A VLSI architecture for real-time edge linking

Amjad Hajjar and Tom Chen
Department of Electrical Engineering
Colorado State University
Fort Collins, CO 80523
Tel: (970) 491-6574, Fax: (970) 491-2249
Email: amjad@lance.colostate.edu, chen@lance.colostate.edu

Abstract
Image recognition may consist of three main steps: edge detection, edge linking, and object/template matching. Edge detection algorithms usually produce thin edges with discontinuities. In this paper, a real-time algorithm and its VLSI implementation for linking broken edges is presented. First, all broken edge points inside a 12×12 moving window are identified. The 12×12 window scans the input gray level edge map converting it into three levels of intensities using two threshold values. Decisions of linking the stronger edge break points are made based on their directions and/or with the guidance of the weak edge lines. The proposed VLSI architecture is capable of running the proposed edge linking algorithm in real-time outputting one pixel of the linked edge map per clock cycle with a latency of *11n+12* clock cycles, where *n* is the number of pixel columns in the image.

Keywords
VLSI, Edge Linking, Real-Time Image Processing

1 INTRODUCTION

One of the most important features in an image for object recognition is edge information (Marr, *et. al.* 1980). Experiments in the past have shown that human visual system reacts strongly to sharp changes in pixel intensity in an image. Robot systems and many other computer vision applications, as well, are based on using the edge information to extract certain features within the environment (William, *et. al.* 1989).

An edge point, then, is a sudden change in the intensity level of an image over a number of pixels either in a horizontal and/or a vertical directions (Alzahrani, *et. al.* 1997, Ungureanu, *et. al.* 1993, Farag, *et. al.* 1991, Farag, *et. al.* 1995, Xie 1992). But, edges are not all the same. Some of them are much sharper, or stronger, than the others depending on the gray levels of the adjacent areas.

Edge detection algorithms were developed in the past to produce high quality edge maps. The focus of most of these algorithms is to ensure good quality of the edge output regardless of time consumed to produce them. Besides, edge detectors based on the first derivative do not guarantee to produce edge maps with continuous edge contours nor unwanted branches (William, *et. al.* 1989, Breen, *et. al.* 1994, Bernand 1994, Diamond 1983). Edge detectors based on the second derivatives, such as zero-crossing, suffer wrong edge detection in textured images and noisy images. Many object recognition algorithms prefer closed contours of objects for comparisons and matching. Thus, the output edge maps produced by the first type edge detectors may not be useful in follow-up processing steps.

In this paper, we propose a real-time edge linking algorithm and its VLSI architecture that is capable of producing binary edge maps in a rate up to 28.5MHz clock frequency. Our algorithm employs a moving window of size 12×12 pixels that scans the input images. For a VGA sized image, only 12 shift registers of 480 cells are needed to store 12 rows of the input image at a time. The delay time of producing edge maps of different input frames is zero and only a latency of 12×480 clock cycle presents.

2 EXISTING EDGE LINKING ALGORITHMS

William and Shah (William, *et. al.* 1989) proposed a Multiple Scale edge linking algorithm such that all edge points produced by the Canny edge detector (Canny 1986) are stacked in a queue. The output edge map produced by the Multiple Scale algorithm are of high quality and the edge contours are closed and connected. However, the complexity of the algorithm is at the order of 3^n, where n is the number of the image pixels. An edge linking algorithm using a Causal Neighborhood Window was proposed by Xie (Xie 1992) to achieve lower computational complexity. His algorithm performs poorly and produces non-localized edge points when texture images are used. Farag and Delp (Farag, *et. al.* 1991) proposed a linear path metric function for a sequential search process. Their algorithm involves less amount of calculations compared to the Multiple Scale algorithm and performs well for textured images. However, some statistics such as σ and a priori information about the processed images have to be known. Miller and Madeda (Miller, *et. al.* 1993) proposed a different type of edge linking algorithm using template based method. This template based algorithm scans the image three time to produce the final edge map.

