SIMD Processor Arrays for Image and Video Processing: A Review

*Thinh M. Le, W. M. Snelgrove, and S. Panchanathan
*Department of Electrical and Computer Engineering, University of Ottawa
Ottawa, Ontario, Canada, K1N 6N5
URL: http://marge.genie.uottawa.ca
Department of Electronics, Carleton University, Canada
Department of Computer Science and Engineering, Arizona State University, USA

ABSTRACT

SIMD processor arrays are becoming popular for their fast parallel executions of low- to medium-complexity image and video processing algorithms, and most stages of the compression standards (JPEG, H.263, and MPEG's). In many existing techniques, visual data processing algorithms and compression standards possess a high degree of parallelism. In particular, the processing of a certain pixel/block does not generally require data from a distant pixel/block, and the instructions for processing these pixels/blocks are usually identical. Thus, these algorithms map naturally onto the architecture of the SIMD processor arrays. In this paper, the architectures of the recently SIMD processor arrays will be reviewed together with algorithms demonstrating their superior features. Due to the length of the paper, only processor arrays implemented at the chip-level are discussed, especially those whose logic circuits are merged/embedded in the SRAM or DRAM memory process. While some processor arrays are designed by embedding the memory modules onto the existing processors, a significant number of processor arrays are realized by integrating logic circuits onto existing RAM (logic-in-memory) to save the inherently large memory bandwidth, and thus achieving a performance in the order of Tera instructions per second.

Key words: Processor arrays, SIMD architecture, parallel processing, logic-in-memory.

1. INTRODUCTION

SIMD processor arrays are becoming popular for their fast parallel executions of low- to medium-level image and video processing algorithms, and most stages of the compression standards (JPEG, H.263, and MPEG's) due to the high degree of parallelism. For instance, in filter operations such as median filter, spatial filter, etc. the output of each pixel is some combination of the pixel values in a predefined window centered at this pixel, and filter operations on distant pixels can be performed independently and in parallel. In image compression, for example, if Discrete Cosine Transform (DCT) is applied to a 8 x 8 block of pixels, the DCT of the non-overlapped neighbouring blocks can be executed independently. It can be seen that the instructions for processing these pixels/blocks are identical. Therefore, these algorithms map naturally onto the Single-Instruction stream and Multiple-Data stream (SIMD) architecture.

It has also been reported that the processor speed increases 60% while memory speed increases only 7% every year

[Patterson96], and the memory capacity is quadrupled every three year [Hennessy94]. The growing processor-memory bandwidth gap cannot be ignored as it results in a loss of performance. To narrow the processor-memory gap, the usual approach at the system level is to use large caches. At the chip level, however, one approach is to embed memory modules in Application Specific Integrated Circuit (ASIC) processor arrays to remove the data transfer bottle neck at the I/O pins and bus level. The other approach is to integrate modular logic circuitries into the SRAM or DRAM memory to take advantage of the inherently high memory bandwidth while exploiting parallelism across the memory columns. The former is sometimes referred to as *logic-with-memory*, while the later *logic-in-memory*. Due to the length of the paper, only logic-in-memory processor arrays are discussed.

In the following sections, the logic-in-memory SIMD processor arrays are reviewed in section 2. The underlying algorithms will be discussed in section 3, followed by the conclusions in section 4.

2. LOGIC-IN-MEMORY SIMD PROCESSOR ARRAYS

2.1 C*RAM (C*RAM92) [Elliott92]

Computational RAM (C*RAM) is a memory-SIMD hybrid architecture where each column of memory has an associated processing element (PE). C*RAM was primarily designed with the goal of augmenting conventional RAM's (used in computer main memory and video buffers) with computation capability. The C*RAM's concept makes use of the large on-chip bandwidth to perform massively parallel bit-serial computations. There are two major functions in a C*RAM: memory and computation. When functioning as a memory device, C*RAM is read or written as part of the host processor address space. When functioning as a computing engine, all PE's execute operations (on their own local memory) in parallel, sequenced by a controller.

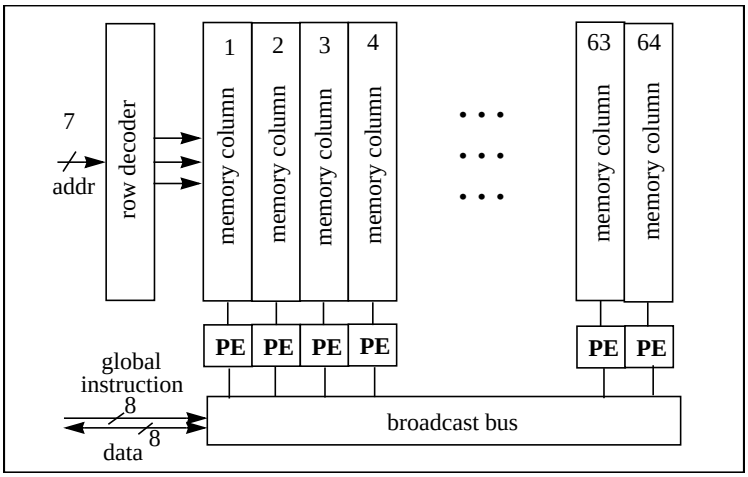

FIGURE 1: C*RAM architecture

The first C*RAM has been fabricated in a 1.2μm SRAM technology (Fig.1). There are 64 PE's attached to the 8Kb memory

