VLSI DESIGN OF A TESTABLE PROCESSOR ARRAY

FOR FEATURE EXTRACTION

by

PATRICE P. AUBRY
B.Sc., University of Victoria, 1986

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
in the Department of
Electrical and Computer Engineering

We accept this thesis as conforming
to the required standard

[redacted]

Supervisor Dr. F. El Guibaly

[redacted]

Dr. V. K. Bhargava

[redacted]

Dr. G. C. Shoja

[redacted]

Dr. W. Little

[redacted]

Dr. M. Serra

ISBN 0-315-46518-2

Supervisor: Professor F. El Guibaly

ABSTRACT

A VLSI CMOS design performing feature extraction for machine and hand-written printed character recognition is described. The chip is designed to efficiently process a character in the form of a two-dimensional binary image. Although the algorithm for the character recognition and its architectural configuration are presented, the processor array necessary for feature extraction is dealt with in detail. Built-In-Self-Test (BIST) is incorporated in each unit cell to enable exhaustive testing. Fault tolerance is allowed for scattered faulty cells. The array is considered faulty if clustered faulty cells are detected. Simulation results and hardware details are presented. Finally, future considerations and applications are discussed.

Examiners:

[redacted]

Supervisor Dr. F. El Guibaly

[redacted]

Dr. V. K. Bhargava

[redacted]

Dr. G. C. Shoja

[redacted]

Dr. W. Little

[redacted]

Dr. M. Serra

Table of Contents

List of Figures

Acknowledgements

I would like to thank Dr. Fayez El Guibaly for his guidance during the course of this project and for his helpful advice during the preparation of this manuscript. Financial assistance received from Dr. F. El Guibaly and the University of Victoria was greatly appreciated.

I would also like to thank Dr. Micaela Serra of Computer Science at Uvic, for her help in the fault simulation of the design.

To my Wife,
Jeanette
for her encouragement and support

Chapter 1

Introduction

Character recognition is a subset of the pattern recognition field. Much work has been done on character recognition, with the recognition of handwritten characters remaining as the most difficult due to the extent of the character set. Recognition can be simplified by constraining the character set (fixed-font) as with magnetic and mechanical scanners, which makes these recognizers very reliable. Magnetic scanners produce a waveform from a single scan. Recognition consists of correlating the scanned waveform with stored waveforms. Mechanical scanners use tactile or surface relief methods to identify a character.

Complexity is augmented in multi-font reading, due to the larger character set and the presence of noise due to varying ink quality and paper whiteness. However, the set of fonts is finite, limiting the variability of the character shapes. Template matching has been widely used for multi-font recognition since the character set is small enough to be stored in memory.

The greatest complexity is present in the recognition of handwritten characters. Here, the noise problem is magnified because, in addition to the noise encountered in multi-font reading, the variability of a person's writing is present. There is, essentially, an infinite set of possible fonts. Thus, more sophisticated methods than template matching must be used.

The recognition process is typically broken into several stages. To start, a character is optically scanned. The output is either a grey scale or a binary (bit map) image. Next, preprocessing is performed in the form of thresholding, location and isolation of the character, as well as noise removal. Information, consisting of various features, is then extracted from the character. This is followed by classification, usually by comparison with a preacquired knowledge base. Finally, a decision is made, identifying the character. The details of an optical character recognition (OCR) system will be presented in the subsequent sections of this Chapter.

Since the character is represented in matrix form, the number of pixels to be processed is large. For example with a scanner resolution of 300 pixels/inch, more than 4 Megapixels must be processed for an average 8.5 in. by 11 in. document. Depending on the preprocessing required and the method used for extraction of information, the recognition speed can be slow. Existing software systems for multi-fonts tend to use less sophisticated recognition techniques to minimize computer time. Research of available literature showed that only one OCR system for handwritten

characters [1]. This system however operates on the basis of constrained handwriting which requires the user to enter characters in a prescribed manner to minimize variations in character shapes.

This thesis is an attempt at realizing unconstrained handwritten character recognition in hardware by designing a fast feature extraction chip. The objective is to develop special-purpose testable hardware capable of achieving high throughput rates. This work is only concerned with the recognition of the handwritten, unconnected English character set. The chip is capable of high processing rates by using a parallel VLSI architecture for feature extraction. Further, the chip incorporates a self-testing technique using signature analysis. The speed-up of the process allows the addition of preprocessing steps and character analyses to enhance correct recognition. These additional steps could be implemented using the VLSI design for the feature extraction stage, with simple modifications.

The entire system would be composed of VLSI chips for the preprocessing and feature extraction stages, and a fast "off-the-shelf" chip for the recognition and learning stages. The learning capability is incorporated to enable the recognition of different individuals' handwriting, as well as new fonts. The system would also work as an omni-font OCR, since this is a subset of unconstrained handwritten characters.

Chapter 2 describes the algorithm used for the recognition process. Sources of noise are defined, and the algorithm's robustness is estimated.

The simulation and evaluation of the algorithm are also presented. Chapter 3 discusses VLSI design methodology. Included here are the mapping of the algorithm to hardware, design methodology and design for testability. Chapter 4 presents the implementation of the processor array. The architectural requirements are discussed, as well as circuit techniques. The built-in self-test method (BIST) is presented in detail. The hardware testing, evaluation and gate level representation are also discussed. In chapter 5, fault and hardware simulation results are given. Finally, chapter 6 presents the summary and future considerations. Applications of the system are also considered. The appendix outlines well known theorems with proofs regarding the error detection properties of a linear feedback shift register.

In this chapter, we present an overview of character recognition, the recognition methods and contemporary optical character recognizers (OCR's). This is followed by a description of the relation between VLSI design issues and methodologies and the requirements of character recognition (CR).

1.1 Optical Character Recognition

Optical character recognition can be subdivided into two types: on-line and off-line [2]. On-line CR is performed as the character is being drawn. This method requires a graphics tablet or a light pen actively linked to a computer. Recognition is simplified since stroke sequences as well as geometric information are available. The knowledge of stroke sequences

removes some of the constraints on size and orientation of a character.

In contrast, off-line systems recognize the characters after they have been drawn. Off-line CR can be categorized as follows:

1. *Fixed Font Character Recognition* for the recognition of standard fonts of machine printed characters.

2. *Handwritten Character Recognition* for the recognition of printed and unconnected handwritten characters. The segmentation process is a simple task since the character are separated.

3. *Script Character Recognition* is similar to handwritten recognition but for cursive and connected characters. Preprocessing, segmentation and classification are most difficult to achieve, considering that humans have difficulty reading cursive script. High input resolution is required and processing rates are usually slow. Further, large data bases are necessary. Cursive script recognition will benefit from custom VLSI hardware to improve throughput. This area of research is closely related to that of speech processing, as work is being done to recognize words or syllables rather than single characters.

The general components of a typical character recognition system are illustrated in Fig.1.1.

Below we explain those main components.

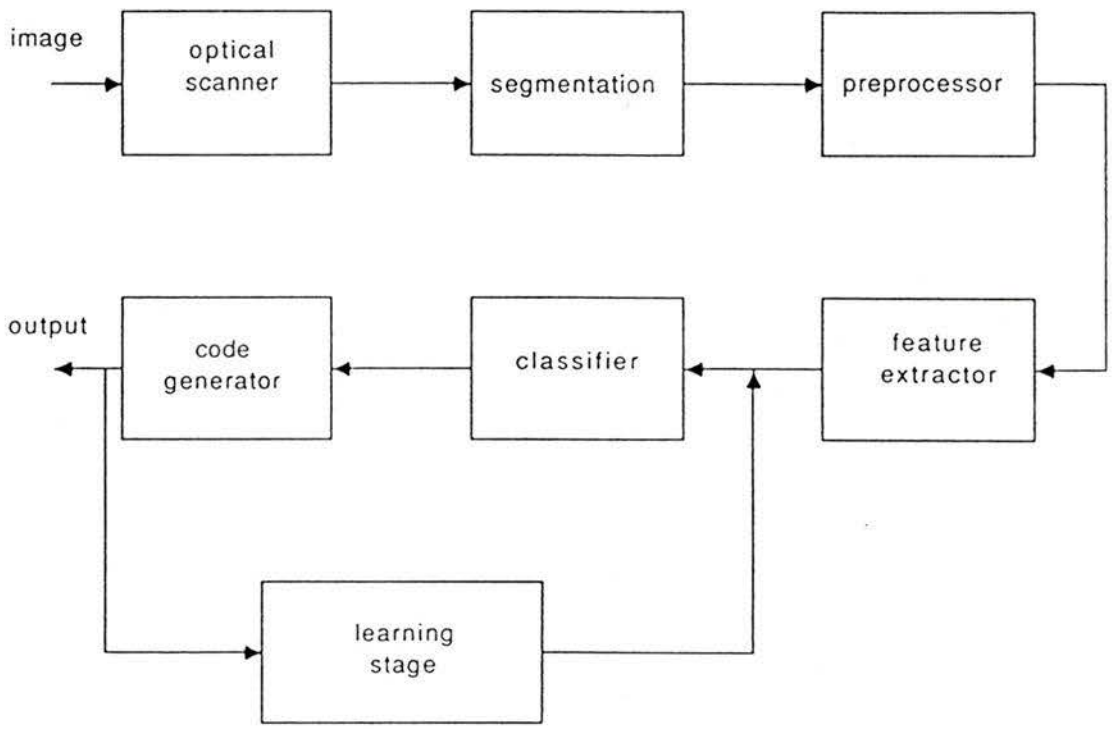


Figure 1.1: Typical optical character recognition system

1.1.1 Scanning Methods

The first stage in any OCR system is the optical scanner, which digitizes the printed character to allow processing and recognition. Text presented to the scanner is in the form of a grey-scale image. The scanner digitizes this image into a binary matrix whose size depends on the resolution of the scanner. Common scanning methods are:

1. *Divided Slit Scanner*: A linear array of photocells scan the document; many linear scans are necessary to cover the vertical extent of a character.

2. *Mechanical Scanner*: A disk (Nipkow disk) with a spiral set of holes allows focused light, via a slit, from a character to be detected by a photomultiplier. As the disk turns and the character is scanned horizontally, a complete vertical raster sequence of the character is generated.

3. *Vidicon Scanner*: A tube developed for television cameras, is used to scan a single printed line at a time.

4. *Flying Spot Scanner*: A single spot of light from a cathode-ray tube is used to to illuminate the document. The document is stationary and information is obtained from the brightness of the reflected light. The advantage here is that the document can be selectively scanned

(e.g. a document containing both graphics and print).

5. *Laser Beam Scanner*: This is an advancement on the above method, making use of the coherent property of the laser light. Advantages over the flying spot scanner are that the spot brightness is constant and it will not contain ultraviolet components which can cause unwanted fluorescence.

6. *Photodetector Array*: A two-dimensional array of photocells is employed to scan a complete character. This method is the most popular in industry.

The highest scanning speeds are obtained with photocell matrix, laser beam or divided slit scanner. Scanning speeds can be greater than 3000 characters/sec, which is the rate used as a criterion in this research.

1.1.2 Segmentation

This step isolates single characters from a line of text, assuming lines of text have been isolated in a document, which has been previously divided into text and graphics [3,4]. The process of segmentation varies in complexity depending on font and handwriting styles. If the characters are not connected and clearly separated, then the detection of a blank vertical column determines the separation between two characters. There are two

more possibilities (the case of connected letters as in regular handwriting is not addressed here). First, adjacent characters can touch or overlap each other. Touching characters can be divided by knowing the approximate character width, or by using various histogram techniques which examine the projections of the character. Secondly, for overlapping characters such as italics, separation can be achieved by contour or "snake" segmentation which essentially tries to find a path between two characters [5,20].

1.1.3 Preprocessing

This stage consists of preparing the input character for the recognition process. A single character is contained in a $r \times r$ matrix. This matrix might first be compressed, to match the dimensions of a VLSI array or of stored character templates. A common method is to subdivide the matrix to be compressed into a grid of 3×3 cells. The value of the new central pixel in each cell will be determined by some appropriate method such as averaging or the median of the window. Below we discuss some of the common preprocessing steps.

Binarization is the process in which a grey-scale is transformed into an image consisting of 1's and 0's, called a bit map. This simplifies the recognition process by reducing numerical calculations to logical ones. It is necessary not only for ease of processing, but also because of detrimental effects of sloppy printing, different whiteness levels of document paper and

general document handling.

Binarization is achieved through thresholding. Given a grey-level pattern P, a bit map is obtained by

$$p_{ij} = \begin{cases} 0 & p_{ij} < T \\ 1 & p_{ij} \geq T \end{cases} \qquad \forall\, p_{ij} \in P$$

where T is the threshold or cut-off point which must be chosen in such a way so as not to degrade the resulting image. Connectivity of the character must remain intact, since subsequent preprocessing steps could change the shape of a character (e.g. G to C). One solution is to scan the grey-level matrix with a 3×3 window, such that the threshold is determined from the 8 pixels surrounding the central pixel by

$$T_{ij} = \frac{1}{8}\left(\sum_{l=-1}^{1}\sum_{k=-1}^{1} p_{i+k,j+l} - p_{ij}\right)$$

T_{ij} is simply the average of the nearest-neighbors and is a function of position. This method resembles that of the human eye [5]. Similar methods would use median or mode values.

A suitable threshold can also be determined by assuming that the intensity distribution is bimodal. This implies that a histogram of the image (number of pixels versus intensity) will have two peaks. If the assumption is valid, the peaks should be clearly separated. The threshold value would be the intensity corresponding to the lowest point in the valley between the peaks. Other methods can be found in [7].

Scanners can also be set-up to perform the threshold operation directly, thus eliminating this preprocessing step. The quality of ink and the uniformity of paper whiteness are some of the factors which will indicate the most suitable approach. Although thresholding does remove some spurious noise, further processing will enhance the recognition process.

Smoothing reduces breaks (or voids) in the outline of a character. It also removes spurious elements. A common method is to scan the binary image with an N x N window. The value of the central element is based on the count of 1's within the window. The size of the window and the threshold (count) are empirically determined based on resolution and font size. This is called smoothing by averaging. Other techniques include median filtering, min/max and the use of logical rules on the window patterns [7].

Thinning is the process by which thick lines of a character are reduced to thin lines (one pixel wide). The skeleton obtained after thinning is called the medial line of the character. It is also helpful in reducing spurious noise. Again windowing is applied repeatedly until no more pixels change value [8]. Thinning should preserve the end points and T-junctions of the character.

Edge detection can be used to determine the location, bounding rectangle and shape of a character. *Alignment* and *normalization* methods such as perspective transformation will transform a skewed or slanted character into a standard (undistorted) character [5].

Windowing techniques used in most preprocessing steps are suitable for hardware implementation. 3×3 windowing requires nearest-neighbor communication only, allowing the preprocessing to be performed concurrently throughout the array.

1.2 Recognition Algorithms

Pattern recognition deals with the classification of patterns into classes. Patterns fall into classes based on their similarities called features. Feature extraction reduces the dimensionality of the pattern space to that of the feature space. This demands that the features be well selected, in the sense that an additional extraction of a feature will improve recognition. Thus feature extraction is necessary for classification and recognition, and is used in all recognition techniques. The only exception would be that of point-by-point comparison. This is equivalent to *template matching*, where the degree of correspondence between each pixel of the image and that of the stored reference is extracted. This is really a special case of global feature extraction. For large patterns, this technique is not economical. Dimensionality is often reduced by comparing only fragments or "hot spots" of the pattern. The template which meets the matching criteria better than the others is the recognized pattern. This method is not appropriate for handwriting recognition because of the variations in an individual's handwriting.

Another global technique is that of *global transformation.* Features which are invariant under certain transformations are extracted by transforming the image. The standard transformations are the Hough and Fourier transforms. Another is the Mellin transform which is both scale and rotation invariant [9]. These methods are computationally demanding, and not always applicable (e.g. Fourier transforms of some letters resemble each other too closely) [5].

Dimensionality reduction can also be achieved by extracting features which describe distributions of points. The method of *Moments* uses a mechanical analog to describe the moments of 1's about some center (e.g. center of gravity). *Zoning* uses densities of 1's in regions of the pattern as features [5,10]. Most of these methods require supplementary techniques, and can be difficult to implement.

Another category of feature extraction generates features which contain information about the geometry and topology of the character. This category can be partitioned into *decision-theoretic* and *syntactic methods* [11]. In the *decision-theoretic approach*, characteristic geometric features are extracted from the patterns for classification. Recognition of each pattern is usually made by partitioning the feature space. The features include presence of lines and loops, a profile, size and location. Most of the early work on character recognition was performed (until '70's) using this method. On the other hand, the *syntactic approach* extracts information about the

structural aspects of the pattern for recognition. Description of the character is done using a graph relating line junctions and shapes. This method is usually employed where the pattern is complex and the number of features is very large (e.g. chinese characters), thus making the hierarchical comparison of subpatterns (primitives) attractive. Since the classification of varied and complex patterns is not practical, recognition is satisfied by description. It should be noted that there is no clear distinction between these two approaches. Both of these techniques require preprocessing such as segmentation, thresholding and thinning. Further, these two techniques can be used together. For example, in recognition of sentences, the syntactic method would be used in determining sequences of words, whereas the decision-theoretic technique would be more appropriate for the recognition of individual letters. Thus these two techniques appear as complementary to each other. Primitives, used in structural methods, are sensitive to noise and distortion since they are defined in terms of local properties of a pattern. Thus, the decision-theoretic approach is better suited for handwritten character recognition. Chapter 2 will discuss in more detail the nature of this noise. Both these methods are well suited to VLSI implementation and are fairly easy to map onto hardware parallel architectures [11].

1.3 VLSI Design Issues

Pattern recognition is demanding due to the large amount of data to be processed. Typically, image processing operations must be performed at high speeds. It has been demonstrated that VLSI architectures are well suited to implement image processing algorithms [12,13].

