CHAPTER 5 - HARDWARE IMPLEMENTATION OF SKELETONIZATION ON HEXAGONAL GRID

5.1 Introduction

Skeletonization operation plays an important role in many preprocessing stages. The skeleton of an object is a line connecting point’s midway between the boundaries. It is an important representation for the shape analysis which is useful for many pattern recognition applications. A skeleton has the same connectedness ("topology") as the original object. Present-day sensor and signal processing systems are challenged by complex, dynamic environments and often require real-time processing capabilities. An attempt is made in this work to design architecture for skeletonization and reconstruction on the hexagonal grid with the proposed modified algorithm.

Many algorithms exists to find the skeleton of an image [67, 68], [21, 71, 119-122]. Among them, there are two main approaches for skeleton extraction, based either on iterative thinning or distance transform. Although the latter methods are less sensitive to a boundary noise, methods based on iterative thinning has several features, which become beneficial when the primary goal is a VLSI implementation. Nikolaos Bourbakis etc., [123] implemented an application specific array processor (ASAP) desirable for high speed parallel skeletonization of binary images. Sudha [124] designed the architecture for skeletonization based on distance transform method. These architectures were designed in order to obtain skeleton of images represented on rectangular grid.

N. Lopich and Dudek [59] designed the architecture for skeletonization of binary image on hexagonal lattice using asynchronous cellular processor array. Although the above architecture was consuming less power, many resources of FPGA are wasted as input latches and output latches. In the proposed architecture, the system clock is used for the input and output latches, instead of clock signal generated by the control unit using the modified algorithm (Section 5.4) and the hardware requirement was found to be less. The performance was also compared with rectangular array processor. An iterative thinning algorithm has been chosen, since it overcomes demerits of distance transform method by transferring data only between the nearest neighbors. As
the rectangular processing element deals with 8 neighbors, it is using more logic elements than the hexagonal processing element which has 6 neighbors. Hexagonal processing element saves 54% hardware, when compared to rectangular processing element using the proposed method.

5.2 Overview of Skeletonization

Skeletonization or thinning operation is done by iteratively eroding away nonessential pixels from the image until one is left with a stick figure, or skeleton, which describes the object (Figure 5.1).

![Figure 5.1 Examples of skeleton images](image)

The skeleton hence obtained must have the following properties: 1) as thin as possible 2) connected 3) centered. If we restrict ourselves to a 3 x 3 neighborhood, then we can view the thinning operation as a window that repeatedly scans over the (binary) image and sets the center pixel to “0” under certain conditions. There are some general requirements that affect the design of the algorithm such as 1) Thinness 2) The skeleton should have the same connectivity (topology) as the original pattern 3) Once a skeleton (or part of it) is obtained, say after \( n \) passes, it should not be eroded away by subsequent passes. Criteria for good thinning algorithms include sensitivity to noise, preservation of topological and geometric properties, isotropy, ability to reconstruct and efficiency.

All thinning algorithms can be classified as one of two broad categories 1) Iterative thinning algorithms 2) Non-iterative thinning algorithms (Figure 5.2). In general, iterative thinning algorithms perform pixel-by-pixel operations until a suitable skeleton is obtained. Iterative algorithms may be classified as either sequential or parallel. Sequential thinning algorithms which examine contour points of an object in a predetermined order. In parallel thinning
algorithms the decision for individual pixel deletion is based on the results of
the previous iteration.

Non-iterative thinning is not based on examining individual pixels. Some popular non pixel based methods include medial axis transforms, distance transforms, and determination of centerlines by line following. Medial axis transforms often use gray-level images where pixel intensity represents distance to the boundary of the object. Distance transform based methods compute the distance to the image background for each object pixel and use this information to determine which pixels are part of the skeleton.