column which may effectively be considered as 64 independent computing units (CU) having a PE and 128 bits of local memory. Each PE (Fig.2) has two 1-b registers, X and Y, and a 1-b ALU implemented as a multiplexer. Inputs to the bit-serial arithmetic ALU can be contents of registers X, Y and a memory bit M. An 8-b global instruction from the C*RAM controller chooses between the 256 standard/user-defined functions of the ALU. After each computation, the results are written back into the memory or the registers. Each X, Y register has a left and a right connection enabling data shifting and neighborhood communication between PE's.

In addition to the X,Y registers and ALU, the Write Enable (W) register permits conditional operations. The W register functions as a mask, blocking write back to undesired memory columns in various operations. This enables all the PE's in the C*RAM to participate in global operations without modifying the masked memory contents.

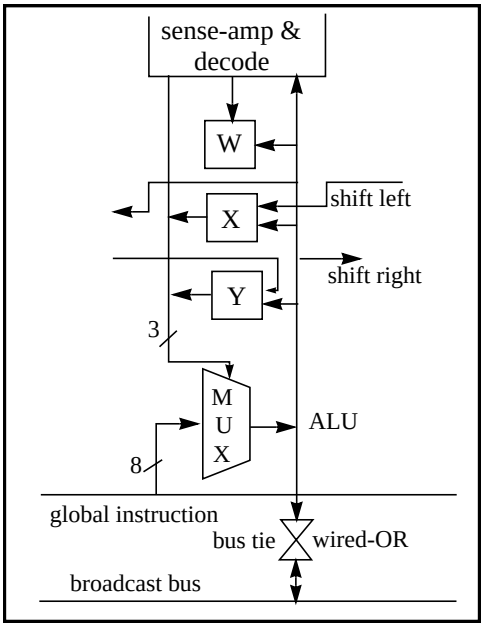

FIGURE 2: PE model

A bidirectional bus-tie circuit performs the wired-OR logic of all PE's local results. This global wired-OR is fed back to all the PE's. The same line may also be used by the C*RAM controller to broadcast a 1-b constant value to all the PE's.

There are a number of similar C*RAM's [Cojocaru93], [Foss96], and [McKenzie97]. In his first attempt to prove the C*RAM concepts, Cojocaru designed a 1Kb x 64PE C*RAM (C*RAM93). This chip was fabricated in a 0.8μm BiCMOS chip integrating on an SRAM core [Silburt92]. There are 64 PE's attached to the 64Kb of memory making 64 independent CU's. The architecture and PE design are similar to that of C*RAM92. The memory read/write cycle is 40ns, and in C*RAM mode, the *read-operate-write* cycle time is, however, 60ns. The *operate* cycle can also be sandwiched between the read/write cycle, which effectively results in a 20ns cycle. Therefore, when all 64 PE's are operating in parallel at 25MHz, C*RAM93 can achieve a peak performance of 3.2GIPS.

A C*RAM controller has been built for this chip. This controller provides the interface between the C*RAM and the host processor. Typically, C*RAM system is implemented as an extension card on the host processor expansion or local bus. C*RAM/controller integrated architecture is also being investigated. In this implementation, the controller interfaces the C*RAM to the PCI bus for PC environment. The PCI Interface Unit implements a 33MHz/32-bit PCI target interface protocol compliant with the PCI Local Bus Specification Revision. Burst data transfer and parity check are supported. Data and instruction transfer rate between host and the C*RAM is 132MB/s. The current controller prototype has been implemented in a Xilinx 4031 FPGA.

Another variation of C*RAM design was reported in [Foss96] (C*RAM96). A 16Mb DRAM is directly connected to 4096 PE's resulting in 4096 CU's each having 4Kb of local memory. The PE's are pitch-matched to groups of 4 DRAM columns. Recently in [McKenzie97], a 1024 PE C*RAM is implemented on a 16Mb IBM DRAM module (C*RAM97). Distinctions from previous C*RAM modules are: column redundancy is available and no extra control pins are needed, remaining JEDEC compatibility.

2.2 Integrated Memory Array Processor (IMAP94) [Yamashita94]

IMAP consists of a large memory cell array, serial shift registers to input/output images, row registers, and a 1-D processor array embedded into a video RAM (Fig.3). All PE's execute in a SIMD fashion. Video image input/output is performed independently using the internal processing clock via serial ports. The image data can also be accessed through random access port.

