Fundamenta Informaticae 76 (2007) 75–89
IOS Press

A Co-processor for Computing the Euler Number of a Binary Image using Divide-and-Conquer Strategy [*]

Sabyasachi Dey
Texas Instruments Pvt. Ltd.
Bangalore - 560 017, India
dsabyasachi@ti.com

Bhargab B. Bhattacharya[†], Malay K. Kundu,
Indian Statistical Institute, 203, B.T. Road,
Calcutta - 700 108, India
{bhargab,malay}@isical.ac.in

Arijit Bishnu,
Indian Institute of Technology
Kharagpur - 721 302, India
arijit.bishnu@iitkgp.ac.in

Tinku Acharya
Avisere, Chandler
AZ 85226, USA
tinku.acharya@ieee.org

Abstract. Euler number is a fundamental topological feature of an image. The efficiency of computation of topological features of an image is critical for many digital imaging applications such as image matching, database retrieval, and computer vision that require real time response. In this paper, a novel algorithm for computing the Euler number of a binary image based on divide-and-conquer paradigm, is proposed, which outperforms significantly the conventional techniques used in image processing tools. The algorithm can be easily parallelized for computing the Euler number of an $N \times N$ image in $O(N)$ time, with $O(N)$ processors. Using a simple architecture, the proposed method can be implemented as a special purpose VLSI chip to be used as a co-processor.

Keywords: Binary image, digital imaging, Euler number, VLSI.

1. Introduction

Topological properties serve the purpose of representing geometric shape of an image. They remain invariant under any arbitrary *rubber-sheet* transformation [1, 2, 7] and hence, are very useful in image

[*]This work is funded by a grant from Intel Corp., USA (PO # CAC042717000)
[†]Address for correspondence:Indian Statistical Institute, 203, B. T. Road, Calcutta - 700 108, INDIA

characterization for matching shapes, recognizing objects, image database retrieval, and in numerous other image processing and computer vision applications. An important topological feature of an image is the Euler number (or genus), which is the difference between the number of connected components and the number of holes [1, 7].

With the emergence of fast Internet facilities, a growing demand for distributed and high-performance image retrieval systems is being felt. Euler number can play an important role in such applications. It can also be used in medical diagnosis from cell images, e.g., detection of malaria infected cells, as the Euler number of an infected cell is often different from that of a good one. Critical image processing applications involve large amount of data and at the same time demands for real-time response. Fast computation of the Euler number of an image is therefore, an indispensable task in various missions.

Dyer proposed an algorithm to compute the Euler number of an image represented by a quadtree [3]. Samet and Tamminen improved the algorithm further by using a new staircase type of data structure to represent the blocks that have already been processed [4]. However, it is not suitable for VLSI implementation, as the sizes of the leaf nodes are unequal, and the number of leaf nodes varies widely for different image samples. Gray [5] has described a method based on local pattern counting which is used in commercial image processing tools like MATLAB [8]. The time complexity of this method is $O(N^2)$ for an $N \times N$ image, which is linear in the number of pixels in the image. For image processing tasks, where the data is huge, the constant term that is so often hidden in the *big-Oh* notation, becomes important. For large images, even a linear time sequential algorithm may be inadequate to meet critical time requirements. A faster method is thus needed to handle large binary images. Recent advances in parallel processing and VLSI technology can be exploited to develop high performance algorithm and architecture that achieves real-time goals. A pipeline architecture for computing the Euler number on-chip has been recently reported [12]. The concept of Euler number as a discriminatory feature, has been extended further to characterize a gray-tone image [13].

In this paper, a new sequential algorithm that computes the Euler number of a binary image is proposed, based on divide-and-conquer approach. Although the worst case complexity of the algorithm is $O(N^2)$, its average-case behavior outperforms the earlier algorithms significantly. Next, a parallel version of the algorithm is described that takes $O(N)$ time for an $N \times N$ image, using $O(N)$ simple processors, which can be readily mapped to a simple VLSI architecture. The proposed algorithm can easily be implemented with a special purpose VLSI chip that can serve as a co-processor of the host computer, to expedite computation.

2. Preliminaries

The *Euler number* (or genus), of a binary image is the difference between the number of connected components (objects), and the number of holes [1, 5, 7]. Let the binary image be represented by a 0-1 pixel matrix of size $(N \times M)$, in which an object (background) pixel is denoted as 1 (0). In a binary image, a *connected component* is a set of object pixels such that any object pixel in the set is in the 8 (or 4) neighborhood of at least one object pixel of the same set. A *hole* is a set of background pixels such that any background pixel in the set is in the 4- (or 8-) neighborhood of at least one background pixel of the same set and this entire set of background pixels is enclosed by a connected component. The sets referred to in the definition are sets contained in the image. As an example, the figure of '9' shown in Figure 1 has a single connected component and only a single hole. So, its Euler number is 0.

