Pyramid Architecture for 3840 X 2160 Quad Full High Definition 30 Frames/s Video Acquisition

Wei-Min Chao and Liang-Gee Chen, *Fellow, IEEE*

***Abstract*—Image pipeline processing is crucial to generating high quality images in applications using complementary metal-oxide-semiconductor (CMOS)/charge-coupled device sensors. The on-chip line buffer normally dominates the total area and power dissipation due to the needed filter window buffering. As image resolution and filter support increase, the area and power requirement increase accordingly. This paper presents a novel pyramid architecture to efficiently process a system that the image pipeline is between an image sensor and video coding engine. By utilizing the features of the pyramid structure and block-based video/image encoders, the proposed architecture is scalable from low to high image resolution and filter size. The input image is first partitioned into floors of tiles to reduce the frame line buffer. Two computing schemes, immediate result reuse and vertical snack scan, are utilized to reduce the overlapping redundant computations. A 90 nm CMOS chip design with 7 × 5 filter support for 3840 × 2160 quad full high definition video at 30 frames/s is designed to demonstrate the performance of power and area efficiency. Compared with traditional architectures with frame line buffers, the proposed design has shown that the power consumption is reduced by 25% to 108 mW from 145 mW. The chip area is reduced by 65% to 309 K from 888 K logic gates. The external memory bandwidth increases to 8286 Mbit/s from 5972 Mbit/s for YUV4:2:0, from 7963 Mbit/s for YUV4:2:2, and is reduced by 30% from 11 944 Mbit/s for YUV4:4:4.**

***Index Terms*—Camcorder, digital image processing pipeline, digital still camera, high definition video, VLSI architecture.**

I. Introduction

THE demands of image and video acquisition are increasing in technologies such as digital still cameras, digital camcorders, and security internet protocol (IP) cameras. Furthermore, advances in the integration of image and video-acquisition devices with consumer and information technology products, such as mobile phones, notebook computers, monitors, and interactive game players, have facilitated the development of new applications.

For these applications, the image pipeline processes incoming pixels from a sensor and then feeds pixels to a video or image encoder for storage or network transmission. It is the core technology that restores vivid videos for the subsequent video compression codec and intelligent video processing [1]–[3]. The main aim of the image pipeline is sensor noise reduction (NR), color interpolation (CI), gamma correction, color conversion, and edge enhancement (EE) [4]–[8]. Although no universal algorithm exists, several image processing algorithms are linked together to achieve these specific goals. The technological trend for the image-processing pipeline is that resolution has increased to more than 10 Mpixels. Although the shot-to-shot performance is being shortened for capturing digital still images, the frame rate of the full view increases more than 30 frames/s to achieve a quick response of two-way video interactions. Image and video devices must be sufficiently compact such that they can be integrated into existing appliances. Low power dissipation is a requirement for portable and mobile devices. Image quality has been improved continuously via complex algorithms. Linear filters are now replaced with nonlinear filters that adapt to image contents. Filter window size increases to enhance image quality. Pixel depth has increased to more than 8 bits to achieve a high dynamic range and wide gamut.

Unfortunately, heterogeneous algorithms are linked and the inconsistent data flow of these algorithms makes efficient implementation difficult. Different architectures have been proposed to solve these issues. For an embedded system, single instruction multiple data and very long instruction word digital signal processor architectures [9]–[12] were developed to carry out the image processing pipeline. A raw image is normally retrieved and stored in external memory, and then one processor or multiple processors read these raw images for subsequent algorithmic processing. The computing load and bandwidth to and from off-chip memory become bottlenecks. With a single core, a long shot-to-shot time is required and is not suitable for high-frame-rate video acquisition. Over time, advanced silicon technology drives integration of more processor cores [11] to carry out full high definition resolution at 30 frame/s. Although these architectures are flexible, the chip area is huge and power efficiency is poor compared to those of hardwired solutions.

The dedicated hardware engines with the frame line buffer-based architecture [13]–[16] are widely adopted to achieve real-time performance and acceptable quality. Fig. 1 shows a typical design for two links of the 3×3 2-D filter algorithms. Several two-port static random access memories (SRAMs) are used as line buffers to store incoming scan-line pixels. A window of pixels is then transferred concurrently to the register array of the filter core. However, when resolution increases and advanced algorithms need a large filter window,

Manuscript received October 29, 2009; revised March 12, 2010; accepted April 15, 2010. Date of publication September 20, 2010; date of current version November 5, 2010.

The authors are with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: hydra@video.ee.ntu.edu.tw; lgchen@video.ee.ntu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2010.2077770

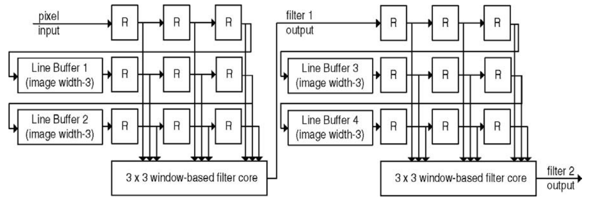
pixel input
R
Line Buffer 1 (image width-3)
Line Buffer 2 (image width-3)
3 x 3 window-based filter core
filter 1 output
Line Buffer 3 (image width-3)
Line Buffer 4 (image width-3)
filter 2 output

Fig. 1. Frame line buffer-based filter architecture.

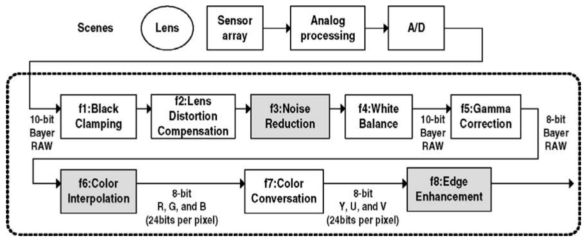

Fig. 2. Image pipeline for video acquisition.

dozens of line buffers are necessary and the storage amount dominates the total chip area and power consumption. A multiple pixel processor architecture [17] was developed for the image-processing pipeline. Although multiple processing cores can have massive computing capability, power and area scalability are problems, such that only 256 × 256 pixels of resolution is supported. A reconfigurable architecture [18] was proposed for the image-processing pipeline. Based on a detailed analysis of the characteristics of algorithms, special types of elementary computing units have been implemented and coarsely integrated into the processor architecture. Thus, computing efficiency can be improved considerably. However, the problems of a huge memory bandwidth and many frame line buffers are still not resolved, especially for high-resolution images and large filter kernel size.

This paper proposed an efficient architecture for a capturing system including image sensor, image pipeline, and video/image encoder, such as Joint Photographic Experts Group (JPEG) [19] and H.264 [20]. By analyzing the data flow and structure of these units, the proposed pyramid architecture of image pipeline is a scalable design without large on-chip buffer and power dissipation while the image resolution and filter size increases. A design example for 3840 × 2160 quad full high definition (QFHD) at 30 frames/s videos is designed in United Microelectronics Corporation (UMC) 90 nm complementary metal-oxide-semiconductor (CMOS) standard performance and regular voltage threshold (SP-RVT) technology. Compared to a typical architecture, simulation results show that power consumption is reduced to 108 mW from 145 mW. The equivalent area is reduced to 309 K from 888 K logic gates. The external memory bandwidth increases to 8286 Mbit/s from 5972 Mbit/s for YUV4:2:0, from 7963 Mbit/s for YUV4:2:2, and reduced by 30% from 11 944 Mbit/s for YUV4:4:4.