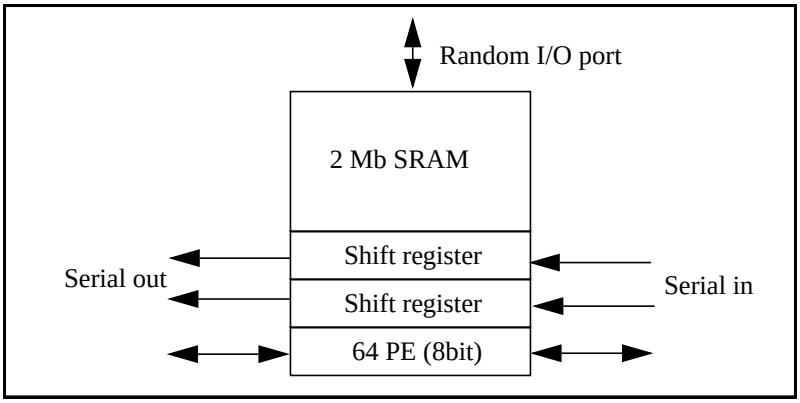

FIGURE 3: Integrated Memory Array Processor

Each IMAP chip consists of a 1-D array of 64 8-b PE's with 2Mb of SRAM for storing image (32Kb/PE), and two input/output shift registers. The IMAP does not have a sequencer nor a program memory. An instruction stream is provided from an off-chip controller. Instructions are latched in the pipeline registers and then broadcast to all PE's.

Each PE has an 8-b processor consisting of an ALU/shifter, 12 general registers, a data register, an address register, an inter-PE register, and a mask register. The mask register enables the PE's local memory to stay out of global operations. Data exchange among adjacent PE's is performed through inter-PE register. The address register is used to implement indexed addressing mode which are preferable when calculating histogram and Hough transform. With the memory data register and the address register, the memory read/write operation and ALU/shifter computation can be executed in simultaneously. Since all 64 PE's are operating in parallel at 40MHz, IMAP achieves a peak performance of 3.84GIPS, and 1.28GB/s on-chip processing bandwidth.

