Hardware Implementation of Edge detection on Hexagonal Sampled Image Grids

Veni. S
Assistant Professor
Department of Electronics & Communication Engineering
Amrita Vishwa Vidya Peetham
Coimbatore-641112, INDIA

K. A. Narayanankutty
Professor
Department of Electronics & Communication Engineering
Amrita Vishwa Vidya Peetham
Coimbatore-641112, INDIA

Mohammad Raffi
Student (M.Tech VLSI)
Department of Electronics & Communication Engineering
Amrita Vishwa Vidya Peetham
Coimbatore-641112, INDIA

ABSTRACT
Increasing Processing capabilities of graphic devices and recent improvements in CCD technology have made hexagonal sampling attractive for practical applications. Also, hexagonal representation has special computational features that are pertinent to the vision process. This paper describes Edge detection operation on hexagonally sampled images and its hardware implementation based on Cellular Logic Array Processing (CLAP) algorithm. This architecture builds up a virtual hexagonal grid system on the memory space of computer and processing algorithms can be implemented on such virtual spiral space, thereby decreasing the computational complexity. These operations were done on hexagonal sampled grid using MATLAB version 7 and the results were compared with rectangular sampled grid. MODELSIM and Quartus II software were used for analysis and synthesis. The performance was tested using Altera Cyclone II FPGA. It was observed from the results that there is a marginal improvement while processing with hexagonal sampled grid. Hardware utilization is found to be less for the image sampled on hexagonal grid compared with rectangular grid.

General Terms
Image processing, Edge detection, VLSI architecture.

Keywords
Hexagonal image processing, CLAP algorithm, FPGA implementation of CLAP algorithm.

1. INTRODUCTION
Classically, edge detection algorithms are implemented on software. With advances in the VLSI technology hardware implementation has become an attractive alternative. Since Golay [1], the possibility of using a hexagonal structure to represent digital images and graphics has been studied by many researchers. He has proved that hexagonal pattern processing is well suited to the studies of patterns which can be presented to the computer in any arbitrary orientation, such as blood cell photographs, bubble chamber photographs or Ariel photographs. Also, he proposed a parallel computer based on hexagonal modules which could be connected to perform different morphological operations and require fewer interconnections compared to a similar square based architecture. Mersereau [2] has shown that for circularly band limited signals, 13.4% fewer sampling points are required with the hexagonal grid to maintain equal high frequency image information with the rectangular grid. Thus it requires less storage and less computation time and coding efficiency can be increased if the hexagonal sampling scheme is used for image coding application. Kamgar-Parsi [3, 4] developed formal expressions for estimating quantization error in hexagonal spatial sampling and found that, for a given resolution capability of the sensor, hexagonal spatial sampling yields an average quantization error which is better than square sampling by 2%. Connectivity between pixels is an important concept which is used for the analysis of regions and boundaries. The hexagonal grid offers consistent connectivity (6 - way connectivity) which yields equidistant property i.e. the distance between neighbouring pixels are equal [5,6]. Neighbouring pixels in hexagonal grid have always one common edge which results in easier and most efficient algorithms such as thinning algorithm. Deutsch [7] developed thinning algorithms for various types of arrays as rectangular, triangular and hexagonal and compared them experimentally. He concluded that the hexagonal array offers a balance between the rectangular and the triangular arrays in that it requires almost the same storage, yields a midway average reduction value, and has the shortest processing time. Staunton [8-10] has designed specialised hardware to generate hexagonal images and worked on a variety of applications such as edge operators and thinning. He found that, for the industrial inspection application studied, the hexagonal grid processing system was able to outline defects more accurately, than the square processing system.

Vitulli.R et. al. [11] compared the Nyquist constraints i.e., the minimum sampling densities without aliasing, between rectangular and hexagonal. They have shown that using hexagonal grid, wider spectra can be sampled without aliasing with the same number of pixels, or less pixel than using square grid. Goodman [12] demonstrated that the Fourier transform (FT) of a hexagonal lattice is still a hexagonal lattice. Serra [13] has developed many hex-morphological operators that were currently used for image processing. According to him, the connectivity definition and the higher symmetry, has lead to simpler processing algorithms in the hexagonal grid. A novel spline family, especially designed for hexagonal lattices, has been proposed by Van De Ville et.al. [14] recently. Hex-splines were already successfully applied to some printing applications. Laurent Condat and Van De Ville [15] demonstrated the relevance of combining multidimensional splines with quasi-interpolation for reconstruction and resampling from data defined on 2-D lattices. They proposed practical solutions with

theoretically optimal FIR and IIR prefilters to be associated with box-splines and hex-splines. Eric Anterrieu et.al. [16] assessed the performances of apodization functions to be applied to complex visibilities distributed inside a star-shaped window over a hexagonally sampled grid. They have shown, in particular, how discrete Fourier calculations over hexagonal grids can be performed without the development of a specific algorithm, by simply reusing standard FFT algorithms designed for Cartesian grids. Also they developed an interpolation formula for resampling data from hexagonal grids without introducing any aliasing artifacts in the resampled data.