The most efficient and simplest algorithm, reported so far, for computing Euler number of an image works by looking into local patterns [5, 7]. Consider the following set of 2×2-pixel patterns called bit quad:

$$\mathbf{Q_1} = \left\{ \begin{matrix} 0 & 0 \\ 1 & 0 \end{matrix}, \begin{matrix} 0 & 0 \\ 0 & 1 \end{matrix}, \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix}, \begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix} \right\}$$

$$\mathbf{Q_2} = \left\{ \begin{matrix} 0 & 1 \\ 1 & 1 \end{matrix}, \begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix}, \begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix}, \begin{matrix} 1 & 1 \\ 0 & 1 \end{matrix} \right\}$$

$$\mathbf{Q_3} = \left\{ \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix}, \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \right\}$$

Let C_1, C_2 and C_3 be the number of patterns Q_1, Q_2, Q_3 respectively in the image S. It has been shown in [5] that under the definition of four-connectivity the Euler number can be computed as

$$- \quad E(S) = \frac{1}{4}\left(C_1 - C_2 + 2 \cdot C_3\right) \tag{1}$$

and for eight-connectivity

$$- \quad E(S) = \frac{1}{4}\left(C_1 - C_2 - 2 \cdot C_3\right) \tag{2}$$

This method is used in MATLAB image processing tool box [8]. Henceforth, we shall refer this method as Gray's algorithm.

In the proposed divide-and-conquer approach, the input image is partitioned into a number of disjoint atomic images. By atomic is meant a small part of the input image which need not be decomposed further. The Euler number of each atomic image, which is substantially smaller in size compared to the input image, can be computed by the method described above. Once the Euler numbers of all atomic images are evaluated, the Euler number of the original image can be computed by using a simple arithmetic rule.

3. Proposed Algorithm

3.1. Divide-and-conquer approach

The given image is partitioned recursively by arbitrary cut-lines.

Definition 1. A cut line is a sequence of pixels from one boundary of the image to its opposite boundary, where each pixel has exactly two neighboring pixels along the cut-line (except the start and end pixels which have only one neighbor each).

Without any loss of generality, we restrict the choice of cut-lines to only horizontal and vertical directions. Since a binary image is normally represented as a 2-D $(0-1)$ pixel matrix, any row or column of the matrix can be used to designate a cut-line.

Definition 2. A *run* on a row (or column) is defined to be a maximal sequence of consecutive 1-pixels.

As cut-line is designated by any row or column, *run* on a cut-line is defined accordingly. In the following definition, we define *adjacent runs*.

Definition 3. Consider two runs R$_1$ $(p_1, p_2, \ldots, p_n)$ and R$_2$ $(q_1, q_2, \ldots, q_m)$. They are adjacent if and only if there exists atleast one pixel $p_i \in R_1$ and one pixel $q_j \in R_2$ such that p$_i$ is in neighborhood of q$_j$ and vice versa.

Definition 4. The *union* $(\cup)$ of two images S_a and S_b is defined as a simple juxtaposition of S_a and S_b either vertically or horizontally, without any overlap. The *intersection* $(\cap)$ of S_a and S_b is the image formed by the last row (or column) of S_a, and the first row (or column) of S_b, if the images S_a and S_b are joined horizontally (or vertically).

Let S be the given binary image, and L denotes a cut-line that partitions S into two sub-images S_1 and S_2. Let ρ be a run on line L and let there be r such runs. Denote by $k_1(\rho)$ the number of runs on the boundary line of S_1 that are adjacent to (neighbor of) ρ, and by $k_2(\rho)$, the number of such runs in S_2.

Lemma 1. [5, 6] Euler number satisfies the *local additive property*. Given three images S_1, S_2 and L with Euler numbers $E(S_1)$, $E(S_2)$ and $E(L)$ respectively, the Euler number of the image $S = S_1 \cup S_2 \cup L$ is given by: $E(S) = E(S_1 \cup S_2 \cup L) = E(S_1) + E(S_2) + E(L) - E(S_1 \cap S_2) - E(S_1 \cap L) - E(S_2 \cap L) + E(S_1 \cap S_2 \cap L)$.

For details of the Lemma 1, see [5, 6].

Observation 1. The cut line L is a column (or row) lying in between S_1 and S_2, and separating S_1 and S_2 such that $S_1 \cap S_2 = \phi$. So, $E(S_1 \cap S_2) = 0$ and $E(S_1 \cap S_2 \cap L) = 0$. Therefore, $E(S) = E(S_1 \cup S_2 \cup L) = E(S_1) + E(S_2) + E(L) - E(S_1 \cap L) - E(S_2 \cap L)$.

Observation 2. There can be no holes in a one-row (one-column) or two-row (two-column) wide image.

Lemma 2. For a cut line L, which is a single row (or column), the Euler number, $E(L) = r$, where r is the number of runs in L.

Proof:
The proof follows from Observation 2 and the fact that the number of connected components in L is r. □

Note that, the intersection image is always two pixel row (or column) wide.

Lemma 3. $E(S_i \cap L)$ = the number of neighboring runs between S_i and L.