Gains in performance are a result of higher densities (faster communication) and swiching speeds. However, the development of novel architectures and algorithms which utilize concurrency are responsible for most recent advances in image processing. This last statement implies a close relationship between computational algorithms and VLSI architectures. A restricted, but powerful example, is DIASTOL [14] which maps uniform recurrence equations [15] (e.g. convolution, matrix multiplication) onto a family of systolic arrays. It should be emphasized that the speed of VLSI components is limited, such that advancements in performance will emerge from concurrency of operations, and suitability of the algorithm to the architecture.

New technology and architectures bring about increasing design complexities. Design methodology demands regularity, simplicity, modularity, duplication and locality. A regular design is one which displays a uniform and ordered structure. Regularity and simplicity reduce efforts in design for testability and communication paths, communication being the limit-

ing factor on speed. Modularity denotes the use of composition cells which are constructed from simpler cells. Duplication requires the repeated usage of individual cells. Thus modularity and duplication allow designs to be more easily constructed and extended. Locality means that input/output paths for a cell are restricted to the nearest neighbor cells. Further, high performance can be obtained by making use of concurrency by operating identical small processing elements in parallel.

Thus, VLSI high performance can be obtained from architectures with the following attributes:

- regular structure composed of simple processing elements
- communication with nearest neighbours
- simple and regular data and control paths
- concurrency

VLSI design methodology is discussed in greater detail in Chapter 3.

Chapter 2

Recognition Algorithm

Assume that the character has been isolated, and is represented as a bit map within a $n \times n$ matrix. Assume also that the character has been thinned. The recognition method presented here extracts feature vectors derived from the geometry of the character [16]. Although this method has been used earlier in various forms, the algorithm described in this chapter was chosen because of its suitability to parallel VLSI implementation. The chosen feature parameters can be readily implemented in a regular architecture such as a square-mesh processor array. The algorithm chosen does not require the use of mathematical operations like floating-point multiplication or square roots. This simplifies hardware design and allows for higher integration of the array. Further, the concomitant improved performance allows for the addition of further classification tests or preprocessing steps to improve correct recognition.

2.1 Description of the Algorithm

The algorithm requires first of all the determination of the boundaries of the smallest rectangle enclosing the character. Thus, edge detection has to be carried out within the array to determine this rectangle or working field. This is performed by vertical and horizontal projections of the pattern bits to the edges of the array. The non-zero elements define the area bounding the character. This is necessary to generate rejection thresholds for the feature vectors.

As was discussed in Chapter 1, feature extraction involves deriving parameters from the structure of the character. The parameters chosen are listed below:

- openess in a certain direction (e.g. A is open to the bottom)
- presence, location and size of bubbles (e.g.: A and B have 1 and 2 bubbles, respectively)
- presence, location and size of horizontal and vertical lines .
- vertical and horizontal projections vectors (sums of columns and rows respectively)
- presence of round corners

The presence of bubbles and openess in a character are termed global

features, since they do not involve details of the character. As the term implies only general information is extracted. Global features allow the patterns to be separated into classes which contain subclasses. This decreases the number of characters to be considered for recognition, and reduces the time for recognition. The alphanumeric characters can be classified into classes as shown in Fig.2.1.

Two further global features are the profile and corner vectors, which are used to analyse characters within a class. The profile vectors are essentially histograms of the vertical and horizontal bit distributions of a character. The corner vectors are four-bit vectors, each bit in the vector corresponds to a corner. A bit has the value 1 if a round corner is detected.

Secondary or local features consist of the location, size and number for each of three features: bubbles, vertical and horizontal lines. These features enable the subclasses of a given class to be uniquely identified. The amount of noise will determine the distance or uniqueness between patterns of a subclass. Location, size and number are represented by the position of 1's within a binary string.

To enable detection of the recognition parameters mentioned above, the outer fields, G, are first determined. These are the patterns generated by propagating inward from all four sides of the bit array. Initially, these patterns are exact duplicates of the pattern field P, which is a bit map of the character. Every 0 is changed into a 1 (on each row and column), until

	L	R	B	T	O	
	0	0	1	0	1	A
L left	0	0	0	0	1	B D O P Q 8 4 0
R right	0	1	0	0	0	C E F
B bottom	0	1	0	0	1	G 6
T top	0	0	1	1	0	H M N W
O bubble	1	0	0	0	1	9
	1	0	0	0	0	3
	1	1	1	1	0	X
	0	0	0	1	0	U V Y
	0	1	1	0	1	R
	0	0	0	0	0	L T 1 7
	0	1	1	1	0	K
	1	0	0	1	0	J
	1	1	0	0	0	I S Z 2 5

Figure 2.1: Global features representing the Class 1 vectors. A "1" indicates the presence of concavity (L,B,R,T) and bubbles (O). For example, the letter A is open from the bottom and has one bubble.

a 1 (representing the edge of the character) is reached. The outerfields are obtained by removing (setting to zero) the bits corresponding to the character. Thus, there are four outer fields G^k, $k \in (l, b, r, t)$ where l, r, b, t stand for left, right, bottom and top, respectively. By overlaying the outer fields with other fields, the required features can be extracted.

The processing performed at each pixel D_{ij}, $1 \leq i, j \leq n$, given the outerfield bits G^k_{ij} and the pattern bit P_{ij}, is as follows.

The inner field, represents bubbles and regions that cannot be reached by the outerfields. This field is obtained by overlaying the pattern and outerfields. The only zero elements will occur in regions completely (or partially) enclosed by the character. The complement results in the inner field:

$$I_{ij} = \overline{P_{ij} \vee G^l_{ij} \vee G^r_{ij} \vee G^t_{ij} \vee G^b_{ij}}$$

The open field is obtained in a similar fashion by overlaying the pattern and inner fields, and all outerfields, except the one corresponding to the direction of concavity (openess):

$$O^b_{ij} = \overline{P_{ij} \vee I_{ij} \vee G^r_{ij} \vee G^t_{ij} \vee G^l_{ij}}$$

(similarly for O^l_{ij}, O^r_{ij}, O^t_{ij})

The horizontal and vertical lines are obtained by shifting the pattern field one unit cell, and logically ANDing the shifted and original pattern fields. Horizontal and vertical lines will be detected by performing horizon-

tal and vertical shifts, respectively:

$$H_{ij} = P_{ij} \wedge P_{ij-1}$$

$$V_{ij} = P_{ij} \wedge P_{i-1j}$$

The corner field is determined by observing that round corners do not coincide with the boundary of the working field, whereas a right angle will. Adjacent outerfields are ANDed for each of the four corners:

$$C_{ij}^{rb} = G_{ij}^{r} \wedge G_{ij}^{b}$$

(similarly for C_{ij}^{lt}, C_{ij}^{lb}, C_{ij}^{rt})

The resulting projection vectors for each field must be analyzed carefully. The effects of digitization can create false results. Therefore, thresholds are established empirically to minimize the effects of noise. For example, the length of the inner field's vertical projection vector must be one quarter of the width of bounding rectangle, and its horizontal projection vector one quarter of the height of the bounding rectangle. Otherwise, no bubble is detected.

2.1.1 Sources of Noise

As previouly mentioned handwriting varies dramatically among different individuals and to a lesser degree for one person. This is a source of noise that must be considered by the recognition algorithm. In general, noise can be classified into two types: local and global.

Local noise is equivalent to speckle or spurious noise. It is present throughout the image, but it does not affect the shape of the character. Examples are specks of dust on the document, broken lines, variations in line thickness. Preprocessing can filter out this type of noise almost completely. Operations such as thinning and hole-filling are adequate methods, which can be readily implemented in VLSI.

The effect of global noise is more serious. Here, the character shape is affected. Size differences are not important (scale invariance), but slanting of a character can affect the recognition process (rotation sensitivity).

The algorithm described above minimizes these adverse effects in two ways. First, it uses redundancy by performing many tests on a character before making a decision. Formally, it strives to maximize separation of the classes in feature space. This method is used by people during listening or reading. It has been shown that understanding deteriorates if redundancy is reduced. Similarly, human error in reading out-of-context reaches 6% [2].

Secondly, the algorithm allows for supervised learning. Thus, exposing the system to a few samples of an individual's handwriting will reduce the effects of global noise. Essentially, the differences between representations of a single character are averaged.

2.2 Classification and Recognition

As shown in Fig.2.1, each element of the class vector is either 1 or 0 depending whether the global feature is present or not. Redundancy is used in the selection of possible candidates to improve the recognition performance. Two classes are used to determine possible candidates for recognition. Class 1 represents characters which match exactly the global features. Class 2 represents characters which closely resemble Class 1. Thus, the classes (Class 2) which are a Hamming distance 1 of the selected class (Class 1) are also included in the recognition process. The Hamming distance is defined to be the number of corresponding vector elements which differ in value. The Hamming distance is appropriate since it represents exactly characters which differ in only one global feature. Also, it can be implemented using simple hardware.

The candidates (i.e. subclasses) belonging to Class 1 receive a ranking value of 1, while those in Class 2 receive $\frac{1}{2}$, thus favoring the Class 1 candidates. Next, three analyses are carried out: profile, corner, and local features analyses. The ranking value of all analyses are added to form a final ranking of the candidates. The highest ranking candidate is the recognized character.

Inner products are performed between the normalized corner and profile vectors, and the stored vectors. The corner vector is a four element vector

consisting of 1's and 0's, depending on the presence of round corners. The candidates are ranked according to the magnitude of the vector products.

To generate profile descriptors from the projected pattern fields, the corresponding vectors must first be of the same dimension as those stored in memory. The source vector $\vec{S}$ is geometrically projected onto the transformed vector $\vec{T}$ (Fig.2.2). The elements of $\vec{S}$ pointing to the element t_i are added to form the transformed element t_i. If only a fraction α of the element s_i is pointing to t_i, it is also added, such that (1 - α)s_i is added to t_{i+1}. In the case of binary vectors, logical ORing is applied instead of addition [17]. The resulting vector is then normalized.

Euclidean distance is used to determine likeness Δ_l between the resultant projection vector $\vec{T}$ and the stored vectors $\vec{B^l}$, where $l = 1, 2, \ldots, 36$ (26 letters and 10 numerals)

$$\Delta_l = (t_0 - b_0^l)^2 + (t_1 - b_1^l)^2 + \cdots + (t_m - b_m^l)^2$$

where $\vec{T}$, $\vec{B^l} \in \Re^m, m < n$, n is the array size

and $|\vec{T}| = |\vec{B^l}| = 1$.

Thus the Δ_l closest to unity indicates the most likely character, l. Euclidean distance between normalized vectors is the same as the cosine of the angle between them. For unnormalized vectors, the distance depends on the vector length. This is illustrated in Fig.2.3. Thus vector normalization is required to preserve scale invariance. The corresponding Δ_l's from

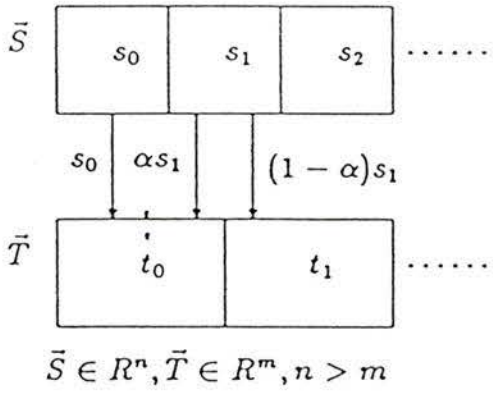


Figure 2.2: Vector compaction.

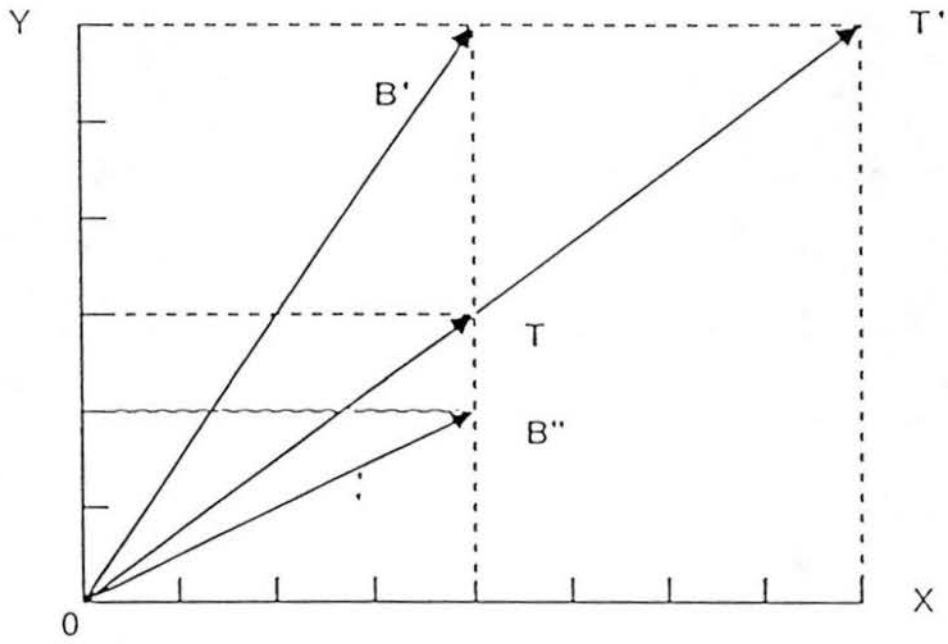


Figure 2.3: By the Euclidean distance rule, the pattern T will get assigned to the class B''. However, T' will get assigned to the class B', even though $T' = kT$. Thus, vector normalization must be used.

the horizontal and vertical projections are added to form a single profile descriptor.

Learning and recognition of the local feature vectors are based on the Perceptron model [18]. This heuristic model, which appeared in the 50's, served as a foundation for the development of non-parametric learning theory. The essential idea in this model is that a machine can be taught to recognize signals without information about the signals. In other words, a perceptron is capable of classifying patterns, based on detected local features, but unable to determine relations between these features. If $\vec{L}(p)$ is the local feature vector, a function of the pattern p, each element $l_j(p)$

$$l_j(p) = \begin{cases} 1 & \text{if } V_j(p) > T_j \\ 0 & \text{otherwise} \end{cases}$$

where $V_j(p)$ is some transformation on the pattern p, and T_j is the threshold. Thus local features are represented as a binary string. In our case, all transformations on the pattern consist of field projections.

Each of the three local feature vectors (horizontal and vertical lines, and bubbles) is 28-bits long. The first four bits represent the number of occurrences of a feature.

$$1 \implies 1000$$

$$2 \implies 0100$$

$$3 \implies 0010$$

$$4 \implies 0001$$

The next 12 bits, 3 bits for each occurrence, indicate the size or length of the bubbles or lines.

$$\begin{aligned} \text{short} &\Longrightarrow 100 \\ \text{medium} &\Longrightarrow 010 \\ \text{long} &\Longrightarrow 001 \end{aligned}$$

The final 12 bits are similarly encoded to indicate the location of the occurrence within the working field. The local feature vectors can be thought of as one 84-bit vector.

A linear classifier is used to determine which candidate will yield the greatest discriminant function by multiplying the local feature vectors by a weighing matrix, stored in external memory. A single-valued function $F_l(\vec{X})$ of the local feature vector $\vec{X}$ is called a discriminant (or threshold) function if the following recognition rule is used to assign $\vec{X}$ to class l such that

$$F_l(\vec{X}) > F_k(\vec{X}) + \rho$$

for all $k \neq l$, and where ρ is rejection/substitution coefficient.

If ρ is close to zero, then substitution will prevail. If ρ is too large, all patterns will likely be rejected. In this work, ρ was set to be zero. This allowed us to assess the properties of the algorithm.

Here, the discriminant function is

$$F_i(\vec{X}) = \sum_{j=1}^{84} w_{ij} x_j$$

where $i = 1, 2, \ldots, 36$, $x_j \in \{0, 1\}$

and the weight matrix $\mathbf{W}$ with $w_{ij} \in \Re$.

For a given character i, it consists of summing the elements w_{ij} for all $x_j \neq 0$.

The formation of the weight matrix $\mathbf{W}$ will be described in the learning section.

Ranking is performed by adding the ranking values of each candidate for each analysis. The ranking value for each analysis is equal to the inverse of the candidate's position (e.g. third place $\Longrightarrow$ rank $= \frac{1}{3}$). Thus the higher rank implies a greater resemblance.

The candidate having the highest total ranking is the recognized character.

2.3 Learning

Initially, reference feature vectors for each character must be learned and stored. Here a supervised learning method is used [16,19]. The teacher must initially tell the system whether it has recognized the letter correctly or not.

A mathematical average is performed on the elements of the corner vector, profile vectors, and the corresponding stored vectors. This simulates the forgetting of incorrect characteristics, and reinforces the correct ones. Each learning iteration will bring the stored vector closer to the required vector. The convergence of this relaxation method is quite rapid (geometric).

The local feature vectors are multiplied with a weighing matrix. During the learning period, the individual weights can be increased or reduced to enhance correct characterization. Initially the matrix is initiated (randomly) with $0 < w_{ij} < 1$. This is adequate to guarantee the convergence of the learning process.

As the system is exposed to characters, a reward and punishment scheme (perceptron-like) is employed. If the system recognizes a character correctly, the weights for that character are slightly increased. Simultaneously, the weights of the other candidates are decreased slightly to reinforce correct recognition. The reward rule for the correct character i_r is

$$w_{ij}^{new} = w_{ij}^{old} + \lambda x_j \theta_i^{rew}$$

$$\theta_i^{rew} = \begin{cases} 1 & i = i_r \\ -1 & \text{otherwise} \end{cases}$$

The constant $\lambda = 0.03$ has been found to be adequate for learning in a few iterations.