Most of the hexagonal processing algorithms require an addressing scheme to process data in a mathematical framework. Many image processing operations like Edge detection and morphological operations like skeletonization, thinning, dilation, erosion, opening and closing of the images can be done on a hexagonal lattice. Her [17] developed a symmetrical hexagonal coordinate frame, denoted as *R3, which uses three coordinates x, y, z, instead of two, to represent each pixel on the grid plane. Here the distance between two neighbouring grid points is defined as one unit. Middleton and Sivaswamy [18] proposed Spiral Addressing for hexagonal grid. The Spiral Architecture has some distinguishing features compared to the square image processing. The one dimensional addressing scheme leads to an efficient storage and the placement of the origin at the centre of the image which simplifies geometric transformations of a given image. Also, the hexagonally sampled image allows non-traditional neighborhoods with consistent boundary connectivity, which is useful for many computer vision applications. Sampling lattice is one aspect of the sensing methodology used in computer vision. Thus we can represent a hexagonal grid of pixels on the existing rectangular screens for modeling and processing purpose, which is more suitable for computer vision modeling.

Many resampling techniques were proposed like brick wall, quincunx sampling, least squares approximation of splines, suppressing alternate rows and columns from the square sampled image, etc., For the representation of hexagonal grid we used the virtual hexagonal grid system. Once the sampling lattice is digitized into hexagons, various image processing operations can be performed on these sub-sampled images. Wu and Hintz [19] constructed a virtual hexagonal structure which is an important milestone for the theoretical research and the practical application exploration of this architecture. It is based on Virtual Spiral Architecture and builds up a virtual hexagonal grid system on memory space on computer. Then, processing algorithms can be implemented on such virtual space. Finally, resulting data can be mapped back to rectangular architecture for display (Figure.1). This mimicking operation nearly does not introduce distortion or reduce image resolution, which is the most remarkable advantage over other mimicking methods, while keeping the isotropic property of the hexagonal architecture. We are creating virtual hexagonal grid system using sampling scheme by shifting alternate rows of rectangular lattice acquired , by half pixel as shown in Figure 2 & 3 (referred by Staunton [9]) so that almost all pixel values are preserved.

After resampling, edge detection was performed using a framework called Cellular Logic Array Processing (CLAP). CLAP algorithm was proposed by E.G.Rajan et.al. [20] where various **Basis Structures** are used to filter the values of interest from the scan window. They used the sub sampling scheme by suppressing alternate rows and columns of existing rectangular grid. This method introduces loss of information due to the suppressed pixels whereas the sub sampling scheme used in this work as illustrated in Figure 3 preserves almost all pixel values. The hardware implementation details will be discussed in section 3. Though a lot of advantages are associated with the hexagonal acquisition and display systems, such devices are not readily available. Due to the non-availability of hexagonal display systems, the image processing operations were simulated using the existing rectangular lattice display devices only.

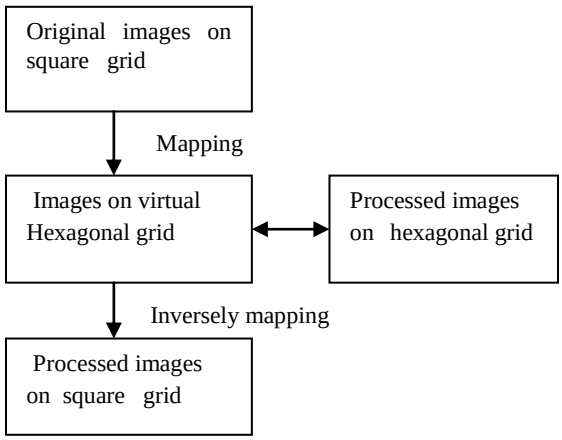
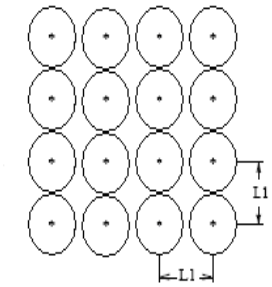


Figure.1 Image processing on virtual hexagonal grid system



$L1 = 1.0 ; L2 = 2.0 /(3.0)^{1/2}$

Figure.2 Pixels on a Square Grid

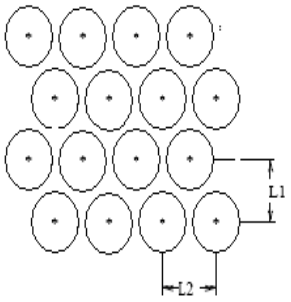


Figure.3 Pixels on a Hexagonal Grid

The paper is organized as follows. Section 2 describes CLAP algorithm based edge detection. A new architecture for CLAP algorithm based edge detection is proposed. Description about the modules used in the architecture are discussed in section 3. Verilog simulation results and FPGA implementation results of CLAP algorithm based edge detection are discussed in section 4. From the results obtained, it was observed that the sub sampled as well as the processed images over the hexagonal grid have more clarity and visual appeal than that over the rectangular grid.

2. EDGE DETECTION

Edge detection is an important operation in both biological recognition and computer vision. In biological vision, there is significant evidence that primary visual cortex serves to spatially arrange the visual stimuli into maps of oriented edges. In computer vision, edge detection is a pre-processing step in many applications such as object recognition, boundary extraction and segmentation [18]. The basic assumption used in most edge detection is that the edges are characterized by large changes in intensity. Hence, at the location of the edge, the first derivative of the intensity function should be a maximum or the second derivative should have a zero crossing [21]. Middleton and Jayanthi Sivaswamy [22] performed edge detection on hexagonal grid with three techniques using Prewitt, Laplacian of Gaussian (LoG) and the canny edge detector in HIP framework which is possible with the use of the spiral addressing scheme. It was proved that for noisy images, the performance of edge detection improved with hexagonal images and also the HIP framework for edge detection has computational advantages due to the use of the novel index addressing scheme. In this work, for hardware implementation, CLAP algorithm based edge detection is found to be suitable since the cellular sites can be updated in parallel.