2.3 PC-RAM [Cojocaru94]

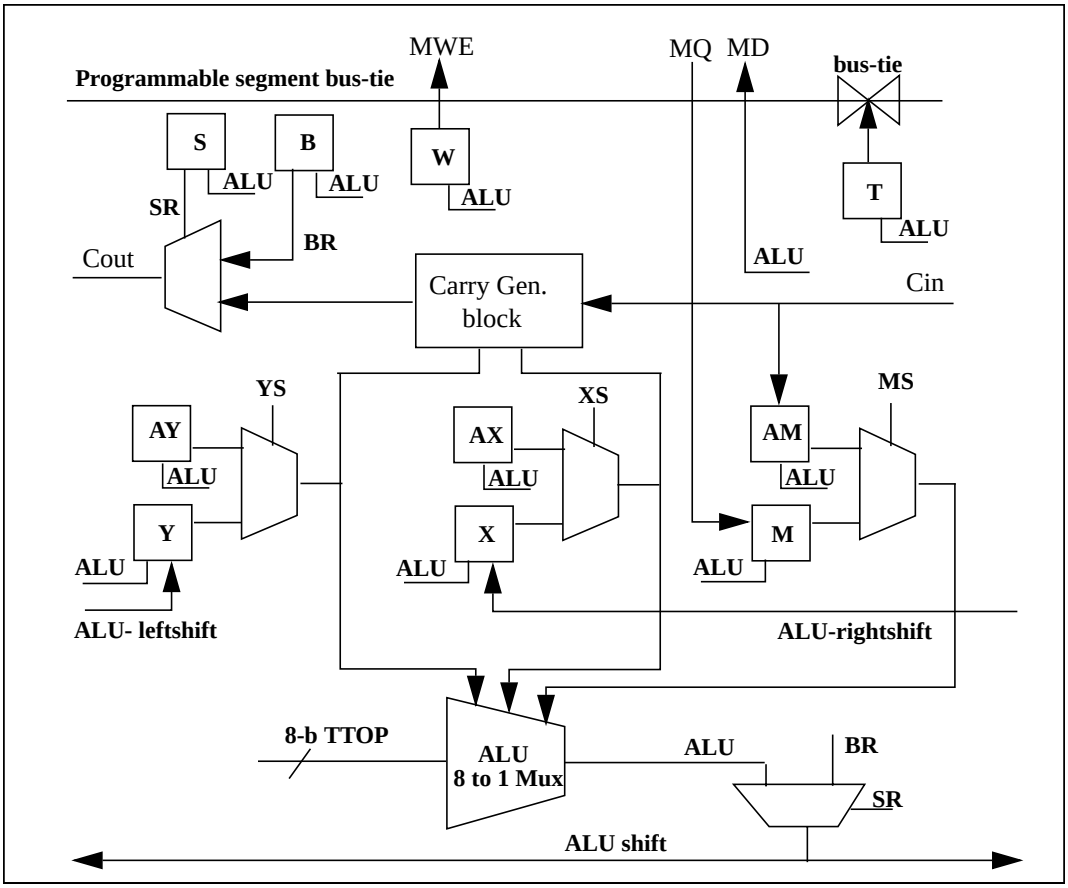

FIGURE 4: Extended PE architecture

In this 0.8μm SRAM-based PC-RAM, the PE is enhanced to allow non-transposed memory accesses and bit-parallel operations. In additional to the existing functionality of C*RAM92, this PC-RAM has the following features: *programmable segment bus-tie* (PSB), *programmable word boundary* (PWB), and *ripple-carry* circuit for addition, and the extended register set.

The baseline PE offers only two data registers X and Y. The third data input to the ALU, M is assumed to be registered in the

memory interface circuitry. In order to support bit-parallel multiplication, four new registers are defined in the extended PE (Fig.4). The M register is now the explicit source of data line M. M is a dual access latch and can be written by both the ALU and the local memory output MQ. Three more registers, AX, AY, and AM constitute the “alternate register set”. The word “alternate” indicates that only one in a register - alternate register pair can source the ALU at a time. The selection is made by three 2-to-1 multiplexors, controlled by three new global signals: XS, YS, and MS.

Multiples of 2 PEs can be grouped to form a CU which is implemented using a bus-tie register T placed at every other PE. The value stored in this register is used to disable the transmission gate connecting the PSB. All ALU outputs sharing the same segmented bus are wire-ORed, and the result is fed back to the registers of the corresponding CU. The PSB circuit is extensively used in zero-value checking and bit-extension operations.

A ripple-carry circuit is implemented by adding the carry generator (CG) block. Recall that for addition (or subtraction), two operations are required: carry (or borrow) and sum (or difference) generation. Inputs to the CG block are the *carry-in* (C_{in}) from the right-neighbour PE, and 2 other operands stored in registers X and Y (or alternatively, AX and AY). We note that the value of carry-in is set to 0 for addition and 1 for subtraction. The *carry-out* (C_{out}) will propagate to the left-neighbor PE if the PWB does not reached. Otherwise, the carry-out is stopped. The PWB is implemented by the Select register (S) and By-pass register (B), placed at every other PE. The bit-parallel C*RAM eliminates the need for a data transposer between itself and the host computer. The extended PE is, however, larger which effectively reduces the number of PE’s by 30%. When all 512 PE’s are operating in parallel at 25MHz, PC-RAM can achieve a peak performance of 25.6 GIPS.

2.4 SRC-PIM [Gokhale95]

An integrated SIMD project developed at the Supercomputing Research Center (SRC) in Maryland, USA, the Processor-in-Memory (PIM) is a chip that integrates a linear array of 64 PE’s with 2Kb of local memory. The PE processor is a bit-serial ALU. The processor is functionally divided into upper half and lower half where upper half executes the actual computation on the data, and the lower half executes routing and masking operations.

There are three primary registers, A, B, and C, supplying input to the ALU. At each clock cycle, the pipelined ALU can either load data from memory or store data to memory, but not both at the same time. Also, at each clock cycle, the ALU produces three outputs that can be either selected for storage (under mask control) or selected for recirculation. In addition, data can be sent to other processors via the routing network. The processor can input data through a multiplexer (MUX) from either the parallel prefix network, the global OR network, the partitioned OR network, or the internal mask/register control.

Interprocessor communication is conducted through a global OR network, a partitioned OR network, and a parallel prefix network. The partitioned OR network can execute logic-OR on groups of power of 2 of PIM chips. The PIM parallel prefix network allows PE’s at fixed positions to send data to groups of neighbouring PE’s in 16 configurations. When all 64 PE’s are

operating in parallel at 10MHz, SRC-PIM can achieve a peak performance of 0.64 GIPS.

