PAPER *Special Section on VLSI Design Technology in the Sub-100 nm Era*

VLSI Architecture Study of a Real-Time Scalable Optical Flow Processor for Video Segmentation

Noriyuki MINEGISHI[†,††a)], ***Member***, **Junichi MIYAKOSHI**[†††], ***Student Member***, **Yuki KURODA**[†*], **Tadayoshi KATAGIRI**[†], **Yuki FUKUYAMA**[†††], **Ryo YAMAMOTO**[†††], ***Nonmembers***, **Masayuki MIYAMA**[†], **Kousuke IMAMURA**[†], **Hideo HASHIMOTO**[†], ***and*** **Masahiko YOSHIMOTO**[†††], ***Members***

SUMMARY An optical flow processor architecture is proposed. It offers accuracy and image-size scalability for video segmentation extraction. The Hierarchical Optical flow Estimation (HOE) algorithm [1] is optimized to provide an appropriate bit-length and iteration number to realize VLSI. The proposed processor architecture provides the following features. First, an algorithm-oriented data-path is introduced to execute all necessary processes of optical flow derivation allowing hardware cost minimization. The data-path is designed using 4-SIMD architecture, which enables high-throughput operation. Thereby, it achieves real-time optical flow derivation with 100% pixel density. Second, it has scalable architecture for higher accuracy and higher resolution. A third feature is the CMOS-process compatible on-chip 2-port DRAM for die-area reduction. The proposed processor has performance for CIF 30 fr/s with 189 MHz clock frequency. Its estimated core size is $6.02 \times 5.33\,\text{mm}^2$ with six-metal 90-nm CMOS technology.

key words: *optical flow, processor architecture, video segmentation*

1. Introduction

Practical video segmentation processes are attractive for various applications in vehicle safety systems, robot manipulation systems, and multi-media distribution systems. Several rough video segmentation techniques such as lane detection and overtaking vehicle detection have been reported [2]–[4]. However, demands for more complicated and fine-graded video segmentation continue to increase. Optical flow derivation is a critical process to meet such demands because the granularity of optical flow determines the video segmentation quality. Several optical flow processors have been reported [5]–[10], but all of them can achieve only limited image resolution and frame rates. For example, Ref. [5] could not achieve the performance for CIF resolution video with 30 fr/s. Also the obtained accuracy was not an enough level according to its insufficient memory resource. The value of mean-absolute-error (MAE) which is inverse of accuracy was 18.30 degree at 100% density, and it was still large for many applications.

Manuscript received July 20, 2005.
Manuscript revised October 20, 2005.

[†]The authors are with the Dept. of Electrical & Electronic System, Kanazawa University, Kanazawa-shi, 920-8667 Japan.
[††]The author is also with Information Technology R&D Center, Mitsubishi Electric Corp., Kamakura-shi, 247-8501 Japan.
[†††]The authors are with the Dept. of Computer and Systems Engineering, Kobe University, Kobe-shi, 647-8501 Japan.
[*]Presently, with Hitachi Ltd.
a) E-mail: Minegishi.Noriyuki@aj.MitsubishiElectric.co.jp

DOI: 10.1093/ietele/e89–c.3.230

This paper presents an optical flow processor architecture that allows real-time and higher resolution (CIF 30 fr/s) capability with 100% pixel density and higher accuracy (MAE < 10). A Hierarchical Optical flow Estimation (HOE) algorithm based on existing algorithm [1] was chosen and an appropriate bit-length and iteration number (450 times) were optimized for VLSI architecture. The proposed optical flow processor architecture comprises a common execution element (CE), external memory controller and CPU. A 4-SIMD architecture was adopted for the CE to achieve high-throughput flow derivation with 450 iterations. An on-chip DRAM was introduced to minimize the core size. Moreover, to meet requirement for higher resolution and higher optical flow accuracy, a scalable architecture was developed using multiple CE configuration.

The HOE algorithm and optimization for VLSI implementation are described in Sect. 2. Section 3 addresses proposed optical flow processor architecture, which is followed by conclusion in Sect. 4.

2. Practical Optical Flow Algorithm

2.1 HOE Algorithm

A HOE algorithm based on existing algorithm [1] was chosen and modified to be suitable for VLSI processor architecture. This section describes HOE algorithm and how optimize the bit-length to balance accuracy and hardware cost.

Figure 1 shows the HOE operation flow. The HOE is modified from existing algorithm [1] based on the Horn and Schunck [11] algorithm which uses differential techniques and extracts a higher-precision optical flow with 100% density than other algorithms [12]–[17]. The HOE introduced multi-dimensional gradient filter and hierarchical image creation.

2.1.1 Hierarchical Image Generation

The basic algorithm assumes a "smooth" constraint, namely a neighboring pixel or pixels in a local region move smoothly. Consequently, if a video sequence has large movement, the motion detection might be incorrect. For that reason, a hierarchical image method was adopted to create an image pyramid as illustrated in Fig. 2. The hierarchy level is determined by motion distance and image

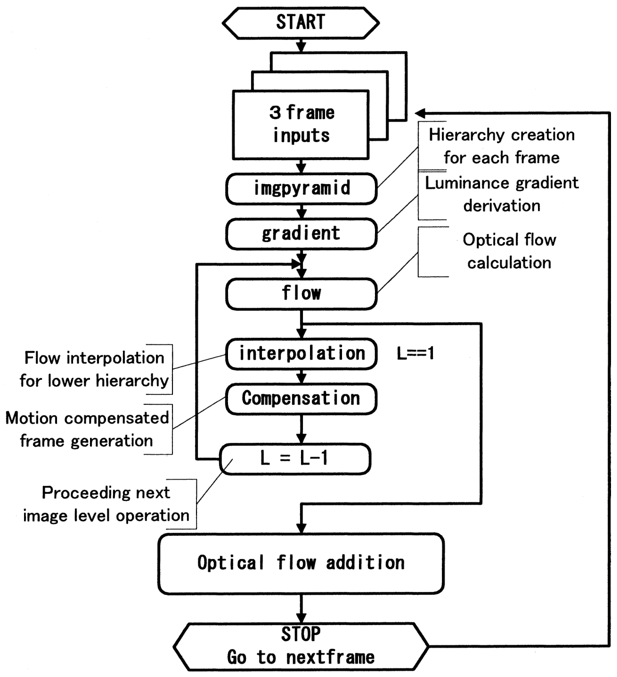

START
3 frame inputs
Hierarchy creation for each frame
imgpyramid
Luminance gradient derivation
gradient
Optical flow calculation
flow
Flow interpolation for lower hierarchy
interpolation
L==1
Motion compensated frame generation
Compensation
L = L-1
Proceeding next image level operation
Optical flow addition
STOP
Go to nextframe

Fig. 1 Operation flow of HOE.

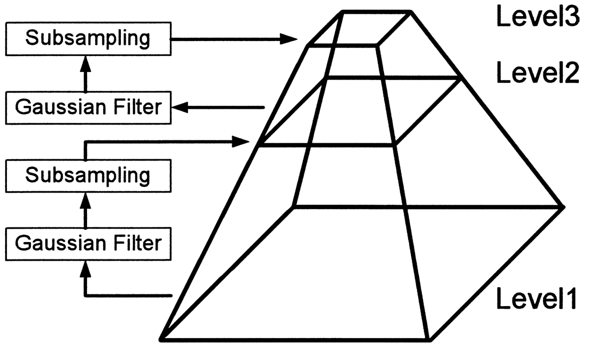


Fig. 2 Hierarchical image creation.

size. Considering the maximum motion distance found by the algorithm and a expected motion distance in sequences, the hierarchy level is set to three in case of CIF resolution. A Gaussian filter and 2:1 sub-sampling are utilized to create an upper-level hierarchy. Hence, large movement is apparently transferred to a small one, thereby satisfying the above constraints.

2.1.2 Multi-Dimensional Gradient Filter

The basic algorithm derives optical flow from a luminance gradient, which indicates luminance differences. The luminance gradient is sensitive to image noise because it is computed with luminance difference. So traditional algorithms such as Horn and Schunck employ a low-pass filter, which requires seven frames. To minimize frame memory, a multi-dimensional gradient filter [18] that requires only three frames has been adopted in this paper. The multi-dimensional gradient filter is a spatial-temporal filter. To reduce the number of filter taps of time directions, a spatial signal conditioning has been applied. Hence the multi-dimensional gradient filter has three taps for low-pass-filter and three tap for gradient filter. The proposed architecture adopts the coefficients summarized in Table 1, which has been introduced in [18].

Table 1 Filter coefficients of the multi-dimensional filter.

Gradient · LPF Coefficient	tap#1	tap#0 and 2
Low pass coefficient	0.551580	0.224210
Gradient	0.0	0.455271

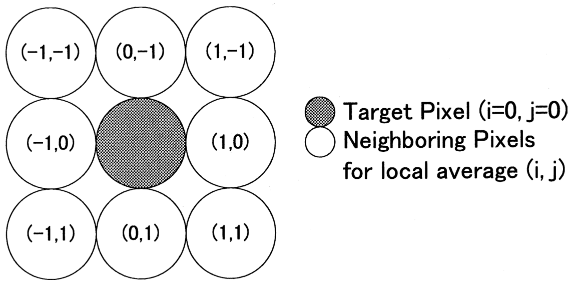


Fig. 3 Neighboring pixels for local average.