For the hardware implementation, the edge detection algorithm can be performed by considering a group of pixels. In case of edge detection using sobel or prewitt operator, a 3 x 3 window with a group of 9 pixels can be selected. In moving window architecture discussed in [23], the latency is very high for the edge detection using rectangular lattice. A new architecture is proposed in this work with address generator, 'In system memory' or Block RAM and demultiplexer for both rectangular and hexagonal lattices. (In this paper, 'In system memory' and Block RAM are used interchangeably).

In CLAP algorithm, the given digital image is scanned by the 3 x 3 or 5 x 5 windows. On each move, the 3 x 3 or 5 x 5 sub image covered by the scan window is checked for the different basis structures as explained in [20]. The values in this polygon or basis structure in the sub image are examined to see whether the gray-distance, say ***D***, that is the difference between the maximum gray value ***GMax*** and the minimum gray-value ***GMin***, is less than or equal to a threshold value, say ***T***. If ***D*** is less than or equal to ***T***, then the central cell is assigned the gray-value 0; otherwise the original value contained in the central cell is left as it is. This procedure is continued till the entire image is scanned. The overall effect is that the boundaries of various regions in the given image, that appear to be uniform, are retained and their interior parts are erased thus giving us the edge. In order to have comparison between the rectangular sub sampled image (Figure 4) and hexagonal sub sampled image (Figure 5), edge detection was performed on both the lattices.

With reference to the Figure 4 & 5, The convex polygon connecting all the neighboring pixels of the central pixel is one of the basis structures. Other polygons can be obtained by removing the neighborhood pixels in various combinations as explained in [21]. To explain edge detection operation using CLAP algorithm, consider the rectangular lattice of Figure 4 with 3 x 3 window. The window consists of the pixels as circled where pixel 13 is a center pixel. The pixels 3, 11, 15 and 23 are forming a polygon and its basis structure is examined. A user defined threshold value is to be selected for edge detection. The gray distance D is computed by taking the difference between the maximum and minimum values of this polygon. This gray distance D is compared with the user defined threshold value. If $D \leq T$, then pixel value (13) = 0; Then the structuring element is moved to the next position till the image is completely scanned. Other such polygons of the selected window are pixel values (2,11,15,24), (2,4,22,24), (4,11,15,22), (2,15,22), (4,11,15) and so on. Like this, 16 convex polygons for rectangular grid and 18 convex polygons for hexagonal grid can be formed for each window of the image. Each polygon is scanned using CLAP algorithm and edge is detected.

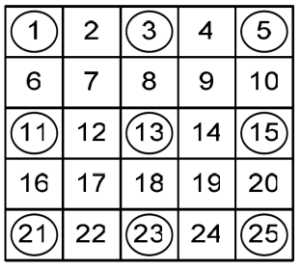


Figure. 4 Neighborhood of pixels in the rectangular sampled image

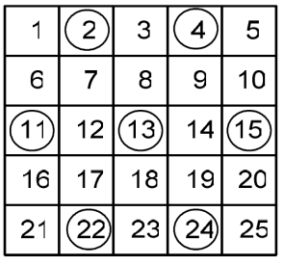


Figure. 5 Neighborhood of pixels in the hexagonal sampled image

In [24] the canny edge detection architecture has been developed using reconfigurable architecture and hardware modelled using a C-like hardware language called Handle-C. The proposed architecture was capable of producing one edge-pixel every clock cycle. The hardware model was implemented using the DK2 IDE tool on the RC1000 Xilinx Vertex FPGA based board. As this method of edge detection is based on convolution operation, it needs two longer FIFOs and multipliers. It will consume more memory and more time for multiplication. Hussmann and Thian [24] proposed FPGA implementation of a real-time sub-pixel edge detector. The proposed architecture in this work does not contains multipliers as it is based on CLAP algorithm and needs less registers compared with canny edge detection architecture. Hardware utilization is found to be less using the proposed architecture. Also, the proposed architecture is able to perform edge detection operation on both rectangular and hexagonal lattices. The results are discussed in section 4.

3. VLSI IMPLEMENTATION OF EDGE DETECTION

For the VLSI implementation Altera DE-1 board was used. Altera's low-cost Cyclone II FPGA family is based on a 1.2-V, 90-nm SRAM process with densities over 20K logic elements (LEs). It has features like embedded multipliers to support high-performance DSP applications and phase-locked loops (PLLs) for system clock management. The Cyclone II 2C20 FPGA that is included on the DE1 board provides dedicated memory resources called *M4K blocks* ('In system memory'). Each M4K block contains 4096 memory bits, which can be configured to implement memories of various sizes. Some aspect ratios supported by the M4K block are 4K x 1, 2K x 2, 1K x 4, and 512 x 8. 52 such M4K blocks are there in Altera Cyclone II 2C20 FPGA. We have used only 32 of it. We have divided these

32 M4K blocks into two parts for storing input image in one and output image in another.

Figure 6 shows the general block diagram of the proposed architecture for edge detection based on CLAP algorithm. It has the modules such as 'In system memory' for storing and retrieving the image data, Address generator to generate the address in order to create the virtual hexagonal grid structure, R_block - a set of registers (to store the data selected depending on the select line of the Demultiplexer) and S_block registers called window registers to create the hexagonal window.

