An Early Visual Processing VLSI Employing Directional-Edge-Flag Cache Memory

A dissertation submitted to the Department of Frontier Informatics,

Graduate School of Frontier Science of

The University of Tokyo

In partial fulfillment of the requirements for the degree of

Master of Science

Övgü Öztürk

(47-46347)

Thesis Supervisor

Professor Tadashi Shibata

August 2006, Tokyo

Abstract

There are many on going researches to develop human-like intelligent recognition systems. Most of these efforts focus on software algorithms and system development. However hardware systems are also very important and necessary to achieve real time robust recognition and produce new devices to the service of humans. To achieve real-time human-like recognition, biologically inspired and hardware friendly algorithms have been developed in our laboratory.

According to the researches about human brain, it has been discovered that human perception relies on the directional edge information of the objects. Inspiring from this idea, in our recognition system, edge information is used to create feature vectors to represent the objects distinctively and to perform matching for recognition. In this system, edges from input images are extracted in different directions and obtained edge flag bits are projected in appropriate directions to create the corresponding feature vector. In this thesis, a new architecture employing directional-edge-flag cache memory has been designed for feature vector generation. Directional-edge-flag information of the entire input scene is stored in a cache memory and made accessible anytime on demand. Processing circuitries are employed together with the cache memory for high-speed feature vector generation. This architecture provides us to access the edge-flag bits from arbitrarily selected regions of the input scene on demand. Furthermore any kind of projection is made possible to apply to any size of recognition windows efficiently to generate feature vectors for different applications. With this new architecture it is possible to perform real-time flexible feature vector generation from large areas of visual scenes and to further enhance the performance of the conventional recognition system.

Acknowledgements

With my deep gratitude, I would like to express my great appreciation to my supervisor, Professor Tadashi Shibata and Professor Yoshio Mita for their continuous support, invaluable guidance and encouragement during all those years throughout my study.

I would like to give my special thanks to my colleagues, Mr. Kiyoto Ito, Mr. Hideo Yamasaki for their instructive advices, continuous support all the time. I have learned many things from them as a researcher and an engineer. I am also grateful to all my colleagues in our laboratory, especially Mr.Yasufumi Suzuki, Mr.Hitoshi Hayakawa and all other members for their comments, advices and friendship. I would like to thank Ms.Mori, Ms.Inagaki, Mr. Murai and Mr.Hagita for their precious support; they were there whenever I needed.

I would like to give my special thanks to the government of Japan for granting the scholarship and giving me a chance to study at the University of Tokyo. The VLSI chips in this study were fabricated in the chip design fabrication program of VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Hitachi Ltd. and Dai Nippon Printing Corporation, Rohm Corporation and Toppan Printing Corporation.

Also, I would like to thank many friends in Japan for their help and precious friendship, which made my life enjoyable. I have spent very memorable times with Ms. Aoki, Ms. Nagashima, Mr. Saito, Mr. Sasaki always receiving warm support, continuous care and advices.

Finally, I would like to express my deepest gratitude and appreciation by heart to my beloved family, my parents, sister and to my special friends for their everlasting support and endless love. Everything is possible with them.

Övgü Öztürk
The University of Tokyo

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Following the explosion in the development of new technologies in the last decade, many new research areas have appeared to interest various researchers. With the fascinating progress of microprocessor technologies it became possible to produce many electronic devices for different purposes, which was just a dream half century ago. Small size, high-performance processors yielded in compact high-speed computers making very difficult algorithms possible to be calculated in milliseconds. Software and hardware development triggering each other resulted in the development of real-time responding systems with new applications to improve human life. On the other hand, human brain is still the fastest processor with magnificent capability of operation and it is still full of secrets attracting many people.

Today, human-like intelligent systems and recognition is one of the most popular areas. Many studies are being done to solve the working process of biological systems, especially human brain. Producing recognition systems which mimic human brain is the first step of developing intelligent systems. There are many on going researches about object recognition and many systems developed to realize object recognition by means of software or hardware. These systems are used in a wide area of applications. Security cameras, which are every where now in our daily life, employ object recognition system to detect certain objects. Medical imaging is one of the areas, which is bringing very practical solutions to health sector with the help of new recognition systems. Human-robot systems, motion tracking, intelligent user-computer interfaces, etc.. are just a few examples utilizing object recognition technology.

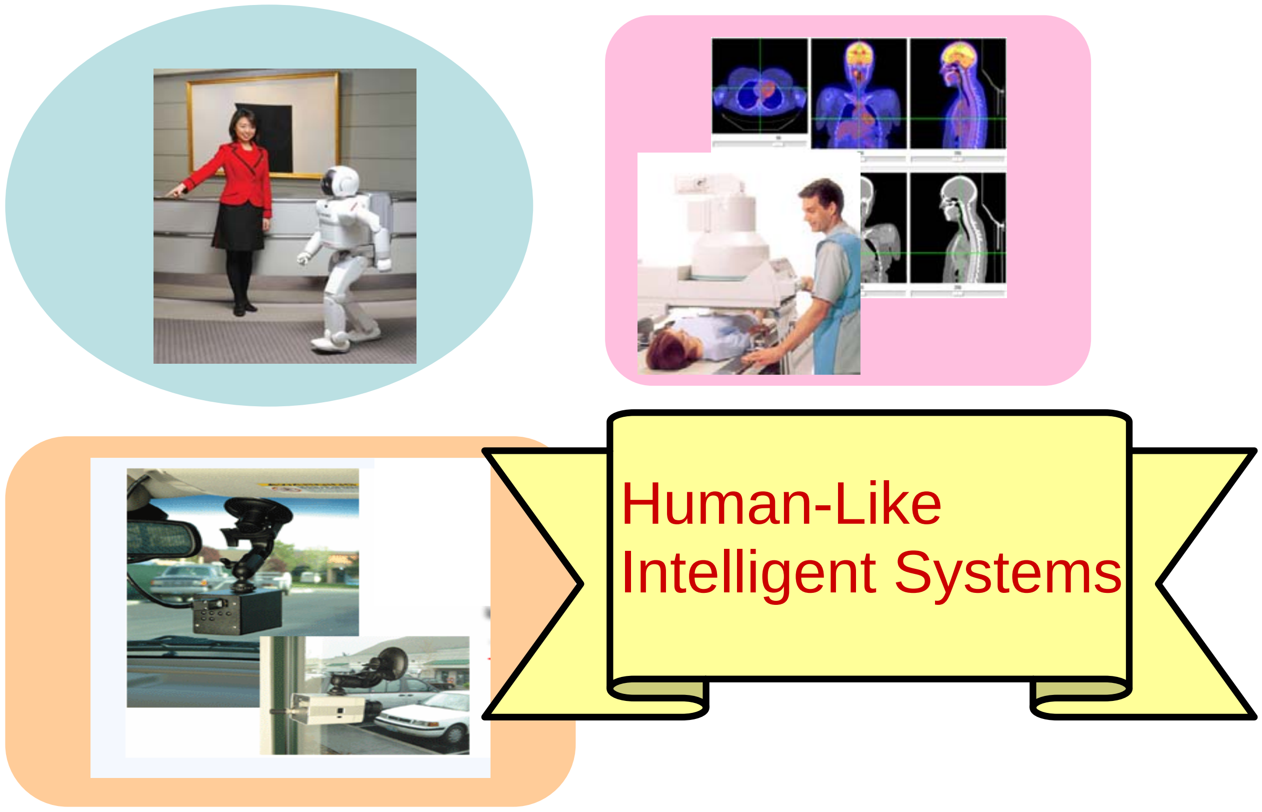


Figure 1-1. Application of Object Recognition Systems

However existing object recognition systems are still not enough to realize the dreams in researcher's mind to create the new era, where there are intelligent machines serving for humans. Extremely powerful computational power of computers provided some solutions to process many software algorithms, to do many difficult processes. However, it is still just at the beginning to bring intelligence, human-like judgment and recognition to the systems. There are a few hardware systems, being studied separately from software point of view because of the difficulty to apply such complicated software algorithms to hardware systems. These systems are still far away from expectations. At this point, it is important to associate the software and hardware approaches to achieve real-time responding intelligent systems capable of human-like recognition and judgment.

In our laboratory, inspiring from the working principle of human-brain, we have developed hardware friendly algorithms[4] and employed them to VLSI architectures to realize compact, high-speed human-like object recognition systems[2,4,9]. In this work, a new VLSI architecture is introduced to improve the performance of the conventional system and to provide the infrastructure for further development of the recognition system. This chapter is organized as follows. In Section 1.2, the general flow of the object recognition system that has been developed in our laboratory is explained.

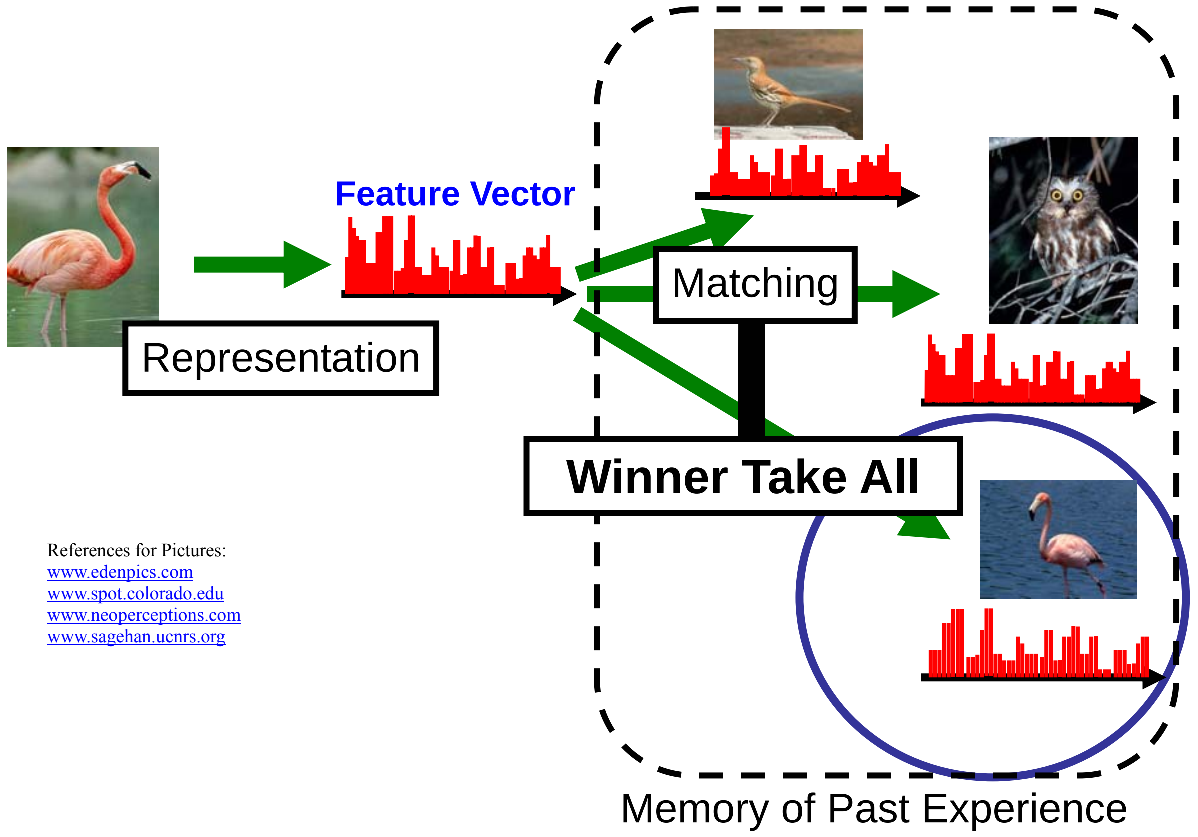
Then Section 1.3 introduces the software algorithms employed in our system. In Section 1.4 conventional hardware architectures are described in detail. And the problem of the conventional system and purpose of this work is explained in Section 1.5. The organization of the dissertation is outlined in Section 1.6.

1.2 About Human-like Image Recognition System

Human brain is so complicated that, governing principles are not still clearly understood by humans. It is well known that humans use only a few percentage of their brain's real capability. Yet, this small percentage is enough to process all the huge information encountered in every day life and to do billions of operation in each second, to make judgments and to give instantaneous commands. It is still a dream to build such a magnificent machine like human brain, however it is somehow possible to mimic the specific characteristics and to create processing systems close to human brain. One of the major goals in this aspect is to develop real-time intelligent systems, which are capable of human-like recognition and judgment. In our laboratory, we have been studying such a system to achieve high-speed robust image recognition by means of both software and hardware.

We are encountering many kind of huge information on each day. All those information is inputted into the system as images or video sequences and stored in memory to be used later. Image recognition is the process of detecting the known objects (previously stored in the memory) in a scene, which objects exist and where they exist in the scene.

To develop a real-time image recognition system, first of all two major points are very important. First one is the representation of the input information(image) in the system, second one is the matching of the new information(image) with the existing information in memory. As shown in Fig.1-2 our system is composed of two major steps; one is the image representation step including the processors to reduce the input information to much more compact form, the other is the matching step including the memory and processors to keep the previous information and to compare existing information with the new input.



References for Pictures:
www.edenpics.com
www.spot.colorado.edu
www.neoperceptions.com
www.sagehan.ucnrs.org

Figure 1-2. Object representation and recognition system

Input information gathered from outside world is too much. Input images contain lots of information, some of which is very necessary, some of which is redundant. Most of the time noise exists in the input because of the pre-processes during recording. In this case, first of all we have to select the most significant information from input and reduce the amount of data to make it possible to be processed in real-time. It is important to find the specific features of the input image, change these into a compact form to be represented in the system.

As a result of biologically studies, it has been proved that human brain focuses on edges of the surrounding objects and uses these edges during perception and recognition. Edges represent the major outlines and characteristic curves of an object. This information is very useful to describe the object and distinguish it in a given database. Inspiring from this idea, we extract edge information from input and utilize it to represent the input image in the system. This is the most critical part and forms the basis of our system. Extracted edges contain the most characteristic parts of the objects, which almost remain same under changing circumstances. Besides, these extracted edges are free of most of the noise coming from recording process. Fig. 1-3 shows a sample input image and edges extracted from it.

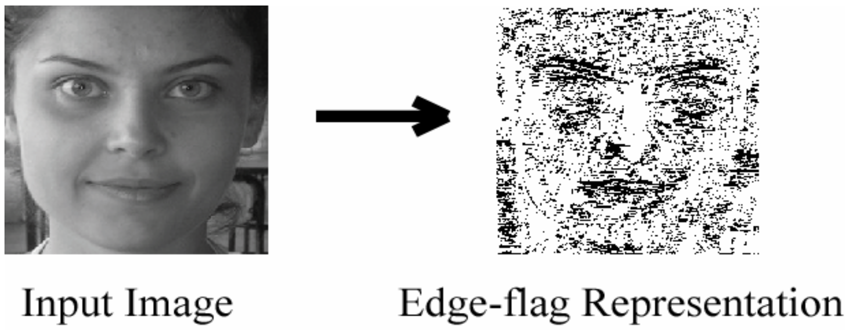


Figure 1-3. Edge-flag based object representation

The term edge describes the strength of the intensity difference of an image in one direction at a specific point. In other words, edge gives the characteristic of an image point in a particular direction. If there is a strong intensity difference in one direction, it means there exists a curve-like characteristic at that point having the normal in that direction. Edge information can be influenced a little by circumstances, however never disappears and remains there representing the characteristic of that point for that image. To achieve reliable robust recognition, it is not enough to study the edges only in one direction, it is also necessary to examine the image characteristic in different directions. In our system, images are studied in four main directions: horizontal, vertical, +45 degrees, -45 degrees. According to the results of many tests and simulations, these four directions encapsulate the major characteristics of the image and are enough to investigate the whole image to extract the edges. Edges of an input image are extracted by convolving the input image with corresponding appropriate kernels for each direction. In Fig. 1-4, edge-flag maps in four directions of an input image are demonstrated.

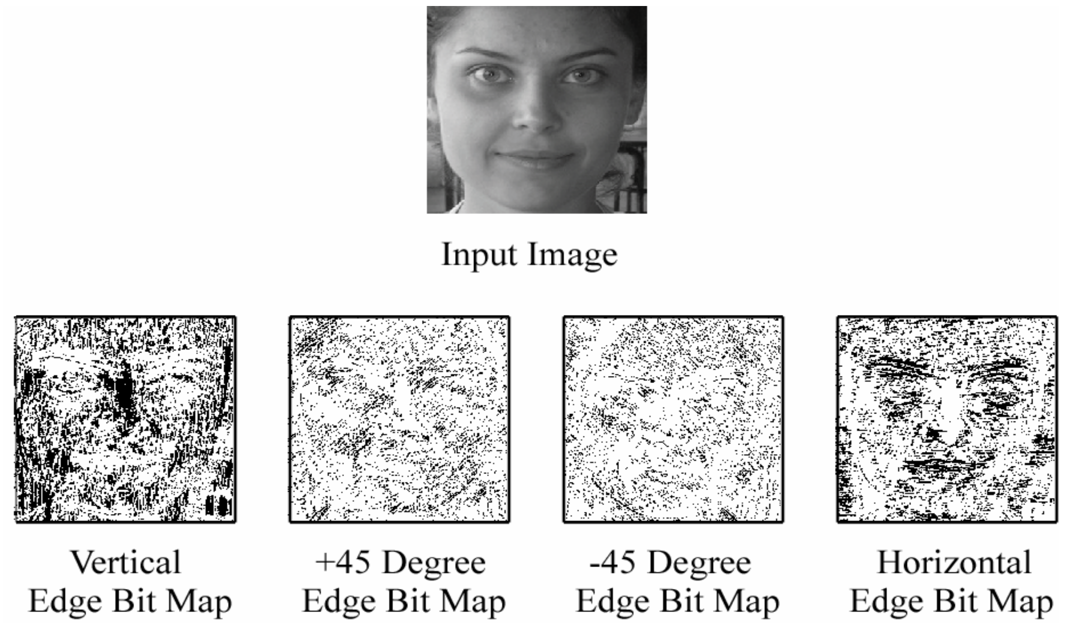


Figure 1-4. Directional-edge-flag maps of a sample input image

However, edges extracted from a given image are not so effective when it is used without any further process. Information extracted from an input image is still too much to be used to classify that type of object during recognition. Besides there are additional errors (false edge information or missing edge information) coming from the edge extraction process. To achieve a reliable real-time recognition, more compact information representing the characteristics of an image is needed. At this point a well-known approach to use the edge information as efficiently as possible is to apply projection method. Projection method is a widely used method in many areas including image recognition. After extracting edges form a two dimensional image, a two dimensional edge map, which is composed of existing edges, is obtained. From this two dimensional edge map, one dimensional characteristic vector, called feature vector, is constructed by projecting this information to an appropriate axis as shown in Fig. 1-5. Here projection means dividing the edge map into equally sized groups and adding the edge-flag bit in each group to form one vector element.