The rest of this paper is organized as follows. Section II presents the fundamental algorithms of the image pipeline. Section III describes the proposed architecture of the pyramid image signal processor. Next, Section IV summarizes implementation results and comparisons. Finally, a conclusion is drawn in Section V.

II. Fundamental Algorithms

Fig. 2 is a typical image pipeline [8] for video acquisition from a CMOS image sensor. Heterogeneous algorithms are linked together to achieve transformation and enhance visual quality from the Bayer raw color space to the YUV color space. These processes are classified into two categories

TABLE I
Image Pipeline Algorithms

Filter No.	Processing Unit	Operation Type
f1	Black level adjustment	Per-pixel
f2	Lens shading compensation	Per-pixel
f3	NR	Per-block
f4	White balance	Per-pixel
f5	Gamma correction	Per-pixel
f6	CI	Per-block
f7	Color conversion	Per-pixel
f8	EE	Per-block

according to the basic processing unit. One process is the per-pixel operation and the other is the per-block operation (Table I). The image pipeline can be formalized as (1c). For one pixel output y_i, the operand of the per-pixel function is the co-located pixel, x_i, and the operands of the per-block function are the window Ω of pixels around the co-located center pixel x_i. The data flow of per-pixel functions is consistent and efficient for a pipelined hardware architecture. However, applying this intuitive implementation method for the links of per-block functions is difficult. These combined 2-D adaptive filter operations are NR, CI, and EE (gray in Fig. 2). Because no international standard is defined for image pipeline algorithms, there may be simple algorithms which degrade image quality. Hence, we present modified NR, CI, and EE algorithms in our implementation. In Section III, we then present our architecture to carry out these 2-D nonlinear filters to achieve good visual quality

$$Y = f_8(f_7(f_6(f_5(f_4(f_3(f_2(f_1(X)))))))) \tag{1a}$$

$$y_i = f_k(x_i, \Omega_i), \text{ where } k \text{ is 3, 6, and 8} \tag{1b}$$

$$y_i = f_k(x_i), \text{ where } k \text{ is 1, 2, 4, 5, and 7.} \tag{1c}$$

A. NR Filter

While photons are captured and converted into digital signals by the image sensor, shot noise, thermal noise, readout noise, reset noise, and fixed pattern noise are combined with the signals [21]. These noises are typically considered a mathematical mixture of Gaussian noise and impulse noise [22]. Nonlinear approaches have proven very attractive for accurate filtering of image data, because nonlinear algorithms can distinguish between unwanted noise and image signals based on the image characteristics. Here, a bilateral filter [23] and [24] combined with a median filter is implemented in

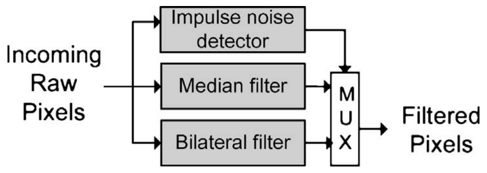

Fig. 3. Soft switch of the median filter and bilateral filter.

the Bayer domain to efficiently remove Gaussian noise and impulse noise. Fig. 3 shows the data flow. Let x_j be the pixel intensity at position j, Ω be the position set of the same color components within filter window i, y_i be the filtered result at position i, and σ_s and σ_r are scaling factors. The bilateral filter is shown in (3). This filter simultaneously weights pixels based on distance from the center pixel and distance from pixel intensity. The effect is reduced Gaussian noise by smoothening images while preserving edges based on both their geometric closeness and photometric similarity. Although an ordinary median filter can remove the impulse noise efficiently, it also distorts an image and lose fine details. To avoid altering a significant amount of the original information in an image, a switch mechanism is used to select the median filter or bilateral filter for incoming pixels instead of using the cascade of the bilateral and median filter. If the intensity of the center pixel is larger or smaller than the average intensity of surrounding pixels over a specific threshold, the median pixel intensity of the window is selected. Otherwise, the bilateral result is chosen

$$y_i = median_{j \subset \Omega_i}(x_j) \tag{2}$$

$$y_i = \frac{\sum_{j \subset \Omega_i} e^{-\frac{|j-i|}{2\sigma_s^2}} e^{-\frac{|x_j - x_i|}{2\sigma_r^2}} x_j}{\sum_{j \subset \Omega_i} e^{-\frac{|j-i|}{2\sigma_s^2}} e^{-\frac{|x_j - x_i|}{2\sigma_r^2}}}. \tag{3}$$

B. CI Filter

To produce a color image, a pixel must have at least three color components. Due to cost-effective considerations, most image sensors are monochromatic and have a Bayer pattern of color filter array [25] for capturing alternative three red-green-blue (RGB) primary colors. Based on this mosaic-like gray-scale image, the two missing color components have to be reconstructed from adjacent pixels using algorithms so called CI. Many CI algorithms have been developed [26]–[28]. Most adapt the spatial characteristics of a local area. They detect edges according to image gradients. Missing color components are then assigned by weighted averages of neighboring pixels with the similar image characteristics. This paper adopted [28] owing to the detection of comprehensive directions (N, S, E, W, NW, SW, NE, and SE). Although computational complexity is high, this algorithm achieves better mean square error than gradient solutions.

C. EE Filter

In addition to optical blur and sensor blur, the aforementioned NR and CI algorithms, altering pixels by weighted average of pixels in a filter window, may blur an image, resulting in image degradation. Therefore, the EE filter is applied to increase the sharpness during the last processing. The EE filter has two stages, as shown in (4a). First, edge intensities are obtained using a high-pass filter. High-pass filtering subtracts an unsharp mask from an image [29]. An unsharp mask is a blurred image produced by spatially filtering an original image with a Gaussian low-pass filter, k_j. Second, EE identifies edge boundaries according to the predefined threshold, and then enhances the contrast in the identified area immediately around the edges by creating subtle bright and dark highlights

$$z_i = x_i - \sum_{j \subset \Omega_i} k_j x_j \tag{4a}$$

$$y_i = \begin{cases} x_i + \alpha z_i & \text{if } z_i > \text{threshold} \\ x_i & \text{otherwise.} \end{cases} \tag{4b}$$

III. Pyramid Architecture Design of Image Pipeline for Video Encoding

A. Data Flow Analysis

Fig. 4(a) shows a data flow of a general subsystem in digital still camera, digital camcorder, and security IP camera applications. A buffer is needed to reorder line scan pixels to block scan ones. The size of this buffer is generally two macroblock rows of pixels (W $\times$ 16 $\times$ 2 $\times$ 1.5 bytes) for double buffering, and an external dynamic random access memory (DRAM) is usually adopted for cost-efficiency. Most CMOS/charge-coupled device (CCD) sensors adopted Bayer RGB format and a color component is used to represent a pixel. Besides, to represent visual linear intensity, a gamma of 0.45 is usually adopted in the image pipeline to encode linear RGB values into nonlinear ones. It means to generate three 8-bit color components for a video encoder, it is necessary to acquire a 10-bit component in Bayer RGB color space. According to JPEG [19] and H.264 [20] image and video coding standards, U and V component can be sub-sampled to 4:2:0 for acceptable chrominance representation or up to 4:4:4 for high quality. Because data amount becomes larger from 10 bits to 12 bits or 24 bits/pixel in average after the image pipeline, it is beneficial for line buffer optimization if we moved the reorder buffer before the image pipeline. Hence, we proposed an efficient architecture for the image processing pipeline for Fig. 4(b). In addition, since a raster-to-block conversation is a must, the requirements for inefficient block-style access to the external memory are similar in Fig. 4(a) and (b). In Section IV-B, we compare the bandwidth impacts according to this data flow proposal.

