Abstract—In this paper, we present a parallel connected component labeling method and its VLSI architecture design. The proposed method can assign labels to three pixels simultaneously for the raster scan input and then generate three label equivalences rapidly. We also present 3 arrays to process all label mergerce. Based on the proposed method, we develop the hardware design for real-time application. The parallel architecture efficiently reduces total execution cycle significantly. From the experimental results, our 3-pixel labeling design can save 66% and 33% of the execution cycle comparing with the designs by 1-pixel labeling and 2-pixel labeling approaches, respectively.

Keywords—component labeling; VLSI architecture

I. INTRODUCTION

Connected component labeling (CCL) is an important operation in image analysis and machine vision. It can assign unique label for each connected component. Fig. 1 shows an example which assigns labels for two connected components. According to the labeled results, the features of each connected component can be extracted by calculating the information of the labeled image. Various algorithms have been proposed to deal with connected component labeling. A general classification is made with parallel processing [1-2] and sequential [3-5] processing approaches. Parallel approaches perform CCL in efficient time complexity but need to be implemented on parallel machine models with mesh-connected processors. The labeling can be completed by local operations to propagate label among processors in parallel until no label change. Sequential approaches use only single processor to label connected component through whole image. It leads to cost efficient but more time-consuming for label propagation. Among sequential algorithms, it can be separated into single-pass, two-pass [3-4, 6-8], and multiple-pass [5]. In single-pass, the algorithms find the object pixel in the scanning path and keep tracing connected pixel along the contour of component. It only scans whole image once but need to search and access pixel in irregular fashion. Two-pass method scan whole image twice regularly to merge two different labels in the connected component by recording label equivalences. For the multiple-pass method, it removes the requirement of equivalence mechanism but needs to scan whole image iteratively until no further label change is occurred. It causes more time complexity and unpredictable for completing the whole labeling. For the requirement of real-time applications, regular memory access and simple operation are more suitable for hardware implementation [8]. Based on two-pass scheme, several hardware designs have been developed. [3-4, 7] presented one-pixel labeling architectures and [6] proposed row-based labeling architecture. However, one-pixel designs are more time-consuming and row-based design needs large amount of hardware cost for parallel operation. In order to compromise timing and cost complexity, the specific hardware designs for parallel labeling have been developed in [8-9]. [8] presented two-pixel-based labeling with linear array architecture while four-pixel-based parallel architectures are presented in [9]. Parallel approach can deal with multiple pixels to short process time but generating multiple label equivalences that restrict the scalability.

In this paper, we present parallel execution method with 3-pixel labeling approach. This method uses two-pass scheme. In the first pass, our method performs simultaneously assigns three input pixels for raster scan image buffer, and generates the label equivalences in parallel. Next, the second pass merges the related label equivalences, and updates the label assigned in the first pass.

The rest of this paper is organized as follows. Sec. 2 describes the 3-pixel labeling method for parallel connected component labeling. Based on our proposed approach, the VLSI architecture design and implementation are shown in Sec. 3. Sec. 4 shows the performance comparison with different approaches and simulation result. Finally, a conclusion is made in Sec. 5.

Fig. 1 The labeling for two connected components
II. THE 3-PIXEL LABELING METHOD

In general, the connected component labeling uses the previous neighbor labeled result to determine the label assignment for the current input pixel. In this section, we firstly illustrate the 3-pixel labeling window, the manner of label selection and label equivalence generation. The intra and inter equivalence label merging is illustrated for final label update.

A. Basis label selection and equivalence label

Fig. 2(a) shows typical labeling window that scans the binary image from left to right and top to bottom. The box denoted as B stands for the input binary data while the box denoted as L represents the labeled data of the neighbor pixel. In labeling assignment, four labeled data must be taken into account for assigning their labels to the input binary pixel when the input is an object pixel. The assigning strategy is that L4 has highest priority while L1 has lowest priority. On the other hand, the cases of label equivalence generation are described in Fig. 2(b). Since different labels meet through the input binary data, the phenomenon indicates that these labels represent the same connected component. We only consider two situations as shown in Fig. 2 (b), because the problem occurs on U-shape and V-shape patterns.

![Fig. 2 The basic labeling strategy (a) label assign scheme (b) equivalence label cases](image)

B. The proposed 3-pixel label assignment

As mentioned before, multiple pixel labeling can speed up the label assigning time to match the requirement of real-time applications. We extend the single pixel label assignment to 3-pixel label assignment to perform multiple pixels labeling on three consecutive rows in parallel. Fig. 3 shows the labeling window for processing 3 pixels in parallel. Pixels L1 to L8 are labeled data which have been assigned when their binary pixel inputted, and the current input pixels B1, B2 and B3 are binary pixels.