Earlier attempts to map edge linking algorithms into VLSI included work by (Ungureanu, *et. al.* 1993) who used a programmable gate array chip of ACTEL type to produce linked closed edge contours. Their algorithm involves path search procedures to track the contour lines. The processed image pixels

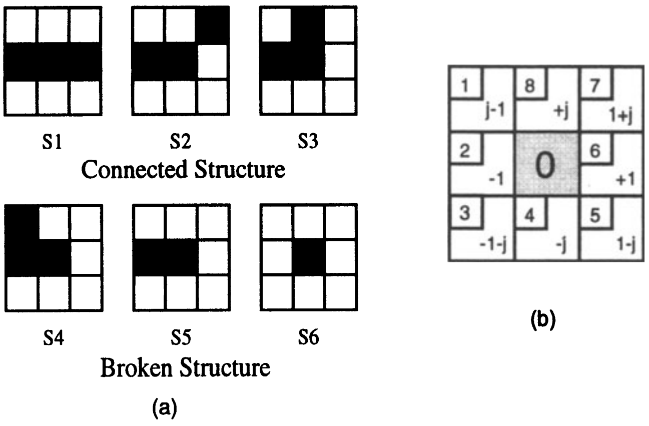

Figure 1 (a) The Basic Structures (b) The direction representation and position.

should first be stored in an off-chip memory before the linking process starts. An initial edge point should be chosen manually for the chip to start tracking the contour lines from it. The time needed to process one image exceeds 30-50 ms for images of size 256×256.

3 THE EDGE LINKING ALGORITHM

3.1 Basic Concepts

In this paper, we use *six basic structures* of which all edge lines take at any point. These basic structures form a complete set of edge line shapes, and they are divided into *connected basic structures and broken basic structures* as shown in Figure 1(a). Edge lines can be decomposed into a number of basic structures in many different ways. In general, we define a break point as *an edge point from which it is impossible to extract any connected basic structure.*

Once a break point in some position in an edge map is identified, its direction must also be determined. We define the eight directions by using complex number notations as shown in Figure 1(b). The real part of a break point direction represents the horizontal shift that the next linking edge point should take, and the imaginary part represents the vertical shift of it. A break point at position *(x,y)* is said to have a direction *a+jb, where a,b* ϵ {*-1,0,+1*}, if it is connected to a previous edge point located at the position *(x-a,y-b).*

To guide the edge linking process effectively, some weak edge points are preserved during the intensity edge map conversion. Thus, two different threshold values are used such that the larger value is used to produce only the wanted edge points without any branches and the smaller threshold value is used to preserve some weak edge points. The first threshold value is referred to as *the strong threshold.* The other threshold value is referred to as *the weak threshold.* The edge map output resulted from using two thresholds is referred to as *the trinary edge map.*

3.2 The Algorithm

When using two threshold values to produce the trinary edge map, finding the break points and determining their directions is applied only for the strong edge map. Unlike some existing edge linking algorithms which tend to operate on a large area of an image thus requiring large amount of memory, our approach takes a small portion of an image at a time utilizing the weak edge points as guides to maintain accuracy while requiring much smaller amount of memory. The small portion area is marked as *window.* The entire image is then scanned by the window once to complete the edge linking process. Assuming that the image pixels are fetched in a row order form, top left to bottom right, the scanning pattern follows the same direction. Generally speaking, the choice of window size depends on the requirement for edge linking accuracy and the amount of memory and other hardware related constraints, bearing in mind that our goal is to map the algorithm onto a VLSI chip. The tradeoff resulted in the choice of a 10×10 window size with a need of the information from the surrounding pixels. Thus, a window size of 12×12 is chosen.

Once the window is moved to a new location, all break points inside the window will be counted; their directions will also be determined. In order to start linking break points, the window itself is divided into four sub-windows; each sub-window has the size of 5×5 pixels from the core area. The linking process will start first within each sub-window. Once the process is completed, another linking process between sub-windows will start. The edge linking algorithm consists of the following steps:

Step 1 Intra-Column Linking

For each column in each sub-window, there are three scenarios with regard to break points. First, if there is no break points in a column, no action is required and move to the next step. Second, if there is one break point in a column, record its presence and move to the next step. Third, if there are two break points in a column, check the directions of these break points by testing the direction values using the following condition:

$$Link = (\mathbf{P}.\mathbf{d_1} \leq cos\theta) \wedge (-\mathbf{P}.\mathbf{d_2} \leq cos\theta) \wedge (-\mathbf{d_1}.\mathbf{d_2} \leq cos\theta), \quad (1)$$

where: $\mathbf{P} = (x_2 - x_1) + j(y_2 - y_1)$, the position vector; d_i is the direction of the break point i; (x_i, y_i) is the location of the break point i; [.] denotes the vectors dot product; θ is the maximum allowed directional difference between two break points; and θ is set to 45^o

Step 2 Inter-Column Linking

In this step, one pair of the unlinked break points from step 1 will be tried to be linked across different columns using the same direction matching criterion. The way of selecting the pair of break points in this step is to choose the pair with the farthest horizontal distance. Choosing the farthest distance points instead of the closest was based on some experimental observations which indicates better linking among broken edges. If the selected pair cannot be linked, one of the break points will be tried to be extended with the guidance of some nearly weak edge points if possible. The other break point in the pair is passed on to the next step for further processing. All the remaining unlinked break points are to be tried again when the window is moved to the next location.

Step 3 Inter-Sub-Window Linking

During this step, each sub-window may have one break point that has to be linked with the others. Thus, a maximum of 4 break points are to be considered within the entire window. There are 6 possible combinations of pairing any two break points out of possible 4. Linking will be tried on each pair using the same direction matching criterion. After linking the break points, any unlinked break point will be tried to be extended using nearly weak edge points as guidance.

Step 4 Changing Window Location

In this step, the window is moved to the next location and steps 1 to 3 are repeated.

4 EXPERIMENTAL RESULTS

The Lenna image was used to determine the effectiveness of the proposed edge linking algorithm. Applying the ADM edge detection algorithm (Alzahrani, *et. al.* 1997) to the Lenna image, the intensity edge map produced is shown in Figure 2(a). This intensity edge map is the input to our edge linking algorithm. The trinary edge map produced by the two thresholding process is shown in Figure 2(b).The edge linking algorithm outputs the final linked edge map shown in Figure 2(c).

5 VLSI IMPLEMENTATION OF THE EDGE LINKING ALGORITHM

In order to allow real-time edge linking, our edge linking algorithm was mapped into hardware. The algorithm was first translated into the structural level design using VHDL and verified before gate level design took place. The edge linking circuit is organized in three major functional blocks: The Receiver Block, the Loop Block, and the Mask Block. Figure 3(a) is the top level block diagram which shows the signal flow of the image coming out from the edge

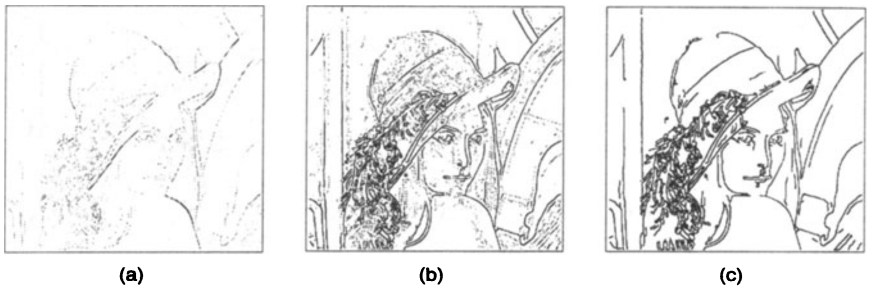

Figure 2 (a) The intensity edge map of ADM algorithm (b) The trinary edge map by the proposed algorithm (c) The final output edge map.

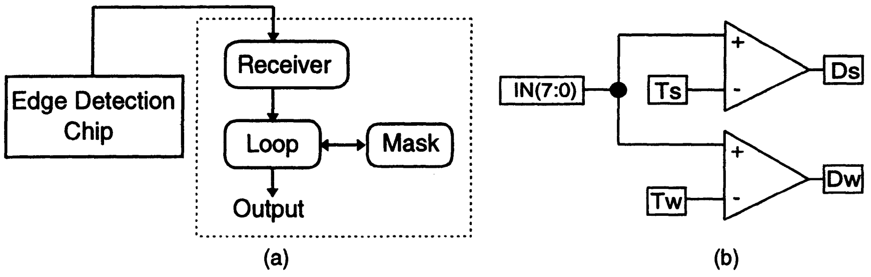

Figure 3 (a) The overall circuit diagram (b) The Receiver block diagram.

detection chip and passing through the proposed edge linking units till it is finally produced.