2.1.3 Optical Flow Calculation

Equations (1) and (2) based on the Horn and Schunck algorithm are used to obtain optical flow in HOE. These equations are lead from two assumption; 1) the pixel luminance value is preserved for a short time, 2) nearby pixels in the image plane move similarly.

$$u_{n+1} = \overline{u_n} - I_x \frac{I_x \overline{u}_n + I_y \overline{v}_n + I_t}{\alpha^2 + I_x^2 + I_y^2} \tag{1}$$

$$v_{n+1} = \overline{v_n} - I_y \frac{I_x \overline{u}_n + I_y \overline{v}_n + I_t}{\alpha^2 + I_x^2 + I_y^2} \tag{2}$$

In those equations, u is the x vector component and v is the y vector component of optical flow, and $\overline{u}$ and $\overline{v}$ indicate values of the velocity in directions x and y in some neighborhood of (i, j) as shown in Fig. 3. The respective luminance gradients of the x, y, and t dimensions are I_x, I_y and I_t. The α is the parameter to determine emphasis of two assumptions. The calculation is iterated until the difference of the current value (n+1) and the previous value (n) is less than a threshold value.

2.1.4 Interpolation and Compensation

The optical flow of the hierarchy is interpolated and motion compensated image is generated for lower hierarchy with following steps [19], [20].

1) The optical flow in top hierarchy image level is calculated.

Table 2 Variables and bit assignments of HOE.

Module	Variable	Bit length of each mode (Integer Part : Fractional Part)				
		Default	16Fixed	32Fixed	24Fixed	16–24 Fixed
Img–pyramid	kernel	float	8:8	16:16	12:12	8:8
	a	float	8:8	16:16	12:12	8:8
	ybuffer	double	8:8 (unsigned)	16:16	12:12	9:15
Gradient	lpf kernel	float	8:8	16:16	12:12	9:15
	diff kernel	float	8:8	16:16	12:12	9:15
	tmp1, tmp2	float	8:8 (unsigned)	16:16	12:12	9:15
	temp	float	8:8 (unsigned)	16:16	12:12	9:15
	temp2	float	8:8	16:16	12:12	8:8
	E_x, E_y, E_t	float	8:8	16:16	12:12	8:8
Flow	u, v	float	8:8	16:16	12:12	4:20
	ave_u, ave_v	float	8:8	16:16	12:12	4:20
	Ex, Ey, Et	float	8:8	16:16	12:12	8:8
	UU, VV	float	8:8	16:16	12:12	4:20
	tmp_u, tmp_v	float	8:8	16:16	12:12	4:20
	tmp	float	8:8	16:16	12:12	4:20
	k	float	8:8	16:16	12:12	4:20
	DIFF, ave_dif	float	8:8	16:16	12:12	4:20
	diff_u, diff_v	float	8:8	16:16	12:12	4:20
	diff	float	8:8	16:16	12:12	16:20
Inter–polation	UU, VV, tmp	float	8:8	16:16	12:12	8:8
	u, v	float	8:8	16:16	12:12	8:8
	threshold	float	8:8	16:16	12:12	8:8
Compen–sation	UU, VV	float	8:8	16:16	12:12	8:8
	u, v	float	8:8	16:16	12:12	8:8
	s, t	float	8:8	16:16	12:12	8:8

2) Bilinear interpolation with uniform weights is applied to adjust vector size for lower hierarchy.
3) Motion compensated image is created by the optical flow which is calculated in 2). Both forward and backward compensated images are created.
4) The optical flow of lower hierarchy is calculated by using the compensated images.
5) The interpolated optical flow in 2) and calculated optical flow in 4) are summed. Consequently, optical flow for corresponding image level is calculated.
6) Above operations are repeated to the lowest hierarchy and finally optical flow is derived.

2.2 Bit-Length Optimization for VLSI Design

The bit-length affects on the optical flow accuracy. However longer bit-length needs higher hardware-cost. Thus appropriate bit-length which balances accuracy and hardware costs should be considered. In addition, the HOE algorithm requires large work loads because iteration continues until the difference of the current and previous intermediate optical flows is less than a threshold value. The iteration also leads to the increase of the operation clock frequency and the memory bus bandwidth of the VLSI processor. Consequently, accuracy and number of iterations were evaluated for various bit-lengths condition. Table 2 shows various processes' variables and bit assignments.

A 32-bit floating-point number is used for all variables ("default" in Table 2) in the original source code. We prepared three bit-length variable sets: 16-bit fixed point (16 Fixed), 24-bit fixed point (24 Fixed), and 32-bit fixed point (32 Fixed). Figure 4 describes simulation results. Three moving picture sequences; translating-tree (Trans), diverging-tree (Div), Yosemite (YOS), which were used optical flow evaluation [21], and original composite sequence (Hash) are used for simulation. These sequences are artificially created so that each sequence has correct flow. The accuracy is evaluated by MAE (Mean Angle Error) which indicates angle differences between simulation results and correct optical flow. The parameter "α" which described in Sect. 2.1 was set "10," and iteration was continued until differences between previous and current optical flow is less than 0.0001 to satisfy the accuracy for all moving picture sequences.

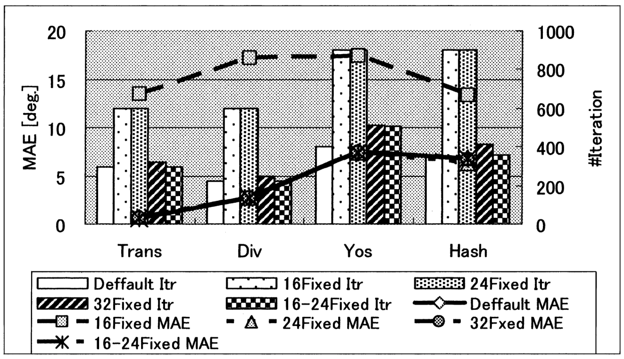


Fig. 4 Effects of various bit-length assignments.

Simulation results show that 16 Fixed did not achieve sufficient accuracy. Though 24 Fixed achieved sufficient accuracy, numerous iterations are necessary so that the reduction of bus bandwidth cannot be expected. By careful analysis of values that the variables takes in "default" case of all sequences, several bit-length combinations and positions of decimal points were set as a candidate. The bit-length and position of decimal point for all variables were tuned to achieve with the almost same accuracy and iteration number as "default." Finally, 16–24 Fixed mode was assessed. It assigned 16-bit and 24-bit lengths adaptively for each variable. It offers the best result: the 16–24 Fixed mode achieves almost identical accuracy (MAE=7.44 degree) and iteration repetitions to those of "default."

3. Optical Flow Processor Architecture

A processor architecture that allows 450 iterations was proposed to realize real-time CIF (352×288) 30 fr/s operation. In addition, the processor core is designed by a possible cascade connection to achieve higher resolution and higher accuracy.

3.1 Analyses of On-Chip Memory Size and Bus Bandwidth

An important issue to realize the processor core is the numerous memory accesses needed to derive optical flow. The

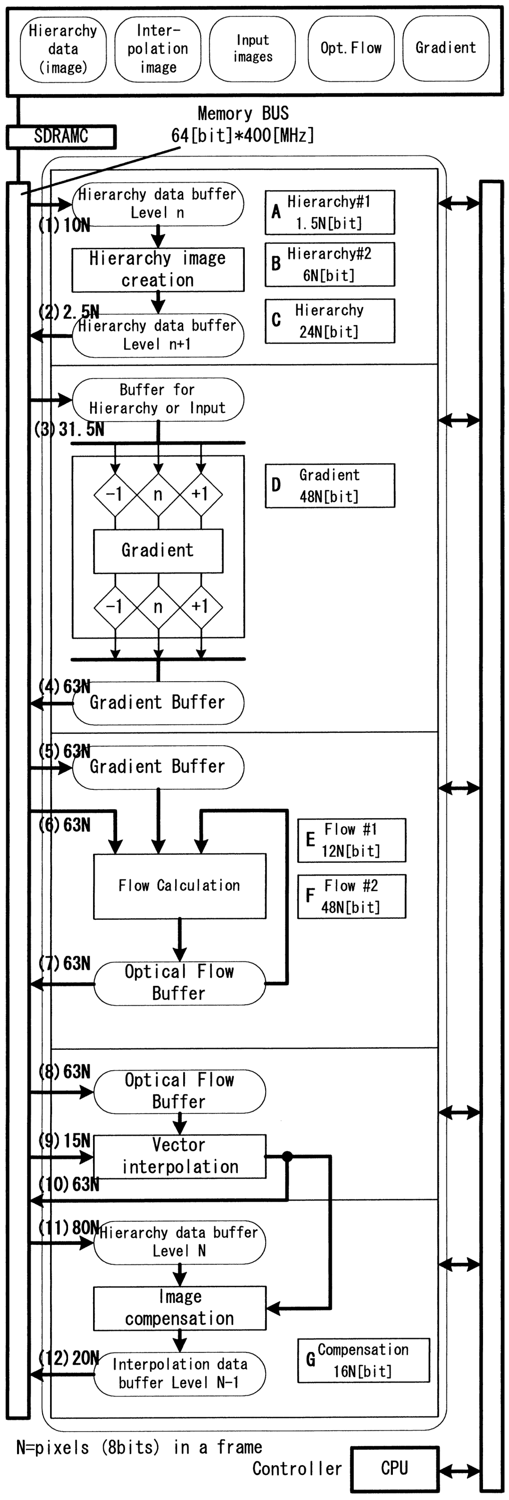


Fig. 5 Block diagram of the processor and data flow.

trade-off of on-chip memory size and external bus bandwidth were investigated. Figure 5 shows a block diagram and processor data flow. The processor consists of a main execution unit, an external memory controller, and a CPU. Numbers (1)–(12) represent bus accesses between the memory and processor; characters "A" to "G" represent the required memory volume. Table 3 indicates the data access and transaction volume in the case that all data are stored in external memory. Table 4 lists the required memory capacity for each process in the case that all data are stored in on-chip memory. Symbols "N" in Tables 3 and 4 represent the pixel (8 bits) numbers of a frame. For resolution of CIF size, N equals $352 \times 288 = 101,376$.

