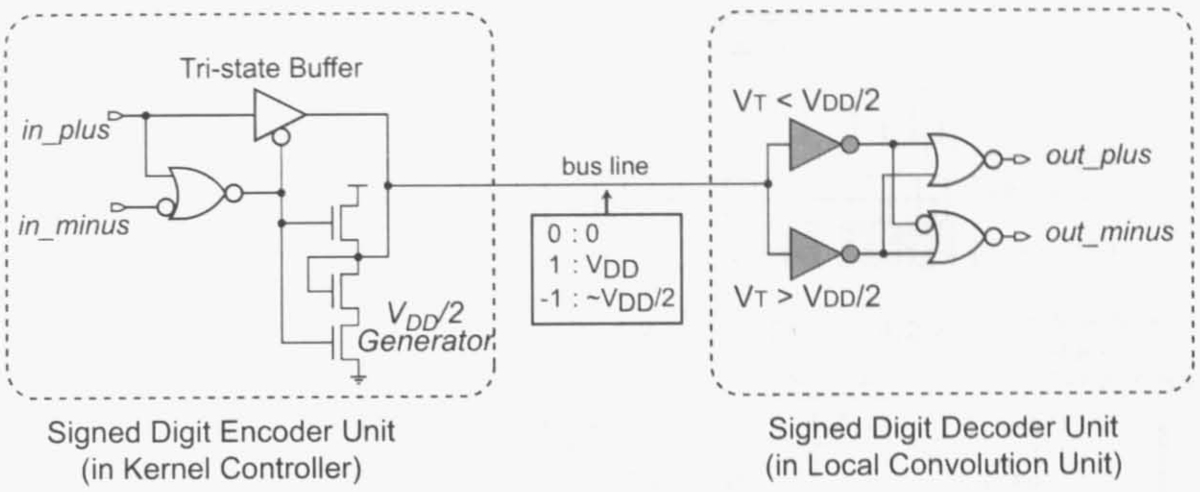


Figure 2.6: Encoder and decoder unit of radix-2 signed digit data transfer.

the decoder in the LCU is composed of two inverters having different logical threshold voltages: one larger than $V_{DD}/2$ and the other smaller than $V_{DD}/2$. Finally, decoded results *out_plus* and *out_minus* are provided by simple standard logic circuitry composed of two NOR gates and an inverter.

Since the $V_{DD}/2$-generator in the encoder unit is composed as a switched source follower, a DC current flows when "-1" is transmitted. However, it is not critical in terms of the power dissipation in the bus drive because the number of -1's in the kernel is only a fraction of 8×8 matrix. In addition, the dynamic power dissipation for bus state transition is more dominant than the static power dissipation due to the shortcut currents as the operation frequency increases. In this manner, the problem of increased chip real-estate and power dissipation due to multiple bus lines necessary for parallel processing has been alleviated.

2.2.6 Local Convolution Unit

Fig. 2.7 shows the block diagram of the local convolution unit (LCU). It consists of pixel registers, filtering units, and a 16-input carry save adder. In carrying out a convolution operation, the LCU receives 128 bit data ($4\times4\times8$bpp) from the SRAM module, and stores the data in the pixel registers. The filtering unit is composed of the radix-2 SD decoder (Fig. 2.6), three NAND

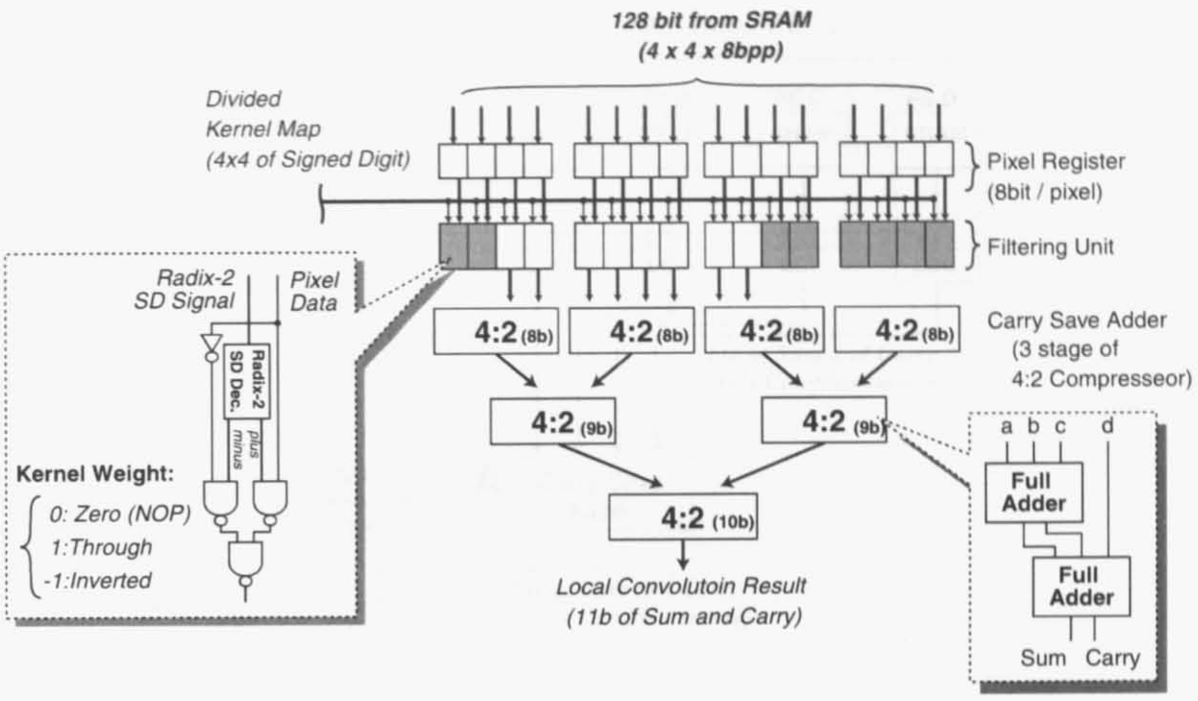


Figure 2.7: Local convolution unit.

gates and an inverter. It manipulates each pixel data according to the weight value received from the KC. The pixel data manipulation is carried out as in the following. When the weight is "1", the pixel data are passed through as they are, while they are bit-inverted as 1's complementary negative number when the weight is "-1". The pixel data are converted to zero when the weight is "0". Then, the all processed pixel data are accumulated by the carry save adder which consists of three stages of 4:2 compressors. Each 4:2 compressor is composed of two full adders, and the number in the parenthesis indicates bit-length of the data handled at each stage. As shown in the Eq. 2.4, the carry save adder conducts only the summation of positive numbers and 1's compliment negative numbers. Therefore, the output of the carry save adder is a set of two 11b numbers, the sum and the carry, which are the summation results of 4×4 8b numbers. Finally, the output is transferred to the global convolution unit to obtain the final filtering result.

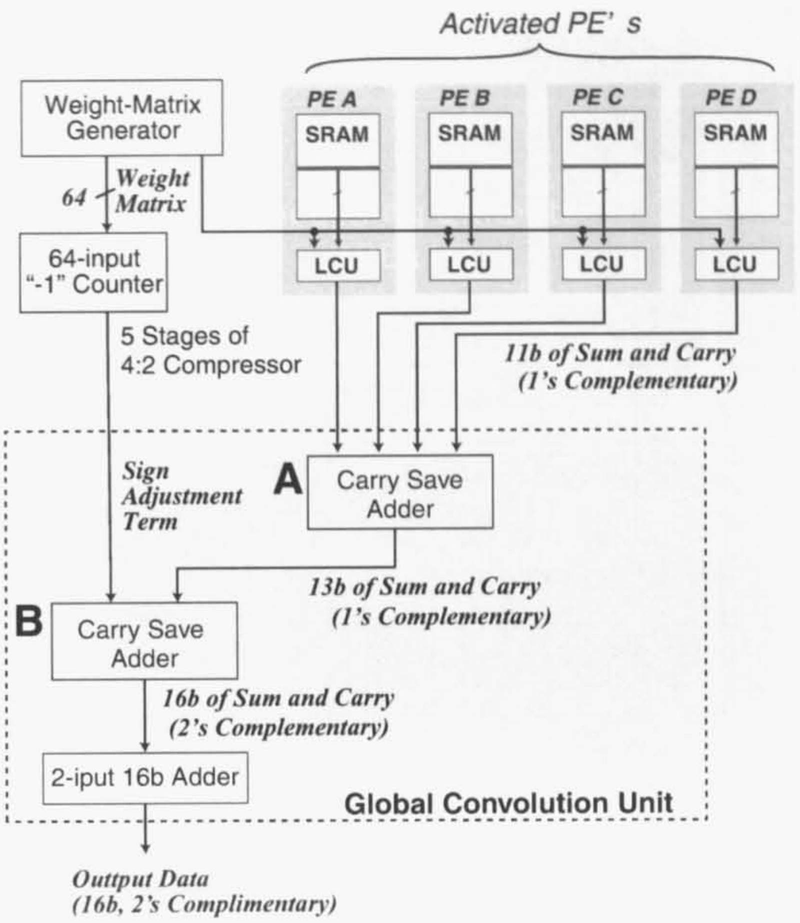


Figure 2.8: Overall data path and global convolution unit.

