REDUCED PROCESSING ELEMENT ARCHITECTURE FOR PARALLEL LOCAL IMAGE PROCESSING

Po-Ning Chen, Yung-Sheng Chen and Wen-Hsing Hsu

Institute of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan 30043, R.O.C.

Key Words: image processing, local operation, convolution.

ABSTRACT

The object of this paper is to propose new architecture which can reduce the number of processing elements for parallel local image processing under the premise of real-time performance. For large-sized local image processing, this architecture will save much space as it is suitable for being designed into VLSI chip. For example, the traditional parallel architecture will use 9 PEs for a 3×3 convolution, while the Reduced Processing Element Architecture (RPEA) only requires 2 PEs to achieve the real-time performance.

一個減少處理單元數目的局部影像處理運算器之設計

陳伯寧  陳永盛  許文星

國立清華大學電機工程研究所

摘  要

在本論文中，針對影像處理高達化的需求，提出一套新的理論及架構。此架構能在即時處理的時間內，以最少數目的處理單元發揮最有效之利用去完成影像的後端運算。例如傳統的架構對 3×3 的後端運算，通常使用九個處理單元，而本架構只需三個處理單元即可達到即時處理的要求。本局部影像處理運算器之設計適合製成超小型積體電路，並且與傳統式設計相比較，其複雜度亦大為降低。

INTRODUCTION

Many widely used operators in the image processing, such as local smoothing, Sobel operator and Laplacian operator, can be replaced by a local convolutional operator. But the common constraint of these operators is the slow processing speed of software due to millions of operations. This has limited the applications of these operators to only speed-don’t-care areas or simple binary images. For example, a 3×3 convolution requires 9 multiplications and 8 additions. Hence an N×N image requires a total of 9 N^2 multiplications and 8 N^2 additions. This will take much time to finish the convolutional operations with a general-purpose computer.

However, with recent advances in VLSI technology, we can implement a complex circuit into a chip to obtain a system much faster than the traditional general-purpose computer. For example, it takes only one or two time periods to perform a convolutional operation by hardware, while just a single multiplication performed by 8086 CPU takes about seventy time periods. Hence, from the standpoint of hardware, the potential of real-time applications of these operators in the area of large-sized gray-scale images increases by

+ Also with Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.
an astonishing rate.

Recently, relative researches have been done in this area, e.g., the systolic 2-D convolution chip by Kung and Song [4] and the ISP chip by Fukushima [1]. (Their architecture is extended from the concepts of Cytocomputer proposed by Sternberg [6].) Since the above architecture consists of so many PEs, the cost of these chips can not be reduced to a reasonable and popular price if the mask size is large.

Because a multiplier is often slower and more expensive than an adder, other researchers have used some tricks to replace the multiplications by simpler operations in order to reduce the designing complexity and system cost. For example, Kittler [2] used only additions and subtractions to implement edge detecting operation of Sobel 3x3 mask [3]. And Mcllroy [5] implemented edge detecting operation by a 2x2 Robert operation without multiplications. But the above tricks are only feasible in special cases and can not be extended to other local processings.

However, in some special cases that require large mask sizes, if the traditional architecture is used, the number of PEs will increase to an impractically large amount. For example, the template matching between two patterns may use the mask size of 8x8. If ISP PE-adding architecture is used, then 64 PEs or 16 chips will be required for implementation. Although Fukushima can use PE-saving architecture to save the number of PEs, the performance will decrease four times. This is because the pipeline cycle of PEs is the same as the memory cycle in the ISP chip. However it is common that the bandwidth of the memory is much slower than that of PEs until now. Hence if we use less PEs and add buffers to balance the bandwidth gap between PEs and memories, we can achieve the same performance as the above architecture.

It is fair to say, the speed of traditional pipeline architecture such as an ISP chip is in proportion to the pipeline cycle or the memory cycle, but the speed of reduced processing element architecture (RPEA) is purely proportional to the speed of PEs. With recent VLSI technology, the implementation of a very fast special PE is possible. In order to make the best use of PEs, we propose a new architecture, RPEA, whose processing speed only depends on the speed of PE itself.

In the next section, we will explain our opinions by means of simple mathematics.

THE BASIC IDEA OF RPEA

Before introducing the basic idea, we first define the following variables:

- Image size: \(m \times m\) (in \(m^2\) pixels)
- \(N_s\): number of computations required per local operation
- \(N_{\text{delay}}\): number of \(m\)-stage delay lines required in the system, where \(m\) is the size length of the image
- \(N_p\): number of PEs in the system
- \(T_s\): processing time per computation
- \(T_{\text{delay}}\): shifting time of delay line
- \(T_e\) (external): the time required for the whole local operation
- \(T_r\) (real-time): 1/30 or 1/60 sec for TV camera
- \(C \times C\): the mask size of the convolution \((N_r = C^2)\).