If the system incorrectly recognizes the character i_r, weights are increased for the correct character i_c and reduced for all others as follows

$$w_{ij}^{new} = w_{ij}^{old} + \lambda x_j \theta_i^{pun}$$

$$\theta_i^{pun} = \begin{cases} 2 & i = i_c \\ -2 & i = i_r \\ -1 & \text{otherwise} \end{cases}$$

The penalty has been made larger to increase the effectiveness of the punishment.

Therefore, the system is completely adaptive, allowing recognition of a variety of different handwriting styles. Once the system has learned different sets of patterns, the operator can simply call the required set of patterns. It should be emphasized that the learning capabality is a feature of the algorithm, not of the feature extraction chip.

2.4 Simulation

The algorithm was trained on typewritten quality (noiseless) uppercase characters. Noisy characters were used for determining the proportion of correctly recognized characters. The simulation program, written in the C language, was executed on a SUN workstation. Each character was represented by a 20×20 bit map, which was generated on the computer.

The results indicate that better than 95 percent of the characters were recognized correctly. These results are comparable to those published earlier [16].

A number of improvements can be added to enhance correct recognition. A larger matrix (i.e.greater resolution) will reduce the staircase effect. It will also minimize the smearing effect of compressing the profile vectors. Another improvement would be to weigh each analysis, depending on the amount of information provided by the analysis. For example, the profile and local analyses provide more information than does the corner analysis.

Pavlidis *et al.* [1] have reported a correct recognition better than 97% for typed multi-fonts, using a similar method with a few added heuristics and a Bayesian classifier (also a linear classifier such as the one used for local classification). For a single font, the algorithm will yield better than 99% correct recognition in some commercial products. For handwritten numerals, Badreldin *et al.* [20] have reported an average of 99% correct recognition. Their method uses outerfields to extract the features. These results were obtained using high-resolution bit maps.

Thus it seems that a larger correct recognition performance can easily be attained by augmenting this algorithm to encompass other techniques or heuristics. As an example, another local feature vector representing the size,length and location of the outerfields, should help to achieve higher correct recognition rates.

Therefore the three most important aspects of this algorithm are:

1. It is well suited to cellular VLSI inplementation (parallelism). This

can be seen by observing that the operations on all pixels are exactly the same and can be performed concurrently.

2. It is adaptive (supervised learning).

3. It offers better than 95% correct recognition (to be discussed later).

Chapter 3

Structured Design and Testability

This chapter describes the steps taken and the methods employed to arrive at a final design which meets the system specifications. A structured VLSI design methodology is first presented as well as the decomposition of the design process. Design for testability is then discussed in detail.

3.1 Structured Design

Modern technology has made possible the implementation of designs containing thousands of transistors. But the complexity in designing VLSI chips is not due solely to the sheer number of transistors. The designer is faced with further problems in the areas of design synthesis and verification. The correctness of design is of critical importance. Debugging a faulty chip is very difficult, and necessary alterations cannot usually be achieved. Therefore, it is imperative that a VLSI implementation be testable to iden-

tify the causes of faults and to prevent shipping faulty chips.

Given a system specification, the designer can choose from almost unlimited design options on circuit styles, clocking strategies, physical placements, to name a few. However, this flexibility is limited and constrained by design parameters such as size, speed and power, as well as of testability. Therefore, a structured design methodology is necessary to reduce the overall design process to a manageable set of sub-processes. This section focuses on a structured hierarchical design method.

Any structured VLSI design methodology can be broken into three phases:

1. specification

2. implementation

3. evaluation

The specification phase consists of determining design goals which suggest a possible architecture. However, as the design process approaches the physical level, the initial architectural definition is often modified in order to reach the full performance potential of the design. It is also important for the designer, at this stage, to visualize the design in the (x,y,t) space [21]. This three-dimensional representation incorporates both the static and dynamic aspects of a design. Such a space-time approach to VLSI

design has contributed to novel architectures such as the processor array, discussed in the following chapter.

The implementation and evaluation phases can be simplified by the use of abstraction and constraints. Constraints allow the automation of certain processes such as design rule checking and physical level design. Abstractions allow the designer to layer the design processes into levels of increasing complexity.

A recent technique which incorporates the above approach is the *floorplanning* method [21]. Essentially, *floorplanning* is a hierarchical method which allows a check on the feasibility of design at a high level, throughout the design process. Hierarchical design employs a "divide and conquer" approach, subdividing modules into simpler ones, until the level of complexity is manageable.

The design process begins by specifying a bounding box, which represents the entire chip. At this stage, it is necessary to define global signals which consist of ground, power, clock, as well as data and control flows. The floorplan or root cell is subdivided into leaf cells (simpler floorplans). This method is similar to that used in software development, where a large program is broken down into simpler functions. Thus floorplanning is characterized as a top-down technique. The only bottom-up designing is performed at the physical level, where for example a NAND gate will be constructed from simple transistors.

Floorplanning exhibits two important advantages:

1. Verification of the design is greatly simplified, since the leaf cells can be evaluated individually.

2. Optimization is greatly enhanced by the ability of the designer to alter both the placement of the leaf cells and the leaf cells themselves, as the design progresses.

A CAD package which incorporates support for the floorplanning capability is VIVID [22]. This package was used for implementation of this project.

Each cell or floorplan obeys a set of rules:

- Each cell is defined by a rectangular bounding box, which lies on a cartesian (x,y) virtual grid.

- Instances of cells are global to the design, as are standard functions in high-level language such as C.

- Named pins (i.e. ports) represent inputs, outputs, specific signals or wire connecting points, and should lie on the edges of the bounding box to facilitate connection by abutment.

- Overlap of instances (subcells) is prohibited.

The placement of cells, is accomplished both by symbolic (e.g. pin2 of cell B) or absolute (e.g. x,y = 2,17) location. Symbolic placement allows, for example, the lower left corner of cell A to be placed at the upper left corner of cell B. This capability of relative positioning of cells enables the alteration of the floorplan at a higher level without destroying the connectivity of subcells and without respecifying their location, as would be necessary with absolute placement.

A further enhancement in VIVID, is the capability of symbolic layout [23]. Symbolic layout simplifies the physical level design process by capturing a mask-level design in "stick" format. For example, rather than drawing a rectangle for a wire, a single line is used. This procedure essentially hides the design rules from the designer. Again a parallel in software exists: the mask-level description corresponds to machine code and the "stick" format corresponds to an assembly language.

Finally, VIVID's graphics editor ICE, not only accelerates the design process, but also lets the designer verify the design interactively by calling a built-in design rule checker, fulfilling the need for true hierarchical design verification.

In summary, floorplanning is a powerful tool in top-down VLSI design. It not only simplifies verification, optimization and design alterations, but also enhances the creation of high performance architectures by helping the designer to map high-level representations of the design.

We have so far discussed the complexity of VLSI design, and how hierarchy can be used to manage this complexity. We now turn to another aspect of VLSI design: the need for testability.

3.2 Design for Testability

The central idea behind testing is that of differentiating between a good chip and a faulty one. A further desirable feature of testing is to isolate and locate faulty component(s), allowing the implementation of a fault tolerant design. Design for testability describes approaches to VLSI design which simplify and enhance test generation, fault coverage and diagnosis. One definition of a testable circuit is

1. it is controllable by test patterns through primary inputs (inputs to the logic block under test)

2. the primary outputs uniquely identify the state of the circuit

These requirements usually entail extra gates and controls. Additionally, difficulties are encountered in the generation of correct test patterns and the estimation of their effectiveness through fault simulation [24]. The emphasis here will be placed on the area of self-testing, which is the technique adopted in our design.

Other structured techniques, which allow testing, exist [25,26]. These include scan path, random access scan and level sensitive scan design (LSSD).

The disadvantages of these testing strategies are that a large test data volume is required, and that these data must be shifted in and out of the chip. However, these scan techniques can be used to isolate faults of a general sequential circuit. Also, area overhead and performance degradation are minimal.

With VLSI networks increasing rapidly in density, and the necessary test patterns growing even faster, on-chip generation and compaction of test patterns has become an attractive solution.

Specifically, we look at an *insitu* (built-in) self-testing technique: Signature Analysis [26]. Insitu, in contrast to *exsitu*, implies that the testing components are part of the system function. The obvious advantages are that overhead in area is minimized and testability is achieved as an integral part of the design, satisfying the requirements of a structured design approach, while providing good fault coverage. Also, no additional pins are required to feed or extract test data, and test time and complexity are reduced.

Other techniques belonging to the same class, include transition counting and syndrome testing. These however have strong disadvantages in terms of fault coverage and VLSI implementation [27,28]. Another technique is the analysis of the Walsh spectral coefficients. This technique is similar to checking the circuit's truth table in the sense that the coefficients of the Walsh function uniquely describe the circuit. It has no advantage

over the verification of each output of the circuit under test in that it requires all possible input patterns. However, good fault coverage can be achieved if a subset of the Walsh coefficients can be determined [28].

Before describing signature analysis we briefly review the properties of Linear Feedback Shift Registers (LFSR's).

A LFSR consists of two basic components: storage devices (e.g. D-latches) and modulo 2 adders (XOR gates). An autonomous (no input) 3-bit LFSR is shown in Fig.3.1.

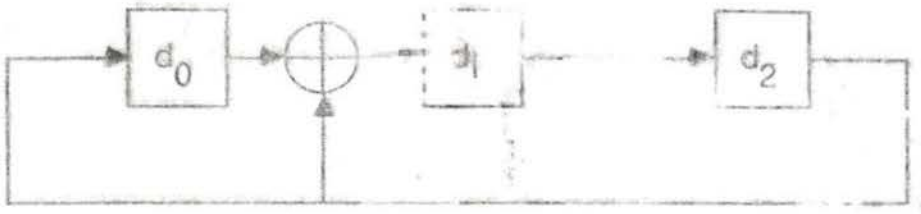
Two important characteristics about a LFSR follow:

1. The maximum number of distinct states that can occur is 2^k, for a k-bit register. However, since the all zero sequence will force all subsequent states to be zero, it is not allowed. Thus the maximum number of states is $2^k - 1$.

2. The present state of an LFSR depends only on the previous state and on the input, if present.

From the above two statements, the sequence appearing in an LFSR will be periodic, with period length less or equal to $2^k - 1$. It will be useful to look at the LFSR, with an input, implemented as a divider. Fig.3.2 shows the modulo 2 dividing process.

A binary string can be represented as a polynomial with binary coefficients. As an example, the string (1101) represents the polynomial



LFSR contents

d_0	d_1	d_2	
0	0	1	← initial state
1	1	0	
0	1	1	
1	1	1	
1	0	1	
1	0	0	
0	1	0	
0	0	1	← repeats sequence
1	1	0	

Figure 3.1: A 3-bit LFSR.

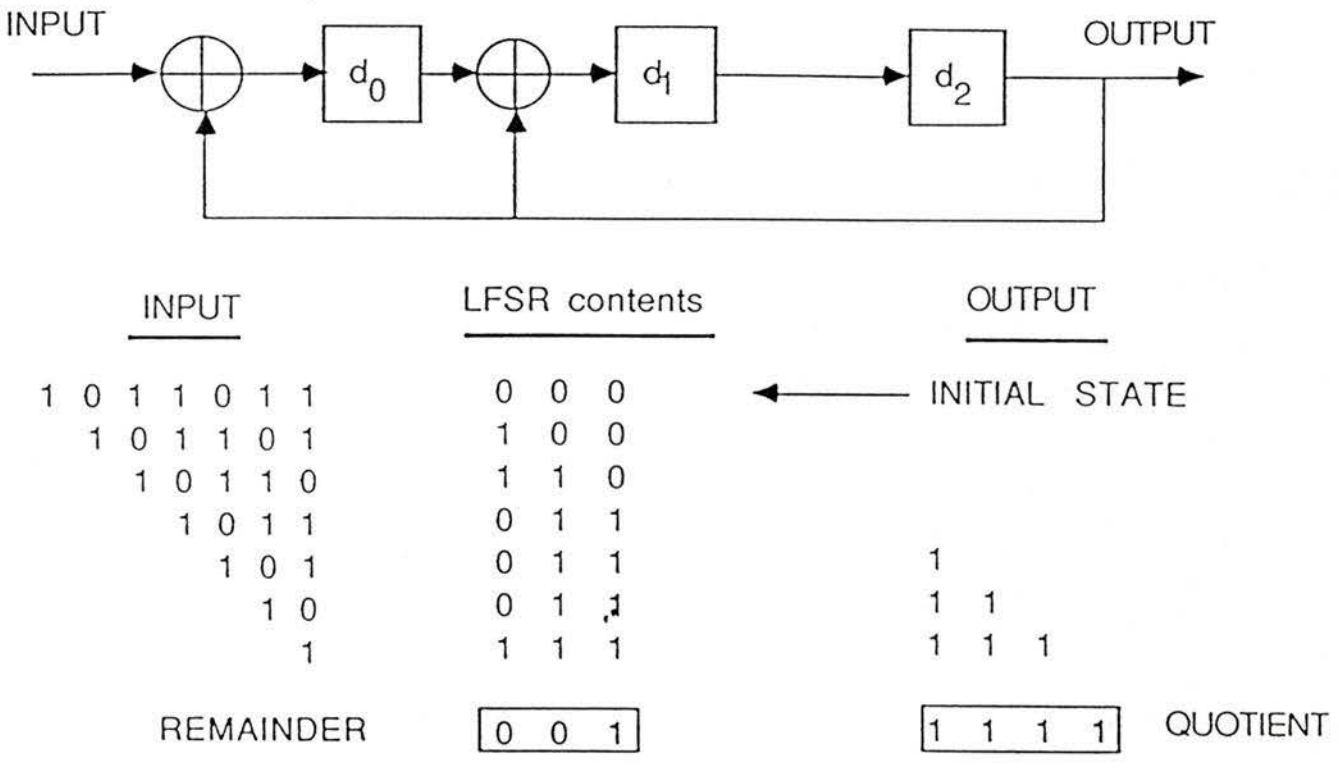


Figure 3.2: LFSR performing mod 2 division of the input $x^6+x^5+x^3+x^2+1$ by x^3+x+1. The remainder is x^2 and the quotient is x^3+x^2+x+1.

$x^3 + x^2 + 1$. A function $F(x)$ of degree n is linear if

$$F(p(x) + e(x)) = F(p(x)) + F(e(x))$$

where $e(x)$ and $p(x)$ are polynomials of degree $\leq n$.

A shift register with an XOR feedback is a linear sequential circuit. This is illustrated in Fig.3.3. We note that the remainders (or signatures) $S(x)$ also follow the linear relationship

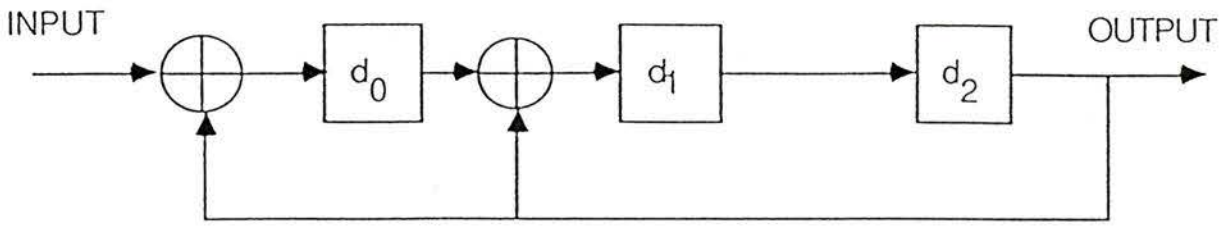
$$S(p(x) + e(x)) = S(p(x)) + S(e(x))$$

We also note that $S(p(x)+e(x))$ differs greatly from $S(p(x))$, eventhough $p(x) + e(x)$ and $p(x)$ differ only in a single bit.

As mentioned earlier, we wish to generate test patterns, which exercise the target logic block, compact the outputs to a manageable size, called the signature of the network, then decide if a fault is present. The following two subsections deal with test pattern generation and signature analysis separately. Before continuing, it should be mentioned that signature analysis is based on algebraic coding theory. A more complete treatment of the following material can be found in reference [29,30].

3.2.1 Test Pattern Generation

Two approaches can be taken for test pattern generation: exhaustive and pseudo-exhaustive pattern generation. The pseudo-exhaustive ap-



INPUT SEQUENCE	REMAINDER
P + E = 1 0 1 1 0 1 1	S(P+E) = 0 0 1
P = 1 0 1 0 0 1 1	S(P) = 1 1 1
E = 0 0 0 1 0 0 0	S(E) = 1 1 0

$$S(P+E) = S(P) + S(E)$$

Figure 3.3: Linearity of the LFSR.

proach is used when the number of patterns to completely test a system is too large. The system is examined to determine if it can be broken down into subsystems which can be tested with fewer than 2^n patterns, where n is the number of inputs. However, this problem is very difficult to solve [26]. Another drawback is that this method is function dependent, that is, it depends on the logic block to be tested.

Exhaustive testing applies all 2^n patterns to the network under test. This is practical only if n is not too large. For example, if a 32-bit adder had to be tested, 2^{65} patterns would have to be applied (two 32- bit input pairs and a carry-in). If a test pattern could be applied every nanosecond, the test duration would exceed 1100 years! Thus, to minimize the number of test patterns some information must be known about the logic block under test. For example, if every output of a logic block does not depend on all inputs, then it is possible to reduce the 2^n patterns to subsets which test each output.

An LFSR is a compact and simple method of generating all necessary patterns. If the k-bit LFSR is capable of generating $2^k - 1$ sequences, it is called a maximal length generator. To achieve this the characteristic polynomial of the LFSR must be a primitive polynomial (or an irreducible polynomial with $2^k - 1$ a prime) [31]. The binary sequence output from the LFSR is called a pseudo-random binary sequence. This means that the sequence has statistical characteristics of randomness, but is still predictable.

