

Skeletonization of Binary Patterns:

A Proposed Algorithm and a Multiprocessor Network

Nabil Jean Naccache

A Thesis

in

The Department

of

Computer Science

**Presented in Partial Fulfillment of the Requirements
for the degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada**

April 1984

© Nabil Jean Naccache, 1984

Abstract

Skeletonization of Binary Patterns:

A Proposed Algorithm and a Multiprocessor Network

Nabil Jean Naccache

In this thesis, we review published literature on algorithms used for skeletonizing binary patterns. Then, we propose another skeletonization algorithm, called the Safe-Point Thinning Algorithm (or SPTA). This algorithm is shown to be very fast and to produce skeletons of good quality. The SPTA has two labelling techniques: the Single Integer Labelling Technique (or SILT) and the Four Integers Labelling Technique (or FILT). Each of these techniques allows the user to reconstruct a pattern from the skeleton obtained. The reconstructed pattern is shown to be very similar to the original pattern from which the skeleton was derived. We also give the experimental results comparing the reviewed skeletonization algorithms to the SPTA. The reconstruction techniques are also tested and compared. Furthermore, we propose a multiprocessor network, called SKELNET, for skeletonization algorithms. This network is described in detail, then simulated and experimentally tested.

Acknowledgments

First, I would like to deeply thank my supervisor, Dr Rajjan Shinghal, for the many hours spent on this thesis since it was started in late 1982. His help, suggestions, criticism and encouragements have been the key to the successful results which were achieved.

I am also very grateful to my friend Michael Payette of Concordia University, with whom the long constructive discussions helped me in the development of the Safe-Point Thinning Algorithm proposed in this thesis.

I also wish to thank Dr Godfried Toussaint and Dr Leslie Olivier of McGill University. Dr L. Olivier found that there were instances in which Stefanelli-Rosenfeld's version of Hilditch's algorithm (Chapter II) cause excessive erosion. It was reported to me by Dr G. Toussaint. This lead me to further investigate Hilditch's algorithm.

I am also very thankful to Dr J.W. Atwood of Concordia University, whose constructive suggestions helped me in the design and the simulation of the multiprocessor network (SKELNET) described in Chapter IV.

I am deeply grateful to my colleague and friend Andrew Pospiech, former student at Concordia University, who

largely contributed in the hardware design of the network SKELNET. The many hours spent together were strongly appreciated.

Finally, I wish to thank all my family and friends whose moral support helped in the achievement of the present thesis.

Table of Contents

Abstract	1
Acknowledgments	ii
Table of Contents	iv
Chapter I. Introduction	
I. 1. Fundamental Notions of Skeletonization	1
I. 2. Fundamentals of Skeletonization Algorithms	5
I. 3. Outline of the Thesis	6
Chapter II. Historical Review of Skeletonization Algorithms	
II. 1. Definitions	8
II. 2. Arcelli's Algorithm (ARCL)	11
II. 3. Bel-Lan's and Montoto's Algorithm (BELA)	14
II. 4. Beun's Algorithm using Orthogonal Neighbours (BEUO)	19
II. 5. Beun's Algorithm using Saraga-Woollon's Criteria (BEUS)	22
II. 6. Hilditch's Algorithm (HILD)	25
II. 7. Ma's and Yudin's Algorithm (MAYA)	30
II. 8. Pavlidis's Classical Algorithm (PAVC)	36
II. 9. Pavlidis's Basic Algorithm (PAVB)	40
II.10. Pavlidis's Algorithm with Reconstruction Ability (PAVR)	45
II.11. Stefanelli-Rosenfeld's Version of Hilditch's Algorithm (SRVH)	50
II.12. Stefanelli-Rosenfeld's Algorithm with 4 Scans per Pass (SR4S)	53

II.13. Stefanelli-Rosenfeld's Algorithm with	
2 Scans per Pass (SR2S)	56
II.14. Tamura's Algorithm ensuring	
Four-Connectedness (TM4F)	59
II.15. Tamura's Algorithm ensuring	
Eight-Connectedness (TM4E)	61
II.16. Tamura's Algorithm with	
2 Scans per Pass (TM2E)	62
II.17. Zhang-Suen's Algorithm (ZHAN)	63
II.18. Other existing Skeletonization Algorithms	66
Chapter III. The Safe-Point Thinning Algorithm	
III.1. Fundamental Definitions	69
III.2. Explanation of the Safe-Point Concept	71
III.3. The SPTA Process	78
III.4. The Labelling Techniques of SPTA	83
III.5. Implementation Techniques of SPTA	92
Chapter IV. Experimental Results of	
Skeletonization Algorithms	
IV. 1. The Smoothing Technique	96
IV. 2. Experimental Performance of the	
Skeletonization Algorithms	105
IV. 3. Comparison of the Reconstruction Techniques	114
IV.4. Excessive Erosions in the	
Skeletonization Algorithms	123

Chapter V. A Multiprocessor Network for
Skeletonization Algorithms

V. 1. Definition of Sequential and Parallel Skeletonization Algorithms	125.
V. 2. The Fundamental Requirements of the Multiprocessor Network	127
V. 3. The Block-Level Architecture of SKELNET	131
V. 4. The Skeletonization Process of SKELNET	133
V. 5. The Pin-Description of SKELNET	136
V. 6. Interaction between the Processor Units	142
V. 7. Software Simulation of SKELNET	144
V. 8. Expandability of SKELNET	151
Chapter VI. Concluding Remarks	
VI. 1. General Conclusion	153
VI. 2. Future Work	154
References	156

Chapter I

Introduction

I.1. Fundamental Notions of Skeletonization

A binary pattern (see Figure I.1) consists of a set of white points, (the background of the pattern) and a set of dark points (the foreground of the pattern). Skeletonization of binary patterns consists of iteratively deleting the dark points (i.e., changing them to white) along the edges of a pattern, until the pattern is thinned to a line drawing (Figure I.2). This line drawing (called a skeleton) must preserve the connectedness and the shape of the original pattern. As shown by Davies and Plummer (1981), the skeleton of a pattern may not be unique. The ideal skeleton lies along the medial axis of the original pattern (Davies et al., 1981; Pavlidis, 1982a). Skeletonization reduces the memory space required for storing the essential structural information of a pattern. Moreover, it also simplifies the data structures required in processing the pattern (Davies et al., 1981). In the past, many researchers have proposed different skeletonization algorithms. These algorithms found applications in Optical Character Recognition (Beun, 1973; Davies et al., 1981; Deutsh, 1972; Pavlidis, 1982a; Stefanelli and Rosenfeld, 1971), in Chromosome Analysis (Hilditch, 1969), and in Fingerprint Classification (Moayer and Fu, 1976; Rao, Prasada and Sarma, 1974). Some researchers (Davies et al.,

1981; Pavlidis, 1982a; Rosenfeld and Pfaltz, 1966) have developed skeletonization algorithms in which a skeleton retains enough information so that a pattern similar to the original pattern can be reconstructed using this information.

The usefulness of such ability of reconstruction has been discussed by Davies and Plummer (1981). For example, a data base of patterns may be stored on disc in their thinned form, thus using less storage. If the full patterns are then required for processing, these skeletons are read into main memory and the desired patterns are reconstructed. Davies et al. (1981) claim that given an ideal skeleton, a fairly simple reconstructing algorithm can ensure that the reconstructed pattern is identical to the original pattern. However, ideal skeletons are practically difficult to achieve. Thus, in general, a reconstructed pattern is usually only nearly identical to the original pattern from which the skeleton was derived.

Figure I.1. A specimen of a binary pattern.
A '*' represents a dark point.

Figure I.2. The skeleton of the specimen pattern of Figure I.1. A '*' represents a point of the skeleton; a '-' represents a deleted point. This skeleton was obtained using the Safe-Point Thinning Algorithm (SPTA) described in Chapter III.

I.2. Fundamentals of Skeletonization Algorithms

Skeletonization usually consists of executing many passes over the pattern, where in each pass a few dark points are flagged. At the end of each pass, all flagged points are deleted (i.e., they are set to white). The points to be flagged must be edge-points; i.e., dark points along the edges of the pattern. However, these edge-points must be such that their deletion (1) does not remove end-points (i.e., dark points at the open extremities of a stroke); (2) does not break the connectedness of the pattern; and (3) does not cause excessive erosion (i.e., a stroke must not be iteratively deleted). The skeletonization stops when after a given pass no dark points are deleted. Researchers (e.g., Arcelli, 1979; Bel-Lan and Montoto, 1981; Beun, 1973; Hilditch, 1969; Hilditch, 1983; Ma, 1983; Naccache and Shinghal, 1984b; Pavlidis, 1982a; Rosenfeld and Pfaltz, 1966; Stefanelli and Rosenfeld, 1971; Tamura, 1978; Zhang and Suen, 1984) usually differ in the manner in which they conduct the tests to meet the above criteria. There may be some common aspects in the approaches of these researchers, but no two of their algorithms can be considered identical. Each researcher's success in skeletonization thus depends on the goodness of the tests conducted. Moreover, should we need to reconstruct the pattern from the skeleton, enough information must be retained in the skeletonization process

such that this information can be used for reconstruction. Retaining this information can cause the skeletonization process to slow down, and it can affect the quality of the skeletons produced.

I.3. Outline of the Thesis

In this thesis, we give a historical review of skeletonization algorithms, and then we propose a skeletonization algorithm called the Safe-Point Thinning Algorithm. All these algorithms are experimentally tested and compared. Finally, we propose a multiprocessor network for skeletonization algorithms.

In Chapter II, we review 16 well-known skeletonization algorithms. For each algorithm we give fundamental definitions and an informal description.

In Chapter III, we propose a skeletonization algorithm for binary patterns, called the Safe-Point Thinning Algorithm (or SPTA). The objective is to develop a skeletonization algorithm which is fast, which prevents excessive erosion, which does not alter the shape information of the original pattern, and which has reconstruction ability. The SPTA and the techniques it employs for reconstruction are described both informally and formally.

In Chapter IV, we give the experimental results obtained by implementing the 16 algorithms described in Chapter III and our SPTA of Chapter IV. The results are then compared and discussed. One of the purposes of the discussion is to compare the SPTA with these 16 algorithms.

In Chapter V, we propose a multiprocessor network for skeletonization algorithms, called SKELNET. The purpose of SKELNET is to speed-up the skeletonization process. This network has the advantage of being expandable and of being financially inexpensive.

Finally, in Chapter VI we give our concluding remarks on the thesis and on the future work which can be done.

Chapter II

Historical Review of Skeletonization Algorithms

II.1. Definitions

Before presenting our historical review, we first give some fundamental definitions which will be used throughout this thesis. Although the terms skeletonization and thinning are not absolutely synonymous, we may use them both interchangeably, depending on the context at this juncture.

The 8-neighbours of a point p of a pattern are defined to be the 8 points adjacent to p (points n_0 to n_7 in Figure II.1). Points n_0 , n_2 , n_4 and n_6 are also referred to as the 4-neighbours of p . A pattern is said to be j -connected (j is equal to four or eight) if between any 2 dark points p_0 and p_n there exists a path of dark points $p_0 p \dots p_{i-1} p_i \dots p_n$ such that p_{i-1} is a j -neighbour of p_i , for $1 \leq i \leq n$.

In most skeletonization algorithms, during any given pass over the pattern, the dark points which are candidate for deletion are not immediately deleted; i.e., they are not immediately set to white. During a first stage, the points are flagged. Then, at the end of the pass, all the flagged points are deleted. Thus, a point p of the pattern may have any of the following three statuses: unflagged dark, flagged dark, or white. The deletion of a dark point p thus depends on the status of its 8-neighbours. The conditions for

deletion are generally represented by a set of 3x3 windows (that we shall call the fundamental windows), or by their counter-clockwise rotation (also denoted c-c rotation) by 90, 180 and 270 degrees.

It is assumed in this thesis that a pattern to be skeletonized is stored in a 2-dimensional matrix, and that while skeletonizing it is scanned rowwise from left to right and from top to bottom.

Reviewing the published literature, we found 16 well-known skeletonization algorithms. We are now going to present them one by one. The 16 algorithms are presented in alphabetical order of the names of the researchers who proposed them.

n3	n2	n1
n4	p	n0
n5	n6	n7

Figure II.1. A point p and its neighbourhood.
 The points $n0$ to $n7$ are called the
 8-neighbours of p . The points
 $n0$, $n2$, $n4$ and $n6$ are also referred
 to as the 4-neighbours of p .

II.2. Arcelli's Algorithm (Arcelli, 1979) --- ARCL (the abbreviation ARCL is the name by which this algorithm shall be referred to for the rest of this thesis)

Arcelli defines four kinds of edge-points: right (a dark point having n_0 white), top (having n_2 white), left (having n_4 white) and bottom (having n_6 white) edge-points. Also, Arcelli defines an end-point as a dark point having at most one dark 8-neighbour.

During any pass over the pattern, a dark point p is flagged if it satisfies all of the following conditions:

ARCL_1: it is an edge-point.

ARCL_2: it is not an end-point. This is called the end-point test.

ARCL_3: its neighbourhood satisfies the following Boolean expression:

$$(\overline{n_0} \cdot n_1 \cdot \overline{n_2}) + (\overline{n_2} \cdot n_3 \cdot \overline{n_4}) + (\overline{n_4} \cdot n_5 \cdot \overline{n_6}) + (\overline{n_6} \cdot n_7 \cdot \overline{n_0}) = \text{FALSE}$$

where a Boolean variable has value TRUE if the corresponding point is dark, and has value FALSE otherwise. This condition is called the break-point test; it is done to ensure that the deletion of p would not break the connectedness of the pattern.

ARCL_4: its neighbourhood satisfies the predefined Boolean expressions of Table II.1. This table indicates that if, say, p is a right edge-point, then the Boolean expression

$n2.n4.n6$ must be false. This condition is also a break-point test.

Then, at the end of ~~the~~ pass, all flagged points are deleted. The passes are repeated until there are no points flagged in a pass. The set of remaining dark points constitutes the skeleton of the original pattern.

Kind of edge-point	Boolean expression tested in ARCL_4
RIGHT	$n2 \cdot \overline{n4} \cdot n6 = \text{FALSE}$
TOP	$n4 \cdot \overline{n6} \cdot n0 = \text{FALSE}$
LEFT	$n6 \cdot \overline{n0} \cdot n2 = \text{FALSE}$
BOTTOM	$n0 \cdot \overline{n2} \cdot n4 = \text{FALSE}$

TABLE II.1.

The Boolean expression to be tested in ARCL_4, of algorithm ARCL. For example, if the point p is a right edge-point, then the Boolean expression to be tested is $n2 \cdot \overline{n4} \cdot n6 = \text{FALSE}$.