Firstly, considering the constraint of real-time performance, we have the following inequality:

\[
N_{\text{delay}} \times T_{\text{delay}} + m^2 \times T_s < T_r. \tag{1}
\]

The first term of Eq. (1) represents the time to fill out the delay buffers. The consideration of fill-out time is the same as that of the traditional Cytocomputer or any data-fetch pipe.

The second term in Eq. (1) represents the basic computational time for the \(m \times m\) image.

Secondly, we describe the time required for the whole local operation which depends on the number of PEs used, by the following formula:

\[
T_e = \frac{N_p \times T_s}{N_s}. \tag{2}
\]

Usually \(T_e \gg N_{\text{delay}} \times T_{\text{delay}}\), so from Eq. (1),

\[
T_e < \left(\frac{T_r - N_{\text{delay}} \times T_{\text{delay}}}{m^2}\right) \Rightarrow T_e < \frac{T_r}{m^2}. \tag{3}
\]

From Eqs. (2) and (3),

\[
\frac{N_p \times T_s}{N_s} < \frac{T_r}{m^2} \Rightarrow m^2 \frac{N_p \times T_s}{T_r} > 1. \tag{4}
\]

Then from Eq. (4), we conclude that the number of PEs to achieve real-time performance is at least \(m^2 N_p \times T_s / T_r\).

For example, if \(m = 256\), \(T_s = 100\) ns, \(T_r = 33.3\) ms and for \(3 \times 3\) convolution, then

\[N_p = \frac{256^2 \times 9 \times 100\text{ ns}}{33.3\text{ ms}} = 1.77 \Rightarrow 2.\]

From the above description, we get two performance evaluation diagrams, which are shown in Figs. 1 and 2.

In Fig. 1, we assume that if the memory cycle is \(T_s\) sec and the size of weighting coefficient mask is \(C \times C\) \((N_r = C^2)\), there must exist PEs with the processing speed greater than \((T_s / C)^2\) sec/computation. In general, this is always true for a small \(C\). Therefore, from Fig. 1, only 2 PEs are needed for a \(3 \times 3\) convolution and 3 PEs for a \(4 \times 4\) convolu-
tion to achieve the performance of 33.3 ms, if the capability of PEs is 100 ns/computation.

Figure 2 shows the comparison of numbers of PEs between RPEA and the traditional architecture under the same real-time performance. It shows that the number of PEs of the traditional architecture is increasing by a function of $C$, where $C \times C$ is the size of the weighting coefficient mask. However, the number of the PEs of RPEA increases probably as slow as the linear function. (From Eq. (4), it is true that $O(256^2 \times C^2 \times 100 \text{ns}/33.3 \text{ms}) = O(0.2 \times C^3).$) This feature justifies our approach.

So it is obvious that under the premise of real-time performance and widely-applied local processing, we are able to reduce the number of PEs; that is, to reduce the cost and chip-space.

In the next section, we will introduce the method for implementation of the convolutional operator with the least number of PEs.

### THE FUNDAMENTAL ARCHITECTURE

The basic formula of the convolutional operation is

$$g_{m} = \sum_{k} h_{mk} f_{m+k, m-1} \tag{5}$$

where $[g_{m}]$ is the convolved image, $[f_{m}]$ is the image to be convolved and $[h_{mk}]$ is the matrix of weighting coefficients. In order to clearly introduce the RPEA, we use the simplest case, 2 PEs doing a 3x3 convolution, as an example.

#### 1. The arrangement of the image

Before implementing Eq. (5), we must first decide which two (or more) convolutional computations should be done simultaneously.

In ISP chips, the nine multiplications of the 3x3 convolution are done by different PEs and then are summed by nearby (or outside-the-PE’s) adder (See Fig. 3(a)). However, in our architecture, the nine products of each 3x3 convolution are computed by the same PE and then are accumulated by the inside accumulator (See Fig. 3(b)).

Figure 4(a) shows the step flows of using 2 PEs to implement a 3x3 convolution in a 10-by-10 image.