B. Image Decomposition into Tiles in a Pyramid Structure

Fig. 5 shows the data partition and flows of 2-D images in the pyramid architecture. The first floor is the source image, which is divided into tiles. Four kinds of tile sizes are defined in Table II for the same floor. Each tile is processed by the raster scan order of the same floor. Pixels inside one tile in the lowest floor are received by the tile processing element (TPE), which outputs filtered results concurrently to the top floor. Each TPE applies one 2-D filter (f3, f6, or f8) to pixels

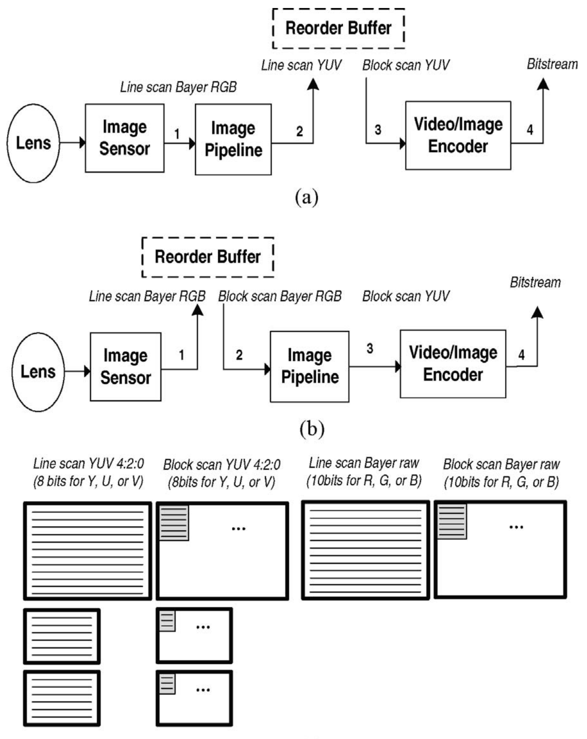

Fig. 4. Data flow in target applications. (a) Data flow of a general subsystem. (b) Data flow after block-scan reorder. (c) Access types of (a) and (b) flows.

TABLE II
TILE SIZE IN VARIOUS REGIONS AND FLOORS

	Floor 1	Floor 2	Floor 3	Floor 4
Region 1	$(s+3m)\times(t+3n)$	$(s+2m)\times(t+2n)$	$(s+m)\times(t+n)$	st
Region 2	$s(t+3n)$	$s(t+2n)$	$s(t+n)$	st
Region 3	$(s+3m)t$	$(s+2m)t$	$(s+m)t$	st
Region 4	st	st	st	st

within one tile. In addition to TPEs, several pixel processing elements (PPEs) exist, each handling one per pixel operation (f1, f2, f4, f5, and f7). The general filter design replaces each pixel value in an image with the filter result of its neighbors and itself. The lower tiles have to be larger than the upper ones. If the expected size of the upmost tile is s by t and the filter kernel size is $(m + 1) \times (n + 1)$, the co-located tile size in the lowest floor will be $(s + 3m) \times (t + 3n)$. By considering overlapping areas, images are partitioned into four regions in the lowest floor (source image). On the lowest floor, the left-topmost tile is $(s + 3m) \times (t + 3n)$ pixels. Except for the top-leftmost tile, all topmost tiles are $s \times (t + 3n)$ pixels and the leftmost tiles are $(s + 3m) \times t$ pixels. The other tiles are $s \times t$ pixel size. With these tile partitions, the frame line buffer can be reduced and is independent of image width. This makes the image pipeline scalable for different image resolutions.

C. Immediate Result Reuse

Although tile partitioning can reduce the size of line buffers, the overlapping areas of 2-D filter operations across tiles results in poor hardware utilization. For example of the four

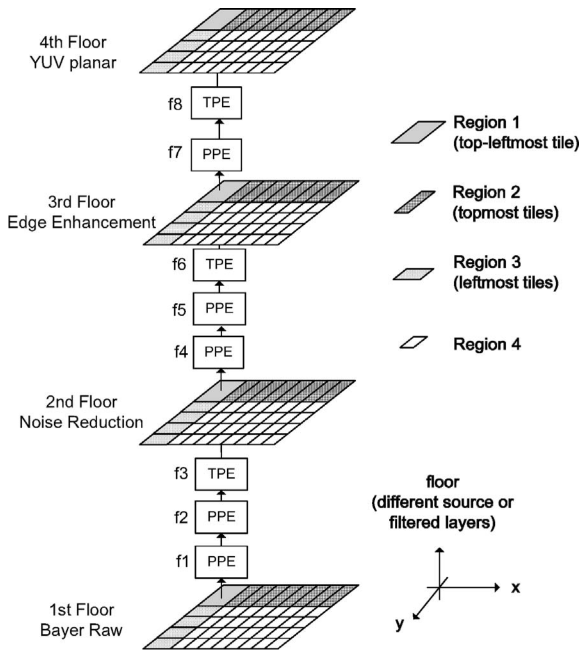
4th Floor YUV planar
f8 TPE
f7 PPE
3rd Floor Edge Enhancement
f6 TPE
f5 PPE
f4 PPE
2nd Floor Noise Reduction
f3 TPE
f2 PPE
f1 PPE
1st Floor Bayer Raw
Region 1 (top-leftmost tile)
Region 2 (topmost tiles)
Region 3 (leftmost tiles)
Region 4
floor (different source or filtered layers)
x
y

Fig. 5. Data partition and flow of the pyramid architecture.

floors of the pyramid architecture with a 7×5 filter kernel size (Fig. 6), the 16×16 output pixels need 34×28 pixels on the first floor. This makes hardware utilization inefficient and 3.72 times input image bandwidth is required.

The proposed immediate result reuse (IRR) data flow is based on the reuse of filter results among floors in the horizontal and vertical directions (Fig. 7). To simplify the representation, Fig. 8 shows a 1-D example that the kernel size is 3 pixels and the tile width is 4 pixels. The pixels on the top floor are convoluted results of the coefficients of the filter kernel and the three pixels around the co-located position on the bottom floor. Three-directional arrows are utilized to indicate one convolution. We assume N is the number of floors in the pyramid, the length L_j on the jth floor is the expected output size plus the overlapping regions P in gray. If the rightmost m pixels of filtering results of this tile are kept and reused for the next tile, R pixels in backslash can be propagated to the top floor directly without convolution again.

Fig. 9 shows the computational order of the 1-D pyramid architecture after applying the IRR data flow. The computation of adjacent filtering results can be eliminated after the leftmost tiles. Exact s pixels of the tile on each floor are calculated so that input pixel throughput is the same as output pixel throughput, and the hardware utilization of the filter core is 100% for all the non-leftmost tiles.

