Parallel Execution of a Connected Component Labeling Operation on a Linear Array Architecture

KUANG-BOR WANG, TSORNG-LIN CHIA[*], ZEN CHEN[+]
AND DER-CHYUAN LOU
Department of Electrical Engineering
Chung Cheng Institute of Technology
Taoyuan, 335 Taiwan
[*]*Department of Information Management*
Ming Chuan University
Taoyuan, 333 Taiwan
[+]*Institute of Computer Science and Information Engineering*
National Chiao Tung University
Hsinchu, 300 Taiwan

This work presents a novel parallel algorithm and architecture for finding connected components in an image. Simulation results indicate that the proposed algorithm has an execution time of N^2+6N-4 cycles for an $N \times N$ image using an architecture containing 4 parallel processors. The proposed hardware can process a 128×128 image in 0.8574 ms and uses only 4 processors, compared to 0.85 ms and 128 processors for the work of Ranganathan *et al.* [14], and 94.6 ms and 16384 processors for the MPP [22]. Among the advantages of the novel architecture are modularity, expandability, regular data flow, and simple hardware. These properties are extremely desirable for VLSI implementations. Additionally, the execution time of the algorithm is independent of its image content; thus, it is quite flexible.

Keywords: connected component labeling, parallel algorithm, parallel processing, linear array, processing element

1. INTRODUCTION

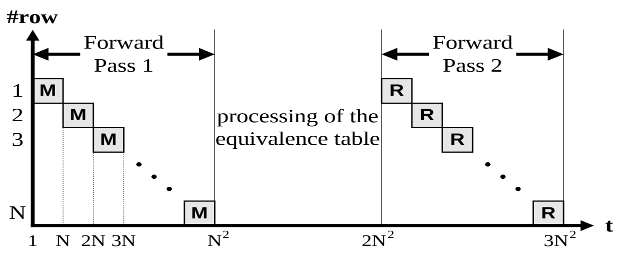
Connected component labeling is a conventionally used image processing procedure. The input for this procedure is a binary [1-3] or gray level image, while its output is a symbolic image. All connected pixels with the same input value are assigned the same identification label, which is a unique name or index of the region to which the pixels belong.

Since the shape of the object contour can be arbitrary, connected component labeling is essentially a global operation; thus, it involves significant data computation and communication between the pixels in the image. To solve the above problem, several sequential algorithms [1-8] and parallel algorithms [9-14] have been proposed. The clas-

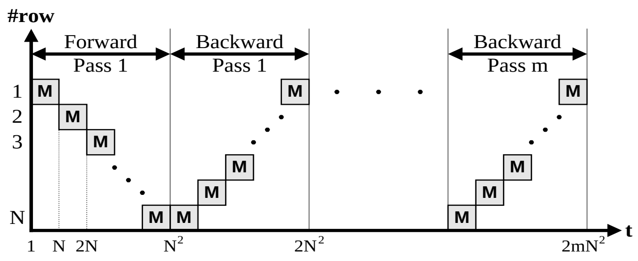
Received June 1, 2001; revised December 24, 2001; accepted February 1, 2002.
Communicated by Kuo-Chin Fan.

sical algorithm makes only two forward raster scan passes through the image but requires a large global table for recording equivalence labels, as described by Rosenfeld and Pfaltz [5]. In the first forward pass, the image is processed from top to bottom and all of the equivalences are stored in the equivalence table. Meanwhile, the minimum current equivalent label is extracted and recorded in the equivalence table for each label. The second forward pass is then used to reassign each label with its minimum equivalent label. Although this standard technique works, it is unsuitable because of the large number of distinct regions in a large image. In such cases, the number of distinct regions can become rather large. The iterative algorithm by Haralick [6], unlike the classical algorithm, stores no equivalence table, and consists of an initialization step and a sequence of top-down label propagation followed by bottom-up label propagation iterated until label change ceases. However, the execution time of the algorithm depends on the image content. A space-efficient two-stage algorithm that uses a local equivalence table was presented by Lumia *et al.* [7]. The algorithm uses a forward raster scan for the first stage and a backward raster scan for the second stage. Both stages use similar operations performed during the period of execution. The equivalence table is declared for each row of the image. For each row, the processing time can be divided into two passes. In the first pass, pixels are assigned a label according to their connectivity with their neighbors, as in [5]. Any label equivalences are found and included in an equivalence table for processing at the end of the row. The equivalence table determines the minimum equivalent label of each existing label, and a second pass is made to correct the labels assigned to the pixels in the row. By resolving the regional equivalences in each row, the bottom row of each region is guaranteed that the minimum label is assigned to any pixel in the region. The key advantage of Lumia's algorithm is that it processes label equivalences on a row-by-row basis. Lately, Suzuki *et al.* [8] proposed a sequential local operation algorithm. The algorithm combines ones in the [5-7]. Although, it has been shown that the algorithm is more efficient than the conventional sequential algorithms in [5-7]. But, the execution time is directly proportional to the number of pixels in connected components in an image. Therefore, the execution time of the algorithm is dependent on the image content. Furthermore, Nicol [15] presented a systolic array using the sequential two-pass raster scan algorithm. The algorithm is similar to that of Lumia except that label equivalences are found *on-the-fly* rather than stored in a table which is processed at the end of each row. When two labels are merged, label equivalence is generated and used to relabel the N most recently labeled pixels in the raster scan, producing an image of size $N \times N$. This algorithm has a processing time of $2N^2 + 2N$ cycles. Ranganathan *et al.* [14] recently proposed a systolic algorithm and architecture. The algorithm's total required processing time is $2N^2 + 5N$ cycles, for processing an $N \times N$ image using an architecture containing N parallel processors. Fig. 1 illustrates the time-space diagrams of the five algorithms in [5-7, 14], and [15] for an $N \times N$ image. The time duration M (Merge) collects the connectivity for each region in the image executed by PE's in this period. The task of assigning new labels for each pixel in the image is executed in time duration R (Re-label). Table 1 lists the differences in the required PEs number and execution time for [5-7, 14, 15] and the novel method.