In our system, 64x64-pixel window of input images are studied during learning step. For each object, edges are extracted from 64x64-pixel image and one dimensional 64-element feature vector is constructed. Then this feature vector is stored in a memory representing that object in the system. For different types of objects, unique feature vectors are stored in a memory. When a scene is given, different regions of input scene with window size 64x64-pixel, called recognition window, are processed to create feature vector for each region. Feature vector obtained from a 64x64-pixel recognition window is compared with the vectors in the memory. If it matches one of the vectors in the memory, then the region is decided as containing the object which is represented by the matched feature vector in the memory.

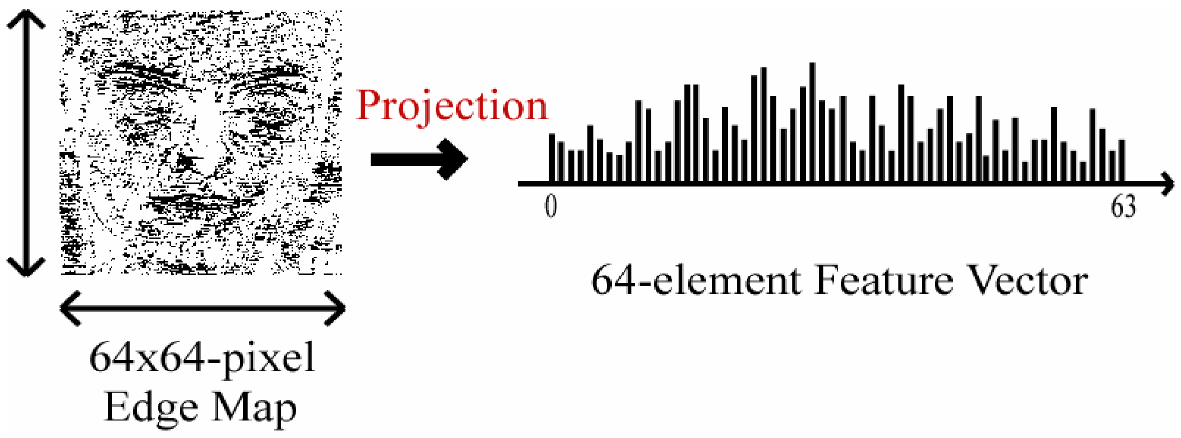


Figure 1-5. Feature Vector Generation from Edge Map

As explained above, the key point of recognition in this system is feature vector. It is the critical element that decides the final result whether object exists or not. Due to the several experiments done by other students in the same laboratory, there may be different ways of projection and results change depending on these. In next section three different methods of projection will be explained in detail.

1.3 Review of the Software Algorithms

As mentioned before, it is very important to represent the huge information obtained from images in a compact and distinctive way for each object. Depending on the application, sometimes one method of projection works very well, e.g. for face detection, CED-will be explained later[18]- works very well, where others don't give the same accuracy. Besides it is very time consuming process to generate feature vectors. Therefore, generation of feature vectors form the critical step for real-time responding systems. Below, three important projection methods developed in our laboratory will be explained in detail.

1.3.1 Projected-Principle Edge Distribution (PPED)

Fig. 1-6 illustrates a typical procedure generating an edge-based feature vector from a 64x64 pixel region [4]. In this procedure, first of all, edge flag bits of an input image of 64x64-pixel are extracted in four directions and edge bit maps are obtained. Next, each edge bit map is projected to an appropriate axis to construct 16 elements, totally 64 elements. For each direction, projection is accomplished by dividing the edge bit map into 16 equal groups and adding edge-flag bits for each group to form one vector element. For example, for vertical edges, edge flag bits in every four columns are accumulated to obtain one vector element. Similar procedures apply to other edge bit maps. Finally, 64 elements coming from four different directions form the 64-element feature vector of the image. Feature vector is used to represent the image and stored in the memory to be used for matching during recognition later.

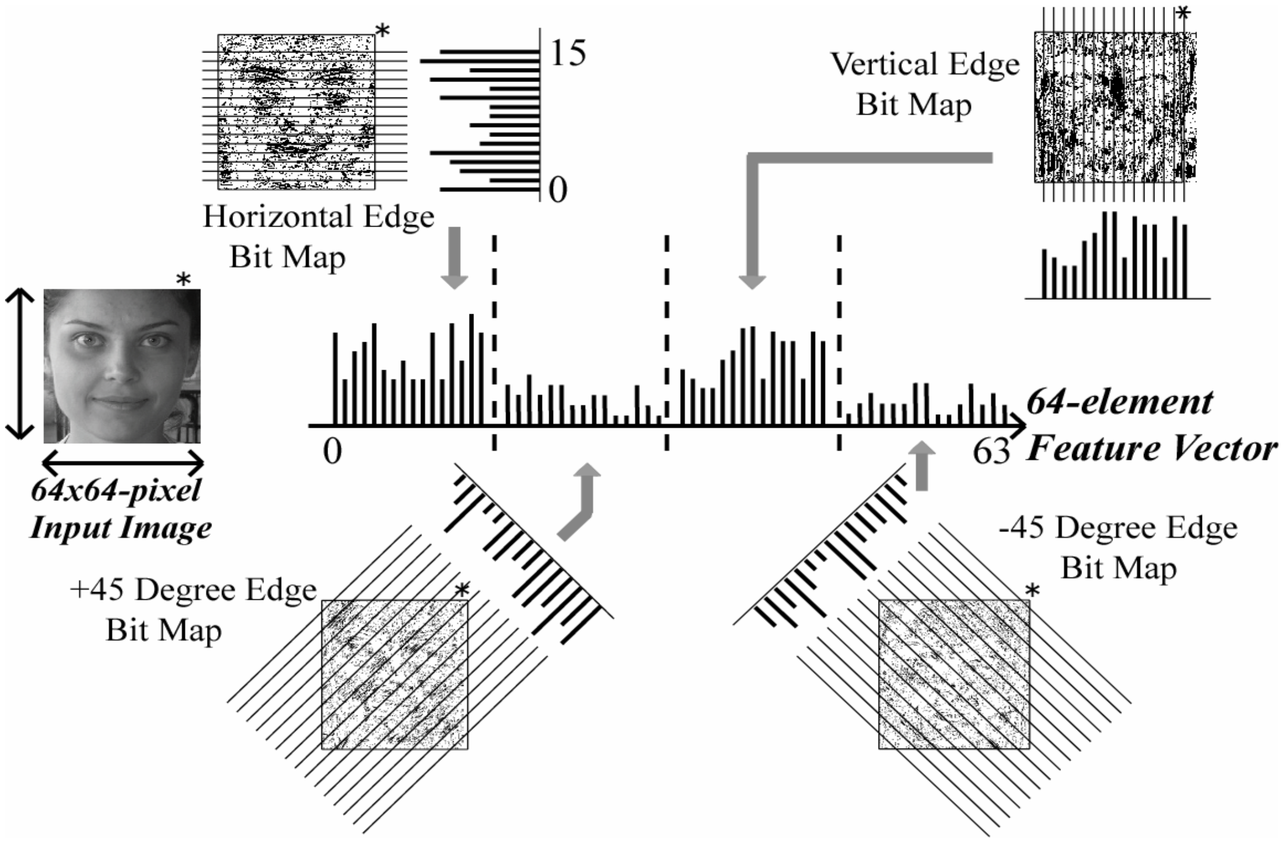


Figure 1-6. Feature Vector Generation in Projected-Principle Edge Distribution(PPED)

1.3.2 Cell Edge Distribution (CED)

Generation of the other feature vector, the cell edge distribution (CED) vector[18] is illustrated in Fig. 1-7. In this method edge bit map is divided into 4×4 square cells. Namely, each cell contains 16×16 pixels. Each edge bit map is divided into 16 blocks as illustrated in Fig. 1-7, thus 64 blocks are generated from four directional edge bit maps. Each element in the 64-dimension feature vector represents the number of edges within the corresponding block in the feature maps.

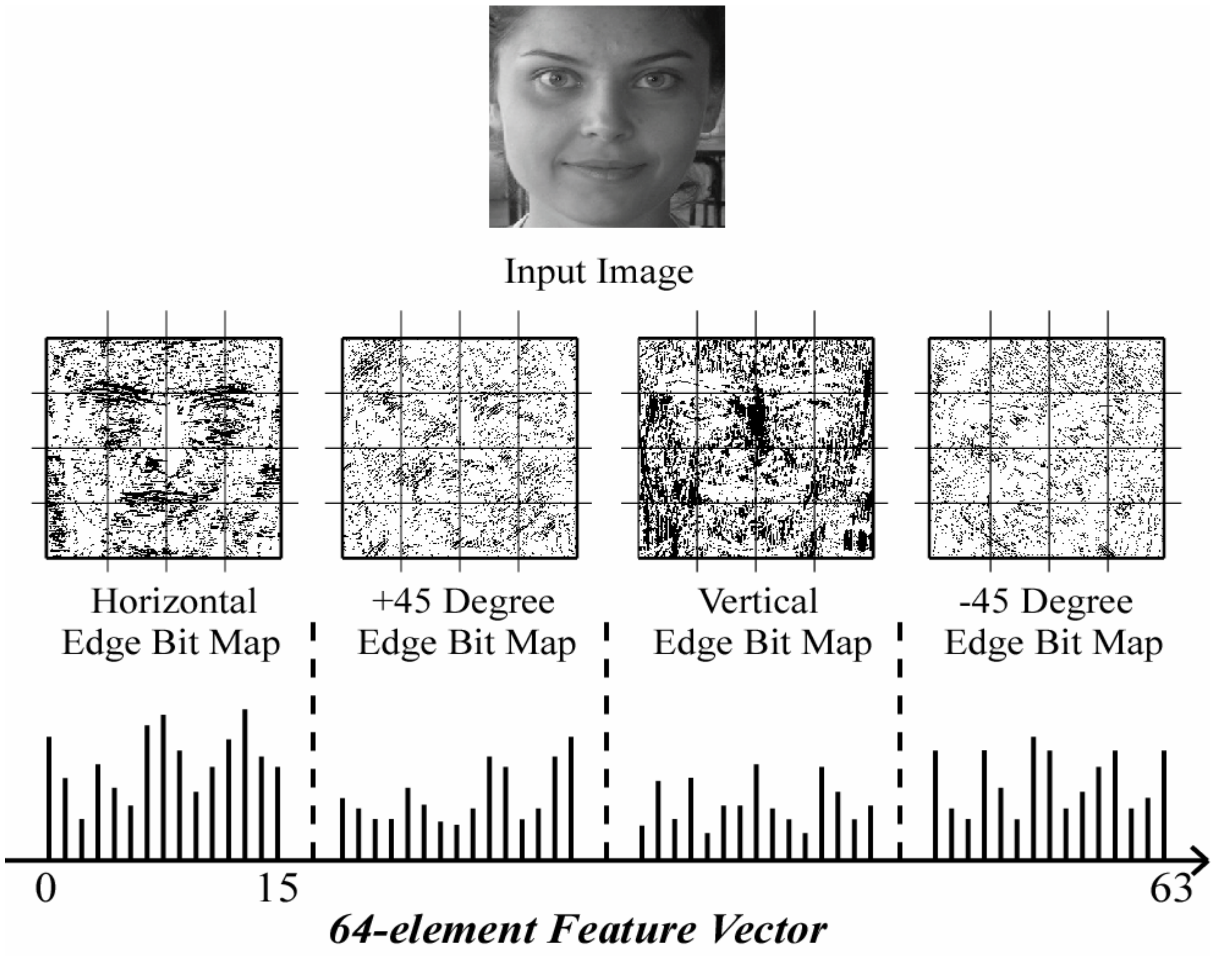


Figure 1-7. Feature Vector Generation in Cell Edge Distribution

1.3.3 Eyes-Mouth Extraction for Face Detection

A face specific feature vector generation scheme called the eyes-and-mouth (EM) extraction[18-19] is shown in Fig. 1-8. Two 16-pixel-high bands of rows are cut from the horizontal edge bit map. They correspond to the location of eyes and a mouth when the 64 ×64-pixel window encloses a human face. Then the number of edge flags in two neighboring columns is counted to yield a single vector element in a 64- element feature vector. The region cut from eyes part form the elements from 0 to 31, whereas the region of mouth forms the elements from 32 to 63. By this method of projection, feature vectors containing more information about the most specific parts of the face result in better recognition of faces in a given scene.

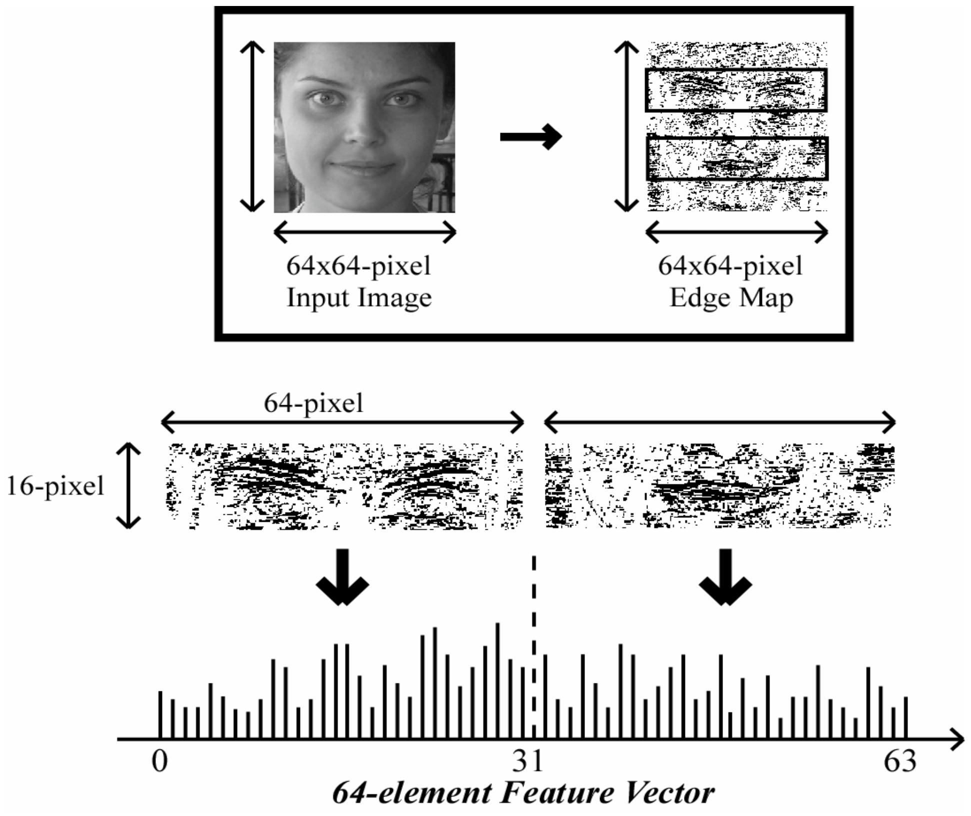


Figure 1-8. Feature Vector Generation in Eyes-Mouth Extraction Algorithm during Face Detection

1.3.4 Ego-motion Detection

Feature vector, generated by another projection method, is employed in the algorithm developed in our laboratory for ego-motion detection[20]. In this method a feature vector constructed from vertical edge bit map is used for local motion detection in horizontal direction and a feature vector constructed from horizontal edge bit map is used for local motion detection in vertical direction. For instance, 64x32-pixel block is considered and vertical edge bit map of this block is used to create 32-element feature vector. Each 64 rows for each column are added to form one element of the feature vector. Similar procedure takes places for horizontal edge bit map of region 64x32-pixel.

1.4 Review of the Conventional Hardware Work

Before discussing about conventional works, some important terms used in this thesis are described as follows:

- Feature vector: 64-element vector representing obtained from 64x64-pixels region and each element has 8 bits.

- Input Scene: Input image of any size, e.g.MxN-pixels, where a specific object is looked for.
- Input Image of an Object: Input image of size 64x64 containing a specific object entirely. It is used for generating the feature vector for the specific object, then this vector is stored in the memory representing that object type.
- Recognition Window: Small region (of size 64x64-pixels in our case) taken from the input image. Feature vector is generated for this region, then compared with the existing vectors in the memory.

Until this work, there have been different earlier implementations of edge detection and vector generation procedure. However, there was one common point of those approaches. In those approaches while generating feature vectors from recognition window, first of all extracted edge-flag bits of 64x64-pixel region are stored and processed to create the feature vector and thrown away. Then another recognition window is considered, again extracting edge-flag bits from that region, then creating the feature vector and throwing the extracted edge-flag bits. To create a feature vector from recognition window, we have to first extract edge-flag bits and then do projection, than go on with another region. It was very slow at the beginning. Then much faster hardware architecture was developed to generate feature vectors from an input scene. It was the best between conventional approaches and will be explained here.

In conventional work, corresponding to 64x64-pixel, 64x64-array of shift registers is employed. General flow is shown in Fig. 1-9. Starting from the top-left corner of the input scene, the whole image is scanned by 64x64-pixels recognition window, sliding the window 1-pixel at each time. Since it is constructed by shift registers, it is easy to slide the edge-flag bits and scanning is easily achieved. It proceeds in the same manner until the end of the scene; each time extracting the edges for new recognition window, generating the feature vector and throwing away used edge-flag bits after it.

Each unit of the shift register array is composed of shift register, 4 bits mask memory for each direction, totally 16 bits for four directions, masking circuitry and there are local adders and final adder to get the desired sums. Projection is done by giving 6 bits instruction, 4 bits for selecting the mask ID and 2 bits for selecting the edge-flag type (horizontal, vertical, etc…). Each edge-flag bit map is divided into 16 groups by assigning mask IDs to each pixel from 0 to 15 for each map. So for one pixel, there are 4 mask memory to store mask IDs corresponding to four different maps. After edge-extraction process, each pixel is assigned to be belonging to one of the edge-flag maps. Then 6 bits instructions are broadcasted in order from 0 to 63, only cells that match the broadcasted ID transfer their edge-flag to the local adders and final adder at each instruction. Hence projection is achieved in 64 steps for all edge-flag bit maps of 64x64-pixels region.

