An Effective Memory–Processor Integrated Architecture for Computer Vision

Youngsik Kim, Tack-Don Han, Shin-Dug Kim, and Sung-Bong Yang
Dept. of Computer Science, Yonsei University, Seoul, 120-749, Korea.
{yskim,hantack,sdkim,yang}@kurene.yonsei.ac.kr

Abstract
In this paper an effective memory–processor integrated architecture, called memory-based processor array (MPA), for computer vision is proposed. The MPA can be easily attached into any host system via memory interface. In order to measure the impact of the memory interface structure an analytical model is derived. The performance improvement on the proposed model for the memory interface architecture of the MPA system can be 6% ~ 40% for vision tasks consisting of sequential and data parallel tasks. The asymptotic time complexities of the mapping algorithms are evaluated to verify the cost-effectiveness and the efficiency of the MPA system.

1 Introduction
Because the current memory technology can support the gigabit DRAMs, a single memory chip would cover the memory volume needed for the computer systems in the future. A number of studies for the memory–logic integration have utilized both high internal memory bandwidth and the available chip density [1, 2, 3, 4, 5, 6].

In this paper an effective memory–processor integrated architecture, called memory-based processor array (MPA), for computer vision is proposed. It is an effective SIMD array which is based on the memory–processor integration structure. Thus, it can be easily attached into any host system from a personal computer to a multiprocessor. It can be considered as a portion of the single linear address space given for the host system. The MPA approach is to design a hybrid system that can be performed selectively for different levels of processing steps in vision tasks cost-effectively.

Especially, a couple of machine architectures for both the low level vision tasks and the intermediate to high level vision tasks are chosen as the MPA and the host system, respectively. These are integrated into a single machine tightly coupled via a single system bus. The low level and data parallel processing steps are executed by the MPA and the intermediate to high level processing steps with complex or sequential processing can be executed by the host system selectively. Furthermore any interaction by the program and the shared data between the host system and the MPA can be resolved by means of simple memory reads and writes. In this paper, parallel algorithms for thresholding by the best threshold (TBT) mapped onto the MPA are presented and the asymptotic time complexities of the algorithms are estimated with the big-oh notation. Also, the MPA system shows a significant performance improvement, illustrated with the proposed analytical model for the impact of the memory interface structure.

2 The MPA System
In this section, the MPA system architecture is described. An effective interfacing mechanism with any host system is specified as the basic building block of a complete system construction.

2.1 The Structure and Configuration
In the memory–processor integrated array approach, a complete MPA system structure consists of a host processor (HP), the HP memory module (HM), a system bus, and an MPA as shown in Figure 1. The MPA can be configured as the two different operational modes, i.e., simply as the memory and as the SIMD array.

In the MPA, a set of m PMs forms a pool of two-dimensional memory cells as shown in Figure 2. When the MPA system is used simply as for memory, the top and local decoders in Figure 2 can be operated as the conventional memory decoder and the HP can access every memory cell by the conventional memory row and column addresses. Each PU is constructed as an ALU including an adder and a shifter, a set of registers, and two bidirectional links to its immediate left and right neighboring PUs.
A model describing the impact of the memory interface can be derived from the operational model for the MPA system. Thus, the following variables are defined to develop this analytical model.

**Definitions**

The code and data blocks need to be defined to represent the size of code and the amount of data. Conceptually, the size in time corresponds to the amount of the time to execute any code block, divided by the execution time of a unit instruction.

- \( |A| \): the number of bytes in data parallel code blocks of high level instruction words.
- \( |A| \) and \( |B| \): the number of machine instructions required to execute the data parallel code blocks and the non-data parallel code blocks, respectively.
- \( |A^D| \) and \( |B^D| \): the number of bytes in the data needed in the \( A \) code blocks and the \( B \) code blocks, respectively.
- \( |A^D| \cap |B^D| \): the number of bytes in the shared data between \( A^D \) and \( B^D \).

A set of system parameters is defined as follows.

- \( T_S \): the time that a PU for both the conventional SIMD array and the MPA executes a basic operation. It is assumed that the PU executes complex operations such as multiplication and division in multiples of \( T_{ex}^{S} \), i.e., \( 4T_{ex}^{S} \) and \( 16T_{ex}^{S} \), respectively.
- \( T_H \): the time that the HP executes a basic arithmetic or logic operation. The time for the complex operation is assumed to be a multiple of \( T_{ex}^{H} \).
- \( T_P \): the time to send or receive a unit of data (one byte) between the host system and the SIMD system only for the conventional SIMD systems.