5.2 Classification of thinning algorithms

Due to their central role in the preprocessing stage, thinning algorithms have been a very active area of research from the sixties (more than 300 published papers until now). Application areas include handwritten / character recognition, classification of cells, chromosomes, X-ray image analysis, analysis of coronary arteries, processing of bubble chamber negatives, visual system of automaton, fingerprint classification, measurement of soil cracking patterns, automatic visual analysis of industrial parts and printed circuit boards, etc., Classically, skeletonization algorithms are implemented on software. With the advances in the VLSI technology hardware implementation has become an attractive speedy alternative.
5.3 Algorithm for Skeletonization on Hexagonal Grid

N. Lopich and Dudek [59] designed the architecture for skeletonization of binary image on hexagonal lattice using using Hilditch’s skeletonization algorithm [125]. Hilditch thinning algorithm is widely used as a useful method of pre-processing in image processing. Hilditch’s algorithm for hexagonal sampled grid is as follows. A set of 6-neighbourhood of a pixel $P_{ij}$ is defined and the notation is shown as below.

$$N_6 = \{P_{ij}^1, P_{ij}^2, P_{ij}^3, P_{ij}^4, P_{ij}^5, P_{ij}^6\}$$ (5.1)

The input binary image consists of background pixels (logic ‘1’) and object pixels (logic ‘0’) on a hexagonal lattice.

![Figure 5.3 Local hexagonal PE neighborhood $N(\ P_{ij})\ $](image)

The algorithm for skeletonization consists of two steps. By considering these 7 pixels (centre pixel $P_{ij}$ and its 6 neighbors), three functions are evaluated and three conditions are verified. Let us define three functions:

1. $$B_y = \sum_{k=0}^{5} \overline{P_{ij}^k}$$ (5.2)

2. $$A_{ij} = \left| \text{sign} \left( \sum_{k=0}^{5} \left( P_{ij}^k \land \overline{P_{ij}^{(k+1)\mod 6}} \right) - 1 \right) \right|$$ (5.3)
3. \[ C_{ij} = \prod_{k=0}^{5} (P_{ij}^{(5+k) \mod 6} \lor P_{ij}^{k} \lor P_{ij}^{(k+1) \mod 6} \lor A_{ij}^{k}) \]  

(5.4)

Where

- \( B_{ij} \) = the sum of number of zeros in the neighborhood, excluding the centre pixel.
- \( A_{ij} \) = a function, for which result depends on the number of ‘0 to 1’ or ‘1 to 0’ changes.
- \( C_{ij} \) = a function, for which result depends on three consecutive pixels and \( A_{ij} \) of the middle pixel.

An object pixel is then considered to be not a skeletal point and is removed if all three following conditions are satisfied:

4. \( 1 < B_{ij} < 6 \)  
   (5.5)

5. \( A_{ij} = 0 \)  
   (5.6)

6. \( C_{ij} = 1 \)  
   (5.7)

Since \( B_{ij} \) is the sum of number of zeros in the neighborhood, \( B_{ij} \) indicates whether the given pixel is isolated pixel or internal pixel or endpoint. The term \( A_{ij} \) indicates the junctions. If \( A_{ij} \) is 1, then centre pixel is a junction for other neighbors, and it should not be deleted. If \( A_{ij} \) is 0, it is not a junction, and it can be deleted. The condition A.4 is required in prevention of two-pixel lines vanishing. The algorithm produces 2-pixel wide skeletons. Single-pixel wide skeletons can be achieved afterwards with a single synchronous iteration.
5.4 Proposed Modified Algorithm for Hexagonal Processor Array and Hardware Implementation

The following modified algorithm (Figure 5.4) is used in order to obtain the skeleton image on hexagonal lattice where the notations are as explained in section 5.3.