The IRR can be extended to the 2-D pyramid architecture (Fig. 10). If the expected image size is $W \times H$ pixels, the input image resolution must be $(W + 3m) \times (H + 3n)$ pixels. The leftmost tiles and topmost tiles become larger than $s \times t$ from the ground floor to the upmost floor. Other tiles retain the size of $s \times t$ pixels, which is independent of the number of floors

$$L_j = s + P \tag{5a}$$

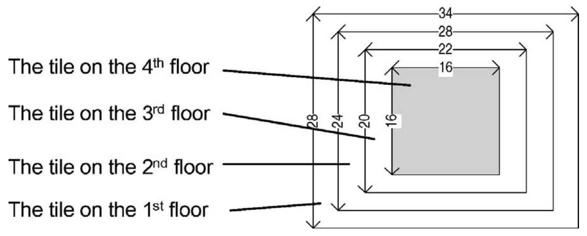
The tile on the 4th floor
The tile on the 3rd floor
The tile on the 2nd floor
The tile on the 1st floor
34
28
22
16
28
24
20
16

Fig. 6. Co-located tiles of the pyramid of various floors.

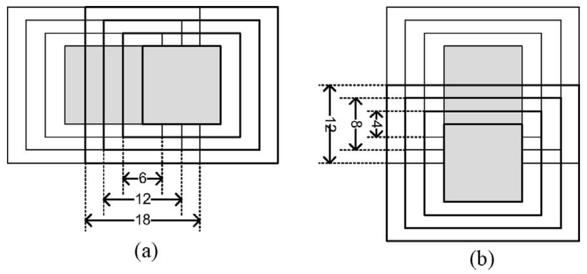

Fig. 7. (a) Overlapped regions between horizontal neighbor tiles. (b) Overlapped regions between vertical neighbor tiles.

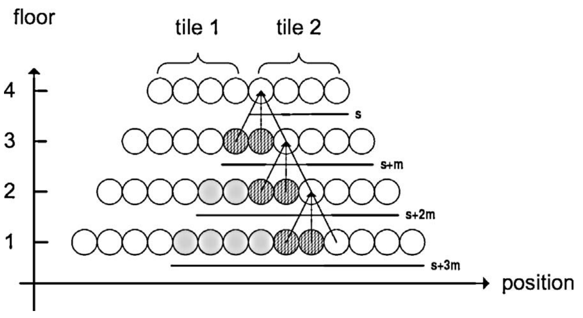

Fig. 8. IRR of the 1-D pyramid.

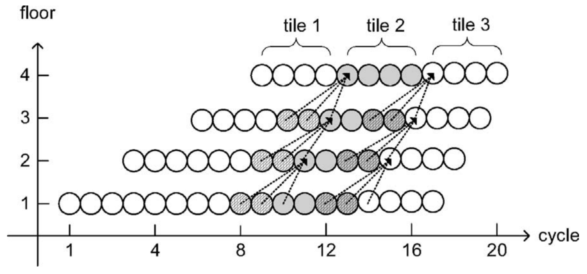

Fig. 9. Computation order of the 1-D pyramid architecture.

$$P = (N - j)m \tag{5b}$$

$$R = (N - 1)m. \tag{5c}$$

D. System Architecture

Fig. 11 shows a system block diagram with the pyramid tile-based image processor (PTISP). The sensor interface obtains pixels from the image sensor and continuously feeds them into the external memory via the memory controller. The required external memory size is one source image frame and $(3 \times n)$ lines of pixels for the IRR. Once one row of tiles in an image is ready in the external memory, the PTISP loads these tiles for image processing one by one. Inside the PTISP, the PPEs handle the pixel-based image processing algorithms and the TPEs handle the block-based image processing algorithms. The sequencer pre-fetches source image pixels or the immediate filtering results of lower tiles from the external memory and feeds data to tile elements. If the 16×16 tile size is chosen, the output tile can be fed into a typical video encoder for video compression without any frame reorder buffer. Restated, the proposed architecture converts raster-scan pixels into block-scan pixels. This is efficient for typical video codec architectures. Furthermore, one 32-bit advanced high-performance bus (AHB) slave interface is employed for memory-mapped register settings and one 32-bit AHB master interface with embedded control logics is utilized for data transfer. After initial settings, the AHB master can generate memory addresses automatically for reading and writing without the host processor. Finally, a 7×5 pixel kernel size is chosen for the NR, CI, and EE filters, and the output tile size is 16×16.

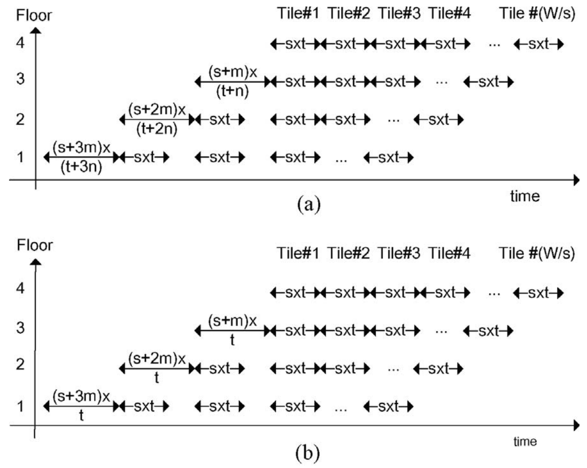

Fig. 10. Computation order of the 2-D pyramid architecture. (a) Topmost row of tiles. (b) Other rows of tiles.

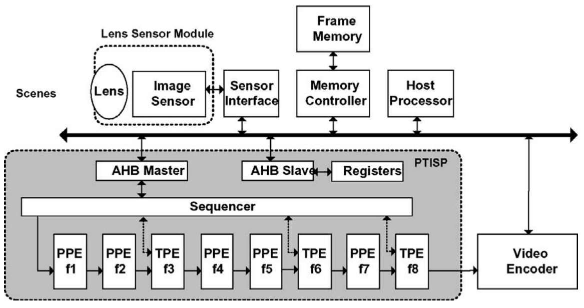

Fig. 11. Block diagram of the video-acquisition system and pyramid tile-based image signal processor.

E. Tile Processing Element

Fig. 12 shows the proposed architecture for tile filtering with the IRR data flow. Initially, pixels in the filter window must be loaded into the window registers before the filter core performs logic operations. If the filter window moves continuously, most registers can be reused; otherwise, all registers must be reloaded and the filter core should wait until the data are ready. As the image is divided into tiles, many break points exists during the filter window moving according to the typical line scan. When the direction breaks frequently, pixel registers to filter cores are reloaded frequently. This results in the poor hardware utilization. Therefore, this paper uses a novel vertical snack scan (VSS) for continuous filtering. Fig. 13 shows the directions and break points in different schemes. The frame line scan will encounter H break points. A similar line scan after tile partition will be $H \times (W/s)$ times, resulting in poor data reuse and hardware utilization. Via the proposed VSS, the number of break points is reduced to H/t times.