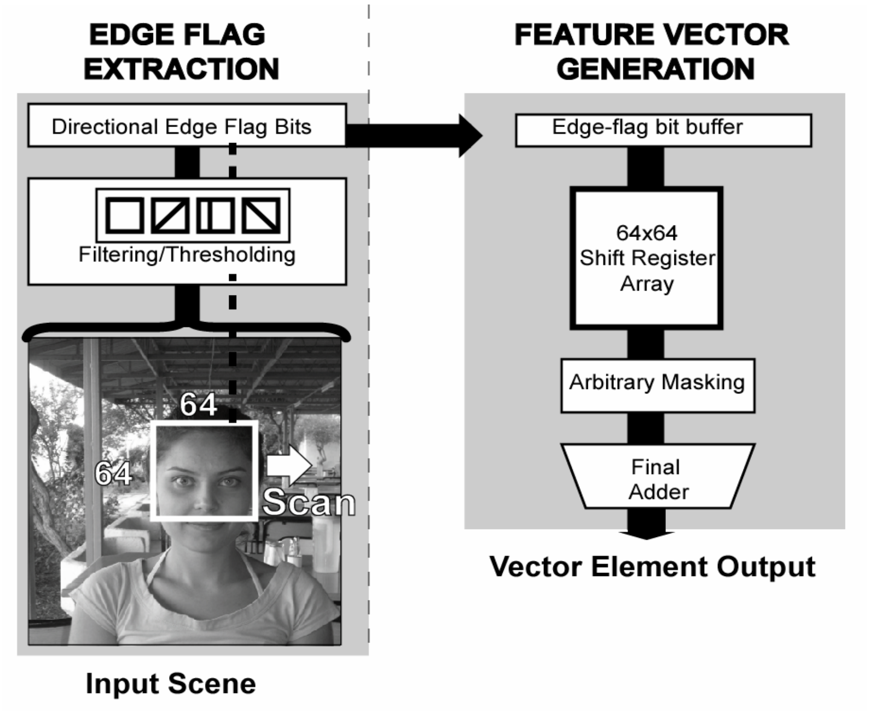


Figure 1-9. General Architecture of the Conventional System

1.5 Problem and Purpose of This Work

Edge-extraction is a complicated and very time-consuming process. In conventional approach as explained above, for each region, each time, edge flag bits are extracted, used and thrown away after the operation, causing a waste of computationally very expensive information. At each step, edge-extraction should be done again. There are many redundant operations, which is not desirable for real-time recognition systems. Moreover, with this region-by-region approach edge information of the different regions can never be accessible at one time, preventing the coordination of the different regions and further development of the recognition system. One more point is that conventional architecture is limited by hardware structure so that it is only useful for 64x64-pixels recognition window size. So feature vectors from different size of regions can not be produced with conventional architecture, which is called scalability problem. Also, in conventional system, masking variability is limited by 4-bit mask ID memory for each pixel. It means it is only possible to split each bit map into 16 groups at most, which is inadequate for EM projection case, where horizontal edge-flag bits are divided into 32 different groups.

The purpose of the present work is to further enhance the performance and capability of recognition system by introducing a new architecture. Fig. 2-1 illustrates the proposed system. To provide the reusability of extracted edge-flag bits edges, extracted edge-flag bits of the input scene are temporarily buffered in a cache memory. Hence, it is possible to access edge-flag bits of arbitrary region on demand and to avoid the repetition of edge extraction process. This allows more efficient perception of images in a larger area of visual scenes compared to conventional works chip. Furthermore with this new architecture, recognition system is no more dependent on the previously determined recognition window size and mask ID. Any size of feature vectors can be created from arbitrary sized regions, providing high-flexibility, in other words scalability, to the system. Moreover by just adjusting single control signal, various kinds of projection, including all introduced projections above, can be achieved in this system.

1.6 Organization of This Thesis

In this paper, we describe a new approach which proposes to store the edge-flag bits of the whole image in a cache memory as shown in Fig. 2-1. It provides an on demand real-time vector generation from arbitrarily accessed regions. The organization of this thesis is as in the following. In Chapter 2, Directional-Edge-Flag Cache Memory Approach is explained and proposed architecture is presented in detail. Chapter 3 talks about the small-scale proof-of-concept chip architecture and FPGA implementation of the idea for verification. Important details about main parts of the proposed architecture are given in this chapter with simulation results and discussions. Later, prototype chip, which is designed for larger memory size and larger input scene, is discussed in Chapter 4. Prototype chip has a small improvement compared to the proof-of-concept chip, which gives us much faster and efficient processing with low power consumption. Finally Chapter 5 summarizes the conclusions of this thesis along with future directions for research.

Chapter 2

Directional-Edge-Flag Cache Memory Approach

2.1 Introduction

In this work extracted edge-flag bits are proposed to be stored in a cache memory providing arbitrary access of edge-flag bits from any region of the given scene on demand. Proposed architecture is shown in Fig. 2-1. Since memory technology is improving day by day, cheap and high-speed memories are possible to use for any purpose. In this respect cache memory architecture brings a very useful and effective solution for real-time object recognition system. However, important and most difficult point is to develop a good architecture to process the information read from memory in a fast and efficient way. A new approach of processing was developed to create feature vectors by reading from the memory. In this chapter, first of all the theory of the proposed vector generation method is discussed with examples. Later, general structure of the hardware architecture is presented in detail.

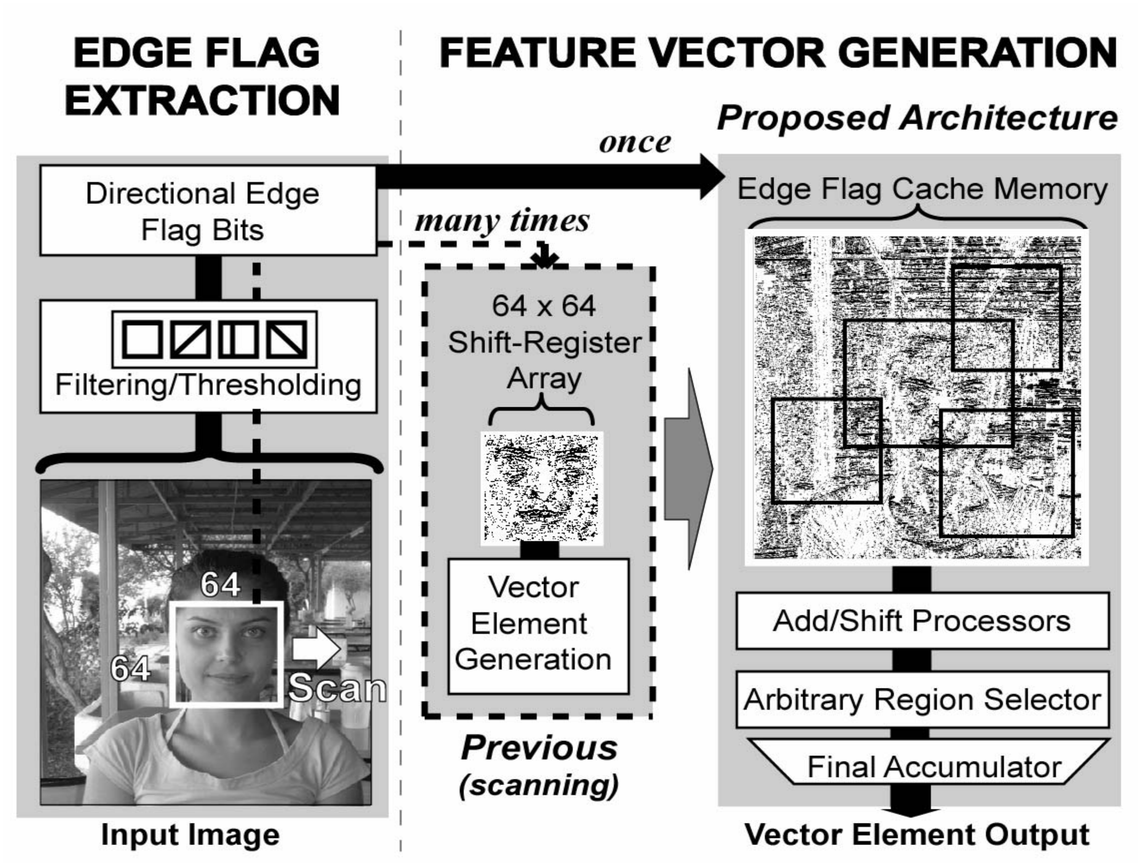


Figure 2-1 General Architecture of the Proposed System

2.2 Vector Generation Method Employing Cache Memory

The basic operation in a feature vector generation process is the summation of the edge flag bits within periodic slots. Considering the edge-flag bit cache memory, in a simple way, we can do it by reading the edge-flag bits and adding them in an appropriate way consecutively to create vector elements. However, it is the key point to find a way to generate feature vector, in other words to do projection, which is applicable to all cases. As explained before there are different ways of projection, for instance, PPED, CED, EM… Vector generation method employing cache memory should be effective for all kinds of projection in real-time.

Before explaining the proposed approach in this thesis, there is one point that should be made clear. In our recognition system, there are four different kinds of edge-flag bit maps and projection method is sometimes same, sometimes different for all these four maps. To represent the four different maps in the system, for different bits are used corresponding to each pixel position. 4x64x64 bits information is stored for 64x64-pixel region. For instance for 64x64-pixel region, considering cache memory, it is possible to keep the whole information in a 128x128-bit memory.

However this kind of approach is not practical to do different projections depending on the edge-flag bit type.

In this work, another approach is applied to store edge-flag bits for four kinds of edge-flag bit maps. In this approach, for each kind of edge-flag bit map, a cache memory is assigned to store the edge-flags. Namely, different cache memories are considered holding only one bit information corresponding to each pixel position of the scene. For 64x64-pixel region, we can think about different cache memories of size 64x64 bits for different edge-flag bit maps. Furthermore we have improved this idea by using the analogy between horizontal and vertical bit map pair and +45 degrees and -45 degrees bit map pair. When we rotate the horizontal edge-flag bit map by 90 degrees, it becomes similar to vertical edge-flag bit map. In the same way, if we rotate the -45 degrees bit map by 90 degrees, it becomes similar to +45 degrees bit map. By this way we can think about only two kinds of cache memory approach. One is storing edge flag bits for just vertical addition of flags and the other is for diagonal addition of flags.

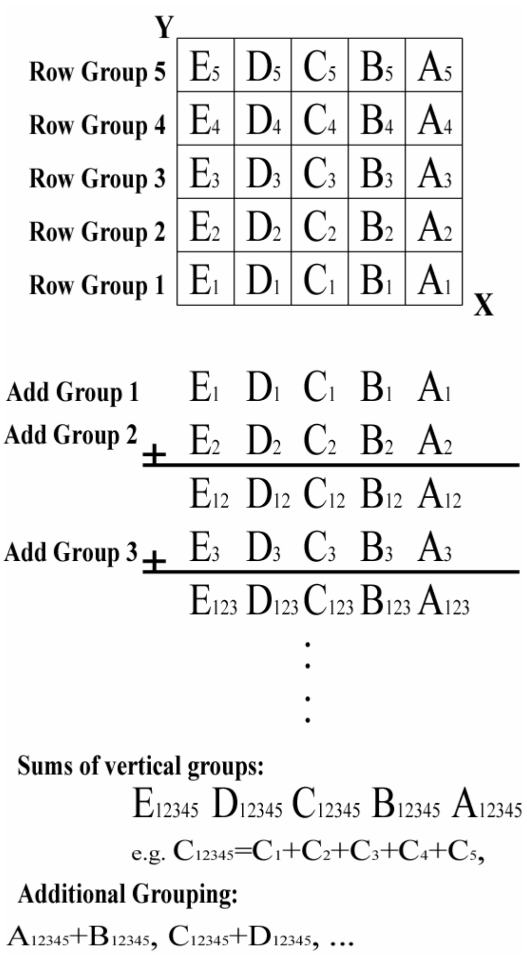
As stated at the beginning of this section, thinking about different architectures for different kind of bit maps is not an efficient way. A robust real-time responding system should be effective for different kinds of projection and it should be small to produce compact hardware systems. In this work, single general cache memory architecture applicable to various projection types is developed.

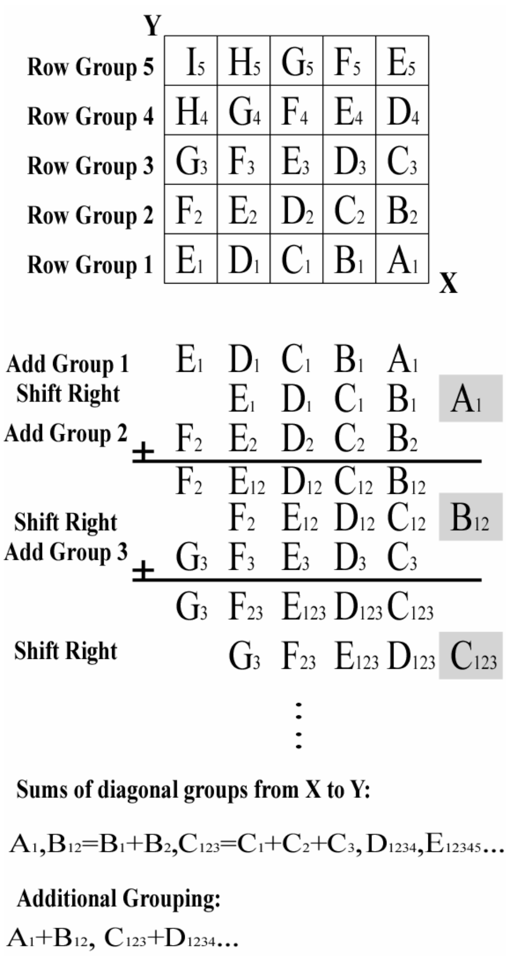
Regarding the different projections types explained before, it can be said that there are two basic operations. One is the addition of vertical elements in the memory; the other is the addition of diagonal elements. For vertical case, it can be easily done by consecutive add operations of the data read out from the memory. However, for diagonal case, the addition of the diagonally adjacent edge-flag bits is very complicated. To achieve both vertical and diagonal projections, add and shift algorithm has been introduced as explained below.

2.2.1 Add and shift algorithm

Fig. 2-2a illustrates the basic steps for vertical case. To obtain the sums of the elements in each column, first one row is read from the memory, and then at each step next row is read and added to the previous value. Successive add operations are enough to get the sum of each column.

For diagonal case Fig. 2-2b shows the basic steps. When add operation is applied, one row group is read from the memory and added to the existing value. When shift operation is applied, the value in the group is shifted to the right. For diagonal case, we only need to add one row group and shift it to the right and add the next row group to obtain the sum of the diagonally adjacent units. Most of the time, not the sum of only one group but sum of adjacent groups is required to construct one vector element. For example, for PPED 4 columns are added to construct one vector element. So to group the sums, one more periodic addition operation can be done after all basic sums are obtained as shown in Fig. 2-2b. The details of the whole algorithm and hardware units are explained in next section.


Y
Row Group 5 E5 D5 C5 B5 A5
Row Group 4 E4 D4 C4 B4 A4
Row Group 3 E3 D3 C3 B3 A3
Row Group 2 E2 D2 C2 B2 A2
Row Group 1 E1 D1 C1 B1 A1
X
Add Group 1 E1 D1 C1 B1 A1
Add Group 2 + E2 D2 C2 B2 A2
E12 D12 C12 B12 A12
Add Group 3 + E3 D3 C3 B3 A3
E123 D123 C123 B123 A123
Sums of vertical groups:
E12345 D12345 C12345 B12345 A12345
e.g. C12345=C1+C2+C3+C4+C5,
Additional Grouping:
A12345+B12345, C12345+D12345, ...


Y
Row Group 5 I5 H5 G5 F5 E5
Row Group 4 H4 G4 F4 E4 D4
Row Group 3 G3 F3 E3 D3 C3
Row Group 2 F2 E2 D2 C2 B2
Row Group 1 E1 D1 C1 B1 A1
X
Add Group 1 E1 D1 C1 B1 A1
Shift Right E1 D1 C1 B1 A1
Add Group 2 + F2 E2 D2 C2 B2
F2 E12 D12 C12 B12
Shift Right F2 E12 D12 C12 B12
Add Group 3 + G3 F3 E3 D3 C3
G3 F23 E123 D123 C123
Shift Right G3 F23 E123 D123 C123
Sums of diagonal groups from X to Y:
A1,B12=B1+B2,C123=C1+C2+C3,D1234,E12345...
Additional Grouping:
A1+B12, C123+D1234...

2-2a. Addition of vertical units 2-2b. Addition of diagonal units.

Figure 2-2a and 2-2b. Explanation of the proposed theory for vector generation.

2.3 Explanation of Cache Memory Architecture

In the proposed architecture extracted directional edge-flag bits of the entire image are stored in a cache memory and made accessible on demand. There are two key issues: first one is the minimum-latency projection of the directional edge flag bits by reading from the memory; second one is to define the borders of the region of interest to be projected for feature vector generation. General view of the proposed architecture is demonstrated in Fig. 2-3.

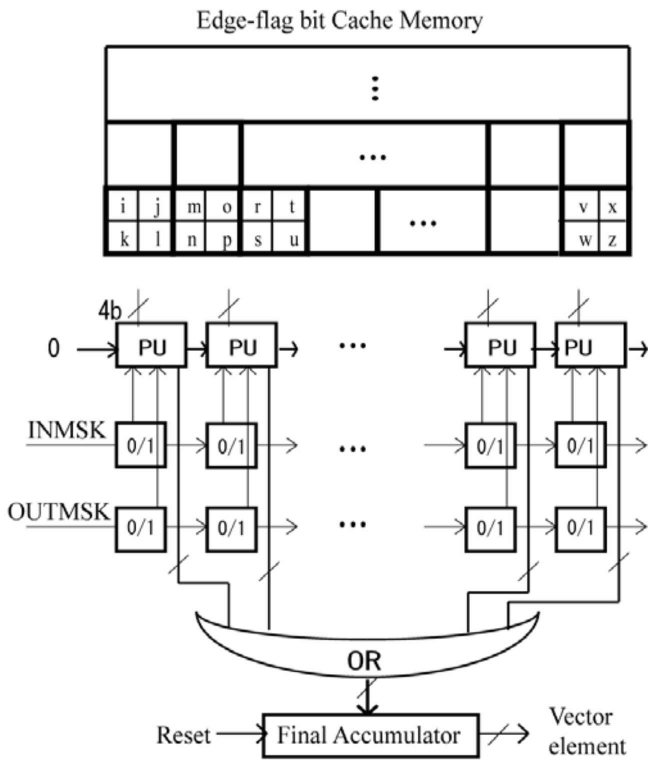


Figure 2-3. General Structure of Edge-Flag bit Cache Memory Vector Generation Architecture

In the proposed algorithm, instead of dealing with each bit separately; edge flag bits are grouped in (2x2) 4-pixels units for the sake of simplicity and efficient processing. In other words, whole region is assumed to be divided into groups vertically and horizontally, so that two adjacent rows make one row group and two adjacent columns make one column group. And a processing unit(PU) is attached to each column group(two column pairs) for add/shift operations. When add operation is applied, 4-pixels unit is read from the memory for each column group and added to the existing value. When shift operation is applied, the value in the group is shifted to the right. To achieve simultaneity, 2-port memory architecture is employed. Therefore two adjacent rows can be read at the same time such that for each column group 4-pixels unit of the 2x2 group can be read and passed to the PU.