The maximum bus bandwidth between the external memory and the chip is impractical in the case where all data are stored in external memory. The total volume of on-chip memory in the case where all data except the original image input are stored in on-chip memory is also impractical. We investigated the bus bandwidth and on-chip memory capacity for six cases of data allocation to external memory and internal memory. Table 5 represents a summary of those considerations. Assuming by using PC3200 SDRAM, Case 4 or Case 5 seems to offer the best combination (less than 3.2 GByte/s). However, three types of luminance gradient

Table 3 Data transaction.

	bandwidth[bps]	contents (N = pixel number)
(1)	10N	input luminance of hierarchy #1 and #2 = 8bit*(1(Level1 Hier.)+1/4(Level2 Hier.))*N
(2)	2.5N	output created hierarchy #2 and #3 = 8bit*(1/4(Level1 Hier.)+1/16(Level2 Hier.))*N
(3)	31.5N	input three hierarchy luminance = 8bit*3frame*(1(Level1 Hier)+1/4(Level2 Hier.) +1/16(Level3 Hier.))*N
(4)	63N	output three luminance gradient (Ex, Ey, Et) = 16bit*3(Ex,Ey,Et)*(1(Level1 Hier.) +1/4(Level2 Hier.)+1/16(Level3 Hier.))*N
(5)	63N	input three luminance gradient (Ex, Ey, Et) at one iteration = 16bit*3(Ex,Ey,Et)*(1(Level1 Hier.) +1/4(Level2 Hier.)+1/16(Level3 Hier.))*N
(6)	63N	input optical flow data (u, v) at one iteration = 48bit(u,v)*(1(Level1 Hier.) +1/4(Level2 Hier.)+1/16(Level3 Hier.))*N
(7)	63N	output optical flow data (u, v) at one iteration = 48bit(u,v)*(1(Level1 Hier.) +1/4(Level2 Hier.)+1/16(Level3 Hier.))*N
(8)	63N	input optical flow data (u, v) = 48bit(u,v)*(1(Level1 Hier.) +1/4(Level2 Hier.)+1/16(Level3 Hier.))*N
(9)	15N	input optical flow data of hierarchy #2 and #3 = 48bit(u,v)*(1/4(Level2 Hier.)+1/16(Level3 Hier.))*N
(10)	63N	output interpolated optical flow of each hierarchy = 48bit(u,v)*(1(Level1 Hier.) +1/4(Level2 Hier.)+1/16(Level3 Hier.))*N
(11)	80N	input luminance of hierarchy #1 and #2 = 8bit*4pixel*2frame*(1(Level1 Hier.)+1/4(Level2 Hier.))*N
(12)	20N	input compensated luminance of hierarchy #1 and #2 = 8bit*2frame*(1(Level1 Hier.)+1/4(Level2 Hier.))*N

Table 4 Data volume.

	capacity[bit]	contents (N = pixel number)
A	1.5N	3 frames luminance of hierarchy #3 = 8bit*3frame*1/16(Level3 Hier.)*N
B	6N	3 frames luminance of hierarchy #2 = 8bit*3frame*1/4(Level2 Hier.)*N
C	24N	3 frames luminance of hierarchy #1 = 8bit*3frame*N
D	48N	luminance gradient (Ex, Ey, Et) (for lowest hierarchy) = 16bit*3(Ex,Ey,Et)*N
E	12N	optical flow (u, v) for hierarchy #2 = 48bit(u,v)*1/4(Level2 Hier.)*N
F	48N	optical flow (u, v) for hierarchy #1 = 48bit(u,v)*N
G	16N	2 frames of compensated luminance = 8bit*2frame*N

Table 5 Data allocation trade-off.

		case1		case2		case3		case4		case5		case6	
		on	ext.	on	ext.	on	ext.	on	ext.	on	ext.	on	ext.
Memory Volume	(A)	0	1.5	1.5	0	0	1.5	0	1.5	0	1.5	0	1.5
	(B)	0	6	6	0	0	6	0	6	0	6	0	6
	(C)	0	0	24	0	0	0	0	0	0	0	0	0
	(D)	0	48	48	0	48	0	32	16	16	32	0	48
	(E)	0	12	12	0	0	0	0	0	0	0	0	0
	(F)	0	48	48	48	48	48	48	48	48	48	48	48
	(G)	0	16	16	0	0	16	0	16	0	16	0	16
	Input Image	0	32	0	32	0	32	0	32	0	32	0	32
	total	**0**	**163.5**	**155.5**	**80**	**96**	**103.5**	**80**	**119.5**	**64**	**135.5**	**48**	**151.5**
	MB	**0**	**2.07**	**1.97**	**1.01**	**1.22**	**1.31**	**1.01**	**1.51**	**0.81**	**1.72**	**0.61**	**1.92**
Bus band-width	(1)	10		10		10		10		10		10	
	(2)	2.5		0		2.5		2.5		2.5		2.5	
	(3)	31.5		0		31.5		31.5		31.5		31.5	
	(4)	63		0		0		21		42		63	
	(5)*	9450		0		0		3150		6300		9450	
	(6)*	9450		0		0		0		0		0	
	(7)*	9450		0		63		63		0		0	
	(8)	63		0		0		0		0		0	
	(9)	15		0		0		0		0		0	
	(10)	63		0		0		0		0		0	
	(11)	80		20		80		80		80		80	
	(12)	20		20		20		20		20		20	
	total	**28698**		**50**		**207**		**3378**		**6486**		**9657**	
	GB/s	**10.91**		**0.02**		**0.08**		**1.28**		**2.47**		**3.67**	

(E_x, E_y, E_t) used for optical flow calculation are divided to on-chip memory and external memory. Thereby, data control is too complicated. Case 6 is another architecture candidate, but 3.7 GBytes/s of band-width is too large for practical implementation. In fact Case 3 was selected as the best candidate for the proposed architecture because it offers the lowest bus bandwidth along with a simple data control solution.

3.2 Common Processor Architecture

As described in a previous section, the iterated optical flow calculation is the major workload to derive optical flow while other operation loads are smaller. For that reason, most hardware aside from that used for optical flow calculation, is unused most of the time. The trade-off between operation cycle and hardware cost was considered hence, a processor with shared operation elements was designed for all necessary process to minimize hardware. The processor is based on the design for optical flow calculation – the heaviest operation. Other operations are realized with identical elements by additional data path. Figure 6 shows the proposed common execution element (CE) to realize high throughput operation for optical flow calculation. The CE comprises four common processor elements (PE), two adders for hierarchical image creation, and an accumulator to accumulate amount of update of optical flow.

Figure 7 shows a block diagram of a PE that comprises six blocks: an average operation block (AVE) which executes average, addition and shift operations, a base element block left (BEL) and a base element block right (BER) which calculates sum of products filter, a divide or add block (Div_Add), an update and low-pass filter block (Upd_LPF) which operates inner product computation, and a differential block (Diff) which computes subtraction and filter operations.

The CE allows execution of all necessary operations by programmable data-paths that are implemented for each block in the PE. Data-paths are structured to achieve high-

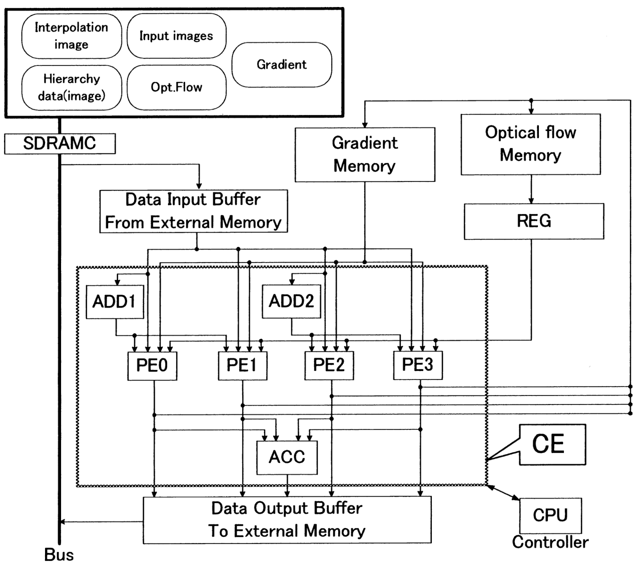


Fig. 6 Common execution element (CE).