II.3. Bel-Lan's and Montoto's Algorithm (Bel-Lan and Montoto, 1981) --- BELA

Bel-Lan and Montoto define 4 kinds of edge-points: right (having n_0 white and n_4 dark), top (having n_2 white and n_6 dark), left (having n_4 white and n_0 dark) and bottom (having n_6 white and n_2 dark) edge-points. Initially, all white points are labelled with value 0 and all dark points with value 1. In BELA, a pass over the pattern consists of four scans, where in each scan only one kind of edge-points is tested for deletion. Thus, in Scan 1 left edge-points are detected, in Scan 2 right edge-points, in Scan 3 top edge-points, and in Scan 4 bottom edge-points. In BELA, the passes are sequentially numbered 1, 2, 3, ... During any pass i over the pattern, all the dark points are labelled according to the decision tree of Figure II.2. From that decision tree, one can see that the points are never flagged but directly deleted from the pattern. Therefore, there is no need for an extra scan at the end of the pass in order to delete dark points.

During any pass over the pattern, a dark point p is deleted (i.e., it is set to white) if it satisfies all of the following conditions:

BELA_1: it is an edge-point.

BELA_2: its neighbourhood does not match the two fundamental windows of Figure II.3, or their c-c rotation by 90, 180 and

270 degrees, with respect to Table II.2. This table shows that if, say, the point p is a right edge-point, then its neighbourhood is matched against windows (i) and (ii), and their c-c rotation by 270 and 180, respectively. This condition performs the end-point test and the break-point test simultaneously.

As mentioned above, the labelling technique used by BELA prevents the algorithm from flagging dark points; these points are directly deleted. Nevertheless, an extra scan is still required: during this scan, if no dark non-edge-points are detected, then the skeletonization process stops. Thus, at the end of BELA, the skeleton consists of a set of dark points having value greater than or equal to 1. It should be noted that the algorithm requires two matrices: one for storing the pattern, the other for copying the points with their new label as the first matrix is scanned. At the end of each pass, the second matrix is copied back into the first one.

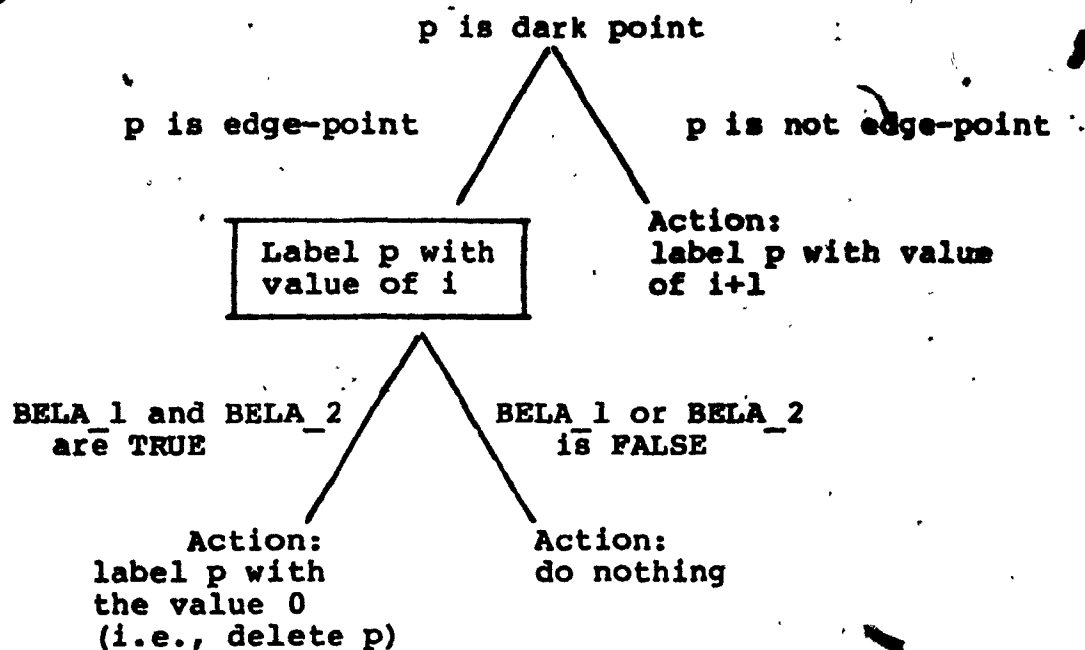


Figure II.2. The decision tree used by BELA to label a dark point p . 'i' indicates the current pass number, where any one pass consists of 4 scans, each scan testing only one kind of edge-points (right, top, left or bottom edge-points).

w		*
w	p	
w	w	w

(i)

w	*	
w	p	*
w		

(ii)

Figure II.3. The two fundamental windows used in condition BELA₂ of algorithm BELA. A '*' represents a dark point, and a 'w' is a "don't care" condition; i.e., the darkness or whiteness of the corresponding point is immaterial.

Kind of edge-point	Rotation, in degrees, of windows of Figure II.2.	
	window (i)	window (ii)
RIGHT	0, 270	0, 180
TOP	0, 90	90, 270
LEFT	90, 180	0, 180
BOTTOM	180, 270	90, 270

TABLE II.2.

The windows of Figure II.2 to be used in condition BELA₂, of algorithm BELA. For example, if p is a right edge-point, then the condition refers to window (i) of Figure II.2 and to its c-c rotation by 270 degrees, and to window (ii) and to its c-c rotation by 180 degrees.

II.4. Beun's Algorithm using Orthogonal Neighbours (Beun, 1971) --- BEUO

In BEUO, Beun defines an edge-point as a dark point having at least one white 4-neighbour, and an end-point as a dark point having at most one dark 8-neighbour.

During any pass over the pattern, a dark point p is flagged if it satisfies all of the following conditions:

BEUO_1: it is an edge-point.

BEUO_2: it is not an end-point.

BEUO_3: its neighbourhood does not match the fundamental window (i) of Figure II.4 or its c-c rotation by 90 degrees, nor does it match the fundamental window (ii) or its c-c rotation by 90, 180 and 270 degrees. This condition conducts the break-point test.

BEUO_4: its neighbourhood does not match the fundamental window of Figure II.5. This condition is performed to prevent excessive erosion; it is called the excessive erosion test.

Then, at the end of the pass all flagged points are deleted. The passes are repeated until there are no points flagged in a pass. The set of remaining dark points constitutes the skeleton of the original pattern.

x	x	x
	p	
y	y	y

(i)

x		*
x	p	
x	x	x

(ii)

Figure II.4. The fundamental windows used in condition BELA₃ of algorithm BELA. A '*' represents a dark point. At least one 'x' and one 'y' must be dark.

	p	*
	*	*

Figure II.5. The fundamental window used in condition BELA 4, of algorithm BELA, to conduct the excessive erosion test.

II.5. Beun's Algorithm using Saraga-Woollons' Criteria

(Beun, 1971) --- BEUS

In BEUS, Beun defines an edge-point as a dark point whose neighbourhood matches the fundamental window of Figure II.6 or its c-c rotation by 90, 180 and 270 degrees. This edge-point criterion is called the Saraga-Woollons criterion (Saraga and Woollons, 1968).

During any pass over the pattern, a dark point p is flagged if it satisfies all of the following conditions:

BEUS_1: it is an edge-point.

BEUS_2: its deletion does not cause the introduction of a loop; i.e., a white point whose 8-neighbours are all dark.

BEUS_3: its neighbourhood does not match the fundamental window (i) of Figure II.4 or its c-c rotation by 90 degrees, nor does it match the fundamental window (ii) or its c-c rotation by 90, 180 and 270 degrees. This test conducts the break-point test, as it is identical to condition BEUO_3 of algorithm BEUO.

BEUS_4: its neighbourhood does not match the fundamental window of Figure II.5. This condition conducts the excessive erosion test, as it is identical to condition BEUO_4 of algorithm BEUO.

Then, at the end of the pass all flagged points are deleted. The passes are repeated until there are no points

flagged in a pass. The set of remaining dark points constitutes the skeleton of the original pattern.

x	y	y
x	p	y
x	y	y

Figure II.6. The fundamental window of the Saraga-Woollon criterion, for edge-point detection (Saraga et al., 1968). At most one 'x' must be dark, and at least three 'y's must be dark.

II.6. Hilditch's Algorithm (Hilditch, 1969) --- HILD

Hilditch defines an edge-point as a dark point having at least one white 4-neighbour, and an end-point as a dark point having at most one dark 8-neighbour. Hilditch also defines the crossing-number $X(p)$ of a point p to be the following:

$$X(p) = \sum_{i=0}^3 b_i, \quad b_i = 1 \text{ if } n(2i) \text{ is white AND} \\ \text{either } n(2i+1) \text{ or } n(2i+2) \text{ is dark} \\ b_i = 0 \text{ otherwise}$$

One can notice that if the crossing-number of p is not equal to 1, then p is a break-point.

In any given pass, a dark point p is flagged if it satisfies all of the following conditions:

HILD_1: it is an edge-point.

HILD_2: it is not an end-point.

HILD_3: $N(p) \geq 1$, where $N(p)$ is the number of unflagged dark 8-neighbours of p . This test prevents dark points at "tip of a thin line" or in "approximately circular subsets" from being iteratively deleted, as stated by Hilditch (1969);

HILD_4: it is not a break-point; i.e., $X(p) = 1$;

HILD_5: either n_2 is unflagged, or $X_2(p) = 1$, where $X_2(p)$ is the crossing-number of p if we temporarily assume that n_2 is white. This test prevents excessive erosion.

HILD_6: either n_4 is unflagged, or $X_4(p) = 1$. This test.

also prevents excessive erosion, as it is similar to HILD_5.

Then, at the end of the pass all flagged points are deleted. The passes are repeated until there are no points flagged in a pass. The set of remaining dark points constitutes the skeleton of the original pattern.

Investigating HILD, we noticed that the condition HILD_3 always returns TRUE, except for 16 configurations around the point p. For test HILD_3 to return FALSE, none of the points n0 to n7 must be unflagged dark points. Moreover, if any of these points are to be flagged, they can only be n1, n2, n3 or n4, because these are the only points the algorithm would have encountered travelling rowwise in a left to right, top to bottom scanning sequence. Thus the 16 configurations for which HILD_3 returns FALSE are these: when n5, n6, n7, n0 are white points, and n1, n2, n3, n4 are either white or flagged dark points. These 16 configurations are shown in Figure II.7 as windows (i) to (xvi). We also noticed that for the 10 windows (i) to (x) when HILD_3 returns FALSE, then at least one other test of the algorithm also returns FALSE. For example, for window (i), HILD_2 will also return FALSE. Thus, we see that there exists only 6 windows (xi) to (xvi) where condition HILD_3 is crucial. We empirically found that for the 5 windows (xi) to (xv) the point p will be flagged only if each of the

windows is surrounded by white points, (in which case the whole window will ultimately be deleted). In Optical Character Recognition, such configurations are not expected too often. In fact, they can be considered as noise we would want to be deleted, anyway. For the remaining configuration (xvi), we empirically found that there were only 5 combinations of points to the left and top of (xvi) for which the point p will get flagged, and ultimately the configuration will be deleted. These 5 combinations are shown in Figure II.8.

Hoping that such combinations occurred rarely, we implemented a modification to HILD wherein we heuristically deleted condition HILD_3. Tested on a data base of hand-printed characters, described later in Chapter IV, we observed that algorithm HILD always gave the same skeletons as the modified algorithm HILD (Naccache and Shinghal, 1984a). However, the saving in computation time was only marginal (approximately 5%).

We certainly do not recommend that other researchers blindly delete condition HILD_3. One must be familiar with the nature of one's data before planning to delete test HILD_3. Our purpose in discussing all this was to point out to other researchers that test HILD_3 may be deleted, but only with caution.

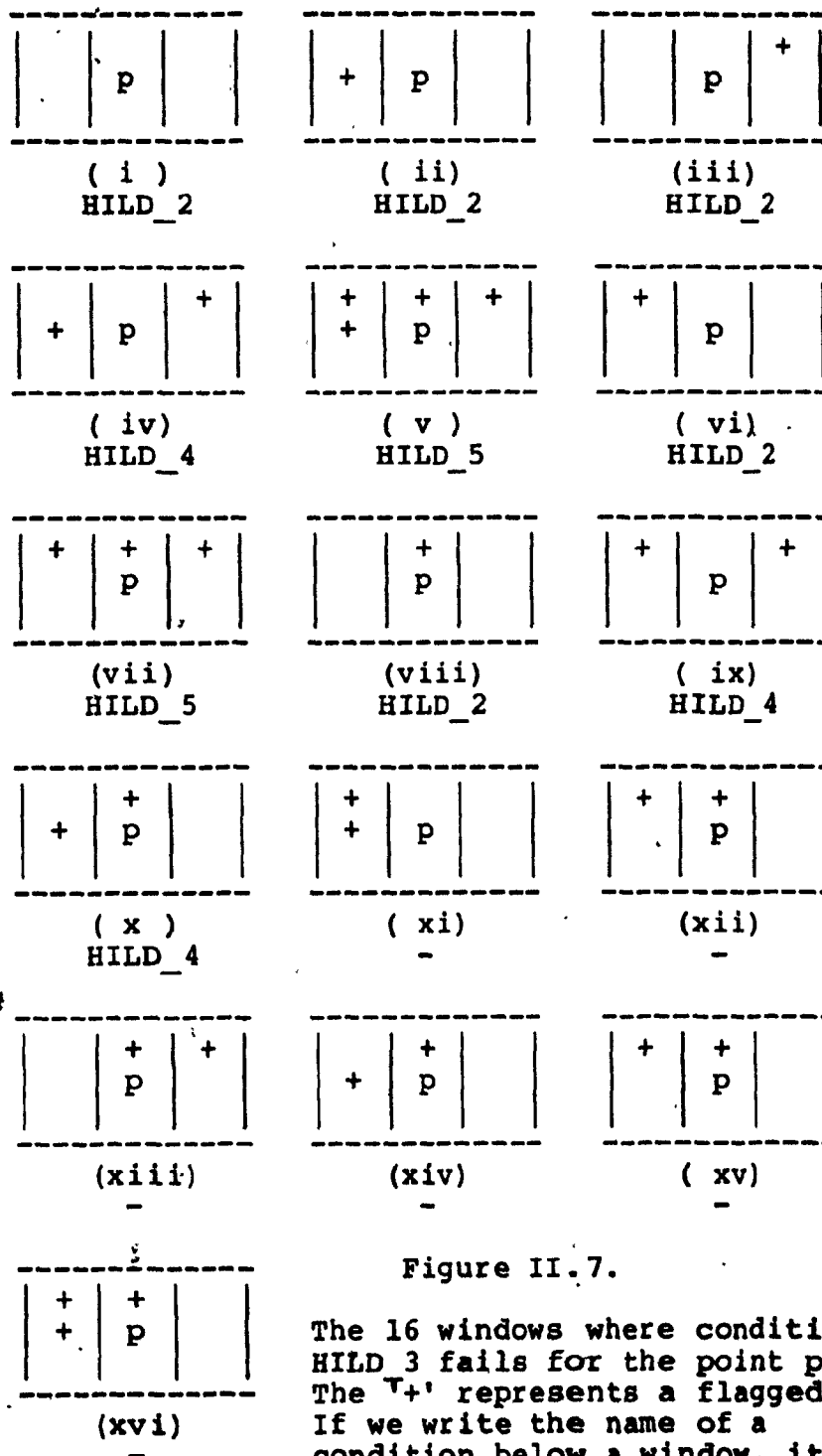


Figure II.7.