\[
\text{always @ (posedge clk)} \begin{align*}
    \quad & \text{begin} \\
    & \quad \text{if (start)} \\
    & \quad \quad \quad \text{flag} = 0; \\
    & \quad \quad \else \\
    & \quad \quad \quad \text{begin} \\
    & \quad \quad \quad \text{evaluate functions 5.2, 5.3, 5.4} \\
    & \quad \quad \quad \text{if conditions 5.5, 5.6, 5.7 are satisfied} \\
    & \quad \quad \quad \quad \text{begin} \\
    & \quad \quad \quad \quad \quad \text{Pij\_out}=1; \\
    & \quad \quad \quad \quad \quad \text{Flag}=1; \\
    & \quad \quad \quad \quad \end{align*} \\
& \quad \quad \quad \text{end} \\
& \quad \quad \quad \text{else} \\
& \quad \quad \quad \text{Pij\_out} = \text{Pij\_in}; \\
& \quad \text{end} \\
& \text{end} \\
\]

Figure 5.4 Algorithm for Skeletonization

Working of the algorithm is described along with hardware implementation as follows. The inputs for the Synchronous Processing Element
(SPE) in Figure 5.5 are Pixel value \( (P_{ij}) \), Neighbors \( P_{ij} \) values \( (P_{ij}^1, \ldots, P_{ij}^6) \), and Neighbors \( A_{ij} \) values \( (A_{ij}^1, \ldots, A_{ij}^6) \). When the ‘START’ signal is given, the ‘FLAG’ value in the combinational unit, and output Flip Flops will reset. The combinational block evaluates the functions 5.2, 5.3, 5.4. If all the three conditions are satisfied, then the output \( P_{ij} \) value is changed to ‘1’ and ‘FLAG’ signal is changed to ‘1’. If any condition is not satisfied, then the combinational unit again evaluates the conditions 5.5, 5.6, 5.7 continuously, until the output \( P_{ij} \) value changed to ‘1’.

If at any stage of processing the pixel value \( P_{ij} \) has been changed to logic ‘1’ the corresponding pixel doesn’t belong to a skeleton and has been removed. Once the output \( P_{ij} \) value changes to ‘1’, the ‘FLAG’ value is set to ‘1’. Then the combinational block will be disabled and it won’t respond to any changes in the input.

![Figure 5.5 Architecture of Rectangular Processing Element](image)

In the proposed method, the system clock is used as the clock signal for the input and output latches which will function as control input. When PEs are connected in an array like structure, output of one PE is connected to other PEs. If one latch is used at the output of PE, this latch will act as the input latch for the neighbouring PEs. To compare the merits and demerits of skeletonization on hexagonal grid, rectangular grid operation is also implemented. The algorithm given by Zhang and Suen [126] has been chosen for the rectangular array implementation since it is based on iterative thinning algorithm.
5.4.1 Processing Element for Hexagonal Processor Array

The features of The Cyclone II 2C20 FPGAs were described in the Chapter 4. The proposed architecture comprises of a number of identical Processing Elements (PEs). Once processing elements are designed, it may be arranged in an array with each cell connected to its neighbors to form the desired lattice as illustrated in Figure 5.6.

Figure 5.6 (a) Rectangular Processor Array (b) Hexagonal Processor Array

By using the processing element explained above, a hexagonal processing array was formed (Figure 5.6(b)) and skeleton of the image was obtained. The proposed architecture of processing element is implemented in verilog using Modelsim software and processor array of size 256 x 256 is constructed as hexagonal array. Overall architecture is shown in Figure 5.7. The input and output memories of size 256 x 256 bits were created. The input data can be read from input memory, processed by the hexagonal processor array and the results
were stored in the output memory. The simulation results are discussed in the section 5.5.1.

Figure 5.7 Overall architecture (a) Rectangular processor array (b) Hexagonal processor array

5.5 Results and Discussions

5.5.1 Results of Hexagonal Processor Array

Some standard images were considered for simulation process. The implementation steps are 1) The input image was converted into text file using MATLAB 2) The input memory is initialized in Verilog using ‘$readmemb’ 3) The skeletonization operation was performed on the rectangular and hexagonal array structure using the procedure as explained in previous sections 4) Output image was saved in text file and it was displayed using MATLAB.