The main aim of this research is to be able to do projection in arbitrary regions on demand. So we should be able to determine the borders of our recognition window anytime. While defining the borders of the interest region masking method is used. Two control signals are used to define the beginning and the end of the region. INMSK signal is used to define the beginning of the interest region, OUTMSK signal is used to define the end. For each PU, an input control mask value(0/1), adjusted by INMSK signal, and an output control mask value(0/1), adjusted by OUTMSK signal, are stored. If input mask of a PU is 1, then it means that PU takes the shift input from the previous-left-PU. If it is 0, then shift input of the PU is zero. Hence no value is passed to the PU, making PU as the starting point of the region. Similarly, if output mask of a PU is 1, it means that PU sends down its current value to the common OR circuitry. If it is 0, then PU does not send its value but it sends only zero. Here, the output of a PU going to the next-right-PU is not affected by output mask and this will be explained in the detailed structure of processing units later in next chapter. Therefore, output mask determines the end of the region, PU having the output mask of 1 becomes the last PU unit, where the sums of the column groups are sent as a result of successive shift operations. And finally it is possible by an accumulator to group these sums according to the period of the slot of the projection type.

Final accumulator is controlled by RESET signal. If RESET signal is 0, it adds the incoming input value with zero. If RESET signal is 1, then it adds the incoming value to the existing value.

2.4 Summary

In this chapter, a new approach employing edge cache memory architecture was presented. The algorithm to generate feature vectors by reading edge-flag bits from memory for different kinds of projections was explained with examples. And then proposed hardware architecture was discussed with important characteristics. Proposed architecture is a single compact architecture to generate feature vectors for any kind of edge-flag information, e.g. horizontal, vertical, etc…As mentioned above, for each kind of edge-flag bit map, one cache memory architecture is accommodated. And with the parallel processing of these architectures, vector elements constructed from different edge-flag bit maps are obtained in parallel. Besides, when the projection method for one edge-flag bit map is changed, it is enough only to change the add/shift control signals to achieve different projection. Furthermore, a change in recognition window size also doesn't affect the hardware architecture. Again it can be easily done by just changing control mask signals. With the proposed approach, high-speed generation of feature vectors is possible from arbitrary regions on demand for various kinds of projection. For more complicated systems, this architecture is very appropriate for parallel processing of several edge-flag bit maps.

Chapter 3

Proof-of-Concept Chip and FPGA Implementation

3.1 Introduction

For verification of the proposed architecture, a proof-of-concept chip has been designed employing small size, 16x12-bits, cache memory. Furthermore, FPGA implementation of the proof-of-concept chip has also been established. In this chapter first of all, general architecture of the prototype chip is explained. Next, design strategy and details of the major circuits are explained. After that, some performance characteristics of the proof-of-concept chip and used technology are introduced. Later, details of FPGA implementation are given with measurement results and further discussions.

3.2 Explanation of General Architecture

To verify the proposed architecture, a proof-of-concept chip was designed employing edge-flag cache memory of 16x12-bits. General architecture is shown in Fig. 3-1. Memory part includes 4x16 decoder and 16rowsx12columns dual-port SRAM memory. Since 2x2-bits grouping is utilized in this architecture, dual-port memory is required to read data from adjacent rows simultaneously.

In this system, all the PUs are working. They are either calculating the sums of the input 4-bits or shifting their value to the next PU depending on the control signals. The region of interest is determined by INMSK and OUTMSK signals. INMSK signal decides the starting PU. For one of the PU, it cuts the flow of data coming from previous stages, making zero to enter as shift input. OUTMSK signal selects the final data going to the final accumulator. Output coming from the selected PU is passed to the OR circuit, for other PUs zero will be passed.

To develop a real-time responding system, one of the major priorities is the speed of hardware units. On the other hand reasonable power consumption should also be in consideration. In many systems memory part has been demonstrated to be the main power-consuming unit. During this design, main consideration is to develop high-speed robust hardware directional-edge-flag vector generation architecture with minimum area. Besides this, some low power-consumption techniques have been applied while designing the cache memory part. So different surveys and simulations about different circuit structures have been done and circuit structures having high-speed, low-power characteristics have been chosen.

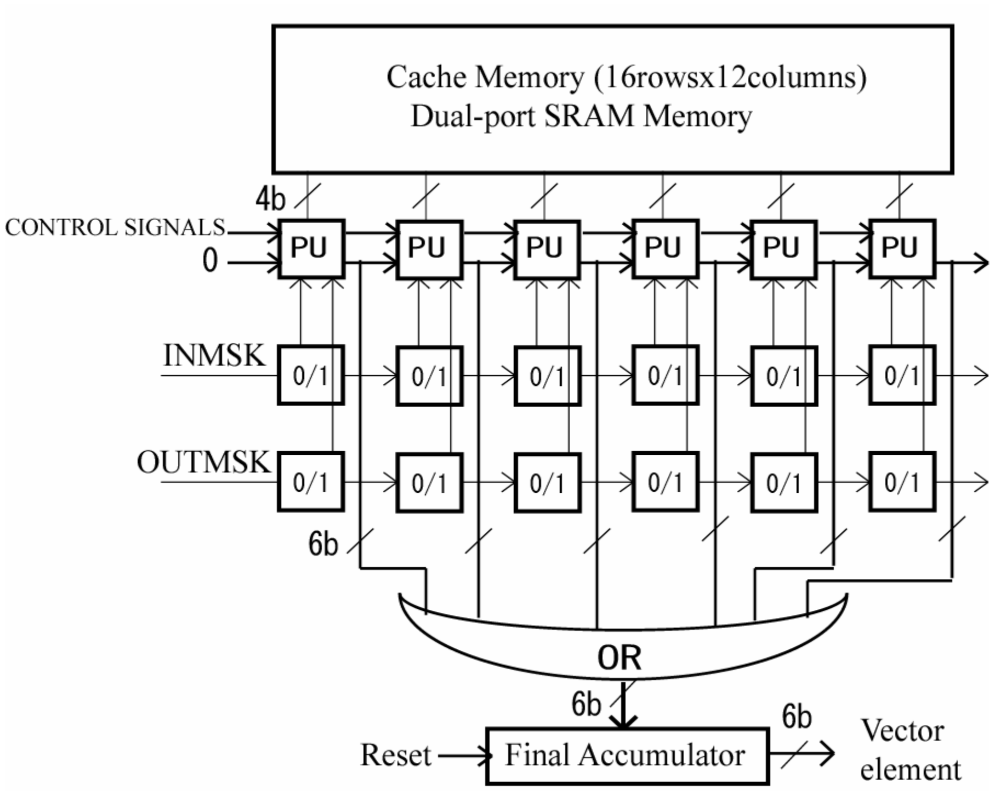


Figure 3-1. General Architecture of the Proof-of-Concept Chip

3.2.1 SRAM Details

In our design, speed of SRAM read/write operation is very important. SRAM is mainly composed of decoding circuit, pre-charging circuit, sense amplifier and write drivers. To design a high-speed SRAM memory, design of the decoder and sense amplifiers must be considered carefully. It is known that read operation is always slower than write operations, because it takes time bit lines to be pre-charged to the default value and to be discharged depending on the value stored in memory bit. So a good sense amplifier design is required to achieve fast accurate read operation. On the other hand, decoding stage takes much longer time (about %70-%80 of the read cycle) compared to the other stages during memory operation cycle. Moreover since outputs of the decoders are connected to the word lines and these lines become longer for large sized memories, when the size of the memory increases, the delay in decoding stage also increase. A fast and effective decoder design is required for SRAM to work in desired time properly for all memory cells.

So different circuits for decoding, pre-charging, sensing have been investigated and examined and finally optimum circuit structures have been chosen to utilize fast low-power architecture. The details of these circuits and memory are explained in the following.

To process two rows at the same time, dual-port memory has been utilized. Dual-port memory cell is shown in Fig 3-2. below. In this dual-port memory architecture wordline1 selects the first row to be read, while wordline2 selects the second row to be processed simultaneously. Two decoders, one for each word line have been used in the design. All the read/write circuits (pre-charging, decoder, sense-amplifier, write drivers) have been used for each bit line pair.

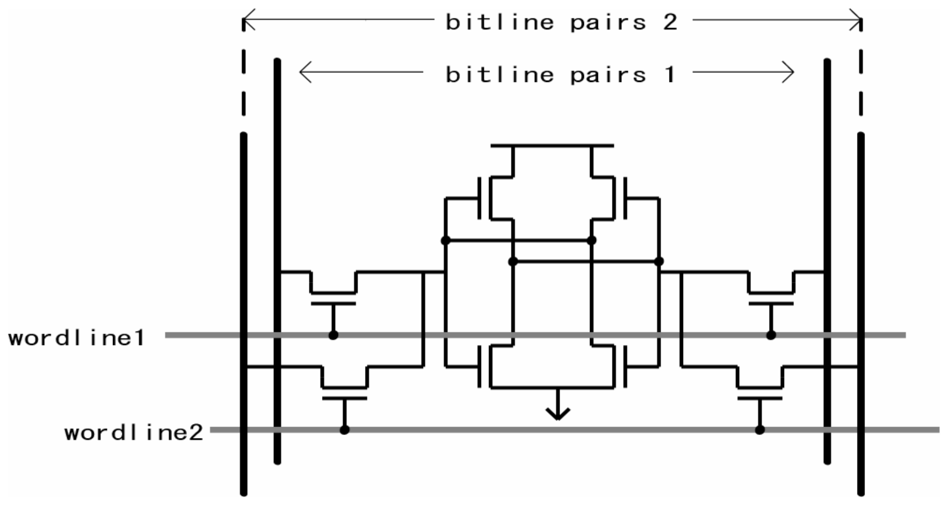


Figure 3-2. Two-port SRAM Memory Cell Structure

During memory design, some circuit techniques have been applied to increase the speed and decrease the power. These are: pulsed width line (PWL) for decoding and isolated bit line (ISB) for sense amplifier stages. Besides, skewing has been applied to decoding circuits to enhance the performance as explained below.

<u>Decoder</u>

The decode gate delay can be significantly reduced by using pulsed circuit techniques [20-22], where the word line is not a combinational signal but a pulse which stays active for a certain minimum duration and then shuts off. The idea of PWL is to minimize the duration of active input on word lines by deactivating the word lines (and SRAM cells) before the bit line voltages make a full swing. Thus, before any access all the word lines are off, and the decoder just needs to activate the word line for the new row address. This leads to a reduced power consumption and enhanced speed. To reset the output, there are many different circuit techniques, among them DRCMOS is proved to have the least logical effort and hence the lowest delay and used in this design. DRCMOS circuit structure is shown Fig. 3-3.

In pulsed decoders the input stage of the decoder needs to be gated by a clock; we can take advantage of this by implementing the NAND function in the domino style, thus lowering the loading on the address inputs. Dynamic NAND in domino style consume the least amount of power as they disconnect the direct VDD/GND path all the time [29]. This way the dynamic power is minimized, while controllable pulses are generated at the output (for PWL). Since the word line selection requires each gate in the critical path to propagate an edge in a single direction, the transistor sizes in the gate can be skewed to speed up this transition and minimize the decode delay. By reducing the sizes for the transistors which control the opposite transition, the capacitance of the inputs and hence the logical effort for the gate is reduced, thus speeding up the decode path.

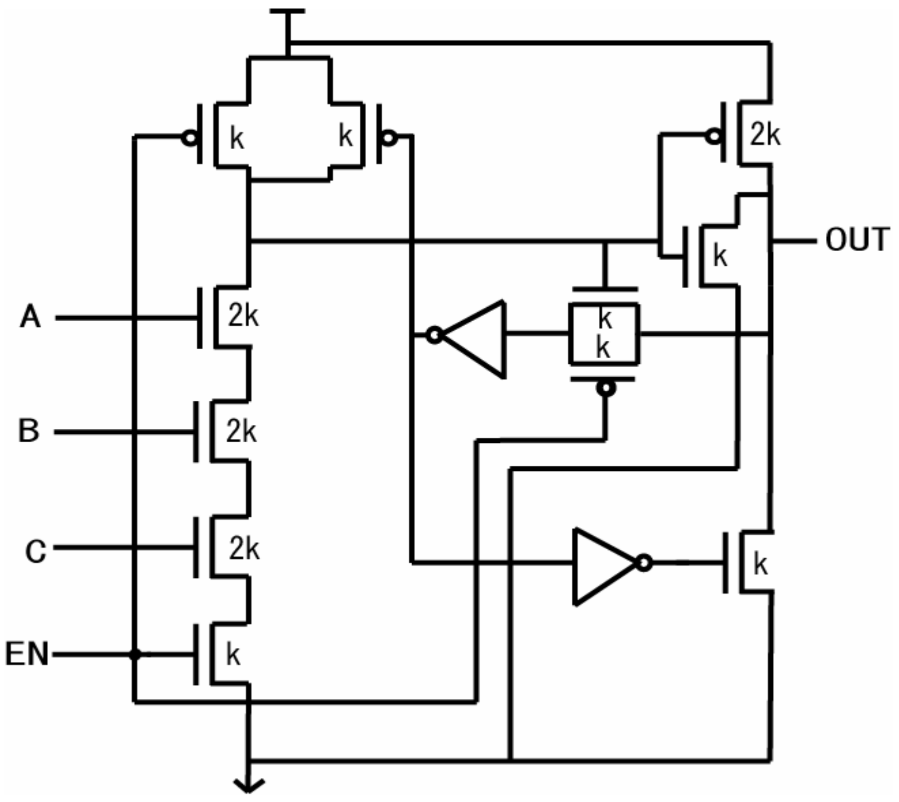


Figure 3-3. Circuit Structure of Skewed 3-input DRCMOS NAND.

Sense amplifier

A number of different sense amplifier circuits have been proposed in the past and they essentially fall into two categories: the linear amplifier type and the latch type[20-22]. In the linear amplifier type, the amplifier needs a DC bias current to set it up in the high gain region prior to the arrival of the bit line signal. Because they consume biasing power and since they operate over a limited supply voltage they are not preferred for low power and low voltage designs. In our design the latch type sense amplifier is used as demonstrated in Fig. 3-4. It consists of a pair of cross coupled gain stages which are turned on with the aid of a sense clock when an adequate input differential is set up. The positive feedback in the latch leads to a full amplification of the input signal to a full digital level. While this type consumes the least amount of power due to the absence of any biasing power, a very careful timing is required for correct operation. If the sense clock arrives before enough input differential is set up, it could lead to a wrong output value.

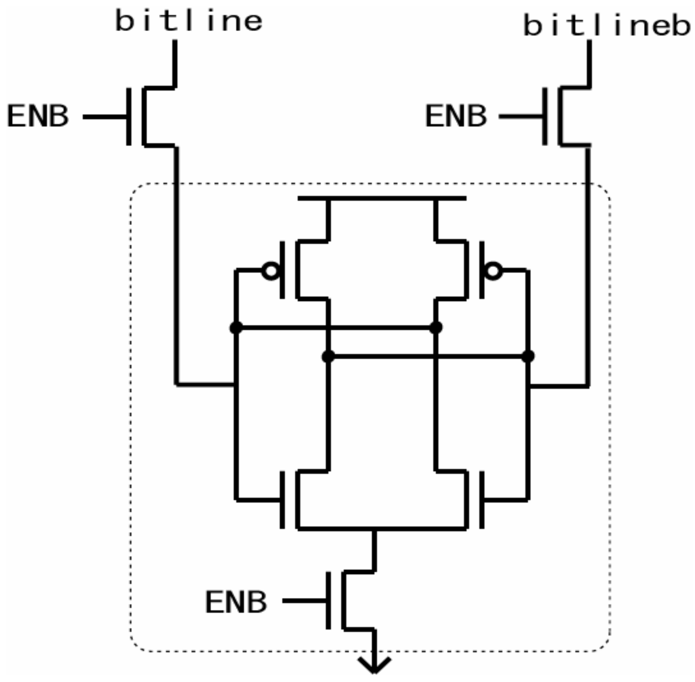


Figure 3-4. Latch Type Sense Amplifier with Isolation Transistors

The last technique used in this design is the Isolated Bit line Scheme (IBL). Sense amplifiers are included in the memory-read circuitry to speed up the read operation. In IBL scheme a pair of isolation transistors is used to disconnect the sense amplifier from the bit lines by the time the correct data is detected (Figure 3-4). Here, the sense amplifiers attached to the bit lines are isolated after they detect a sufficient voltage difference on the bitline and bitlineb. This prevents a full swing on the entire bit line and saves energy. The sense amplifiers are also isolated from the bit lines during the entire write operation as they are not needed.

Pre-charge Circuit

Pre-charging circuitry used in our design is demonstrated below.