The basic data flow of Fig. 4(a) is that the even-address result of the convolutional operation is done by PE 2, such as $g_{2n, 2m}, g_{2k, 2m}, \ldots$, etc, and the odd-address result, by PE 1, such as $g_{1n, 1m}, g_{1k, 1m}, \ldots$, etc. From Fig. 4(a), the data pointed by the arrows must be saved for the computation of the next step. For instance, at step 3, both PE 1 and PE 2 use image data $f_{2n}, f_{2k}$, and $f_{2j}$. And at step 4, PE 2 re-uses the same image data $f_{2n}, f_{2k}$, and $f_{2j}$. Since
### Fig. 3(a). ISP architecture.
- LB: line buffer
- DCB: data control buffer
- PE: processing element

### Fig. 3(b). RPEA architecture.

**Original Image**

<table>
<thead>
<tr>
<th>g01</th>
<th>g02</th>
<th>g03</th>
<th>g04</th>
<th>g05</th>
<th>g06</th>
<th>g07</th>
<th>g08</th>
<th>g09</th>
<th>g10</th>
</tr>
</thead>
<tbody>
<tr>
<td>g11</td>
<td>g12</td>
<td>g13</td>
<td>g14</td>
<td>g15</td>
<td>g16</td>
<td>g17</td>
<td>g18</td>
<td>g19</td>
<td>g20</td>
</tr>
<tr>
<td>g21</td>
<td>g22</td>
<td>g23</td>
<td>g24</td>
<td>g25</td>
<td>g26</td>
<td>g27</td>
<td>g28</td>
<td>g29</td>
<td>g30</td>
</tr>
<tr>
<td>g31</td>
<td>g32</td>
<td>g33</td>
<td>g34</td>
<td>g35</td>
<td>g36</td>
<td>g37</td>
<td>g38</td>
<td>g39</td>
<td>g40</td>
</tr>
<tr>
<td>g41</td>
<td>g42</td>
<td>g43</td>
<td>g44</td>
<td>g45</td>
<td>g46</td>
<td>g47</td>
<td>g48</td>
<td>g49</td>
<td>g50</td>
</tr>
<tr>
<td>g51</td>
<td>g52</td>
<td>g53</td>
<td>g54</td>
<td>g55</td>
<td>g56</td>
<td>g57</td>
<td>g58</td>
<td>g59</td>
<td>g60</td>
</tr>
<tr>
<td>g61</td>
<td>g62</td>
<td>g63</td>
<td>g64</td>
<td>g65</td>
<td>g66</td>
<td>g67</td>
<td>g68</td>
<td>g69</td>
<td>g70</td>
</tr>
<tr>
<td>g71</td>
<td>g72</td>
<td>g73</td>
<td>g74</td>
<td>g75</td>
<td>g76</td>
<td>g77</td>
<td>g78</td>
<td>g79</td>
<td>g80</td>
</tr>
<tr>
<td>g81</td>
<td>g82</td>
<td>g83</td>
<td>g84</td>
<td>g85</td>
<td>g86</td>
<td>g87</td>
<td>g88</td>
<td>g89</td>
<td>g90</td>
</tr>
<tr>
<td>g91</td>
<td>g92</td>
<td>g93</td>
<td>g94</td>
<td>g95</td>
<td>g96</td>
<td>g97</td>
<td>g98</td>
<td>g99</td>
<td>g100</td>
</tr>
</tbody>
</table>

**MASK**

- $h_{01}$
- $h_{02}$
- $h_{03}$
- $h_{11}$
- $h_{12}$
- $h_{13}$
- $h_{21}$
- $h_{22}$
- $h_{23}$

### Fig. 4(a). The step flows of $3 \times 3$ convolution in two-PE system.
- A and X in the access control indicate accessing image data from LB and not-accessing image data from LB, respectively.

<table>
<thead>
<tr>
<th>Step Number</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Control</td>
<td>A</td>
<td>A</td>
<td>X</td>
<td>A</td>
<td>A</td>
<td>X</td>
<td>A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PE 2</th>
<th>$f_{01}$</th>
<th>$f_{11}$</th>
<th>$f_{21}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE 1</td>
<td>$f_{01}$ $f_{02}$ $f_{03}$ $f_{11}$ $f_{12}$ $f_{22}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PE 2</td>
<td>$f_{02}$ $f_{03}$ $f_{04}$ $f_{12}$ $f_{13}$ $f_{14}$ $f_{22}$ $f_{23}$ $f_{24}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PE 1</td>
<td>$f_{03}$ $f_{04}$ $f_{05}$ $f_{13}$ $f_{14}$ $f_{15}$ $f_{23}$ $f_{24}$ $f_{25}$ $f_{26}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
the image data is scanned only once, we must add a second buffer to store the data \( f_{ob}, f_{13}, \) and \( f_{23} \) for the computation of step 4, and use a switch to route the data into the proper PEs at the proper time, as shown in Fig. 3(b).

To summarize, there must be three delay lines between buffers and the image, so that we can put the data of three consecutive lines at the same address into the buffers, such as \( f_{ob}, f_{13}, \) and \( f_{23} \). And a second buffer must be added to store the image data for the computation of the next step as explained above (See Fig. 3(b)).