Table 5.1 Timing Analyzer Summary of Hexagonal Processor Array

<table>
<thead>
<tr>
<th>Type</th>
<th>Actual Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst-case tsu</td>
<td>8.305 ns</td>
</tr>
<tr>
<td>Worst-case tco</td>
<td>6.544 ns</td>
</tr>
<tr>
<td>Worst-case th</td>
<td>-3.236 ns</td>
</tr>
</tbody>
</table>
Table 5.1 shows timing analyzer summary from the implementation of the processing element of hexagonal processor array. From the results, the hardware utilization was found to be less (5%) and the actual time taken was in the order of nano seconds. Figure 5.8 below shows the simulation results of hexagonal processor array for the images shown during different iterations.

Figure 5.8 (a) Original Image (b) Skeleton Image after 40 ns (c) Skeleton Image after 120 ns

One of the main applications of skeletonization is in finger print recognition to remove the redundant information without changing the connectivity. This algorithm is giving satisfactory results while thinning the finger print images (Figure 5.9).

Figure 5.9(a) Original Image (Finger print image) (b) Skeleton Image after 120 ns
Once the image has been edge detected, it is often necessary to thin the edges. This serves to remove all the redundant points which are contained within the edge image whilst retaining its basic structure and characteristics and is used in pattern recognition (fewer the pattern points, better the recognition). The edge detected image (Figure 5.10(a)) was given as input and the skeleton images are shown in Figure 5.10 (b) after 120 ns.

Figure 5.10(a) Original image (Edge detected image) (b) Skeleton image after 120 ns

5.5.2 Results of Rectangular Processor Array

Table 5.2 Timing Analyzer Summary of Rectangular Processing Element

<table>
<thead>
<tr>
<th>Type</th>
<th>Actual Time for Processing Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst-case tsu</td>
<td>10.613 ns</td>
</tr>
<tr>
<td>Worst-case tco</td>
<td>7.437 ns</td>
</tr>
<tr>
<td>Worst-case th</td>
<td>-3.511 ns</td>
</tr>
</tbody>
</table>
From the timing summary and flow summary of processing elements of rectangular array, the following observations were made (i) time taken for rectangular processing element was more with that of hexagonal array (ii) the total logic elements utilized for rectangular processing element was more with that of hexagonal processing element (Table 5.3). It was observed that Hexagonal PE saves 54% hardware compared to rectangular PE with the modified algorithm.

<table>
<thead>
<tr>
<th>Hardware Requirements</th>
<th>Hexagonal PE</th>
<th>Rectangular PE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Elements</td>
<td>23 / 18,752</td>
<td>25 / 18,752</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25 / 18,752</td>
</tr>
<tr>
<td>Registers</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Pins</td>
<td>17/315</td>
<td>12 / 315</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 / 315</td>
</tr>
</tbody>
</table>

The skeleton image of rectangular processor array and hexagonal processor array are shown in Figure 5.11 for comparison. It was possible to get the skeleton using hexagonal processor array with some branches which is the limitation of the algorithm but the algorithm works well for reconstruction. But for characters, Skelton obtained using rectangular processor array yields better results (Figure 5.12).
5.6 Reconstruction

Reconstruction operation was performed on both rectangular and hexagonal arrays. Reverse operation of iterative thinning algorithm was applied. Algorithm Calculates the number of ones ($B_{ij}$) in the Neighborhood. If $B_{ij} > 1$, it indicates that the centre pixel is not isolated and it can be changed to object pixel to ‘0’ and background pixel to ‘1’ for rectangular sampled images and object pixel to ‘1’ and background pixel to ‘0’ for hexagonal sampled images. This algorithm is similar to dilation operation. Reconstructed images using rectangular processor array and hexagonal processor array are shown in Figure 5.13 and 5.14 respectively.
It was observed from the Figure 5.13 and 5.14, that the algorithm works well for the reconstruction of the image from the skeletons obtained on rectangular grid and hexagonal grid. From the results of Figure 5.15, it was observed that the reconstruction of the image using the skeleton obtained from the hexagonal processor array is closely related to the original image than the skeleton obtained using rectangular processor array.
Figure 5.15 (a) Skeleton image (b) Reconstructed image using hexagonal processor array (c) Reconstructed image using rectangular processor array (d) original image.