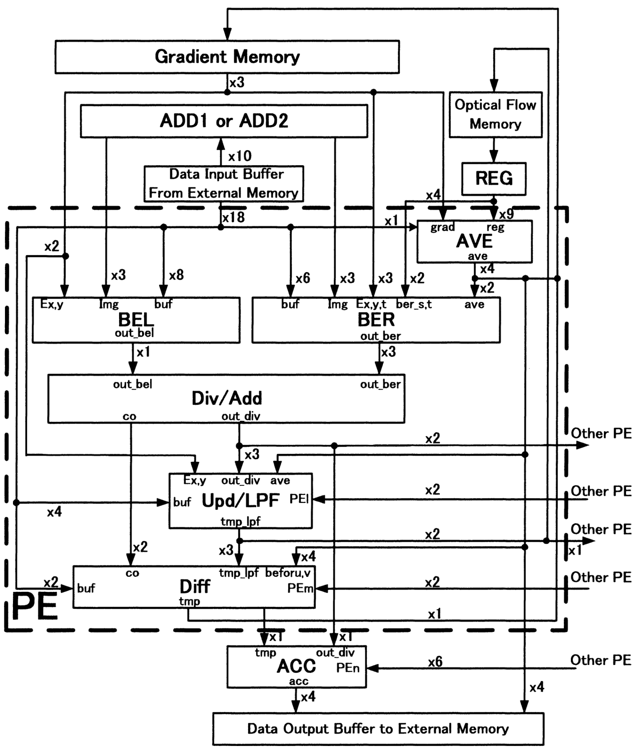


Fig. 7 Common processor element (PE).

throughput operation. Figure 8 shows how optical flow operations are performed in the CE. The AVE block calculates the local average. The BEL and BER blocks derive the denominator and numerator, respectively. The Div_Add block calculates division. Finally optical flow is derived through the Upd_LPF block. The Diff block calculates amount of update for all of pixel which are accumulated at ACC block to determine whether to finish iteration.

Figure 9(a) shows process flow of luminance gradient

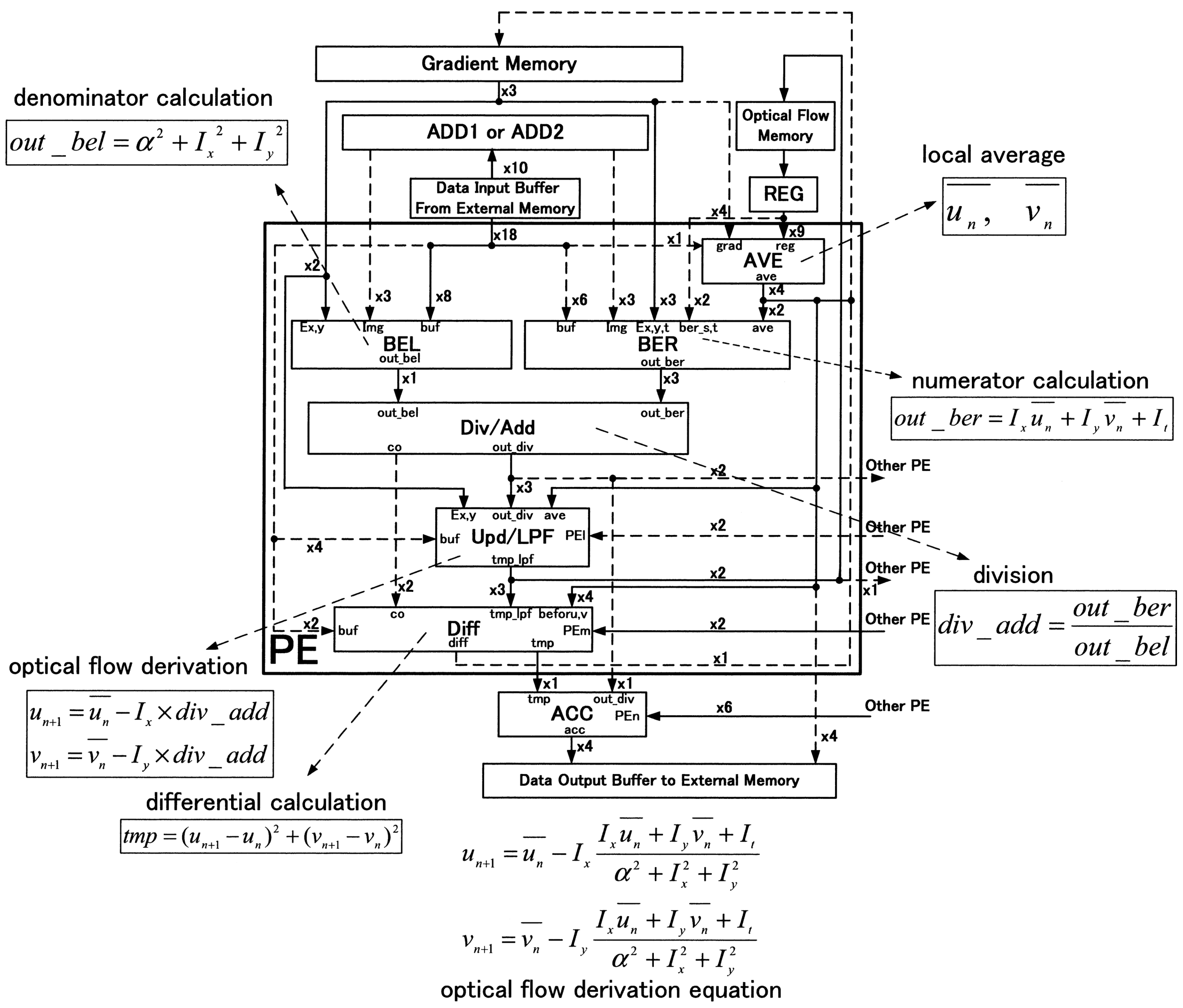

Gradient Memory
denominator calculation
$out\_bel = \alpha^2 + I_x^2 + I_y^2$
ADD1 or ADD2
Optical Flow Memory
REG
local average
$\overline{u_n}, \overline{v_n}$
Data Input Buffer From External Memory
AVE
grad
reg
ave
BEL
Ex,y
Img
buf
out_bel
BER
Ex,y,t
ber_s,t
out_ber
numerator calculation
$out\_ber = I_x\overline{u_n} + I_y\overline{v_n} + I_t$
Div/Add
co
out_div
Other PE
Upd/LPF
PEl
tmp_lpf
division
$div\_add = \frac{out\_ber}{out\_bel}$
Diff
tmp_lpf
beforu,v
PEm
diff
tmp
PE
optical flow derivation
$u_{n+1} = \overline{u_n} - I_x \times div\_add$
$v_{n+1} = \overline{v_n} - I_y \times div\_add$
ACC
PEn
acc
Data Output Buffer to External Memory
differential calculation
$tmp = (u_{n+1} - u_n)^2 + (v_{n+1} - v_n)^2$
$u_{n+1} = \overline{u_n} - I_x \frac{I_x\overline{u_n} + I_y\overline{v_n} + I_t}{\alpha^2 + I_x^2 + I_y^2}$
$v_{n+1} = \overline{v_n} - I_y \frac{I_x\overline{u_n} + I_y\overline{v_n} + I_t}{\alpha^2 + I_x^2 + I_y^2}$
optical flow derivation equation

Fig. 8 Optical flow execution status (CE).

operation. As described in Sect. 2.1, low-pass filter (LPF) for noise reduction and gradient derivation filter are implemented as three-taps FIR filter, with different coefficients. Each filter is used in order of LPF for "t," LPF for "y," and gradient for "x" to obtain the gradient in the direction of x. To obtain the gradient in the direction "y," each filter is used in order of LPF for "t," LPF for "x," and gradient for "y." For direction of t, each filter is used in order of gradient for "t," LPF for "y," and LPF for "x."

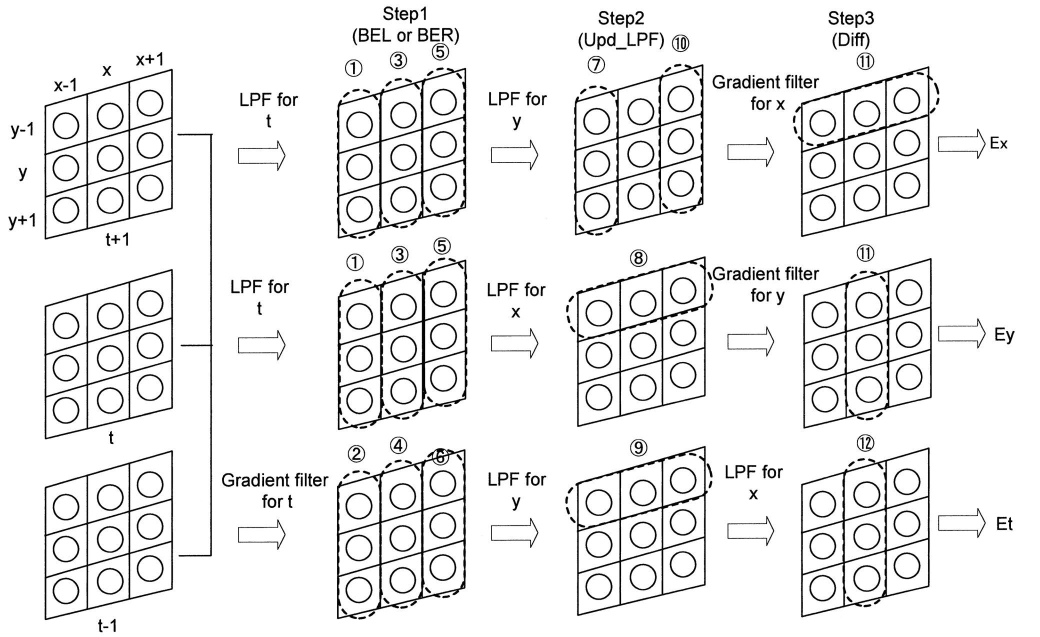
Figure 9(b) shows the data path which computes 3 gradient data in one clock cycle. For simplicity, some elements in the PEs are omitted. The number ①–⑫ attached to elements in PEs corresponds the number in Fig. 9(a). The luminance data and filter coefficients are fed from data input buffer. The filter operations to derive x-, y- and t-gradient are executed simultaneously.