Each of \( T_{ex}^{S} \) for each PU, \( T_{ex}^{H} \) for the HP, \( T_P \) cannot exceed \( O(1) \). However, in order to evaluate the analytical model the ratio of \( T_{ex}^{S} \) to \( T_{ex}^{H} \) and the ratio of \( T_P \) to \( T_{ex}^{S} \) are defined as \( r_1 \) and \( r_2 \), respectively.

To show how effective the interaction mechanism between the HP and the MPA is, two timing parameters, the \( T_{ex}^{S} \) and \( T_{ex}^{H} \) of the MPA system are assumed to be equal to those of the conventional SIMD system, respectively. Parallel execution time \( \tau^{SIMD} \) by the conventional SIMD systems for any pair of code blocks, i.e., \( A \) and \( B \) can be obtained as

\[
\tau^{SIMD} = \frac{|A^D|}{|A^D|} + \frac{|B^D|}{|B^D|} + \frac{|A| + |B| + |A^D| \cap |B^D|}{|A|} T_P, \tag{2}
\]

where \( |A^D| \cap |B^D| \) is a portion of \( A^D \) for the host system and a portion of \( B^D \) for the SIMD array. Parallel
execution time \( \tau_{MPA} \) by the MPA system for any pair of \( A \) and \( B \) can be represented as

\[
\tau_{MPA} = \frac{\tau^H + \tau^S}{\tau^C - \tau^S}.
\]  

(3)

By the impact of the memory interface, the performance improvement \( P1 \) of the MPA system comparing with the conventional SIMD systems can be specified as

\[
P1 = \frac{(\tau_{SIMD} - \tau_{MPA}) \times 100}{\tau_{SIMD}} = \frac{\left( |A| + |A^D \cap B^P| \right) \tau_r \times 100}{\left( |A| + |A^D \cap B^P| \right) \tau_r} = \frac{|A| + |A^D \cap B^P|}{|A|}. \tag{4}
\]

The performance improvement on the impact of the memory interface is exemplified by considering a mixed set of vision tasks consisting of parallel and sequential tasks in Section 3.2 later.

3 Low Level Computer Vision Tasks

To clarify the impact of the memory interface structure, a detailed discussion of a vision task, called thresholding by the best threshold (TBT) \[7\], is presented.

Some of the assumptions made for the applications are first described. Each PU can process \( N \times 1 \) partitioned subimage. In the striped partitioning, the image pixels are divided into groups of complete columns, each MPA chip module is assigned for one such group, and each PU is assigned for the pixels of a column. TBT considered in this paper is composed of a couple of parallel tasks and a sequential task. The parallel tasks are histogramming and thresholding required to be executed by the MPA and the sequential task is to find the best threshold required to be executed by the HP. Algorithm 2 of Figure 3 to execute TBT on the MPA system calls sequentially three procedures, Histogramming-MPA, Finding-Best-Threshold-HP, and Thresholding-MPA.

Histogramming, which is not a window processing, is a time consuming task. The procedure, Histogramming-MPA, represented in data parallel codes is shown in lines 4–16. Histogramming-MPA is based on the algorithm presented in \[9\]. Specifically, the number \( \tau_{histogram}^{MPA} \) of computation steps to execute histogramming by the MPA system with \( P \) PUs, \( P = N \), is obtained as

\[
\tau_{histogram}^{MPA} = N \left\{ \left( \sum T_{exe}^S \right) + 16T_{exe}^S + N \left( T_{exe}^S + T_{exe}^S + T_{exe}^S \right) \right\} = O(N^2). \tag{5}
\]

The procedure Finding-Best-Threshold-HP is a sequential order of calculations followed by the computation of the best threshold \( BT \) executed by the HP as in lines 17–27 of Figure 3. Finding the BT in this procedure is based on Otsu's algorithm \[8\]. Let \( R[1], \ldots, R[G] \) represent the histogram probabilities of the observed gray values \( 1, \ldots, G \), respectively. Here, the BT is the threshold which is chosen in such a way that the weighted sum of the group variances should be minimized.