5.7 Comparison with the Existing work

Although the architecture proposed by N. Lopich and Dudek [59] is consuming less power, many resources of FPGA are wasted as input latches, output latches and control unit. Control unit (CU) is responsible for (Processing Element) PE self-timing. It will generate the necessary clock signals. When the input pixel $P_{ij}$ is changed to ‘1’ then CU stops generating clock signal so that combinational block (CB) is disabled. PE will not respond to any input changes until ‘START’ signal will be set to ‘1’. If the algorithm is changed as explained in the section 5.4, based on the system clock as the clock signal for the input and output latches, it is possible to reduce number of processing elements (Table 5.4). The time taken for the eroding time is 95 ns as per the existing architecture for 20 x 20 image [59]. But the eroding time was 120 ns to obtain single pixel skeleton for 256 x 256 image using proposed architecture (Figure 5.9(b) and 5.10(b)).

Table 5.4 Comparitive results between Existing Architecture by N. Lopich and Dudek [59] and the Proposed Architecture

<table>
<thead>
<tr>
<th></th>
<th>Actual PE</th>
<th>Proposed PE</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Logic Elements</td>
<td>36</td>
<td>19</td>
</tr>
<tr>
<td>No. of Pins</td>
<td>25</td>
<td>17</td>
</tr>
<tr>
<td>Maximum Frequency</td>
<td>227.12 MHz</td>
<td>380.08 MHz</td>
</tr>
</tbody>
</table>
Layer Eroding time | 120ns for 256 X 256 image | 96 ns for 20 X 20 image

The reconstruction of skeletonization on hexagonal grid is not addressed yet. Using the proposed architecture, the reconstruction was attempted and it was possible to obtain better results on hexagonal grid compared with reconstruction on rectangular grid.

5.8 Summary

The processing elements for hexagonal array and rectangular array were designed based on the Synchronous Cellular Array Processor. The performance was tested using Quartus II development software. Hexagonal processing element saves 54% hardware, when compared to rectangular processing element using the proposed architecture. Rectangular processor array needs one extra memory called ‘Intermediate Memory’ to save the intermediate results. Interconnections of rectangular processor array are more compared with Hexagonal Processor Array. Also, it was observed that (i) Hexagonal skeletonization process is faster, when compared to the rectangular skeletonization process (ii) Skeletons were found to be centered inside the underlying shape (iii) The algorithm works well for the standard and fingerprint images using hexagonal processor array. For characters, the rectangular processor array provides better results (Figure 5.12) (iv) In reconstruction process, hexagonal processing generates a more representative image (v) The proposed architecture is utilizing less Processing elements than the existing architecture proposed by N. Lopich and Dudek [59] and time taken for eroding was found to be less. (vi) Hardware implementation of Skeletonization algorithm on rectangular grid can be designed based on single-phase algorithm in order to validate the performance on hexagonal grid.
CONCLUSIONS AND SCOPE FOR THE FUTURE WORK

As Human Visual System (HVS) involves a set of parallel and quasi-independent channels which are sensitive to visual signal with some specific frequency and orientation, it can recognize different textures easily. With this motivation, the study and analysis of HVS and BCS was performed. Hexagonal sampling scheme was found to be the alternate tessellation to gather the information because of its geometry and nature. This alternative view of the visual world may present researchers with some advantages in representation and processing of the visual information. Furthermore, such a study may illuminate the importance of the role which the sensors play in computer vision.