The 16 windows where condition HILD 3 fails for the point p. The '+' represents a flagged point. If we write the name of a condition below a window, it indicates that the condition also fails; e.g., for window (i) condition HILD_2 also fails.

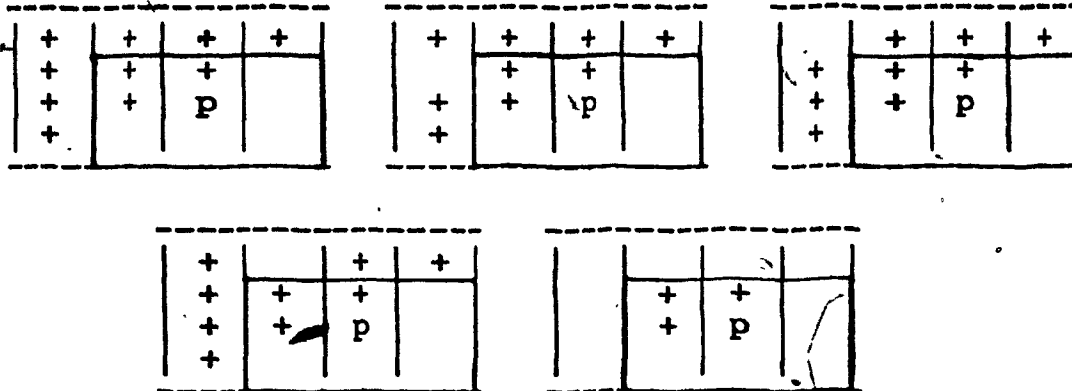


Figure II.8. The 7 cases of window (xvi) of Figure II.7 where point p will be flagged. The '+' represents a flagged point. Points below and right of these 5 configurations are "don't care"s.

II.7. Ma's and Yudin's Algorithm (Ma, 1983) --- MAYA

Ma and Yudin define an edge-point as a dark point having at least one white 4-neighbour.

During any pass over the pattern, a dark point p is flagged if it satisfies all of the following conditions:

MAYA_1: it is an edge-point.

MAYA_2: it is not an end-point. Moreover, this 8-neighbour must have at least two dark 8-neighbours.

MAYA_3: its neighbourhood does not match the fundamental window (i) of Figure II.9 or its c-c rotation by 90 degrees, nor does it match the fundamental window (ii) or its c-c rotation by 90, 180 and 270 degrees. This condition conducts the break-point test.

MAYA_4: it is not a loop-point; i.e., it does not match any of the two fundamental windows of Figure II.10;

MAYA_5: it is not a corner-point; i.e., it does not match the fundamental window of Figure II.11 or its c-c rotation by 90, 180 and 270 degrees.

MAYA_6: its deletion does not cause excessive erosion; i.e., its neighbourhood does not match the fundamental window (i) of Figure II.12. However, if the neighbourhood of p matches window (i), then p is flagged and $n0$ is set to dark. The resulting configuration is window (ii) of Figure II.12.

MAYA_7: it is an isolated point; i.e., its 8-neighbours are all white.

Then, at the end of the pass all flagged points are deleted. The passes are repeated until there are no points flagged in a pass. The set of remaining dark points constitutes the skeleton of the original pattern.

Taking a closer look at condition MAYA_4, one can see that the neighbourhood of an edge-point p will never match window (i) of Figure II.10. Indeed, if p is an edge-point then at least one of its 4-neighbours must be white, which is never true for window (i) of Figure II.10. Also, looking at condition MAYA_7, one can see that if p has only white 8-neighbours, then conditions MAYA_1 through MAYA_6 will return TRUE, anyway. Thus, the isolated-point condition MAYA_7 is redundant and should not be considered as one of the conditions for flagging a point. It should also be noted that the condition MAYA_3 is identical to condition BEUO_3 of algorithm BEUO and to condition BEUS_3 of algorithm BEUS, for break-point test.

x	x	x
	p	
y	y	y

(i)

x		*
x	p	
x	x	x

(ii)

Figure II.9. The fundamental windows used in condition MAYA₃, of algorithm MAYA. A '*' represents a dark point. At least one 'x' and one 'y' must be a dark point.

w	*	w
*	p	*
w	*	w

(i)

*	w	*
w	p	w
*	w	*

(ii)

Figure II.10. The fundamental windows used in condition MAYA 4, of algorithm MAYA. A '*' represents a dark point, a 'w' a "don't care"; i.e., the whiteness or darkness of the corresponding point is immaterial.

x	e	x
x	p	e
x	e	x

Figure II.11. The fundamental window used in condition MAYA 5, of algorithm MAYA. A 'x' represents a "don't care", a 'e' an edge-point.

	p	x
	e	e

(i)

	+	*
	e	e

(ii)

Figure II.12. The window (i) is used in condition MAYA 6, of algorithm MAYA. If the neighbourhood of the point p matches window (i), then p is flagged and n0 is set to dark, resulting in window (ii). A '*' represents a dark point, a 'x' a "don't care", a 'e' an edge-point and a '+' a flagged-point.

II.8. Pavlidis's Classical Algorithm (Pavlidis, 1982a) ---

PAVC

In PAVC, Pavlidis defines 4 kinds of edge-points: right (having n_0 white), top (having n_2 white), left (having n_4 white) and bottom (having n_6 white) edge-points. Pavlidis also defines an end-point as a dark point having at most one dark 8-neighbour. In PAVC, a pass consists of 4 scans, where in each scan only one kind of edge-points is tested for deletion. Thus, Scan 1 tests right edge-points, Scan 2 top edge-points, Scan 3 left edge-points and Scan 4 bottom edge-points.

During any pass over the pattern, a dark point p is flagged if it satisfies all of the following conditions:

PAVC_1: it is an edge-point.

PAVC_2: it is not an end-point.

PAVC_3: its neighbourhood does not match the fundamental window (i) of Figure II.13 or its c-c rotation by 90 degrees, nor does it match the fundamental window (ii) or its c-c rotation by 90, 180 and 270 degrees, with respect to Table II.3. This table shows that if, say, the point p is a right edge-point, then its neighbourhood is matched against window (i) of Figure II.13, and against window (ii) and its c-c rotation by 270 degrees. This condition conducts the break-point test.

Then, at the end of the pass all flagged points are deleted. The passes are repeated until there are no points flagged in a pass. The set of remaining dark points constitutes the skeleton of the original pattern.

Examining algorithm PAVC, one can see that the conditions PAVC_1 to PAVC_3 are the same as conditions BEUO_1 to BEUO_3 of algorithm BEUO. The sole differences between algorithms PAVC and BEUO are that PAVC performs 4 scans per pass whereas BEUO performs one scan per pass, and that PAVC does not require an excessive erosion test whereas BEUO does (condition BEUO_4).

x	x	x
	p	
y	y	y

(i)

x		*
x	p	
x	x	x

(ii)

Figure II.13. The fundamental windows used in condition PAVC 3, of algorithm PAVC. A '*' represents a dark point. At least one 'x' and one 'y' must be dark.

Kind of edge-point	Rotation, in degrees, of windows of Figure II.13.	
	window (i)	window (ii)
RIGHT	0	0, 270
TOP	90	0, 90
LEFT	0	90, 180
BOTTOM	90	180, 270

TABLE II.3.

The windows of Figure II.11 to be used in condition PAVC 3, of algorithm PAVC. For example, if the point p is a right edge-point, then the neighbourhood of p is matched against window (i) of Figure II.13, and against window (ii) and its c-c rotation by 270 degrees.

II.9. Pavlidis's Basic Algorithm (Pavlidis, 1981; Pavlidis, 1982a) --- PAVB

As in PAVC, Pavlidis defined in PAVB 4 kinds of edge-points: right (a dark point having n0 white), top (having n2 white), left (having n4 white) and bottom (having n6 white) edge-points. Pavlidis defines an end-point as a dark point having at most one dark 8-neighbour.

In PAVB, a dark point is said to be "multiple" if it satisfies all of the following conditions:

MULT_1: it is not an end-point.

MULT_2: its neighbourhood does not match the fundamental window (i) of Figure II.14 or its c-c rotation by 90 degrees, nor does it match the fundamental window (ii) or its c-c rotation by 90, 180 and 270 degrees, with respect to Table II.4.

In PAVB, a dark point is said to be "tentative multiple" if it satisfies all of the following conditions:

TMUL_1: none of its 8-neighbours is a dark non-edge-point.

TMUL_2: its neighbourhood does not match the fundamental window of Figure II.15 or its c-c rotation by 90, 180 and 270 degrees.

In PAVB, a pass consists of 3 scans. During the first scan, all edge-points are detected and marked. During the second scan, a dark point p is flagged if it satisfies all of the following conditions:

PAVB_1: it is an edge-point.

PAVB_2: it is either a "multiple" or a "tentative multiple" point.

During the third scan, all "tentative multiple" points are re-examined. A "tentative multiple" point is unflagged if it does not satisfy all of the following conditions:

PAVB_3: either n0 or n4 is white.

PAVB_4: none of its 8-neighbours is already flagged.

Then, at the end of the pass all flagged points are deleted. The passes are repeated until there are no points flagged in a pass. The set of remaining dark points constitutes the skeleton of the original pattern.

x	x	x
	p	
y	y	y

(i)

w		*
w	p	
w	w	w

(ii)

Figure II.14. The fundamental windows used in condition MULT 2, of algorithm PAVB, to detect "multiple points". A '*' represents a dark point; a 'w' is a "don't care" condition. At least one 'x' and one 'y' must be dark.

x	x	z
	p	e
y	y	z

Figure II.15. The fundamental window used in condition TMUL 2, of algorithm PAVB, to detect "tentative multiple" points. A 'e' represents an edge-point. At least one of 'x', 'y' and 'z' must be a dark point.

Kind of edge-point	Rotation, in degrees, of windows of Figure II.14.	
	window (i)	window (ii)
RIGHT	0	0, 270
TOP	90	0, 90
LEFT	0	90, 180
BOTTOM	90	180, 270

TABLE II.4.

The windows of Figure II.14 to be used in condition MULT 2, of algorithm PAVB. For example, if the point p is a right edge-point, then the neighbourhood of p is matched against window (i) of Figure II.14, and against window (ii) and its c-c rotation by 270 degrees.

II.10. Pavlidis's Algorithm with Reconstruction Ability (Pavlidis, 1981; Pavlidis, 1982a) --- PAVR

In PAVR, Pavlidis defines an edge-point as a dark point having at least one white 4-neighbour. Pavlidis also defines "multiple" and "tentative multiple" points as in algorithm PAVB described in Section II.9 above. In PAVR, all dark points are initially labelled 1 and all white points are labelled 0. The successive passes are numbered sequentially 1,2,3... A pass consists of 3 scans. During the first scan, all edge-points are detected and marked. During the second scan, a dark point p is flagged if it satisfies all of the following conditions:

PAVR_1: it is an edge-point.

PAVR_2: it is either a "multiple" or a "tentative multiple" point.

PAVR_3: it is not a corner-point; i.e., its neighbourhood does not match the fundamental window of Figure II.16 or its c-c rotation by 90, 180 and 270 degrees.

During the third scan, "tentative multiple" points are re-examined. As in algorithm PAVB, a "tentative multiple" point is unflagged if it does not satisfy all of the following conditions:

PAVR_4: either n_0 or n_4 is white.

PAVR_5: none of its 8-neighbours is already flagged.

After the three scans, if the point p is not flagged then it is labelled with the value i , where i is the current pass

number; otherwise the point is deleted. The passes are repeated until no flagged points are detected at the end of a pass. Thus, at the end of the skeletonization, the skeleton consists of a set of dark points, each dark point having a label i ($i \geq 1$). Intuitively, the label i of a dark point p is proportional to the approximate distance of p from the nearest edge in the original pattern.

Pavlidis then proposes an iterative technique describing how the labels in the skeleton can be used for reconstruction. The skeleton is scanned to find the highest value of label i , say i_{\max} . Set j to i_{\max} and begin this iteration, scanning the full set of dark points. For a dark point p with label j , any white 4-neighbour of p is changed to a dark point and labelled $j-1$. The 4-neighbours of p which are already dark remain unchanged. At the end of the scan, set j to $j-1$ and repeat the above iteration, the last iteration being performed for $j = 2$. Thus, the number of dark points padded around a point p is a function of the label of p . At the end of the reconstruction, all points with a label greater than zero are considered to constitute the reconstructed pattern.

Pavlidis has theoretically proved that the skeletons of his technique shall always retain the connectedness of the pattern and that it should be possible to achieve a perfect reconstruction; i.e., the reconstructed pattern should be

absolutely identical to the original pattern.

To remove any ambiguity in our explanation, we give below a formal description of Pavlidis's reconstruction⁵ technique.

	p	e
	e	w

Figure II.16. The fundamental window used in condition PAVR₃, of algorithm PAVR. A 'e' represents an edge-point, a 'w' is a "don't care" condition.

{ this procedure reconstructs a pattern from the skeleton,
using Pavlidis's technique (Pavlidis, 1982a) }

procedure RECONSTRUCT;

var j : integer; { the label of a point p }

begin

1- j := highest label in the pattern (i_max);

2- while j > 1 do
begin

3- for all rows and columns do
begin

4- if p = j then
for all 4-neighbours (n) of p do
if n is white then
n := j - 1;

end;

5- j := j - 1;

end; { while }

end; { RECONSTRUCT }

II.11. Stefanelli-Rosenfeld's Version of Hilditch's Algorithm (Stefanelli and Rosenfeld, 1971) --- SRVH

Hilditch (1969) did not present her algorithm HILD (see Section II.6) in a formal manner. Stefanelli and Rosenfeld (1971) presented what they called the "simplified version" of Hilditch's algorithm HILD. It is our contention that SRVH does not reflect exactly the approach described by Hilditch (Naccache and Shinghal, 1984a). We show below the "simplified version" of HILD, as described by Stefanelli and Rosenfeld.

In SRVH, Stefanelli and Rosenfeld define $A(p)$ to be the number of white to unflagged dark transitions when taking a counter-clockwise walk around p (i.e., along $n_0, n_2, \dots, n_7, n_0$). One can notice that if $A(p)$ is not equal to 1, then p is a break-point. It is also assumed in SRVH that a point of the pattern has the Boolean value TRUE if it is dark or flagged, and value FALSE otherwise (i.e., if it is white).