Figure 7 shows the detailed block diagram of the architecture. It includes the comparators in order to compare the pixel values stored in the S_block to obtain maximum and minimum values in order to perform CLAP algorithm. The working of each block is explained in the next section.

3.1 VLSI architecture modules

3.1.1 Memory

Recent FPGAs are having 'In system memory' where data can be stored for processing. Initially the image pixel values are stored into the 'In system memory' and retrieved from it for processing. The image can also be stored in off chip memory or sent serially while processing using RS232 port. Since 'In system memory' access is faster, we are using it for data storage.

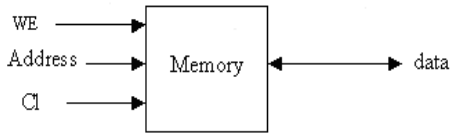


Figure. 8 Memory

The inputs to the memory (Figure 8) are Address Bus (ADDRESS), Data Bus (DATA) for data, Write enable pin (WE), clock pin (CLK). Whenever a clock is applied, at the positive edge of the clock, memory will give the data corresponding to the ADDRESS of the memory location. When the WE =1 then this block RAM will write the input data to the ADDRESS of the corresponding memory location.

For example, consider an 8 x 8 image on rectangular lattice (Figure 9).

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
.	.	.	.				
.							
.							
56	57	.	.	.	.	.	63

Figure. 9 Pixel representation in a rectangular lattice

As shown in Figure 9, initially, the image is stored into the 'In system memory' of FPGA. The address generator generates the address of each pixel. In the proposed architecture, the data can be read such a way that the read pixels are forming hexagonal window. After the reset, the address generated in the address generator will be sent to the system memory for every clock cycle.

Figure 10 shows the implementation of address pointer for hexagonal window. Two address pointers called Right edge address generation and Left edge address generation are used where each pointer is incremented by one for every clock cycle.

3.1.2 Address Generation and Edge Detection

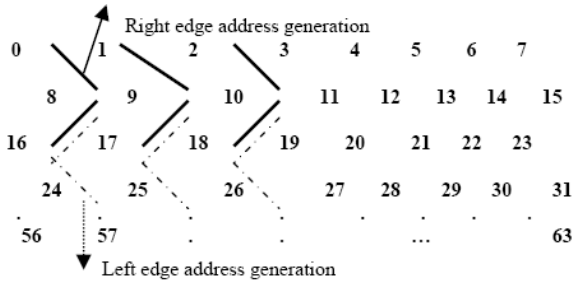


Figure. 10 Address pointer for hexagonal window

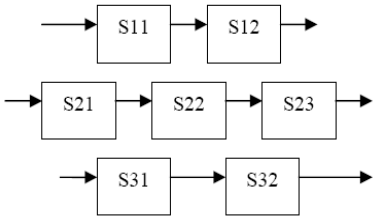


Figure. 11 S_block : A group of registers

S_block is a group of registers (Figure 11) which is forming a hexagonal window with 6 registers S11, S12, S21, S22, S23, S31 and S32 to read the data. The window movement for every three clock pulses for an image of size 8 x 8 is explained as follows. The window is initialized with zeros as shown in Figure.12.

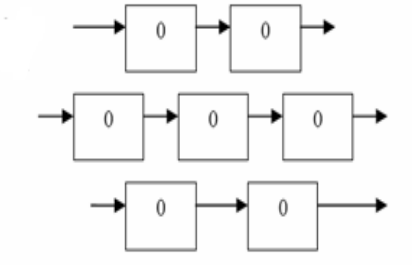


Figure. 12 Operation of S_block registers

After the first three clock cycle (Figure 13) the first pixel of each row is stored in the S11, S21 and S31 registers of Figure 11 (The pixel values 0, 8 and 16 respectively with reference to the Figure 10) using right edge generation.

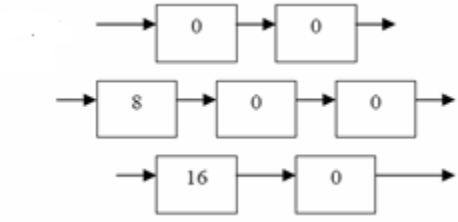


Figure. 13 Status of S_block registers after first three clock cycle

Edge is computed for the center pixel value of register S22 using comparator blocks C1, C2, C3, C4, and C5 (Figure 7).The pixel values in the S12, S21, S32 registers are given to C1 comparator. Similarly, corresponding pixel values are given to the comparators C2 to C5 as mentioned in the Figure 7 for various polygons in the hexagonal window. In each comparator, we are finding maximum and minimum pixel values and the difference (D) between them. Difference (D) is Compared with the threshold value (T) to find out an edge. The result is '1' when D is greater than T, else '0'.The result is again stored into the 'In system memory' for further processing. User defined threshold value is used for comparison with gray distance. Then window is moved to the right and after next three clock cycle, the status of the S-block registers is shown in Figure 14.