Figure 10 shows operation example in the data path of BEL. Figure 10(a) illustrates the case of denominator calculation and Fig. 10(b) shows the case of a low pass filter. The name of input ports are corresponded with BEL ports in Fig. 7; E_x and E_y (input from Gradient Memory), Img (input from ADD1 or ADD2), and buf (input from Data Input Buffer). But valid input port names such as "axa" are changed as variables of equation in Fig. 10.

3.3 High-Throughput SIMD Architecture

As describe in previous section, CE has four PEs to enable simultaneous optical flow derivation for four pixels. We also adopted 4-SIMD architecture to achieve high throughput operation. Figure 11 shows the iteration of optical flow derivation. Figure 11 is drawn based on Fig. 8, but for simplicity, 3-SIMD architecture is shown and the data flow for iteration is extracted. "Data Input Buffer from External Memory," "ADD1 or ADD2," and "Gradient Memory" are omitted. Pipeline registers described as "REG" in Figs. 6 and 8, are illustrated with pipeline scheme in Fig. 11.

Optical flows of the eight neighboring pixels obtained in the previous iteration are employed to derive current optical flow data. Previous optical flow contained in the optical flow memory are transferred sequentially to the shift registers. The outputs of the shift registers are fed into AVE in PEs. The optical flows for each pixels of columns "01," "02" and "03" in the current fame are calculated until reaching to



①③⑤ $= E\_ml \times lpf0 + E\_mm \times lpf1 + E\_mn \times lpf0$

②④⑥ $= E\_ml \times grad0 + E\_mm \times grad1 + E\_mn \times grad0$

⑦⑧⑨⑩ $= (E\_l + E\_n) \times lpf0 + E\_m \times lpf1$

⑪ $= grad0 \times (E\_i - E\_k) \quad or \quad grad0 \times (E\_j - E\_g)$

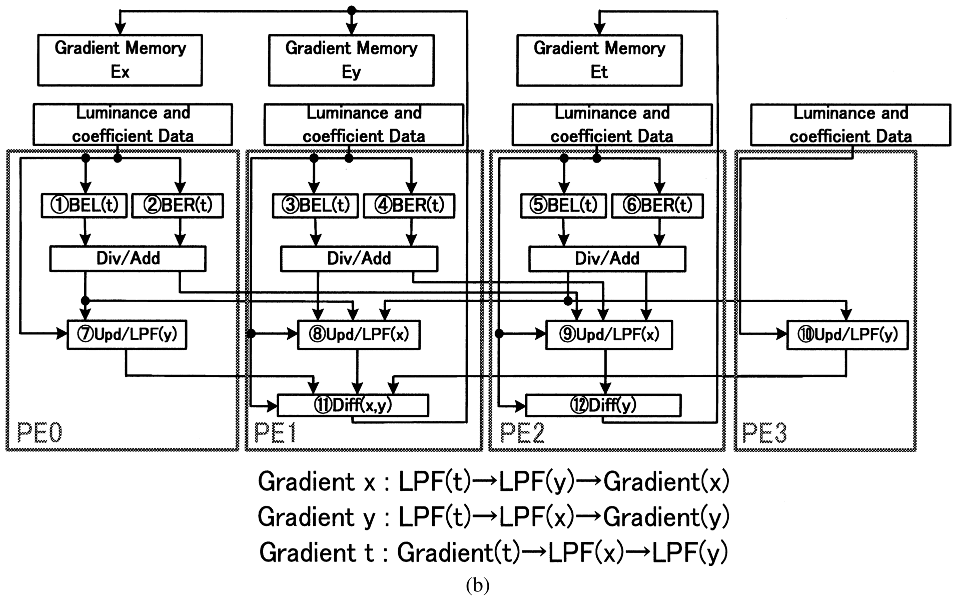
⑫ $= (E\_i + E\_k) \times lpf0 + (E\_j + E\_g) \times lpf1$

E_mx : luminance

lpf_x : LPF_coeff

$grad\_x$: Gradient_coeff

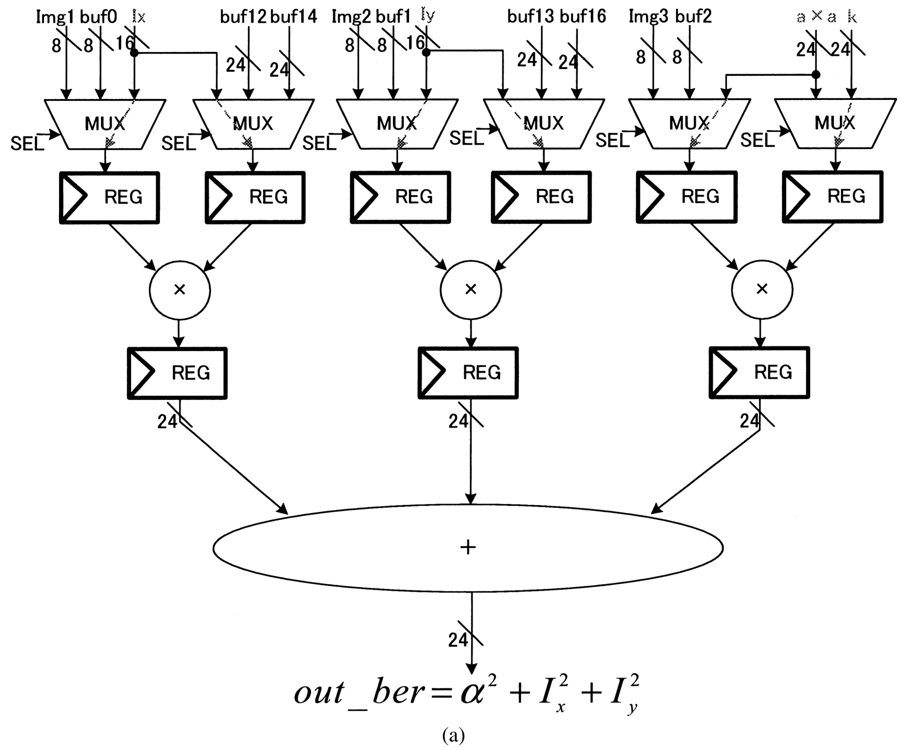
(a)


Gradient Memory Ex
Gradient Memory Ey
Gradient Memory Et
Luminance and coefficient Data
Luminance and coefficient Data
Luminance and coefficient Data
Luminance and coefficient Data
①BEL(t)
②BER(t)
③BEL(t)
④BER(t)
⑤BEL(t)
⑥BER(t)
Div/Add
Div/Add
Div/Add
⑦Upd/LPF(y)
⑧Upd/LPF(x)
⑨Upd/LPF(x)
⑩Upd/LPF(y)
⑪Diff(x,y)
⑫Diff(y)
PE0
PE1
PE2
PE3
Gradient x : LPF(t)→LPF(y)→Gradient(x)
Gradient y : LPF(t)→LPF(x)→Gradient(y)
Gradient t : Gradient(t)→LPF(x)→LPF(y)

(b)

Fig. 9 (a) Gradient calculation process. (b) Data path for gradient calculation.

the bottom of the frame. Subsequently, optical flows for pixels of the next column—“04,” “05” and “06” are calculated. At this point, the input data of columns “03” and “04” must be previous optical flow data, but data of column “03” are updated and “04” must be retained. To manage this operation, optical flow data memory must be a 3-port memory and complicated addressing is necessary. Hence, the previous flow buffer is introduced. Address control becomes simplified and all optical flow memory can thereby be 2-port memory. Figure 12 illustrates optical flow operations with the previous flow buffer. The optical flow data outputs of the right two column memory (“Mem D” and “Mem E”) are connected to the previous flow buffer (“Mem A” and “Mem B”). The previous flow buffer is operated as a one vertical-line-delay storage. Thus at the operation of columns “04,” “05” and “06,” previous data of “03” and “04” are provided