Proof:
The proof follows from Definitions 3 and 4 and Observation 2 and the fact that the number of connected components is equal to the number of neighboring runs for a two row wide image. □

Theorem 1. If a binary image S is partitioned into two sub-images S_1 and S_2 along a cut-line L such that,

1. $S = S_1 \cup S_2 \cup L,\ and$

2. $S_1 \cap S_2 \cap L = \phi$

then, the Euler number $E(S)$ of the image S is given by

$$E(S) = E(S_1) + E(S_2) + Cont(L) \tag{3}$$

where,
Cont(L) = $\sum_{\rho=1}^{r}\{1 - k_1(\rho) - k_2(\rho)\}$. □

Proof:
From Observation 1, we have $E(S) = E(S_1) + E(S_2) + E(L) - E(S_1 \cap L) - E(S_2 \cap L)$. From Lemma 2, we have $E(L) = r$, and from Lemma 3, we have $E(S_1 \cap L) + E(S_2 \cap L) = \sum_{\rho=1}^{r}\{k_1(\rho) + k_2(\rho)\}$. Thus, $Cont(L) = E(L) - E(S_1 \cap L) - E(S_2 \cap L) = \sum_{\rho=1}^{r}\{1 - k_1(\rho) - k_2(\rho)\}$. □

Example 1. Image S (see figure 1) is partitioned into S_1 and S_2 by cut L_1. The component S_1 is again partitioned into S_{11} and S_{12} by cut L_2. Similarly, S_2 is partitioned into S_{21} and S_{22} by cut L_3.
$E(S_1) = E(S_{11}) + E(S_{12}) + Cont(L_2)$ where $Cont(L_2) = \sum_{r \in L_2}\{1 - k_1(\rho) - k_2(\rho)\} = 0$, and $E(S_{11}) = E(S_{12}) = 1$. Hence, $E(S_1) = 1 + 1 + 0 = 2$.
$E(S_2) = E(S_{21}) + E(S_{22}) + Cont(L_3)$ where $Cont(L_3) = \sum_{r \in L_3}\{1 - k_1(\rho) - k_2(\rho)\} = 1 - 1 - 1 = -1$, and $E(S_{21}) = E(S_{22}) = 1$. Hence, $E(S_2) = 1 + 1 - 1 = 1$.
$E(S) = E(S_1) + E(S_2) + Cont(L_1)$ where $Cont(L_1) = \sum_{\rho \in L_1}\{1 - k_1(\rho) - k_2(\rho)\} = 3 - 3 - 3 = -3$, and $E(S_1) = 2$, and $E(S_2)$ = 1. Hence, $E(S) = 2 + 1 - 3 = 0$.

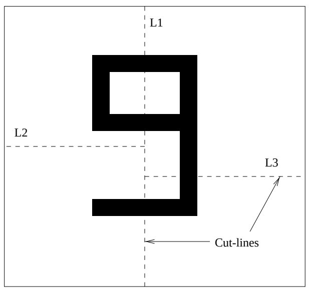


Figure 1. Divide-and-conquer illustrated

In the next subsection, we describe a sequential algorithm based on recursive application of the above partitioning scheme.

3.2. Sequential algorithm

The recursive procedure *ComputeEuler* computes the Euler number of an image. If both the dimensions of the partitioned image become smaller than some predefined value MIN, then we compute its Euler number by procedure *Euler*. Procedure *Contribution* computes contribution of the cut-line. We shall show how to determine the value MIN after analyzing the parallel algorithm in section 4.3.

procedure ComputeEuler(S, hlen, vlen)

```
begin
   if ( hlen < MIN and vlen < MIN ) then
   begin
       e = Euler(S);
   end
   else
   begin
    /*
       S is partitioned by cut-line
       L into S1 and S2
     */
     if ( hlen > vlen ) then
       begin
          x = ComputeEuler(S1,hlen/2,vlen);
          y = ComputeEuler(S2,hlen/2,vlen);
          z = Contrib(L);
          e = x + y + z;
       end
       else
       begin
          x = ComputeEuler(S1,hlen,vlen/2);
          y = ComputeEuler(S2,hlen,vlen/2);
          z = Contrib(L);
          e = x + y + z;
       end
   end
   return e;
end.
```

Procedure *Euler* computes the Euler number of an atomic image by the method described in Gray [5]. The basis of procedure *Contrib* is described below.

3.3. Contribution computation

In this section, we describe a method for computing contribution of the line L that partitions the image S into two sub-images S_1 and S_2 as mentioned in equation (3). Recall,

$$Cont(L) = \sum_{\rho \in L} \{1 - k_1(\rho) - k_2(\rho)\}$$

We thus have to count the number of runs along L, as well as number of runs along the boundary lines of S_1 and S_2 that are adjacent to runs on L. Consider the following 2×3 bit pattern.