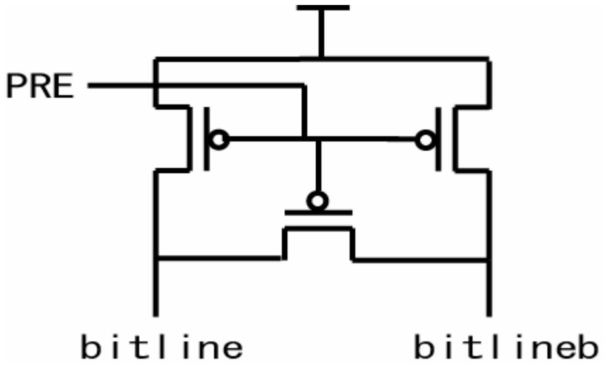


Figure 3-5. Pre-charging Circuit Structure

3.2.2 Explanation of Processing Units

Processing unit circuit is demonstrated in Fig. 3-6. It is a very simple architecture which is designed for add and shift operations for 5-bits data at most. It is composed of adder units, selector, register and output masking units. Adder unit adds the incoming four bits from memory columns. Selector unit decides the values to be sent to the next step, which will be added with the values coming from memory. Either existing value coming from 5-bit 2-input Adder unit or value coming from previous stage, 5-bit shift input, can be chosen to be sent to the next step. S control signal decides this value. When S is zero, it means shift the input coming from the previous stage or zero(depending on the INMSK signal) to the next stage. When S is one, it means add from the memory with the existing sum. INMSK is the 1 bit control signal coming from the input control mask; it decides weather to pass the shift input coming from PU or zero. When this PU is the start of the region of interest then zero is passed as a shift input. Output masking circuit is composed of simple AND gates. Depending on the 1 bit output mask control signal, either calculated sum in PU is passed as down output or zero is passed instead. This output goes to the general OR stage in the system. 5-bits output of the register holding the calculated sum goes directly to the next PU.
To obtain fast operation characteristic, transmission gate based adders have been employed. Also carry-propagate adders are used for long bit inputs.

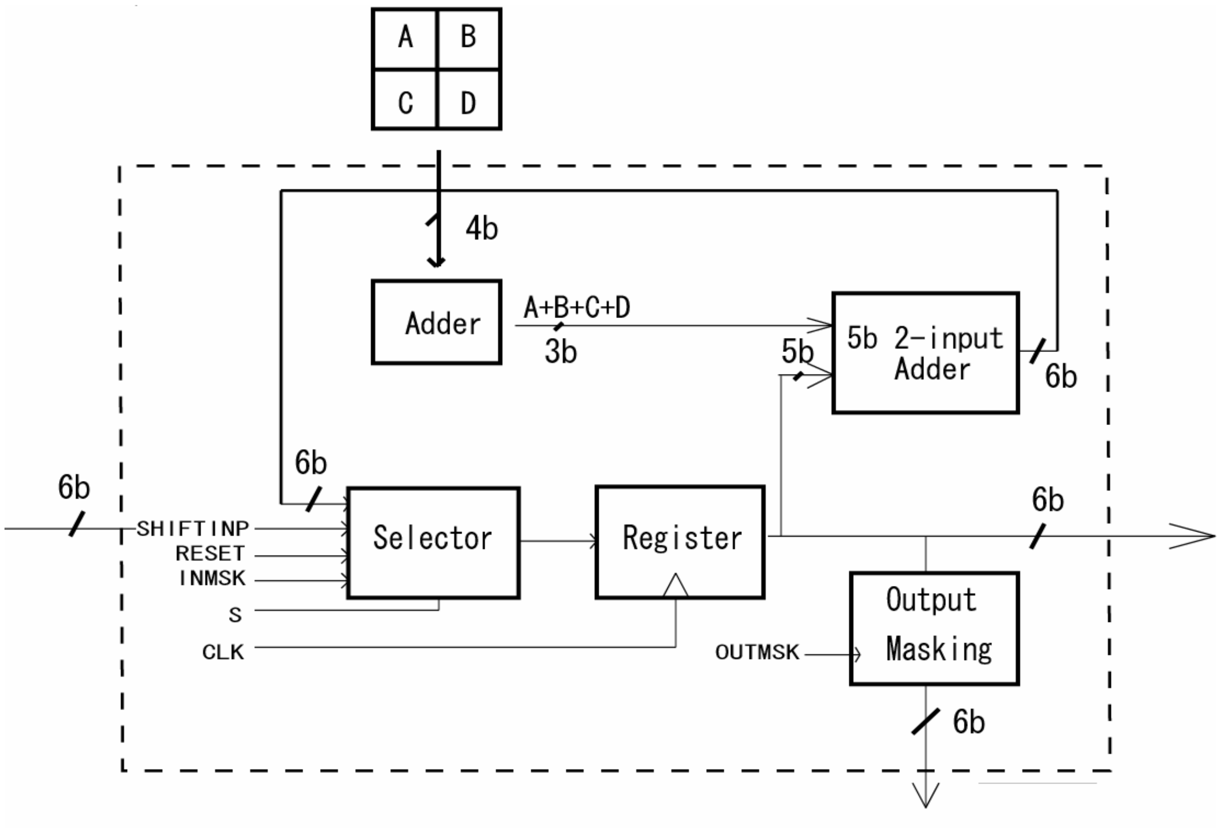


Figure 3-6. Detailed structure of processing unit(PU).

3.3 Chip Characteristics

The proof-of-concept chip employing 16x12-bits cache memory architecture was designed concentrating on the robustness and speed and low power. Proof-of-concept chip of size 2,8mmx2,8mm was designed in 0.18μ5-metal-layer technology. Core has the size 0,3mmx0,2mm and only cache memory part has the size 0,12mmx0,12mm. The simulations have been done with the clock of 500MHz frequency. Layout of the chip is demonstrated in Fig. 3-7a and 3-7b. A small size memory architecture has been designed to test clearly and it is seen in the middle of the frame in Fig. 19a. Since input lines from pins to the circuits are longer than usual, registers are used to adjust timing. Fig. 19b shows the core in a closer perspective. Upper is the cache memory with two decoders on the sides. Each processing unit is laid under column groups of 2. Finally OR and final accumulator circuits are placed at the bottom. Table 3-1 shows the characteristic of the proof-of-concept chip.

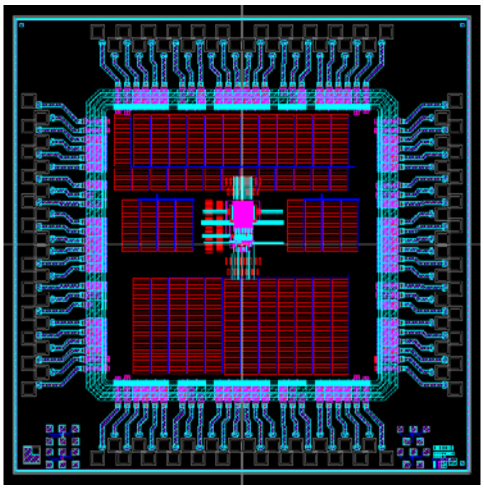
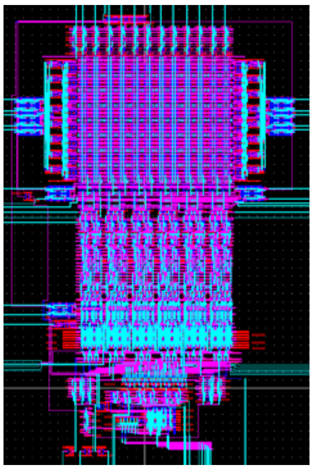



Figure 3-7a and 3-7b. Layout of designed proof-of-concept chip

Technology	CMOS,0.18mm,5-metal layer
Supply Voltage	1.8V
Die Size	2.8mm x 2.8mm (0.3mmx0.2mm)
Number of Transistors	6429
Cache Memory Size(mm)	0.12mmx0.2mm
Cache Memory Size	16x12-bits
Operation Frequency	500MHz

Table 3-1. Proof-of-concept chip characteristics.

3.3.1 Clocking Strategy

In our system, actually there are two basic operations: add and shift. During add operation it takes longer than shift operation, since edge-flag bits are read from memory. Two-phase clocking strategy is followed in this design for each mode operation with longer clock cycle and shorter clock. Shorter clock cycle is just the half of longer clock cycle. For diagonal projection, for example for 6x6-pixels region(3x3 groups) the sequence is like, add, shift, add, shift, add, shift, add, shift, shift…During consecutive add and shift operations, both can be completed in one longer clock cycle. Then for remaining shift operations, shorter clock cycle is enough to process as shown Fig. 3-8. For vertical projection, again for 6x6-pixels region, the sequence is as, add, add, add, shift, shift, shift… As a result if we call the longer clock cycle as "1 clk", then shorter cycle is "0,5 clk". For the example in Fig. 20, we can say that total process is 3xclk + 2x0,5clk = 4clk for diagonal case and 3xclk + 3x0,5clk = 4,5clk for vertical case. At the each rising edge of the clock, data is shifted to the next stage and consequently final shift output goes to the final stage.

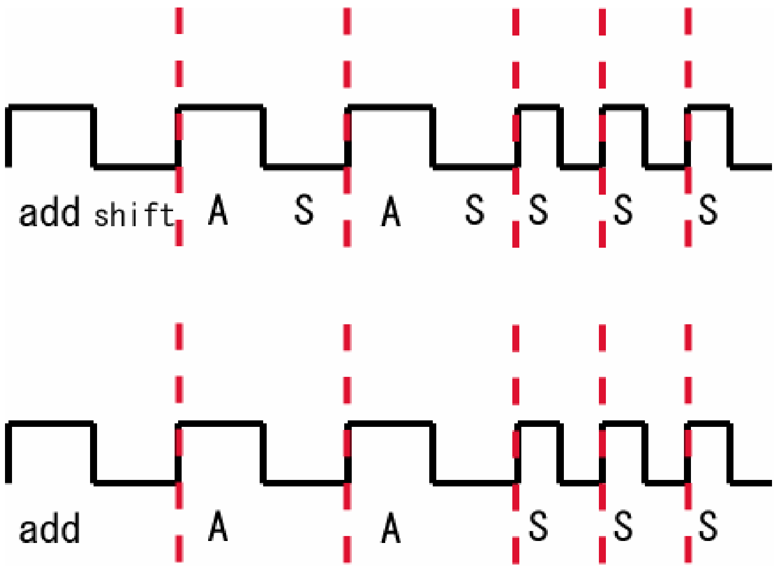
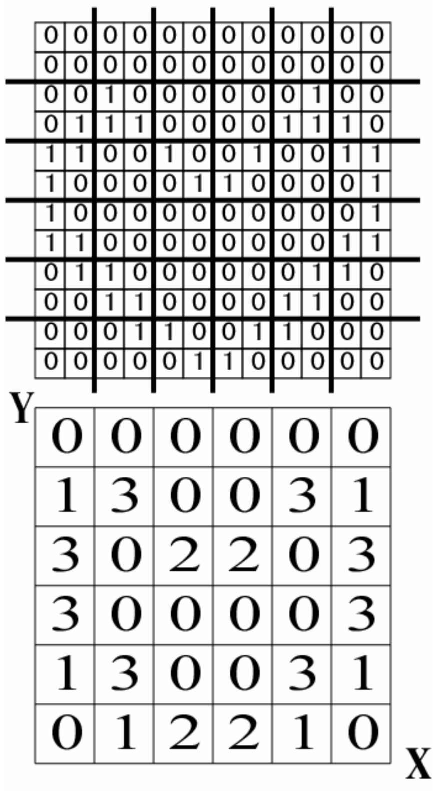


Figure 3-8. Clocking strategy of the system.

3.4 FPGA Implementation

To verify the proposed architecture, FPGA implementation of the proposed architecture was also done. The designed FPGA architecture includes 12x12-bits edge cache memory and 6 units PU structure. Cache memory containing 12x12-pixel edge-flag bits and 2x2-pixels units are shown in Fig. 3-9. ALTERA cyclone EP1C12Q240C8 FPGA device was programmed and measurements were done in Logic Analyzer with the minimum frequency possible (125MHz).

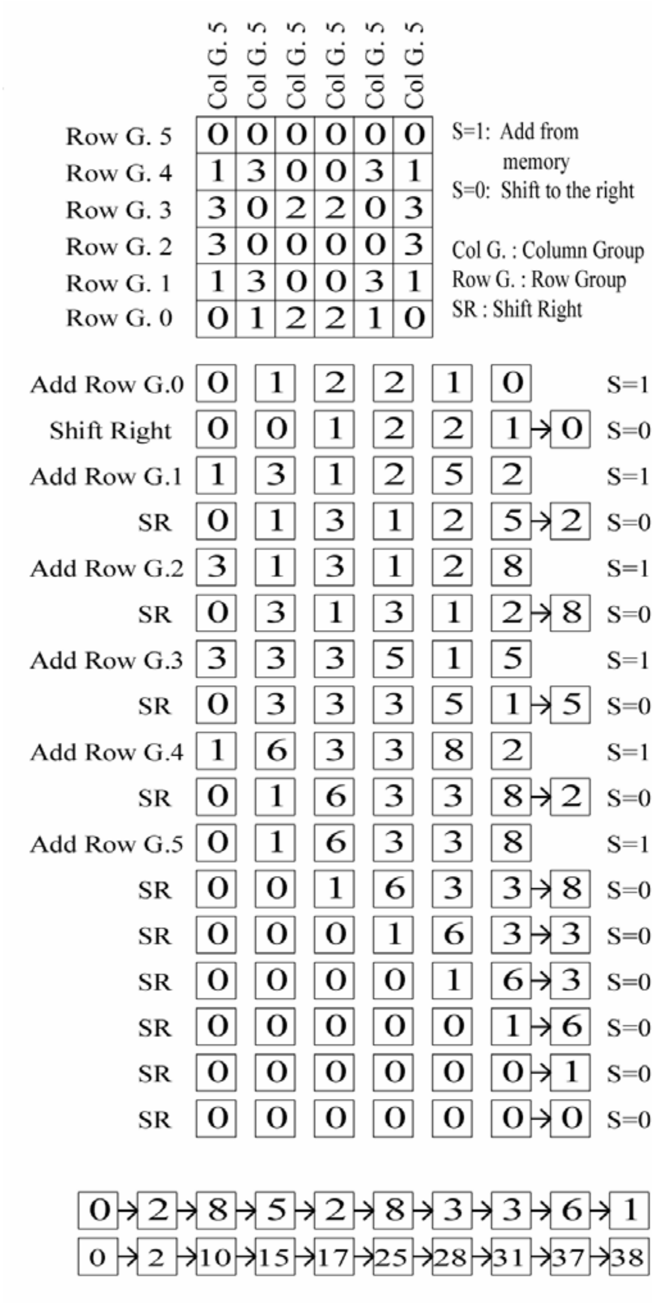


0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	1	1	1	0
1	1	0	0	1	0	0	1	0	0	1	1
1	0	0	0	0	1	1	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	1	0	0	0	0	0	0	0	0	1	1
0	1	1	0	0	0	0	0	0	1	1	0
0	0	1	1	0	0	0	0	1	1	0	0
0	0	0	1	1	0	0	1	1	0	0	0
0	0	0	0	0	1	1	0	0	0	0	0

0	0	0	0	0	0
1	3	0	0	3	1
3	0	2	2	0	3
3	0	0	0	0	3
1	3	0	0	3	1
0	1	2	2	1	0

Figure 3-9. Sample edge-flag bit cache memory contents and groupings for FPGA

In Fig. 3-10, there is a sample flow of diagonal projection of the memory contents given above. Cache memory containing 12x12-pixel edge-flag bits and 2x2-pixel units are shown. Sums of the diagonally adjacent units are as shown sequentially from bottom-right corner to top-left corner. At the bottom of the figure, the results from the final shift output are given. Finally, the outputs from final accumulator are shown in order. Final accumulator is sequentially adding the coming inputs at each clock cycle. There is no reset operation in this example. However it is possible to periodically reset the accumulator and obtain periodic sums. For example it is possible to add the shift outputs in three element groups.



Row G. 5	0	0	0	0	0	0		
Row G. 4	1	3	0	0	3	1		
Row G. 3	3	0	2	2	0	3		
Row G. 2	3	0	0	0	0	3		
Row G. 1	1	3	0	0	3	1		
Row G. 0	0	1	2	2	1	0		
Add Row G.0	0	1	2	2	1	0		S=1
Shift Right	0	0	1	2	2	1	→ 0	S=0
Add Row G.1	1	3	1	2	5	2		S=1
SR	0	1	3	1	2	5	→ 2	S=0
Add Row G.2	3	1	3	1	2	8		S=1
SR	0	3	1	3	1	2	→ 8	S=0
Add Row G.3	3	3	3	5	1	5		S=1
SR	0	3	3	3	5	1	→ 5	S=0
Add Row G.4	1	6	3	3	8	2		S=1
SR	0	1	6	3	3	8	→ 2	S=0
Add Row G.5	0	1	6	3	3	8		S=1
SR	0	0	1	6	3	3	→ 8	S=0
SR	0	0	0	1	6	3	→ 3	S=0
SR	0	0	0	0	1	6	→ 3	S=0
SR	0	0	0	0	0	1	→ 6	S=0
SR	0	0	0	0	0	0	→ 1	S=0
SR	0	0	0	0	0	0	→ 0	S=0

$$\boxed{0} \rightarrow \boxed{2} \rightarrow \boxed{8} \rightarrow \boxed{5} \rightarrow \boxed{2} \rightarrow \boxed{8} \rightarrow \boxed{3} \rightarrow \boxed{3} \rightarrow \boxed{6} \rightarrow \boxed{1}$$

$$\boxed{0} \rightarrow \boxed{2} \rightarrow \boxed{10} \rightarrow \boxed{15} \rightarrow \boxed{17} \rightarrow \boxed{25} \rightarrow \boxed{28} \rightarrow \boxed{31} \rightarrow \boxed{37} \rightarrow \boxed{38}$$

Figure 3-10. A Sample data and diagonal projection of it. Add and Shift commands are also shown.