2.2.7 Global Convolution

Fig. 2.8 depicts the accumulation data path of the overall system. The results from four activated PE's are transfered to the global convolution unit, and they are accumulated by the carry save adder **A**. Up to this stage, negative numbers are represented as 1's complimentary. The sign adjustment term is generated in coherent to these accumulation operations. Namely, the number of -1's in the 8×8 weight matrix is counted by the 64-input "-1" counter composed of five stages of the 4:2 compressors. Then, the result of the accumulation and the sign adjustment terms are added by the carry save adder **B** according to the Eq. 2.4, thus being converted to the 16b of 2's complementary

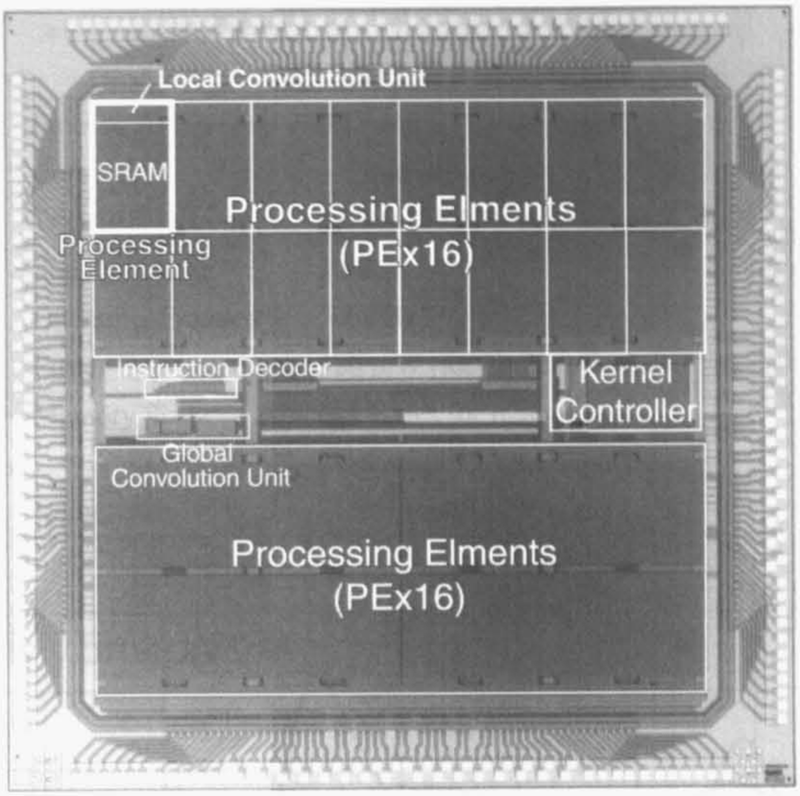


Figure 2.9: Photomicrograph of prototype chip.

representation. Finally, the two 16b numbers (the sum and the carry) are added by the 2-input adder to yield the final result.

2.3 Experimental Results

2.3.1 Chip Fabrication

A prototype chip was designed and fabricated in a 0.18 μm CMOS technology with five metal layers. All the modules were designed by hand layout to achieve a high packing density. A chip photomicrograph is shown in Fig. 2.9 and the chip specifications are summarized in Table 2.1.

Fig. 2.10 displays the measured waveforms from the prototype chip, executing the vertical edge enhancement filter shown in Fig. 2.11(d). 17ns is required for the total operation with 6ns for

Table 2.1: Prototype chip specifications

Process Technology	0.18 μm CMOS, 5-Metal
Core Size	4.5 mm $\times$ 4.6 mm
Transistors Count	4,000k (SRAM: 3,500k Logic: 500k)
Supply Voltage	1.8 V
Operating Frequency	50M Hz
Power Consumption	180 mW (for Typical Kernel)
Memory Size	64k Bytes for Image
	2k Bytes for Kernel Data
Kernel Size	$\sim$ 5x5 Variable

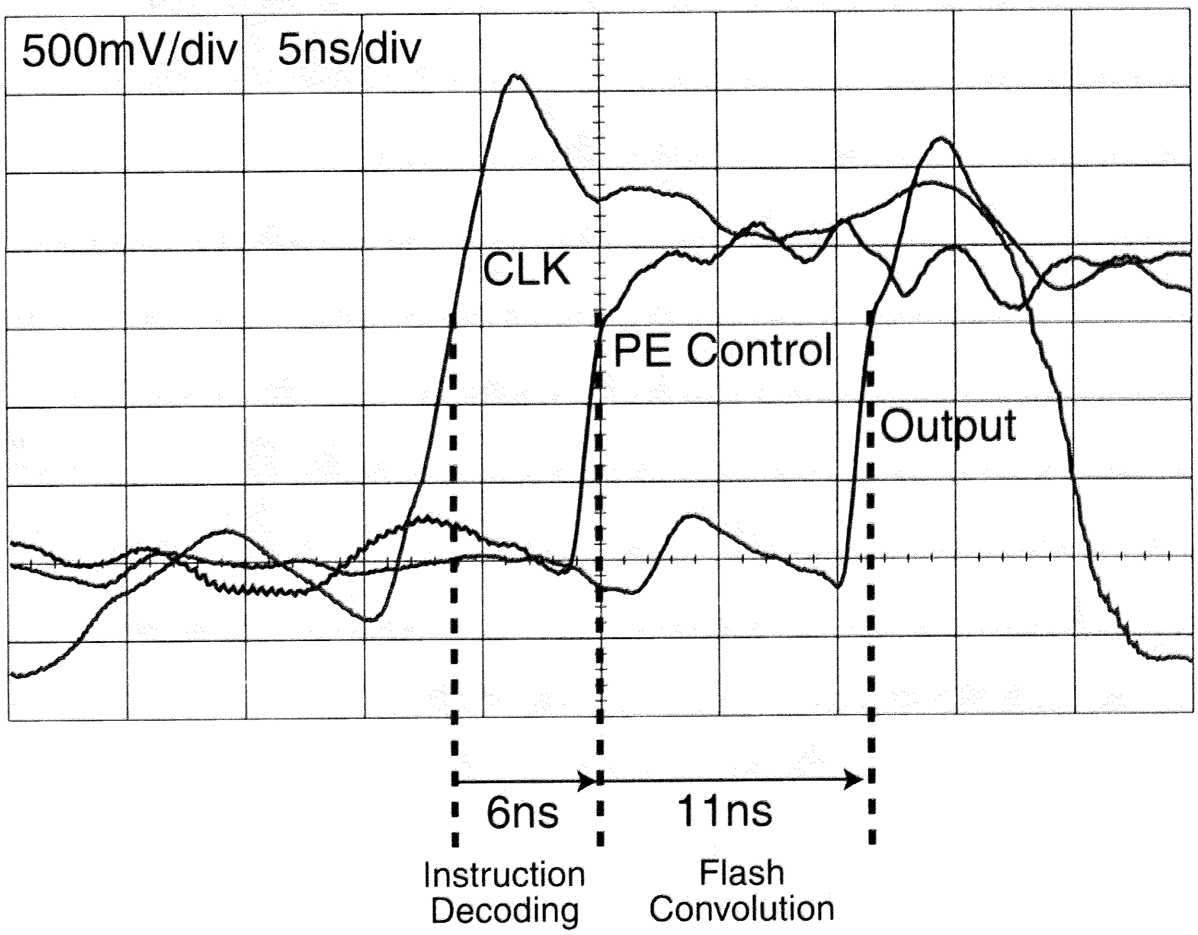


Figure 2.10: Measured waveforms of prototype chip.

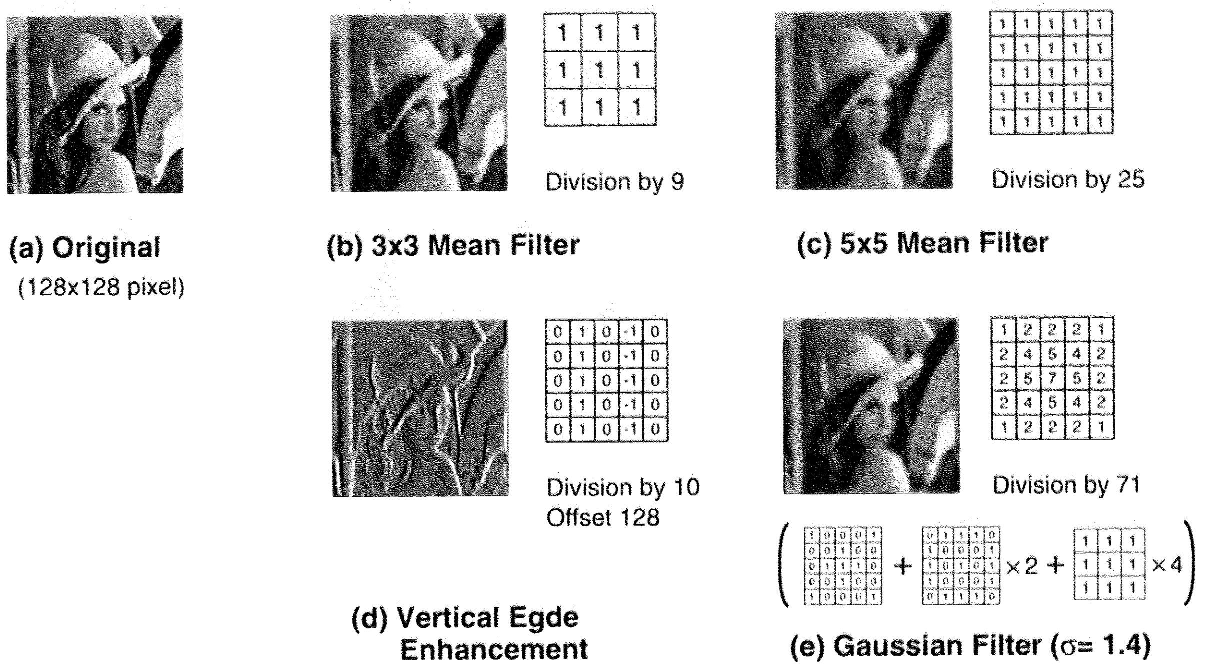


Figure 2.11: Measurement results of image filtering.

instruction decoding and 11ns for calculation. A typical operation frequency was 50MHz at the supply voltage of 1.8V. Since the present version was developed as a proof-of-concept chip for the flash convolution scheme, all the operations are executed within a single clock period. Therefore, introduction of a pipeline architecture will increase the operation frequency.

The experimental results of image filtering using the fabricated processor are demonstrated in Fig. 2.11. The 128×128 original image and four filter data sets were stored in the processor. Typical filtering was finished in a single clock cycle (Figs. 2.11(a), (b), (d)). In the case of a 5×5 Gaussian filter shown in Fig. 2.11 (c), three sets of weight values were utilized consecutively, thus requiring three clock cycles to complete the convolution. This is because only three weight values +1, 0, -1 are available on the chip.