This predictability of the sequence allows the designer to determine how a "good" network should behave. It can be shown that the pseudo-random sequence (maximal length) has the following properties associated with randomness:

1. The number of ones and zeros is approximately the same.

2. Runs of ones and zeros frequently occur, with short runs more frequent (similar to an ideal coin toss).

3. The auto-correlation function is peaked and two-valued.

Thus, a pseudo-random sequence can be viewed as locally random while being repeatable.

We note at this point that a maximal length LFSR does not generate the all zero sequence. To allow the all zero pattern, and yet not lock the LFSR, a non-linear feedback element such as a NOR-gate is required. For example, in a k-bit LFSR, the first $k-1$ bits are connected to a NOR-gate, the output of which is fed to the feedback XOR-gate. This non-linear element allows all 2^k patterns to be generated.

We now turn to the problem of data compaction and signature analysis. The effectiveness of signature analysis is also discussed.

3.2.2 Signature Analysis

Signature analysis for digital networks was originally developed by Hewlett Packard [27]. The approach is based on the cyclic redundancy checking (CRC) coding scheme. The signature of a network is the residue in the LFSR after an n-bit data stream has been shifted in. Fig.3.4 shows a self testing network. Essentially, the input data stream is compressed into a k-bit string with a k-bit LFSR. To obtain the signature, the LFSR is initialized to a known state. The usual initial state is the all zero state, since it is easily implemented by a CLEAR operation. The input data stream is then shifted in. A fault in the network is detected if the resultant signature differs from the "good" network signature. If, however, the input sequence is compressed to the same signature as the fault-free signature, the network is either fault-free or undetectable errors have occurred. It should be emphasized that the testing is self-contained, initiation and checking of the signatures being the only external operations. We now discuss the error detection properties of the signature analysis method. We also examine proposed measures of effectiveness in error detection. The appendix contains the theorems and their proofs which describe the properties of the LFSR presented below. The proofs presented follow the approach in [30].

As mentioned in the beginning of this section, a binary sequence can be represented as a polynomial (or vice-versa). Thus, if $p(x)$ is a binary

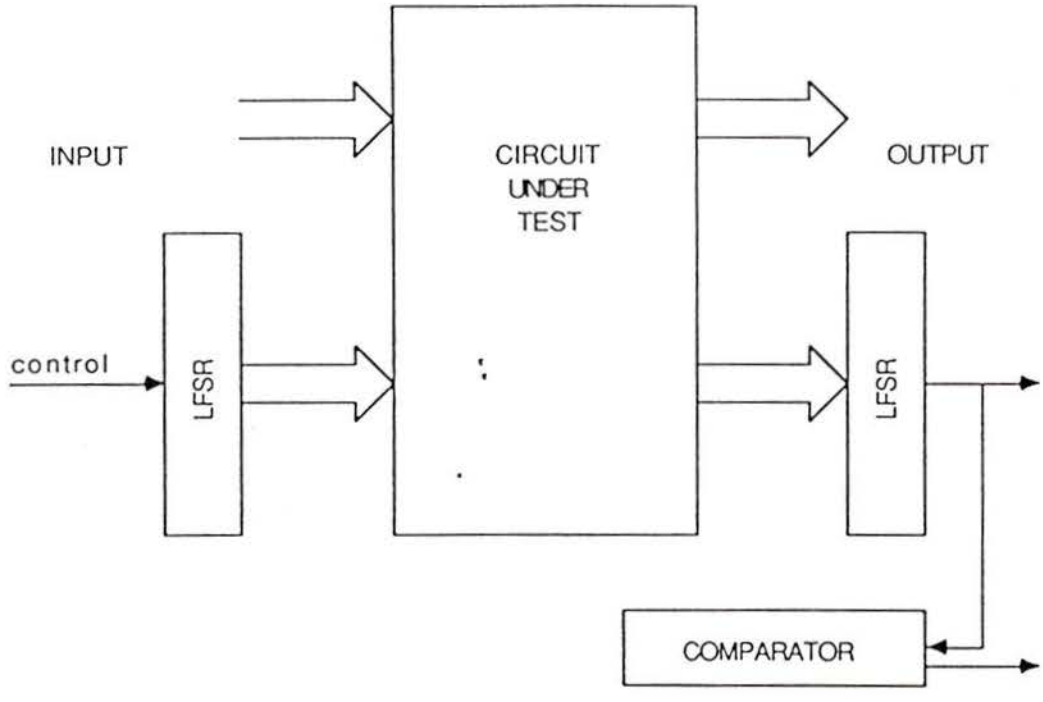


Figure 3.4: Built-In-Self-Testing (BIST) network.

sequence of length n, it can be represented as a polynomial of degree $n-1$. If we now express the characteristic polynomial of the LFSR (the divisor) $d(x)$ as a k-degree polynomial, the relationship between input is

$$p(x) = q(x)\ d(x) + s(x) \tag{3.1}$$

where $q(x)$ is the quotient.

If we let $e(x)$ represent the error sequence, the erroneous sequence $f(x)$ is given by

$$f(x) = p(x) + e(x) \tag{3.2}$$

The sequence $f(x)$ will contain undetectable errors if it satisfies

$$f(x) = \tilde{q}(x)\ d(x) + s(x) \tag{3.3}$$

where $s(x)$ is the fault-free signature.

This implies that an undetectable error sequence will arise only if it is a multiple of the characteristic polynomial (the divisor) of the LFSR [Appendix, Theorem 1]. It is also seen that two identical input sequences will necessarily produce identical signatures.

Another property of the LFSR is that single bit error detection is guaranteed, if the divisor polynomial has at least two non-zero coefficients [Appendix, Theorem 2]. This result can be seen by observing the events in the LFSR. Once a single bit propagates in a single feedback circuit, there are not subsequent errors to cancel the effect of the first bit. It should be noted

that a simple LFSR such as $d(x) = x + 1$ (one D-latch and one feed-back XOR-gate) is sufficient to detect all single bit errors. Such an LFSR is a parity checker and will detect an odd number of errors.

We now turn to the detection of general multiple bit errors. The following measure of detection effectiveness was first proposed in [27]. The probability that an LFSR of length k will not detect errors in an input sequence of length n, assuming a uniform distribution of errors, is

$$probability\ of\ aliasing = P_{al} = \frac{2^{n-k} - 1}{2^n - 1}$$

We can examine two special cases, and try to ascertain the validity of this result [Appendix, Theorem 3]. First, if the input sequence length equals (or is less than) the register length, then

$$P_{al} = 0$$

This means that all errors are guaranteed to be detected. This can be seen to be true by looking at the division performed by the LFSR. The input is shifted in and entirely remains in the LFSR. This is the same as saying that the quotient $q(x) = 0$ such that $p(x) = s(x)$.

The second, and most important, case occurs when the input sequence is large. From above, if $n \gg k$, then the probability of aliasing becomes

$$P_{al} = \frac{1}{2^k}$$

For a 16-bit register the probability of aliasing is about .002 percent. Although, this theorem predicts excellent error detection, it says nothing about which polynomial to use. It only prescribes the length of the register. Thus, $d(x) = x^k$ (no feedback) would suffice. But, this is equivalent to looking at the last k bits of the input sequence, i.e., the test sequence is of length k. Another interpretation of P_{al} is that for $n \gg k$, the probability of a single bit of the signature to be in error is $1/2$, independently of the others.

In this light, signature analysis seems to be redundant for the detection of faults which are equally likely. The reason for this contradiction is that logic faults are not equally likely nor independent as implied. However, it has been shown that $P_{al} = \frac{1}{2^k}$ is a good approximation and is true as the input sequence length $n \rightarrow \infty$. Thus, empirically at least, more complex polynomials ($d(x)$) are more effective in multiple fault detection.

We now look at detection measures based on dependent faults [30,28]. Coding theory is invoked to determine useful polynomials. More precisely, polynomials which generate error detecting cyclic codes are examined. Burst error detection is defined to be the detection of d errors in n consecutive bits. The results show a trade-off between the number of errors detected and the length of the code. Specifically of interest is that a primitive polynomial generates a 2-bit error detecting code of length $2^k - 1$, where k is the register length. Therefore, all 2-bit error patterns will be

detected. Similarly, Theorem 4 in the appendix shows this to be true for r errors in r consecutive bits.

The choice of primitive polynomials over others is also supported in a recent paper [32]. The error sequences were not assumed to be equally likely. The main result is that primitive polynomials converge much faster to the aliasing bound $P_{al} = \frac{1}{2^k}$ than non-primitive polynomials, when errors are not equally likely.

Various approaches can be taken to improve this aliasing probability [28]. One such technique is to extend the length the LFSR by one, reducing the probability of aliasing by 1/2. Another approach, for an LFSR implemented as a true divider, is to take the divisor polynomial to consist of the product between a primitive polynomial and the polynomial $x + 1$. Such an LFSR will detect any odd number of errors in the input, since the parity of the input sequence will be reflected in the signature. If the number of errors is even, then as before, the errors will be detected if $e(x)$ is not a multiple of $d(x)$.

In summary, the advantages and disadvantages of signature analysis are as follows:

1. high fault coverage (low P_{al})

2. Test patterns are generated on-chip.

3. The network response is compacted and evaluated on-chip.

4. The test data volume is reduced and testing equipment is not needed.

5. Few additional pins are required.

6. No special flip-flops are needed as in LSSD.

7. Overhead for insitu implementation consists of the LFSR's and control circuitry.

8. Initiation and checking signatures are the only operations required.

9. Does not provide information for fault isolation. However, some work has been done on fault isolation by observing that different faults will usually cause different signatures, thus, enabling fault location.

Chapter 4

Processor Array Design

4.1 Architectural Requirements

The description of the character recognition algorithm in Section 2.1 suggests a parallel architecture for its implementation. A block diagram of the proposed chip and its peripheral circuitry is shown in Fig.4.1. Concurrency is maximized if the bit map representation of the pattern is mapped onto an array of identical processing elements (PE), each corresponding to a single pixel. Once the input pattern is supplied to the array, all PE's operate simultaneously. Performance can be further enhanced by ensuring that each PE employs bit-parallel operations, such that the duration of each PE task is minimized. Therefore, the whole array maps on an $n \times n$ square grid, as shown in Fig.4.2.

Having established the target architecture further details remain: data flow, control and timing strategies, testability, fault tolerance and intercon-

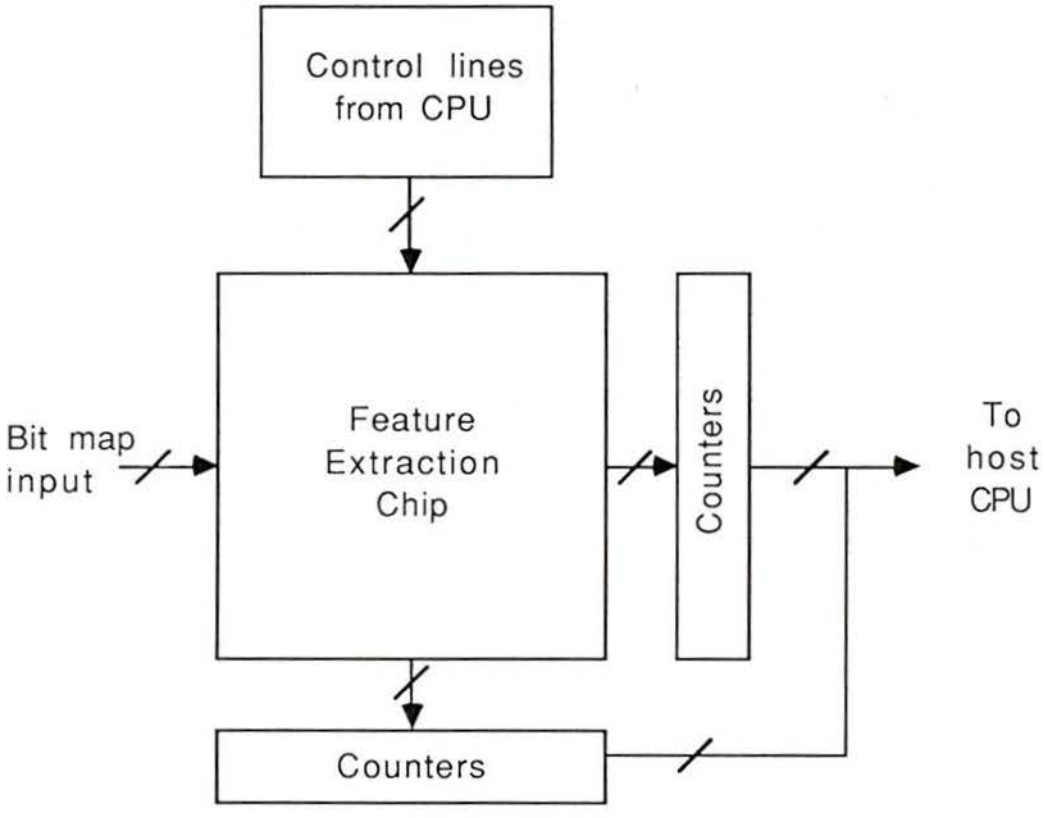


Figure 4.1: Feature extraction system.

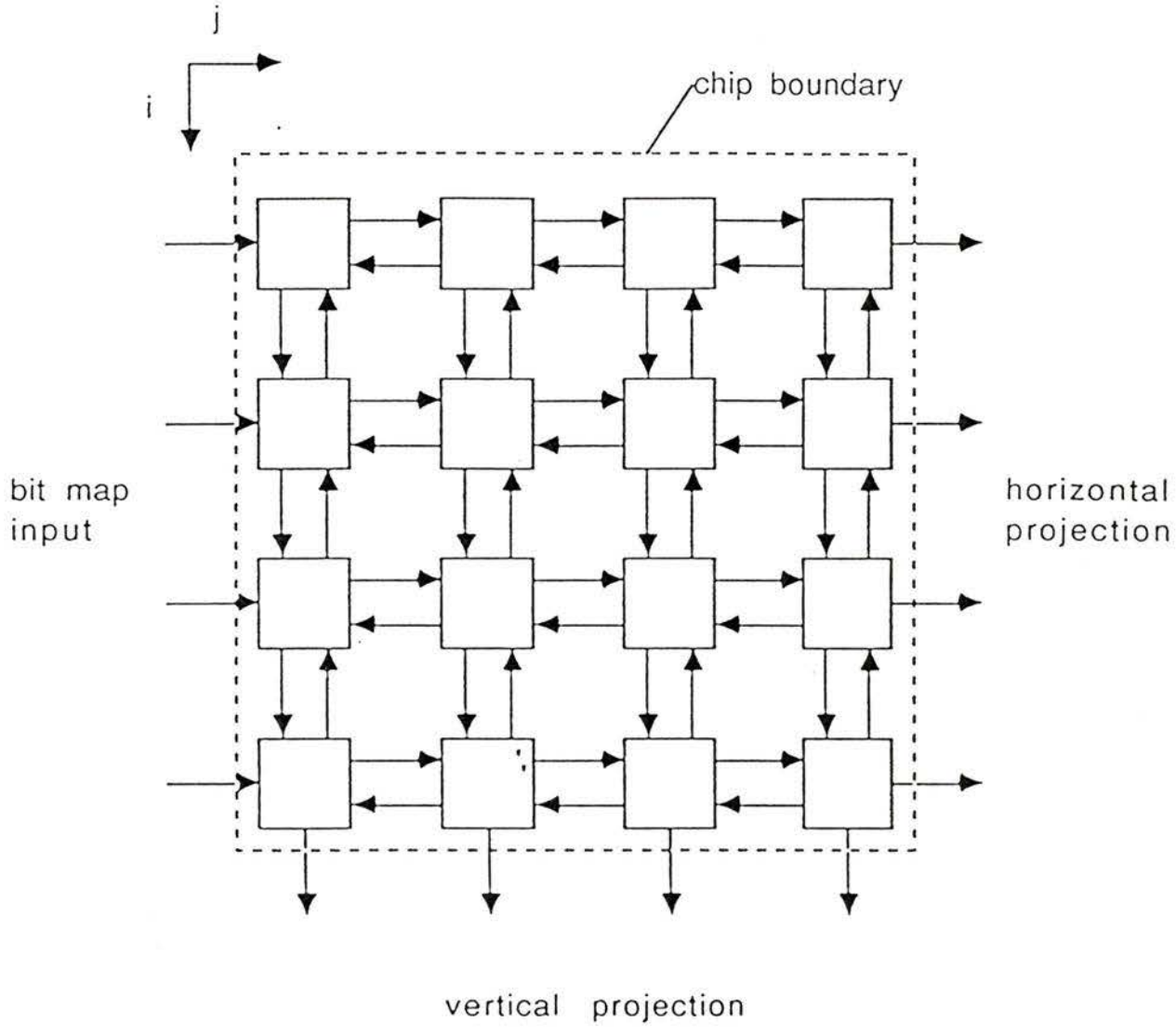


Figure 4.2: A square mesh processor array.

nection topology.

Data flow and interconnection topology can be determined from the algorithm. Again the idea of mapping the algorithm is utilized. The pattern must first be loaded into the array. The resulting output of each PE, must then be propagated to the edges of the array, since the algorithm is based on histogramming the various fields for feature extraction. To meet the requirements of local communication and simplicity, shifting all the rows and columns of the bit map in parallel to the next nearest-neighbor is chosen.

Interconnection topology needs to reflect not only the data flow structure, but also the demands of the individual PE's. This leads to a nearest-neighbor interconnection. A PE is connected to neighboring PE's lying north, south, east and west. The extension to 8-connectedness will be discussed in the last chapter.

The control strategy is determined by the operations of the PE's. Since PE's perform identical tasks simultaneously, the control signals must be broadcast to the entire array. Although broadcasting of control signals violates the requirement of local communication, it is simpler and costs less in terms of area overhead than the self-timing approach. With self-timing, each PE would perform its task asynchronously when the required data are available [33]. To execute these operations, extra logic would be needed to detect data availability and to store control information. Thus,

control signals applied externally from a sequencer will operate all PE's synchronously.