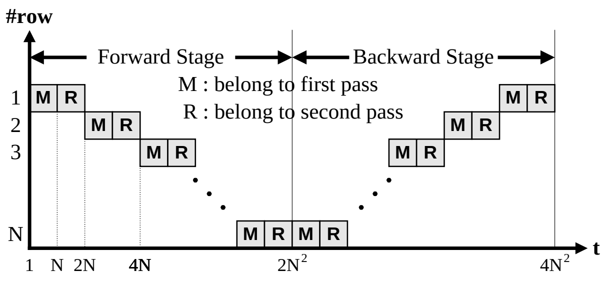
Connected component labeling can be implemented either sequentially or in parallel. From the perspective of real-time applications, most of the existing sequential algorithms for connected component labeling do not meet the execution time requirements. Hence, the use of a parallel algorithm is indispensable. Table 2 lists several parallel algorithms using a 2-D array. Clearly, those architectures are too expensive (for example N^2 processing elements), their hardware circuits are complex (such as, pyramid or tree architectures) and difficult to manufacture. The number of PEs must be reasonable to make the parallel architecture suitable for VLSI implementation. Consequently, for practical applications, the number of PEs must be fixed and usually must also be smaller than the $N \times N$ ($N \geq 1024$) image. A parallel algorithm for connected component labeling is presented herein. Although, the algorithm is shown to have an execution time of $N^2 + 6N - 4$ cycles for an $N \times N$ image, its architecture contains only 4 parallel processors, suitable for VLSI implementation, and its execution time is independent of the image content; making it quite flexible. The proposed hardware can process a 128×128 image in 0.8574 ms using only 4 processors, while the method developed by Ranganathan *et al.* [14] takes 0.85 ms and 128 processors, and the MPP method [22] requires 94.6 ms and 16384 processors, as shown in Table 3.



(a) Rosenfeld's method [5].

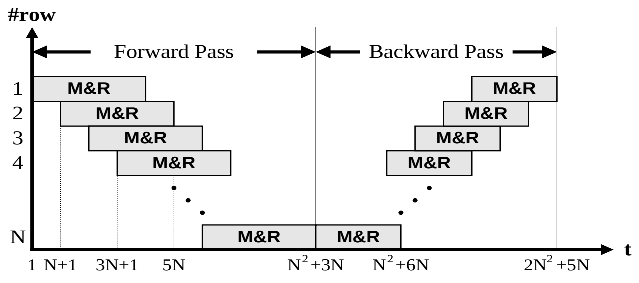


(b) Haralick's method [6].

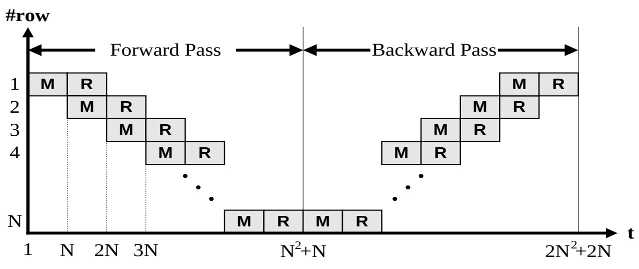


(c) Lumia's method [7].

Fig. 1. Time-space diagrams of the five connected component labeling algorithms in [5-7, 14, 15].



(d) Ranganathan's method [14].



(e) Nicol's method [15].

Fig. 1. (Cont'd) Time-space diagrams of the five connected component labeling algorithms in [5-7, 14, 15].

Table 1. Comparison among the different algorithms in [5-7, 14, 15] and the novel method.

Author	Hardware Architecture	PE	Execution Time
Rosenfeld [5]	1 PE + N^2 memory	1	$3N^2$ cycles
Haralick [6]	Iterative	1	$2mN^2$ cycles
Lumia [7]	1 PE + N memory	1	$4N^2$ cycles
Nicol [15]	1 PE + N register	1	$2N^2 + 2N$ cycles
Ranganathan [14]	Systolic Array	*N*	$2N^2 + 5N$ cycles
The Novel Method	Linear Array	4	$N^2 + 6N – 4$ cycles

m: the number of using backward pass

Table 2. Comparison among several parallel algorithms using 2-D arrays.

Author	Hardware Architecture	PEs	Time Complexity
Nassimi [16]	$N \times N$ Mesh	N^2	O(N)
Miller [17]	$N^{2/3} \times N^{2/3}$ Mesh	$N^{4/3}$	$O(N^{2/3})$
Kumar [18]	$N \times N$ Mesh of Tree	N^2	$O(\log N^2)$
Lim [19]	N^2 EREW PRAM	N^2	O(logN)
Miller [20]	Pyramid	N^2	$O(N^{1/2})$
Cypher [21]	Hypercube	N^2	O(logN)
Cypher [22]	N^2 EREW PRAM	N^2	$O(\log N^2)$
Agrawal [23]	N^2 EREW PRAM	N^2	$O(\log N^2)$

Table 3. Time comparison among the works of Ranganathan[14], Nicol[15], Cypher[22] and the novel method.

Item	Author	Hardware architecture	PEs	Clock cycle	Execution Time
1	The Novel Method	Linear Array	4	50ns	0.8574ms $N^2 + 6N - 4$ cycles
2	Ranganathan[14]	Systolic Array	128	25ns	0.85ms $2N^2 + 5N$ cycles
3	Nicol[15]	1PE + N Registers	1	40ns	1.32ms $2N^2 + 2N$ cycles
4	Cypher[22]	*MPP*	16384	100ns	94.6ms $O(\log N^2)$

The rest of this paper is organized as follows: section 2 presents a parallel connected component labeling algorithm based on four scanning passes. Section 3 then presents the corresponding linear array architecture and also analyses the time complexity of the approach. Conclusions are finally made in section 4.