The output signal from the ADM edge detection chip (Alzahrani, *et. al.* 1997) is a serial flow of the intensity edge map pixels on an 8-bit bus. This 8-bit signal is to be mapped into 2-bit signal using two threshold values. Figure 3(b) shows the circuit design for the Receiver block. Two 8-digit comparators are contained in this block to produce two binary signals D_s and D_w such that $D_s = \text{IN} \geq T_s$, $D_w = \text{IN} \geq T_w$, where: IN is the input signal of 8 bits, T_s, T_w are the strong and weak threshold values, respectively.

The main function of the Loop block is to prepare the image data for the Mask block to produce final output edge map. At the beginning of the linking process, the data of a 12×12 block from the top-left corner of the image must first be ready. Therefore, the first twelve rows of the image need to be stored in shift registers. Once the data of the first 12×12 block are ready in the Mask block, the linking process starts. Figure 4 shows the signal flow inside the Loop and the Mask blocks. Notice that we divided the Loop into three pieces to illustrate the circular movement of the signal. The Loop block contains 468×12 storage cells. Each cell has two D flip-flops: one to pass the strong edge points map and the other is for the weak edge points map.

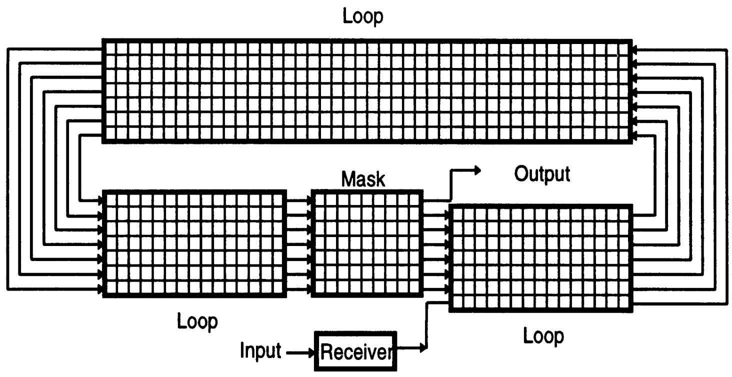

Figure 4 The signal flow in the edge linking circuits.

The Mask block contains 10×10 internal Cell units to accommodate the core part of the moving window, as described in the edge linking algorithm section. Figure 5 shows the arrangement of the units inside the Mask block. There are 44 Coat units located around the core area making the Mask window 12×12 in size. The Coat cells are used to provide boundary information to the 10×10 mask, no edge linking process is applied to them. The Mask block itself is divided internally into 4 sub-windows, each having a size of 6×6 pixels. The communication between the Cell units is conducted via Net units. Within each sub-window, the Controller5 unit is responsible for the Intra-Column and Inter-Column linking processes. Links between the sub-windows is controlled by the Controller10 unit. In general, we can list the functionality of the Mask as follows:

- The Cell units identify all break points and determine their directions. Figure 6 shows its schematic diagram.
- Through the Net units, the signals from each Cell unit in a sub-window are sent to the Controller5 in order to process the Intra-Column links and the Inter-Column links.
- Each Controller5 selects a pair of break points, if any, from each column using five MuxC units. Figure 7(a) shows the logic structure of the MuxC unit. The MUX5x2 selects two break points with the longest distance and passes them to the follow-up logic for further identification.
- The pairs from each column is to be linked after checking their directions. Five Match5x1 units are used for this purpose, one for each column. Figure 7(b) shows the design of a Match5x1 unit.
- The break points information from the each column is then passed to the MuxR unit to perform step 2 of the algorithm (Inter-Column linking). The way MuxR works is very similar to that of MuxC in the sense that it chooses two break points from among five columns. It then outputs the information