Therefore, the number \( \tau_{BT}^{MPA} \) of computation steps in finding the BT on the HP is obtained as

\[
\tau_{BT}^{MPA} = \begin{array}{c}
(7G + 1)T_{exe}^S + 6(G + 1)T_{exe}^S + 2G + 1 \\times T_{exe}^S + G \times T_{exe}^S = O(G). \tag{6}
\end{array}
\]

To perform the thresholding for the MPA system, a parallel algorithm in a pseudo code is shown in lines 28–37. Therefore, the number \( \tau_{threshold}^{MPA} \) of computa-
from /2/5/6 to /4/0/9/6/, since the ratio is

$$\text{p o n e n t s}.$$ 

Finding/-Best/-Thr eshold/-HP in Algorithm /2 by the con v en tional SIMD system and
structure/. MP A system can b e op erated as a passiv e memory fer the shared data to the lo cal memory b ecause the
Ho w ev er/, the HP of the MP A system needs not trans/-
HP and the SIMD system/. Th us the ratio of the
Figure /4/: The p erformance impro v emen ta s
Nvary /.

![Figure 4: The performance improvement as r1, r2, and N vary.](image)

calculation/. Th us/, the p erformance impro v emen t from
structure o v er the con v en tional bac k/{end in terfac/e-
dd
MP A thr eshold
the execution times/, The MP A system could gain the p erformance im/-
and
Nvar y /.

$$N = 4096$$

$$N = 512$$

$$N = 256$$

$$N = 1024$$

$$N = 512$$

$$N = 256$$

$$N = 128$$

$$N = 64$$

$$N = 32$$

$$N = 16$$

$$N = 8$$

$$N = 4$$

$$N = 2$$

$$N = 1$$

$$r_1, r_2, \text{ and } N \text{ vary.}$$

can b e reduced to

$$\frac{N^2 + G\lfloor \log_2 N \rfloor}{2N^2 + G\lceil \log_2 N \rceil}.$$ 

In the conventional SIMD systems environment, the SIMD array needs the additional time to fetch the original image data from the host system. Also, the host system needs the additional $G\lceil \log_2 N \rceil$ data transfer time before executing of the procedure Finding-Best-Threshold-HP. However, the HP of the MPA system need not transfer the shared data to the local memory because the MPA system can be operated as a passive memory structure.

The execution times, $r^{C-SIMD}$ and $r^{MPA}$, of TBT in Algorithm 2 by the conventional SIMD system and the MPA system can be calculated as following components.

$$r^H = r^{HP} + r^{T_{FBT}},$$

$$r^A = r^{histogram} + r^{threshold},$$

$$r^T = \left( |A| + N^2 + G\lceil \log_2 N \rceil \right) T_{tr}.$$ 

Since $|A|$ is very small, it can be ignored in the calculation. Thus, the performance improvement from

the memory interface structure for Algorithm 2 can be expressed using equations (8), (9), and (10) as

$$P_1 = \frac{r^{C-SIMD} - r^{MPA}}{r^{C-SIMD}} \times 100 = \frac{r^T}{r^H} \times 100$$

$$= \frac{\left( N^2 + G\lceil \log_2 N \rceil \right) r_2}{N(2N + 20) + (7G + 20)r_1 + \left( N^2 + G\lceil \log_2 N \rceil \right) r_2}.$$ 

(11)

Figure 4 shows that $P_1$ rises drastically as $r_2$ increases from 0.2 to 2.0 and $P_1$ rises very slowly as $r_1$ decreases from 1.0 to 0.1 while $N$ varies from 256 to 4096. Thus for a given problem $P_1$ is dominated by $r_2$ which can emphasize the additional transfer time of the shared data only for the conventional SIMD sys-
tem. The performance improvement according to the memory interface architecture of the MPA system can be 6% ~ 40% while $N$ varies from 256 to 4096. Thus, the MPA system can achieve a performance gain from the memory interface structure in performing a con-
vention al program composed of a mixed set of code blocks.

4 Conclusions

The MPA system is to design a hybrid parallel system that can perform selectively for different types of parallelism in computer vision tasks. The MPA sys-
tem is constructed by integrating two different types of parallel architectures tightly into a single machine. Operational interaction between these two systems can be performed via the conventional subroutine calling mechanism to execute data parallel code on the MPA. This research is to design an underlying base architecture that can be executed for a broad range of vision tasks from the low level to the high level pro-
cessing. The MPA has been shown to provide significant performance improvement and cost effectiveness for parallel applications having a mixed set of tasks.

References