In algorithm SRVH, during any pass over the pattern, a dark point p is flagged if it satisfies all of the following conditions:

SRVH_1: it is an edge-point but not an end-point; i.e., it has at least two and at most six dark 8-neighbours.

SRVH_2: it is not a break-point; i.e., $A(p) = 1$;

SRVH_3: either the neighbourhood of p satisfies the Boolean

expression:

$n0 \cdot n2 \cdot n6 = \text{FALSE}$

or $A(n0) \neq 1$. This condition performs the excessive erosion test.

SRVH_4: either the neighbourhood of p satisfies the Boolean expression:

$n0 \cdot n2 \cdot n4 = \text{FALSE}$

or $A(n2) \neq 1$. This condition also performs the excessive erosion test, as it is similar to condition SRVH_3.

Then, at the end of the pass all flagged points are deleted. The passes are repeated until there are no points flagged in a pass. The set of remaining dark points constitutes the skeleton of the original pattern.

Implementing SRVH for some preliminary investigation, we found that Figure II.17a was skeletonized to Figure II.17b, whereas HILD produced Figure II.17c. Thus, we see that whereas HILD preserves the shape property of the original pattern, SRVH does not. Further comparisons between SRVH and HILD are given later in Chapter IV, with our experimental results.

A slanting stroke of width 2, represented by two parallel rows of asterisks. The top row starts with two asterisks and the bottom row starts with one, both decreasing in length as they move down and to the right.

a. A slanting stroke of width 2.

The skeleton of the slanting stroke obtained by the SRVH algorithm, represented by a single row of dashes. The dashes are arranged in a single line that follows the general path of the original stroke.

b. The skeleton obtained by SRVH.

The skeleton of the slanting stroke obtained by the HILD algorithm, represented by a single row of asterisks and dashes. The asterisks are placed at the original positions of the dark points, and dashes are placed at the positions of the deleted points.

c. The skeleton obtained by algorithm HILD.

Figure II.17. A slanting stroke and the skeletons that could be obtained from it, using SRVH and HILD. A '*' stands for a dark point, and a '-' for a deleted point.

II.12. Stefanelli-Rosenfeld's Algorithm with 4 Scans per Pass (Stefanelli and Rosenfeld, 1971; Rosenfeld, 1975) ---
SR4S

Stefanelli and Rosenfeld define 4 kinds of edge-points: right (having n_0 white), top (having n_2 white), left (having n_4 white) and bottom (having n_6 white) edge-point. In SR4S, a pass consists of 4 scans, where in each scan only one kind of edge-point is tested for deletion. Thus, during Scan 1 bottom edge-points are tested. During Scan 2, Scan 3 and Scan 4, top, left and right edge-points are tested, respectively. During any pass over the pattern, a dark point p is flagged if it satisfies all of the following conditions:

SR4S_1: it is an edge-point.

SR4S_2: its neighbourhood does not match any of the eight fundamental windows of Figure II.18, with respect to Table II.5.

Then, at the end of the pass all flagged points are deleted. The passes are repeated until there are no points flagged in a pass. The set of remaining dark points constitutes the skeleton of the original pattern.

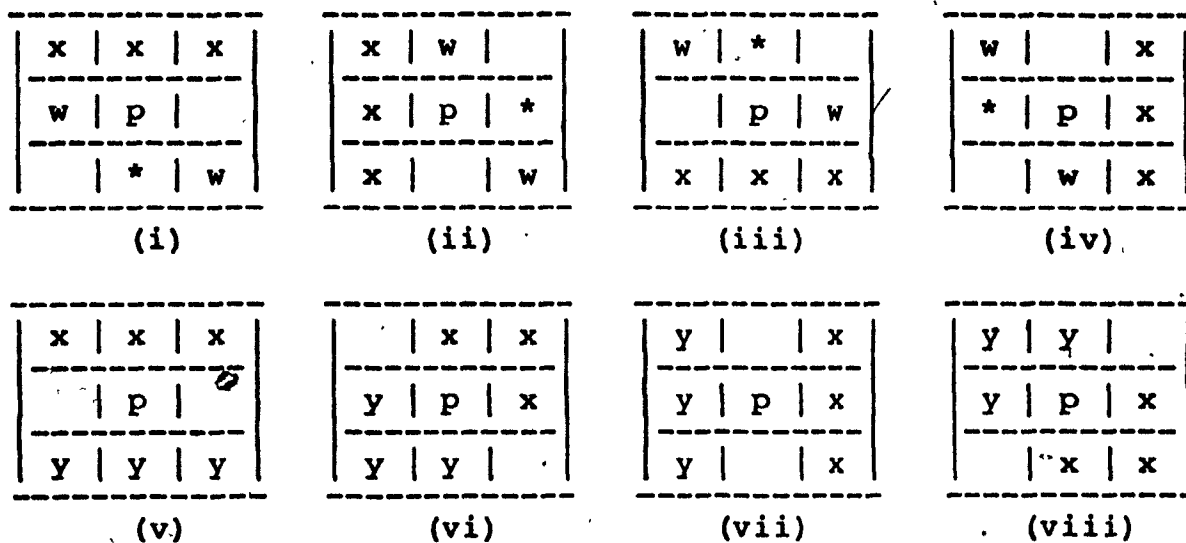


Figure II.18. The eight fundamental windows used in condition SR4S₂ of algorithm SR4S. A '*' represents a dark point, a 'w' a "don't care". At least one 'x' and one 'y' must be dark. These windows are also used in condition SR2S₂ of algorithm SR2S, described below in Section II.13.

Kind of edge-point	The windows to be used to satisfy condition SR4S_2
RIGHT	(ii), (iii), (v) to (viii)
TOP	(iii), (iv), (v) to (viii)
LEFT	(i), (iv), (v) to (viii)
BOTTOM	(i), (ii), (v) to (viii)

TABLE II.5.

The windows of Figure II.18 to be used in condition SR4S_2. For example, if p is a right edge-point, then windows (ii), (iii), and (v) to (viii) have to be used. This table is also referred to in condition SR2S_2 of algorithm SR2S, described below in Section II.13.

II.13. Stefanelli-Rosenfeld's Algorithm with 2 Scans per Pass (Stefanelli and Rosenfeld, 1971; Rosenfeld, 1975) ---
SR2S

The algorithm SR2S is somewhat similar to SR4S, described in Section II.12 above. Stefanelli and Rosenfeld define edge-points as in SR4S. There are two scans per pass. During Scan 1, bottom and left edge-points are tested for deletion, and during Scan 2 top and bottom edge-points are tested.

During any pass over the pattern, a dark point p is flagged if it satisfies all of the following conditions:

SR2S_1: it is an edge-point.

SR2S_2: its neighbourhood does not match the eight fundamental windows of Figure II.5, with respect to Table II.5.

SR2S_3: its neighbourhood does not match the four fundamental windows of Figure II.19, with respect to Table II.6.

Then, at the end of the pass, all flagged points are deleted. The passes are repeated until there are no points flagged in a pass. The set of remaining dark points constitutes the skeleton of the original pattern.

<table><tr><td>w</td><td>w</td><td>w</td></tr><tr><td></td><td>p</td><td>w</td></tr><tr><td>*</td><td></td><td>w</td></tr></table>	w	w	w		p	w	*		w	<table><tr><td>w</td><td>w</td><td>w</td></tr><tr><td>w</td><td>p</td><td></td></tr><tr><td>w</td><td></td><td>*</td></tr></table>	w	w	w	w	p		w		*	<table><tr><td>w</td><td></td><td>*</td></tr><tr><td>w</td><td>p</td><td></td></tr><tr><td>w</td><td>w</td><td>w</td></tr></table>	w		*	w	p		w	w	w	<table><tr><td>*</td><td></td><td>w</td></tr><tr><td></td><td>p</td><td>w</td></tr><tr><td>w</td><td>w</td><td>w</td></tr></table>	*		w		p	w	w	w	w
w	w	w																																					
	p	w																																					
*		w																																					
w	w	w																																					
w	p																																						
w		*																																					
w		*																																					
w	p																																						
w	w	w																																					
*		w																																					
	p	w																																					
w	w	w																																					
(i)	(ii)	(iii)	(iv)																																				

Figure II.19 The four fundamental windows used in condition SR2S₃ of algorithm SR2S. A '*' represents a dark point, a 'w' a "don't care".

Kind of edge-point	The windows to be used to satisfy the condition SR2S_3
BOTTOM or LEFT	(i) and (ii)
TOP or RIGHT	(iii) and (iv)

TABLE II.6.

The windows of Figure II.19, to be used condition SR2S 3 of algorithm SR2S. For example, if p is a bottom or a left edge-points then windows (i) and (ii) are used.

II.14. Tamura's Algorithm Ensuring Four-Connectedness
(Tamura, 1978) --- TM4F

Tamura proposed conducting the same 4 scans as in SR4S, but using the four windows of Figure II.20 instead of the fundamental windows (i) to (iv) of Figure II.18. This modification, he said, would eliminate the "mistakes" of SR4S. Tamura does not explain clearly what "mistakes" he found in SR4S. He cites two other papers of his, which we have not been able to read as the papers are in Japanese, a language we do not know. Tamura states that the skeletons obtained by TM4F are always four-connected.

w	*	
x	p	*
w	x	w

(i)

	*	w
*	p	x
w	x	w

(ii)

w	x	w
*	p	x
	*	w

(iii)

w	x	w
x	p	*
w	*	

(iv)

Figure II.20. The modified windows used by TM4F, instead of the windows (i) to (iv) of Figure II.18.

II.15. Tamura's Algorithm Ensuring Eight-Connectedness

(Tamura, 1978) --- TM4E

Tamura further modified his own algorithm TM4F by using the four windows (v) to (viii) of Figure II.18 and the four windows of Figure II.19. By changing the windows, Tamura showed that whereas TM4F gave four-connected skeletons, TM4E gave eight-connected skeletons.

II.16. Tamura's Algorithm with 2 Scans per Pass (Tamura, 1978) --- TM2E

Just as he did for SR4S, Tamura suggested changes to SR2S also. He thus proposed conducting the same 2 scans as in SR2S but using the windows of Figure II.20 instead of the windows (i) to (iv) of Figure II.18. Tamura showed that the skeletons produced were eight-connected.

II.17. Zhang-Suen's Algorithm (Zhang and Suen, 1984) ---

ZHAN

As in algorithm SRVH, Zhang and Suen define $A(p)$ as the number of white to dark transitions when taking a counter-clockwise walk around p . Moreover, in ZHAN, a Boolean variable has value TRUE if its corresponding point is dark or flagged, and value FALSE if it is white. A pass consists of 2 scans. During the first scan, a dark point p is flagged if it satisfies all of the following conditions:

ZHAN_1: it is an edge-point but not an end-point; i.e., it has at least two and at most six dark 8-neighbours;

ZHAN_2: it is not a break-point; i.e., $A(p) = 1$;

ZHAN_3: its neighbourhood satisfies the Boolean expression:

$$n0 \cdot n2 \cdot n6 = \text{FALSE}$$

ZHAN_4: its neighbourhood satisfies the Boolean expression:

$$n0 \cdot n4 \cdot n6 = \text{FALSE}$$

The conditions ZHAN_3 and ZHAN_4 conducts the excessive erosion tests. At the end of the first scan, all flagged points are deleted. During the second scan over the pattern, a dark point p is flagged if it satisfies conditions ZHAN_1 and ZHAN_2, as well as the following conditions:

ZHAN_5: its neighbourhood satisfies the Boolean expression:

$$n0 \cdot n2 \cdot n4 = \text{FALSE}$$

ZHAN_6: its neighbourhood satisfies the Boolean expression:

$$n2 \cdot n4 \cdot n6 = \text{FALSE}$$

The conditions ZHAN_5 and ZHAN_6 conduct the excessive erosion tests, as they are similar to ZHAN_3 and ZHAN_4. At the end of the second scan all flagged points are deleted. The passes are repeated until there are no points flagged in any of the two scans. The set of remaining dark points constitutes the skeleton of the original pattern.

One can observe, however, that conditions ZHAN_1 and ZHAN_2 are the same as conditions SRVH_1 and SRVH_2 of Stefanelli-Rosenfeld's version of Hilditch's algorithm (SRVH). The conditions ZHAN_3 through ZHAN_6 are a modification of conditions SRVH_3 and SRVH_4 of algorithm SRVH. Wondering if ZHAN would prevent the excessive erosion that SRVH failed to do (see Figure II.17), we tested Zhang-Suen's algorithm on the slanting stroke of Figure II.21a. The skeleton which was obtained is shown in Figure II.21b. Clearly, it is a case of excessive erosion; i.e., ZHAN fails like SRVH in maintaining the shape property of the original pattern.



a. A slanting stroke
of width 2.



b. The skeleton
obtained by
ZHAN.

Figure II.21. To test if ZHAN would give excessive erosion, we tested the algorithm on Figure II.21a. The skeleton obtained is shown in Figure II.21b.

II.18. Other existing skeletonization algorithms

The 16 algorithms discussed above have been experimentally tested and compared (see Chapter IV). However, we concede that there exists other skeletonization algorithms which we did not review in this thesis. We will briefly discuss below 10 well-known such algorithms and we will explain why we did not implement and test them.

Chaudhuri (1978) describes a skeletonization algorithm based on the thickness of the strokes of the original pattern. This method has the disadvantage that the average thickness and curvature angles of the original pattern have to be calculated before applying the algorithm. Pavlidis (1982b) follows a similar approach to skeletonize the patterns.

Arcelli and di Baja (1981) propose a skeletonization algorithm based on the detection of prominences in the original pattern. Although this algorithm has been shown to give skeletons of good quality (i.e., skeletons preserving the shape property of the original patterns), it is heavy in computation requirements and is thus quite slow.

Hilditch (1983) proposes some modifications to algorithm ARCL. However, these modified algorithms would ideally require to be implemented on a multiprocessor, a

facility we do not have at Concordia University, Montreal.

Stentiford and Mortimer (1983) use a modified version of Yokoi's algorithm (Yokoi, Toriwaki, Fukumura, 1973), after heuristically smoothing the original pattern. This heuristic smoothing requires to perform many passes over the pattern before the skeletonization process. Moreover, the Stentiford-Mortimer algorithm is applicable only to hand-printed characters, and is not generalized to other kinds of patterns.

Davies and Plummer (1981) propose a skeletonization algorithm based on Beun's algorithm BEUS, but requiring more scans per pass. This algorithm is thus slower than BEUS. Moreover, Davies and Plummer propose a reconstruction technique based on the Rosenfeld-Pfaltz (1966) technique for reconstruction. They also mention that the Rosenfeld-Pfaltz skeletonization algorithm does not produce connected skeletons.