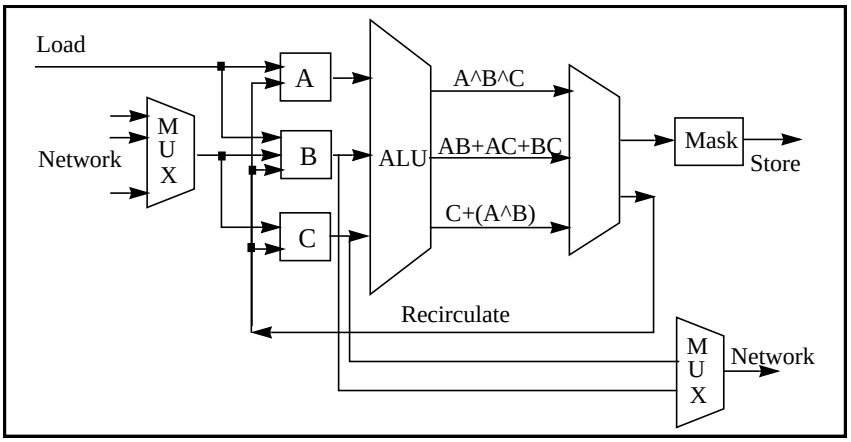

FIGURE 5: A processor-in-memory processor

2.5 CNAPS [Hammerstrom96]

CNAPS is a specialized 1-D SIMD processor array developed by Adaptive Solutions Inc. Several different processor arrays designed: CNAPS-1064 and CNAPS-1016, corresponding to 64-, 16-PE (or processor node - PN) on the 0.8μm SRAM technology. The CNAPS-2032 (32 PE's) was later designed on a 0.6μm SRAM.

As shown in Fig.6, the PE's are connected in a 1-D array with global broadcast from the sequencer to all the PE's (InBus), and a global output bus (OutBus) from all the PE's to the sequencer. Each PE is connected to its nearest neighbour via a 1-D Inter-PE bus. The architecture offers several arbitration techniques for the OutBus, the most common being sequential transmission. In addition to data transfer, the inter-PE bus is also used to signal the next PE during sequential arbitration. If multiple PE's transmit at once, the array performs a global AND of all PE outputs (the default transmission is all 1's). The OutBus goes into the sequencer, which can wrap the data back around to the InBus and/or write it to memory via a DMA channel (StdOut). The sequencer likewise can read data from the memory (StdIn) and put it on the InBus for broadcast to the PE array.

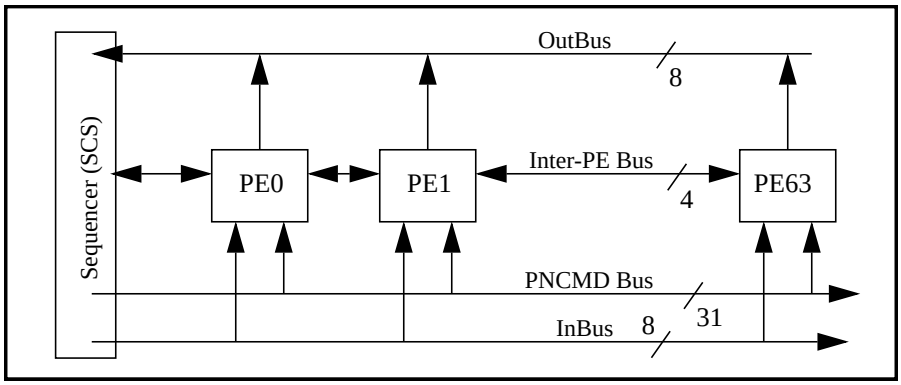

FIGURE 6: CNAPS Architecture

Each PE is a simple 16-b computing engine, similar to a DSP, each having the following internal components: 9-b x 16-b multiplier; 32-b adder; 32 element register file (16-b); 16-b shifter/logic operation unit; 12-bit memory address adder; and 4KB of SRAM, accessible as either 8-b or 16-b values.

When running at 25MHz, all 64 PE's of the 1064 can achieve a peak performance of 3.2 GIPS, while the 2032 can achieve a 2.56 GIPS maximum performance at 40MHz.

2.6 EXECUBE [Sunaga96]

EXECUBE, a combined DRAM and logic chip has been developed for massively parallel processing applications. A 4Mb CMOS DRAM technology is used to fabricate the chip. The 5-V 0.8μm technology merges 100K gate custom logic circuits and 4.5Mb DRAM onto a 14.7x14.7mm^2 die. The DRAM design is based on a 32K x 9-b self-consistent macro form. It has independent address inputs, data I/O ports, access control circuits, and redundancy fuses and elements. The logic part of the chip consists of 8 PE's and some broadcast logic circuits. Each PE and two DRAM macros (64KB) comprise a 16-b CU, and hypercube connections among eight CU's are made for the scalable MPP capability.

Each of the 8 CU's consists of 64KB memory, a 16-b PE, and DMA interconnect circuits. The CU's are arranged in a 3-D hypercube configuration. Each CU is connected to other three CU's by inter-CU buses. In addition, it also has a separate external port which enables a direct connection to similar links on other chips in almost any topology.

A broadcast logic circuit in the center of the chip works as an interface to an off-chip common control device. Inside the chip, it contains a broadcast interface which consists of common and separate buses to CU's. A 16-b multiplexed address/data bus is common to all CU's.

This architecture enables the chip run in either MIMD or SIMD mode. In MIMD mode, each of the 8 CU's fetches instructions and data from its own memory. The external common control device can send instructions into an arbitrary subset of the 8 CU's through the broadcast logic and interface. Each CU has a 16-b data read back output to the broadcast logic. The CU read back data can be shifted out of the chip at a rate of two bits per clock cycle. When running at 25MHz, each chip delivers a performance of 50 MIPS.