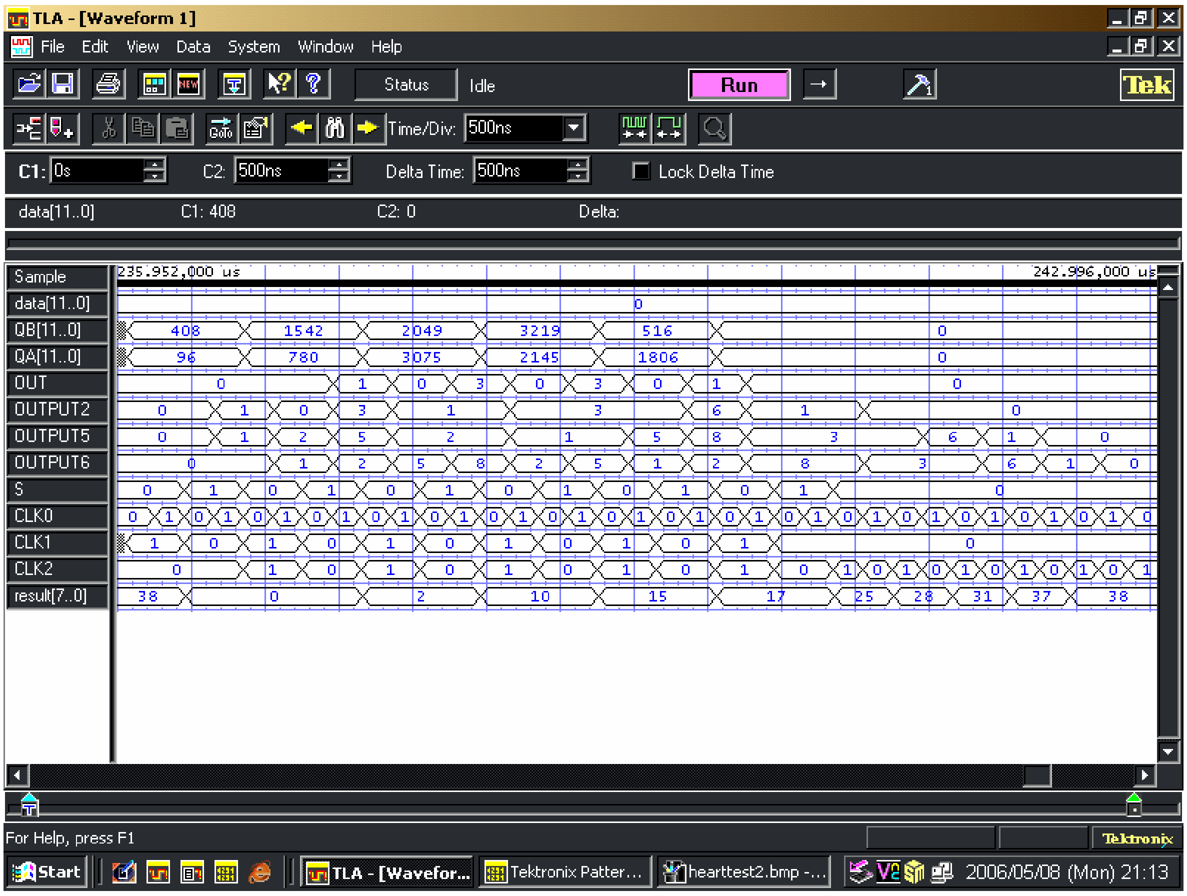
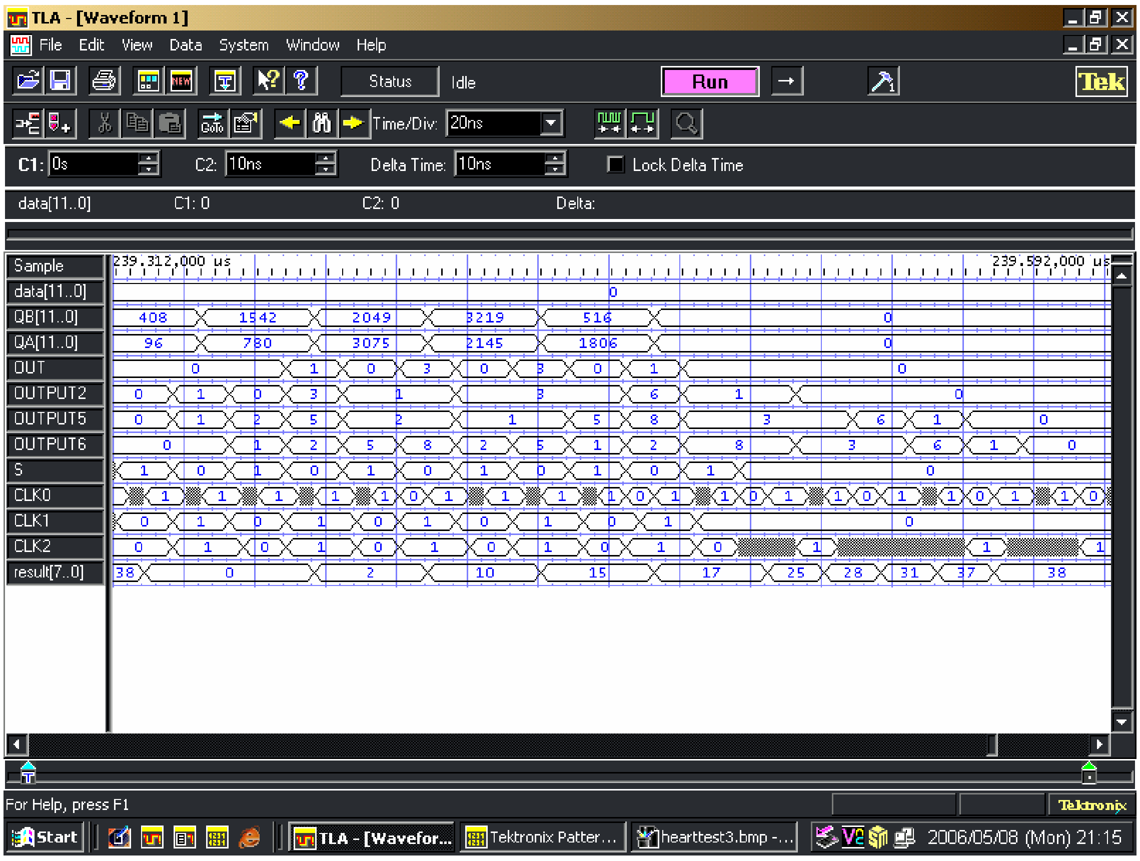


Figure 3-11. FPGA Measurement Results for diagonal projection at 2,5MHz to see the results clearly.

Fig. 3-11 shows the measurement results of the FPGA implementation of the architecture. Sample data in Fig. 3-9 is used to store in cache memory and diagonal projection has been done for the same region given in Fig. 3-10. It is observed that the results match with the expected results from theory. In Fig. 3-11, S shows the control signal for add, shift operations; in this case when S is 1, it means add operation and when S is 0, it means shift operation. CLK2 shows the two phase clocking strategy. During add, shift consecutive operations it takes longer time. However once no more add operation is needed, shift operation can be done in a shorter time. At each rising edge of CLK2, final shift output coming from the final PU can be seen in consistent with the predicted outputs in Fig. 26. Figure 28 shows the same simulation with higher frequency. Since it is possible to see whole flow clearly in this frequency, Figure 3-11 and Figure 3-12 are given for the same data measurement with different frequencies.


TLA - [Waveform 1]

Status Idle

Time/Div: 20ns

C1: 0s C2: 10ns Delta Time: 10ns Lock Delta Time

data[11..0] C1: 0 C2: 0 Delta:

Figure 3-12. FPGA Measurement Results for diagonal projection. Logic Analyzer clock is around 125MHz, minimum possible value. Main clock of the architecture (CLK2) is around 33MHz.

3.5 Proof-of-concept Chip Measurement Results

Designed proof-of-concept chip has arrived from fabrication and under measurement process at the moment. Figure 3-13 shows sample output from memory read-out. At each rising edge of the clock signal, enable command is entered to the memory and address inputs are decoded, then memory value corresponding the given address is read-out. It takes around 1ns in average to read-out one memory row after the clock signal is entered. This means next enable signal can be entered to the system after 1ns passes. In other words, clock signal for the memory can be sent with 1ns period and memory values can be read-out successively. However, at the moment the highest frequency measured for the operation is 60 MHz because of measurement machine settings. The simulation results expect the system work around 500MHz. Measurements for higher frequencies are left for future work. Figure 3-14 illustrates the total time required for generating one vector element after clock signal is entered to the system. In average 2.8ns takes to generate one vector element from the rising edge of the initial clock entering to the memory. This time shows the total time of reading out from the memory, doing add/shift process, selecting the final processing unit and successive addition of incoming values to the final accumulator.   Vector element is the result coming from the output of final accumulator.

Measurements are done at various frequencies. At 5 MHz, the architecture consumes 1,8mW power and at 60MHz it consumes 9mW power and compatible with the expectations of the low-power design strategy followed during design process.

Figure 3-15 and 3-16 shows the measurement results, when all 1's are written to the 16x12-bit memory of the proof-of-concept chip. OUTAA3_0 shows the output of the first processing unit(PU) and OUTDD3_0 shows the output of the fourth PU from left. Fourth PU is selected as the final PU of the interest region and output mask bit is assigned to 1. OUTD11_10,OUTU11_10 show the read-out results of the first two columns of the two adjacent rows of the memory, these are the inputs to the first PU. OUTD5_4, OUTU5_4 show the bits read-out from the corresponding memory column and going into the fourth PU. At each step, sum of the bit values in 2x2 unit, which is 4, is added to the value in each PU and shifted. SUM4_0 shows the output of the selected PU and it is shown that it is same as the output of fourth PU. OUPUT5_0 shows the result of final accumulator. It adds the values coming from SUM4_0 at each rising edge of FINRES signal.

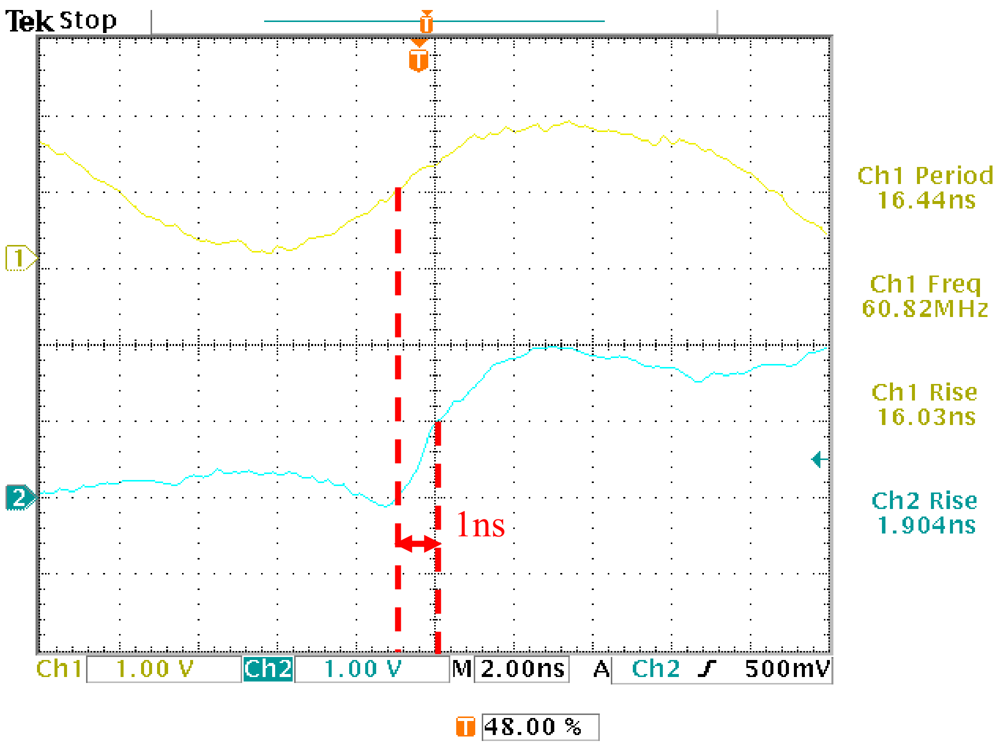


Figure 3-13. Sample memory bit read-out and clock waveform. Edge-flag bit is read-out from the memory with 1ns average delay.

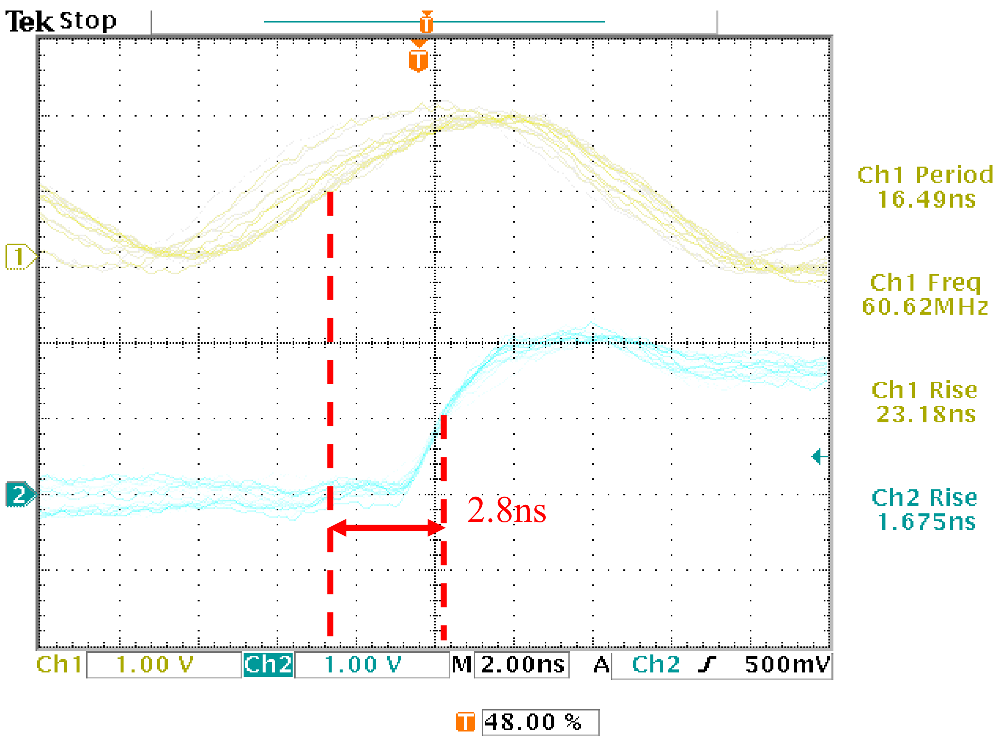


Figure 3-14. Sample vector element bit and clock waveform. It takes about 2.8ns to generate one vector element after the clock signal is inserted to the system.

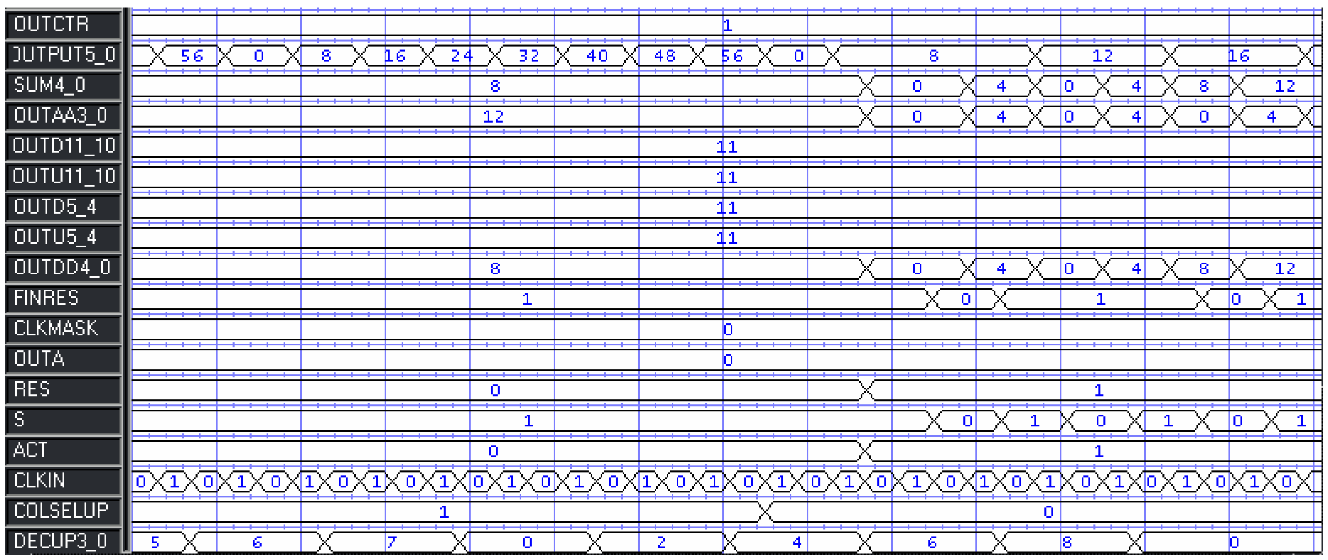


Figure 3-15. Sample measurement results for memory read-out data of 11.

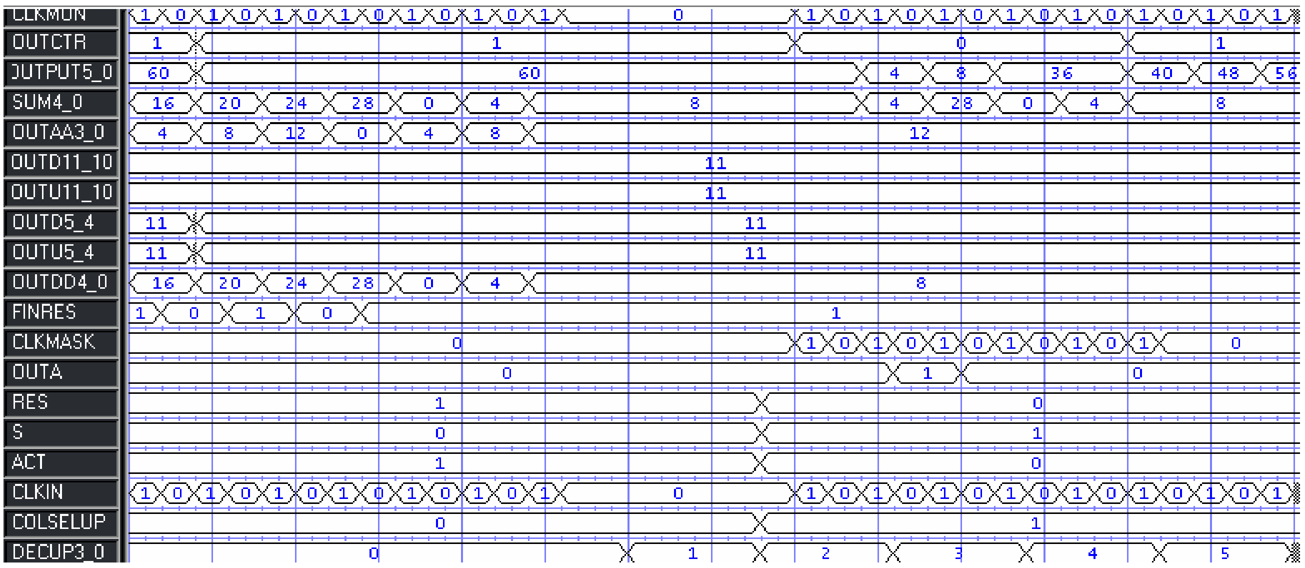


Figure 3-16. Sample measurement results for memory read-out data of 11 continues.