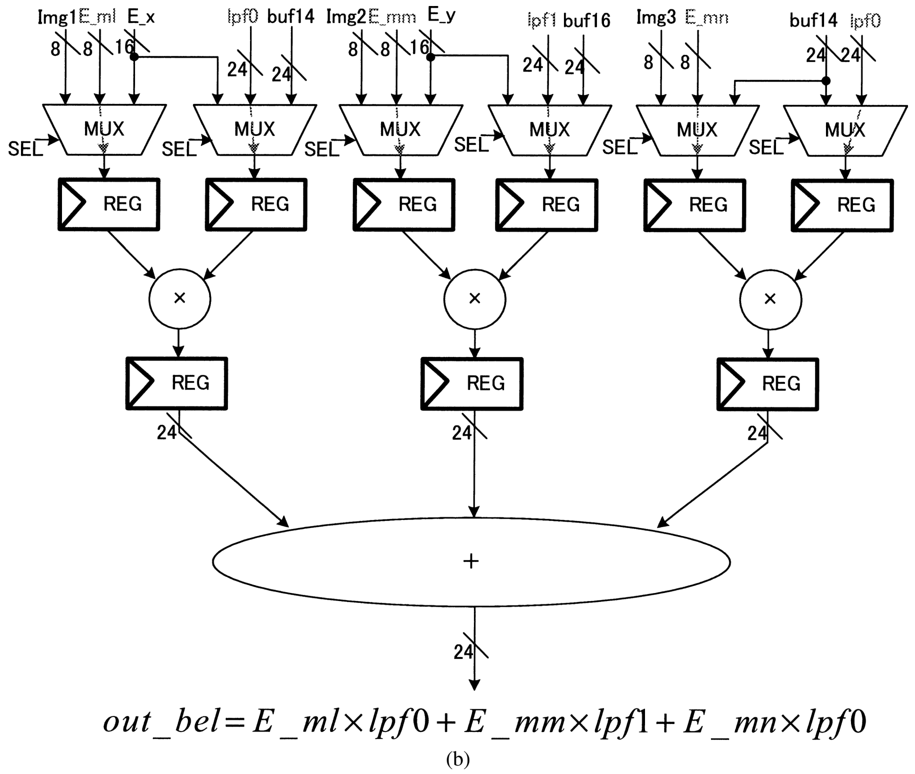


Fig. 10 (a) Data path of BEL for denominator calculation. (b) Data path of BEL for LPF.

from the previous flow buffer.

3.4 Scalable Processor Architecture

One CE can perform 450 iterated calculations for CIF 30 fr/s video. But more iterations might be required for higher accuracy or higher resolution. Consequently, a scalable processor architecture is necessary to meet requirements for higher iteration. Figure 13 shows how two CEs are connected to double iterations. By connecting two CEs, simul-

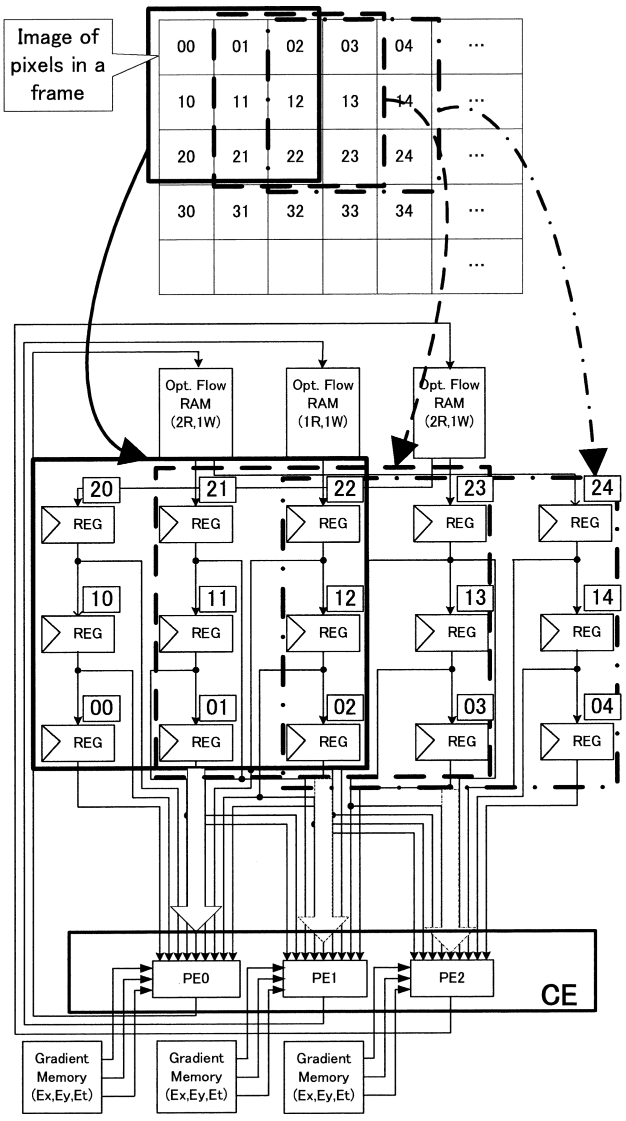


Fig. 11 Optical flow data flow with SIMD architecture.

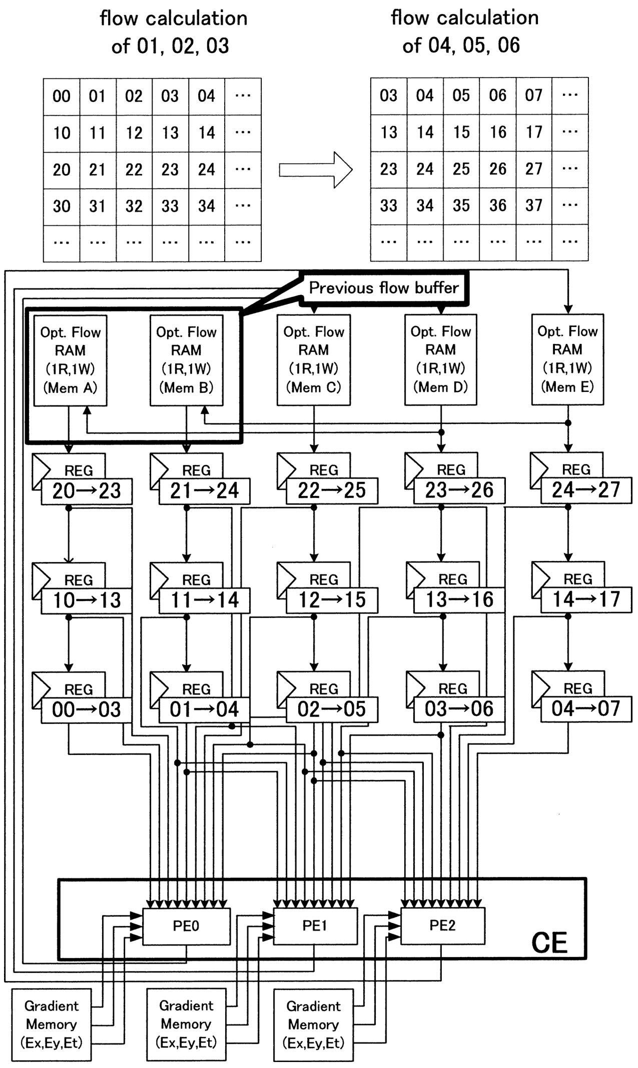

flow calculation of 01, 02, 03
flow calculation of 04, 05, 06
Previous flow buffer
Opt. Flow RAM (1R,1W) (Mem A)
Opt. Flow RAM (1R,1W) (Mem B)
Opt. Flow RAM (1R,1W) (Mem C)
Opt. Flow RAM (1R,1W) (Mem D)
Opt. Flow RAM (1R,1W) (Mem E)
REG
20→23
21→24
22→25
23→26
24→27
10→13
11→14
12→15
13→16
14→17
00→03
01→04
02→05
03→06
04→07
PE0
PE1
PE2
CE
Gradient Memory (Ex,Ey,Et)

Fig. 12 Optical flow operation using previous flow buffer.

taneous eight-pixel optical flow calculation is achievable. As described in Sect. 3.3, optical flows of the two right edge columns are contained in the previous flow buffer. In this case, the flows of the two right-most columns of CE1 are connected with the previous flow buffer.

The optical flow derivation for the NTSC resolution video is achievable using a four-CE structure. Figure 14 shows a frame division scheme by four CEs (0, 1, 2, 3). By dividing optical flow derivation for horizontal pixels, the size of optical flow memory and gradient memory of respective CEs are unchanged even if the vertical pixel size is increased.

3.5 Design of Optical Flow Memory

As described in Sect. 3.1, large capacity buffers are required to achieve real-time optical flow operation. However, the processor core size is intended for practical use. Optical flow derivation is an iteration of identical operations and the same memory address is accessed repeatedly within a constant time interval. For that reason, SRAM is not needed for optical flow memory. It can be a DRAM with smaller area. Therefore, an on-chip 2-port DRAM was designed as optical flow memory. A 2-port SRAM cell employs eight transistors and occupies 3.79 μm^2, assuming 90 nm CMOS technology. Our proposed DRAM cell, however, achieves 2-port memory function with four transistors and the area of the memory cell is 2.13 μm^2. Figure 15 shows a circuit diagram of the DRAM cell. The CMOS configuration of the write- access- gate enlarges a low-voltage margin. The DRAM cell can retain data within the iteration interval. The interval time of optical flow calculation is 149 μs. By applying back-bias voltage, the leakage current is suppressed so that data are retained for over 200 μs at a 100°C condition, which is longer than the optical flow calculation interval time.