2.7 Parallel Image Processing (PIP-RAM) [Aimoto96]

The PIP-RAM consists of 128 PE's and 128 DRAM elements (with 4 redundant PE-DRAM pairs), a main word decoder (MWD), clock buffers, and control circuits (Fig.7). Each PE has a 8-b ALU, a shifter, 24 general purpose registers, 5 special purpose registers, and a DRAM element of 128Kb.

The DRAM element is divided into 8 memory units (MU), each of which includes 16Kb memory cell array and 64 latched

sense-amplifiers. A single sub-word line is selected by using main word and sub-word addresses in each MU, while an individual MU is activated by using an MU address. 8-b data, specified by the column addresses, are transferred between the activated MU and PE through a data line (DL) pair and dynamic data amplifier. While main word addresses are externally supplied and universal over all DRAM elements, sub-word, column, and MU addresses may be supplied either internally (by each PE) or externally.

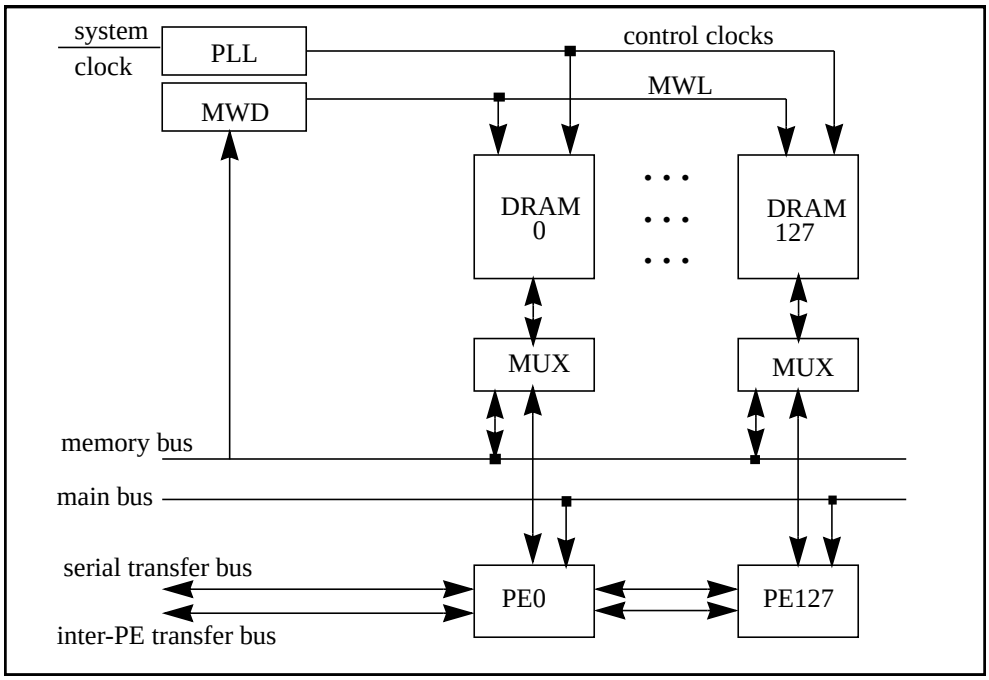

FIGURE 7: PIP-RAM architecture

Since all the 128 PEs operate in parallel at 30MHz, and read/write 8-b data from/to 128 DRAM elements at the same time, the PIP-RAM achieves 7.68 GIPS peak processing performance and 3.84GB/s peak memory bandwidth.

2.8 HDPP [Gealow96]

This design is based on large processor-per-pixel arrays implemented using IC technology. A high density parallel processor (HDPP) using DRAM is described. In this architecture, the layout pitch of the one-bit logic is matched to the pitch of the memory cells to form high density 2-D PE arrays.

The system design features an efficient control path implementation, providing high processing element array utilization without demanding complex controller hardware. Sequences of array instructions are generated by a host computer before processing begins, then stored in a simple controller. Once processing begins, the host computer initiates stored sequences to perform pixel-parallel operations.

In the HDPP, each of the 256 PE's is integrated with 128b DRAM columns in place of DRAM column decode logic. Three

registers A, B, and C provide inputs to a function generator. Control signals $f_7..f_0$ may specify any of the 256 three-input boolean functions. Register A also provides inputs to neighboring PEs. Values from PE to the North, South, East, and West are combined with function generator results according to control signals f_N, f_S, f_E, and f_W, respectively. Final results may be loaded into registers B, C, D, and/or E. Value stored in E register is used to control the write driver.