The address generator (AG) controls the memory access and TPE data flow. The addressing is in circular order, such that the overlapping pixels of neighbor tiles can be reused. The tile element has two phases. One is the loading phase of the next tile, and the other phase is the filtering phase of this tile. Except for the leftmost tiles of rows, these two phases function concurrently and are pipelined in one tile unit. During the loading phase, pixels come from the source raw pixels, immediate results loaded by the sequencer, and outputs of the PPE on the same floor. The AG places incoming pixels with control signals, including vertical synchronization, horizontal synchronization, start of tile, and pixel validity, to the corresponding position of the tile buffers. During the filtering phase, the AG generates addresses for the tile buffers to output pixels to the active registers and shadow registers, and outputs control signals to the other TPE or PPE on the top floor. Notably, $(t + 2)$ two-port register file SRAMs are used to store tile pixels. Fig. 14 shows the memory organization and access. Every 8 pixel columns are grouped as one strip. Inside one strip, eight SRAMs store one pixel column individually. For the leftmost tile of one tile row, the tile pixels are loaded into the entire tile buffer and then read out for filtering without a tile pipeline to save the buffer size. For other tiles, the reading and writing of strips occur concurrently. In this design, $(16 + 3 * 4) * 5$ words are sufficient, such that the overlapping regions will not be overwritten by the next neighbor tile. The array of active registers is utilized as an input window for the filter core. The array of shadow registers is employed for seamless window movement during transition from one column to another column. Within the active and shadow arrays, pixels are displaced in three directions: upward, downward, and leftward. For even columns (0, 2, 4, and so on) in one tile, the active and shadow arrays are simultaneously shifted one position up every cycle. For odd columns (1, 3, 4, and so on) in one tile, the active and shadow arrays are simultaneously shifted one position down each cycle. Once the arrays reach the top of the even columns or the bottom of odd columns, the active and shadow arrays simultaneously shifted one position left. In addition, the start position of next odd or even column is t+1 pixels against the topmost or bottommost position, respectively.

Fig. 15 shows an example of this VSS. For simplicity, 6×6 tile size and 3×3 filter kernel size are chosen. The solid rectangles indicate pixels stored in the active array and dotted-line rectangles indicate pixels stored in the shadow array. At a given clock cycle, say, $c = 0$, the active array moves downward

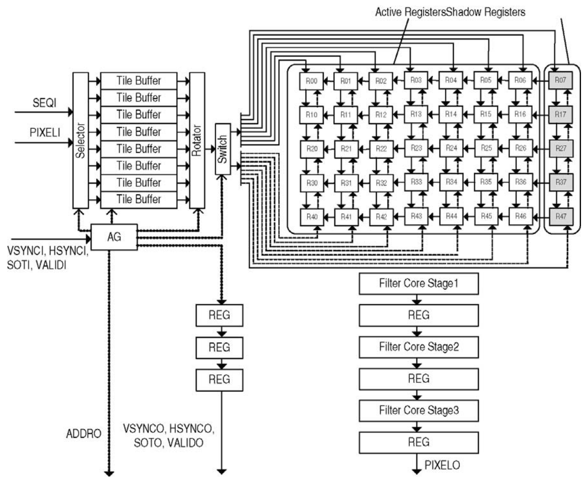

Fig. 12. TPE. Considering filter kernel size is 7×5 and tile size is 16×16.

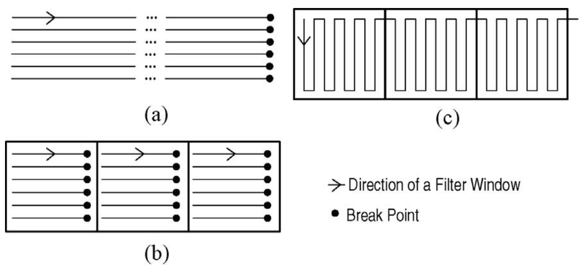

Fig. 13. (a) Frame line scan. (b) Line scan with tile partitions. (c) VSS with tile partitions.

[Fig. 15(a)]. At $c = 6$, the active array reaches the bottom, and at $c = 7$, the registers in the shadow array shift to the left such that the active array contains pixels ready for the filter core. Five pixels are fed into the filter core from the 4-pixel distance from the bottom of each new odd column and 4-pixel distance from the top of each new even column. Using this flow, active registers are ready for filter cores each cycle such that the hardware utilization is 100%. Various filter cores have been designed for the NR, CI, and EE. These cores are dedicatedly pipelined in several stages to achieve a high frequency and throughput is 1 pixel/cycle.

F. Pixel Processing Element

The PPE performs per-pixel operations. Fig. 16 shows a generic architecture. Different filter cores are implemented for black level adjustment, lens shading compensation, white balance, color correction, gamma correction, and color conversion. These elements are designed in a pixel pipeline fashion. Throughput is 1 pixel/cycle.

IV. Implementation and Comparisons

In this section, the proposed architecture is implemented and compared with the frame line-based image signal processor (FLISP). The same algorithms and three links of 2-D filters with 7×5 filter kernel size are chosen to achieve the same quality and performance. Therefore, 12 frame line buffers exist in this FLISP.

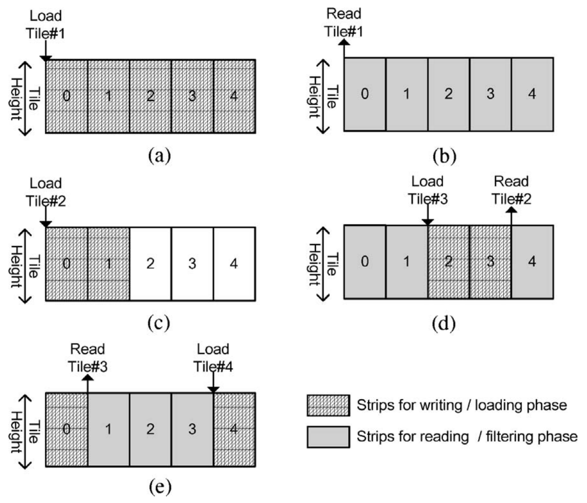

Fig. 14. Memory organization and access of tile buffers. (a) Load first tile. (b) Read first tile for processing. (c) Load second tile. (d) Load third tile and read second tile for processing. (e) Load fourth tile and read third tile for processing.

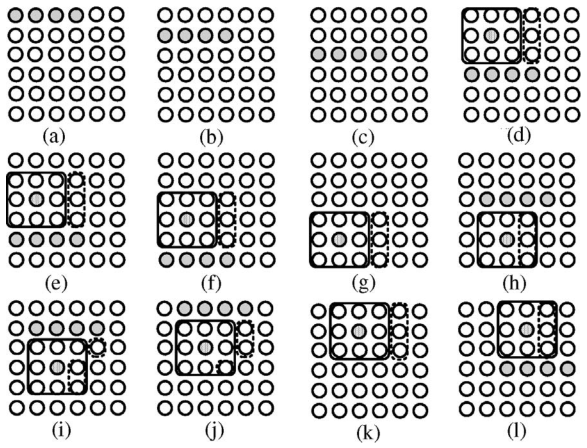

Fig. 15. Data flow of the TPE.

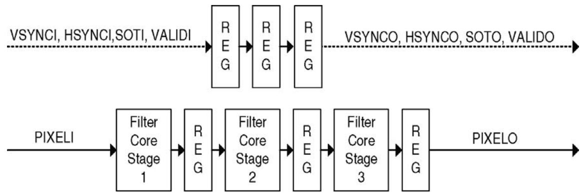

Fig. 16. Pixel processing element.