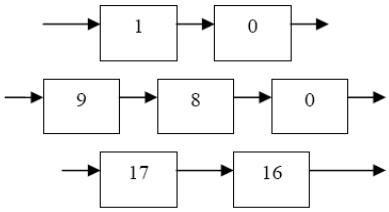


Figure. 14 Status of S_block registers after second three clock cycle

i.e. The data in the S11, S21, S31 registers are shifted to S12, S22, S32 registers respectively and the next pixel values (1,9,17 in the above example) are stored into the S11,S21,S31 registers. The process is continued till the last column of the first row. After the completion of the first row the address pointer is incremented using left edge address generation. The even number row values and odd number row values will be stored using left edge address generation and right edge address generation respectively. Likewise, the entire image is scanned and the edge is computed for every three clock cycles.

The computed values are stored in the 'In system memory'. Using the same procedure, edge is computed for rectangular lattice which is shown in Figure 9. For this, data can be read directly from the 'In system memory' with out using address pointers and then CLAP algorithm is executed. The results are compared with the edge detection on hexagonal window which will be discussed in section 4.

3.1.3 Demultiplexer and selection counter:

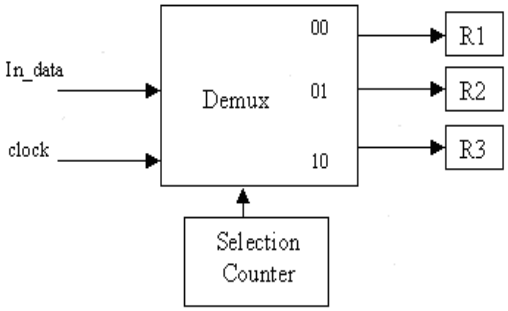


Figure. 15 Demultiplexer and selection counter

With reference to the Figure15, for each clock cycle the input data is fed to corresponding register through 1:3 Demultiplexer and the selection counter. For every three clock pulses the data in the registers R1, R2 and R3 is moved to the hexagonal window at a time as explained in the previous section.

4. RESULTS AND DISCUSSION

4.1 Verilog Simulation Results

Figures 16-18 shows the verilog simulation results of the designed architecture for the edge detection on rectangular lattice and hexagonal lattice. From the results, we were able to obtain thinner images using hexagonal sampling than rectangular sampling. It has been observed that there is a marginal improvement while processing with hexagonal sampling. Since there is no dedicated hardware available for hexagonal-based image capture and display, conversion has to be done from square to hexagonal image before hexagonal-based image processing. The difference will be clear only if we have hexagonal based image capture and display systems.

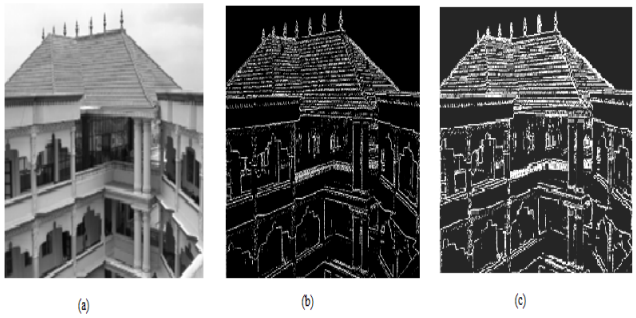


Figure. 16 (a) Original image (b) Edge detected image using rectangular structure (c) Edge detected image using hexagonal structure

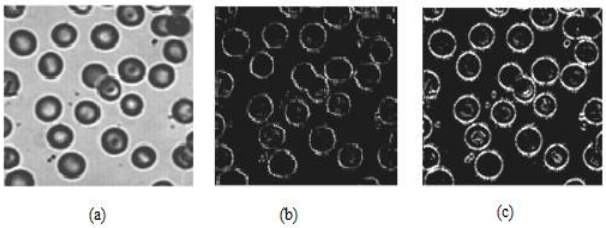


Figure. 17 (a) Original image (b) Edge detected image using rectangular Structure (c) Edge detected image using hexagonal Structure

Figure. 18 (a) Original image (b) Edge detected image using rectangular Structure (c) Edge detected image using hexagonal Structure

4.2 FPGA Implementation Results

Due to memory constraints, while implementing into FPGA (cyclone II 2C20F484C7), the 64 x 64 image was downloaded into 'In system memory', processed it and stored the result again in the same memory. Table1 shows the timing summary for this architecture for a 64 x 64 image. The maximum clock frequency was found to be 109.63 MHz. Table 2 shows the flow summary. Hardware utilization of FPGA implementation for edge detection of hexagonal sampled image of size 64 x 64 is summarized in the Table 3. Total logic elements used was 943/18,752 (only 5 %).

Table. 1 Timing Summary

Type	Slack	Required time	Actual time
Worst-case tsu	N/A	None	5.379 ns
Worst-case tpd	N/A	None	2.810 ns
Worst-case th	N/A	None	3.701 ns
Clock setup: in_clk	N/A	None	109.63 MHz (period= 9.122 ns)

Table. 2 Flow summary of the proposed architecture using hexagonal structure

Flow status	successful
Quartus II version	6.0 Build 178 SJ Web edition
Revision Name	edge_hex
Top-level Entity Name	edge_hex
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	No
Total logic elements	943/18.752(5%)
Total Registers	455
Total pins	6/315(2%)
Total virtual pins	0
Total memory bits	131,072/239,616(55%)
Embedded Multiplier 9-bit element	0/52 (0%)
Total PLLs	0/4 (0%)