Finally, Tamura (1978) compares some other skeletonization algorithms, essentially Rutovitz's (Rutovitz, 1966), Deutsch's (Deutsch, 1969; Deutsch, 1972) and Yokoi's (Yokoi, 1973; Yokoi, 1975) algorithms. The first two algorithms do not always produce connected skeletons. The third one gives good skeletons but appears to be relatively slow in terms of cpu-time required.

Moreover, it does not have reconstruction ability.

If we did not review these 10 algorithms, it is because their major advantages and drawbacks have already been discussed in the above-mentioned papers, and also because most of them have common concepts with the 16 algorithms reviewed earlier in this chapter. Thus, as it pertains to this thesis, a detailed review of these 10 algorithms would have been unnecessary; we intentionally omitted them.

We are now going to present our proposed Safe-Point Thinning Algorithm (or SPTA), in Chapter III. Later, in Chapter IV, we will experimentally compare the performance of our SPTA with the 16 algorithms we have described in Chapter II.

Chapter III

The Safe-Point Thinning Algorithm

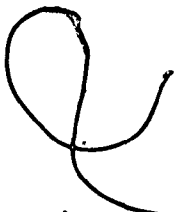
We will present in this chapter our Safe-Point Thinning Algorithm (Naccache and Shinghal, 1984b). The algorithm is described first informally then formally. Later, we will see how some appropriate labelling techniques of the points of the pattern can speed up the algorithm and provide it with reconstruction ability.

Our objective is to develop a skeletonization algorithm which will be fast, which will prevent excessive erosion, and one which will give good skeletons (i.e., skeletons which retain the shape information of the original pattern and do not have spurious tails) (Davies and Plummer, 1981). Furthermore, the algorithm should be such that enough information is retained in the skeleton to reconstruct as far as possible a pattern which is similar to the original pattern.

III.1. Fundamental Definitions

It is assumed that the input pattern is a smoothed pattern; that is, all pattern irregularities and all noise have been removed.

In essence, the SPTA consists of executing many passes over the pattern, where in each pass a few dark points are



flagged. A flagged point must be such that it is an edge-point, but not an end-point, nor a break-point, and nor must its possible deletion cause excessive erosion in the pattern. At the end of the pass, all flagged points are deleted. If no points are deleted at the end of a pass, then the skeletonization procedure stops. The SPTA differs from the algorithms reviewed in Chapter II in the manner in which the tests are conducted and the details of implementation, all of which are described below. For the sake of easy readability, initially we shall present our description in a semi-formal manner; the formal description appears later in this chapter.

In the SPTA, it is sometimes convenient to refer to the 4-neighbours n_0 , n_2 , n_4 and n_6 of a point p as the right neighbour n_R , top neighbour n_T , left neighbour n_L and bottom neighbour n_B , respectively. We use both notations interchangeably, depending on which notation would be easier to understand at that juncture.

The SPTA identifies the four following kinds of edge-points: a) a left edge-point, having its left neighbour n_L , i.e., n_4 white (see Figure II.1); b) a right edge-point, having n_R , i.e., n_0 white; c) a top edge-point, having n_T , i.e., n_2 white; and, d) a bottom edge-point, having n_B , i.e., n_6 white. It should be noted that an edge-point can be of more than one kind; for example, a dark point having

n_T white and n_L white will be a top edge-point and a left edge-point simultaneously. We also define the following: an end-point is a dark point having at most one dark 8-neighbour, and a break-point is a dark point the deletion of which would break the connectedness of the pattern.

In the following discussion, we will refer exclusively to left edge-points. The discussion is later generalized to other kinds of edge-points.

III.2. Explanation of the Safe-Point Concept

A left edge-point p which is not an end-point, nor a break-point and nor would its deletion cause excessive erosion is called a left flag-point. Any left edge-point which is not a left flag-point is called a left safe-point. We show below that to identify such a left edge-point p , SPTA needs to compare the neighbourhood of p with the 4 windows shown in Figure III.1. If the neighbourhood of p matches any one of the 4 windows, then p is not flagged. It should be noted that the points shown as x's and y's in the windows are "don't care" points (i.e., their whiteness or darkness is immaterial). We now examine the 4 windows one by one, to justify why these are the windows required by the SPTA to conduct the break-point, end-point and excessive erosion tests on p .

If the neighbourhood of p matches any of the windows (i), (ii) or (iii) of Figure III.1, then two situations may occur: 1) if all x 's are white, then p is an end-point; 2) if at least one of the x 's is a dark point, then p is a break-point. Thus, in either of these cases, p should not be flagged.

Now let us examine window (iv) of Figure III.1. If at least one ' x ' and at least one ' y ' are dark, then p becomes a break-point and thus it should not be flagged. For further analysis, let us assume for the time being that all x 's are white. Then there are 8 possible configurations as shown in Figure III.2. Configurations w_1 , w_2 , and w_3 make p an end-point, and configuration w_4 makes p a break-point. The possible deletion of p in configurations w_5 and w_6 could cause excessive erosion in slanting strokes of width 2; e.g., Figure III.3a being reduced to Figure III.3b. In configurations w_7 and w_8 , the point p is in fact noise. Since the patterns have already been smoothed before they are fed to our SPTA, such noise is always removed and configurations w_7 and w_8 shall never exist at the beginning of the skeletonization process. If configuration w_7 were to occur in an intermediate stage of skeletonization, then p would be a spur due to a short tail in the original pattern (e.g., as in chromosomes). Such a point may, however, retain some shape information of the pattern and it must not be deleted. Similarly, if configuration w_8 were to occur in

an intermediate stage of skeletonization, then it would be a singleton (isolated point); its deletion would totally erase the last remaining segment of the pattern. Therefore, in all of the above configurations p must not be flagged. By symmetry, we extend our argument to the case in window (iv) when all the y's are white and x's take on varying values of whiteness or darkness.

Reinforcing our analytic arguments above, we have exhaustively verified by experiment that the four windows of Figure III.1 are the only ones we require to conduct end-point, break-point and excessive erosion tests on a dark point p; i.e., the safe-point test.

Using the four windows of Figure III.1, it can be easily shown that for a left safe-point the Boolean expression S_4 is FALSE, where:

$$S_4 = n_0.(n_1+n_2+n_6+n_7).(n_2+n_3).(n_6+n_5).$$

A Boolean variable has the value TRUE when its corresponding point is dark and unflagged, and it has value FALSE otherwise (i.e., if the point is a flag-point or it is white). Thus, to test whether a left edge-point is a left safe-point, the SPTA needs only to test against Boolean expression S_4 . The subscript in S_4 indicates that the Boolean expression was derived for left edge-points, i.e., they have their left neighbour n_4 white.

Similarly, we can derive the following Boolean expressions:

for right safe-point:

$$S_0 = n_4 \cdot (n_5 + n_6 + n_2 + n_3) \cdot (n_6 + \overline{n_7}) \cdot (n_2 + \overline{n_1});$$

for top safe-point :

$$S_2 = n_6 \cdot (n_7 + n_0 + n_4 + n_5) \cdot (n_0 + \overline{n_1}) \cdot (n_4 + \overline{n_3}); \text{ and,}$$

for bottom safe-point:

$$S_6 = n_2 \cdot (n_3 + n_4 + n_0 + n_1) \cdot (n_4 + \overline{n_5}) \cdot (n_0 + \overline{n_7}).$$

These expressions are derived from a counter-clockwise rotation of the windows of Figure III.1 by 180, 270 and 90 degrees, respectively.

Investigating further, we noticed that window (iv) of Figure III.1 is the same for the left and the right safe-point tests. In other words, a dark point which is simultaneously a left edge-point and a right edge-point, is also a left safe-point and a right safe-point. Such a point is called a left-right safe-point. We define the two Boolean variables:

E_0 , having value TRUE if n_0 is dark (flagged or unflagged) and value FALSE otherwise; and,

E_4 , similarly linked to n_4 .

Now, if E_0 and E_4 are both FALSE, then the dark point p is obviously a left-right safe-point. If E_0 is TRUE but E_4 is FALSE, then p is a left edge-point; it will become a left safe-point only if S_4 is also FALSE. A similar reasoning holds when E_4 is TRUE but E_0 is FALSE. Finally, if E_0 and

E_4 are both TRUE, then p is not a left edge-point nor a right edge-point. Thus, using the variables E_0 and E_4 , we can merge both Boolean expressions S_0 and S_4 into the following expression:

$$S_{04} = (E_0 + E_4) \cdot (E_4 \cdot (E_0 + S_0)) \cdot (E_0 \cdot (E_4 + S_4))$$

If the expression S_{04} is FALSE, then the dark point p is either a left safe-point, or a right safe-point, or a left-right safe-point.

Similarly, we can derive the following Boolean expression for top and bottom safe-points:

$$S_{26} = (E_2 + E_6) \cdot (E_6 \cdot (E_2 + S_2)) \cdot (E_2 \cdot (E_6 + S_6))$$

where E_2 and E_6 are similar to E_0 and E_4 , as they are linked to n_2 and n_6 , respectively.

Now we describe how we use the above definitions in the skeletonization process of SPTA.

*		x
	p	x
x	x	x

(i)

x	x	x
	p	x
*		x

(ii)

x		
	p	*
x		

(iii)

x	x	x
	p	
y	y	y

(iv)

Figure III.1. If the neighbourhood of a dark point p matches any of the four windows above, then SPTA does not flag p. A '*' indicates a dark point. x's and y's are "don't care"s (their whiteness or darkness is immaterial).

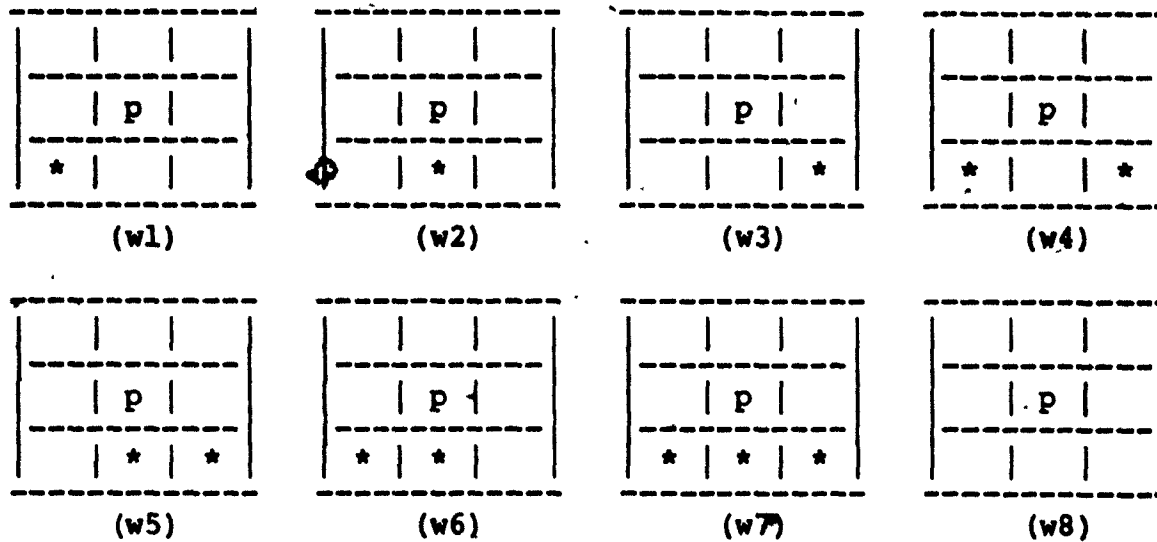


Figure III.2. The 8 possible configurations that could exist in the window (iv) of Figure III.1, when all x's are white.

III.3. The SPTA Process

The SPTA consists of executing many passes over the pattern, where in each pass a few dark points are deleted. A pass in SPTA consists of 2 scans, where a scan examines every point in the pattern. The pattern may be scanned rowwise or columnwise at the user's choice. The choice of the scanning sequence may lead to different skeletons. It should be noted that for a given pattern, a skeleton may not be unique (Davies and Plummer, 1981). During the first scan, all left flag-points, right flag-points, left safe-points and right safe-points are detected. Similarly in the second scan, SPTA detects all top flag-points, bottom flag-points, top safe-points and bottom safe-points. Then, at the end of the pass (i.e., the two scans), all left, right, top and bottom flag-points are deleted. These passes are iteratively executed as long as there are flag-points in the pattern. A question arises at this point: would it not be better to have a single scan per pass, where the scan could detect all 4 kinds (left, right, top, bottom) of flag-points and safe-points? We have experimentally found that a single scan approach can sometimes cause excessive erosion. For example, the slanting stroke of Figure III.3a was skeletonized by the one-scan approach to a set of two dark points as shown in Figure III.3b. That was clearly a case of excessive erosion. However, using our two-scan approach, the skeleton of Figure III.3a is shown in Figure

III.3c. Thus, it is essential to have 2 scans per pass.

We show below a Pascal-like formal algorithmic version of our SPTA. Reading our formal description, one may wonder as to how we effect the deletion of flag-points. The deletion is in fact incorporated within the technique we have adopted to label the points as they are scanned. Moreover, the labelling is also useful when we attempt to reconstruct the pattern from the skeleton. In the following section, we discuss details of the labelling techniques used in SPTA.