Hex-Gabor kernel was generated by adding the kernels obtained in the three directions $0^0$, $60^0$, and $120^0$. The response of the Hex-Gabor was tested on (i) Mimicking hexagon using half-pixel shift approach (ii) True rectangular lattice. The applications of Hex-Gabor kernel include (i) Image enhancement (ii) Interpolation and (iii) Edge detection.

The properties and features of Hex-Gabor can be summarized as:

- Hex-Gabor is doing a nonlinear filtering operation using an isotropic kernel.
- It was found that at sigma = 2/π the kernel was isotropic and nonlinear. Filtering quality is found to be good at this sigma and reduces substantially for a slight increase/decrease in the value of sigma after 2/π.
- It was found that the effect of Hex-Gabor is pronounced in other pseudo-lattices, where suboptimal image quality is obtained for three directional operation for this sigma, as in regular hexagonal lattice. Apart from what is stated here, there could be other reasons for this.
- It would not be necessary to use methods like half pixel shift method to get similar results to that we obtained, since Hex-Gabor is considering the intrapixel distances and directions of adjacent pixels, in any regular lattice.
- Hex-Gabor filters in regular lattices are suitable for enhancement and interpolation. Much research is required to add the effect of orthogonal direction enhancement in pseudo lattices. It also shows that for an
interpolation by a factor of two, it provides a positive slope (magnitude at 0.5) as that required for smoothing images by heat diffusion equation. To show that the results are even better than Kramer’s method [99] which considers both signs, the processed images were given for a comparison in Chapter 3.

- Edge enhancement also creates conditions for a better edge detection. Some results of applying canny edge detection after a preprocessing by Hex-Gabor were discussed in Chapter 4. It has additional ability to show gray scale region boundaries as edges and able to bring out active contours in the image while performing edge detection.

- Inverse problems are often the attraction of research in filtering images. It may be said that our operation is reversible as this method is not iterative, but proving this is beyond the scope of this thesis. The results with the Hex-Gabor filter also support the results of Gabor [34] in respect of solution by heat equation using a negative sign for the $\Delta u$ where $u$ denotes the image.

In addition with the Hex-Gabor generation and applications our original work also include the FPGA implementation of CLAP algorithm based edge detection, thinning and reconstruction over a hexagonal lattice which are included in the thesis.

Finally, it is concluded that all the operations required by human visual system using Gabor filters in hexagonal domain is now much more established by the results in terms of its perfection to view minutest detail, and where sigma is also tuned for detailed and/or global vision. The nonlinear aspect of Hex-Gabor kernel has been uncovered in this work. A Hex-Gabor filter arrangement obtained for sigma > 2/pi for a more global filtering was shown. While this is viewed from top it shows regular hexagonal pattern as we had originally placed to obtain Hex-Gabor filter.

**6.1 Scope for the Future Work**

- Gabor filter models may also be specially developed for X-ray images considering the optical and interfering characteristics of objects.
➢ Image enhancement operation using Hex-Gabor is to be tested for high definition TV similar to image based kernel approach and selection of suitable sigma value is important for this application.

➢ Improvement in the performance of hardware based edge detection using CLAP algorithm can be achieved by smoothing operation and adaptive thresholding technique.

➢ FPGA implementation of edge detection was performed on 64 x 64 image due to limitation of hardware and this work has to be extended with the large image size using recent FPGAs.

➢ Hardware implementation of skeletonization algorithm on rectangular grid can be designed based on single-phase algorithm in order to validate the performance on hexagonal grid.

➢ Biological visual system can be modeled as a cascade of sub filters covering the major processing layers found in mammal visual systems—from photoreceptors over ganglion cells in the retina up to simple cells in the primary visual cortex [3]. The hardware implementation of this filter bank is very much useful for the real time bio-inspired visual encoding system. The proposed hardware based edge detection can be used for this purpose.

➢ In [127], a high performance FPGA-based cellular neural network which implements a Gabor-type filter was presented. It is possible to design a Gabor type filter using Hex-Gabor and it can be interfaced with the retina chip in order to develop entire silicon retina system for vision based applications.