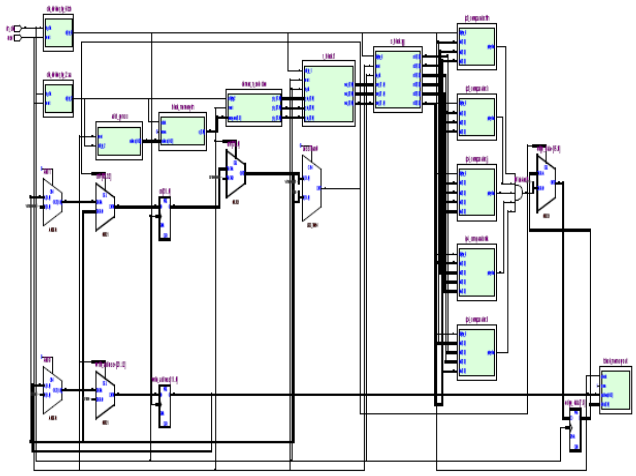


Figure. 19 Syntheis Report of the designed Architecture

Table. 3 Hardware Utilisation for edge detection of 64 x 64 image

Hardware	Usage
Logic elements	942/18,752 (5%)
Registers	455
Pins	6/315 (2%)
Memory bits	131,072/239616 (55%)
M4k Blocks	32/52

Figure 20 shows FPGA results using ALTERA DE-1 board. Images of size 64 x 64 were created and performed edge detection operation on both rectangular lattice and hexagonal lattice using CLAP algorithm. Edge has been detected successfully . Hexagonal sampled images have thinner edges than the rectangular sampled images. Due to memory constraints of FPGA, size of the image is restricted to 64 x 64 and the image size can be extended with the advanced FPGA. In [24], the edge detection algorithm was modeled in Handel-C on a DK2 environment. The design was implemented on RC1000-PP Xilinx Vertex-E FPGA based platform. On a PC with PentiumIII 1.3 GHz, time taken for canny edge detection of a 256 x 256 size grey scale image was 47 ms. We were able to get less computational time in the order of nano seconds and results are shown in the Table1.

5. CONCLUSIONS

An architecture for CLAP algorithm based edge detection has been proposed which can perform edge detection operations on both rectangular and hexagonal lattices. The addressing scheme using half pixel shift method does not introduce distortions. We were able to obtain edge detected images on both rectangular and hexagonal sampled grids for all standard images. Based on the verilog simulation results, we can conclude that hardware utilization of the architecture is less which may be seen from Table 3. The limitations of our architecture are (i) memory constraints of the present FPGA used and (ii) there is no dedicated hardware for capturing images and displaying it with hexagonal lattices so that we may able to visualize the difference between the edge detection operation on rectangular and hexagonal grid.

6. REFERENCES

[1] M.Golay, "Hexagonal Parallel Pattern Transformation," *IEEE Transactions on Computers*, vol.18, No.8, pp.733-740, 1969.

[2] Russelle M. Mersereau, "The Processing of Hexagonally Sampled Two-Dimensional Signals," *Proceedings of the IEEE*, vol .67, No.6, pp.930-949, 1979.

[3] Kamgar-Parsi, B. and W.A. Sander, III, "Quantization error in spatial sampling: comparison between square and hexagonal pixels," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Proceedings CVPR '89*, pp. 604-611, 1989.

[4] Kamgar-Parsi, B., "Quantization error in hexagonal sensory configurations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.14, No. 6, pp. 665-671,1992.

[5] Mylopoulos, J.P. and T. Pavlidis, "On the topological properties of quantized spaces, I. the notion of dimension," *Journal of the ACM (JACM)*, vol.18, No.2, pp. 239-246, April 1971.

[6] Mylopoulos, J.P. and T. Pavlidis, " On the topological properties of quantized spaces, II. connectivity and order of connectivity," Journal of the ACM (JACM), vol.18, No.2, pp.247- 254, 1971.

[7] Deutsch, E.S., "Thinning algorithms on rectangular, hexagonal, and triangular arrays," *Communications of the ACM*, vol.15, No.9, pp. 827-837,1972.

[8] Staunton, R.C., "An analysis of hexagonal thinning algorithms and skeletal shape representation," *Pattern Recognition*, vol.29, No.7, pp. 1131-1146, 1996.

[9] Staunton, R.C., "A one pass parallel hexagonal thinning algorithm in Image Processing and Its Applications," *Seventh International Conference (Conf. Publ. No. 465)*, pp.841-845, 1999.

[10] Staunton, Richard C. and Storey Neil, " A comparison between square and hexagonal sampling methods for pipeline image processing," Proc. SPIE , Vol. 1194, pp. 142-151,1989.

[11] R. Vitulli, "Aliasing Effects Mitigation by Optimised Sampling Grids and Impact on Image Acquisition Chains," *Geoscience and Remote Sensing Symposium(IGARSS '02*), pp. 979-981, 2002.

[12] Goodman, J.W., Introduction to Fourier optics, San Fran.: Mcgraw-Hill, 1968.

[13] J.Serra, Introduction to Mathematical Morphology, *Computer Vision, Graphics and Image Processing*, pp. 283-305, 1986.

[14] D. Van De Ville, T. Blu, M. Unser, W. Philips, I. Lemahieu, and R. Van De Walle, "Hex-spline: a novel family for hexagonal lattices," *IEEE Transactions on. Image Processing*, vol. 13, No. 6, pp.758–772, June 2004.