A. Hardware Utilization

Table III shows the hardware core utilization for various regions in a frame. Table IV shows the processing cycles and utilization of all tiles in an image. Utilization is the number of cycles of output pixels divided by the number of cycles of input pixels and the input pixel throughput is 1 pixel/cycle. Via the proposed VSS of the filter core, the drawbacks associated with tile partitions are eliminated and hardware utilization remains high.

TABLE III
UTILIZATIONS OF REGIONS IN THE PYRAMID ARCHITECTURE

	Output Pixels/Input Pixels
Region 1	(s*t)/[(s+3m)*(t+3n)]
Region 2	(s*t)/[s*(t+3n)]
Region 3	(s*t)/[(s+3m)*t]
Region 4	(s*t)/(s*t)

TABLE IV
AVERAGE UTILIZATIONS FOR VARIOUS RESOLUTIONS

	QFHD	FHD	HD	VGA
Width	3840	1920	1280	640
Height	2160	1080	720	480
No. of tiles in Region 1	1	1	1	1
No. of tiles in Region 2	239	119	79	39
No. of tiles in Region 3	134	67	44	29
No. of tiles in Region 4	31 787	7795	3397	1092
Average utilization	98.67%	97.35%	96.03%	93.58%

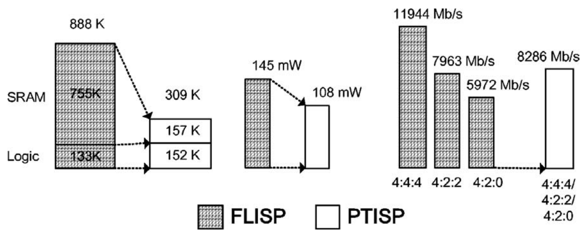

Fig. 17. Area, power dissipation, and bandwidth of the FLISP and PTISP based on the UMC 90 nm CMOS SP-RVT process.

B. Memory Bandwidth

By considering a general camcorder application [1], [3], a video encoder will be cascaded after the image pipeline and one or several DRAMs are used as frame buffers. This scenario is adopted to compare the PTISP and FLISP architectures for QFHD videos running at 30 frames/s. Table V shows the system bandwidth of the PTISP to the external memory; Table VI depicts that of the FLISP. For video coding standards such as H.264 [20], input data must be ordered in 16×16 pixel blocks. The FLISP architecture should have an explicit reorder procedure from the raster scan to the block scan. Thus, the FLISP needs two frames of memory access between the image pipeline and video encoder. However, the proposed PTISP internally do the raster to block scan if the 16×16 pixel tile is selected. Therefore, the FLISP retains the same bandwidth requirement while the chrominance resolution grows. In the case of YUV4:2:0 and YUV4:2:2, the bandwidth of the PTISP is more than those of FLISP. For the high quality YUV4:4:4 format, the proposed PTISP is less than FLISP.

The vertical access is inefficient to DRAM memory; the sequencer of the PTISP packs three TPE vertical reuse data into a burst write and read of continuous addresses for each tile. Thus, the PTISP has additional $2 \times (W/16) \times (H/16)$ times of transactions for the vertical data reuse per frame for all cases. It increases 12.5% transactions for a pyramid tile-partitioned frame compared to a block-partitioned frame.

TABLE V
BANDWIDTH OF THE PTISP FOR QFHD VIDEOS AT 30 FRAMES/S

Type	Direction	Data Format	Bits/Frame	Bandwidth (Mbits/s)
Input source image	SENOR to EXT. MEM	10 bit Bayer raw	(W+3 m)*(H+3n)*10 bits	2520
Input source image	EXT. MEM to PTISP SEQ	10 bit Bayer raw	(W+3 m)*(H+3n)*10 bits	2520
IRR of TPE 1	PTISP SEQ to EXT.MEM	10 bit lens compensated pixels (f2)	(W+3 m)*(H/t)*n*10 bits	625
IRR of TPE 1	EXT. MEM to PTISP SEQ	10 bit lens compensated pixels (f2)	(W+3 m)*(H/t)*n*10 bits	625
IRR of TPE 2	PTISP SEQ to EXT.MEM	8 bit gamma corrected pixels (f5)	(W+2 m)*(H/t)*n*8 bits	500
IRR of TPE 2	EXT. MEM to PTISP SEQ	8 bit gamma corrected pixels (f5)	(W+2 m)*(H/t)*n*8 bits	500
IRR of TPE 3	PTISP SEQ to EXT.MEM	8 bit Luma pixels (f7)	(W+m)*(H/t)*n*8 bits	499
IRR of TPE 3	EXT. MEM to PTISP SEQ	8 bit Luma pixels (f7)	(W+m)*(H/t)*n*8 bits	499
Total				8286

TABLE VI
BANDWIDTH OF THE FLISP FOR QFHD VIDEOS AT 30 FRAMES/S

Type	Direction	Data Format	Bits/Frame	Bandwidth (Mbits/s)
Raster scan	FLISP to EXT.MEM	8 bit YUV4:2:0	W*H*8 bits*1.5	2986
Block scan	EXT.MEM to CODEC	8 bit YUV4:2:0	W*H*8 bits*1.5	2986
Total for YUV4:2:0				5972
Raster scan	FLISP to EXT.MEM	8 bit YUV4:2:2	W*H*8 bits*2	3982
Block scan	EXT.MEM to CODEC	8 bit YUV4:2:2	W*H*8 bits*2	3982
Total for YUV4:2:2				7963
Raster scan	FLISP to EXT.MEM	8 bit YUV4:4:4	W*H*8 bits*3	5972
Block scan	EXT.MEM to CODEC	8 bit YUV4:4:4	W*H*8 bits*3	5972
Total for YUV4:4:4				11 944

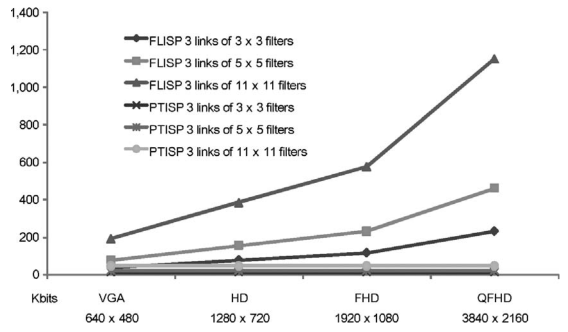

Fig. 18. Memory requirements of the FLISP and PTISP at various resolutions and filter sizes.

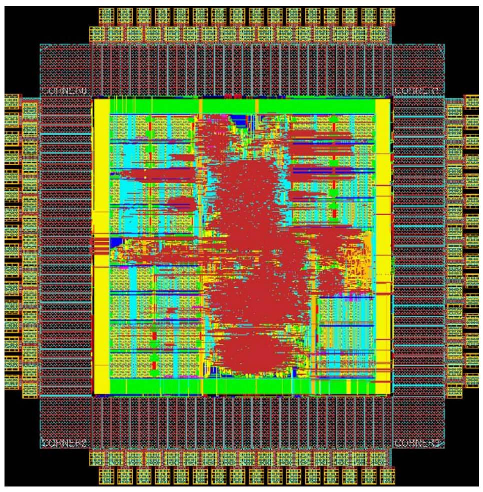

Fig. 19. Chip layout of the PTISP.