2. BASIC CONCEPTS AND THE PARALLEL ALGORITHM

Rosenfeld [5] and Lumia [7] demonstrated that connected component labeling could be completed within two scan passes of an image. Based on their concepts, significant space remains in time-space diagrams for these algorithms (see Fig. 1), thus providing the possibility of performance improvement using a few processing elements and a well-chosen parallel data flow. To overcome the complexity of connectivity relations in the parallel manner, the proposed algorithm adopts the concept of divide-and-conquer. By using the PE number for connected component labeling, the algorithm can be classified as follows. First, the typical algorithm only using a processing element (PE) for connected component labeling was described in Lumia *et al.* (refer to Fig. 1(c)). Assigning pixels are labeled according to the connectivity with their neighbors. When any two labels are merged, the label equivalences are generated and entered into the equivalence table of the row in duration M. The equivalence table is then resolved to determine the minimum equivalent label. The label of the pixel is then re-labeled in duration R according to the content of the equivalence table. Following the re-label cycle, the label of the pixel is guaranteed to be correct with the minimum label [7]. Consequently, when two PEs exist, PE_1 of processing row 1 in cycle R simultaneously enables PE_2 of processing row 2 in cycle M. The 3×3 neighborhood is considered herein. The connected component label of the preceding neighboring pixels in the row must be ready for use by the current pixel. Consequently, the input data flow of the preceding row must be two cycles ahead of the current row (refer to Fig. 2). The time-space diagram is shown in Fig. 3.

Recently, Nicol [15] designed a systolic array using a PE and N register hardware architecture. Their approach resembles the method using two PEs. The merger of two labels generates a label equivalence record which is used to re-label the N most recent labeled pixels in the raster scan, for an image of size $N \times N$ (refer to Fig. 1(e)). In Fig. 1(e), Nicol's algorithm contains significant space remaining in the time-space diagrams.

P(x-1 , y-1)	P(x , y-1)	P(x+1 , y-1)
P(x-1 , y)	P(x , y)	

(a)

P(x-1 , y)	P(x , y)	
P(x-1 , y+1)	P(x , y+1)	P(x+1 , y+1)

(b)

Fig. 2. Adjacent pixels of pixel P(x, y) (a) in the forward stage and (b) in the backward stage.

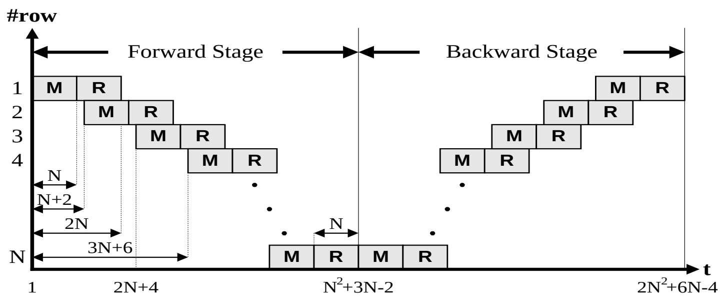


Fig. 3. Time-space diagram of the method using two PEs.

This excess space allows for possible performance improvement; if PE_2 of processing row 2 and PE_1 of processing row 1 can record its label equivalence, and the label equivalences of PE_1 can at the same time propagate to PE_2, then the duration M of row 2 can advance. Therefore, when two PEs are used, the label equivalence produced from row 1 can propagate to row 2, and PE_1 and PE_2 can then conduct parallel processing in duration M. Fig. 4 shows the time-space diagram. However, the start of execution in row 3 must be delayed until the ending time of row 1 with PE_1.

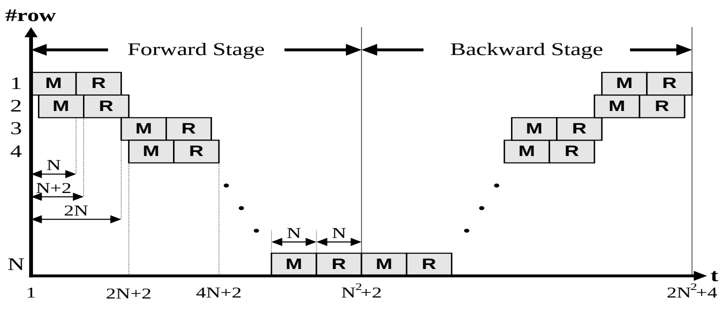


Fig. 4. Time-space diagram of the method using two PEs, where PE_2 can receive label equivalence from PE_1.

Similarly, for three PEs, label equivalence produced in row 1 can propagate to rows 2 and 3, and label equivalence produced from row 2 can propagate to row 3. PE_1 of processing row 1, PE_2 of processing row 2 and PE_3 of processing row 3 can then work in parallel. Based on the above-mentioned concept, the time-space diagram of using N PEs is shown in Fig. 5. From the perspective of hardware implementation, the Nth PE has to synchronously deal with itself, and the label equivalence is propagated from the 1st to the (N – 1)th PE. Consequently, the hardware circuit becomes very complex, and the processing time increases owing to the increasing N. Table 4 lists the different algorithms using the PE number. For example, when N PEs are used, the Nth PE must have N – 1 data paths which receive label equivalence from (N – 1)'s PE to the 1st PE, and must then store these label equivalences sequentially in the label equivalence table. The Nth PE thus needs (N – 1) cycles per pixel and N^2 memory cells to complete the task. When the space-time diagram (refer to Fig. 5) is used, the time complexity of the algorithm is easy to analyze. For each PE, it will process 8N-4 pixels, but in reality (N – 1) cycles are required per pixel, meaning that the total execution time is $8N^2 - 12N + 4$ cycles for N PEs. Hence, from Table 4, the optimum approach is to adopt a set of two PEs for parallel execution.

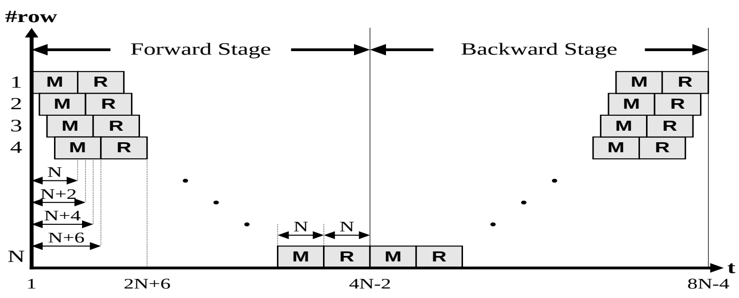


Fig. 5. Time-space diagram of the method using N PEs, where the Nth PE can receive label equivalence from the 1st PE to (N – 1)'s PE.

Table 4. Comparison of the algorithms using the PE number.