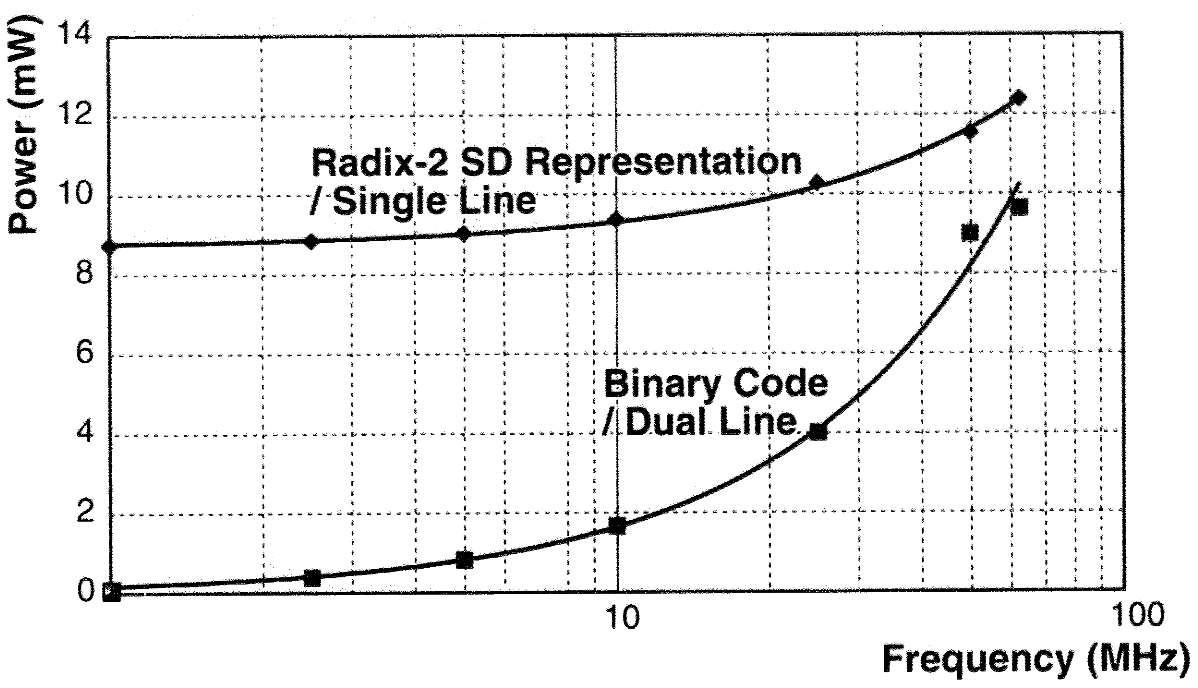


Figure 2.12: Comparison of the power dissipation between the radix-2 signed digit (SD) representation and binary codes obtained by measurement.

2.3.2 Power Dissipation in Radix-2 Signed Digit Data Transmission

Fig. 2.12 compares the bus power dissipation between the radix-2 signed digit (SD) representation and a binary code. The data were obtained from the measurement of the fabricated chip as described below.

By changing the number of -1's in the 8×8 weight matrix, the variance in the total power dissipation were observed and the average power due to the "-1" data transmission was estimated. The data are plotted as a function of operation frequency up to 50MHz, the maximum frequency of the present chip. The power dissipations due to the binary code transmission was measured in a similar manner by changing the number of +1's in the matrix. Since two lines are necessary to transfer three types of digit (+1, 0 and -1) with binary representation, the data shown in the figure are two times the data estimated for a single line.

Although the power dissipation is higher for the radix-2 SD representation in the lower frequency region due to the static current in the $V_{DD}/2$-generator, the power is less sensitive to the

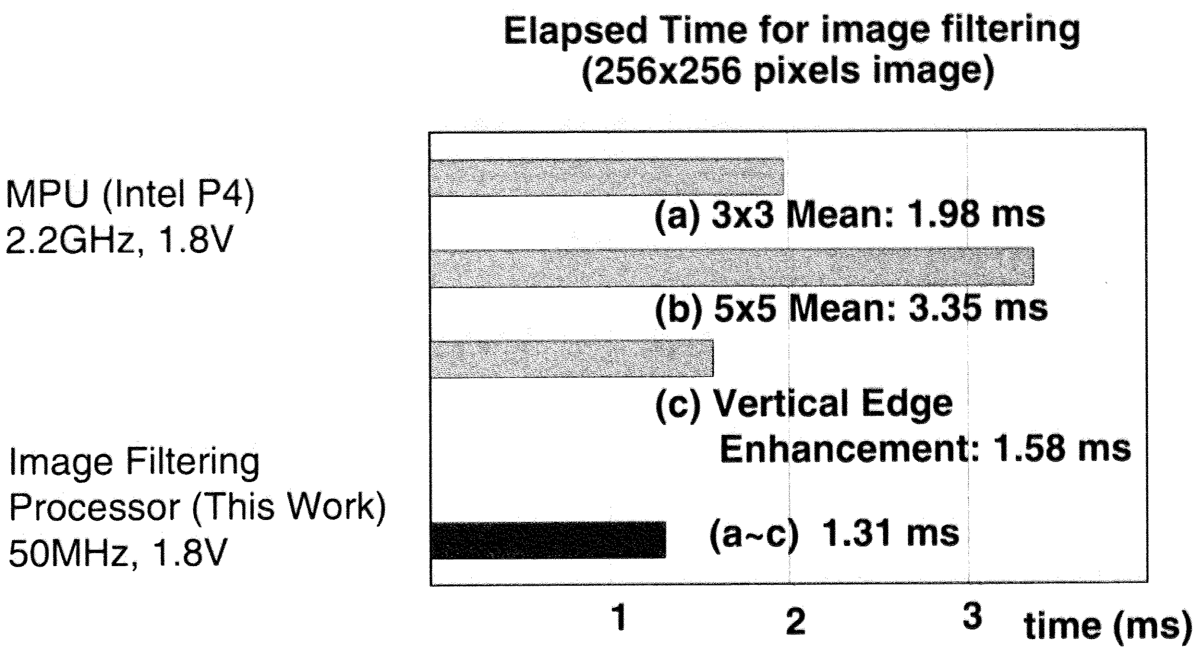


Figure 2.13: Performance comparison to general purpose MPU system.

increase in the frequency as compared to the binary representation. This is because the signal swing employing the $V_{DD}/2$-generator is half as large as the swing employing the digital buffers. The power dissipation for binary code transmission would exceeds that for the radix-2 SD transmission at frequencies larger than 100MHz.

2.3.3 Performance Comparison

Fig. 2.13 demonstrates the performance comparison between the image filtering processor and the software processing running on a MPU (Intel P4 2.2GHz). The image filtering with three kernels, a 3×3 mean filter, a 5×5 mean filter and a vertical edge enhancement filter (shown in the Fig. 7(b) $\sim$ (d), respectively), were executed by the image filtering processor on a 256×256 pixel image, and the execution time for filtering was compared with the time for which the MPU carried out an image-filtering software compiled by GNU C compiler with the most aggressive optimization option. While the execution time of MPU varies from 1.58m to 3.35ms depending on the type of

Table 2.2: Performance comparison to SPA processor: 5 × 5 convolution

	SPA Processor [11] 64×64 Square Array	(256×256)	Image Filtering Processor 256×256 pixel
Technology	0.6μm	(0.25μm)	0.18μm (Only Gate) 0.35μm (Other Layers)
Memory Type	DRAM		SRAM
Power Supply	2.5 V	(1.0 V)	1.8 V
Power Dissipation (Typical)	300 mW	(2000 mW)	180 mW
Area / Pixel	19222μm^2	(3337μm^2)	315μm^2
Time / Frame	730μs	(304 μs)	1.31ms
Area Efficiency for Speed	14 μm^2·s	(1.02 μm^2·s)	0.41 μm^2·s
Energy / Pixel	55 nJ	(9.28nJ)	3.6 nJ

the kernel, the image filtering processor achieves 1.31ms of execution time for any kernels since it carries out these convolutions within a single clock cycle. Note that the image filtering processor works at the operational frequency of 50MHz as compared to 2.2GHz of the MPU. Therefore, the power efficiency of the proposed architecture is about 40 times as large as that of the MPU.

Table 2.2 compares the present work with the recent square processor array (SPA) [11]. This summarizes performance characteristics for 5×5 convolution. Area per pixel is equal to the chip area divided by the number of pixels handled by a single chip. The product of area per pixel and time per frame indicates an area efficiency for processing speed, because the inverse of this figure gives processing speed per unit of silicon area. Energy per pixel are given as the product of typical power dissipation and time per frame divided by the number of pixels. In fact, only a gate length is 0.18 μm in our technology and other layers are laid out under a 0.35-μm process (For example, wire pitches in the first $\sim$ third metal layer is 0.8 μm, and wire pitches in the fourth and top metal is 1.2 μm.) Therefore, this technology seems to be comparable to 0.25-μm for naive comparison. In order to consider technology scaling, the performances of the 256×256 SPA processor in 0.25-μm technology are also estimated. The function of our processor is limited to the image filtering, while the work in [11] integrates 1b ALU for general purpose. However, in terms of the convolution, the table shows that 50% reduction of area efficiency for speed is achieved though 6-transistor SRAM cell is employed in the memory module, and energy per pixel of our processor is about 30% of that of the work in [11]. Moreover, further improvements will be expected with introducing embedded DRAM technology.

2.4 Summary

In this work, a low-power and high-speed image filtering processor conducting a single-clock-cycle convolution with various kernel has been presented. In order to eliminate complicated memory address control and a large number of redundant memory access, a quaternary-tile pixel-mapping method has been developed. In addition, a radix-2 signed digit data transfer scheme is employed in a global data distribution in order to reduce not only wiring area but also power dissipations. A prototype chip was designed and fabricated in a 0.18-μm CMOS technology. Without pipelining, the fabricated processor operated at 50MHz with a 1.8V supply is able to conducts image filtering for a 256 $\times$ 256 image within 1.31ms, which demonstrates better performance than a 2.2GHz MPU systems.

Chapter 3

A Computational Digital-Pixel-Sensor VLSI

3.1 Introduction

Real-time image processing plays an important role in various applications, such as robotics, automotive guidance and security surveillance. However, the image data in real world are massive in quantity, and the processing is computationally very expensive. Regarding the image acquisition, high resolution images are easily obtained using solid-state image sensors, in which a large number of pixels are integrated on a single chip thanks to the remarkable advances in silicon technology. Therefore, how to process the acquired data at each pixel is very important issue to develop real-time image processing systems.

CMOS image sensors have become viable imaging devices due to its inherent technological advantages [41]. Especially, CMOS imagers have a capability to integrate pixel arrays with processing circuits into a single chip, thus providing a potential opportunity to develop high-speed on-chip image processing systems. Moreover, as the CMOS technology scaling advances to deep sub-micrometer levels, integration of more intelligence and further processing into each pixel is becoming feasible [34]. A digital pixel sensor (DPS) [42–44], which performs the analog-to-digital conversion (ADC) at the pixel level, is an example of the new design concepts made possible due to the scaling of CMOS devices. It is already discussed in literatures [34, 45–47], DPS' offer several benefits over analog image sensors, such as higher signal-to-noise ratio, lower-power operation, and wide dynamic range. In addition to these benefits, the compatibility with digital processing is an important feature of DPS' for on-chip imaging systems. Namely, converted data are stored in a digital format at each pixel, thus high-speed readout is realized in a manner similar to a random access memory. For example, the recent DPS design by the Stanford University has integrated both single-slope bit-parallel ADC and 8b DRAM cell into each pixel, thus achieving the throughput of 10,000 frames per second and readout rate of over 1.3 GB/s [48]. Therefore, combination of the DPS architecture and digital processing units is expected as one of the candidates to realize real-time imaging systems [49].