$$\begin{pmatrix} x & u \\ y & v \\ z & w \end{pmatrix}$$

where, $x, u \in S_1$, $z, w \in S_2$, and $y, v \in L$. Without any loss of generality, we can assume that L is the ith row of the image matrix S. Then S(i-1,j-1) = x, S(i-1,j) = u, S(i+1,j-1) = z, S(i+1,j) = w, S(i,j-1) = y, and S(i,j) = v. We traverse the band of three consecutive rows (i-1, i, and i+1) column by column remembering the elements of the last column. Hence, when we look into column j, the elements of (j-1)th and jth columns are known. Let the number of runs on L be α, $\sum_\rho k_1(\rho) = \beta$, and $\sum_\rho k_2(\rho) = \gamma$. Using Lemma 2 and 3 we can write,

$$\sum_\rho \{1 - k_1(\rho) - k_2(\rho)\} = \alpha - \beta - \gamma$$

We define three boolean functions Γ , Ψ , and Φ. We increase $\alpha, \beta,$ and γ as following: (1) if Γ is true then increment β by one, (2) if Ψ is true then increment γ by one, and (3) if Φ is true then increment α by one. Under the definition of four-connectivity, the above boolean functions are defined as: $\Phi = \bar{y} \cdot v$, $\Gamma = u \cdot v \cdot (\bar{x} + \bar{y})$, $\Psi = w \cdot v \cdot (\bar{z} + \bar{y})$; and for eight-connectivity they are defined as, $\Phi = \overline{y} \cdot v$, $\Gamma = \bar{x} \cdot u \cdot v + \bar{x} \cdot u \cdot y + x \cdot \bar{y} \cdot v$, $\Psi = \bar{z} \cdot w \cdot v + \bar{z} \cdot w \cdot y + z \cdot \bar{y} \cdot v$. Thus, using the above equations, $Cont(L)$ can be evaluated easily from the pixel matrix.

3.4. Complexity and results

The time complexity $T(n)$ (where n is the number of pixels) of the divide and conquer algoritm follows the recurrence

$$T(n) = \begin{cases} 1 & \text{if n=1;} \\ 2T(\frac{n}{2}) + O(\sqrt{n}) & \text{if } n \geq 2. \end{cases}$$

Solving the recurrence, we get $T(n) = O(n)$. In our case, as $n = N^2$, we get the time complexity as $O(N^2)$, which is linear in the number of pixels. Although the asymptotic time complexity of procedure *ComputeEuler* is $O(N^2)$ which is same as that of the method described in [5, 7] for a $N \times N$ image, its average case performance is much better, as we have a smaller constant factor. In the Gray's algorithm [5, 7], each pixel of the image is accessed twice except the boundary pixels, which are accessed only once. On the other hand, in the proposed divide-and-conquer method, each pixel along a cut-line is accessed only once, and pixels in the sub-images which are adjacent to 0's in the cut-line, need not be checked. Thus, the number of pixel accesses is significantly reduced on the average and consequently, we save time as we need fewer memory references. Experimental results of our implementation demonstrate the savings in computation time, and are shown below. The algorithm has been implemented in C and the code runs on both Solaris 2.6 and Linux platforms. We have tested the program with several hundred images out of which 10 examples are tabulated in Table 1. We compared our algorithm with that of Gray [5, 7], and the results are shown in Table 1. The columns labeled "Gray" and "Proposed method" correspond to the number of pixel accesses, and the resulting improvement is given in percentage. For the Gray's method, a pixel is accessed twice but the boundary pixels are accessed only once, for determining convexity. Thus, it becomes slightly less than $N \times M \times 2$ for an $N \times M$ image [10].

Table 1. Comparison with the Gray's method

Img	Gray	Prop. meth.	Improvement
tm1	33024	28008	15.19
tm2	131584	104082	20.90
tm3	131584	109086	17.10
tm4	131584	111250	15.45
tm5	33024	27426	16.95
tm6	33024	28598	13.40
ieee1	33024	29086	14.92
ieee2	33024	25705	22.15
vlsi	525312	448774	14.57
text	131584	102880	22.77

3.5. Results on distribution of Euler number

The values of Euler number of the images in the database vary widely from -3796 to 2425. There are 85 images having Euler numbers in the range -3796 to -10; 829 images in the range, -9 to 10; and 125 images in the range, 11 to 2425. See Figure 2 for the range of Euler numbers and the frequency of the images. There are 76 images each with a distinct Euler number; there are 18 cases, where only two images have the same Euler number. The frequency of the images having different Euler numbers is shown in Table 2. This observation justifies that Euler number can be used as a potential tool for image discrimination, search, and retrieval. Some of the the logo images used for our experiments along with their Euler number are shown in Figure 6.

4. Parallel Implementation

4.1. The Algorithm

In the proposed approach, each cut increases the number of sub-images by one. Let there be K atomic images and K-1 cut-lines. The parallel architecture that lends itself most naturally to this divide-and-conquer algorithm is a (binary) tree. We use two types of processing elements (PE's). Type-1 PE computes procedure *Contrib* and evaluates equation (3). Type-2 PE computes the Euler number of an atomic image, i.e., computes procedure *Euler*. The PE's are organized in a binary tree, in which the type-2 PE's are leaf nodes and the type-1 PE's are non-leaf nodes. There are K-1 type-1 PE's denoted by $PE_{1,1}, PE_{1,2}, \ldots, PE_{1,K-1}$ and K type-2 PE's denoted by $PE_{2,K}, PE_{2,K+1}, \ldots, PE_{2,2K-1}$. Each type-1 PE, $PE_{1,i}$ where $i \neq 1$ is connected to $PE_{1,i/2}$ if i is even and to $PE_{1,(i-1)/2}$ if i is odd. Each type-2 PE, $PE_{2,j}$ is connected to $PE_{2,j/2}$ if j even and to $PE_{2,(j-1)/2}$ if j odd. Let the cut-lines be $L_1, L_2, \ldots, L_{K-1}$. The contribution of L_i is computed by $PE_{1,i}$. Let the atomic images be $S_1, S_2, \ldots, S_K$, where the Euler number of S_j is computed by $PE_{2,j+k-1}$. The atomic images S_j and