PEs	Cycle/Pixel	Hardware Architecture	Execution Time
1	1	1 PE + N memory	$4N^2$ cycles
2	1	2 PEs + 2N memory	$2N^2 + 4$ cycles
3	2	3 PEs + 3N memory	$\frac{8}{3}N^2 + 16$ cycles
4	3	4 PEs + 4N memory	$3N^2 + 36$ cycles
5	4	5 PEs + 5N memory	$\frac{16}{5}N^2 + 64$ cycles
N	N – 1	N PEs + N^2 memory	$8N^2 - 12N + 4$ cycles

Fig. 4 illustrates the adopted algorithm using a set of two PEs, showing that this algorithm also has some space remaining in its time-space diagram. While row 2 performs a re-labeling cycle in duration R, each pixel of row 2 is guaranteed to be labeled with the minimum label, and the other sets of two PEs (for row 3 and row 4) can then move forward and perform the merging cycle in duration M. The time-space diagram is shown in

Fig. 6. In this situation, the algorithm only uses 4 PEs. Hence, a parallel connected component labeling algorithm that needs four processing elements is proposed herein. This algorithm employs a processing element for the pipeline operations performed on the pixels in each row, and is divided into forward and backward stages, each of which consists of the first pass (duration M) and second pass (duration R). In the forward stage, the image is processed from top to bottom and left to right. The first pass of the forward stage (such as, Pass 1) assigns labels to pixels according to their connectivity with their neighbors. When two labels are merged, a label equivalence record formed by (L-old, L-new) is entered into the label equivalence table and propagated to the next PE. Simultaneously, the connecting pixel is labeled with the minimum label. The label equivalence table uses CAM (Content Addressable Memory) distributed-memory providing a PSMU (Parallel Search and Multiple Update) operation. The label equivalence table aims to determine the minimum equivalent labels. Meanwhile, the second pass of the forward stage (for example, Pass 2) is made to correct the labels assigned to the pixels in the row, using the information in the processed label equivalence table. Following the forward stage, the bottom row of each region is guaranteed to be labeled with the minimum label assigned to any pixel in the region.

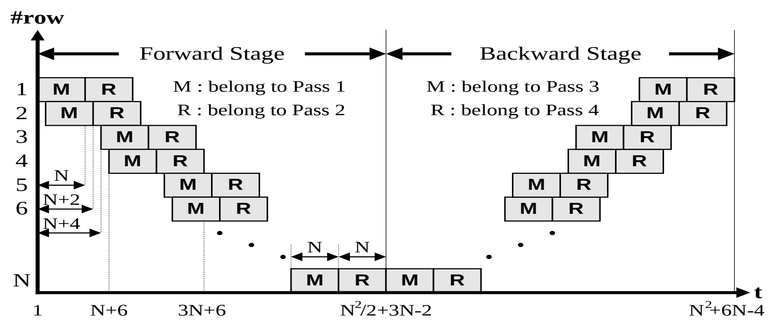


Fig. 6. Time-space diagram of the novel method.

The backward stage consists of two scan passes of Pass 3 (duration M) and Pass 4 (duration R), and the image is processed from bottom to top and left to right. Passes 3 and 4 are similar to the Pass 1 and 2 operations performed for each row during the forward stage, respectively. The backward stage can correctly label with their minimum labels.

Algorithm Proof

The two-stage works is formally described below in Property 1, Property 2 and the Lemma. Lumia *et al*. used a forward raster scan for Pass 1 and a backward raster scan for Pass 2, with the operations performed in each pass being similar. Each Pass is again divided here into first and second passes. The first pass assigns labels to pixels according to the connectivity with their neighbors, and any label equivalences found are placed in a table. The equivalence table is aims to find the minimum equivalent labels, and a second pass of each row is needed to relabel the pixels according to the equivalence table. Meanwhile, Nicol’s algorithm removes the equivalence table of Lumia’s algorithm, and uses an *on-the-fly* technique to relabel the set consisting of the N most recently processed pixels, for an N × N image. In the novel algorithm, label equivalences are processed in parallel, and all label equivalences are stored in an equivalence table after Pass 1 (similar to Lumia's first pass of Pass1) and Pass 3 (similar to Lumia's first pass of Pass2). Meanwhile, in Pass 2 (similar to Lumia's second pass of Pass1) and Pass 4 (similar to Lumia's second pass of Pass2), the pixels are relabeled according to the minimum equivalent labels found. The algorithm proof closely follows that in [7] and [15].

Property 1. Let C denote a top-maximally connected component. In Pass 1 and Pass 2, if the leftmost pixel of the top row of C is assigned label L_0, then

(a) no pixel of C is assigned a label less than L_0, and
(b) all pixels in the bottom row of C are assigned label L_0.

Proof: Lumia et al. has proven some properties when one PE is used for forward raster scan: "Let C be a top-maximally connected group of ones. In Pass 1, if the leftmost pixel of the top row of C is given label i, then (a) no pixel of C is given a label less than i, and (b) all pixels of the bottom row of C are given label i." The novel algorithm uses four PEs, where every PE can record its label equivalence, PE_2 can receive the label equivalences of PE_1, PE_4 can receive the label equivalences of PE_3, and every PE can relabel the pixels according to its equivalence table. Therefore, following Pass 2, the bottom row of each region is guaranteed that the minimum label assigned to any pixel in the region.

Property 2. Let C denote a top-maximally connected component. If Passes 3 and 4 are applied to the image, and the bottom row of C is all labeled L_0, then every member of C will be labeled L_0.

Proof: Similarly, when one PE is used for the backward raster scan, Lumia et al. have proven this property: "Let C be a top-maximally connected group of ones. If after Pass 2 operates on the bottom row of C, the bottom row is all labeled i, then every pixel of C will be labeled i by Pass 2." If every PE can record its label equivalence, PE_2 can receive the label equivalences of PE_1, and PE_4 can receive the label equivalences of PE_3, then every PE can relabel the pixels according to its equivalence table. Therefore, following Pass 4, the first row of each region is guaranteed with the minimum label assigned to any pixel in the region.