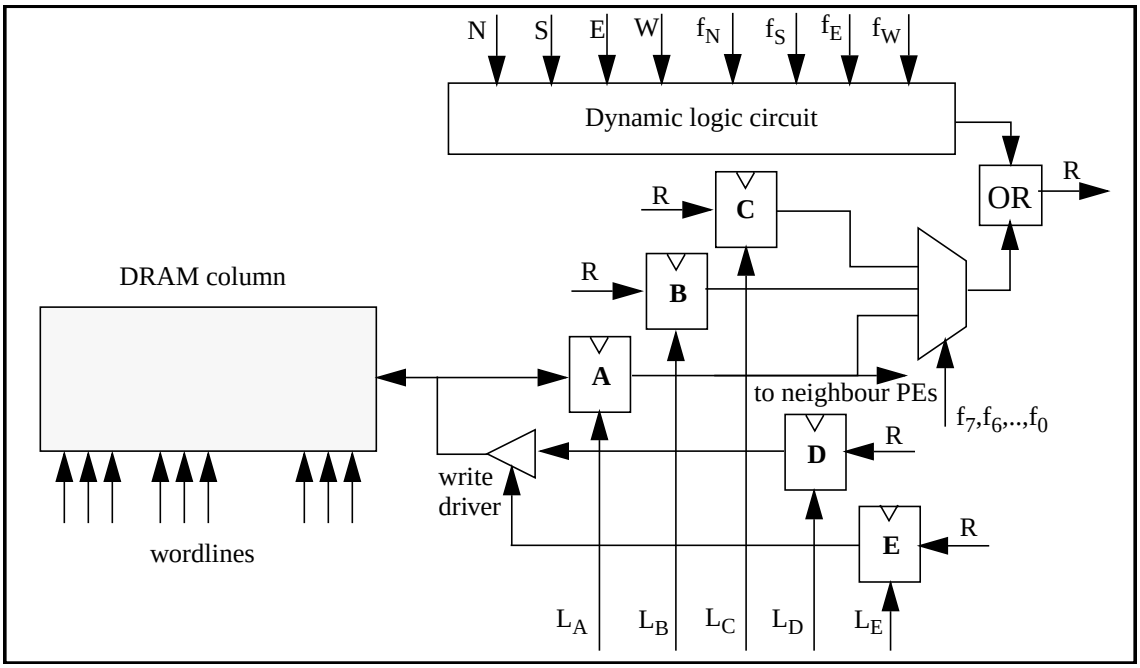

FIGURE 8: PE of the HDPP

Instructions for the PE array specify a memory operation, a Boolean function ($f_7...f_0$), and net work function ($f_N...f_W$), and register load signals (L_A, L_B, L_C, L_D, and L_E). Execution of an instruction begins with initiation of the memory operation, and the read or write value may be loaded into register A. Then the logic operation identified by the specified Boolean and network function is performed. Finally, the result of the logic operation may be stored into registers B, C, D, and/or E.

3. SUMMARY OF FEATURES

The features of the reviewed SIMD processor arrays are summarized in the table below. We note that the number of PE's in brackets are redundant PE's to improve yield. It can be seen that larger designs are heading towards the DRAM direction and have more PE's per chip. Single-bit ALU's are popular, but multiple-bit ALU's are quite significant in number. A popular single-bit PE design involves 3 primary registers feeding inputs to a multiplexor-like ALU. On the memory side, memory size is from 1Kb to 32Kb, which can accommodate one or more 8 x 8 pixel block. The interconnection networks are mostly linear. The SRC-PIM, however, has an interesting parallel prefix network which can be programmed to 1 of the 16 network configurations. The performance of the processor arrays very much depend on the clock rate and the available number of PE's, which, in turn, depends on the available silicon area and the ALU architecture. In the future, an 8-chip 4096-PE processor

array system running at 25MHz can effectively produce a maximum performance in the order of Tera instructions per second.

Table 1: Summary of Features

Name	No. of PE's	Bits/ALU	Memory per PE	Network	Performance (clk)
C*RAM92	64	1	128b SRAM	1-D or linear	---
C*RAM93	64	1	1Kb SRAM	linear	3.2GIPS (25MHz)
IMAP	64	8	32Kb SRAM	linear	3.84GIPS (40MHz)
PC-RAM	512	1 or 2n	480b SRAM	linear	25.6GIPS (25MHz)
SRC-PIM	64	1	2Kb SRAM	program'ed linear	0.64GIPS (10MHz)
CNAPS 64,16	64(16), 16(4)	16	4KB SRAM	linear	3.2 GIPS (25MHz)
CNAPS32	32(?)	16	4KB SRAM	linear	2.56GIPS (40MHz)
EXECUBE	8	16	64KB DRAM	3-D hypercube	50MIPS (25MHz)
PIP-RAM	128(4)	8	128Kb SRAM	linear	7.68GIPS (30MHz)
HDPP	256	1	128b DRAM	2-D mesh	2.56GIPS (10MHz)
C*RAM96	4096	1	4Kb DRAM	linear	---
C*RAM97	1024(?)	1	16Kb DRAM	linear	---

A few image and video processing algorithms are shown in Table 2. Due to the scope of the paper, these algorithms are not discussed.