The two-phase clocking scheme was adopted for chip timing. This arrangement uses a clock ϕ and its complement $\overline{\phi}$. The advantages of this strategy is the reduction in the number of clocks distributed. A danger in using this clocking scheme lies in the possibility of clock skew, where ϕ and $\overline{\phi}$ will overlap. The overlap can lead to race conditions and erroneous data storage. To minimize clock skew, the master clock ϕ is broadcast to the entire chip. Its complement $\overline{\phi}$ is locally generated, as shown in Fig.4.3 [34]. Clock distribution is accomplished by cascading inverters, each logic level increasing in fan-out (Fig.4.4). This approach allows the use of small identical inverters (drivers) which can be placed throughout the design.

Now that the basic architectural design has been determined, the design of a PE must be such that it reduces the number of control lines and task duration to a minimum. Furthermore, testability should be incorporated without unduly increasing control complexity, while achieving high fault coverage and low area overhead.

The following sections deal with the implementation of this architecture.

4.2 Functional Details

Before describing the detailed behavior of a PE, we review its data requirements. On the input side, four outerfields, two neighboring and the

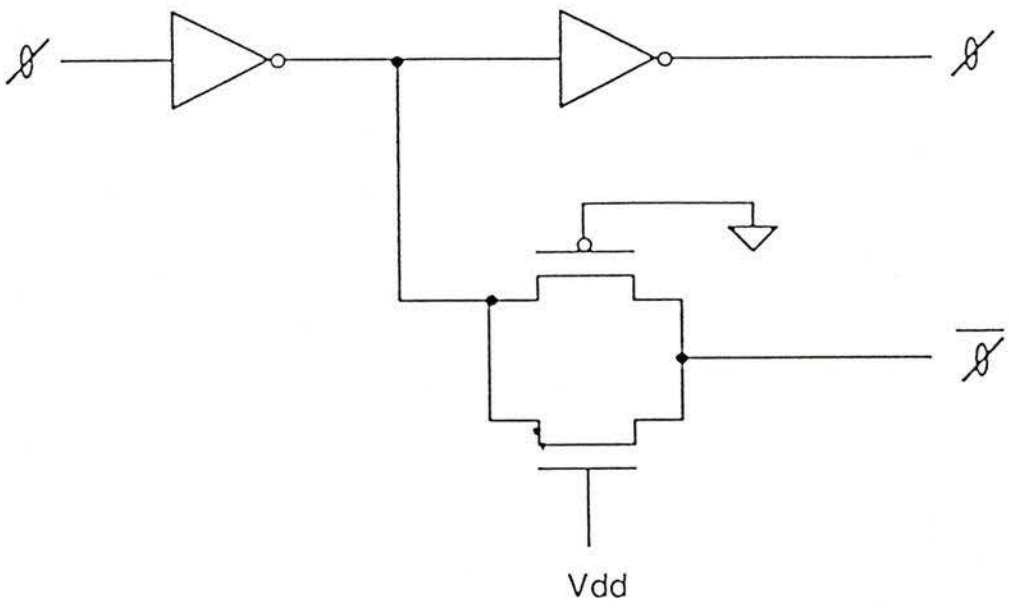


Figure 4.3: Local clock generator which removes clock skew.

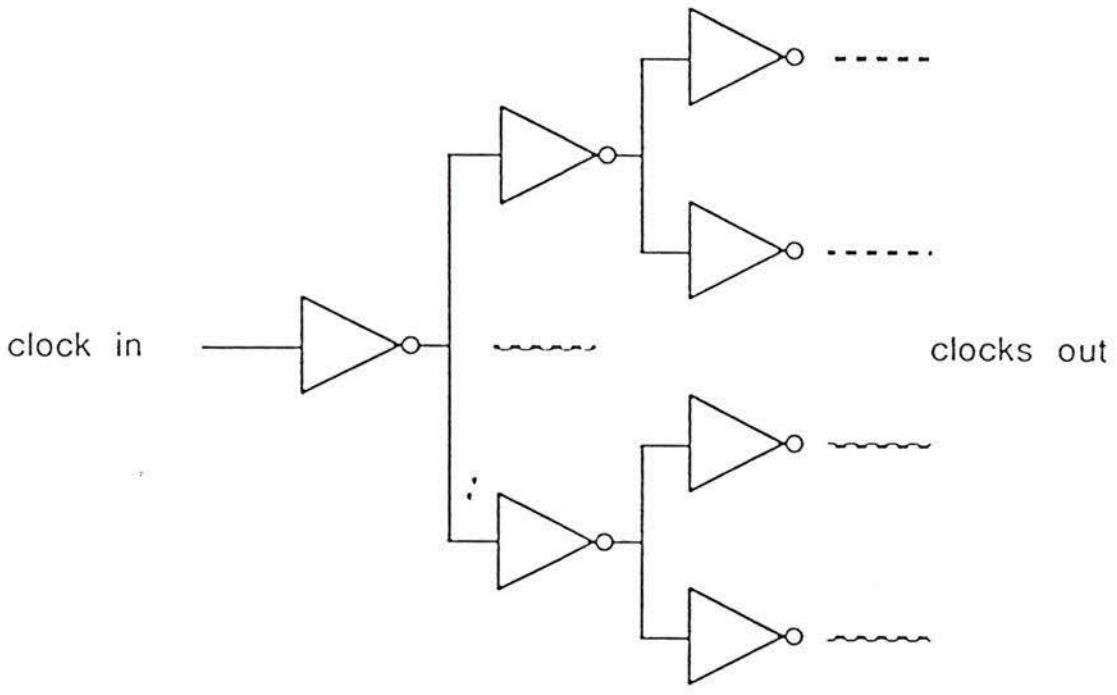


Figure 4.4: Cascading of clock drivers.

local pattern fields, seven bits in all, are required. The output, a total of 12 bits, consists of four corner fields, four open fields, the local pattern field, the horizontal and vertical fields and the inner fields. Thus a 7-bit and a 12-bit register are required for information storage. A logic block between these registers is needed to generate the output. A block diagram of a PE is shown in Fig.4.5.

4.2.1 Logic Block

A programmable path logic (PPL) generates the required outputs. A PPL is a PLA where the OR and AND planes are merged. It results in a more compact design and is easily testable as will be demonstrated. Another advantage of the PPL structure is that it represents logic symbolically, simplifying the mapping of boolean equations to hardware. Before implementing the structure, the functions given in Chapter 2 to generate the output fields can be minimized as follows:

1. Inner field: $I_{ij} = \overline{P_{ij} \vee G^l_{ij} \vee G^r_{ij} \vee G^t_{ij} \vee G^b_{ij}}$
   (no change)

2. Open field: $O^b_{ij} = \overline{P_{ij} \vee \overline{G}^b_{ij} \vee G^r_{ij} \vee G^t_{ij} \vee G^l_{ij}}$
   (simlarly for O^t_{ij}, O^r_{ij}, O^l_{ij})

3. Horizontal line: $H_{ij} = \overline{\overline{P}_{ij} \vee \overline{P}_{ij-1}}$

4. Vertical line: $V_{ij} = \overline{\overline{P}_{ij} \vee \overline{P}_{i-1j}}$

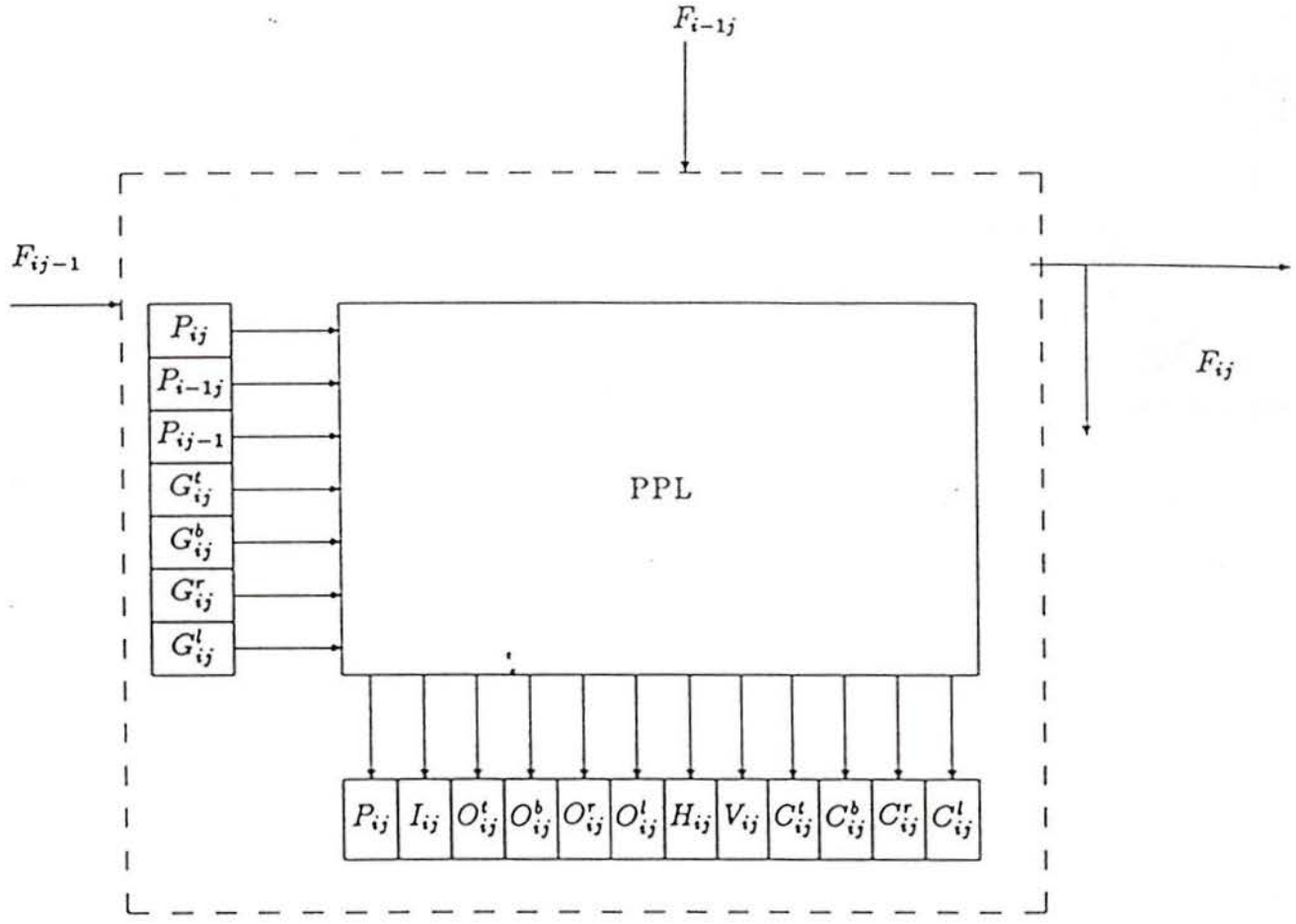


Figure 4.5: Unit cell of the processor array.

5. Corner field: $C_{ij}^{rb} = \overline{\overline{G}_{ij}^{r} \vee \overline{G}_{ij}^{b}}$ (similarly for C_{ij}^{lt}, C_{ij}^{lb}, C_{ij}^{rt})

These NOR operations can be directly mapped onto the PPL as shown in Fig.4.6. Note that the complement of each input is directly available from each register stage.

Dynamic CMOS logic is used. During $\overline{\phi}$, the outputs are precharged to VDD by the p-type transistors. During ϕ, the outputs are conditionally discharged through the n-type transistors.

4.2.2 Testing

As mentioned before, the foremost considerations in the implementation of testable hardware are the achievement of high fault coverage and the minimization of additional hardware.

The Built-in-Self-Test (BIST) approach adopted is shown in Fig.4.7. The test pattern generation is accomplished by a modified LFSR, which generates all 2^k patterns, where $k = 7$. Fig.4.8 shows the modified LFSR, of length 7. Its divisor polynomial is

$$d(x) = x^7 + x^4 + 1$$

The structure of this LFSR is different than that presented in Chapter 3. However, its properties are the same, except that the residue of the LFSR will not be necessarily the remainder of the division. This configuration was chosen to avoid the placement of XOR-gates between register elements, in

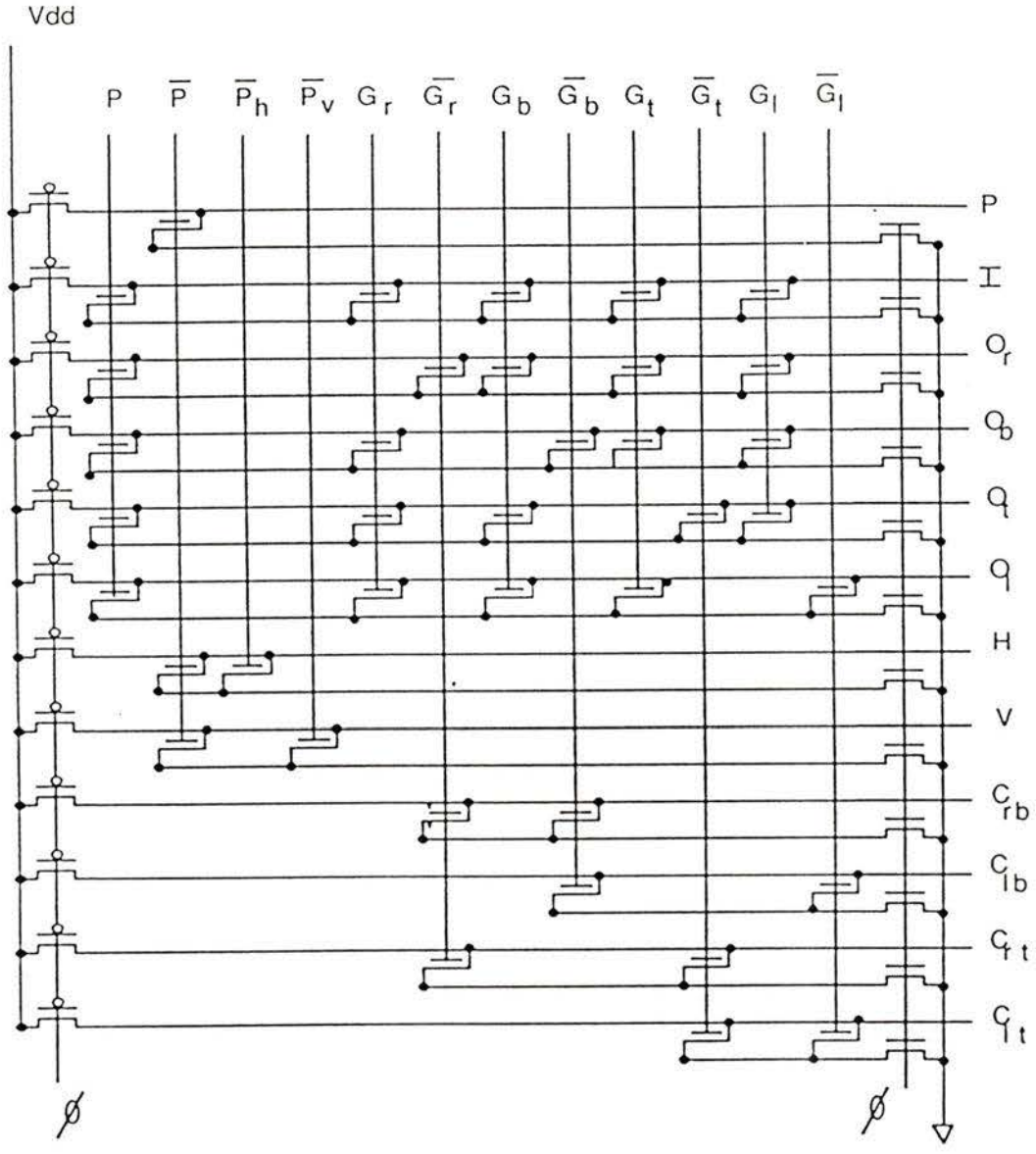


Figure 4.6: PPL to generate feature fields.

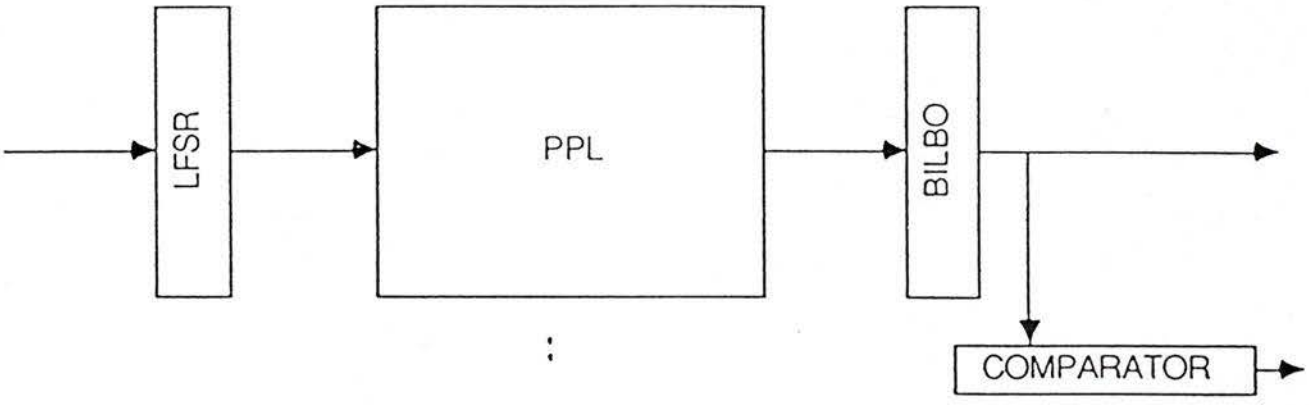


Figure 4.7: Built-in-self-test of a unit cell.