However, in terms of downloading pixel information, like most of analog imaging devices, conventional DPS' employ row-sequential readout scheme. Namely, each row of the two-dimensional pixel arrays are accessed and downloaded at one time, and this operation is repeated row-by-row until required pixel data are readout. However, such a one-row readout scheme, so-called "*line-readout*", is not compatible to image-processing algorithms since pixel processing in the algorithms usually requires coordination of the neighboring pixel informations. For example, image filtering process is carried out on a block of neighbor pixels, e.g., a 3×3 or 5×5-pixel block, etc. In the case of line-readout architecture, data need to be buffered until all five rows of data are downloaded before starting a 5×5 image filtering operation. Therefore, it requires additional buffer memories and control logics, thus causing not only a performance bottleneck but increasing power dissipations.

In this chapter, we have developed a computational digital-pixel-sensor architecture which allows the parallel readout of block-of-pixel data for image processing by SIMD (single instruction multiple data) processing elements. This architecture is capable of seamlessly scanning a 5×5 pixel-kernel filter across the entire pixel array. As a result, the proposed DPS VLSI has compatibility to

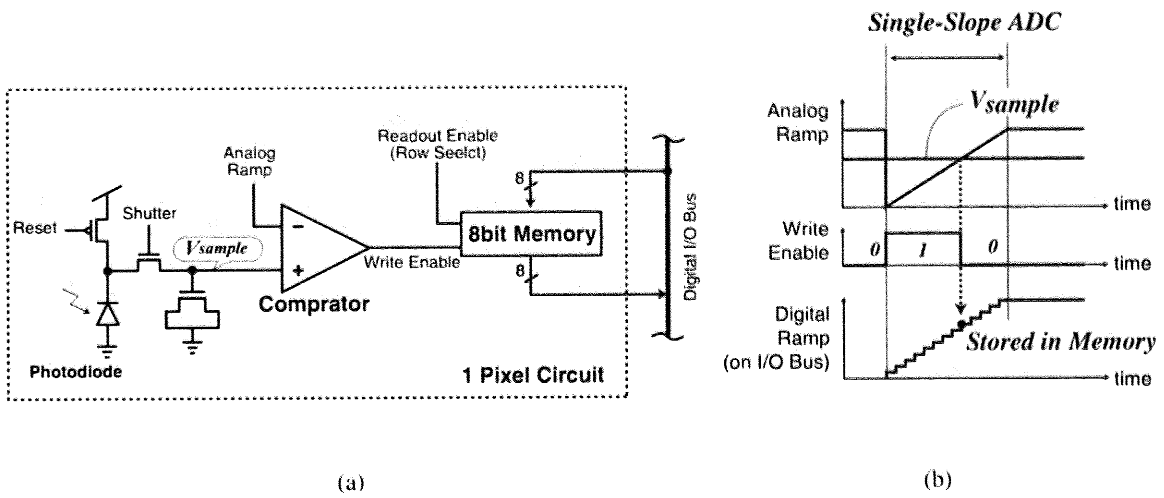


Figure 3.1: Block diagram of digital pixel sensor: (a) pixel configuration; (b) analog-to-digital conversion timing chart.

various image-processing algorithms. In this paper, a rank-order-filtering circuit has also been developed as one of the application to demonstrate on-chip image processing. It unifies the rank order filtering algorithm [50] and binary search tree into one simplified circuit. The concept has been verified by a proof-of-concept chip fabricated in a 0.35-μm CMOS technology. The chip includes a 64×48 DPS array and eight units of the rank-order-filtering circuit element.

The rest of this chapter is organized as follows. The block-readout digital-pixel-sensor architecture and the system configuration are explained in Section 3.2. An organization of the rank-order filtering processor is given in Section 3.3. The prototype chip implementation and the experimental results are presented in Section 3.4. Finally, the summary of this chapter is given in Section 3.5.

3.2 Block-Readout Digital-Pixel-Sensor Architecture

3.2.1 Digital-Pixel-Sensor Concept

Fig. 3.1 shows the basic concept of the digital pixel sensor (DPS). A simplified block diagram of a pixel unit of the DPS is illustrated in Fig. 3.1(a). The pixel unit consists of a photodiode, a reset

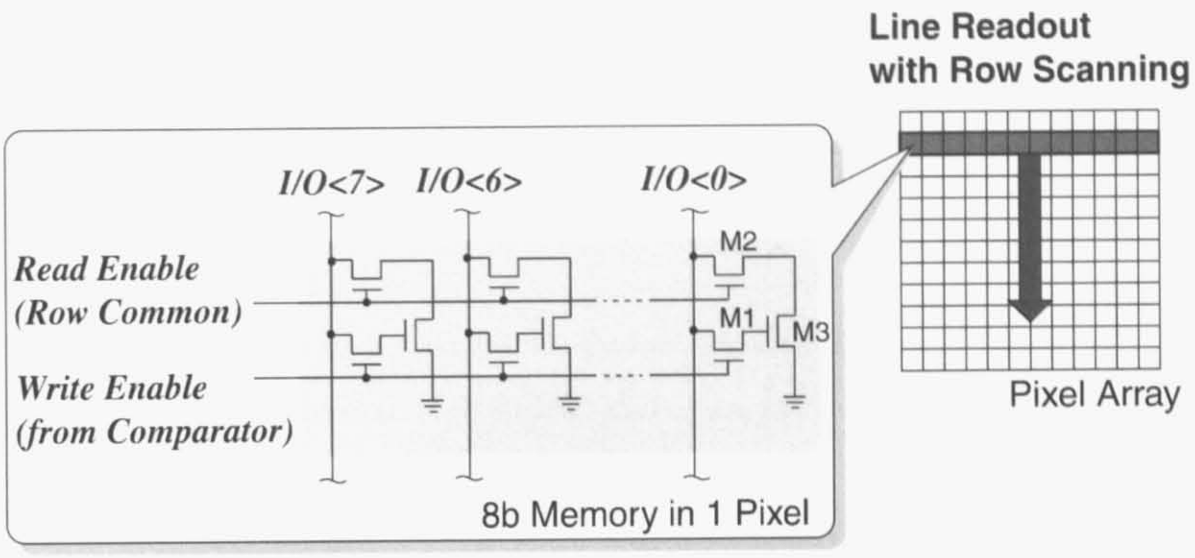


Figure 3.2: Line-readout memory architecture.

transistor, a shutter transistor, a sampling MOS capacitor, an analog comparator, and eight bit memory cells. Fig. 3.1(b) shows the operation timing chart of pixel-level analog-to-digital conversion (ADC). Before ADC, light intensity at each pixel is converted to the analog voltage (V_{sample}) on the MOS capacitor in a similar manner of analog CMOS imagers. Then, the synchronous signals, an analog ramp signal and eight-bit digital ramp codes, are provided for ADC. The analog comparator in each pixel compares V_{sample} with the analog ramp. At the beginning of ADC, the analog ramp is below V_{sample}, and the comparator output *Write Enable* is "1". At the time when the analog ramp voltage exceeds V_{sample}, *Write Enable* turns to "0", thus the memory stores the corresponding value of the digital ramp as the digitized data of V_{sample}. This operation is conducted simultaneously at all pixels, thus achieving massively parallel ADC. This architecture has been proposed by Kleinfelder et al. in 2001 [48], and the same concept for pixel-level ADC is also employed in this work.

3.2.2 Block-Readout Memory Architecture

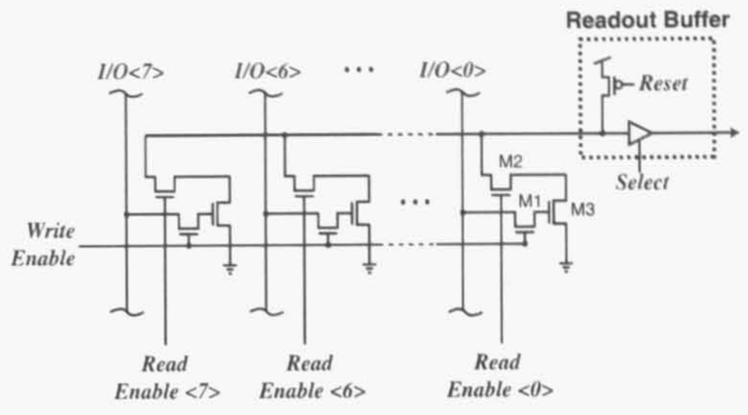
Fig. 3.2 depicts a schematic of the memory architecture utilized in [48], where each memory cell is composed of three transistors: a write-switch transistor M1, a read-switch transistor M2 and the storage transistor M3. All M1's in a pixel are connected to the *Write Enable* which is transfered from the comparator in the pixel unit. As mentioned in the previous subsection, *Write Enable* is controlled according to the pixel intensity, thus recording the data broadcasted on the I/O lines into the M3's. On the other hand, readout operation is controlled by the signal *Readout Enable*. When *Readout Enable* is turned on, the M2 is activated and the recorded data on the M3's are read out through the I/O lines. In the conventional line-readout architecture, *Readout Enable* is shared by pixels in a single row. Therefore, every bit of all pixels in one row are readout in parallel when the *Readout Enable* is turned on. Therefore, it is impossible to access any pixels in the rest of rows because all I/O lines are occupied by the pixels in the activated row. As a result, the pixel data in the array must be read out row-by-row.

Fig. 3.3 shows the block-readout memory architecture, which enables block-readout of data in eight rows simultaneously. Fig. 3.3(a) illustrates the schematic of 8b memory structure in each pixel. In contrast to the line-readout architecture, the read control signals (*Read Enable<7:0>*) are all independently controlled among eight memory cells. Fig. 3.3(b) shows the readout operation employing block-readout memory architecture. One pixel data are read out using a single I/O line via readout buffer. Since only one I/O line per pixel is utilized for readout, the remaining I/O lines are used for other pixels in the same column. In this manner, selected bit data in an eight-row block are read out simultaneously. As a result, this architecture allows us to efficiently carry out bit-serial image processing for blocks of pixel data.