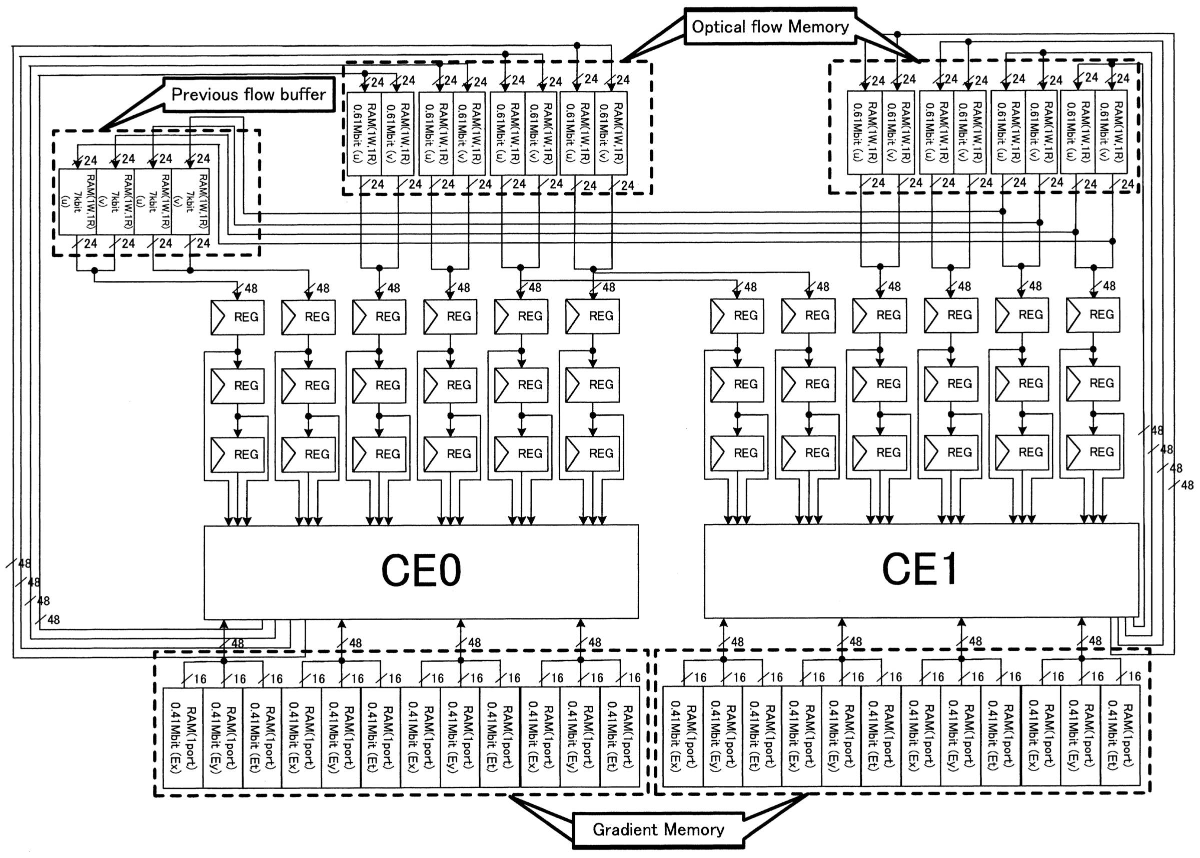


Fig. 13 Scalable processor architecture.

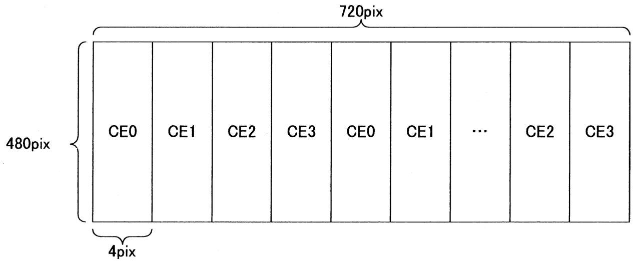


Fig. 14 Frame division by multiple processors.

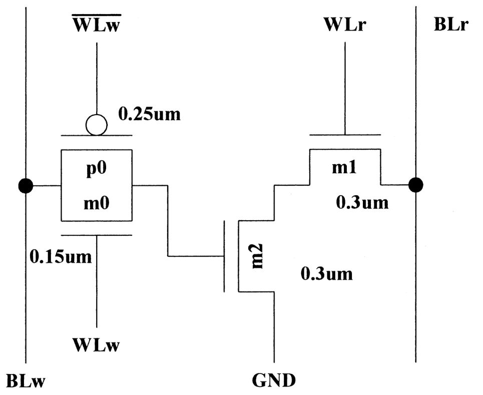


Fig. 15 Circuit diagram of 2-port DRAM cell.

3.6 Estimation of Processor Core Size

The core size of optical processor was estimated with 90-nm CMOS process technology. Figure 16 shows floor plans of: (a) optical flow memory, (b) previous buffer flow memory, and (c) luminance gradient memory. Figure 17 shows the floor plan of the overall processor core. The floor plans are drawn on the basis of the area estimation for each circuit block. The estimated core size is 6.02 mm × 5.33 mm.

Table 6 shows comparison of circuit volume, memory capacity, performance and accuracy between previous works and proposed processor. The proposed processor has achieved best performance (CIF 30 fr/s) and best accuracy (MAE=7.44) with 100% density along with relatively lower logic circuit volume. The proposed processor adopts original on-chip memory implementation suitable for HOE algorithm which described as Sect. 2.

The power consumption is 190 mW for logic, 230 mW for DRAM and 80 mW for SRAM. Totally 500 mW is dissipated for CIF resolution video at frame rate 30 fr/s.

4. Conclusion

This study proposed an real-time optical flow processor architecture. That allows real-time operation with CIF 30 fr/s. Based on Hierarchical Optical flow Estimation (HOE), the

Table 6 Comparison of proposed processor and previous works.

	Circuit Volume (Logic)	Memory Capacity	Performance	Accuracy (MAE)	Algorithm	Note
[5]	1,980KG	12.8KByte(On chip) +External(Unknown)	320x240x24fr/s	18.30	Lucas&Kanade	Density=100%
[6]	XC4020E+XC4005H	N/A	50x50x19fr/s	N/A	Horn&Schunk	
[7]	47mm^2 at 0.7um CMOS	N/A	128x128x50fr/s	N/A	Horn&Schunk	
[8]	EPF10K50RC240-3	46KByte	96x96x22.56fr/s	N/A	Original(correlation based)	
[9]	2 FPGAs(Type Unknown)	16KByte	N/A	N/A	Horn&Schunk	
[10]	MaxVideo200	N/A	252x316x20.9fr/s	9.58	Lucas&Kanade	Density=35.1%
proposed	311KG	2085KByte	352x288x30fr/s	7.44	Original(Horn&Schunk Based)	Density=100%

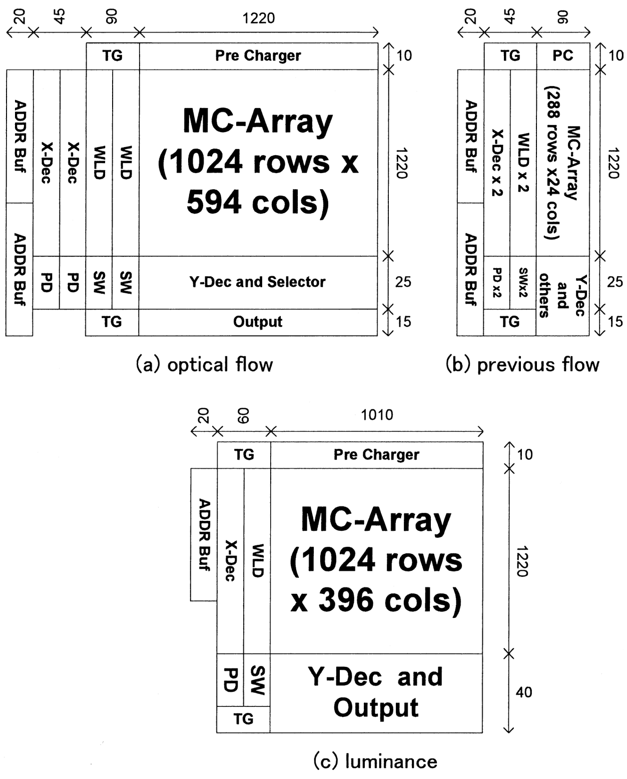


Fig. 16 Memory floor plan.

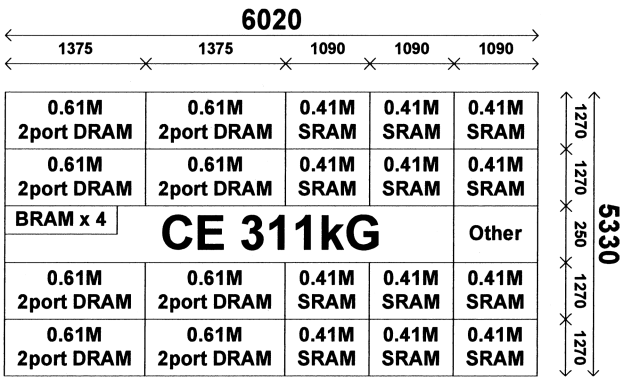


Fig. 17 Processor core floorplan.

appropriate bit-length of variables in the algorithm was determined while maintaining high accuracy (MAE=7.44 degree MAE). The proposed processor architecture achieves CIF 30 fr/s with 450 iterations. As a result of bus band width analysis, the optical flow memory and luminance gradient memory were implemented as on-chip memory to realize real-time operation. We proposed a common processing element that facilitates all necessary optical flow operations and an on-chip DRAM that can keep data within the optical flow data interval time. Hence, we achieved an extensive reduction of silicon area. Moreover the scalable architecture has been developed to meet requirements of larger frame size and higher accuracy. The processor core size is estimated as 6.02 mm × 5.33 mm with six-layer-metal 90-nm CMOS technology. The required operating frequency is 189 MHz and power consumption is 500 mW with CIF resolution at frame rate 30 fr/s.