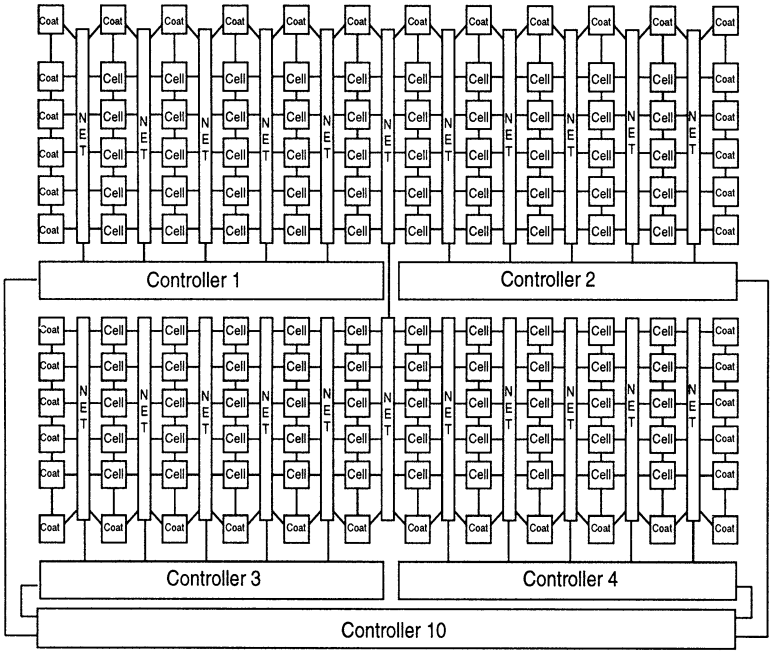

Figure 5 The Mask block diagram.

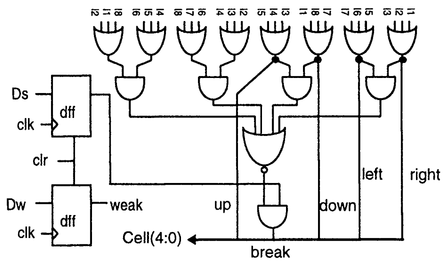

Figure 6 The Cell unit.

of the two selected break points from two farthest possible columns to the Match5x2 unit.

- The Match5x2 unit checks the direction agreement of the two input break points coming from the MuxR unit. Similarly, it passes the location of the two break points along with the decision signal to link them.
- In the Link5x2 unit, two stages are needed to draw a line between the two break points if the linking decision is high. In each stage, two Step5x5 units are used to fill the gap between the two break points. The output from the two Step5x5 units in the first stage is the X-Y location of two new points

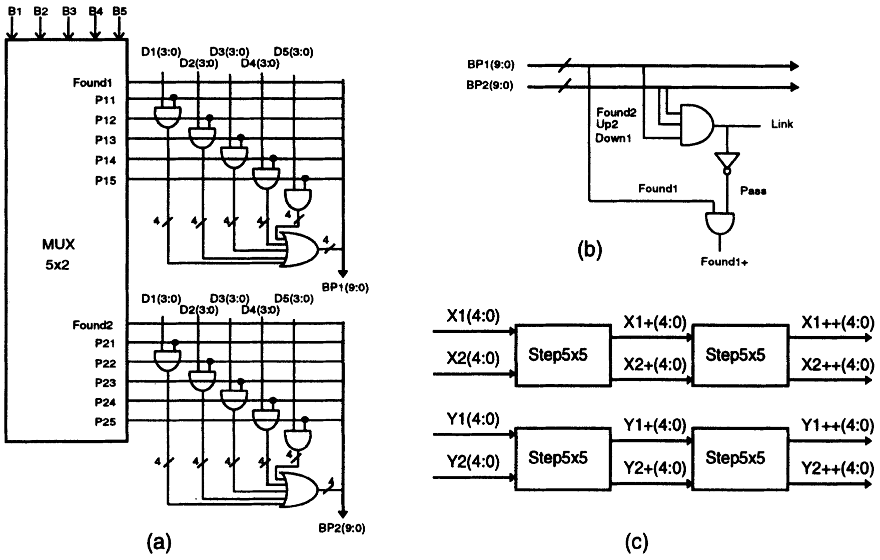

Figure 7 (a) The MuxC unit (b) The Match5x2 unit (c) The Link5x2 block diagram.

such that the gap between them is shorter than the original gap of the break points. Another stage of a similar process is then enough to connect the break point by a linking line. The block diagram for the Link5x2 unit is shown in Figure 7(c).