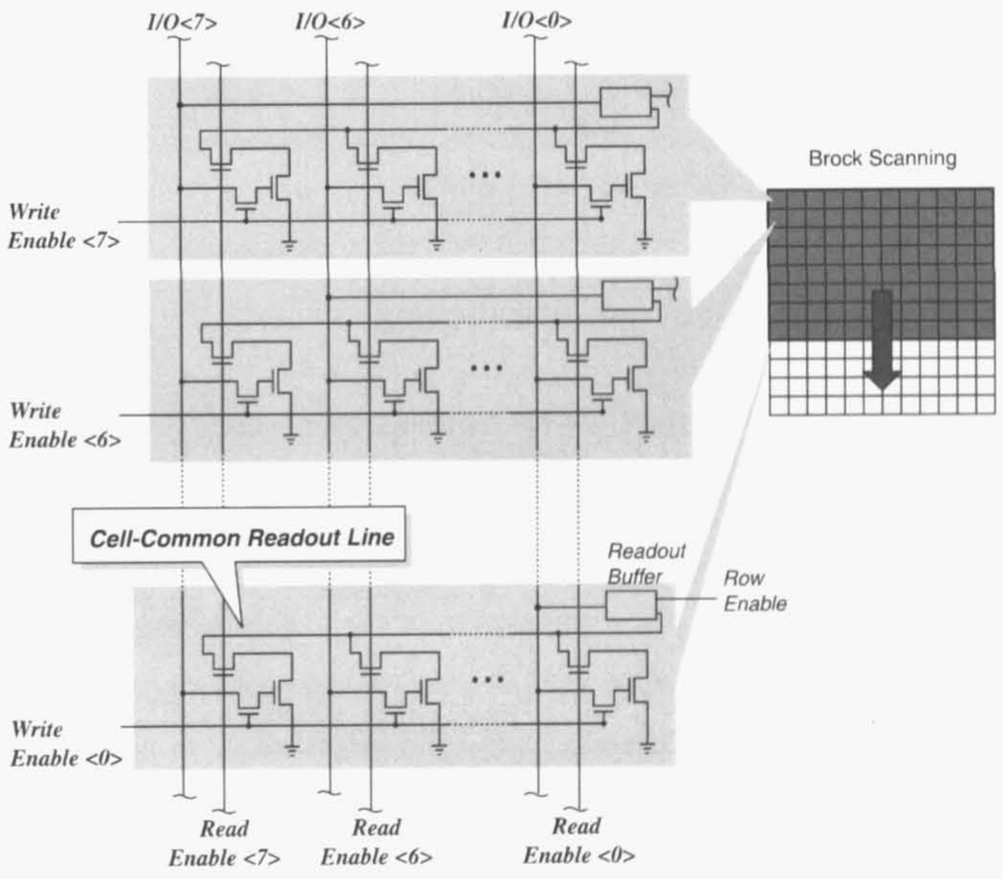
3.2.3 Quaternary-Tiled-Mapping Access Scheme

Fig. 3.4 illustrates an overall architecture of computational digital-pixel-sensor VLSI developed in this work. It consists of 64×48 block-readout pixel array, a row address controller and SIMD processing elements. The VLSI carries out on-chip image filtering operation over image data captured by the DPS array, where seamless scanning with a filtering kernel is realized.

In order to realize the seamless scan of up to 5×5-pixel kernel, the quaternary-tiled access



(a)



(b)

Figure 3.3: Block-readout memory architecture

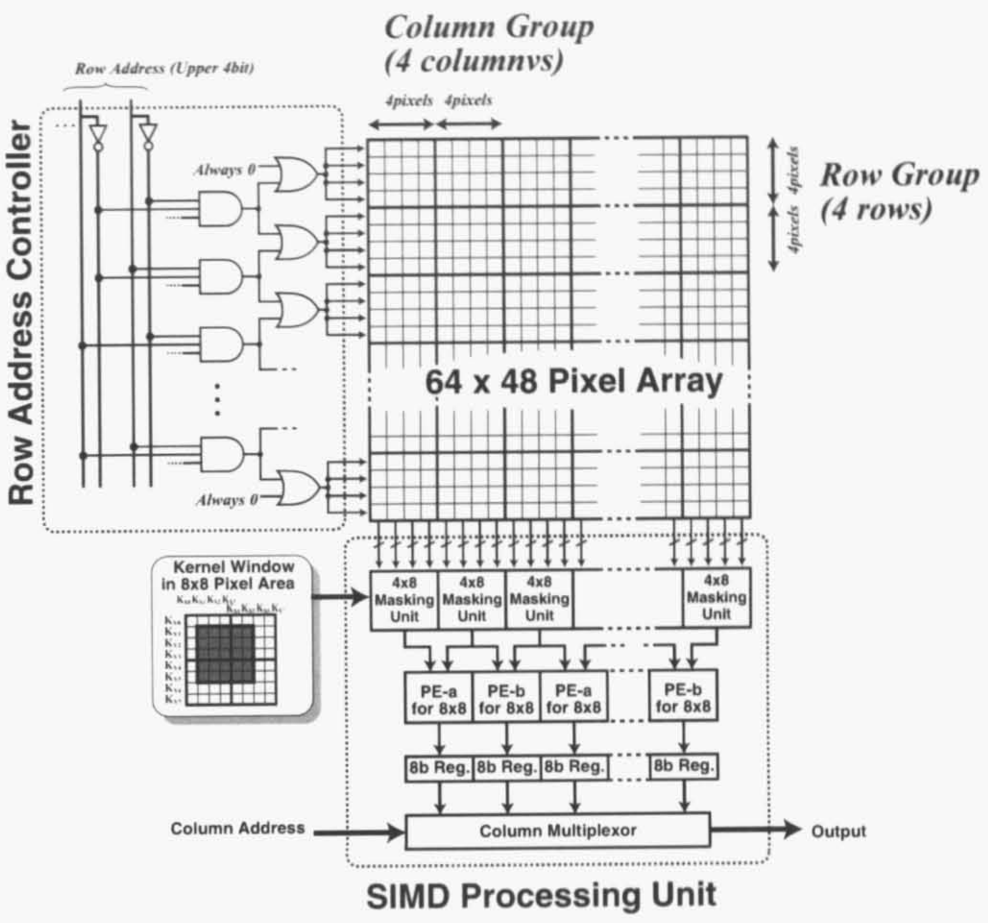


Figure 3.4: Block diagram of computational digital-pixel-sensor VLSI

scheme is introduced [51]. Referring to the quaternary tile mapping model, four rows of the array are considered as one group. The row address decoder, which consists of a series of AND gates and OR gates, activates two groups according to a row address code. Namely, eight rows in the two adjacent groups are activated at one time when a particular row address is provided. Therefore, in each column, a particular bit data in the activated eight-row block are downloaded to the SIMD processing unit due to the block-readout memory architecture explained in the above subsection.

The SIMD processing unit is composed of masking units for pixel manipulation, processing elements (PE) for 8×8 kernel operation, 8b result registers and a column multiplexer for readout. In horizontal directions, four columns are also grouped, and the downloaded data of one group

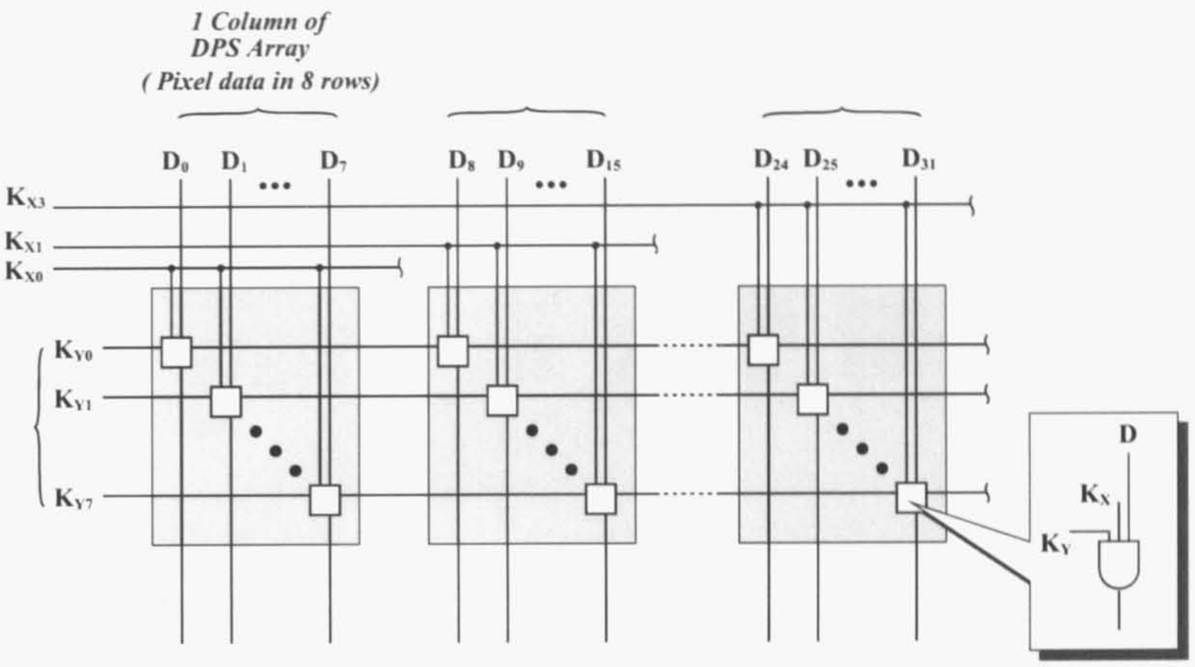
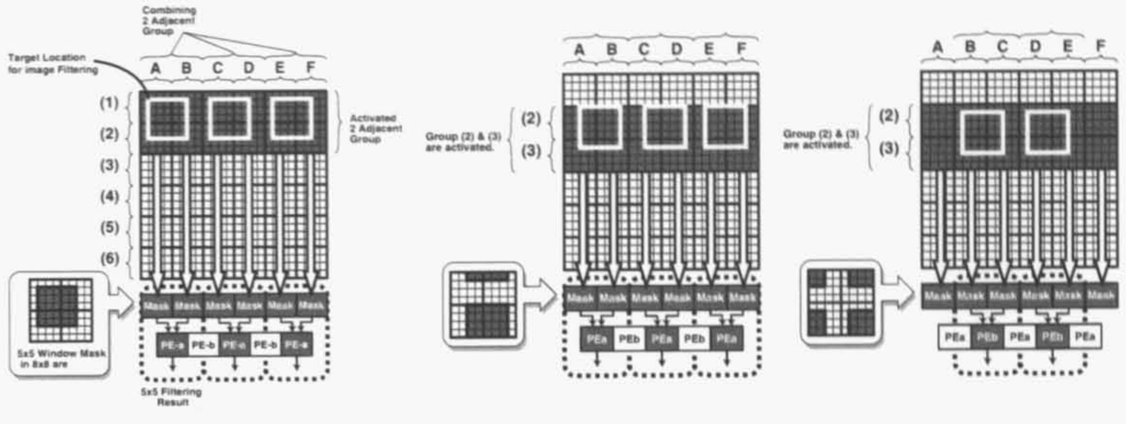


Figure 3.5: Block diagram of 4×8 masking unit. It defines 4×8 kernel window according to $K_{X0} \sim K_{X3}$ and $K_{Y0} \sim K_{Y7}$

(4columns × 8rows) are input to a single masking unit. Fig. 3.5 shows the block diagram of the masking unit composed as 4×8 3-input AND matrixes. It decides whether to pass the downloaded pixel data or not according to the kernel arrangement represented as two binary codes: $K_{X0} \sim K_{X3}$ and $K_{Y0} \sim K_{Y7}$. Thus, neighboring two masking units define the kernel window in the 8×8 pixel area. As explained in Section 2.2.3, this configuration makes possible to carry out seamless scanning of 5×5-pixel kernel window over the DPS array.