Lemma Let C represent a maximally connected component. If the leftmost pixel of the top row of C is assigned the label L_0 by Pass 1, then following Passes 2, 3 and 4, all pixels of C will have the label L_0.

Proof: If C is maximally connected, then it is also top-maximally connected, the Lemma is thus made true by Properties 1 and 2.

In these four passes, the image rows are scanned and processed in parallel. During the forward stage, let the set of adjacent pixels of the pixel P(x, y) be defined by N_f(x, y) = {P(x – 1, y – 1), P(x, y – 1), P(x + 1, y – 1), P(x – 1, y)} (refer to Fig. 2). The label L(x, y) is calculated for pixel P(x, y), x, y ∈ [1, N], based on P(x, y) and N_f(x, y). Each row is scanned from left to right and top to bottom. However, the connected component label of the preceding neighboring pixels in the row must be ready for the current pixel to use, and the input data flow of the preceding row must also be two cycles ahead of the current row. In this scanning fashion, the label L(x, y) for pixel P(x, y) is calculated in PE_i after it receives the information PL(x – l, y – 1), PL(x, y – 1), and PL(x + l, y – l) calculated in PE_{i-1} during the last three cycles and PL(x – 1, y) calculated in PE_i during the last cycle, where PL(x – 1, y – 1) includes the pixel value P(x – 1, y – 1) and label L(x – 1, y – 1) of the pixel at (x – 1, y – 1). Similarly, during the backward stage, the label L(x, y) is calculated based on P(x, y) and N_b (x, y) = { P(x – 1, y + 1), P(x, y + 1), P(x + 1, y + 1), P(x – 1, y) }. Each row is scanned from left to right and bottom to top. Thus, the values and labels of the required adjacent pixels are always ready before calculating the connected component label of the current pixel.

The input image is an N × N multivalue image I = {P(x, y), x, y ∈ N} and the output image is a label map that assigns a unique label (for example, an integer number) to each maximally connected component in the image. During the processing time, the label equivalence table EQTABLE represents a set of equivalence relations that formed by (L-old, L-new). The L-old label is the old label of one pixel belonging to the region, which is relabeled using the L-new or new label. The label equivalence table uses CAM (Content Addressable Memory) distributed-memory providing a PSMU (Parallel Search and Multiple Update) operation. Meanwhile, a CAM module allows content to be updated with an L-new, based on the condition that the L-old is equal to its content.

The novel algorithm below uses the following notations.

A(*x*, *y*): the set of pixels belonging to N_f (x, y) or N_b(x, y) with the same input value of the pixel P(x, y).

$A(x, y) = \{A_f(x, y), A_b(x, y)\}$,

$A_f(x, y) = \{P(m, n)|P(m, n) \in N_f(x, y), P(m, n) = P(x, y)\}$ (in the forward stage),

$A_b(x, y) = \{P(m, n)|P(m, n) \in N_b(x, y), P(m, n) = P(x, y)\}$ (in the backward stage).

AL(*x*, *y*): the set of labels belonging to the label of the pixel in the set *A*(*x*, *y*), where the label does not equal zero.

$AL(x, y) := \{L(m, n)|P(m, n) \in A(x, y)\}$, where L(m, n) is the label of the pixel P(m, n).

ALE(*x*, *y*): the set of minimum L-new labels among the equivalence records with an L-old label.

ALE(*x*, *y*) := {min (L-new)|L-old = L(m, n), L(m, n) ∈ AL(*x*, *y*), (L-old, L-new) ∈ EQTABLE}.

ADD: the function that adds an equivalence relation record (L-old, L-new) to the label equivalence table.

SEND: the function that sends an equivalence relation record (L-old, L-new) to the label equivalence table of the next PE.

Algorithm

```
for x = 0 to N + 1, y = 0 to N + 1 do      /Initialization/
   if (x < 1 or y < 1 or x > N or y > N)
      then P(x, y) = ∞    (a large number)
/Forward stage: raster scan the image from top to bottom and left to right, /
/                     the PE_i processes image row y, where y = i + 4j, i = 1, ..., 4, and
                      j = 0, 1, ..., ⌈N/4⌉ −1. /
/Backward stage: raster scan the image from bottom to top and left to right, /
/                     the PE_i processes image row y, where y = (N + 1) – i – 4j, i = 1, ..., 4, and
                      j = 0, 1, ..., ⌈N/4⌉ −1. /.
```

```
Main_program
begin                    /the work for image row y by PE_i /
   for Pass = 1 to 4 do
      for x = 1 to N do
         switch (Pass)
            case 1: Label_New(1);
            case 2: Relabel( );
            case 3: Label_New(3);
            case 4: Relabel( );
      end for;
   end for;
end begin

Label_New(k)
begin
   if A(x, y) = φ   then
/The PE_i processes image row y, where y = i + 4j, i = 1, ..., 4, and j = 0, 1, ..., ⌈N/4⌉–1. /
      if k = 1 then L(x, y) = y * N + x;      /Give a new label/
               else L(x, y) = min (EQTABLE(L-old = L(x, y)) ;
      end if
      L-old = L-new = L(x, y);
   else
      if k = 1 then L-old = max(ALE(x, y))
               else L-old = min (EQTABLE(L-old = L(x, y));
      end if
      L-new = L(x, y) = min(ALE(x, y));
   end if
   Proc_Label( )
end begin

Relabel( )
begin
   if A(x, y) = φ   then L(x, y) = min (EQTABLE(L-old = L(x, y)) ;
                    else L(x, y) = min(ALE(x, y));
   end if
end begin

Proc_Label( )
begin
   ADD (L-old, L-new, EQTABLE);
   if (i is odd) then SEND (L-old_i, L-new_i) to PE_{i+1} (EQTABLE);
                        /(L-old, L-new) will be propagated to the next PE./
                  else ADD (L-old_{i-1}, L-new_{i-1}, EQTABLE);
                        /Receive (L-old, L-new) from the previous PE./
   end if
end begin
```