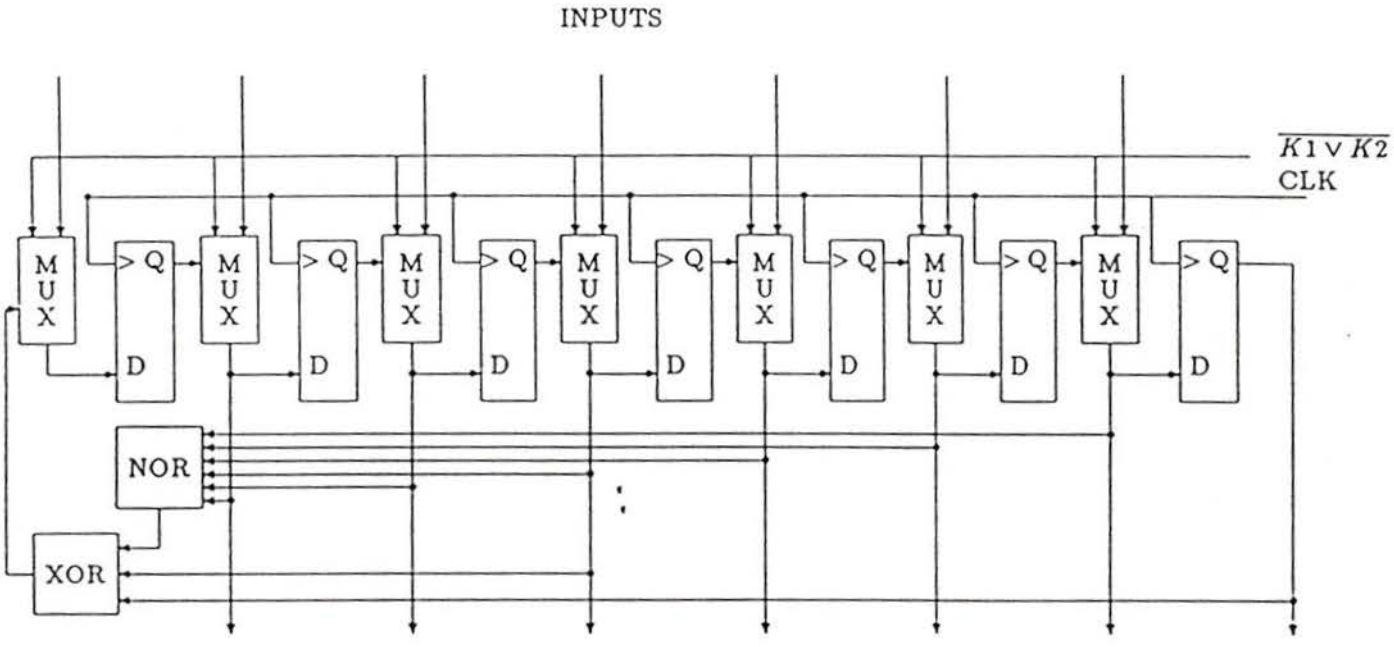


Figure 4.8: Modified LFSR dividing by $x^7 + x^4 + 1$.

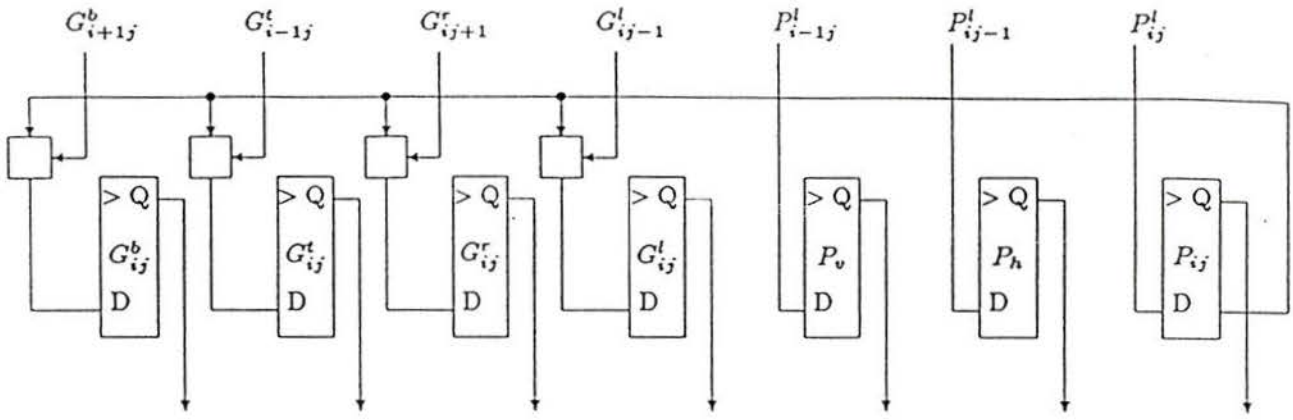
order to improve pitch matching to the PPL [35]. Furthermore, since there are only two feedbacks, no significant delay will be incurred. A NOR-gate is used as the nonlinear element to allow the all-zero sequence. Control gates are used to enable the two required modes: pseudo-random sequence generator and normal modes. These are illustrated in Fig.4.9.

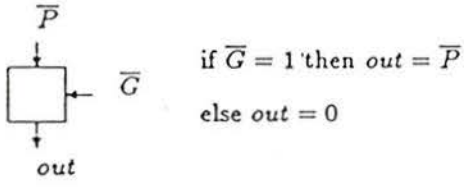
The signature analysis is carried out by a Built-in-Logic-Block-Observer (BILBO). The BILBO structure used for the compaction of the input sequence is shown in Fig.4.10. It consists of a multiple input shift register (MISR) and appropriate control gates. The operation of the MISR is similar to that of a single input LFSR, in the sense that its error detection capabalities are nearly identical to those of an LFSR. The MISR is preferable to multiplexing the outputs to a single LFSR, or the use of a single LFSR for each output.

A MISR has two additional properties not found in an LFSR [28]. The first is the possibility of error cancellation. This occurs whenever errors introduced during one load cycle, are cancelled in the next cycle by other errors. This can occur before the division by the MISR and is therefore independent of the divisor polynomial. The probability of error masking due to error cancellation can be shown to be

$$P = 1/2^{r+m-1}$$

where r is the length of the MISR and m is the number of r-bit patterns





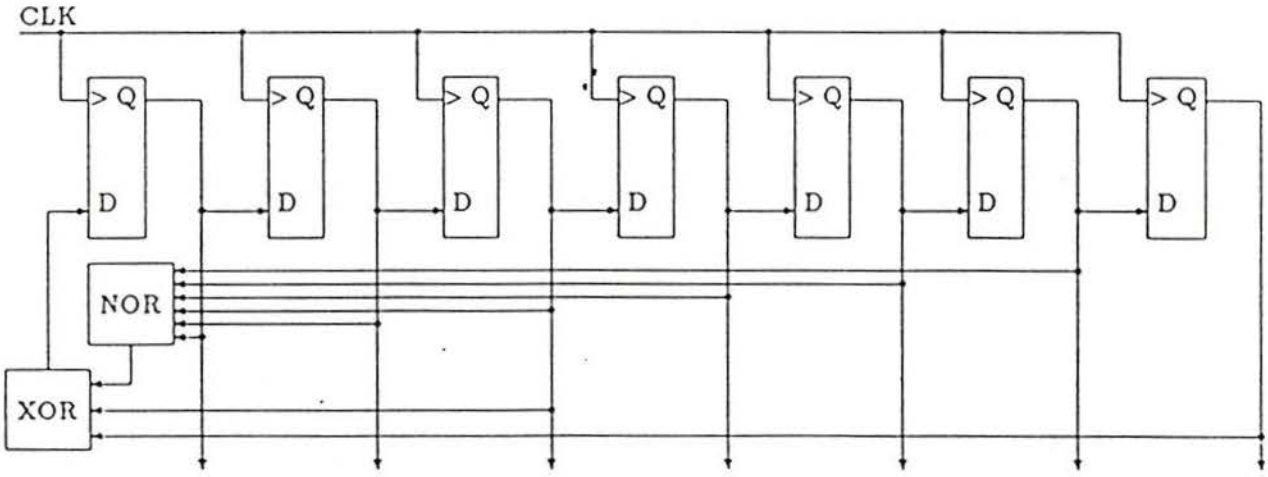


Figure 4.9: Configurations of the Modified LFSR for the normal and test modes.

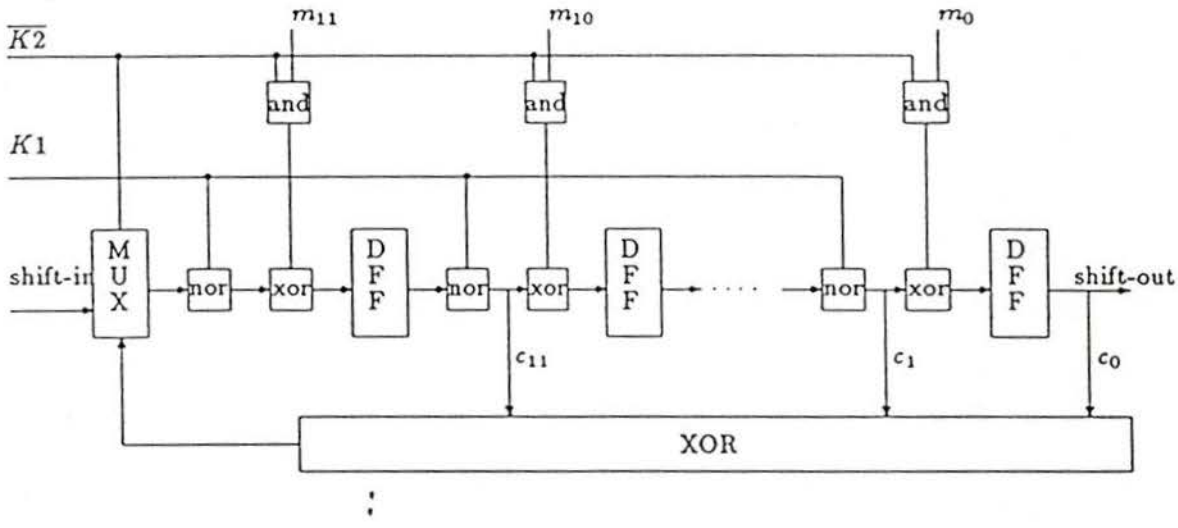


Figure 4.10: BILBO structure

presented to the MISR. This result is based on the assumption that all r bit error patterns are equally likely. In our case $r = 12$ and $m = 2^7 = 128$, resulting in a probability of

$$P = 2^{-139}$$

Thus, fault masking due to error cancellation is highly unlikely.

The second difference lies in the error detection during a single load cycle. If all errors occur during a single cycle, then the error polynomial is of degree $r - 1$ or less, if r is the length of the MISR. The divisor $d(x)$ is of degree r, and all its multiples will be of degree r or greater. Therefore, $e(x)$ cannot be a multiple of $d(x)$, and all errors will be detected. This result also implies that all single-bit errors will be detected, as expected.

A comparator is required at the output of the BILBO structure to indicate the state of the PE, i.e. faulty or not. A dynamic configuration was again chosen (Fig4.11). The advantages are the reductions in area and delay. The comparator was designed with as few gates is possible. The correct signature $\vec{S}$ is burned-in, in the sense that it is reflected in the connectivity of the outputs, $\vec{Q}$, of the LFSR as follows:

$$F = \begin{cases} 1 & \text{if } Q_i = S_i \\ 0 & \text{otherwise} \end{cases} \qquad \forall i = 1, 2, \ldots, 12$$

where the inputs (from the LFSR) are given by

$$Q_i = \begin{cases} \overline{Q_i} & \text{if } S_i = 1 \\ Q_i & \text{if } S_i = 0 \end{cases}$$

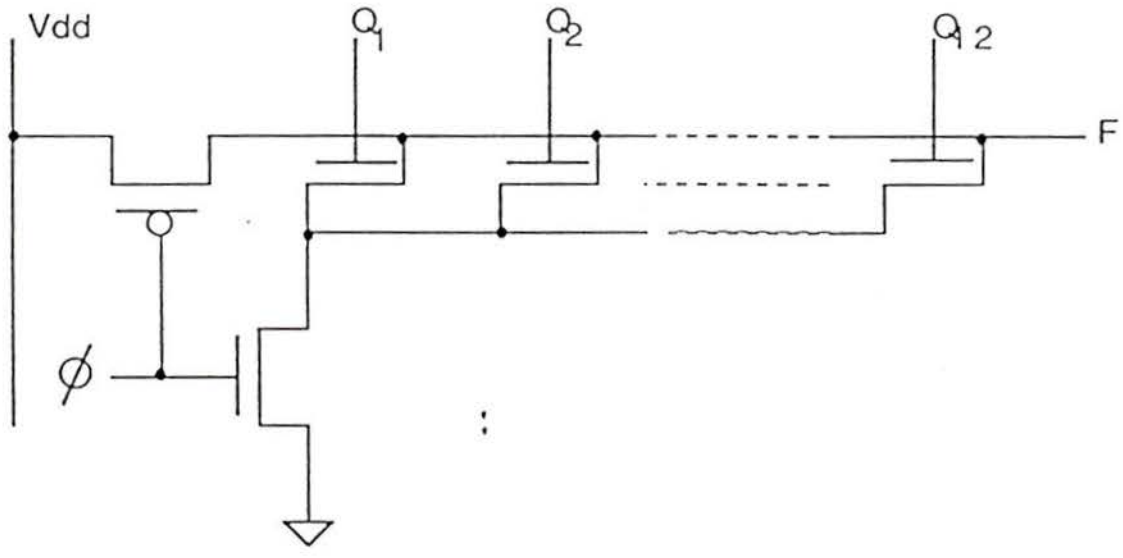


Figure 4.11: Comparator for the signature.

A C program was written to determine the correct signature. The fault-free signature S was found to be

$$S = 110111011001$$

4.2.3 Operation of the Array

Normal Mode

Before the bit map is fed into the $n \times n$ array column by column, each PE is cleared by an asynchronous clear signal. Each row of the bit map is shifted into the array in parallel. After n cycles, the shifting of the bit map is stopped, and field generation can begin.

The outerfields are generated by wavefront propagation starting at all edges, proceeding inward. This propagation can be expressed by the following recurrence relation:

$$G^l_{ij} = \overline{P_{ij} \vee \overline{G}_{ij-1}} \qquad \forall\, i,j = 1,2,...n$$

(similarly for G^t, G^b, G^r)

It should be noted that the outerfields are generated in a self-timed fashion. Thus, when the bit map is loaded the outerfields will also be generated after a maximum of n cycles. Each G^k_{ij} will become a 1, until a $P_{ij} = 1$ is encountered.

At this point, the 7-bit LFSR contains all the required information to generate the outputs. The BILBO is in the parallel load register mode with

$K1 = 1$ and $K2 = 0$. The BILBO is then changed after 1 cycle to the shift register (SR) mode with $K1 = 0$ and $K2 = 1$. The output of the BILBO is multiplexed to the propagating registers by $\overline{K1 \vee K2}$. The SR is shifted once and then disabled by using clock qualifiers shown in Fig.4.12. This qualifier is used throughout the design to control the registers.

The propagating SR's then shift the fields elements to right and bottom edges of the array. These outputs are externally accumulated by counters. The process continues until all 12 fields have been histogrammed.

It can be seen that as soon as the fields of a pattern have been generated in each PE, the next bit map can be loaded, and have its outerfields determined concurrently. The 12 fields require $12(n + 1)$ cycles to be projected, while the loading of the bit map and outerfield generation require $2n$ cycles. Thus for the first pattern, the outputs will emerge after $2n$ cycles, and every clock cycle thereafter.

The procedure described above applies to the NORMAL mode. The TEST mode is now described.

Test Mode

As before, the modified LFSR is initially cleared. The BILBO also needs to be cleared, since the fault-free signature is based on the all-zero initial state. The test mode is enabled by setting the control signals to the following values: $K1 = 0$, $K2 = 0$. The signature is then generated as follows:

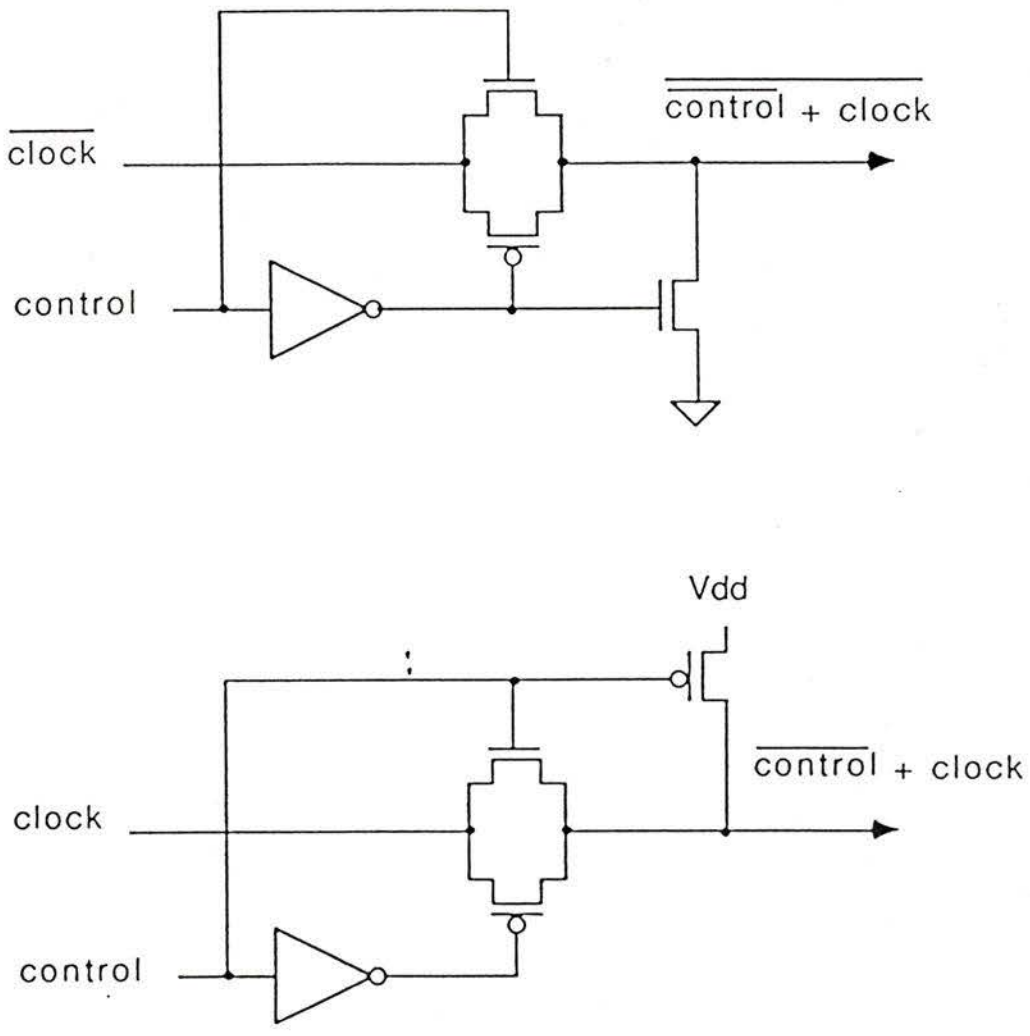


Figure 4.12: Clock qualifiers.