{--- Declaration of global variables ---}

```
type pattern_type: array [1..HEIGHT,1..WIDTH] of integer;
    { type of the pattern, where HEIGHT and
      WIDTH are its dimensions }

    i      : integer;
    { i represents the pass number
      during the skeletonization process }

    edg_pt_kind: {left, right, top, bottom};
    { the different kinds of edge-points }

    point_type : { either } integer;
    { or } array [edg_pt_kind] of integer;
    { point_type depends on the labelling used,
      as shall be seen later in this chapter };
```

{-----}

{The following procedure drives the
the scanning of the pattern}

procedure DRIVER (var pattern: pattern_type);

```
var    j : [0,2];
    { j indicates the 'type' of scanning;
      j = 0 ==> detection of right and left safe-points,
      j = 2 ==> detection of top and bottom safe-points }
```

1- begin

```
2-   i := 0;    { initialize pass number }
3-   repeat
```

```
4-       i := i + 1;
```

```
5-       for both j's do
```

```
6-           SKELETONIZE (pattern, j);
           { procedure SKELETONIZE is shown below }
```

```
7-   until NO_MORE;
```

```
    { NO MORE returns TRUE when there are no
      more points to be flagged in the pattern }
```

```
8- end; { DRIVER }
```

{ The following procedure performs one scan
of skeletonization over the pattern }

procedure SKELETONIZE (var pattern: pattern_type; j:integer);

var p : point_type; { point to be examined }
edge_kind : edg_pt_kind; { kind of edge-point }

9- begin

10- for all the points p do
 { raster scan of the pattern with type j }

11- begin

12- if DARK (p) then
 { if p is dark and not yet a flag-point }

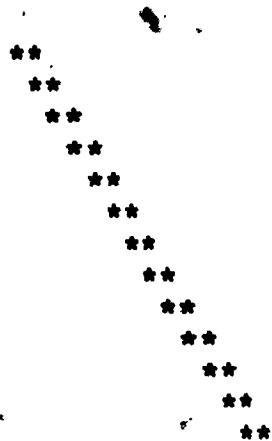
13- if not SAFE POINT (p, j, edge kind) then
 { if p does not satisfy either
 Boolean expression S_j or S_{j+4} ;
 Note: the kind of edge-point (E_j or E_{j+4})
 is returned to edge_kind }

14- FLAG (p, i)
 { p is recognized to be a flag-point
 and is labelled }

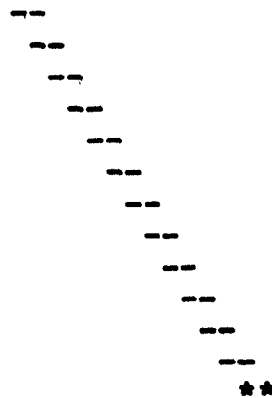
else
15- SAFE (p, i, edge kind);
 { p is recognized to be a safe-point
 and is labelled }

16- end; {of for-loop}

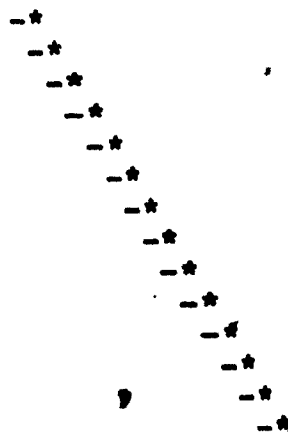
17- end; {SKELETONIZE}



a. A slanting stroke of width 2.



b. This is the skeleton obtained if we do either the following:
 (i) if we flag a dark point p whose neighbourhood matches the configurations w_5 or w_6 of Figure III.2;
 (ii) if we perform only one scan per pass in SPTA, flagging all four types of edge-points that do not satisfy the Boolean expressions S_{04} and S_{26} . Either approach gives excessive erosion.



c. The skeleton obtained by SPTA with two scans per pass.

Figure III.3. A slanting stroke and the skeletons that could be obtained from it. The '*'s represent the dark points. The '-'s represent the deleted points.

III.4. The Labelling Techniques of SPTA

The labelling of the flag-points and the safe-points is shown in lines 14 and 15 of the procedure SKELETONIZE above. We propose two different labelling methods, the Single Integer Labelling Technique (SILT) and the Four Integers Labelling Technique (FILT). The user may decide on which labelling method he wants to adopt. Usually FILT causes the skeletonization algorithm to slow down a bit, but it has stronger reconstruction ability. We discuss below the details of SILT and FILT after our formal presentation of SPTA. Above, we have procedure DRIVER which controls the scanning of the pattern. Procedure DRIVER calls procedure SKELETONIZE which detects flag-points and safe-points in a given scan.

Single Integer Labelling Technique (SILT):

To implement this technique, line 15 in procedure SKELETONIZE above is replaced by SAFE (p, i). Initially in the pattern, all dark points are labelled zero and all white points are labelled minus MAXINT (where MAXINT is the largest possible integer that can be stored in the computer). During any pass of the SPTA, the value of the non-edge-points does not change. If an edge-point p is identified as a flag-point, then it is labelled by the value (i - MAXINT), where i is the current pass number, the passes being numbered sequentially as 1,2,3,... On the other hand,

if p is identified as a safe-point, then it is labelled by the value of i ; that is, the label of a safe-point indicates the smallest iteration number of the pass in which the point was identified as a safe-point. For clarity of understanding, the tree of Figure III.4 shows this labelling technique. With this labelling technique, we are able to incorporate deletion of dark points as follows:

Ordinarily at the end of a pass we would need to travel through the entire pattern deleting all flag-points in order to "prepare the pattern for the next pass. However, the flag-points which are candidates for deletion have been assigned the label $i\text{-MAXINT}$ in pass i . In pass $i+1$, $i\text{-MAXINT}$ becomes a threshold, and all points with a label value less than or equal to this threshold are considered white. The flag-points are thus only conceptually deleted and it results in speeding up the algorithm. The procedures FLAG and SAFE, which are invoked by the procedure SKELETONIZE above, can then be formally described as follows:

```
procedure FLAG (var p: integer; i: integer);
```

```
begin
```

```
  p := i - MAXINT;
```

```
end; { FLAG }
```

```
procedure SAFE (var p: integer; i: integer);
```

```
begin
```

```
  if p = 0 then
```

```
    { p may have been labelled as a safe-point  
      in an earlier pass }
```

```
    p := i;
```

```
end; { SAFE }
```

We observe that at the end of SPTA, if we have a point in the skeleton with label i , then it indicates that in the original pattern the closest edge to this point was $(i-1)$ pixels away. We can then use Pavlidis's technique of reconstruction, which we described in Chapter II (Section II.10).

We next discuss our alternative labelling technique called FILT.

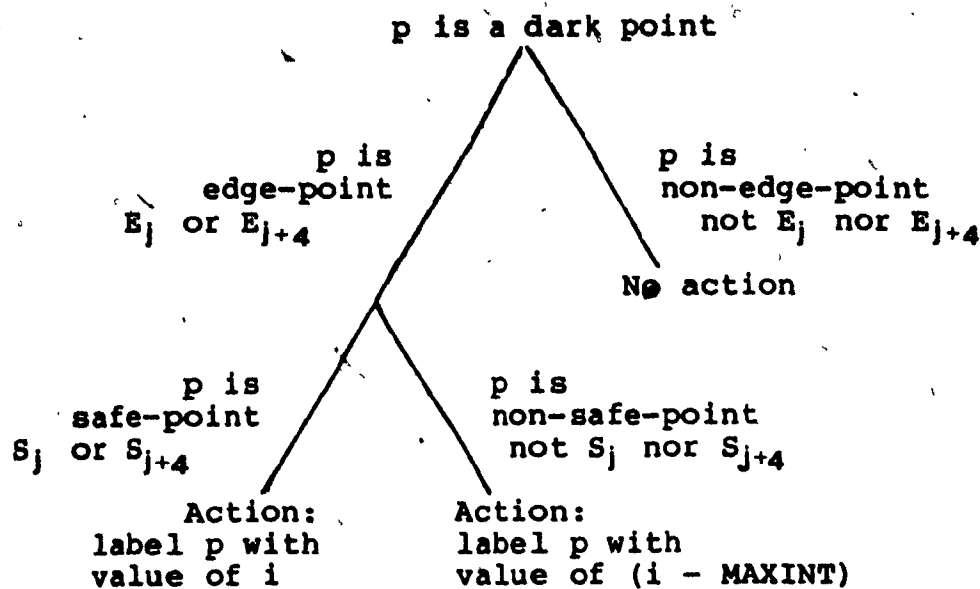


Figure III.4. The decision tree required by SPTA-SILT to label a dark point. The value of j (0 or 2) represents the type of scanning (see procedure DRIVER in Section III.4). The value of i represents the iteration number of the skeletonization process over a pattern. MAXINT is the largest integer storable in a computer.

Four Integers Labelling Technique (FILT):

In this labelling technique, every point of the pattern is labelled by a 4-tuple $\langle v_L, v_R, v_T, v_B \rangle$, where the v 's have an integer value in the closed interval $[-\text{MAXINT}, \text{MAXINT}]$. Initially, all white points are labelled $\langle -\text{MAXINT}, -\text{MAXINT}, -\text{MAXINT}, -\text{MAXINT} \rangle$, and all dark points are labelled $\langle 0, 0, 0, 0 \rangle$. The subscripts L, R, T and B are abbreviated forms for left, right, top and bottom, respectively. As we shall see below, the change in the value of v_L , v_R , v_T or v_B for a point depends on whether the point is a flag-point or a left, right, top or bottom safe-point. During any pass of the SPTA, the values in the 4-tuple of the non-edge-points do not change. For any edge-point p , two situations may occur:

a) if p is identified as any kind of flag-point, then p assumes the label:

$$\langle (i - \text{MAXINT}), (i - \text{MAXINT}), (i - \text{MAXINT}), (i - \text{MAXINT}) \rangle,$$

where i is the current pass number, the passes being sequentially numbered from 1 onwards;

b) if p is identified as a K -safe-point (where $K \in \{L, R, T, B\}$), then v_K takes on the value of i , the other v 's remaining unchanged. The value of v_K indicates the smallest iteration number of the pass in which p was identified as a K -safe-point. For clarity of understanding, the tree of Figure III.5 shows this labelling technique. With this labelling technique, we are able to incorporate deletion of dark points as follows. The flag-points, which are

candidates for deletion, have the label:

$\langle (i - \text{MAXINT}), (i - \text{MAXINT}), (i - \text{MAXINT}), (i - \text{MAXINT}) \rangle$

in pass i . In the pass $i+1$, $i - \text{MAXINT}$ becomes a threshold.

All points with all label elements having values less than or equal to $(i - \text{MAXINT})$ are considered white. Thus, for FILT also the flag-points are only conceptually deleted, as in the SILT approach described above. The procedures FLAG and SAFE, which are invoked from the procedure SKELETONIZE, can then be formally described as follows:

```
procedure FLAG (var p: integer; i: integer);
```

```
var K : edg_pt_kind;
begin
  for K := left to bottom do
    p[K] := i - MAXINT;
end; { FLAG }
```

```
procedure SAFE (var p: point_type; i: integer; K: edg_pt_kind);
```

```
begin
  if p[K] = 0 then
    { p may have been labelled as a K-safe-point
      in an earlier pass }
    p[K] := i;
end; { SAFE }
```

Thus at the end of the SPTA, if for any point p of the skeleton $i_K \neq 0$ ($K \in \{L, R, T, B\}$); i.e., in the 4-tuple of the label any of the values is ~~zero~~, then it means that p was never identified by the SPTA as a K -edge-point. Moreover, if a point p of the skeleton has a label with, say, the 4-tuple values $\langle i_L, i_R, i_T, i_B \rangle$ (where $i_K > 0$, $K \in \{L, R, T, B\}$), then it indicates that in the original pattern the closest

left edge was about (i_L-1) pixels away. Similarly, it indicates that the closest right, top and bottom edges were about (i_R-1) , (i_T-1) and (i_B-1) pixels away, respectively. The approach we use for reconstruction is an extended version of Pavlidis's technique given in the Appendix. Let $w_K = v_K(n_K(p))$, where p is a dark point, $K \in \{L, R, T, B\}$, and where n_L , n_R , n_T and n_B are the 4-neighbours of p (see Figure I.3). That means w_K is the value of the label element v_K of the K -neighbour of p . Initially, the labels of the skeleton are scanned to find the highest value i_{\max} occurring amongst the 4-tuples of these labels. Set j to i_{\max} and begin this iteration, scanning the full set of dark points. For any dark point p , if $v_K(p) = j$ and $w_K = 0$, then set w_K to $j-1$. At the end of the scan, set j to $j-1$ and repeat the above iteration, the last iteration being performed for $j=2$. At the end of the reconstruction, all points in which the label has at least one element of the 4-tuple having a value greater than zero are considered to constitute the reconstructed pattern. To remove any ambiguity, we show below a formal description of this reconstruction approach.


```
{-- We recall that "pattern" is a global variable
    containing the input pattern --}
```

```
procedure RECONS_FILT;
```

```
var K      : {left, right, top, bottom};
           { the elements of a 4-tuple }
  el      : integer;
           { one element of the considered point }
  nK     : array [K] of integer;
           { the K-neighbour of p }
  j       : integer;
           { the current highest label }
```

```
begin
```

```
  j := highest element of the pattern (i_max);
```

```
  while j > 1 do
    begin
```

```
    1-   for all rows and columns do
          begin
```

```
    2-   for K := left to bottom do
          begin
```

```
    3-   el := pattern [row, column] [K];
```

```
    4-   let nK be the K-neighbour
          the point "pattern [row, column]";
```

```
    5-   if el = j then
          begin
```

```
    6-   if nK [K] < zero then
          nK [K] := j - 1;
          end;
```

```
    7-   store the new value of nK [K]
          in "pattern" if changed;
```

```
      end;
```

```
    end;
```

```
  j := j - 1;
```

```
  end; { while }
```

```
end; { RECONS_FILT }
```

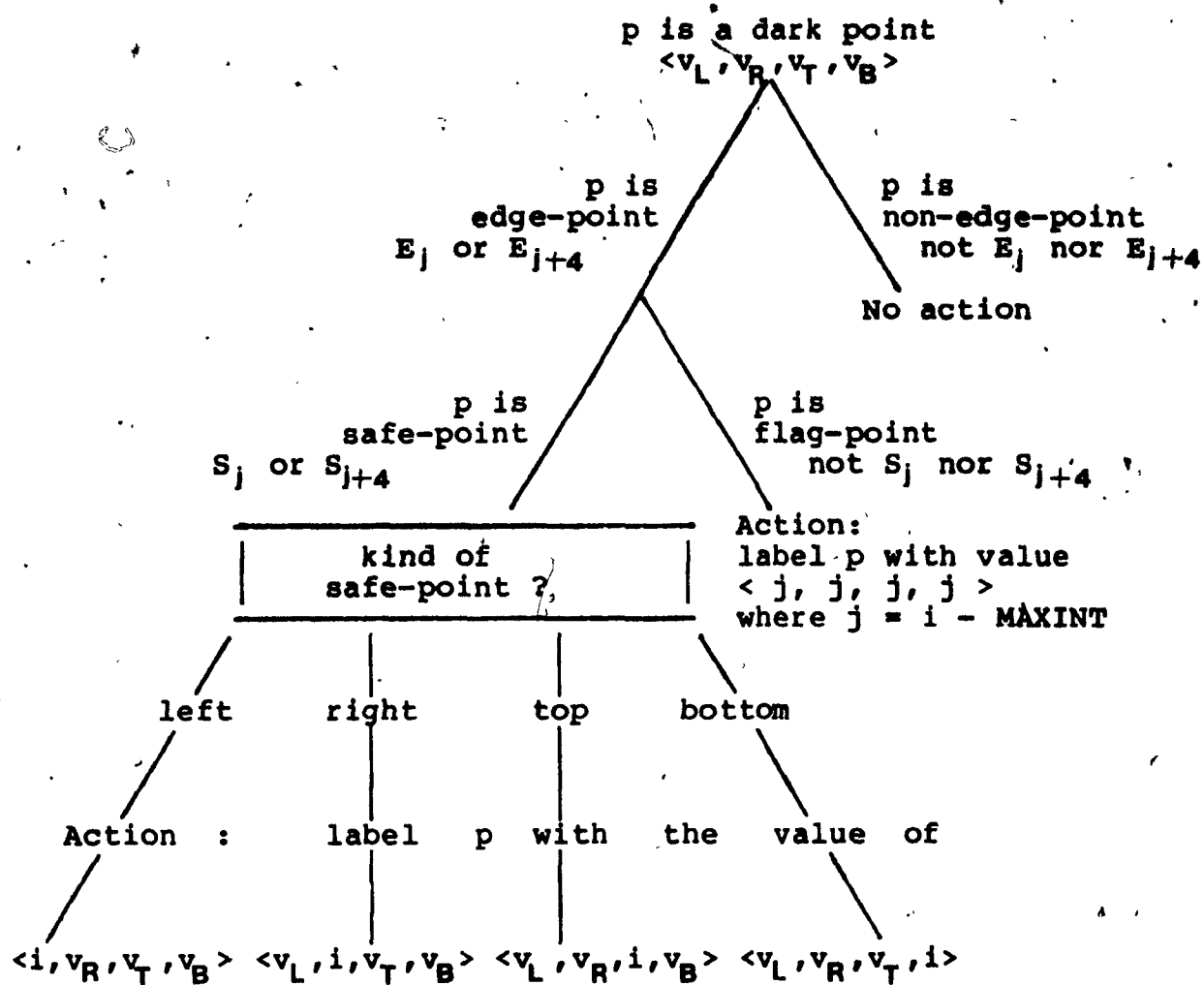


Figure III.5. The decision tree required by SPTA-FILT to label a dark point. The value of j (0 or 2) represents the type of scanning (see procedure DRIVER, in Section III.4). The value i represents the current pass number of the skeletonization process over a pattern. MAXINT is the largest integer that can be stored in a computer.