3.6 Summary

For verification and testing of the proposed directional-edge-flag cache memory system, a proof-of-concept chip has been designed. Since the proof-of-concept chip has been delivered recently, measurements are going on for it. The characteristics of the proposed architecture and comparison with the previous systems are demonstrated in Table 3-2 and 3-3. Depending on the projection type, required time to create a feature vector is variable for the proposed architecture. It is most of the time shorter than the conventional works, whereas it may be longer for some cases. Furthermore, with the proposed architecture it is possible to do different kinds of projection with different periodicity of grouping from arbitrary sized regions on demand. However conventional systems are limited by the size of the hardware units. It is also explained in Table 3-3 that, scalability and reusability of the edges are successfully achieved with the new system. During the design of proof-of-concept chip, different circuit styles have been considered for fast low-power operation. By using DRCMOS circuits, 35% speed has gained in decoder, hence in overall speed.

	Conventional Work 1	Conventional Work 2 (64x64 Shift Register Array)	Proposed Architecture	Proposed Architecture Commands
PPED	**4096clk**	**64clk**	32xclk + 31x0.5 clk =47,5 clk (Horizontal/Vertical)	32 add + 31 shift (Horizontal/Vertical)

			32xclk + 31x0.5 clk =47,5 clk(+/-45 degrees) **Parallel Processing 47,5 clk**	32 addshift +31 shift(+/-45 degrees)
CED	**NA**	**64clk**	32xclk + 128x0.5clk **=96 clk**	32 add + 128 shift
EM	**NA**	**NA**	16xclk + 64x0.5 clk **=48 clk**	16 add + 64 shift
Ego-motion	**NA**	**32clk**	32xclk + 32x0.5 clk **=48 clk**	32 add + 32 shift

Table 3-2. Comparison of previous architectures and proposed architecture for vector generation for various kinds of projections. (NA means not applicable.)

	Conventional Work 1	Conventional Work 2	Proposed Architecture
Speed	4096 clk/vector	64 clk/vector	32~96 clk/vector
Scalability (Arbitrary recognition window size)	NO	NO	YES
Variable Masking	NO	YES	YES
Reusability of Edges	NO	NO	YES

Table 3-3. Comparison of conventional works and the proposed system from speed, scalability, variable masking, reusability point of views

Verification of the proposed architecture was also done with FPGA implementation as explained above. FPGA results show the accurate and efficient processing of the proposed architecture with the expected results obtained from the measurement.

Chapter 4

Prototype Chip

4.1 Introduction

After the design of proof-of-concept chip and FPGA implementation, a prototype chip employing larger size cache memory was designed for large areas of scenes. In this work edge-flag bit cache memory of 256x256-bit and vector generation architecture were developed and designed as a VLSI chip. One of the main aims of the proposed vector generation architecture is to be able to process large areas of scenes with high efficiency and high speed. So a VLSI chip was designed for 256x256-pixel input. With this chip arbitrary size feature vectors can be generated robustly in real-time from arbitrary selected regions on demand. Most significant characteristic of the prototype chip is that it is further faster than the previous architectures in some cases. A small changed has been made to the proof-of-concept chip architecture, as a result a simpler architecture and high speed gain have been obtained. In this chapter, general idea and architecture of this prototype chip is presented and the characteristics of this chip are discussed.

4.2 General Idea

In this approach, most of the parts are similar to the approach as designed in proof-of-concept chip. However instead of dealing with 2x2-bit units, 1x4-bit units are considered to group the cache memory architecture. In other words, there is one PU unit corresponding to four adjacent columns attached to the memory as shown in Fig. 4-1. With this approach, we don’t need to design dual-port memory so one-port memory is used. Hence area for cache memory decreases and robustness of memory architecture increases. (One port memory is always more reliable than two-port memory designed in the same manner). Considering PPED and other projection methods, 4 is the largest possible number to group columns appropriate for all kinds of projections and it is most of the time basic unit to construct vector elements. So 4 column grouping is chosen and very useful. Furthermore, when we think about vertical projection and look at the general architecture, it is understood that at the end of successive add operations, each PU will have the sum of corresponding 4 columns, which is a feature vector element without any additional process. Therefore feature vector elements will already be ready for consecutive recognition windows. This will be explained in detail in the following sections.

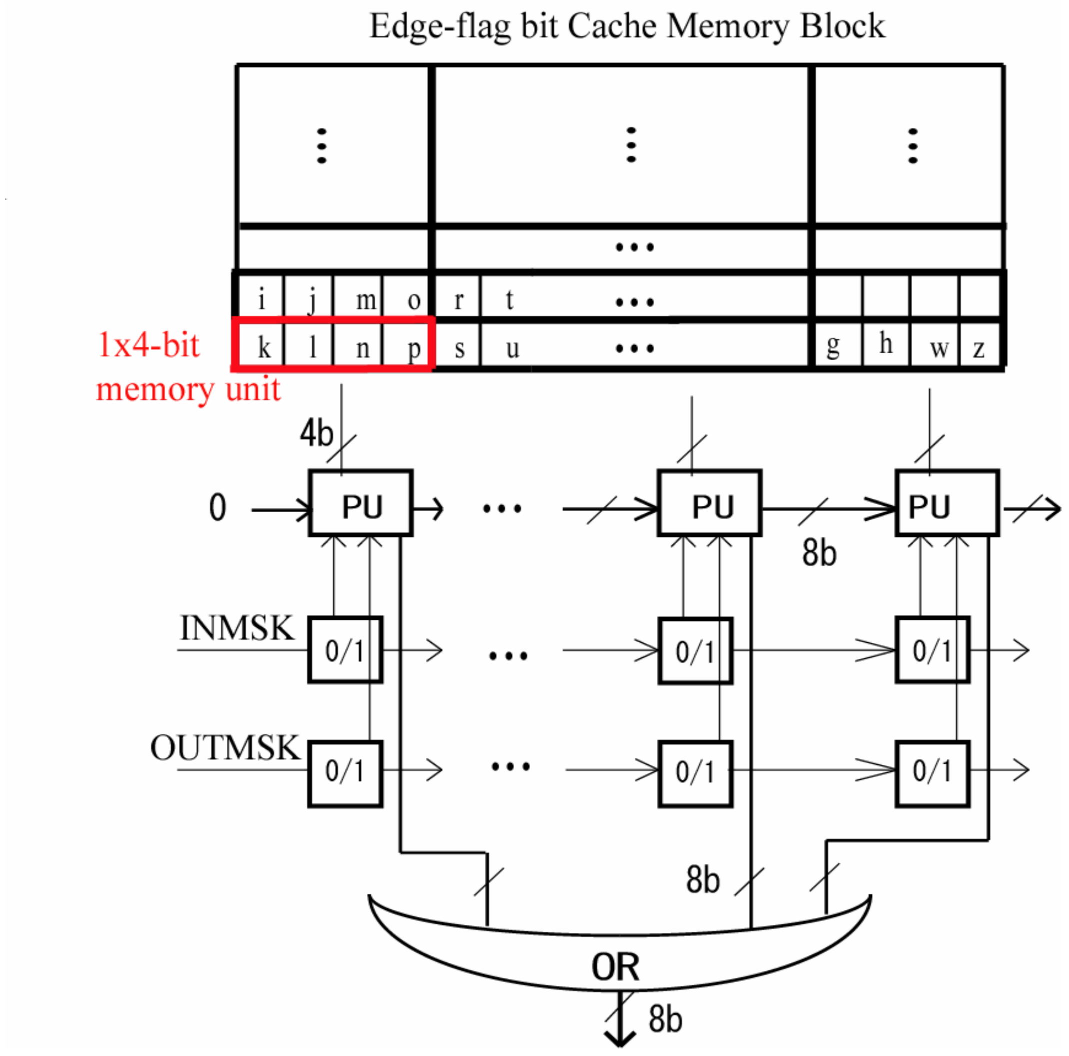


Figure 4-1. 1x4-bit grouping of cache memory elements

4.3 General Structure

In this architecture, 256x256-bit edge-flag bit cache memory is employed. For large memory designs, it is crucial to divide the whole memory into blocks for the sake of less power-consumption and efficiency. Fig. 4-3 illustrates the general architecture. In this architecture, 256x256-bit memory is divided into 8 blocks, each having 256x32-bit. For each block, a processing structure which is composed of 8 processing units, 8 input mask units, 8 output mask units, 8-bit 8-input selector and 8-bit 2-input final accumulator is attached. Later all outputs of the block processing structure goes to a final selector of 8-bit 8-input selector. Each processing unit takes input from corresponding four adjacent columns. 8x256 decoder is used for memory decoding with additional control signals.

In this architecture general flow operation is as follows for a region of MxM recognition window and feature vector generation of size M. First of all, INMSK and OUTMSK control bits are assigned for each PU. M Adjacent 1s are sent to the input control masks of the PUs of the region to be projected, except the starting PU. Starting PU of the region has 0 as input control mask so that it will not receive any shift inputs from previous stages. And only one 1 is sent to the output control mask of the final PU of the region, where the others are all filled with 0s. By this way, interest of region is determined. Then depending on the projection type add and shift operations are held by control signal S as explained in earlier chapters. All the PUs are working at the same time, however only one output is being sent by the final PU to the corresponding accumulator in the processing block. In each processing block there are selectors(SEL) corresponding to each bit of 8-bit data. Each selector applies OR logic to the corresponding bits of the PU outputs in the same processing block. For instance, SEL0 applies OR operation to the 1st bits of the outputs of PUs. However all Pus, except the final one have 0 as output control mask, so they only send 0 to the selector. Then, depending on the desired periodicity of the grouping of the units in the projection, accumulators add the incoming values in groups. For example it adds first two of them, then following two of the inputs then other two of them, dividing into groups of two. As a result, one vector element is made from sum of two adjacent columns or diagonal units in the memory. Again only the accumulator in the processing block which has the final PU, adds values other than 0. The other accumulators are adding zeros only. Finally final selector chooses the output to be sent as vector element from one of the accumulators. Since the accumulators other than the one which has the desired sum are sending all zeros, final selector can easily get the desired sum by just simple OR operation.

Here there are two strategies to store edge-flag bits in the memory. In strategy 1, grouping strategy in prototype chip is kept. The lower two bits of 2x2-bit unit is stored next to the upper two bits in 1x4-bits unit of new architecture. This way, MxM-bit edge-flag bit information is stored in M/2x2M-bit memory part. Other than this, nothing changes about the process of vector generation. In strategy 2, edge-flag bits are kept in the positions as they are in the scene. MxM-bit are stored again in MxM-bit memory part. However vector generation process is changed a little bit. In the second strategy, columns are grouped in 4, so basic sums are addition of 1rowx4 column edge-flag bits. Since 4 is the basic unit for projection methods, it is still very efficient to create feature vectors. Because while add operations increases by 2, required shift operations decreases by 2 also. The details of the required clock cycles are demonstrated for each projection method in Table 4-2. Depending on the projection method and application, edge-flag bits can be placed in any way and desired operation can be achieved.

One big advantage of the strategy 2 is that for vertical projection of PPED, sum of four adjacent columns are already ready in PUs. Independent from region borders, each vector element will be obtained in each PU at the end of only successive add operations. Without shifting the values in PUs, we can shift the output control mask bit and obtain the next vector element in one clk cycle. This will be the only vector element of the next feature vector. Different than the previous one, hence new feature vector can be obtained in one clk cycle as explained in Fig. 4-2 below.

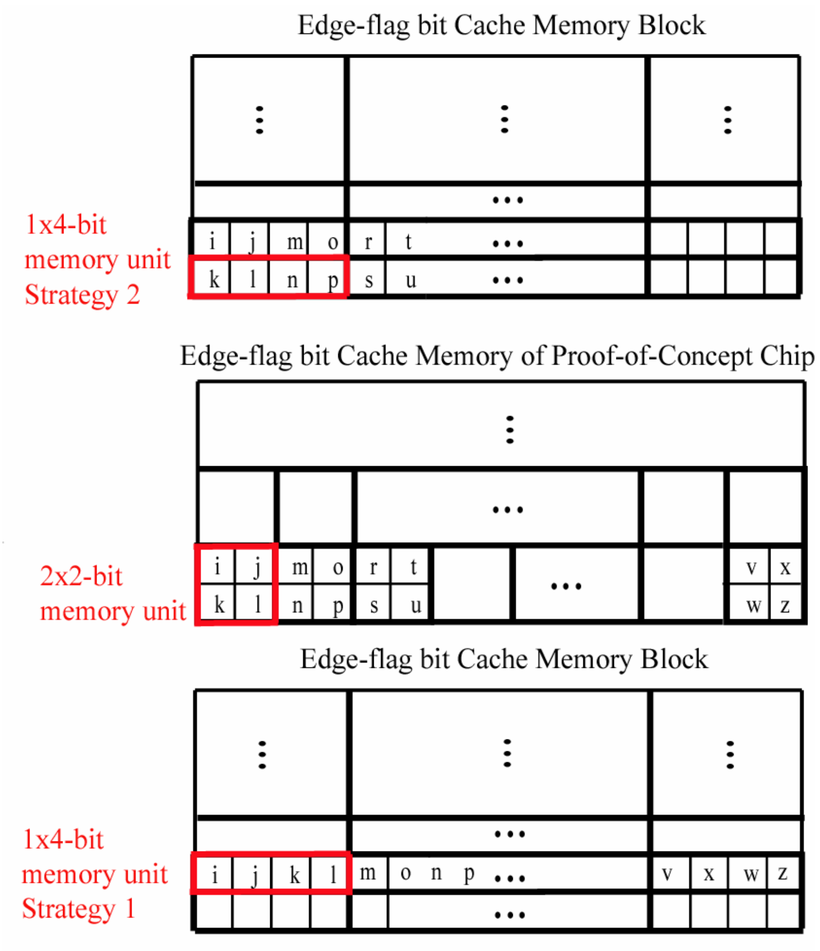

Edge-flag bit Cache Memory Block
1x4-bit memory unit Strategy 2
i j m o r t
k l n p s u
Edge-flag bit Cache Memory of Proof-of-Concept Chip
2x2-bit memory unit
i j m o r t v x
k l n p s u w z
Edge-flag bit Cache Memory Block
1x4-bit memory unit Strategy 1
i j k l m o n p v x w z

Figure 4-2. Different groupings and strategies for cache memory

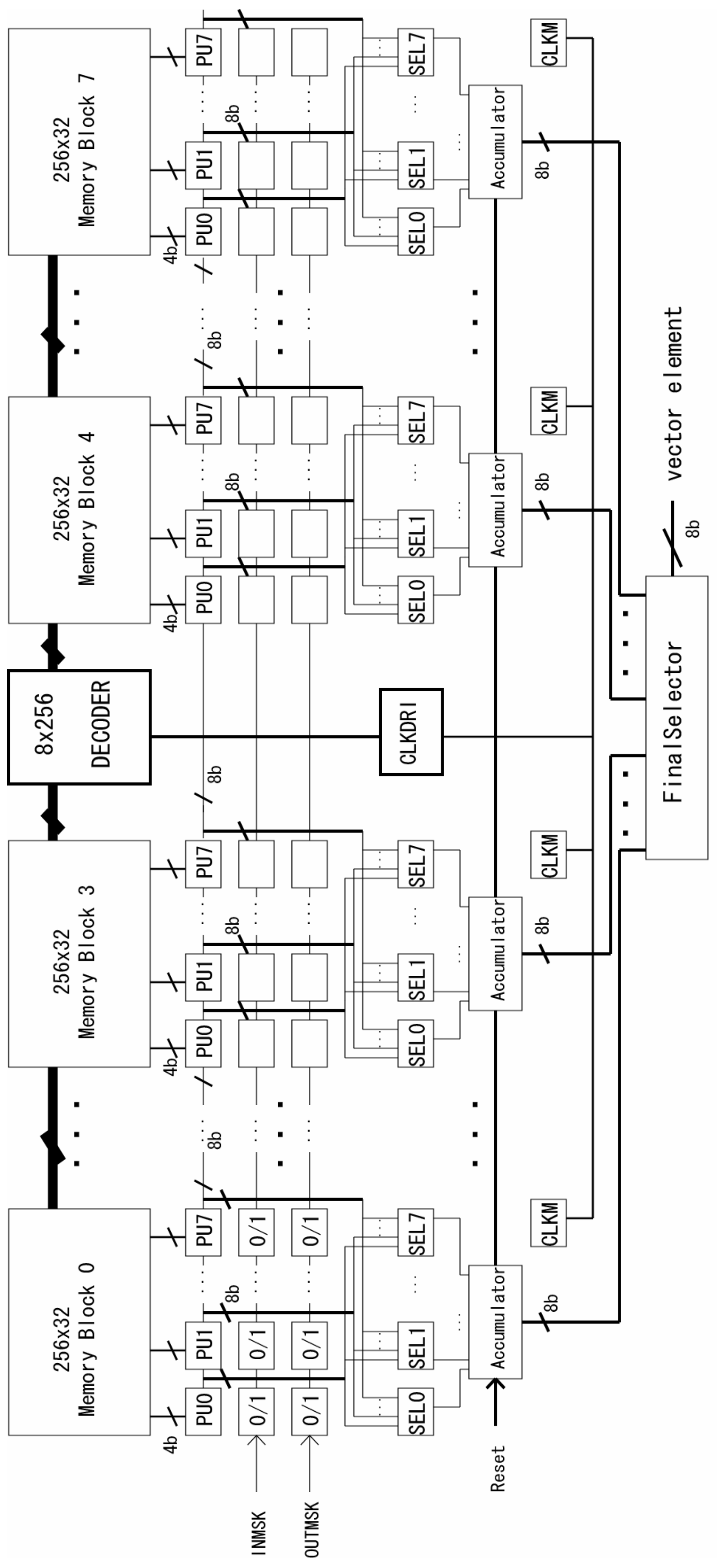


Figure 4-3. General Structure

4.3.1 Memory

256x256-bit cache memory is designed in this chip. Memory parts are designed in the same circuit styles explained for proof-of-concept chip in previous chapter. Since this prototype chip has larger memory size than the proof-of-concept chip, some sizing strategies have been followed to ensure the proper processing of the memory. In some required circuits, sizes of the circuit elements have been increased to compensate for the longer wires, larger capacitance and resistance loads.256 output decoder is designed by combination of 4 6x64 decoders. Separate enable signal is assigned to each decoder. Hence, at each time only one enable signal is active and hence one decoder corresponding to the desired region of operation is active. This way of splitting the decoder will reduce the decoder power consumption and reduce the delay of the decoder, since one step ,2x4 pre-decoding, of decoding is avoided by additional enable signals. 6x64 decoders are implemented by combining the outputs of two 3x8-decoders. For large decoders it has been shown that higher levels of the decode tree should have a fan-in of 2 to minimize the power dissipation as it provides only the smallest number of long decode wires to transition. So 2-input NAND function is implemented at the last stage while combining the 3x8-decoders in 6x64 decoder.