Each processing element (PE) is placed at the midpoint of two adjacent masking units, thus being able to access the data from the two units, i.e. the masked result of the 8×8 pixel area. PE is divided into two groups, PE-a and PE-b, in a manner that the adjacent PE's are assigned to the different groups. The details of the kernel scanning operation is illustrated in Fig. 3.6. The white squares on the pixel array indicate target locations for image filtering. In the example shown in Fig. 3.6(a), the row groups of **(1)** and **(2)** are activated. The masking units manipulates the downloaded data according to the provided kernel window. Then, PE-a's carry out the image filtering on the 8×8

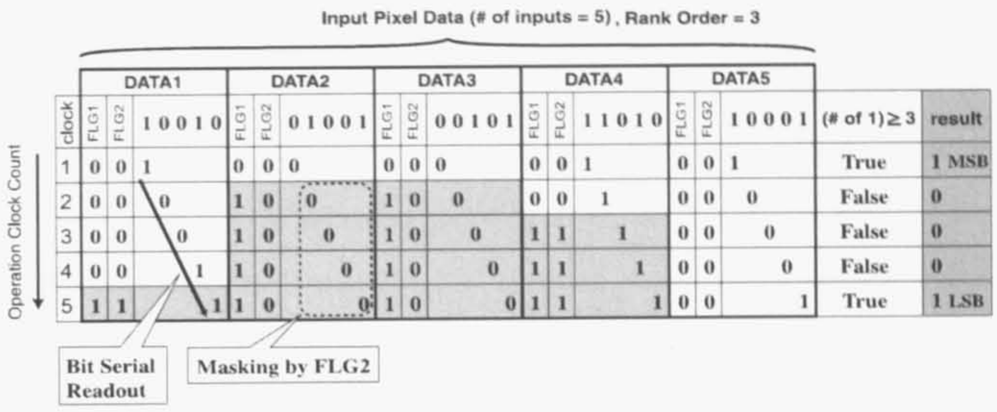

Combining 2 Adjacent Group
Target Location for image Filtering
A B C D E F
(1)
(2)
(3)
(4)
(5)
(6)
Activated 2 Adjacent Group
5x5 Window Mask in 8x8 are
Mask
PE-a
PE-b
5x5 Filtering Result
Group (2) & (3) are activated.
PEa
PEb

(a) (b) (c)

Figure 3.6: Seamless Kernel Scanning.

pixel area. For example, a 8×8 pixel area included by the two vertical groups: group **(1)** and **(2)** and the two horizontal groups: group **A** and **B** is provided to the PE-a on the most left side, etc. In this case, only PE-a's are activated for image filtering. Finally, the image filtering results are available as the result of each PE-a's.

For vertical kernel scanning, the activated row groups are changed. When the kernel location is shifted down by up to 2 pixels from the location shown in Fig. 3.6(a), the activated row groups are not changed. Only the location of the kernel window supplied to the masking units are changed. When the kernel location is shifted down by 3 pixels, as shown in Fig. 3.6(b), the row groups of **(2)** and **(3)** are activated. Similar to vertical directions, the role of PE-a's and the PE-b's are switched for horizontal kernel scanning, as shown in the Fig. 3.6(c). In this manner, the seamless 5×5 kernel scanning is realized.


Input Pixel Data (# of inputs = 5) , Rank Order = 3

	DATA1			DATA2			DATA3			DATA4			DATA5				
clock	FLG1	FLG2	1 0 0 1 0	FLG1	FLG2	0 1 0 0 1	FLG1	FLG2	0 0 1 0 1	FLG1	FLG2	1 1 0 1 0	FLG1	FLG2	1 0 0 0 1	(# of 1)≥ 3	result
1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	True	1 MSB
2	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	False	0
3	0	0	0	1	0	0	1	0	0	1	1	1	0	0	0	False	0
4	0	0	1	1	0	0	1	0	0	1	1	1	0	0	0	False	0
5	1	1	1	1	0	0	1	0	0	1	1	1	0	0	1	True	1 LSB

Operation Clock Count
Bit Serial Readout
Masking by FLG2

Figure 3.7: Rank-order filtering algorithm chart

3.3 Rank-Order-Filtering Processor

3.3.1 Rank-Order Filtering Algorithm

Rank order filter is used in determining the ***n***th order in a set of ***N*** data, i.e., for the maximum value, ***n*** corresponds to 1; for the minimum value, ***n*** corresponds to ***N***. Bit-serial rank order filer algorithm was proposed in [50]. In this paper, the algorithms were modified in order to be easily implemented. We unified the algorithm into one simplified circuit for use in image filtering process. The circuit operates in a bit-serial manner that very well matches to the block-readout digital-pixel-sensor architecture developed in this work.

The adaptive algorithm employed in the implementation is explained by the example in Fig. 3.7. The figure shows the operation of finding the third order data in a five-data set (DATA1 $\sim$ DATA5). The operation begins from MSB and propagates to the following bit in the next clock cycle. FLG1 and FLG2 mark the status of each data. They are set to "0" in the initial state.

In the first clock, all MSB values of five data are summed. If the sum is greater than or equal to the assigned order, the result in this bit is "1". Vice versa, it's "0". In the example, the sum is equal to 3, thus, the search result in this bit is "1". Then, FLG1's of the data of which value is different from the result, i.e. DATA2 and DATA3, is converted to "0", and FLG2 is altered to the same value as its bit. As a result, all of the following bits of DATA2 and DATA3 are masked by their FLG2 which is 0 in the example.

In the second clock, the second MSB values are considered. The same procedure, mentioned in the first clock, is repeated. According to the example, the sum is one, and it is less than the assigned ordinal number. Thus, the result is "0". Here, the less significant bits of DATA4 are altered to "1". The same set of operations is repeated down to LSB. The result is available as soon as the calculation of LSB is done. Therefore, "10001" is determined as the third order in this data set.

3.3.2 Circuit Configuration

Fig. 3.8 shows the block diagram of the rank-order-filtering circuit which conducts 5×5 rank-order filtering processing on a 8×8-pixel area. It consists of 4×8 masking units, control logic circuits for input data manipulation, 32-input 1b adders, 2-input adders and subtracters for result detection, and an 8b shift resistor. As described in Section 3.2.3, two neighboring masking unit defines 5×5 kernel window in the downloaded 8×8-pixel area. Then, each pixel data are manipulated by the control logic circuit. Fig. 3.9 illustrates the details schematic of the logic circuits for one pixel data. The logic circuit is comprised of three parts: a FLG1 logic circuit, a FLG2 logic circuit and a selector, thus handling the pixel data in a bit-serial manner according to the algorithm explained in the above subsection. The pixel data manipulation in the control logic circuit can be carried out in parallel among the pixels, thus control logic circuits are grouped for 4 columns in a similar manner as the 4×8 masking unit in order to reduce the volume of the circuitry. The result of the two column groups are added by 2-input adder, thus being subtracted with the rank order assigned externally. The MSB of the subtraction result is fed back to the control logic circuits for determining the flags (FLG1 and FLG2). Also, the MSB bit is stored into the result shift register. After eight clock cycles, the rank-order filtering result is provided.

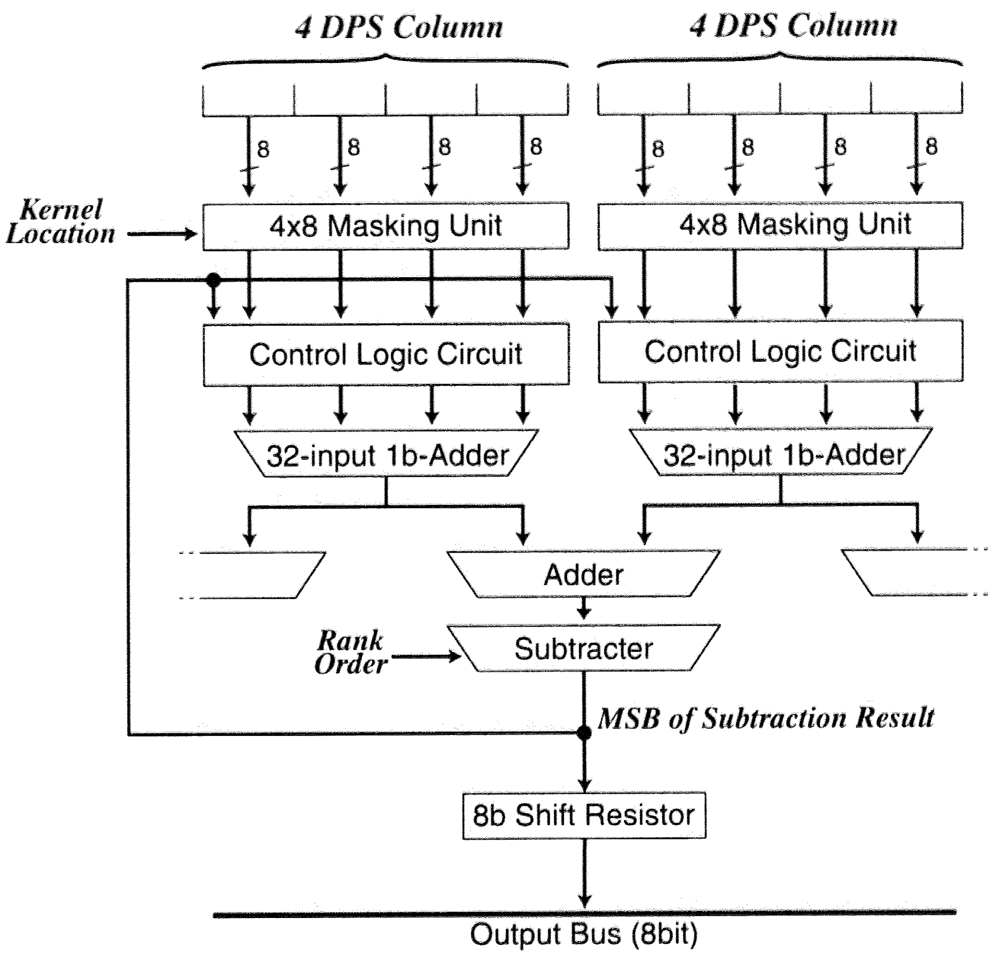


Figure 3.8: Block diagram of rank-order filtering processor.