- If the linking decision signal from the Match5x2 unit is low (decision for not linking), one of the selected break points will be fed to a Weak unit. The function of the Weak units is to extend the input break point one or two pixels toward the break direction if there is weak edge points nearby. The advantage of extending break points along their directions on a weak edge path is to shorten some long gaps between break points. This way, the linking process inside the sub-window becomes effective and sufficient.
- All the functional units mentioned above constitute the Controller5 unit as shown in Figure 8.
- The other unlinked break point left from the Match5x2 unit will be sent to the Controller10 unit which receives up to four possible break points from the four sub-windows. Six possible comparisons will be conducted to draw linking lines at the Inter-Sub-Window linking stage.

6 COMPLEXITY ANALYSIS, CHIP SIZE ESTIMATIONS

The entire edge linking circuit consists of about 10,000 logic gates and 11,520 flip-flops. Estimation of the chip area was carried out by using a $0.8\mu m$ CMOS

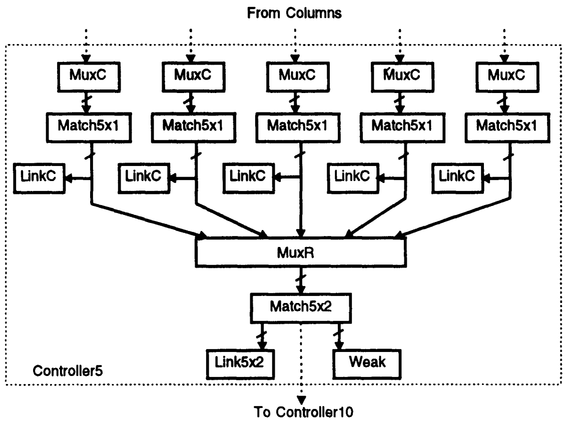

Figure 8 The Controller5 block diagram.

standard cell library. Adding up the areas of the all gates used in the circuit, we get approximately the overall gate area of the chip to be 10 mm^2. The routing area is approximated to be the same as the gate area needed on the chip. Thus, the total chip area is about 20 mm^2.

A critical path measurement is also carried out to determine the potential performance of the edge linking chip. Tracing the critical path in the circuit, we find the critical path consisting of 35 cascaded gates. Figure 9 shows the critical path indicating the delays of each unit at its output. The maximum fanout among these gates on the critical path is 4. Assuming a conservative average gate delay of 1ns using the $0.8\mu m$ CMOS technology, the minimum clock cycle time is 35 nsec. Thus, the maximum allowable clock frequency is about 28.5 MHz. Considering that the chip is capable of outputting a pixel per clock cycle for the linked edge map, this is 3 times the video rate requirement for VGA-sized images, or 1.2 times the video rate requirement for SVGA-sized images.

7 SIMULATION RESULTS OF THE VLSI ARCHITECTURE

The Lenna image was processed by the circuit through a logic simulator as well as a structural level VHDL simulator. Both the schematic logic circuit and the VHDL code are tested using the Lenna image. Figure 10 shows the output edge map of the Lenna image from the simulated VLSI circuit and from the simulated VHDL code. The threshold values used for the Lenna image were $T_s = 40$ and $T_w = 20$.

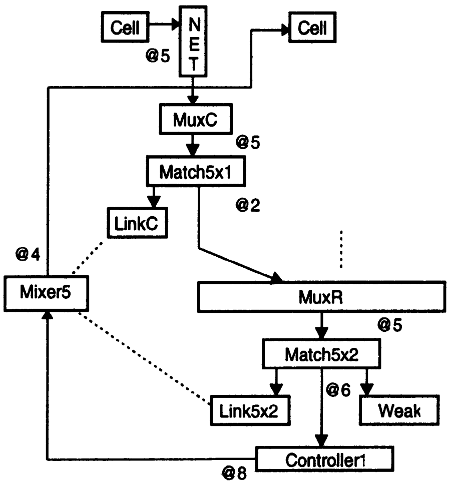

Figure 9 The critical path.