3. LINEAR ARRAY ARCHITECTURE

The above parallel connected component labeling algorithm can be implemented using a linear array architecture to meet real time requirements. The linear array comprises 4 processing elements (PE) to process a binary or gray level image with $N \times N$ pixels, as shown in Fig. 7. During the forward stage, PE_i processes the image row $i + 4j$, where $i = 1, ..., 4$, and $j = 0, 1, ..., \left\lceil \frac{N}{4} \right\rceil - 1$. Similarly, during the backward stage, PE_i processes the image row $(N + 1) - i - 4j$. Initially, the image buffer stores the input binary or gray level image, and as the process continues, the buffer stores the label map. Generally, PE_i receives data from PE_{i-1} and calculates the label $L(x, y)$ in row y in every cycle. Because the data flow shifts one pixel per cycle, and the labels of four adjacent pixels must be completed beforehand, the input image in the image buffer is skewed, meaning that the clock cycle of PE_i is two cycles behind that of PE_{i-1}. Hence, for the pixel $P(x, y)$ at time t, the labels of $P(x - 1, y - 1)$, $P(x, y - 1)$, and $P(x + 1, y - 1)$ are calculated and passed from PE_{i-1} to PE_i during the forward stage at times $t - 3$, $t - 2$, and $t - 1$, respectively. The label of $P(x - 1, y)$ is obtained and stored in PE_i at time $t - 1$. Similarly, PE_i receives the labels of $P(x - l, y + 1)$, $P(x, y + 1)$, $P(x + 1, y + 1)$, $P(x - l, y)$ and $P(x, y)$ before it calculates the label $L(x, y)$ during the backward stage.

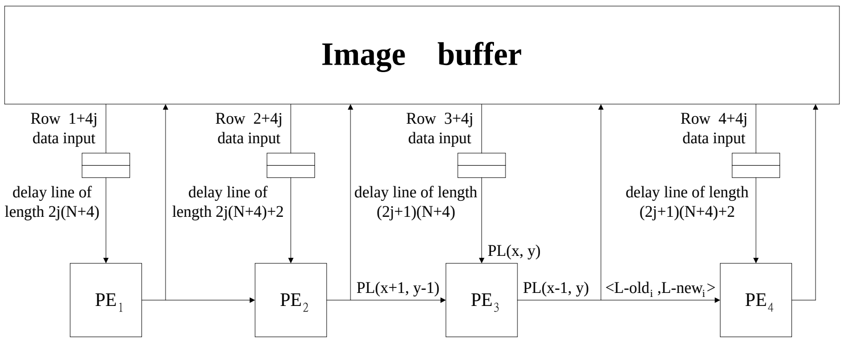


Fig. 7. Linear array architecture.

3.1 Processing Element

Based on the parallel algorithm, the operations performed in PE_i include:

(1) comparing the input values and labels of four adjacent pixels in Passes 1 to 4;

(2) outputting the input value and label of $P(x, y)$ to PE_{i+1} and the image buffer (only the label).

After considering all the necessary operations, the structure of PE_i shown in Fig. 8 is proposed. Two input ports are required to receive the information from the image buffer and PE_{i-1}. The pixel values and/or labels from the image buffer and label equivalence $<L\text{-old}_{i-1}, L\text{-new}_{i-1}>$ from PE_{i-1} are received from Ports 1 and 2, respectively. Meanwhile, in Passes 2, 3, and 4, the input data of PE_i needs $P(x, y)$ and $L(x, y)$; thus, the input data

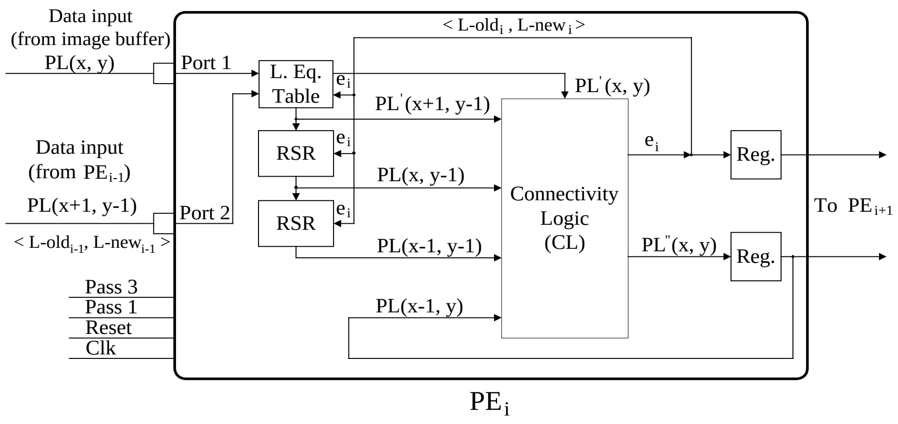


Fig. 8. Organization of a PE.

of Port 1 is connected to the image buffer. For calculating label L(x, y), the data path from PE_{i-1} is also required so that the input values and labels of three adjacent pixels in row y – l, and the label equivalence that was created by PE_{i-1}, are obtained sequentially through Port 2. These input values and labels, and the label equivalence are input with two delay units, and the output label of P(x, y) from PE_i is returned to the image buffer. Consequently, the input data required from row y – 1 is ready before calculation of the label of P(x, y) in the connectivity logic (CL) begins. Meanwhile, a feedback loop is established to link the output of CL through a delay unit to the input of CL. This path provides the label of P(x – 1, y) to CL. Finally, the output is sent to PE_{i+1} and the label map memory.

3.2 CL

To determine the minimum label of the four adjacent input values that are equal to the value of P(x, y) itself, a 4-input NOR gate, an incrementer, three multiplexors, four XNOR gates, and two 4-way comparators are used in CL, as shown in Fig. 9. If the adjacent input values are not the same as the value of P(x, y), then the incrementer will assign a new label to L(x, y), namely L(x, y) = y * N + x. The output of the comparator is the label of P(x, y), which is also passed to PE_{i+1}.

3.3 Relabel Shift Register

The relabel shift register (RSR) is shown in Fig. 10, and contains an MUX, a w-bit XNOR, and four registers. If the pixel label L(x, y) is the same as the old label L-old$_i$, then the label L(x, y) is relabeled using the new label L-new$_i$; otherwise, it remains unaltered. The value of P(x, y) is simply propagated through the register.