Acknowledgements

This work is supported by Semiconductor Technology Academic Research Center (STARC), VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Cadence Design Systems, Inc. and Synopsys, Inc.

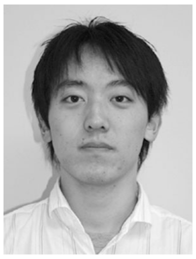
References

[1] T. Yamamoto, K. Imamura, and H. Hashimoto, "Improvement of optical flow by moving-object detection using temporal correlation," Trans. IIITE, vol.55, no.6, pp.907–911, 2001.
[2] P.H. Batavia, D.A. Pomerleau, and C.E. Thorpe, "Overtaking vehicle detection using implicit optical flow," Proc. ITSC, pp.729–734, Nov. 1997.
[3] M. Tistarelli, F. Guarnotta, D. Rizzieri, and F. Tarocchi, "Application of optical flow for automated overtaking control," Proc. IEEE Workshop on Applications of Computer Vision, pp.105–112, Dec. 1994.
[4] A. Giachetti, M. Campani, R. Sanni, and A. Succi, "The recovery of optical flow for intelligent cruise control," Proc. IEEE Intelligent Vehicle Symposium, pp.91–96, Oct. 1994.
[5] J. Diaz, E. Ros, S. Mota, F. Pelay, and E.M. Ortigosa, "Real-time optical flow computation using FPGAs," Early Cognitive Vision Workshop, Talk21, 2004.
[6] P. Cobos and F. Monasterio, "FPGA implementation of the Horn & Schunck optical flow algorithm for motion detection in real time images," Proc. DCIS'98 XIII, pp.616–621, 1998.
[7] T. Röwekamp, M. Platzner, and L. Peters, "Specialized architectures for optical flow computation: A performance comparison of ASIC, DSP, and multi-DSP," Proc. ICSPAT'97, pp.829–833, 1997.
[8] P. Cobos and F. Monasterio, "FPGA implementation of camus correlation optical flow algorithm for real time images," Vision Interface Proceedings VI2001, 14th International Conference on Vision Interface, pp.7–9, 2001.
[9] A. Zuloaga, J.L. Martin, and J. Ezquerra, "Hardware architecture for optical flow estimation in real time," Proc. International Conference on Image Processing, ICIP 98, vol.3, pp.972–976, 1998.
[10] M.V. Correia and A.C. Campilho, "Real-time implementation of an optical flow algorithm," ICPR, vol.4, pp.247–250, 2002.

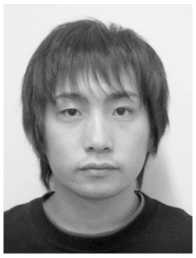
[11] B.K.P. Horn and B.G. Schunck, "Determining optical flow," AI, vol.17, pp.185–204, 1981.
[12] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," Proc. DARPA Image Understanding Workshop, pp.121–130, 1981.
[13] S. Uras, F. Girosi, A. Verri, and V. Torre, "A computational approach to motion perception," Biol. Cybern., vol.60, pp.79–97, 1988.
[14] H.H. Nagel, "On the estimation of optical flow: Relations between different approaches and some new results," AI, vol.33, pp.299–324, 1987.
[15] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," Int. J. Comput. Vis., vol.2, pp.283–310, 1989.
[16] A. Singh, "An estimation-theoretic framework for image-flow computation," Proc. IEEE ICCV, pp.168–177, Osaka, 1990.
[17] D.J. Fleet and A.D. Jepson, "Computation of component image velocity from local phase information," Int. J. Comput. Vis., vol.5, pp.77–104, 1990.
[18] E.P. Simonceli, "Design of multi-dimensional derivative filters," Proc. First IEEE Int. Conf. Image Processing, vol.I, pp.790–794, 1994.
[19] P. Anandan, J.R. Bergen, K.J. Janna, and R. Hingorani, "Hierarchical model-based motion estimation," in Motion Analysis and Image Sequence Processing, ed. M.I. Sezan and R.L. Lagendijk, pp.1–22, Kluwer, Boston, MA, 1993.
[20] S.H. Hwang and S.U. Lee, "A hierarchical optical flow estimation algorithm based on the interlevel motion smoothness constraint," Pattern Recognit., vol.26, no.6, pp.939–952, 1993.
[21] J.L. Barron, D.L. Fleet, and S.S. Beauchemin, "Performance of optical flow techniques," Int. J. Comput. Vis., vol.12, no.1, pp.43–77, 1994.

Noriyuki Minegishi graduated from Toin Technical College in 1985. He joined Mitsubishi Electric Corp. in 1985, where he has been engaged in video processing, chip design, and verification of video processing. He is a member of the Information Technology R&D Center (Kamakura, Japan). From 2003, he is also enrolling in the Doctoral course of Kanazawa University.

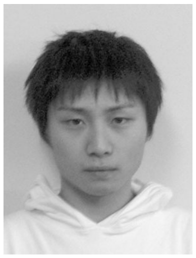
Junichi Miyakoshi received a B.E. degree from Kanazawa University in 2002. He received an M.S. from Kanazawa University in 2004. He is currently enrolled in the Doctoral course of Kobe University. His research focus is low-power VLSI techniques for image processing.



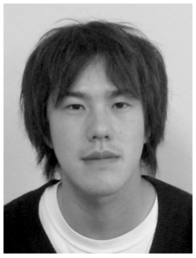
Yuki Kuroda received a B.E. degree and M.S. in Electrical and Information Engineering from Kanazawa University, Ishikawa, Japan, in 2003 and 2005 respectively. He joined Hitachi Ltd., Central Research Laboratory where he has been engaged in low-power design technology and multimedia processing for mobile communications and intelligent transport systems.



Tadayoshi Katagiri was born on February 23, 1982. He received a B.E. in Information and Systems Engineering from Kanazawa University in 2004. He is currently pursing a Master's degree at Kanazawa University. His research interests are video segmentation algorithms based on optical flow techniques.



Yuki Fukuyama was born on March 15, 1982. He received a B.E. in Computer and Systems Engineering from Kobe University in 2005. He is currently a Master's student at Kobe University. His research interest is low-power VLSI techniques for image processing.



Ryo Yamamoto was born on April 26, 1982. He received a B.E. in Electrical and Information Engineering from Kanazawa University, Ishikawa, Japan, in 2005. He is currently a Master's student at Kobe University. His research interest is low-power VLSI system for image processing.

Masayuki Miyama was born on March 26, 1966. He received a B.S. in Computer Science from the University of Tsukuba in 1988. He joined PFU Ltd. in 1988. He received an M.S. in Computer Science from the Japan Advanced Institute of Science and Technology in 1995. He joined Innotech Corp. in 1996. He received a Ph.D. in Electrical Engineering and Computer Science from Kanazawa University in 2004. He is a Research Assistant in the Department of Electrical and Electronic Engineering at Kanazawa University. His present research focus is a low-power design technique for multimedia VLSI.

Kousuke Imamura received B.S., M.S. and Dr. Eng. degrees in Electrical Engineering and Computer Science in 1995, 1997 and 2000, respectively—all from Nagasaki University. He is currently a Lecturer of Information and Systems Engineering, Kanazawa University. His research interests are high-efficiency image coding and image processing.

Hideo Hashimoto received the B.S., M.S. and Dr. Eng. degrees in Electronic Engineering in 1968, 1970 and 1975, respectively, all from Osaka University. He joined Electrical Communication Laboratories of Nippon Telegraph and Telephone Corporation (NTT) in 1975. Since 1993, he has been a professor of Information and System Engineering, Kanazawa University. His research interests are video coding, moving object segmentation and visual communication.

Masahiko Yoshimoto received a B.S. in Electronic Engineering from the Nagoya Institute of Technology, Nagoya, Japan, in 1975, and an M.S. degree in Electronic Engineering from Nagoya University, Nagoya, Japan, in 1977. He received a Ph.D. in Electrical Engineering from Nagoya University, Nagoya, Japan in 1998. He joined the LSI Laboratory, Mitsubishi Electric Corp. Itami, Japan, in April 1977. From 1978 to 1983, he was engaged in the design of NMOS and CMOS static RAM, including a 64-Kb full CMOS RAM with a divided-word-line structure. Since 1984 he has been involved in research and development of multimedia ULSI systems for digital broadcasting and digital communication systems based on MPEG-2 and MPEG-4 Codec LSI core technology. Since 2000, he had been a Professor of the Dept. of Electrical & Electronic System Engineering at Kanazawa University, Japan. Since 2004, he has been a Professor of the Dept. of Computer and Systems Engineering in Kobe University, Japan. His current activities are centered upon the research and development of multimedia and ubiquitous media VLSI systems including an ultra-low-power image compression processor and a low-power wireless interface circuit. He holds 70 registered patents. Dr. Masahiko Yoshimoto is a member of the IEEE. He served on the Program Committee of the IEEE International Solid State Circuit Conference from 1991 to 1993. In addition, he has served as Guest Editor for special issues on Low-Power System LSI, IP and Related Technologies of IEICE Transactions in 2004. He received the R&D100 awards from the R&D magazine for development of a DISP and the development of a real-time MPEG-2 video encoder chipset in 1990 and 1996, respectively.