2. The weighting coefficient queue

In general, the weighting coefficient is stored in the inside-PE memory. But in the RPEA the weighting coefficient is stored in another mask queue (Fig. 3(b)), and dynamically piped into the PEs. Because the mask queue is designed to be adjustable, that is, the stages of the queue are changeable through software commands, the RPEA can do different mask sizes of the convolution, even the number of PEs is fixed. This is another design trick of RPEA.

3. The arrangement and control of switch

As stated previously, PE 1 always uses the data in the first buffer, so we need to link PE 1 to the first buffer directly. PE 2 uses the first buffer data at step \( 3j \) and step \( 3j+2 \), and uses the second buffer data at step \( 3j+1 \), where \( j \) is just an index changing from 0 to \( \lfloor m^2/3 \rfloor \) (See Fig. 4(a)). Hence, we must add a switch before PE 2 to route the data of the first buffer into PE 2 at step 3 and step \( 3j+2 \), and to route the data of the second buffer into PE 2 at step \( 3j+1 \).

4. The generalized architecture

The generalized architecture of a \( C_n \times C_\alpha \) convolution (\( C_n \) may or may not be equal to \( C_\alpha \)) with \( K \) PEs is illustrated in Fig. 5. In summary, we add another variable-length buffer to implement the data re-use, and put several switches to route the data into the appropriate PEs at the right time.
**Fig. 6.** The step flows of 5x5 convolution in two-PE system. A and X in the access control indicate accessing image data from LB and not-accessing image data from LB, respectively.

**Fig. 7.** The 3x3 convolution prototype implemented on a single board.

**Table 1.** The processing speed of 3x3 convolution in 8086 assembly language on NEC 9801E2 using 8 MHz clock; and that of the prototype

<table>
<thead>
<tr>
<th></th>
<th>Smoothing</th>
<th>Edge detection</th>
<th>Laplacian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>3 sec</td>
<td>12 sec</td>
<td>12 sec</td>
</tr>
<tr>
<td>Prototype</td>
<td>69 ms</td>
<td>69 ms</td>
<td>69 ms</td>
</tr>
</tbody>
</table>

**Fig. 8(a).** The image directly from camera.

**Fig. 8(b).** The result of smoothing performed by the convolution prototype.

**Fig. 8(c).** The result of edge detection performed by the convolution prototype.

architecture (that is, the number of PEs is fixed now), we still can do different sizes of the convolutional mask by decreasing or increasing the
stages of variable-length buffers and mask queues, if not considering the processing speed. For example, the step flows of the 5×5 convolution with 2 PEs are shown in Fig. 6. In this case, we must re-use the data at step 3, step 4, and step 5, so the buffer must have totally 15 stages (in comparison with a 3×3 convolution, only 3 stages are required in the two-PE system).

The only problem of adjusting the stages is that the performance of our system may not be real-time if the size of the weighting coefficient mask is too large. For example, Fig. 1 shows that a two-PE system can achieve real-time for 3×3 convolution but not for 4×4 convolution.

**EXPERIMENTAL RESULT**

To confirm the RPEA, we make an experimental system of the two-PE system, as shown in Fig. 7. Because of the slow speed of TDC 1008 chip (above 100 ns per computation) and in order to get the high experimental fidelity, we use 200 ns as the cycle of PEs. We get a performance of 69 ms for a 256×256×8 image, which is close to the theoretical performance.

In Table 1, we show the performance of the prototype and the software which is written in 8086 assembly language. Furthermore, the processing result is shown in Fig. 8.

**CONCLUSIONS**

Due to certain limits in practice, it is often difficult to build a system with a large amount of PEs. Therefore, we propose an optimal PE-saving architecture in this paper for this purpose.

Under recent VLSI technological advances, it is possible to implement very fast special PEs in a VLSI chip. So the possibility of the real-time RPEA system is increasing. This will surely facilitate the gray-scale applications especially for the local operation of large mask size.

Finally, we summarize the basic features of our RPEA:

1. Use least number of PEs to achieve real-time performance.
2. Minimize the idle time of PEs.
3. The RPEA can use the same number of PEs to do different size of convolution, such as 3×3, 4×4, ..., convolutions using 2 PEs.
4. Easy to implement the RPEA to VLSI chip, especially in the area of the large-sized local image processing.

**REFERENCES**


Discussions of this paper may appear in the discussion section of a future issue. All discussions should be submitted to the Editor-in-Chief.

*Manuscript Received: April 21, 1987; 
Revision Received: June 10, 1987; 
Accepted: June 26, 1987.*