4.3.2 Processing Blocks

The details of the implemented processing block are shown in Fig. 4-4 below. The general structure is almost the same as the processing unit of the proof-of-concept. Only processing elements' sizes are different. Processing elements are designed to do 8-bit input, output operations.

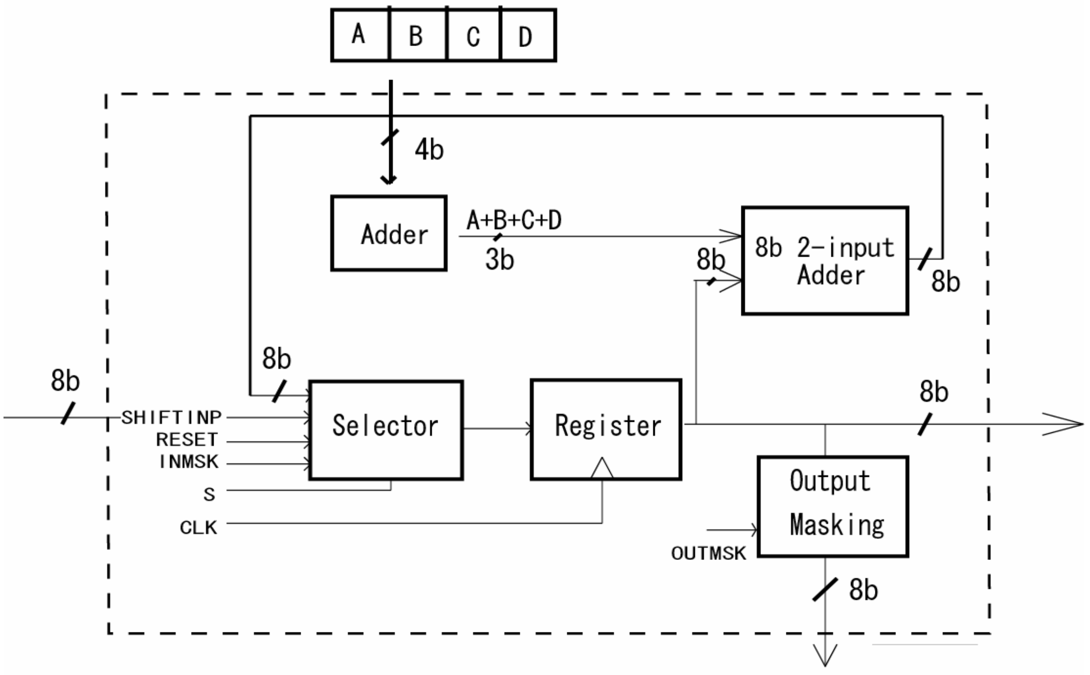


Figure 4-4. Processing unit of prototype chip

As shown in the Fig. 4-4 processing element is receiving its 4-bit input from adjacent columns of the memory.

4.3 Chip Characteristics

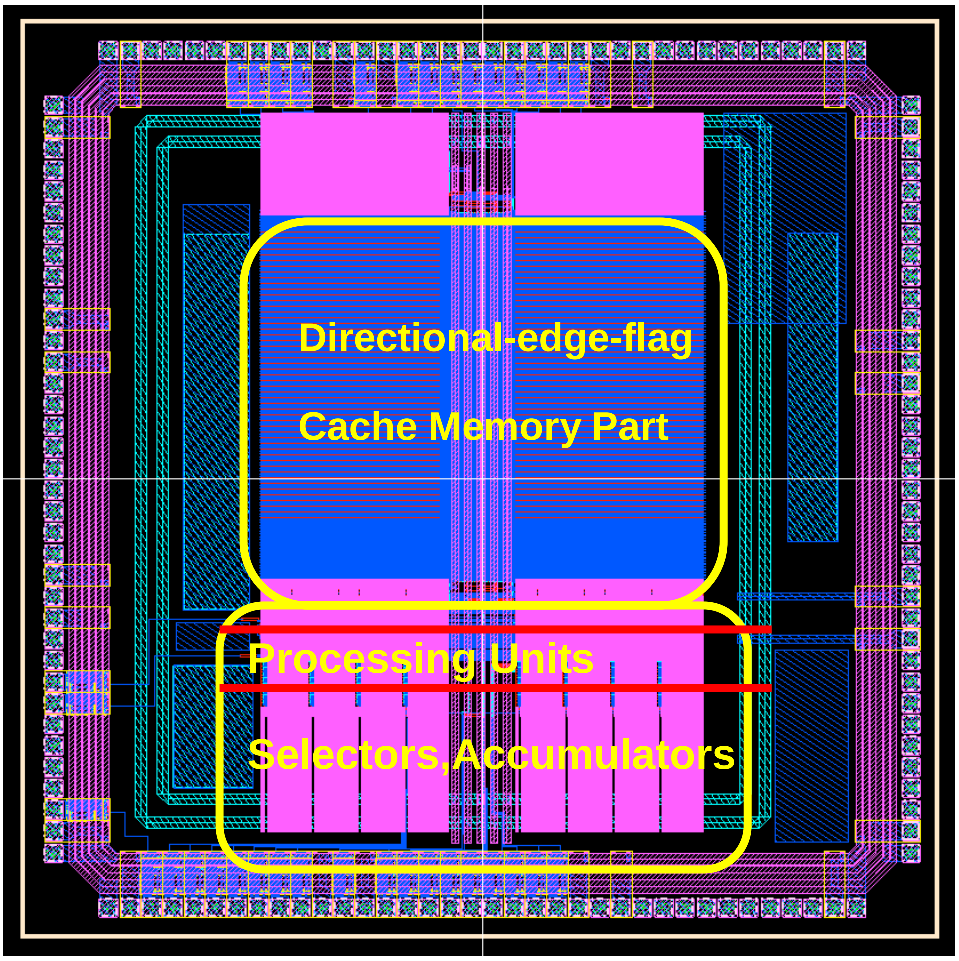


Figure 4-5. Layout of the designed chip.

The proposed architecture employing 256x256-bit cache memory was designed following the design strategy in proof-of-concept chip and considering the feedbacks from proof-of-cencept chip architecture. Chip size is 4,9mmx4,9mm was designed in 0.35μ 3-metal-layer technology. Core has the size 2,8mmx2,4mm and only cache memory part has the size 2,2mmx2,4mm. The simulations have been done with the clock of 200MHz frequency. Layout of the chip is demonstrated in Fig. 4-5. Diagonal-edge-flag cache memory is placed in the upper part of the area. Processing units, selectors, accumulators are placed in the lower part. Each processing unit is laid under column groups of 4. Processing units are clarified with red lines in the figure. It is observed that, PUs take a very small area compared to the other parts. Selectors and accumulators are also in a very small area, most of the area in the lower part is occupied by connecting lines of OR circuitries.

Clocking Strategy

Prototype chip works with in the same two-phase clocking strategy of proof-0f-concept chip.

4.4 Summary

A large-scale prototype chip was designed and sent to fabrication. Major structures are kept same as in the proof-of-concept chip. With this prototype-chip it is possible to process 256x256-pixel images. Feature vectors can be generated from arbitrary regions of this 256x256-pixel image with various projection types on demand. Since the technology is 3-metal technology, it is possible to design much faster and smaller chips of the same architecture with more advanced technologies. Table 4-1 illustrates the prototype chip characteristics. In Table 4-2 comparison of the conventional work and this work is given for different kinds of projections and for two different kinds of strategies explained in previous section.

Technology	CMOS,0.35mm,3-metal layer
Supply Voltage	3.3V
Die Size	4.9mm x 4.9mm (2.8mmx2.4mm)
Cache Memory Size(mm)	2.2mmx2.4mm
Cache Memory Size	256x256-bits
Operation Frequency	200MHz(HSPICE Simulation)

Table 4-1. Characteristics of the Prototype Chip

	Conventional Work 1	Conventional Work 2	Prototype Chip Architecture	Prototype Chip Architecture
PPED (Horizontal/ Vertical/ +/-45 degrees)	**4096clk**	**64clk**	32xclk + 31x0.5 clk =47.5 clk (Horizontal/Vertical) 32xclk + 31x0.5 clk =47.5 clk (+/- 45 degrees)	32 add + 31 shift (Horizontal/Vertical) 32 addshift +31 shift (+/- 45 degrees)

			Parallel Processing 47,5 clk	
CED Strategy1	**NA**	**64clk**	32xclk + 128x0.5clk **=96 clk**	32 add + 128 shift
CED Strategy2	**NA**	**64clk**	64xclk + 64x0.5clk **=96 clk**	64 add + 64 shift
EM Strategy1	**NA**	**NA**	16xclk + 64x0.5 clk **=48 clk**	16 add + 64 shift
EM Strategy2	**NA**	**NA**	16xclk + 32x0.5 clk **=32 clk**	16 add +32 shift
Ego-motion	**NA**	**32clk**	32xclk + 32x0.5 clk **=48 clk**	32 add + 32 shift

NA: not applicable

Table 4-2. Comparison of different projection types for different architectures

	Conventional Work 1	Conventional Work 2 (100MHz, 0.18u technology)	Conventional Work 2 (100MHz, 0.18u technology)	Prototype Chip Architecture (200MHz, 0.35u technology)	Prototype Chip Architecture (200MHz, 0.35u technology)
Speed	4096 clk/vector	64 clk/vector	640ns /vector	32~96 clk/vector	160ns~480ns /vector
Scalability (Arbitrary recognition window size)	NO	NO	NO	YES	YES
Variable Masking	-	YES	YES	YES	YES
Reusability of Edges	NO	NO	NO	YES	YES

Table 4-3. Comparison of conventional work and the proposed architecture in terms of speed, scalability, reusability of edges, variability of masking…

Chapter 5

Conclusions and Future Work

In this thesis a new architecture is proposed for vector generation of early visual processing stage of image recognition system. With the proposed architecture, faster generation of feature vectors is possible from arbitrary regions of large scenes on demand. This architecture also removes the hardware limitations of the conventional structures and brings a flexible approach for feature vector generation. The most important characteristic of the new architecture is that it stores the extracted directional-edge-flag bits of the whole scene in a cache memory, providing the reusability of edge-flag bits at anytime without the need of reprocessing of high-cost operations. Besides it brings scalability of recognition windows in the scene, it is possible to generate feature vectors of any size from arbitrary regions. Also with this new architecture the coordination of different parts of regions can be easily made possible for further development of the recognition system. Consequently over-all performance of the recognition system has been enhanced further for large area of scenes and different kinds of projections and recognition window sizes have been made possible to apply.

However, in the proposed edge-cache memory architecture there is some waste of operation and power, while all the PUs are working at the same time. Since the architecture is divided into memory and processing blocks, it may be possible to turn off the unnecessary blocks by just a few control signals. From now on, first of all it is planned to concentrate on the detailed on going measurements of the proof-of-concept chip and prototype chip. And then research will go on with a more efficient high-speed low-power system architecture design. To achieve real-time responding recognition system, it is important to force the limits of speed of processing. At the moment very fast processing of vertical projection is possible; however diagonal projections are still limited. As a result, measurements and further improvement of the architecture are waiting for us on the way and the research will go on in this direction.

List of Publications

Ovgu Ozturk, and Tadashi Shibata, "An Edge Cache Memory Architecture for Early Processing VLSIs", to be presented in 2006 International Conference on Solid State Devices and Materials (SSDM 2006), Yokohama, Japan, September 2006

References

[1] T. Shibata, "Intelligent Signal Processing Based on a Psychologically-Inspired VLSI Brain Model," IEICE Trans. Fundamentals, Vol. E85-A, No. 3, pp. 600-609, 2002.

[2] M. Yagi, M Adachi and T. Shibata, "A Hardware-Friendly Soft-Computing Algorithm for Image Recognition," Proc. of 10th European Signal Processing Conference (EUSIPCO 2000), pp. 729-732, Sep. 2000.

[3] M. Yagi and T. Shibata, "An associative-processor-based mixed signal system for robust image recognition", in Proc. of 2002 International Symposium On Circuits and Systems (ISCAS 2002), pp. V-137-V-140, May 2002.

[4] M. Yagi and T. Shibata, "An Image Representation Algorithm Compatible With Neural-Associative-Processor-Based Hardware Recognition Systems," *IEEE Trans. Neural Networks*, vol. 14, no. 5, pp.1144-1161, Sep. 2003.

[5] M. Yagi, H. Yamasaki, and T. Shibata, "A Mixed-Signal VLSI for Real-Time Generation of Edge-Based Image Vectors," to be published in Advances in Neural Information Processing Systems 16, Dec. 2003.

[6] K. Ito, M. Ogawa, and T. Shibata, "A Variable-Kernel Flash-Convolution Image Filtering Processor," in IEEE International Solid-State-Circuit Conference Dig. Tech. Papers, No. 26.7, pp. 470-471, Feb. 2003.

[7] T. Yamasaki and T. Shibata, "Analog Soft-Pattern-Matching Classifier Using Floating-Gate MOS Technology," IEEE Trans. Neural Networks, vol. 14, no. 5, pp. 1257-1265, Sep. 2003.

[8] H. Yamasaki and T. Shibata, "A Real-Time VLSI Median Filter Employing Two-Dimensional Bit-Propagating Architecture," Proc. ISCAS, pp. II-349-352, May 2004.

[9] H. Yamasaki and T. Shibata, "A Real-Time Image-Feature-Extraction and Vector-Generation VLSI Employing Arrayed-Shift-Register Architecture," in the Proceeding of the 31th European Solid-State Circuit Conference (ESSCIRC), Grenoble, France, pp.121-124, Sep. 2005.

[10] H. Yamasaki and T. Shibata, "A High-Speed Median Filter VLSI Using Floating-Gate-MOS-Based Low-Power Major Voting Circuits," in the Proceeding of the 31th European Solid-State Circuit Conference (ESSCIRC), Grenoble, France, pp.125-128, Sep. 2005.

[11] Bharadwaj S. Amrutur, "Design and Analysis of Fast Low Power Systems", PhD Thesis, August 1999, Standford Univeristy

[12] Jeyran Hezavei, N. Vijaykrishnan, M. J. Irvin, "A Comparative Study of Power Efficient SRAM Designs", GLSVLSI 2000 Evanston Illinois USA.

[13]M. B. Kamble, K. Ghose, "Energy Efficiency of VLSI Caches: A Comparative Study", Prec. IEEE International Conference on VLSI Design, pp. 261-267, 1997.

[14]Bharadwaj S. Amrutur , Mark A.Horowitz, "Speed and Power Scaling of SRAM's", IEEE

Transactions on Solid State Circuits, Vol. 35, No. 2, February 2000.

[15]Evert Seevinck, Frans J. List, Jan Lohstroh, “Static-Noise Margin Analysis of MOS SRAM Cells”, IEEE Journal of Solid-State Circuits, VOL. SC-22, NO.5, October 1987.

[16]Azeez J.Bhavnagarwala, Xinghai Tang, James D. Meindl, “The Impact of Intrinsic Device Fluctuations on CMOS SRAM Cell Stability”, IEEE Journal of Solid-State Circuits, VOL. 36, NO.4, April 2001.

[17] Y. Suzuki and T. Shibata, “Multiple-clue face detection algorithm using edge-based feature vectors,” in *Proc.ICASSP 2004*, Montreal, Canada, May 17-21. 2004,vol. 5, pp. 737–740.

[18] Y. Suzuki and T. Shibata, “An edge-based face detection algorithm robust against illumination, focus, and scale variations,” in *Proc. EUSIPCO 2004*, Vienna, Austria, September 6-10. 2004, pp. 2279–2282.

[19]Y. Suzuki and T. Shibata, “Multiple-resolution edge-based feature representation for Robust Face Segmentation and Verification” in Proc. EUSIPCO, Antalya, Turkey, Sept. 5-9. 2005.

[20] Jia Hao and T.Shibata, “A VLSI-Implementation-Friendly Ego-Motion Detection Algorithm Based on Edge-Histogram Matching”, Toulouse, France, May 14-19, 2006.

[20] T. Chappell, et. al., “A 2-ns Cycle, 3.8-ns Access 512-Kb CMOS ECL SRAM with a Fully Pipelined Architecture”, *IEEE Journal of Solid State Circuits,* vol. 26, no. 11, pp. 1577-1585, Nov. 1991.

[21] H. Nambu, et. al., “A 1.8ns Access, 550MHz 4.5Mb CMOS SRAM”, *1998 IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 360-361.

[22] G. Braceras, et. al., “A 350MHz 3.3V 4Mb SRAM fabricated in a 0.3□m CMOS process”, 1*997 IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 404-405.

[23] K. Nakamura, et. al., “A 500MHz 4Mb CMOS pipeline-burst cache SRAM with point-to-point noise reduction coding I/O”, 1*997 IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 406-407.

[24] K. Sasaki, et. al., “A 15-ns 1-Mbit CMOS SRAM”, *IEEE Journal of Solid State Circuits,* vol. 23, no. 5, pp. 1067-1071, October 1988.

[25] O. Minato, et. al., “A 20ns 64K CMOS SRAM”, *1984 IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 222-223.

[26] S. Yamamoto, et. al., “256k CMOS SRAM With Variable Impedance Data-Line Loads”, *IEEE Journal of Solid State Circuits,* vol. SC-20, pp. 924-928, October 1985.

[27] K. J. Schultz, et. al., “Low-Supply-Noise Low-Power Embedded Modular SRAM for Mixed Analog-Digital ICs”, *Proceedings of 1992 IEEE Custom Integrated Circuits Conference,* p. 7.1/1-4.

[28] P. Reed, et. al., “A 250MHz 5W RISC Microprocessor with On-Chip L2 Cache Controller”, *1997 IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 412-413.

[29] J. M. Rabaey, Digital Integrated Circuits a Design Perspective, Prentice Hall, 1996.