III.5. Implementation Techniques of SPTA

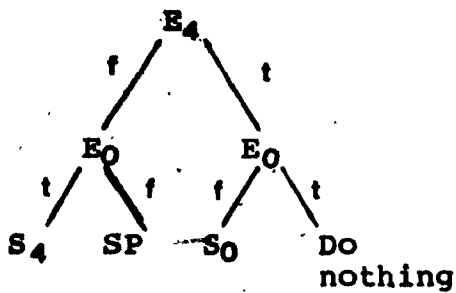
Although our SPTA is conceptually clear in the description given above, there are details that need to be explained in actual implementation so that the algorithm could be speeded up as far as possible. We describe below two implementation techniques that we applied to achieve these objectives: how the number of scans could be minimized, and how a suitable decision tree could be developed to test the Boolean expressions S_{04} and S_{26} . We discuss these two techniques in detail below.

A. Minimizing number of scans

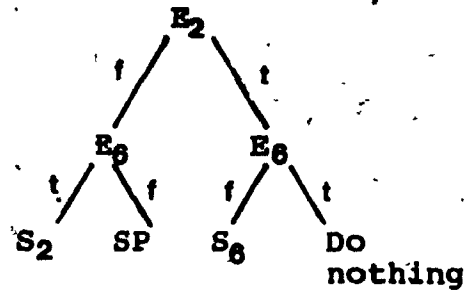
As described in our algorithm above, Scan 1 flags some left and right edge-points, and Scan 2 flags some top and bottom edge-points. Say Scan 1 flags no point but Scan 2 flags a few. The algorithm does not terminate yet, but in the following pass we do not initiate Scan 1. Only Scan 2 is initiated. This is because if Scan 1 did not flag any point in a given pass, it will not flag any point in a subsequent pass. Within any scan, we added a further refinement. Say in a given pass Scan 1 flagged only some left edge-points, but it flagged none of the right edge-points. Then, in the following pass, Scan 1 needs to test only the left edge-points for flagging. Similar refinements in Scan 2 helped in speeding up the skeletonization process of the SPTA.

B. Testing Boolean expressions S_{04} and S_{26}

As discussed above, the Boolean expressions S_{04} and S_{26} are tested to decide on the safe-point status of a dark point p . It is apparent by looking at these Boolean expressions that, to reach a decision, the number of points examined in the neighbourhood of p depends upon the sequence in which these points are examined. Such a sequence may be heuristically decided by a user. For the data base of patterns described in Chapter IV, we found the decision trees given in Figure III.6 and Figure III.7 to be optimal. This means that the fewest number of points in the neighbourhood were examined before reaching a decision on the status of p . For our data base, we found this number to be 4.71, on an average.

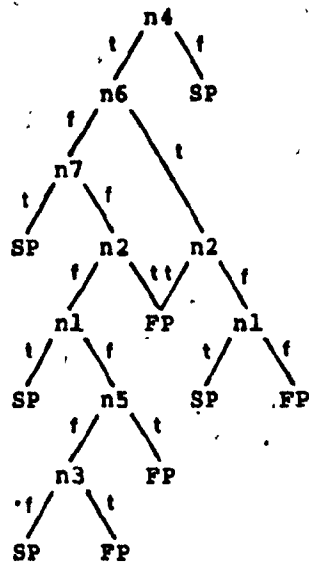


a. Decision tree
for Scan 1

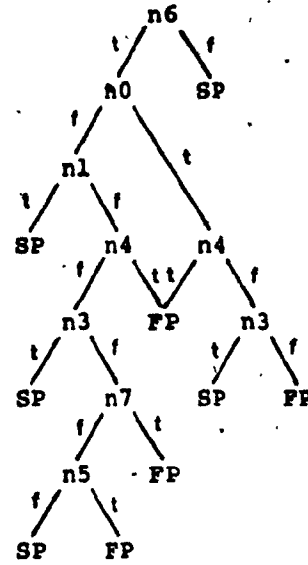


b. Decision tree
for Scan 2

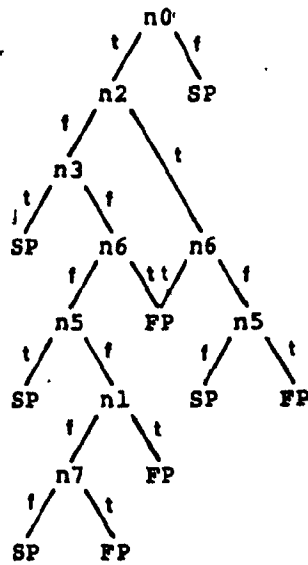
Figure III.6. The decision trees used to test Boolean expressions S_{04} and S_{26} . 'SP' indicates a safe-point. 't' stands for Boolean value TRUE, and 'f' stands for Boolean value FALSE. The decision trees for the expressions S_0 through S_6 are shown in Figure III.7.



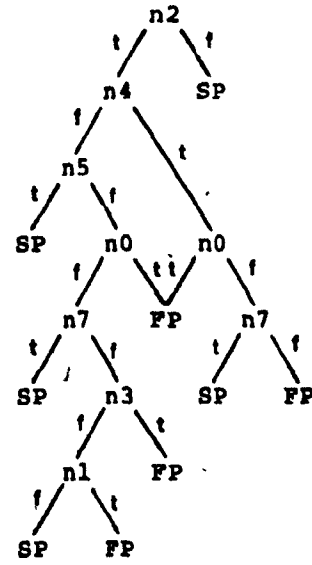
a. Decision tree for S_0



b. Decision tree for S_2



c. Decision tree for S_4



d. Decision tree for S_6

Figure III.7. The tree structure testing the Boolean expressions S_0 through S_6 .
 'n_j' indicates the 8-neighbour being visited (where $0 \leq j < 7$);
 'SP' means that p becomes a safe-point;
 'FP' means that p becomes a flag-point.
 't' stands for Boolean value TRUE, and
 'f' stands for Boolean value FALSE.

Chapter IV

Experimental Results of Skeletonization Algorithms

To compare the average performance of the SPTA to the 16 skeletonization algorithms reviewed in Chapter II, we tested all the algorithms on a data set of 648 hand-printed characters (letters 'A' to 'Z' and numerals '0' to '9'). The average size of the binarily digitized characters was 23 rows and 17 columns, with the maximum size being 40 rows and 35 columns. The characters were hand-printed by different students of Concordia University, Montreal, and were digitized by an ECRM 5200 auto-reader. The algorithms were coded in PASCAL 6000 Version 3.1 and run on a Control Data Cyber 170-835 computer, the coding being as intuitively efficient as possible.

IV.1. The Smoothing Technique

"As the result of a poor sampling system or transmission channel, the digitization operation may introduce in the pattern some spurious effects", called noise (Gonzalez and Wintz, 1977). In Figure II.1, we show a pattern with such noise. The points marked 'd' are dark points called 'pepper' noise. The points marked 's' are white points called 'salt' noise. Intuitively, these points are not part of the pattern, and they alter its shape property. Thus, when processing the pattern, such noise may lead to incorrect results. This is why in a large number of

skeletonization algorithms, this noise has to be removed before processing the pattern. This operation of noise removal is called smoothing. In the past, many hardware and software smoothing techniques for binary patterns have been discussed by the researchers (Budak, 1974; Doyle, 1960; Gonzalez and Wintz, 1977; Kohler and Howell, 1963; Rosenfeld, 1969; Stentiford and Mortimer, 1983; Unger, 1959; Weinberg, 1962). As some skeletonization algorithms we implemented required the input patterns be smoothed, and as we were looking for a very simple smoothing technique for the purpose of this thesis, we chose the one described in (Unger, 1959). We are now going to describe Unger's smoothing technique. We will also propose a modification to this technique.

In Unger's method, the input pattern is scanned rowwise, from left to right and top to bottom. Any dark point p is set to white if its neighbourhood does not match any of the two fundamental windows of Figure IV.2. On the other hand, any white point p is set to dark if at least three of its 4-neighbours are dark. For example, implementing Unger's method and testing it on the pattern of Figure IV.1, we obtained the smoothed pattern of Figure IV.3.

Investigating Unger's method, we noticed the following: if the neighbourhood of a dark point p matches the

configuration of Figure IV.4a, then p will be considered as a noise point and will ultimately be deleted. Thus, the connectedness of the pattern will no longer be maintained (Figure IV.4b). We conclude that for patterns containing lines of thickness 1, Unger's method for smoothing pepper noise is not applicable. We propose below a modification to Unger's smoothing technique.

In this modified version, salt noise are treated as in Unger's method. We also define:

$$R = \sum_r \sum_c f(r) \quad \text{and} \quad C = \sum_c \sum_r f(c)$$

where $r \in [-1,1]$, $c \in [-1,1]$, $(r,c) \neq (0,0)$

and where:

$$f(r) = r \quad \text{if} \quad n_j = \text{dark}, \quad j \in [0,7]$$

$$f(r) = 0 \quad \text{otherwise};$$

$$f(c) = c \quad \text{if} \quad n_j = \text{dark}, \quad j \in [0,7]$$

$$f(c) = 0 \quad \text{otherwise.}$$

Then, to test if a dark point p is a pepper noise, we do the following:

- 1- Set to p and its 8-neighbours the coordinates (r,c) , according to Figure IV.5.
- 2- Set p to white if and only if any of the two following conditions is TRUE:

$$a) \quad |R| = 3 \quad \text{OR} \quad |C| = 3, \quad \text{but not both.}$$

$$b) \quad |R| + |C| = 3 \quad \text{AND}$$

R and C are of opposite signs.

An exhaustive search of all possible combinations of the

8-neighbours of p showed us that this modified version of Unger's method will never break the connectedness of a pattern. Testing our modified version on the pattern of Figure IV.1, we obtained the same smoothed pattern as in Figure IV.3. It should be noted that for both Unger and modified Unger methods, two matrices are required in memory: one containing the original pattern, the other where the smoothed pattern is generated.

Testing the two smoothing techniques on our data base of 648 patterns, we found that the average cpu-time required to smooth one pattern was 21.40 ms when using Unger's technique, and 28.40 ms when using the modified Unger technique. Comparing the smoothed patterns obtained by these two methods, we noticed that, for our data base, they were identical. This means that no case of break in the connectedness of the patterns occurred by either Unger's technique or by its modified version. Thus, if any of the algorithms implemented in this thesis required the input pattern to be smoothed, we adopted the original Unger technique, thus saving time in computations. However, when dealing in other kinds of patterns, mainly where patterns may have lines of thickness equal to 1 (e.g., chromosomes or fingerprints), we strongly recommend the use of the modified version of Unger's smoothing technique.

```

**s
***
***
**s
***
s**
***
***
***
***
***
***
d*****
*****
*****
d d d

```

Figure IV.1. A specimen pattern to be smoothed. A '*' indicates a dark point, a 'd' a pepper noise, and a 's' a salt noise. That is, d is a dark point where it should ideally be a white point, and s is a white point where it should ideally be a dark point.

w	x	x
y	p	x
y	y	w

(i)

x	x	w
x	p	y
w	y	y

(ii)

Figure IV.2. The windows used in Unger's method for the deletion of pepper points. A 'w' indicates a "don't care" i.e., the darkness or whiteness of the corresponding point is immaterial. At least one 'x' and one 'y' must be dark.

```

**s
***
***
***
***
*** )
***
***
***
***
***
***
***
*****
*****
*****
ddd

```

Figure IV.3. The smoothed pattern of Figure IV.1, as obtained by Unger's method and by its modified version. We notice that not all 'd's have become white, nor all 's's have become dark.

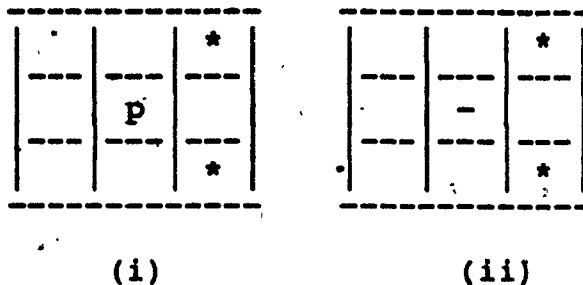


Figure IV.4. If Unger's method is applied to a dark point p whose neighbourhood satisfies the configuration (i), or its rotation by 0, 90, 180 and 270 degrees, then p will be deleted, breaking the connectedness of the pattern (configuration (ii)). A '*' indicates a dark point, a '-' a deleted point.

n3 (-1,-1)	n2 (-1, 0)	n1 (-1,+1)
n4 (0,-1)	p (0, 0)	n0 (0,+1)
n5 (+1,-1)	n6 (+1, 0)	n7 (+1,+1)

Figure IV.5. A point p and its 8-neighbours.
Below each point, we represent
the coordinates (r,c) of that
point. The point p is considered
to have coordinates (0,0).