1. • shift LFSR data from master to slave in each register
   • precharge PPL
   • shift BILBO data from master to slave in each register
2. • shift LFSR
   • conditional discharge of PPL
   • shift BILBO
3. GOTO 1.

The resulting signature is compared and the output of the comparator is propagated to the edges.

Thus the total time required is 2^7 cycles for data compaction, n cycles for propagation, for a total of $2^k + n$ cycles.

If a fault is detected, a "1" is loaded into the propagating registers. It should be noted that a form of fault tolerance is implicit in the algorithm, and that the hardware itself is not fault tolerant. This fault tolerance is only valid if the faulty cells are scattered, and if their number is small. To determine if the chip is acceptable, the test results are propagated to the edges (right and bottom). Histograms (or projection vectors) describing the number of faulty cells in rows and columns are obtained. Without simulation on the effects of the density of faulty cells, two simple approaches to determine an operational threshold emerge:

1. If the number of faults in any row or column is greater than 1, then the chip is rejected. If more than 1 fault is allowed, then this rule is not acceptable since clustering of faults can occur, possibly changing some of the global features of the character.

2. If each faulty cell has no nearest neighbors which are also faulty, then the chip is accepted. The application of this rule will require extra logic at the cell level and a maximum number of faults per row/column is still required.

Rule 1 is adopted at this time because of its simplicity.

The next section presents the physical design of the individual PE components, and the simulation results.

4.3 Physical Design

The area of a PE is critical since it determines the degree of integration on a chip. Therefore, the individual PE's should contain a minimum of components and should be approximately square, since the array is a square mesh. Each PE consists of a modified LFSR, a BILBO, a PPL and propagating registers. Also, a small control section is included to generate and qualify the clocks. The Floorplan of a PE is shown in Fig.4.13. These PE compnents are described below.

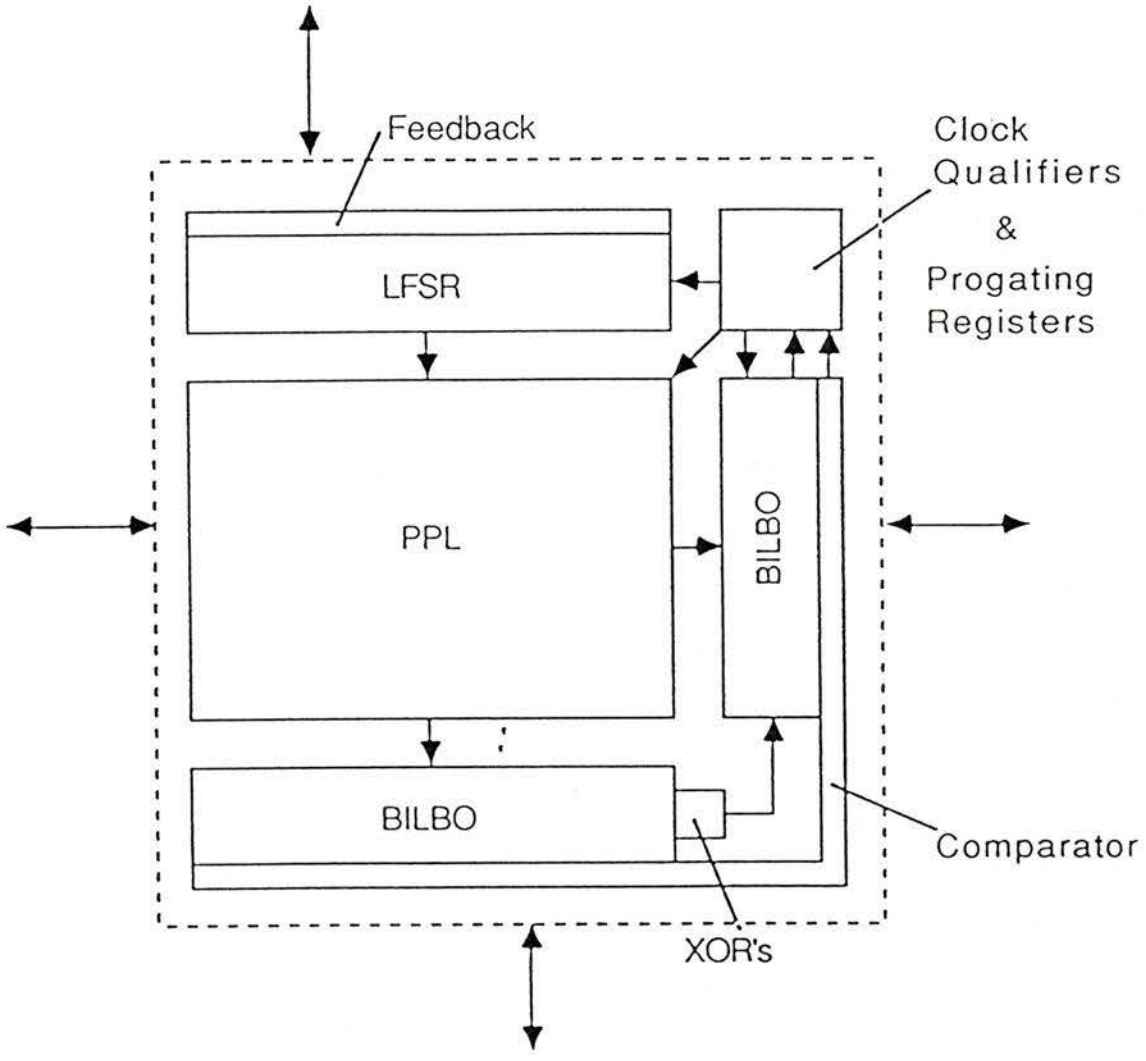


Figure 4.13: PE floorplan.

4.3.1 Registers

The registers consist of static D latches. Fig.4.14 shows a single D-latch. Each register is realized by cascading two latches in a master-slave configuration. These two latches are operated on alternate clock phases. A D-latch is sensitive to clock duration rather than clock transition, as in a D-flip-flop. Also, the D-latch is less costly in terms of area than the D-flip-flop. This design is fairly robust to clock skew, in that if the overlap is short compared to the clock period, the input will settle to the correct value. An asynchronous clear is implemented by a single n-type transistor on the slave latch.

4.3.2 Modified LFSR

Delay was the important consideration in designing the LFSR. The non-linear element and the feedback XOR-gates were optimized to improve operating speed.

The 6-input NOR-gate was chosen as the non-linear element mainly because it requires the least number of transistors. The disadvantage of this gate is its rise time, which is proportional to the number of inputs squared. This can be seen by observing that n transistors in series, each with resistance R, will have a total resistance of nR. The total capacitance (omitting parasitic capacitances) on the output is $nC_d + C_l$, where C_d is the unit drain capacitance, and C_l is the load capacitance. Thus the rise

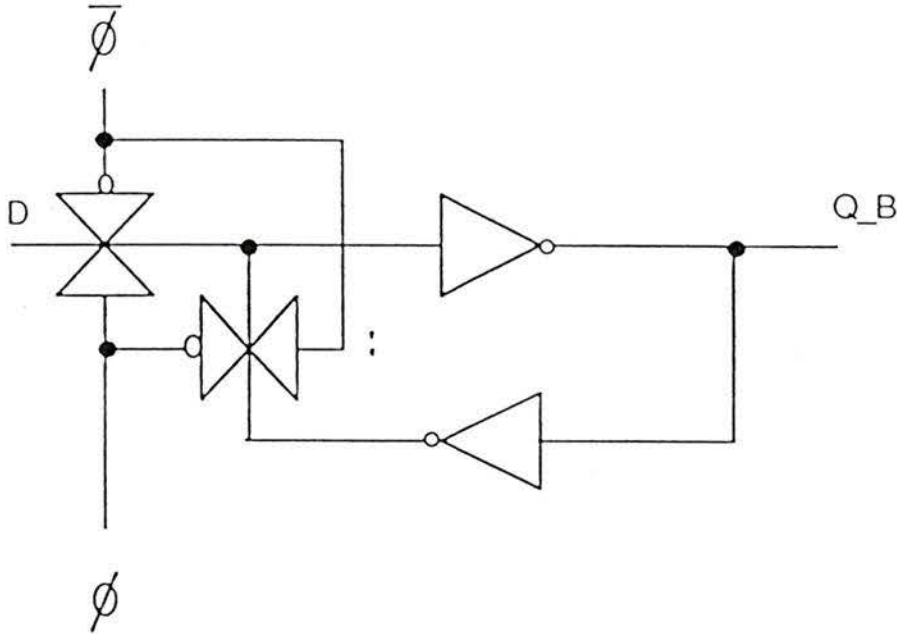


Figure 4.14: Simple D latch.

time for a NOR-gate is proportional to $n^2 C_d R$.

The switching time was decreased by increasing the width of the series transistors, the widest at Vdd [34]. The reason for the speed-up is that parasitic capacitances in the 6 p-type transistors create a capacitive load. The load is largest for the transistor nearest Vdd, and least for the output transistor. Thus the width of the series transistors were scaled accordingly to accomodate current.

The XOR-gate is shown in Fig.4.14. This implementation requires only 6 devices or 4 devices if one of the inputs' complement is available. This gate has fast switching times, compared to a standard multiple-gate implementation.

4.3.3 BILBO

The BILBO is essentially an LFSR with additional control gates. It constitutes the largest component of a PE. It uses the same XOR-gates as the LFSR. The NOR-gates follow the same approach used for the 6-input NOR. In Fig.4.10, AND gates are used to control the inputs to the MISR. To remove the delay introduced by these ANDs, they were implemented as single n-type transistors on the outputs of the PPL. Fig.4.15 shows this for a single output. When K2=1, the ouput F is low. To avoid large current, K2 also disables the precharge phase of the PPL. The hidden benefit is that the large current spikes often incurred by simultaneous precharges

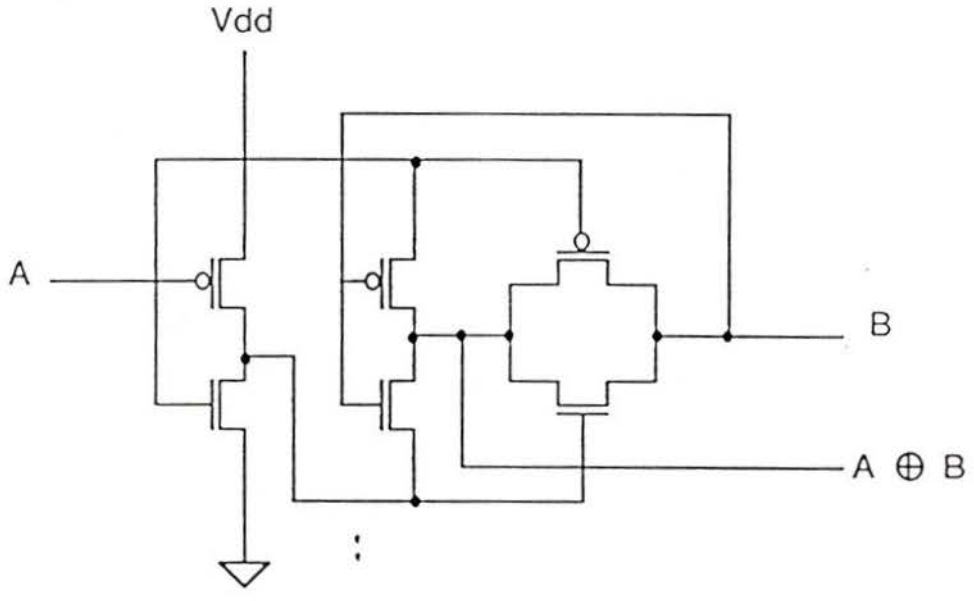


Figure 4.15: Exclusive-OR gate.

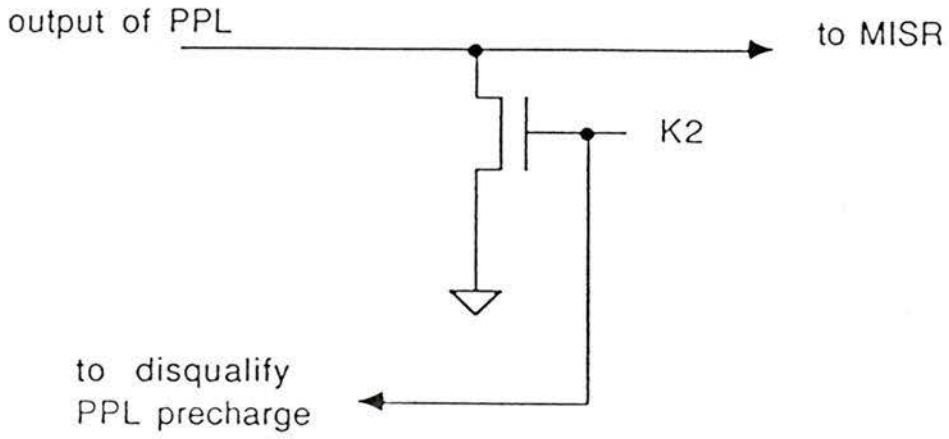


Figure 4.16: AND implementation on the ouputs of the PPL.

are eliminated. Thus the need for numerous power and ground pads is minimized. It can be seen that the NOR/XOR gates between each register determine the fastest rate at which the test will be performed.

4.3.4 Global Routing

All power and ground wires run horizontally in aluminum (first-level metal). Several pads were installed to minimize current spikes.

The four controls ($K1, K2, K3, K4$) and the master clock ϕ were laid out vertically in second-level metal, allowing the wires to overlap devices and first-level metal without shorting.

Chapter 5

Simulation Results

5.1 Fault Simulation

Fault simulation is a method used to ascertain the effectiveness of a set of test patterns. Before this simulation can be performed, a transistor level fault model for the PPL is first developed [36]. The faults considered were transistor stuck-on/ open/short, bridging, missing devices, and input/output stuck-at's. These faults are illustrated in Fig.5.1.

Fig.5.2 shows the effects of the various faults. As can be seen, a reduced set of faults is obtained because of fault equivalence and dominance. Thus, most faults can be modelled as stuck-at faults. The exceptions are the precharge/discharge transistors which can give indeterminate logic levels.

Fault simulation can now be performed. The reduced set of faults were represented at the logic gate level. This was necessary since the available simulator requires combinational logic, made up of two-input gates [37].

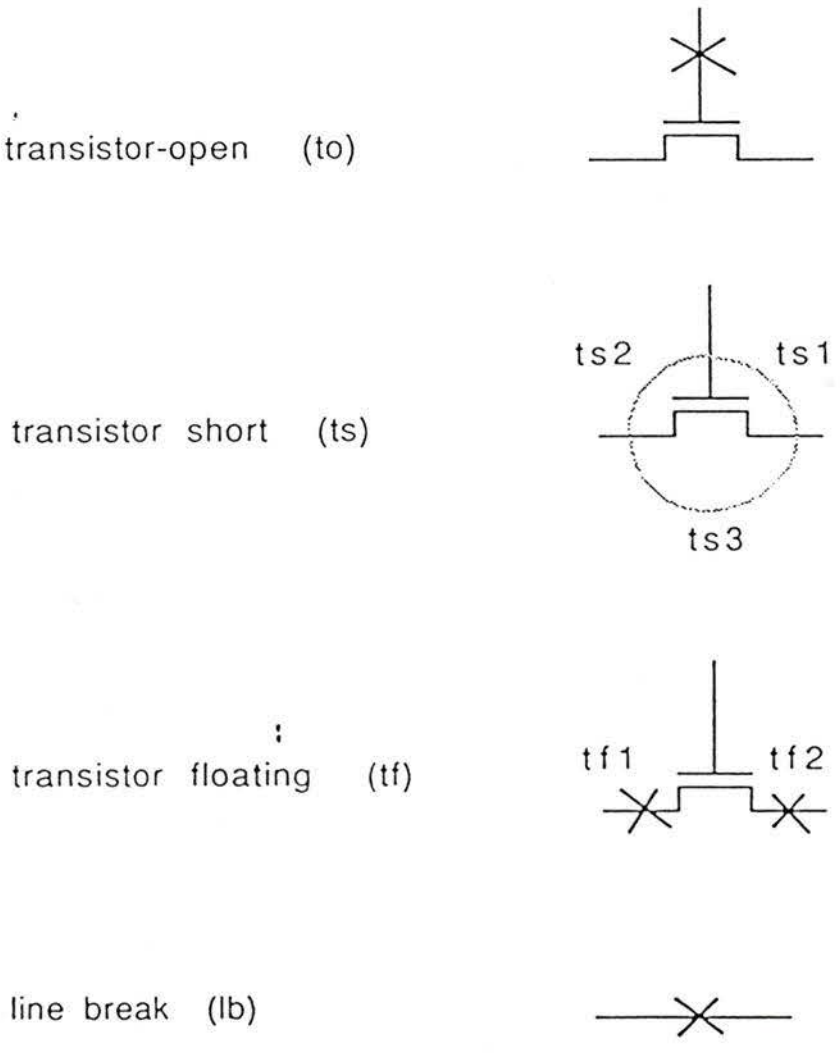


Figure 5.1: Transistor-level faults.