[15] Laurent Condat and Van De Ville, "Quasi-Interpolating Spline Models for Hexagonally-Sampled Data," *IEEE Transactions on Image Processing*, vol. 16, No. 5, pp.1195-1206, May 2007.

[16] Eric Anterrieu, Philippe Waldteufel and André Lannes ," Apodization Functions for 2-D Hexagonally Sampled Synthetic Aperture Imaging Radiometers ," *IEEE Transactions on Geoscience And Remote Sensing*, vol. 40, No.12, pp. 2531-2542, December 2002.

[17] I. Her, "Geometric Transformations on the Hexagonal Grid," *IEEE Transactions on Image Processing*, vol.4, No.9, pp. 1213-1222, 1995.

[18] Lee Middleton and Jayanthi Sivaswamy, "Hexagonal Image Processing – A Practical Approach", *Springer-Verlag London Limited*, 2005.

[19] Q.Wu, X. He and T. Hintz, "Virtual Spiral Architecture," *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, pp. 399-405, January 2004 .

[20] E.G.Rajan, T. Sanjay, and K. Pramod Sankar," Hexagonal Pixel Grid Modeling and Processing of Digital Images using CLAP Algorithm," *International Conference on Systemics, Cybernetics and Informatics, 2004.*

[21] Richard E. Woods, Rafael C. Gonzalez. Digital Image Processing, 2nd Edition, Addison-Wesley, November 2001.

[22] Middleton, L. and J. Sivaswamy, "Edge detection in a hexagonal-image processing framework ," *Image and Vision Computing*, vol.19, No.14, pp.1071-1081, June 2001.

[23] Muthkumar Venkatesan and Daggu Venkateshwar Rao," Hardware Acceleration of Edge Detection Algorithm on FPGAs. www.celoxica.com/cup/papers

[24] Stephan Hussmann nd Thian H.Ho, "A high-speed subpixel edge detector implementation inside a FPGA," *Real Time Imaging*, vol.9, pp.361-368, 2003.