3.4 Experimental Results and Discussions

3.4.1 Chip fabrication

In order to verify the proposed architecture, the proof-of-concept chip was designed and fabricated using 0.35μm standard CMOS technology. The photomicrograph of the chip is shown in Fig. 3.10(a) and the specifications are summarized in Table 3.1. The chip contains 303,500 transistors on 3.9$\times$3.9 mm^2 core. It implements 64$\times$48 pixels DPS array and eight filtering-circuit SIMD processing unit. Fig. 3.10(b) shows the pattern layout of one pixel unit.

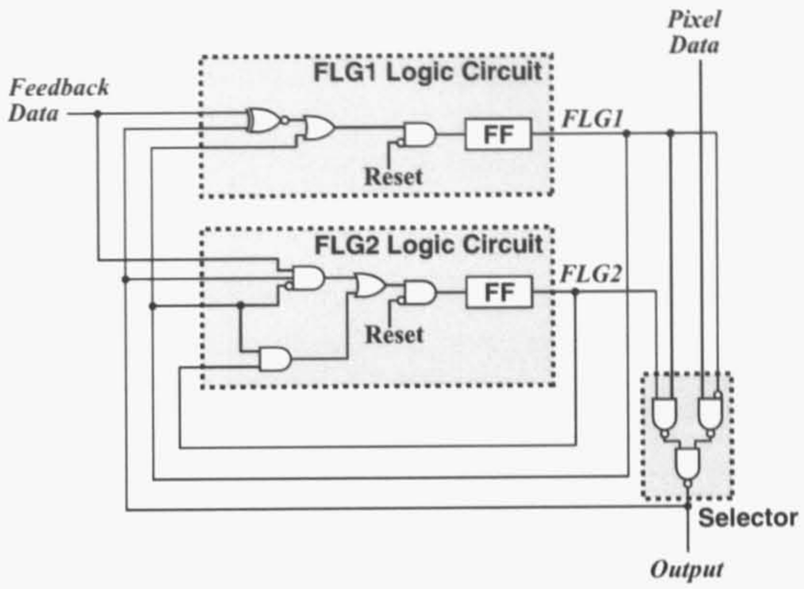
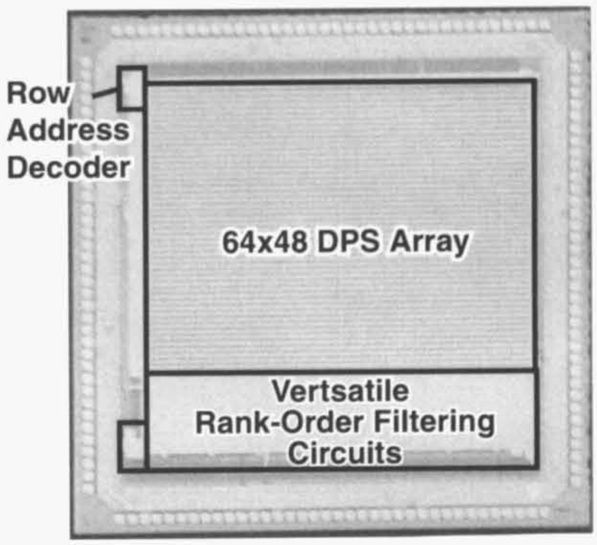
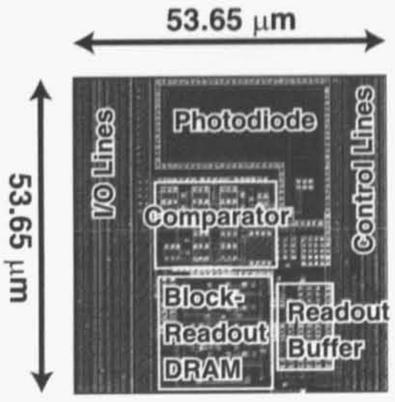


Figure 3.9: The schematic of the control logic circuit



(a)



(b)

Figure 3.10: Proof-of-concept chip: (a) chip photomicrograph; (b) pattern layout of 1 pixel unit.

Table 3.1: Specifications of the proof-of-concept chip

Process Technology	0.35 μm CMOS, 3-Metal
Core Size	3.9 × 3.6 mm^2
# of Transistors	303,500
Supply Voltage	3.3 V
DPS Array Size	64 × 48 pixels
Filtering Function	Rank-Order Filter /
Rank-Order Filter Kernel Size	5×5 (Maximum)
Operation Frequency	10MHz
Pixel Size	53.65 × 53.65 μm^2
# of Transistors (1 Pixel)	50
Fill Factor	14.9 %

3.4.2 Experimental Results

In order to verify the proposed rank-order filtering logic, one element of the rank-order filtering processor was observed by applying eight digital numbers directly. Fig. 3.11 shows the measured waveforms captured by the logic analyzer (TektronixTLA715). The supplied data were "10000111" (135 in a decimal format), "01010101" (85), "00100000" (32) "00010001" (17), "00010000" (16), "00001111" (15), "00000101" (5), and "00000001" (1). The figure shows the rank-order operation of the second smallest number search, and the result is "00000101". The operational frequency of the processor was 10MHz with 3.3V supply. Since the present version is developed as a proof-of-concept chip for the rank-order filtering logic, the elemental circuits are not optimized in terms of the circuit speed. In particular, the adder and the subtracter in the rank-order filtering processor take 80% of the operation time, as estimated by HSPICE simulation. Therefore, optimization of these circuitry will increase the operation frequency.

Fig. 3.12 displays measurement result of the proof-of-concept chip. Fig. 3.12(a) shows the image where the data captured at each pixel site are readout as they are, and Fig. 3.12(b) shows the 3×3 media filtered image, i.e., the data of 5th order in 3×3-pixel kernel window at each pixel site are relocated as the result. On the other hand, Fig. 3.13 shows the measurement result conducting

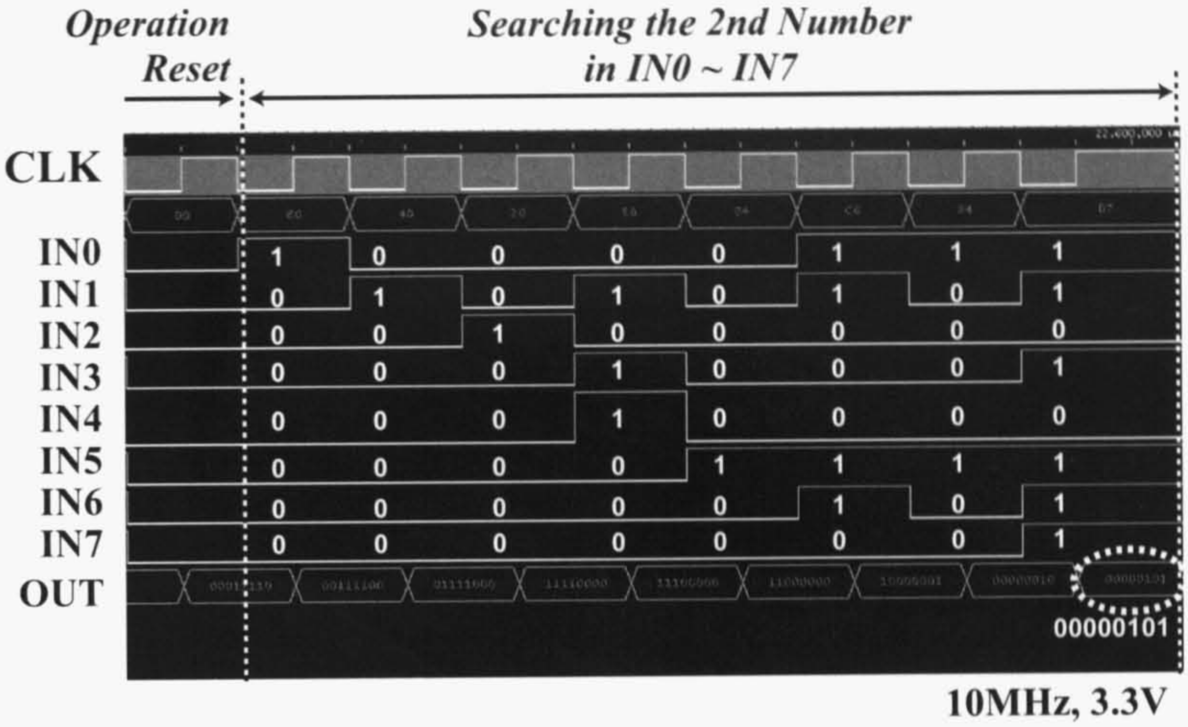


Figure 3.11: Measured waveforms of rank-order-filtering circuit searching the second number in the input IN0 $\sim$ IN7.

5$\times$5 dilation filter where the data 5th order in the 5$\times$5-pixel kernel window.

3.4.3 Comparison of Area Efficiency

Fig. 3.14 shows an area ratio of the circuits in a pixel, which compares the present work and the same design in 0.18-μm 1P5m CMOS technology. The estimation in 0.18-μm technology is done based on the real layout pattern, as shown in Fig. 3.16(a). The number of metal layers utilized in the present work is only three, and the top of the metal layers is employed to cover the array to prevent from exposure. Therefore, more than 40% of the pixel is occupied by wiring metals, while the right of the figure shows that only 16% of the pixel is used for the wiring. As shown in the figure, by introducing an advanced process technology, the digital part becomes more compact in size.