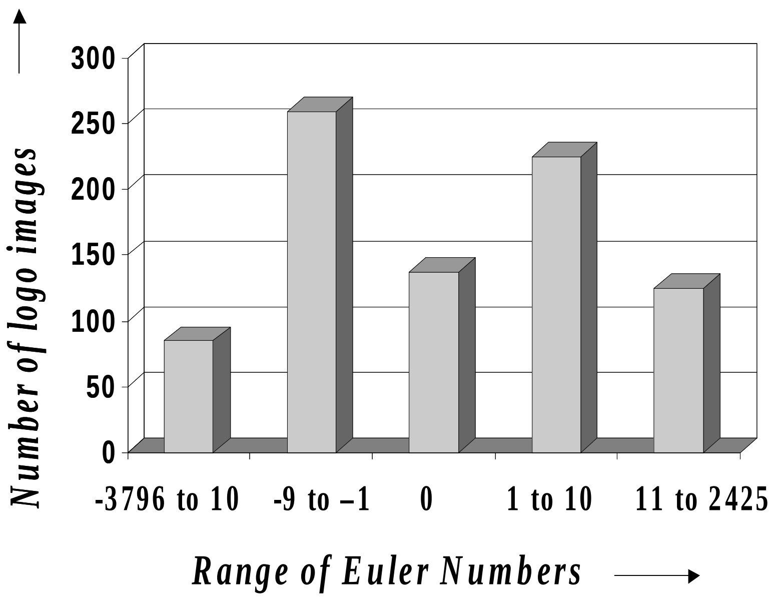


Figure 2. Bar chart showing distribution of ranges of Euler number

S_{j+1}, where j is 1, 3, . . ., K-1, have been generated by the cut-line $L_{(j+k-1)/2}$. If L_i partitions an image S into S_1 and S_2 then all computations for S_1 are done in left sub-tree of node $PE_{1,i}$, and those for S_2 are done in right sub-tree of node $PE_{1,i}$. Let h be the height of the tree.

SIMD SM CREW Algorithm

```
Step 1a : for i = 1 to K-1 do in parallel
    $PE_{1,i}$ computes contribution of line $L_i$;
    endfor.

Step 1b : for i = 1 to K do in parallel
    $PE_{2,i}$ computes $E(S_i)$;
    $PE_{2,i}$ sends the result to its parent;
    endfor.

Step 2 : for l = h-1 downto 2 do
    for i = $2^{l-1}$ to $2^l - 1$ do in parallel          $PE_{1,i}$ evaluates eq.(3);
    $PE_{1,i}$ sends the result to its parent;
    endfor.
    endfor.
    $PE_{1,1}$ evaluates eq.(3) and outputs result.
```

4.2. VLSI Architecture

The organization of the PE's as a binary tree has been described before (see Figure 3).

In this section, we sketch the internal organization of each PE.

Table 2. Distribution of Euler Number for 1039 logo images

Euler Number	Number of images	Euler Number	Number of images	Euler Number	Number of images
-3796	1	-18	3	27	2
-2624	1	-17	3	29	1
-1698	1	-16	1	30	1
-1059	1	-15	2	33	1
-653	1	-14	6	34	2
-650	1	-13	3	35	1
-404	1	-12	6	36	1
-378	1	-11	1	37	4
-302	1	-10	6	38	2
-256	1	-9	12	39	1
-160	1	-8	15	40	1
-154	1	-7	17	41	4
-153	1	-6	24	43	1
-147	1	-5	31	44	1
-142	1	-4	31	46	2
-112	1	-3	49	48	1
-110	1	-2	69	49	2
-95	1	-1	80	50	2
-92	1	0	137	53	1
-87	1	1	121	55	1
-56	1	2	60	56	1
-54	2	3	44	57	2
-52	2	4	46	58	1
-51	1	5	25	60	1
-49	1	6	14	62	1
-48	1	7	16	65	1
-45	1	8	14	66	1
-43	1	9	11	73	2
-42	1	10	13	75	1
-40	1	11	9	80	1
-36	1	12	6	82	1
-35	1	13	7	112	1
-33	1	14	4	114	1
-32	3	15	7	163	1
-31	1	16	3	168	1
-29	1	17	7	204	1
-28	1	18	8	233	1
-26	1	19	1	254	1
-25	2	20	2	276	1
-24	1	21	3	293	1
-23	2	22	3	534	1
-22	2	23	1	861	1
-21	2	24	2	1567	1
-20	2	25	1	2425	1
-19	1	26	3		