4. CONCLUSIONS

This paper, by no means, represents a complete and thorough comparison of the mentioned processor arrays. It, instead, serves as a snapshot at some of the logic-in-memory processor array designs reported to date. A general trend towards large DRAM arrays is visible, together with a focus on image processing as a key application. Interprocessor communication schemes vary, but tend to be very simple.

Table 2: A few mage and video processing algorithms implemented on the SIMD processor arrays.

System using	Operation	Execution time (ms)	Reference
4 HDPP's or 1024 PE's/128Kb running at 10 MHz on 8-b 128 x 128 frame	smoothing and segmentation	4.8	[Gealow96]
	5 x 5 mean filtering	0.274	[Gealow96]
	optical flow	30.1	[Gealow96]
8 IMAP's or 512 PE's/16Mb running at 40MHz on 8-b 512 x 512 frame	3 x 3 average filtering	0.42	[Yamashita94]
	histogram	0.73	[Yamashita94]
4 1016-CNAPS's or 64PE's/8Mb running at 25MHz on 24-b 2k x 3k frame	10-pixel radius convolution filter	8,000	[Hammerstrom96]
8 1016-CNAPS's or 128PE's/16Mb running at 25MHz on 12-b 4k x 4k frame	double 9 x 9 convolutions	14,000	[Hammerstrom96]
8 C*RAM93's or 512 PE's/512Kb running at 25MHz on 8-b 512 x 512 frame	vector quantization	614	[Le94]
a C*RAM93 of 64 PE's/64Kb running at 25MHz on 8-b 64 x 64 sequence	motion estimation	11.6	[Le97]

REFERENCES

[Aimoto96] Y. Aimoto, T. Kimura, Y. Yabe, "A 7.68GIPS 3.84GB/s 1W Parallel Image Processing RAM integrating a 16Mb DRAM and 128 Processors", *ISSCC96*, pp.372-373, 1996.

[Cojocaru94] C. Cojocaru, *Computational-RAM: Implementation and Bit-Parallel Architecture*, Master's dissertation, Department of Electronics, Carleton University, Dec. 1994.

[Elliott92] D. G. Elliott, M. Snelgrove, and M. Stumm, "Computational-RAM: A Memory-SIMD Hybrid and Its Applications to DSP", *IEEE Custom Integrated Circuits Conference*, pp.30.6.1-30.6.4, Boston, May 1992.

[Foss96] R. Foss, "Implementing Application Specific Memory", *ISSCC'96*, Paper FP 16.1, 1996.

[Gealow96] J. Gealow, F. Herrmann, L. Hsu, and C. Sodini, "System Design for Pixel-Parallel Image Processing", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.4, no.1, pp.32-41, Mar. 1996.

[Gokhale95] M. Gokhale, B. Holmes, and K. Iobst, "Processing in Memory: The Terasys Massively Parallel PIM Array", *IEEE Computer*, 28(3), pp.23-31, Apr. 1995.

[Hammerstrom96] D. Hammerstrom and D. Lulich, "Image Processing Using One-Dimensional Processor Arrays", *Proceedings of the IEEE*, vol.84, no.7, pp.1005-1018, Jul. 1996.

[Hennessy94] J. Hennessy and D. Patterson, *Computer Organization and Design*, Morgan Kaufmann Publisher, 1994.

[Hwang93] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, and Programmability*, McGraw-Hill, 1993.

[Le94] T. M. Le, *Computational*RAM Implementations of Vector Quantization for Image and Video Compression*, Master's dissertation, Department of Electrical Engineering, University of Ottawa, Sept. 1994.

[Le97] T. M. Le, M. Snelgrove, and S. Panchanathan, "Fast Motion Estimation Using Feature Extraction and XOR Operations", submitted to Multimedia Hardware Architecture, SPIE'98, Jan. 1998.

[Mckenzie97] R.N. McKenzie, W.M. Snelgrove, and D.G.Elliott, "A 1024 Processing-Element Computational RAM", TRIO, Kingston, May 1997.

[Patterson96] D. Patterson, T. Anderson, and K. Yelick, "A Case for Intelligent RAM: IRAM", Hot chip 8, 1996.

[Sunaga96] T. Sunaga, H. Miyatake, K. Kitamura, P. Kogge, and Eric Retter, "A parallel processing chip with embedded DRAM macros", *IEEE Journal of Solid-State Circuits*, pp.1556-1559, vol.31, no.10, Oct.1996.

[Yamashita94] N. Yamashita, T. Kimura, Y. Fujita, Y. Aimoto, "A 3.84 GIPS Integrated Memory Array Processor LSI with 64 Processing Elements and 2Mb SRAM", *ISSCC'94*, pp.202-203, 1994.