TABLE VII
CHIP SPECIFICATIONS

Maximum computation performance	3840×2160 at 30 frames/s
Maximum output data rate	746 M pixels/s
Maximum video-processing capability	3840×2160 (4:4:4) 30frames/s
Hardware utilization	98.67%
Technology	UMC 90 nm SP-RVT
Logic gate (2-input NAND and included SRAM)	309 K
SRAM bits	37.31 K
Working frequency	266 MHz
Power consumption	108 mW at 266 MHz
Core size	1.54 mm^2
Die area	4.50 mm^2

C. Area and Power Efficiency

Fig. 17 shows the equivalent gate count, power consumption, and global bandwidth requirement between the PTISP and FLISP. In the FLISP architecture, SRAMs dominate the whole area of the design. Moreover, the percentage occupied increases as image width increases. Conversely, the SRAM requirement of the PTISP is fixed with little overhead for logic control circuits. The comparison is shown in Fig. 18. Furthermore, the SRAM area is independent of image width, such that the area and power efficiency are much better than those of the FLISP architecture as image resolution increases.

D. Very Large-Scale Integration Implementation and Comparison

Table VII shows the chip specifications, while Fig. 19 shows chip layout. The core/die size is 4.5 mm^2/1.54 mm^2 with the UMC 90 nm CMOS SP-RVT process. Power consumption is 108 mW at 1.0-V and 266 MHz. This chip can process 1920 × 1080 videos at 120 frames/s or 3840 × 2160 videos at 30 frames/s for high-definition video-acquisition devices.

TABLE VIII
IMPLEMENTATION COMPARISONS WITH APPLICATION-SPECIFIC INTEGRATED CIRCUIT SOLUTIONS

	Loinaz *et al.* [13]	Doswald *et al.* [14]	Chen and Chien [18]	PTISP
Video resolution	352×288	1024×1024	1920×1080	3840×2160
2-D filter size	3×3	9×9	3×3	7×5
No. of 2-D filtering	1 (CI)	1 (CI)	1 (NR, CI, or EE)	3 (NR, CI, and EE)
Input data rate	3 MB/s	50 MB/s	115 MB/s	240 MB/s
Output data rate	9 MB/s	120 MB/s	345 MB/s	746 MB/s
Technology	0.8 μm	0.35 μm	0.18 μm	90 nm
Die size	37 mm^2	49 mm^2	7.72 mm^2	4.50 mm^2
Core size	–	–	5 mm^2	1.54 mm^2
Power dissipation	92 mW at 3.3-V	278 mW at 2.5-V	218 mW at 1.8-V	108 mW at 1.0-V

Table VIII shows chip comparisons. Loinaz *et al.* [13] integrated the image sensor and processing pipeline. Notably, NR and EE were not supported and filter kernel size was small. Doswald *et al.*'s [14] is a pure image signal processor for the image sensor. Larger kernel size is supported. However, NR and EE were not supported. Chen and Chien [18] supported NR, CI, or EE, but they work at different time frames, and filter kernel was small. The proposed design provides the highest pixel throughput while filter kernel size is kept large and three links of 2-D filtering works concurrently.

V. CONCLUSION

This paper presented a novel pyramid structure with IRR and VSS computing schemes for the CCD/CMOS image processing pipeline. The proposal can be applied in a system including image sensor, image processing pipeline, and an image/video encoder, such as digital camera, digital camcorder, or IP security camera. This architecture is scalable and independent of image resolution. To demonstrate its performance, an image processor for 3840×2160 QFHD at 30 frames/s videos was designed in the UMC 90 nm CMOS SP-RVT technology. Compared with traditional architectures with frame line buffers, the proposed design reduced the power consumption by 25% to 108 mW from 145 mW. Chip area is reduced by 65% to 309 K from 888 K logic gates. The external memory bandwidth increases to 8286 Mbit/s from 5972 Mbit/s for YUV4:2:0, from 7963 Mbit/s for YUV4:2:2, and is reduced by 30% from 11 944 Mbit/s for YUV4:4:4.

REFERENCES

[1] T. Nakamm, H. Marumori, M. Takahh, and Y. Fujii, "An MPEG-2 CODEC LSI with an audio accelerator for camcorders," *IEEE Trans. Consum. Electron.*, vol. 48, no. 3, pp. 326–327, Aug. 2002.

[2] M. Kuwahara and K. Yoneyama, "A portable camcorder/server for wireless video transmission," *IEEE Trans. Consum. Electron.*, vol. 51, no. 2, pp. 351–355, May 2005.

[3] Y. Hamamoto, K. Koyama, S. Okada, H. Murata, M. Nishikawa, and N. Itii, "Compact full HD digital movie camera with H.264 codec in single-chip," in *Proc. IEEE Int. Symp. Consum. Electron.*, Apr. 2008, pp. 1–3.

[4] G. Sharma and H. J. Trussell, "Digital color imaging," *IEEE Trans. Image Process.*, vol. 6, no. 7, pp. 901–932, Jul. 1997.

[5] J. Adams, K. Parulski, and K. Spaulding, "Color processing in digital cameras," *IEEE Micro*, vol. 18, no. 6, pp. 20–30, Nov. 1998.

[6] R. Ramanath, W. E. Snyder, Y. Yoo, and M. S. Drew, "Color image processing pipeline," *IEEE Signal Process. Mag.*, vol. 25, no. 1, pp. 34–43, Jan. 2005.

[7] W. C. Kao, S. H. Wang, L. Y. Chen, and S. Y. Lin, "Design considerations of color image processing pipeline for digital cameras," *IEEE Trans. Consum. Electron.*, vol. 52, no. 4, pp. 1144–1152, Nov. 2006.

[8] J. Zhou and J. Glotzbach, "Image pipeline tuning for digital cameras," in *Proc. IEEE Int. Symp. Consum. Electron.*, Jun. 2007, pp. 1–4.

[9] L. Lucas, "High speed low cost TM1300 Trimedia enhanced PCI VLIW mediaprocessor," in *Proc. Hot Chips*, vol. 11. Aug. 1999, pp. 111–120.

[10] S. Agarwala, P. Koeppen, T. Anderson, A. Hill, M. Ales, R. Damodaran, L. Nardini, P. Wiley, S. Mullinnix, J. Leach, A. Lell, M. Gill, J. Golston, D. Hoyle, A. Rajagopal, A. Chachad, M. Agarwala, R. Castille, N. Common, J. Apostol, H. Mahmood, M. Krishnan, B. Duc, A. Quang-Dieu, P. Groves, L. Nguyen, N. S. Nagaraj, and R. Simar, "A 600 MHz VLIW DSP," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1532–1544, Nov. 2002.

[11] S. Arakawa, Y. Yamaguchi, S. Akui, Y. Fukuda, H. Sumi, H. Hayashi, M. Igarashi, K. Ito, H. Nagano, M. Imai, and N. Asari, "A 512GOPS fully-programmable digital image processor with full HD 1080p processing capabilities," in *Proc. Dig. Tech. Papers IEEE ISSCC*, Feb. 2008, pp. 312–313.