3.4 Label Equivalence Table

The label equivalence table provides the minimum label of P(x, y) used for connectivity logic. When two regions were merged, a label equivalence containing the old label

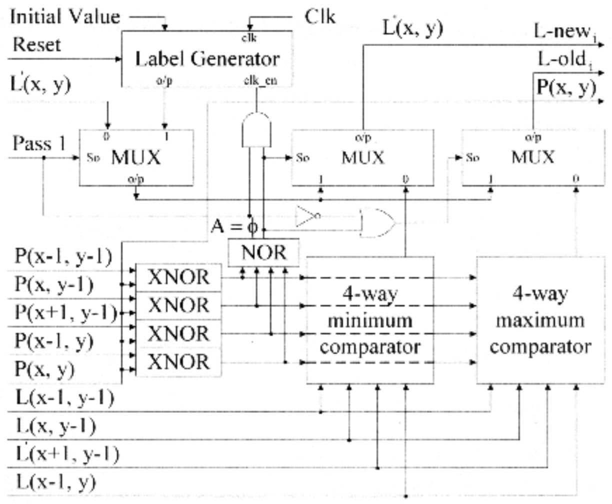


Fig. 9. Structure of the connectivity logic (LC).

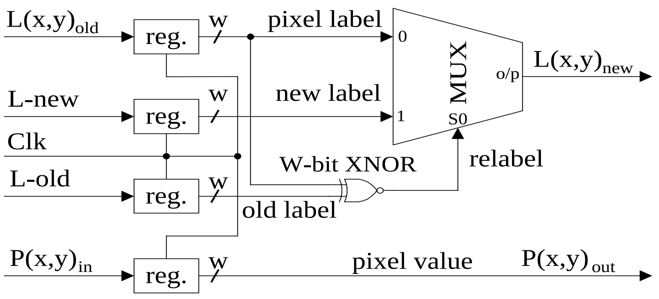


Fig. 10. Structure of the relabel shift register (RSR).

(L-old) and new label (L-new) is recorded, and placed in the label equivalence table. The label equivalence table uses CAM (Content Addressable Memory) distributed-memory providing a PSMU (Parallel Search and Multiple Update) operation. This operation can be illustrated via a simple parallel algorithm, for n CAMs, as follows:

```
PSMU_ Operation _Algorithm
for all M[addr], (1 ≦ addr ≦ n) do_in_parallel
    if (M[addr] = L-old ) then M[addr] := L-new
end for
```

Using the space-time diagram, the time complexity of the algorithm can be easily analyzed. If the input data rate is one pixel per cycle, then the label map is obtained for the forward stage or the backward stage in $\frac{N^2}{2} + 3N - 2$ cycles, where

$$\begin{aligned}\tfrac{N^2}{2} + 3N - 2 &= \text{(double length of the first row)} \times \text{(execution cycle per pixel)} + \\ &\quad \text{(skewed delay per row)} \times \text{(row number of image} - 1) + \\ &\quad \text{(half of the row number of image} - 1) \times \text{(column number of image)} \\ &= 2N \times 1 + 2 \times (N - 1) + (\tfrac{N}{2} - 1) \times N.\end{aligned}$$

Hence, the total execution time for the four passes is $N^2 + 6N - 4$ cycles.

3.5 FPGA Design and Post-Simulation

FPGA was designed using the DesignBook graphics capture package created by Escalade, while synthesis was performed using the Leonardo Spectrum VHDL compiler from Exemplar, and post-simulation was conducted using Modelsim of Modeltech. The proposed hardware architecture, which was first implemented on an A54SX16 FPGA, includes a PE which can perform 8×8 image, and can be scaled to any number of PE's without suffering throughput loss. The PE implementation was verified using a clock frequency of 20 MHz, and extensive timing was performed on the FPGA using Modelsim. The critical path was in FPGA within 46 ns. The chip, thus, operates at a maximum clock frequency of 21.74 MHz using 0.35 micron technology. Transferring the hardware architecture to a more competitive lower micron technology will enhance the operating speed. The hardware algorithm proposed here would require 4 PE's, and for a very conservative clock cycle of 50 ns, approximately 0.8574 ms would be needed to label a 128×128 image.

4. CONCLUSIONS

A parallel algorithm and its hardware architecture for connected component labeling have been described in this paper. Applying the novel architecture, the label map of an $N \times N$ binary image can be obtained within $N^2 + 6N - 4$ cycles using the architecture with 4 PEs. The proposed hardware can process a 128×128 image with a cycle time of 50ns in about 0.8574 ms and uses only 4 processors, while the method of Ranganathan *et al.* [14] requires a clock cycle of 25ns to process the image in approximately 0.85 ms with 128 processors, and the MPP method [22] requires a cycle time of 100ns to process the image in 94.6 ms and uses 16384 processors. Other advantages of the novel architecture include modularity, expandability, regular data flow, and simple hardware. These characteristics are highly desirable for VLSI implementations.

REFERENCES

1. J. T. Schwartz, M. Sharir, and A. Siegel, "An efficient algorithm for finding connected components in a binary image," Technical Report 156, Courant Institute, New York University, 1985.
2. C. Ronse and P. A. Devijver, "Connected components in binary images: The detection problem," *Research Studies Press Ltd.*, John Wiley and Sons Inc., 1984.
3. L. D. Stefano and A. Bulgarelli, "A simple and efficient connected components labeling algorithm," in *Proceedings of International Conference on Image Analysis*

and Processing, 1999, pp. 322-327.