A. Type-1 processing element

The circuit for computing the contribution of a cut-line is very simple and is shown in Figure 4. It consists of three D flip-flop's FF_1, FF_2, FF_3 with inputs u, v, w and outputs x, y, z respectively. In each clock cycle, a new column of the 3-pixel wide band is fed to the inputs of the FF's and the values for the earlier column are already at their outputs. According to the equations in Section 3.3 a combinatorial circuit consisting of basic gates evaluates Γ, Ψ, and Φ. In Figure 4, the combinational blocks for computing Γ, Ψ, and Φ are shown as a triangular object. We use three counters C_1, C_2, and C_3 driven by the values of Γ, Φ, Ψ respectively. The value of contribution is computed as,

$$Cont(L) = C_2 - C_1 - C_3.$$

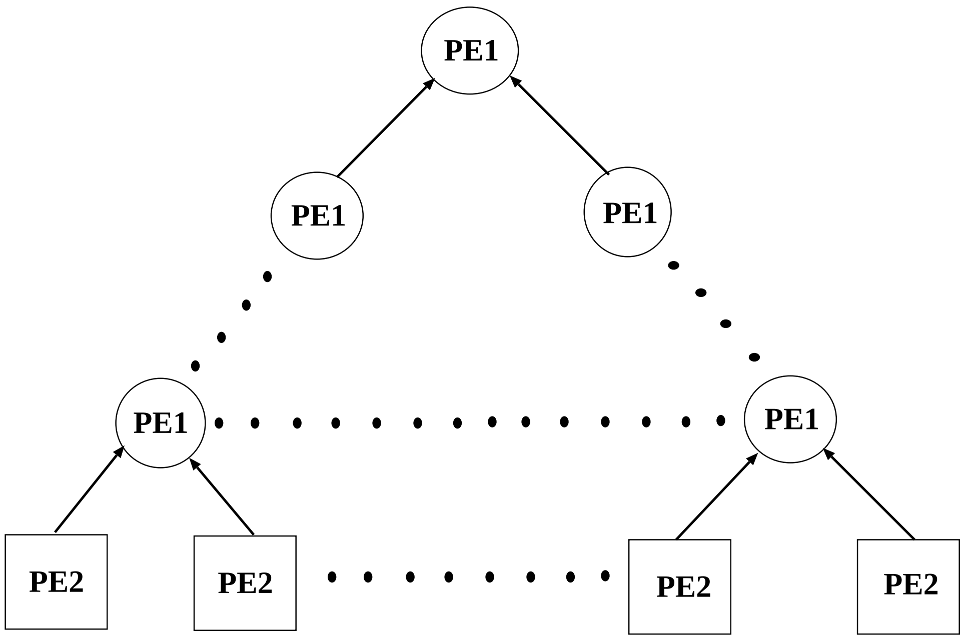


Figure 3. Processor Organization in a Tree

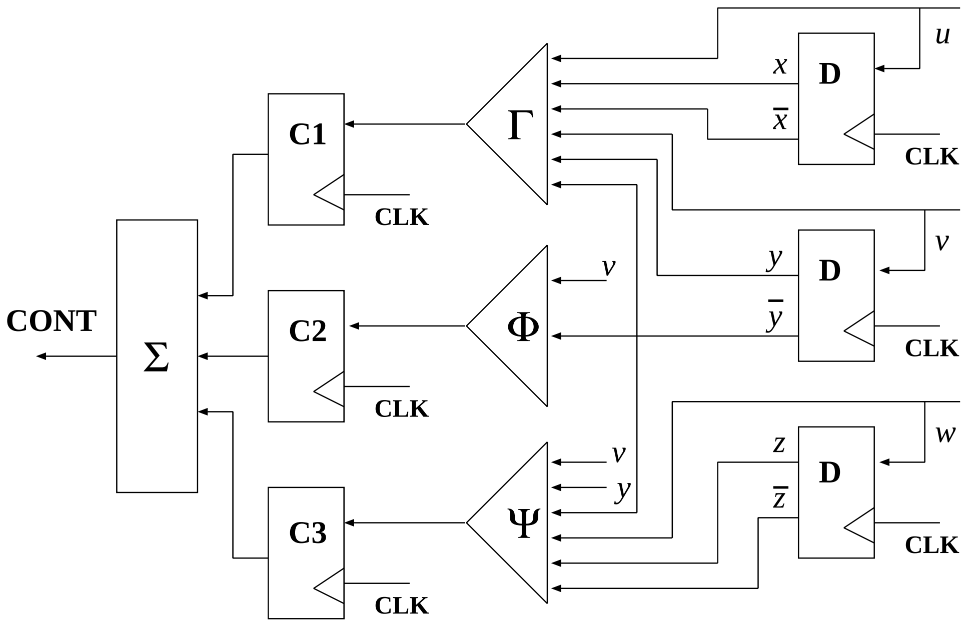


Figure 4. Architecture of Type 1 Processing Element

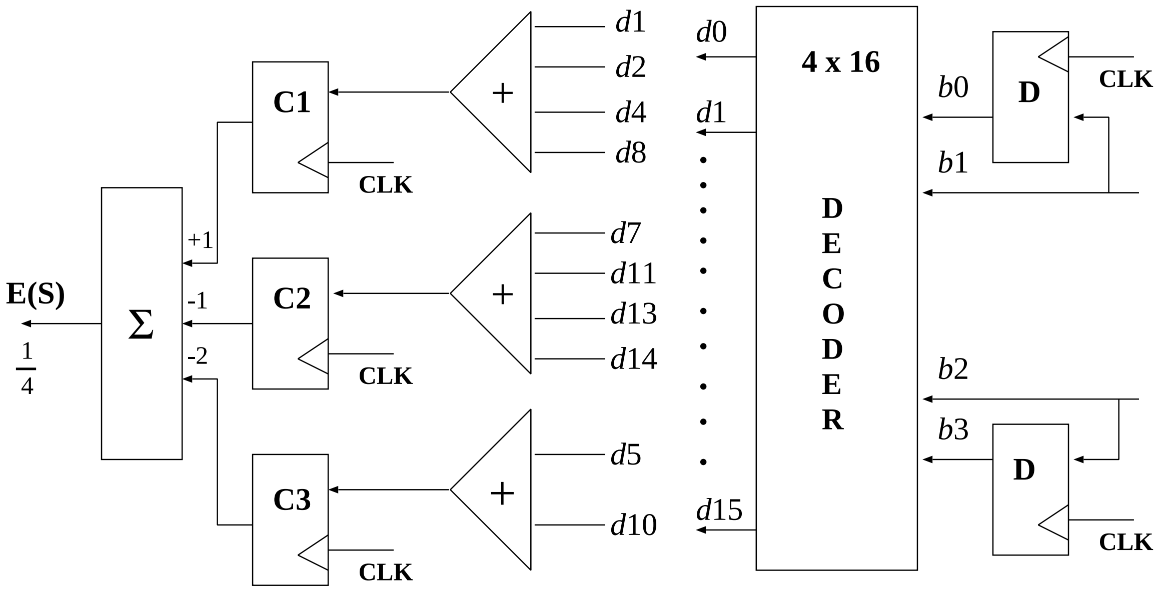


Figure 5. Architecture of Type 2 Processing Element

B. Type-2 processing element

We design a simple hardware that computes the Euler number by counting the number of patterns Q_1, Q_2, Q_3 (see Section 2) in the image S. The hardware is shown in Figure 5. Note that at any point we are looking into only four pixels. For a binary image, each pixel is just one bit. If we label the pixels from the top-left corner in clockwise fashion as $b0, b1, b2$, and $b3$ then we get a bit string of length four where the label of the pixel serves as its position in the string. The bit string is then fed to a (4×16) decoder. The output lines of the decoder are $d_0, d_1, \ldots, d_{15}$. Three counters C_1, C_2, C_3 count the number of patterns Q_1, Q_2, Q_3 respectively. The counters are set to zero initially and incremented as follows:

- C_1 is incremented when line d_1, d_2, d_4, or d_8 is set;
- C_2 is incremented when line d_7, d_{11}, d_{13}, or d_{14} is set;
- C_3 is incremented when line d_5 or d_{10} is set.

When all the pixels are visited, the Euler number is computed from the counter values by using equations (1) and (2).

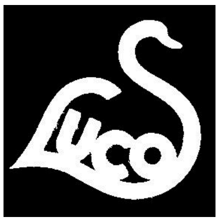
4.3. Computational complexity

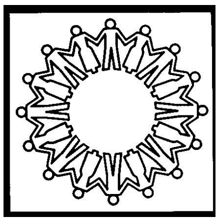
The proposed architecture requires $2K - 1$ PE's for an input image of size $N \times N$ which is partitioned into $4K$ atomic images. Let the maximum size of an atomic image be $n \times m$ $(n < m)$ where

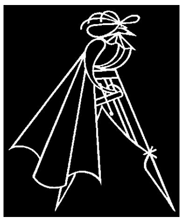
$$N^2 > knm \tag{4}$$

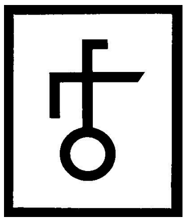
By incorporating a memory prefetch buffer, data can be supplied to the PE's in a single clock cycle. In step 1, all the PE's are active. The time needed for executing step 1 is the maximum time taken by a PE

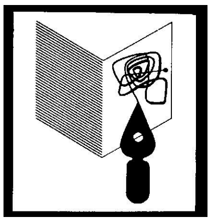
Figure 6. Some Logo Images and their Euler numbers (EN) used in our experiments.