![Fig. 3 The structure of 3-pixel labeling window](image)

C. The equivalence label merging

In two-pass labeling scheme, merging all relative labels that belong to the same component is more critical than label assignment. Since 3-pixel label assignment will result in variable label equivalences generating at the same time. We employ three arrays to save these label equivalences. The merging method is described in Algorithm 1. The algorithm includes three steps. Initially, the content of each array is set as number with its position. The first step, called intra-equivalence label merge, records and updates the individual array independently. The second step, called inter-equivalence label merge, checks and updates three arrays each other. Finally, temporal label lookup arrays to obtain final labeled data.

![Fig. 4 The temporal label image and label equivalence set after label assignment process](image)

**Algorithm 1 Parallel equivalence label merge**

Initialize every class array in increase order.

Step 1: Intra-equivalence label merge

Step 2: Inter-equivalence label merge

Step 3: Final label replacement

Fig. 5 shows the arrays for label merge according to Fig. 4(b). In initial state, all arrays are set. As the label equivalences are generated in record state, the corresponding array will be updated independently. For example, the array 1 receive equivalence label (5,6) and (6, 7). After inter-
equivalence label merge, both positions 5, 6 and 7 are replaced by label 5.

array 1 | array 2 | array 3
---|---|---
0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
(5,6) | (2,3) | (1,2)

(record state) (2,3,4) (1,2,3)

Fig. 5 The intra-equivalence label merge

Due to each array received different label equivalences that results in different record among three arrays. In order to correct all arrays, we present inter-equivalence label merge. Subsequently, we search 3 arrays on the same position to update with the minimum label, as the shown in Fig. 6. For the position 2, we extract three labels {2, 2, 1}. The smallest value is 1, which is applied to replace other arrays.

array 1 | array 2 | array 3
---|---|---
0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
0 | 1 | 1 | 3 | 3 | 5 | 5 | 5 | 0 | 1 | 1 | 3 | 3 | 5 | 5 | 5 | 0 | 1 | 1 | 3 | 3 | 5 | 5 | 5

(a) array correlated for position 2

Fig. 6 The inter-equivalence label merge

After inter-equivalence label merge, all arrays become the same result that the position 1, 2, 3, and 4 are replaced by 1, while position 5, 6, and 7 are replaced by 5. The contents of 3 arrays are consistent and can be used to look up for final labeled data.

III. THE HARDWARE ARCHITECTURE OF 3-Pixel LABELING

The block diagram of proposed 3-pixel labeling is shown in Fig. 6. The input binary data B1, B2, and B3 enter temporal label assign block (TLAB) while neighbor label buffer provides all labeled data from L1 to L8. The TLAB selects the desired label according to all related labels. The neighbor label buffer stores recent labeled data that is referenced in following label assign operation. After label assignment operation, the temporal label results are stored into temporal label buffer. Another function is label equivalence generation block (LEG) which generates label equivalence LE if two different labels meet through the input object pixel. The generated equivalence label will send to label merge block which uses three arrays to record associated label equivalences. After label merging operation, these labels in temporal label buffer can be replaced for final labeled data.

Fig. 7 The block diagram for 3-pixel labeling

IV. PERFORMANCE AND EXPERIMENTAL RESULT

Several hardware designs for connected component labeling have been developed to reduce execution cycles for real-time applications. In this section, we demonstrate the execution cycles of 3-pixel labeling and compare with 1-pixel labeling and 2-pixel labeling approaches and the implementation result.

A. Execution cycle analysis

The execution cycles for 3-pixel labeling comprise three phases: label assigning, label merging and label replacement. In label assigning phase, it performs three pixels label assignment for three consecutive rows in a clock cycle. Since the arrangement of binary pixels in diagonal, each scan spent N+2 cycles for N × N image. In label merging phase, the merge cycle, which depends on the maximum label, will be used to unify three arrays. Finally, the label replacement for each scan needs N cycles. As the result, the total execution cycle is given by

\[
\text{Execution Cycles} = 0.67 N (N+1) + MC
\]

where MC is the merge cycle. We compare our 3-pixel labeling with recent developed designs in terms of execution cycles as shown in Fig. 8. We assume the maximum label is 255 for representing labels in 8-bit. The result shows that our 3-pixel labeling can reduce 66% and 33% clock cycle as comparing with 1-pixel labeling and 2-pixel labeling approaches.

B. Simulation and Result

The proposed architecture has been implemented in Verilog HDL and simulated in Modelsim simulation tool. Fig. 9 shows the simulation waveform where receives two object pixels in first process unit and third process unit to be assigned label with 1 and 2, respectively.
V. CONCLUSION

A parallel 3-pixel labeling and its hardware design are proposed in this paper. Parallel labeling will cause multiple equivalence labels, we exploit three arrays to record associated label equivalence and merge each other. From the comparison, our 3-pixel labeling uses lower execution cycles. It is more attractive for real-time applications.

REFERENCES