On the other hand, Fig. 3.15 compares the area ratio of the circuits in this work with that of the circuits based on the conventional design (The layout pattern of the conventional design is shown in

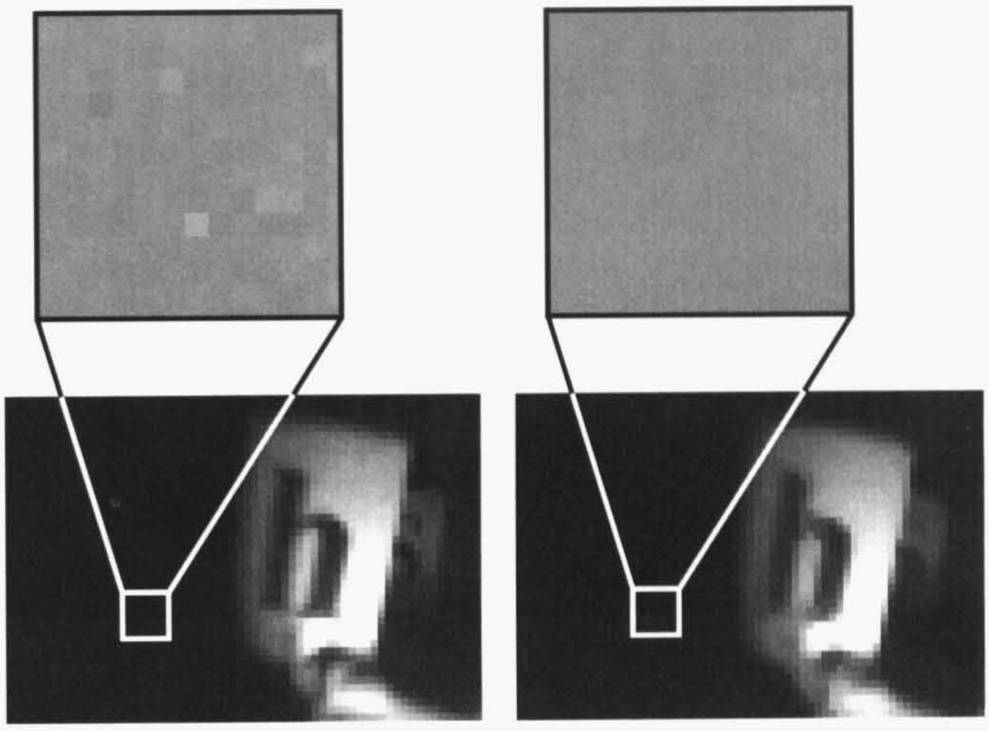


Figure 3.12: Measured image: (a) normal capture: (b) 3x3 median filter.

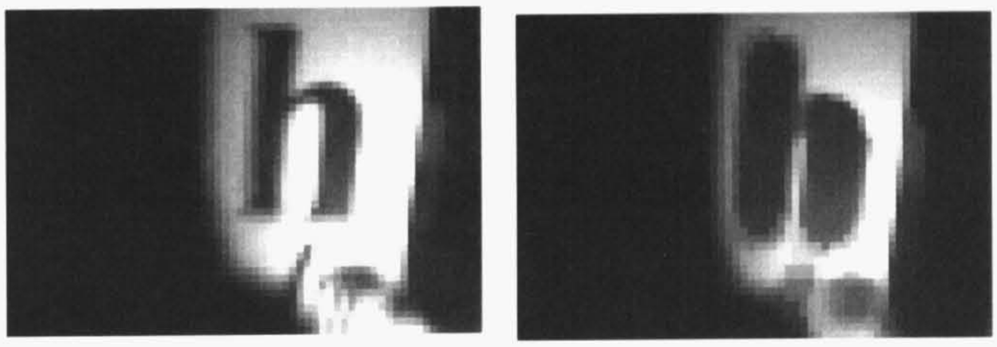


Figure 3.13: Measured image: (a) normal capture: (b) 5x5 dilation filter.

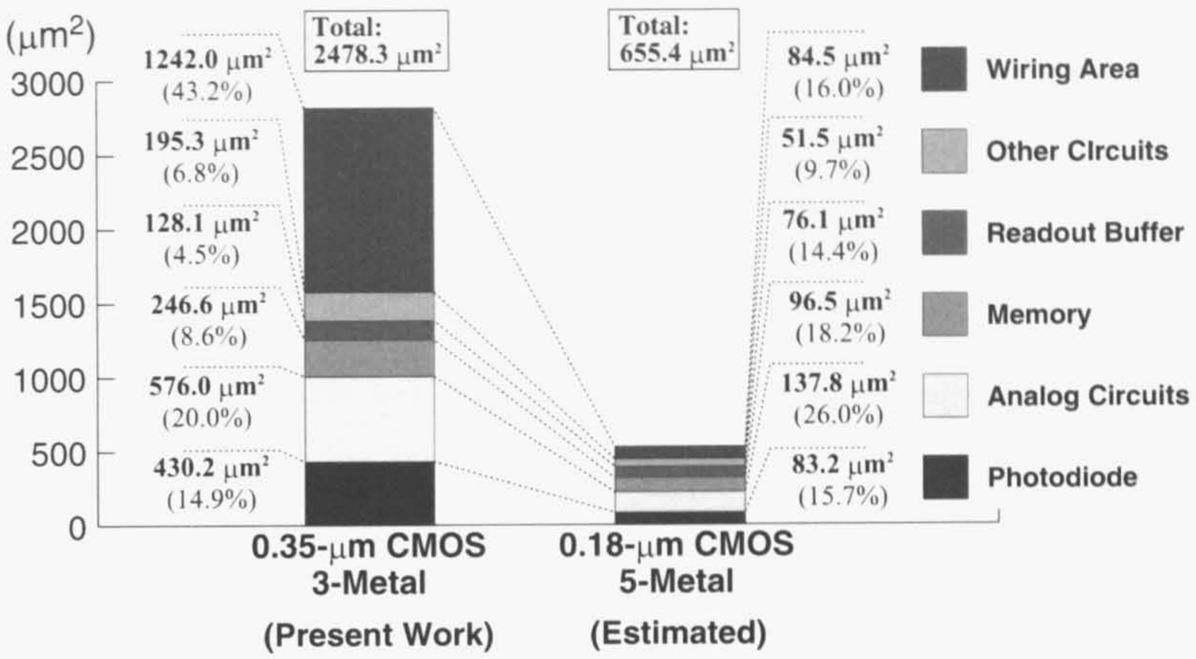


Figure 3.14: Area efficiency against technology scaling.

Fig. 3.16(b)). There are additional areas for block-readout memory architecture, however, the total pixel size is increased by only 15% of the area. As a result, by introducing an advanced process technology, the digital part becomes more compact in size and the area overhead of block-readout architecture would be neglected because the bottleneck of area efficiency would become the area of optics and analog circuitry.

3.5 Summary

A computational digital-pixel-sensor architecture has been presented, which allows us to readout block-of-pixel data from a pixel array. Therefore, the circuit has capability of seamlessly scanning a 5×5-pixel kernel filter across the entire pixel array. The rank-order filter has also implemented in a simplified circuit. The concept has been verified by a proof-of-concept chip fabricated in a 0.35-μm CMOS technology.

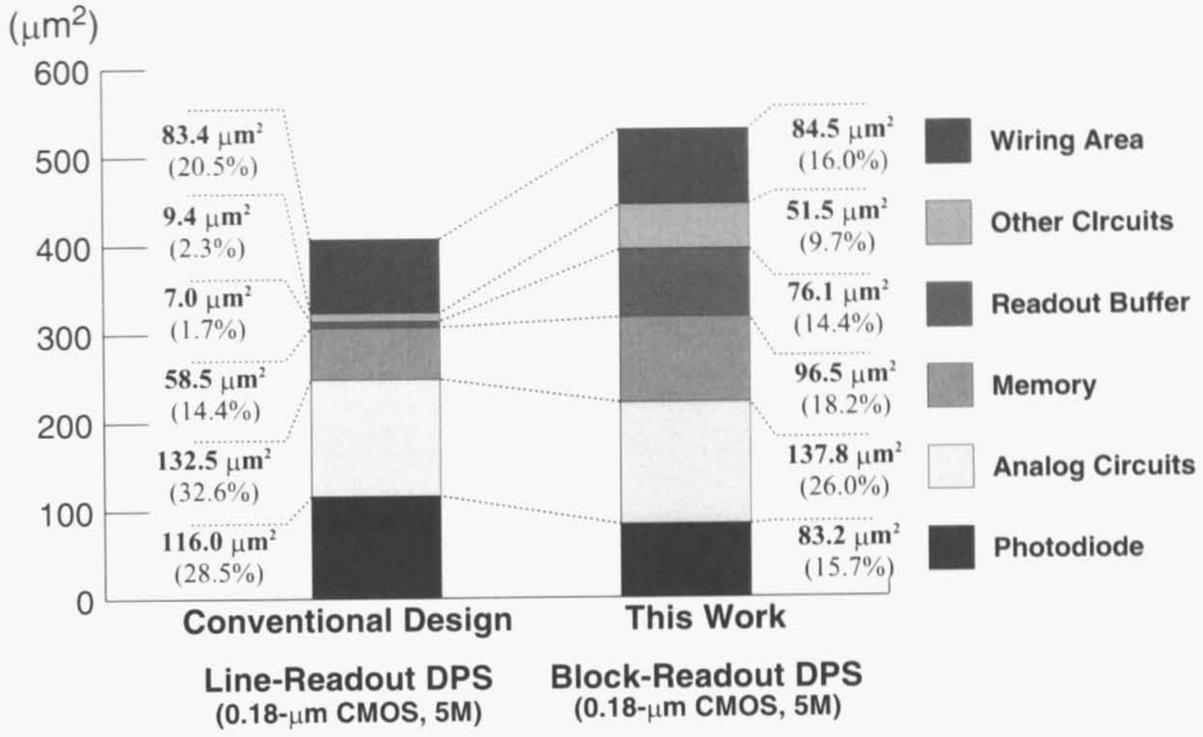


Figure 3.15: Area penalty of block-readout architecture

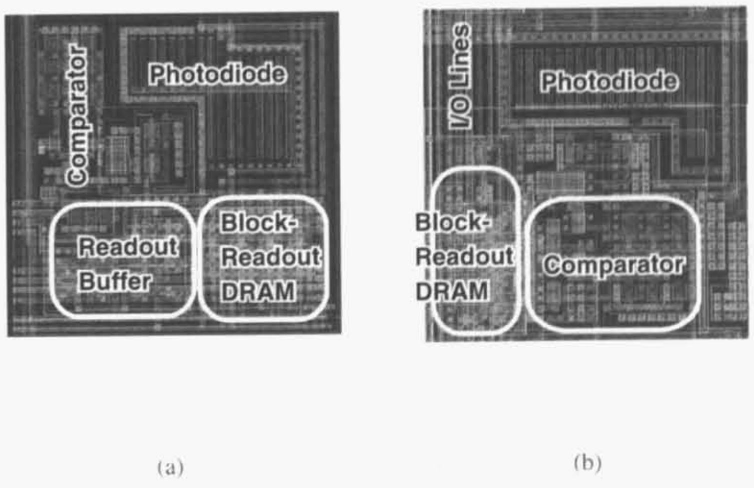


Figure 3.16: Experimental layout of the DPS pixel. (a) Block-readout DPS configuration in 0.18μm 1P5M CMOS technology; (b) conventional line-readout DPS configuration in the same 0.18μm technology.