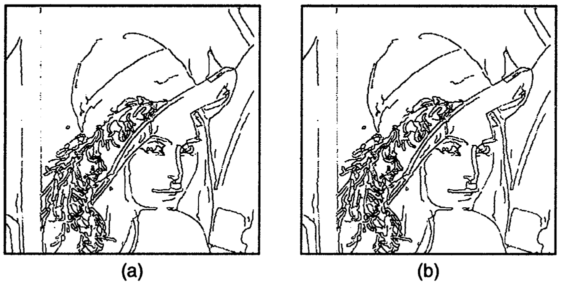

Figure 10 (a) The VHDL code output (b) The logic gate circuits output.

8 CONCLUSION

In this paper, we have presented a new algorithm for edge linking and its VLSI implementation. The software implementation, using a C++ code, was proven to give good output results through four major steps of edge linking: the Intra-Column Linking step, the Inter-Column Linking step, the Inter-Sub-window Linking step, and the Changing Window Location step.

Although our algorithm does not produce better results than other existing edge linking algorithm, the hardware implementations of it have been developed and proven to produce output edge maps in real-time processing environment. In implementing the VLSI circuits, we tried to make the design as simple as possible in order to have smaller die size. The strategy used while building the circuits in the gate level is to have symmetric designs with less inter metal wiring between blocks. The proposed edge linking circuit is able to process 33 VGA/SVGA frames per second in a normal operation mode with the clock frequency of up to 28.5 MHz.

The proposed edge linking circuit has two major limitations. First, the output edge maps are not guaranteed to have closed contours. It maybe the same situation for other well-known edge linking algorithm, but we should consider it as a problem. Second is that the edge linking process is sensitive to the choice of the strong threshold values. That is, fixing an initial number for the edge linking chip to be a permanent threshold value for all possible input images would result in poor edge linking.

One possible solution to the threshold problem is to build a feedback controller circuit that adjusts the threshold values such that a feedback signal from the image matching circuit gives the matching percentage of the detected image versus a known template image. The control circuit then increases or decreases the threshold value correspondingly.

REFERENCES

Alzahrani, F. and Chen, T. (1997) A real-time edge detector: algorithm and VLSI architecture. *to appear in Journal of Real-Time Imaging.*

Bernand, T. (1994) Object contour tracking as inspired by the MAD retina paradigm. *Proceedings of ICIP.* 13-16.

Breen, E. and Peden, G. (1994) Automatic thresholding and edge linking of ferritic steel weld images. *Computer Assisted.* **V. 6, N. 4.**

Canny, J. (1986) A computational approach to edge detection. *IEEE Trans. on PAMI.* **V. PAMI-8, N. 6.** 679-98.

Diamond, M. (1983) The graph labeling model and its application to the problem of edge linking. *The University of Michigan.* **Dissertation.**

Farag, A. and Delp, E. (1991) A path metric for sequential search and its application in linking. *IEEE Conf. Sys., Man, & Cyb..* **V. 1.** 563-8.

Farag A. and Delp, E. (1995) Edge linking by sequential search. *Pattern Recognition.* **V. 28, N. 5.** 611-33.

Marr, D. and Hildreth, E. (1980) Theory of Edge Detection. *Proceedings of the Royal Society of London.* **Series B, V. 207.** 187-217.

Miller, F. and Madeda, J. (1993) Template Based Method Of Edge Linking With Low Distortion. *IEICE Trans. Inf. & Sys..* **V. E76-D, N. 6.**

Ungureanu, D. et. al. (1993) A Real Time Edge Linker. *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition.* 793-4.

William, C. and Shah, M. (1989) Multiple Scale Edge Linking. *SPIE, Application of Artificial Intelligence VII.* **V. 1095.**

Xie, M. (1992) Edge Linking By Using Causal Neighborhood Window. *Pattern Recognition.* **Letter 13.**