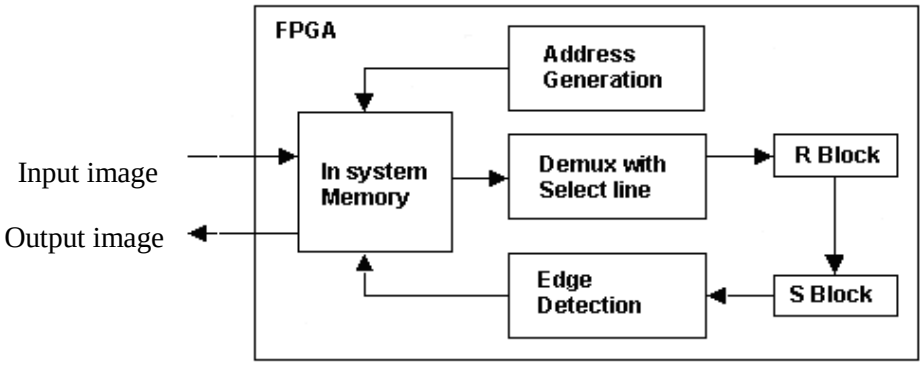


Figure. 6 General block diagram of the Architecture

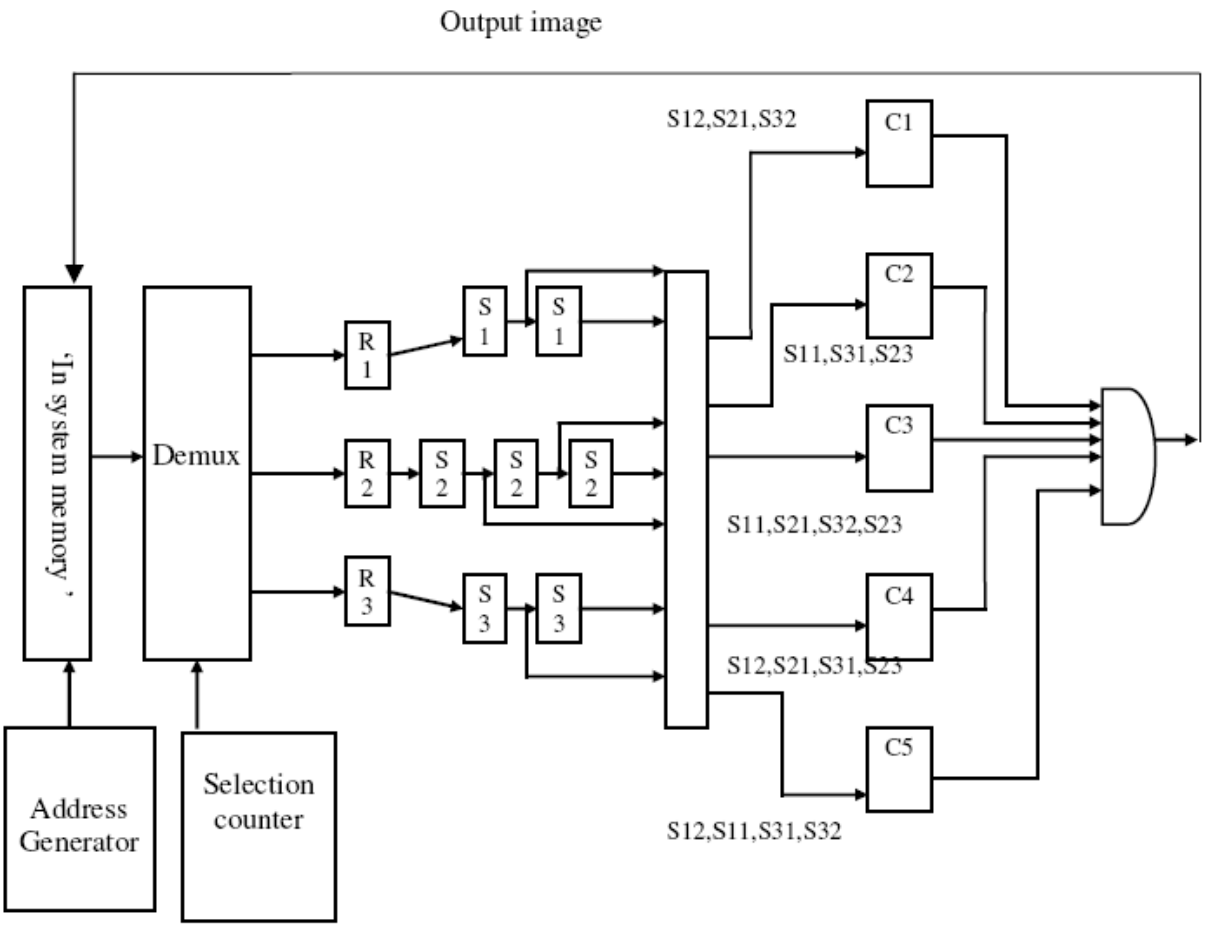


Figure. 7 VLSI architecture for the edge detection of hexagonal sampled image

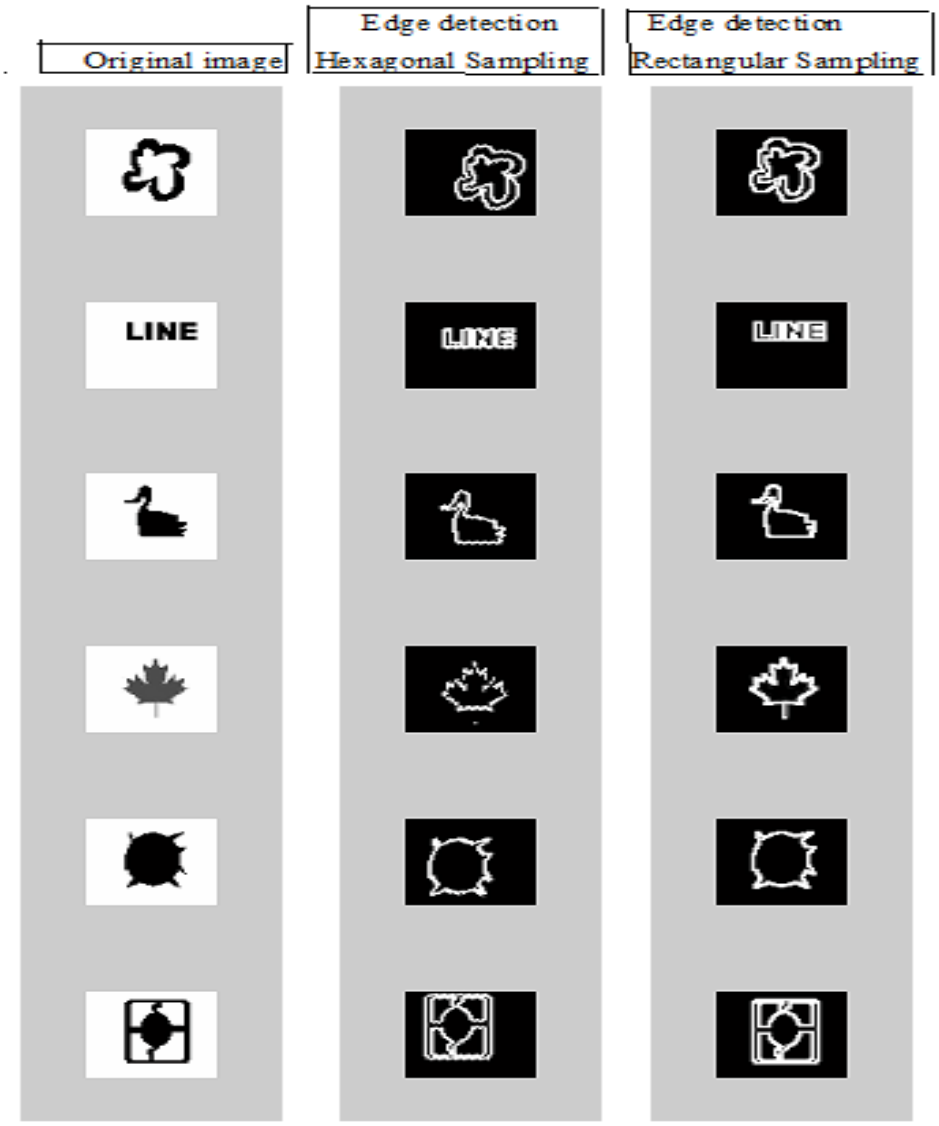


Figure. 20 FPGA results using ALTERA DE-1 board