[12] T. Wada, S. Ishiwata, K. Kimura, K. Nakanishi, M. Sumiyoshi, T. Miyamori, and M. Nakagawa, "A VLIW vector media coprocessor with cascaded SIMD ALUs," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 17, no. 9, pp. 1285–1296, Sep. 2009.

[13] M. Loinaz, K. Singh, A. Blanksby, D. Inglis, K. Azadet, and B. Ackland, "A 200 mW 3.3-V CMOS color camera IC producing 352 × 288 24b video at 30 frames/s," in *Proc. Dig. Tech. Papers IEEE ISSCC*, Feb. 1998, pp. 168–169.

[14] D. Doswald, B. Schreier, S. Oetiker, J. Hafliger, P. Blessing, N. Felber, and W. Fichtner, "A 30 frames/s megapixel real-time CMOS image processor," in *Proc. Dig. Tech. Papers IEEE ISSCC*, Feb. 2000, pp. 232–233.

[15] C. Weerasinghe, W. Li, I. Kharitonenko, M. Nilsson, and S. Twelves, "Novel color processing architecture for digital cameras with CMOS image sensors," *IEEE Trans. Consum. Electron.*, vol. 51, no. 4, pp. 1092–1099, Nov. 2005.

[16] S. C. Hsia, M. H. Chen, and P. S. Tsai, "VLSI implementation of low-power high-quality color interpolation processor for CCD camera," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 14, no. 4, pp. 361–369, Apr. 2006.

[17] A. Gentile, S. Vitabile, L. Verdoscia, and F. Sorbello, "Image processing chain for digital still cameras based on the SIMPil architecture," in *Proc. Int. Conf. Workshops Parallel Process.*, Jun. 2005, pp. 215–222.

[18] J. C. Chen and S. Y. Chien, "CRISP: Coarse-grained reconfigurable image stream processor for digital still cameras and camcorders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 9, pp. 1223–1236, Sep. 2008.

[19] ISO/IEC 10918-1:1994, "Information technology: Digital compression and coding of continuous-tone still images—requirements and guidelines."

[20] T. Wiegand, G. J. Sullivan, G. Gjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[21] G. Cortellazzo, G. A. Mian, and R. Parolari, "Statistical characteristics of granular camera noise," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 6, pp. 536–543, Dec. 1994.

[22] R. Garnett, T. Huegerich, C. Chui, and H. Wenjie, "A universal noise removal algorithm with an impulse detector," *IEEE Trans. Image Process.*, vol. 14, no. 11, pp. 1747–1754, Nov. 2005.

[23] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. IEEE Int. Conf. Comput. Vision*, Jan. 1998, pp. 839–846.

[24] D. Barash, "A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 6, pp. 844–847, Jun. 2002.

[25] B. Bayer, "Color imaging array," U.S. Patent 3 971 065, 1976.

[26] X. Li, B. K. Gunturk, and L. Zhang, "Image demosaicing: A systematic survey," in *Proc. IST/SPIE Conf. Vis. Commun. Image Process.*, vol. 6822. Jan. 2008, pp. 68221J–68221J-15.

[27] W. Lu and Y. Tan, "Color filter array demosaicing: New method and performance measures," *IEEE Trans. Image Process.*, vol. 12. no. 10, pp. 1194–1210, Oct. 2003.

[28] E. Chang, S. Cheung, and D. Y. Pan., "Color filter array recovery using a threshold-based variable number of gradients," in *Proc. SPIE*, vol. 3650. 1999, pp. 36–43.

[29] *Unsharp Masking*. (2009, Apr.) [Online]. Available: http://en.wikipedia.org/wiki/Unsharp_masking.html

Wei-Min Chao was born in Taoyuan, Taiwan, in 1977. He received the B.S. and M.S. degrees from the Department of Electronics Engineering, National Taiwan University, Taipei, Taiwan, in 2000 and 2002, respectively. He is currently pursuing the Ph.D. degree from the Graduate Institute of Electrical Engineering, National Taiwan University.

He was with the Center System-on-Chip Laboratory, Quanta Computer, Inc., Taoyuan, as a Deputy Section Manager from 2002 to 2004, a Section Manager from 2004 to 2007, a Senior Manager from 2008 to 2009, and has been an Associate Director since 2009. His current research interests include video coding algorithms and systems, digital signal processing architecture, very large-scale integration architecture, and system-on-chip design flows.

Mr. Chao received several awards from University/College IC Design Contest, Taiwan, in 2000, the Golden Silicon Award (Macronix) in 2000, the Advanced Chip Award of NSC Chip Implementation Center in 2000, the IC Contest Award in 2001, the IP Contest Award, Taiwan, in 2001, the Golden Silicon Award (Macronix) in 2001, the Master Paper Award of the National Science Council, Taiwan, in 2002, the Long-Term (Acer) Paper Award in 2002, the Young Paper Award of the Chinese Institute of Electrical Engineering in 2002, the Electric Innovation Contest of NTU in 2002, the Advanced Chip Award of NSC Chip Implementation Center in 2003, and the Technology Transfer Outstanding Contribution Award, National Science Council, Taiwan, in 2004.

Liang-Gee Chen (S'84–M'86–SM'94–F'01) was born in Yun-Lin, Taiwan, in 1956. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Cheng Kung University, Tainan, Taiwan, in 1979, 1981, and 1986, respectively.

He was an Instructor from 1981 to 1986, and an Associate Professor from 1986 to 1988 with the Department of Electrical Engineering, National Cheng Kung University. While being in military service from 1987 to 1988, he was an Associate Professor with the Institute of Resource Management, Defense Management College, Taiwan. In 1988, he joined the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan. From 1993 to 1994, he was a Visiting Consultant with the Digital Signal Processing Research Department, AT&T Bell Laboratory, Murray Hill, NJ. In 1997, he was a Visiting Scholar with the Department of Electrical Engineering, University of Washington, Seattle. Currently, he is a Professor with National Taiwan University. Since 2004, he is the Executive Vice President and the General Director of Electronics Research and Service Organization, Industrial Technology Research Institute, Hsinchu, Taiwan. His current research interests include digital signal processing architecture design, video processor design, and video coding system.

Dr. Chen was the General Chairman of the 7th VLSI Computer-Aided Design Symposium. He was the General Chairman of the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He served as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY from 1996 to 2008, and has been an Associate Editor of the IEEE TRANSACTIONS ON VERY LARGE-SCALE INTEGRATION SYSTEMS since 1999. He is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS and an Associate Editor of PROCEEDINGS OF THE IEEE since 2002. He has been an Associate Editor of the *Journal of Circuits, Systems, and Signal Processing* since 1999. He served as a Guest Editor of the *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology* in 2001. He received the Best Paper Award from the ROC Computer Society in 1990 and 1994. From 1991 to 1999, he received Long-Term (Acer) Paper Awards annually. In 1992, he received the Best Paper Award of the 1992 Asia-Pacific Conference on Circuits and Systems in VLSI Design Track. In 1993, he received the Annual Paper Award of the Chinese Engineer Society. In 1996, he received the Outstanding Research Award from NSC, and the Dragon Excellence Award for Acer. He was elected the IEEE Circuits and Systems Distinguished Lecturer from 2001 to 2002. He is a member of the honor society Phi Tan Phi.