(3) EN = 0

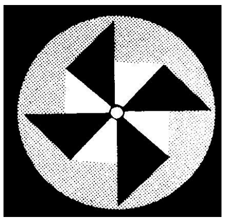

(39) EN = -17

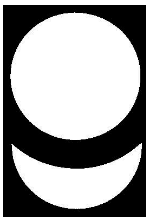

(43) EN = 37


(46) EN = -1


(49) EN = 13

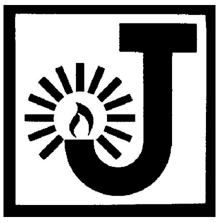
(58) EN = -2

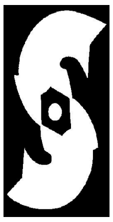

(62) EN = 2425

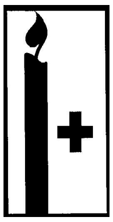

(68) EN = -1

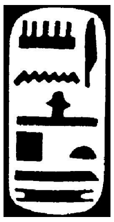
(73) EN = -1

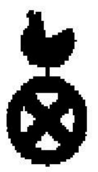
(76) EN = 3

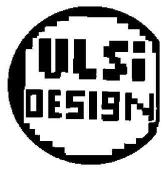

(80) EN = 15


(84) EN = 0


(94) EN = 0


(95) EN = 7


(102) EN = -3



(104) EN = 9

to compute procedure *Euler* of *Contrib*. In a CREW model each type-1 PE needs
(i) 1 clock cycle to read three pixel data,
(ii) 1 clock cycle for the combinational circuit, and
(iii) 1 clock cycle to drive the counters.
Moreover, these three operations can be executed in a three-stage pipeline. Hence, each type-1 PE takes P clock cycles where P is the length of the cut-line which is at the most N. Since Step 1a in the algorithm is executed in parallel, computation of contribution takes only $4N$ clock cycles. Each type-2 PE needs
(i) 1 clock cycle to read pixel data,
(ii) 1 clock cycle to decode and evaluate the Boolean functions, and
(iii) 1 clock cycle to drive the counters.
Similarly these three operations also can be executed in a three-stage pipeline. As step 1b in the algorithm is executed in parallel, computation of the Euler number of all the atomic images takes n^2 clock cycles by the type-2 PE's. If we make $n^2 < N$,the overall time requirement for step 1 becomes N. Step 2 is executed for $h = \log_2 K$ times. Now from eqn. (4) we have $K < \frac{N^2}{nm}$. Hence time requirement for step 2 is $O(\log_2 N)$, and the total computation needs $N + O(\log_2 N)$ clock cycles. Thus, the time complexity of the parallel algorithm is $O(N)$. As the number of PE's is $O(N)$ and computation time is $O(N)$, we achieve a speed-up of $O(N)$ for an $N \times N$ input image. The AT value [9] is $O(N^2)$ which is optimum as the sequential algorithm for computing the Euler number of an binary image takes $O(N^2)$ time for an $N \times N$ image [8]. The constraint $n^2 < N$ determines the value of MIN as mentioned in Section 3.2.

5. Conclusion

We have presented a fast recursive algorithm for computing the Euler number of a binary image that provides a significant savings in computation time. A parallel version of the algorithm is then described, which runs in $O(N)$ time on a tree architecture with $O(N)$ simple processing elements, for an $N \times N$ image. The $AT-$measure of the implementation is $O(N^2)$, which is optimum. A hardware implementation of the scheme has been proposed. Since the architecture needs simple interconnections among only two types of processing elements, it is highly suitable for on-chip VLSI implementation.

Acknowledgment

We thank Prof. Anil K. Jain and Dr. Aditya Vailya of the Michigan State University, USA, for providing us with the trademark database. An earlier version of this work appeared in the *Proc. International Conference on VLSI Design,* IEEE CS Press, pp. 330-335, 2000.

References

[1] Gonzalez R.C. and Woods R.F. Digital Image Processing. *Addison-Wesley,* Reading, Massachusetts, 1993, pp. 504-506.

[2] Greanis E.C. et al., “The Recognition of Handwritten Numerals by Contour Analysis”, *IBM J. Res. Dev.,* Vol. 7, No. 1, Jan. 1963, pp.14-21.

[3] Dyer C. R. “Computing the Euler number of an Image from its Quadtree”, *Comput. Graphics Image Processing,* Vol. 13, No. 3, pp.270-276, July 1980.

[4] Samet H. and Tamminen M. “Computing Geometric Properties of Images Represented by Linear Quadtrees”, *IEEE Trans. Pattern Anal. Mach. Intell.,* Vol. PAMI-7, No. 2, March 1985.

[5] Gray S.B. “Local Properties of Binary Images in Two Dimension”, *IEEE Trans. Computers,* Vol. 20, no. 5, May 1971, pp. 551-561.

[6] Minsky M. and Papert S. Perceptrons. *M. I. T. Press*, Cambridge, USA, 1968.

[7] Pratt W.K. Digital Image Processing. *John Wiley & Sons.,* 1978.

[8] Thompson C.M. and Shure L. Image Processing Toolbox. *The Math Works Inc.*

[9] Thompson C.D. A Complexity Theory for VLSI. *Ph.D. dissertation,* Dept. of Comp. Sc., CMU, Aug. 1990.

[10] Dey S. VLSI for Image Processing. *M.Tech. (CS) Thesis,* Indian Statistical Institute, Calcutta 700 035, India, July 1999.

[11] Jain A.K. and Vailaya A. “Shape-based Retrieval: A Case Study with Trademark Databases”, *Pattern Recognition,* Vol. 31, No. 9, pp. 1369-1390, 1998.

[12] Bishnu A., Bhattacharya B.B., Kundu M.K, Murthy C.A., and Acharya T. “A Pipeline Architecture for Computing Euler number of a Binary Image”, *Journal of Systems Architecture*, vol. 51, pp. 470-487, 2005.

[13] Bishnu A., Bhattacharya B.B., Kundu M.K, Murthy C.A., and Acharya T. “Euler Vector for Search and Retrieval of Gray-Tone Images”, *IEEE Trans. SMC, Part B*, vol. 35, pp. 801-812, Aug., 2005.