4. K. Haris, S. Efstratiadis, N. Maglaveras, J. Gourassas, and G. Louridas, "Artery skeleton extraction using topographic and connected component labeling," in *Proceedings of International Conference on Image Processing*, 2001, pp. 339-342.
5. A. Rosenfeld and J. L. Pfaltz, "Sequential operation in digital picture processing," *Journal of Association for Computing Machinery*, Vol. 13, 1966, pp. 471-494.
6. R. M. Haralick, "Some negihborhood operations," in *Real Time/Parallel Computing Image Analysis,* M. Onoe, K. Preston, and A. Rosenfled, eds., New York: Plenum Press, 1981, pp. 11-35.
7. R. Lumia, L. Shaprio, and O. Zuniga, "A new connected components algorithm for virtual memory computers," *Computer Vision, Graphics and Image Processing*, Vol. 22, 1983, pp. 287-300.
8. K. Suzuki, I. Horiba, and N. Sugie, "Fast connected-components labeling based on sequential local operations in the course forward raster scan followed by backward raster scan," in *Proceedings of 15th International Conference on Pattern Recognition*, Vol. 2, 2000, pp. 434-437.
9. V. Khanna, P. Gupta, and C. J. Hwang, "Maintenance of connected components in quadtree-based image representation," in *Proceedings of International Conference on Information Technology: Coding and Computing*, 2001, pp. 647-651.
10. Y. Shiloach and U. Vishkin, "An O(logN) parallel connectivity algorithm," *Journal of Algorithms*, Vol. 3, 1982, pp. 57-67.
11. M. Manohar and H. K. Ramapriyan, "Connected component labeling of binary images on a mesh connected massively parallel processor," *Computer Vision, Graphic and Image Processing*, Vol. 45, 1989, pp. 133-149.
12. P. Kr. Biswas, J. Mukherjee, and B. N. Chatterji, "Component labeling in pyramid architecture," *Pattern Recognition*, Vol. 26, 1993, pp. 1099-1115.
13. Y. Cheng, J. R. Jensen, T. Huntsberger, and B. Huntsberger, "Hypercube algorithm for image component labeling," in *Proceedings of the Scalable High-Performance Computing Conference 1994, IEEE*, pp. 259-262.
14. N. Ranganathan, R. Mehrotra, and S. Subramanian, "A high speed systolic architecture for labeling connected components in an image," *IEEE Transactions on System, Man and Cybernetics*, Vol. 25, 1995, pp. 415-423.
15. C. J. Nicol, "A systolic approach for real time connected component labeling," *Computer Vision and Image Understanding*, Vol. 61, 1995, pp. 17-31.
16. D. Nassimi and S. Sahni, "Finding connected components and connected ones on a mesh-connected parallel computer," *SIAM Journal of Computing*, Vol. 9, 1980, pp. 744-757.
17. R. Miller and Q. Stout, "Varying diameter and problem size in mesh-connected computers (preliminary version)," in *Proceedings of 1985 International Conference on Parallel Processing*, pp. 697-699.
18. V. K. P. Kumar and M. M. Eshaghian, "Parallel geometric algorithms for digitized pictures on mesh of trees," in *Proceedings of 1986 International Conference on Parallel Processing* (*ICPP*), pp. 270-273.
19. W. Lim, "Fast algorithms for labeling connected components in 2-D arrays," Technical Report, No. 86.22, Thinking Machine Corp., Cambridge, Mass., 1986.
20. R. Miller and Q. Stout, "Data movement techniques for the pyramid computer,"

SIAM Journal of Computing, Vol. 16, 1987, pp. 38-60.
21. R. Cypher, J. L. C. Sanz, and L. Snyder, "Hypercube and shuffle-exchange algorithms for image component labeling," *Journal of Algorithms*, Vol. 10, 1989, pp. 140-150.
22. R. Cypher, J. L. C. Sanz, and L. Snyder, "An EREW PRAM algorithm for image component labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, 1989, pp. 258-262.
23. A. Agrawal, L. Nekludova, and W. Lim, "A parallel O(logN) algorithm for finding connected components in planar images," in *Proceedings of International Conference on Parallel Processing*, 1987, pp. 783-786.

Kuang-Bor Wang (王光伯) was born in Tainan, Taiwan, Republic of China, in 1957. He received the B.S. degree in Electronic Engineering from the Feng Chia University, Taichung, Taiwan in 1984 and the M.S. degree in Electrical Engineering from Chung Hua University, Hsinchu, Taiwan in 1995. He is a Ph.D. candidate at Chung Cheng Institute of Technology now. His research interests are in the digital image processing, pattern recognition, parallel computing and vehicle navigation.

Tsorng-Lin Chia (賈叢林) received the B.S. degree in electrical engineering from Chung Cheng Institute of Technology, Taoyuan, Taiwan, in 1982, and the M.S. and Ph.D. degrees in computer science and information engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1986 and 1993, respectively. He is currently a professor in the Department of Information Management at Ming Chuan University. His research interests include image processing, pattern recognition, computer vision, and parallel algorithm and architecture.

Zen Chen (陳稔) is currently a professor of the Department of Computer Science and Information Engineering, National Chiao Tung Univerisity, Hsinchu, Taiwan. Before he joined National Chiao Tung University, he worked for Burroughs Corporation, Detroit, Michigan from 1973 to 1974, where he was engaged in the development of document recognition systems. From 1981 to 1982 he was a visiting scientist in the Lawrence Berkeley Laboratory, University of California, Berkeley, Califor-nia. From 1989 to 1990 he was a visiting professor at the

Center for Automation Research, University of Maryland, College Park, Maryland. His current research interests include computer vision, visual surveillance system, virtual reality, character recognition and document analysis, and parallel algorithms and architectures. Dr Chen is a member of Sigma Xi and Phi Kappa Phi. He was the founding president of the Chinese Society of Image Processing and Pattern Recognition in Taiwan.

Der-Chyuan Lou (婁德權) was born in Chiayi, Taiwan, Republic of China, on Mar. 18th, 1961. He received the B.S. degree from the Chung Cheng Institute of Technology, Taiwan, R.O.C., in 1987, and the M.S. degree from the National Sun Yat-Sen University, Taiwan, R.O.C, in 1991, both in electrical engineering. He received the Ph.D. degree in 1997 from the Department of Computer Science and Information Engineering at the National Chung Cheng University, Taiwan, R.O.C. Since 1987, he has been with the Department of Electrical Engineering at the Chung Cheng Institute of Technology, National Defense University, R. O. C., where he is currently an associate professor. His research interests include computer arithmetic, parallel and distributed processing, steganography, cryptography. Dr. Lou is a member of the IEICE society and an honorary member of the Phi Tau Phi Scholastic Honor Society. He is the owner of the eleventh Acer Dragon Ph.D. Dissertation Award. He has been selected and included in the fifteenth edition of Who's Who in the World that has been published in 1998.