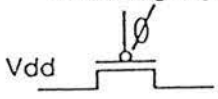
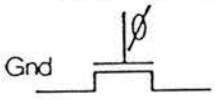
FAULT LOCATION	FAULT	FAULT EFFECTS	PPL OUTPUTS and INPUTS
Precharge gate 	ts1	indeterminate	- - -
	ts2	no precharge	S.A.0
	ts3	short to Vdd	S.A.1
Discharge gate 	ts1	indeterminate	- - -
	ts2	no precharge	S.A.1
	ts3	short to Gnd	S.A.0
	to	floating	S.A.0
	tf	missing device	input S.A.0
GND line	lb	disable 1 or more devices	output S.A.1
Output line	lb	reduce sum terms if all reduced	previous output ouput S.A.0
Input lines	lb	reduce sum terms in one or more functions	
Clock Vdd	lb		output(s) S.A.0
Clock Gnd	lb		output(s) S.A.1

Figure 5.2: Fault effects and equivalences of the PPL. Refer to Fig.4.6 for fault locations and Fig.5.1 for fault definitions.

Each PPL output was represented as a string of two-input NOR gates with one inverting input. It should be noted that this allows all multiple faults to be mapped as single faults. For example, the input P, could be operational for the first string of NOR's, and stuck-at 0 for the remainder, thus simulating a line break.

From Chapter 3, the probability of aliasing for this design is

$$P_{al} = 2^{-12} \approx .0002$$

However, simulation showed no aliasing.

Thus the design is guaranteed to detect a faulty PE. The effects of faulty BILBO circuitry should also be detected, but have not yet been simulated. Similarly, a sheme to ensure that the comparator is working, needs to be implemented.

5.2 Performance Simulation and Fabrication

The entire layout was generated manually using the VIVID package. Each composition cell was checked hierarchically using the built-in design rule checker. This ensured the correct operation of a PE.

Timing simulations were performed on all components of the design using VIVID's FACTS. FACTS is a hybrid simulator, performing slightly less detailed modeling than the device level simulator SPICE and more so than a logic level simulator. This results in the quick simulation of relatively

large circuits and still gives the designer confidence that the implementation will operate as designed.

Each component was simulated with clock rates up to 50MHz. The LFSR and BILBO operated in Test mode at a maximum clock speed of 25MHz. From chapter 4, $12(n+1)$ cycles are required for the projection of the feature fields, where n is the array size. With $n = 20$, the size on which the algorithm was simulated, the total time taken is 252 cycles per character. Operation at a frequency of 25MHz, allows the chip to extract features at a rate of 99206 characters per second. This is well beyond the present day 3000 chars./sec. optical scanning rates. This result clearly indicates the advantage of using a custom parallel architecture to perform low level image processing. The rapid rate at which the character are processed allows correct recognition to be enhanced by adding preprocessing blocks. If smoothing and thinning are performed before extraction, with each process of similar duration, the processing rate becomes 33069 characters per second. This figure is still far above the available scanning rates. The above results are an appreciable improvement over those published in reference [4], where a complete OCR hardware system reads at a rate of 300 chars/sec.

The power requirements were estimated by considering the design to consist of a number N of inverters. Since a single PE contains $\approx$ 1500 transistors, a conservative power consumption estimate can be obtained

from

$$P_{pe} = P_{inv}(f)\ N$$

With $N \approx 750$ and $P_{inv} \approx 14\mu W$ at 25MHz, then the power dissipated by a PE

$$P_{pe} \approx 10.5\ mW$$

The entire design was placed on a predefined pad format supplied by the Canadian Microelectronics Corporation (CMC). The completed chip was then translated into the Caltech Intermediate Form (CIF). This format represents the design at the mask level, necessary for fabrication. The project was then submitted to CMC for fabrication in a 3μ CMOS double-level metal process. Each PE occupies $\approx 1.4mm^2$. The chip will be available for evaluation in late 1988.

Chapter 6

Summary and Future Considerations

6.1 Summary

The design of a self-testable processor array for feature extraction has been presented. The array is capable of extracting the geometric features of a handwritten/printed character. It operates on an $n \times n$ bit map which contains the character. Each processing element has nearest-neighbor connections (4 connected) and performs low-level operations on a single pixel.

Built-In testability was incorporated in each PE using signature analysis. Single and multiple simulation has shown that no fault masking will occur. This simulation did not cover transient faults.

A single PE was implemented in 3μ double level metal CMOS technology. From this chip, an accurate prediction of performance for an entire array will be possible. The active area of the design is approximately 1.4 mm^2.

The simulation results showed a power consumption of $\approx 10.5\ mW$ and a maximal clock rate of 25 MHz. At this frequency, a 20×20 array (the resolution used for software simulation [chapter 2]) is capable of extracting features at a rate of $\approx$ 99,000 characters/second, well beyond the present day scanning rates. Thus, preprocessing stages can be incorporated to enhance correct recognition.

A further advantage of this design is that the size of the array (i.e. resolution) can be of any size, by abutting small arrays (e.g. 10×10). No changes to the design would be required and the time required for projecting the fields would be reduced. Additionally, a smaller array with higher yield, lower cost and a lesser number of pins results in a more economical and reliable chip. Fault location would also be improved.

For example, a design consisting of 30×30 PE's to minimize the digitization effects would contain in excess of 700,000 transistors. The yield for such a chip could prove to be low because of its size (e.g. increased possibility of crystal dislocations). Also, the number of pins ($\approx$ 100) would contribute to a less reliable chip. An improvement both in yield and cost would result if the chip was divided into four 15×15 arrays or nine 10×10 arrays.

6.2 Future Considerations

This thesis described a feature extractor for handwritten character recognition. This is only one stage in the complete recognition process.

The preprocessing stages, which precede feature extraction should also be implemented on custom hardware. Just as feature extraction, the smoothing and thinning stages are time consuming low-level image processing operations, unless performed in some parallel fashion. Each PE in the present design could be augmented, to become 8-connected. This would allow the array to carry out smoothing and thinning algorithms in a few clock cycles. This would also allow each PE to detect fault clustering during the self-test mode.

The output of the feature extractor can be fed to a fast processing chip, such as the TMS-320. Thus, the entire recognition system could reside on a single card, suitable for a personal computer. The main application for this system is office automation.

References

[1] S. Kahan, T. Pavlidis, H.S. Baird, 'On the recognition of characters of any font and size', *Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, pp.274-288, 1987.

[2] J.Mantas, 'An overview of character recognition methodologies', *Pattern Recognition*, vol.19, no.6, pp.425-430, 1986.

[3] J.Landau and J.Williamson, 'System unites image, text processing', *Electronics Week*, pp. 55-58, Oct. 1984.

[4] J-D.Legat,P.de Muelenaere and P.G.A.Jespers, 'A full custom integrated circuit for document analysis systems', *IEEE journal of Solid-State Circuits*, vol. SC-20, pp. 730-740, 1985.

[5] J.R.Ullmann, *Pattern Recognition Techniques*, Butterworth & Co. Ltd, London, chap. 2, 1973.

[6] M. Shridhar and A. Badreldin, 'Recognition of Isolated and Simply Connected Handwritten Numerals', *Pattern Recognition*, vol., pp. 1-

12, 1986.

[7] A. Rosenfeld, A.C. Kak, *Digital Picture Processing*, Academic Press, London, chaps. 6;, 1982.

[8] C.J. Hilditch, 'Comparison of thinning algorithms on a parallel processor', *Image and Vision Computing*, vol. pp. 115-132, 1983.

[9] D. Casasent, D. Psaltis, 'Position, rotation and scale invariant optical correlation', *Applied Optics*, vol., pp. 1795-1799, 1976.

[10] R.H. Davis and J. Lyall, 'Recognition of Handwritten Characters–a Review', *Image and Vision Computing*, vol.4, pp. 208, 1986.

[11] K.S. Fu, *VLSI for Pattern Recognition*, Springer-Verlag, New-York, chap. 1984.

[12] A.Rosenfeld, 'Parallel algorithms for image analysis'; S.Y.Kung, 'VLSI signal processing: From transversal filtering to concurrent array processing', as well as others, in *VLSI and Modern Signal Processing*, S.Y.Kung, H.J.Whitehouse and T.Kailath, Eds., Prentice-Hall, 85.

[13] *VLSI Systems and Computations*,H.T.Kung, B.Sproull and G.Steele, Eds., pp. 245-296, Computer science press, 81.

[14] P. Gachet, B. Joinnault and P. Quinton, 'Synthesizing Arrays Using DIASTOL', in *Systolic Arrays*, W. Moore, A. McCabe, R. Urquhart,

eds, IOP pub.,87

[15] R.M. Karp, R.E. Miller and S. Winograd, ' The Organization of Computations for Uniform Recurrence Relations', *Journal of the ACM*, vol., pp. 563-590, 1967.

[16] W.Porod and D.K.Ferry, 'Pattern recognition in highly integrated circuits', *Pattern Recognition*, vol.18, pp. 179-189, 1985.

[17] K.Kawakami and S.Shimazaki, 'A special purpose LSI processor using the DDA algorithm for image transformation', *11th. Annual Int. Symp. on Computer Architecture*, pp. 48-55, 1984.

[18] V.A. Kovalevsky, *Image Pattern Recognition*, Springer-Verlag, New-York, 1980.

[19] Adaptive, Learning and Pattern recognition systems, J.M.Mendel and K.S.Fu, Eds., Academic Press, New-York, 1970.

[20] M. Shridhar and A. Badreldin, 'Recognition of Isolated and Simply Connected Handwritten Numerals', *Pattern Recognition*, vol.19, pp. 1-12, 1986.

[21] R.J.Offen, *VLSI Image Processing*, Collins, London, 1985.

[22] C.D. Rogers, J.B. Rosenberg, S.W. Daniel, 'MCNC's Vertically Integrated Symbolic Design System', *IEEE 22nd Design Automation*

Conf., pp. 62, 1985.

[23] N. Weste, *Virtual Symbolic Layout, ACM/IEEE 18th Design Automation Conf. Proc.*, pp. 225, 1981.

[24] P.S. Botorff,'Test Generation and Fault Simulation', in *VLSI Testing*,T.W. Williams,Ed. , Elsevier, Amsterdam, 1987.

[25] P.K. Lala, *Fault Tolerant and Fault Testable Hardware Design*, Prentice-Hall, New-York, 1985.

[26] T.W. Williams, 'Design for Testability', in *VLSI Testing*, T.W. Williams, Ed., Elsevier, Amsterdam, 1987.

[27] R.A. Frohwerk, 'Signature Analysis: A New Digital Field Service Method', Hewlett-Packard Journal, May 1977.

[28] P.H. Bardell, W.H. McAnney, J. Savir, *Built-In-Test for VLSI: Pseudorandom Techniques*, John Wiley & Sons, New-York, 1987.

[29] W.W. Peterson, E.J. Weldon, *Error-Correcting Codes*, MIT press, 1972.

[30] J.E. Smith, 'Measures of the Effectiveness of Fault Signature Analysis', *IEEE Trans. on Computers*, Vol. C-29, pp. 510, 1980.

[31] S.W. Golomb, *Shift Register Sequences*, Holden-Day Inc., 1967.

[32] T.W. Williams, W.Daehn, M. Gruetzner, C.W. Starke, 'Comparison of aliasing errors for primitive and non-primitive polynomials', *IEEE Int. Test Conf.*, pp. 282, 1986.

[33] C. Mead, L. Conway *Introduction to VLSI Systems*, Addison-Wesley, Don Mills, 1980.

[34] N. Weste and K. Eshraghian, *Principles of CMOS Design*, Addison-Wesley, Don Mills, 1985.

[35] V.K. Agarwal, 'Easily Testable PLA Design', in *VLSI Testing*, T.W. Williams,Ed., Elsevier, Amsterdam, 1987.

[36] J.A. Abraham,'Fault Modeling in VLSI', in *VLSI Testing*, T.W. Williams,Ed., Elsevier, Amsterdam, 1987.

[37] M. Serra, personal communication.

Appendix

Error Detection in a LFSR

As mentioned in section 3.2, a binary sequence can be represented as a polynomial. Thus, if $p(x)$ is a binary sequence of length n, it can be represented as a polynomial of degree $n-1$. If we now express the characteristic polynomial of the LFSR (the divisor) $d(x)$ as a k-degree polynomial, the relationship between input is

$$p(x) = q(x)\ d(x) + s(x)$$

where $q(x)$ is the quotient.

If we let $e(x)$ represent the error sequence, the erroneous sequence $f(x)$ is given by

$$f(x) = p(x) + e(x)$$

The sequence $f(x)$ will contain undetectable errors if it satisfies

$$f(x) = \tilde{q}(x)\ d(x) + s(x)$$

where $s(x)$ is the fault-free signature.

The following theorem shows when an undetectable error will arise.

Theorem 1 *The sequences $p(x)$ and $e(x)+p(x)$ will have the same signature if and only if $e(x)$ is some multiple of $d(x)$.*

Proof:

If we add (modulo 2) the above two equations, we obtain

$$e(x) = \tilde{q}(x)\ d(x) + q(x)\ d(x) = (\tilde{q}(x) + q(x))\ d(x)$$

The residue of $e(x)$ divided by $d(x)$ is zero. We conclude for arbitrary $e(x)$ and $p(x)$, that $e(x)$ must be a multiple of $d(x)$ $\square$.

It is also seen that two identical input sequences will necessarily produce identical signatures.

The next theorem guarantees single bit error detection.

Theorem 2 *If the charasteristic polynomial $d(x)$ has two or more non-zero coefficients, all single bit errors will be detected.*

Proof:

If an error sequence $e(x)$ corresponds to a single bit error, then it is of the form

$$e(x) = x^i$$

where $0 \leq i \leq n-1$, the degree of the input sequence $p(x)$.

From Theorem 1, $e(x)$ must be some multiple of $d(x)$. But since $d(x)$ has

at least two non-zero coefficients, $e(x)$ cannot be a multiple of $d(x)$, guaranteeing detection of all single bit errors $\square$.

The following theorem describes a measure of detection effectiveness:

Theorem 3 *The probability that an LFSR of length k will not detect errors in an input sequence of length n, assuming a uniform distribution of errors, is*

$$probability\ of\ aliasing = P_{al} = \frac{2^{n-k} - 1}{2^n - 1}$$

Proof:

An input sequence $p(x)$ of length n can be any of 2^n patterns, each represented as a degree $n-1$ polynomial. The error polynomial $e(x)$ can then be represented as a polynomial of degree $n-1$ or less. Thus, there are $2^n - 1$ possible error patterns ($e(x) = 0$ is eliminated). Since $d(x)$ is of degree k, the largest degree (less than n) polynomial which is a multiple of $d(x)$ is $n-k-1$. Thus, there are $2^{n-k}-1$ undetectable errors $e(x)$ $\square$.

A further result on burst error detection is given in the following theorem.Burst error detection is defined to be the detection of d errors in n consecutive bits.

Theorem 4 *A polynomial $d(x)$ of degree r, such that*

$$d(x) = x^r + 1 + \sum_{j=1}^{r-1} a_j x^j \quad a_j \in \{0,1\}$$

is capable of detecting all (r,r) burst errors.

Proof:

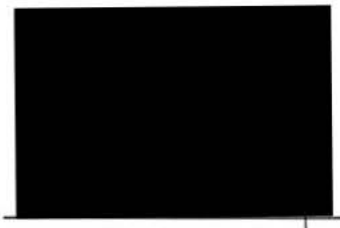
Since $e(x)$ must be a non-zero multiple of $d(x)$ to be undetectable, e(x) must have two coefficients farther apart than r. Therefore, the burst error (r,r) will always be detectable, since $e(x)$ does not have two non-zero coefficients farther apart than r $\square$.

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis:

VLSI Design of a Testable Processor Array
for Feature Extraction



Patrice Aubry

May 5, 1988

VITA

Patrice P. Aubry

Place of Birth: Compiègne, France
Date of Birth: 25^{th} December 1957

Educational Institutions Attended, with Dates of Entering and Leaving:

University of Victoria, B.C. 1981 to 1988

Degrees, Diplomas, Etc., Awarded, with Dates and Names of Institutions:

B.Sc. (Honors) 1986 University of Victoria, B.C.

Honours and Awards:

B.C. GREAT Award 1988

Publications:

1. P. Aubry, F. El Guibaly,'Testable Processor Array for Feature Extraction', *IEEE Transactions on Circuits and Systems*, Submitted May 3 1988.

2. P. Aubry, F. El Guibaly, 'Testable Systolic Array For Character Recognition', *30th Midwest Symposium on Circuits and Systems*, Syracuse, August 1987.

3. P. Aubry, F. El Guibaly, 'VLSI Design of a Character Recognition Algorithm', *IEEE Pacific Rim Conf. on Communications, Computers and and signal processing*, Victoria, BC, June 1987.