IV.2. Experimental performance of the skeletonization algorithms

To give an intuitive feeling on the types of skeletons produced by the different algorithms, Figure IV.6 shows a specimen pattern from our data base. For this pattern, the skeletons produced by the 16 algorithms (reviewed in Chapter II) and by our SPTA, are shown in Figures IV.7a to IV.7q. The skeletons obtained using SPTA-SILT were identical to the ones obtained using SPTA-FILT. Algorithm HILD was implemented in its modified version (see Section II.6). Algorithm MAYA was implemented without making use of condition MAYA_7 (see Section II.7). In Table IV.1 we give for our data base of 648 patterns and for each of the 16 algorithms the following information: the average cpu-time it took to thin a pattern, the average number of passes required to thin a pattern, the average cpu-time required per pass, the connectedness (eight-connected or four-connected) of the skeletons produced, and whether the algorithm necessiated that the input pattern be smoothed. If smoothing was necessary, we have included the smoothing time in the average cpu-time required to thin a pattern. Table IV.2 shows the same information, but as it pertains to SPTA-SILT and SPTA-FILT.

We notice from Tables IV.1 and IV.2 that about half the number of algorithms necessiated smoothed patterns as input.

The three algorithms SRVH, SR4S and TM4F produced four-connected skeletons; all others produced eight-connected skeletons. Usually the algorithms require 2 to 4 passes to complete the skeletonization. If we are to rank the algorithms in ascending order of cpu-time required for skeletonization, we get the following sequence: SPTA-SILT, SPTA-FILT, PAVC, PAVR, PAVB, MAYA, SR4S, TM4E, TM4F, ARCL, BELA, HILD, BEUS, SR2S, TM2E, BEUO, ZHAN, SRVH. We concede that the ranking is not rigid. Indeed, we notice that some algorithms have a difference of a few milliseconds between their average cpu-time. For instance, although we have ranked ARCL (390.61 ms/pattern) before BELA (392.08 ms/pattern), we can say that their speeds are almost the same; thus, a slight improvement to the coding may affect the ranking. However, when the difference in speeds is no longer marginal (as for SPTA-SILT and PAVC), we expect that the ranking should not be affected when we try to get the coding even more efficient. In passing, we wish to make two comments. First, algorithm HILD is faster than algorithm SRVH; i.e., the algorithm originally proposed by Hilditch in (Hilditch, 1969) is faster than the algorithm (Stefanelli et al., 1971) which Stefanelli and Rosenfeld called a "simplified version" of Hilditch's algorithm. Second, Zhang and Suen had claimed that algorithm ZHAN is 50% faster than SR4S and SR2S (Zhang and Suen, 1984). Our experimental results showed that ZHAN is one of the slowest algorithms. Indeed, Zhang and Suen had tested algorithm

ZHAN on only 3 patterns. We notice that testing on such a small data set does not give reliable conclusions about the speed of an algorithm. Moreover, as shown in Section II.17, ZHAN can produce excessive erosion.

From above we see that SPTA is the fastest. The only algorithm as fast as SPTA is PAVC. However, PAVC does not have the reconstruction ability, whereas SPTA does. The only other algorithm which has reconstruction ability is PAVR, but it is slower than SPTA.

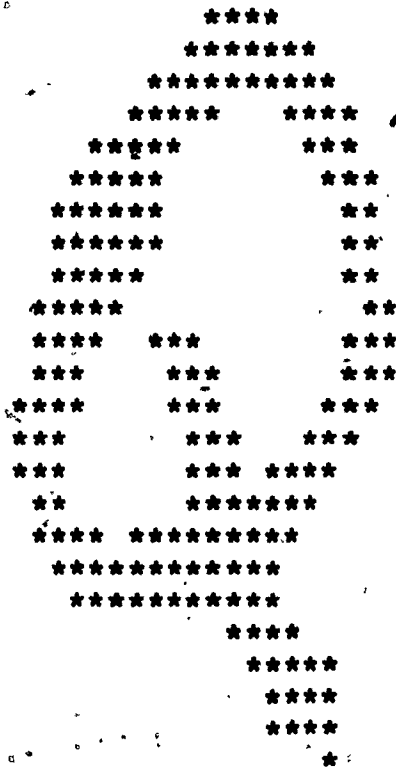


Figure IV.6. A specimen pattern.



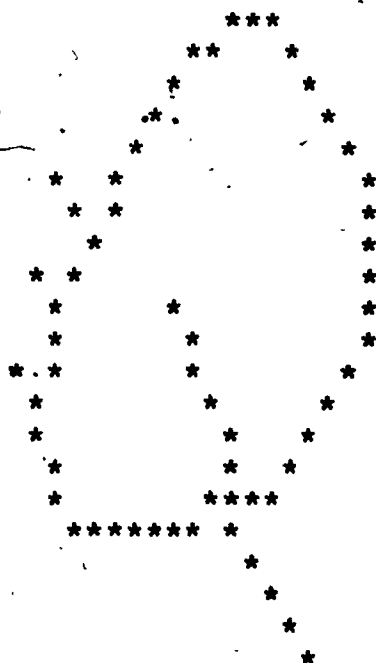
a. ARCL



b. BELA



c. BEUO



d. BEUS

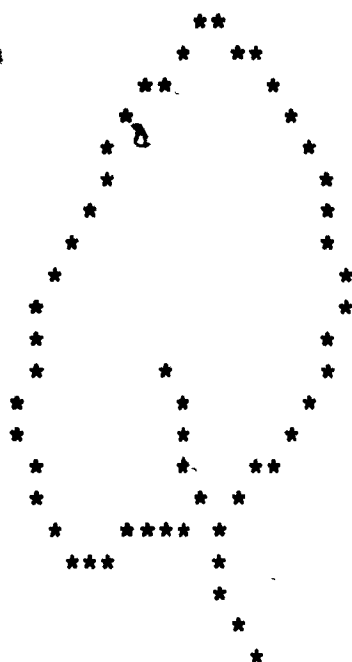


e. HILD



f. MAYA

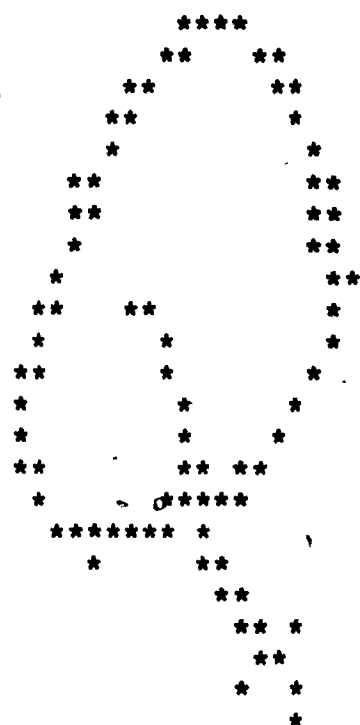
Figure IV.7. (Con't next page)



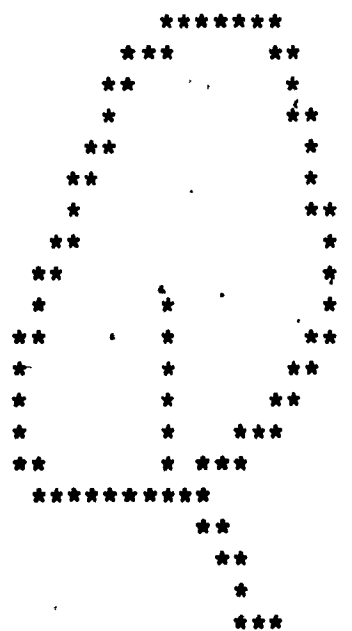
g. PAVC



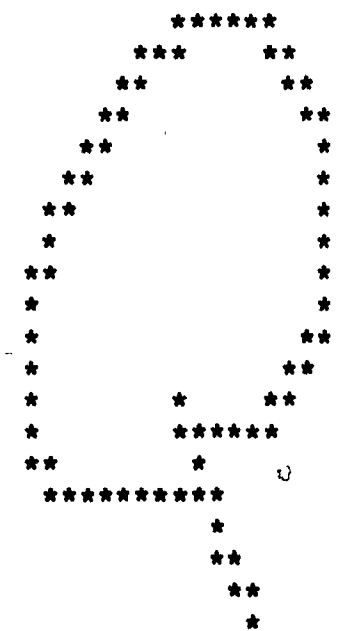
h. PAVB



i. PAVR



j. SRVH

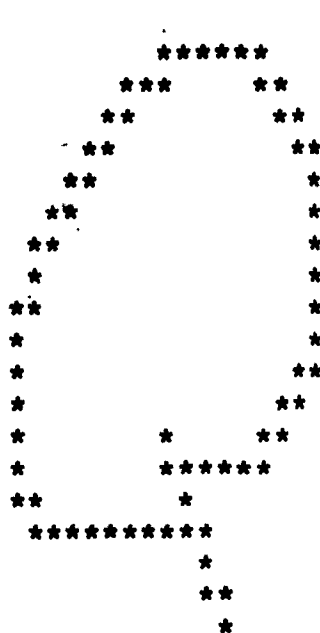


k. SR4S

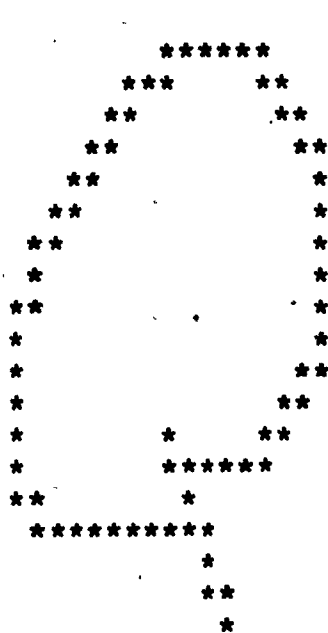


l. SR2S

Figure IV.7. (Con't next page)



m. TM4F



n. TM4E



o. TM2E



p. ZHAN



q. SPTA

Figure IV.7.
(Continued).
The skeletons
obtained by
applying the
17 algorithms
to the pattern
of Figure IV.6.
The name under the
skeleton indicates
the algorithm used
to obtain the skeleton.

ALGORITHM	AVG CPU-TIME PER PATTERN FOR THINNING	AVG NUMBER OF PASSES	CPU-TIME PER PASS	CONNECTEDNESS OF PATTERNS	SMOOTHING (*)
ARCL	390.61 MS	2.87	136.10	EIGHT	NO
BELA	392.08 MS	2.90	135.20	EIGHT	NO
BEUO	431.17 MS	3.46	118.43	EIGHT	YES
BEUS	395.67 MS	3.27	114.46	EIGHT	YES
HILD	393.12 MS	3.32	118.41	EIGHT	NO
MAYA	341.01 MS	3.26	98.27	EIGHT	YES
PAVC	186.22 MS	2.93	63.55	EIGHT	NO
PAVB	319.34 MS	3.91	81.67	EIGHT	NO
PAVR	224.04 MS	2.57	87.18	EIGHT	NO
SRVH	513.02 MS	4.33	118.48	FOUR	NO
SR4S	372.06 MS	2.23	157.25	FOUR	YES
SR2S	400.53 MS	3.00	126.38	EIGHT	YES
TM4F	380.34 MS	2.24	160.24	FOUR	YES
TM4E	375.40 MS	2.23	158.74	EIGHT	YES
TM2E	411.04 MS	3.04	128.17	EIGHT	YES
ZHAN	494.43 MS	2.80	176.52	EIGHT	NO

(*) This column indicates whether smoothing was required by the algorithm before thinning the pattern. Whenever smoothing was required, the column indicating the average cpu-time per pattern included smoothing time (for our data base, 21.40 ms per pattern).

ABBREVIATIONS USED:

ARCL : ARCELLI'S ALGORITHM (Arcelli, 1979)
 BELA : BEL-LAN'S AND MONTOTO'S ALGORITHM (Bel Lan et al., 1981)
 BEUO : BEUN'S ALGORITHM USING ORTHOGONAL NEIGHBOURS (Beun, 1973)
 BEUS : BEUN'S ALGORITHM USING SARAGA-WOOLLON'S CRITERIA (Beun, 1973)
 HILD : HILDITCH'S ALGORITHM (Hilditch, 1969)
 MAYA : MA-YUDIN'S ALGORITHM (Ma, 1983)
 PAVC : PAVLIDIS' CLASSICAL ALGORITHM (Pavlidis, 1982a)
 PAVB : PAVLIDIS' BASIC ALGORITHM (Pavlidis, 1982a)
 PAVR : PAVLIDIS' RECONSTRUCTABILITY ALGORITHM (Pavlidis, 1982a)
 SRVH : STEFANELLI-ROSENFELD'S VERSION OF HILDITCH'S ALGORITHM
 (Stefanelli and Rosenfeld, 1971)
 SR4S : STEFANELLI-ROSENFELD'S ALGORITHM WITH 4 SCANS PER PASS
 (Stefanelli et al., 1971; Rosenfeld, 1975)
 SR2S : STEFANELLI-ROSENFELD'S ALGORITHM WITH 2 SCANS PER PASS
 (Stefanelli et al., 1971; Rosenfeld, 1975)
 TM4F : TAMURA'S ALGORITHM ENSURING FOUR-CONNECTEDNESS (Tamura, 1978)
 TM4E : TAMURA'S ALGORITHM ENSURING EIGHT-CONNECTEDNESS (Tamura, 1978)
 TM2E : TAMURA'S ALGORITHM WITH 2 SCANS PER PASS (Tamura, 1978)
 ZHAN : ZHANG-SUEN'S ALGORITHM (Zhang et al., 1984)

TABLE IV.1. Experimental results of the 16 skeletonization algorithms reviewed in Chapter II. 648 patterns were skeletonized by each algorithm.

ALGORITHM	AVG CPU-TIME PER PATTERN FOR THINNING	AVG NUMBER OF PASSES	CPU-TIME PER PASS	CONNECTEDNESS OF PATTERNS	SMOOTHING (*)
SPTA-SILT	122.17 MS	3.31	30.46	EIGHT	YES
SPTA-FILT	139.79 MS	3.31	35.78	EIGHT	YES

(*) This column indicates whether smoothing was required by the algorithm before thinning the pattern. Whenever smoothing was required, the column indicating the average cpu-time per pattern included smoothing time (for our data base, 21.40 ms per pattern).

ABBREVIATIONS USED:

SPTA-SILT : THE SAFE-POINT THINNING ALGORITHM,
 SINGLE INTEGER LABELLING TECHNIQUE
SPTA-FILT : THE SAFE-POINT THINNING ALGORITHM,
 FOUR INTEGERS LABELLING TECHNIQUE

TABLE IV.2. Experimental results of SPTA-SILT and SPTA-FILT.
648 patterns were skeletonized by each algorithm.

IV.3. Comparison of the reconstruction techniques

Table IV.3 shows our experimental results for thinning and reconstruction, comparing SPTA to PAVR. On an average, PAVR required 224.04 milliseconds for thinning, whereas SPTA required 122.17 milliseconds with SILT and 139.79 milliseconds with FILT. We thus can say that SPTA with either of the two labelling techniques is faster than PAVR.

To give an intuitive feeling on the quality of skeletonization, we show two specimen patterns in Figure IV.8, which were thinned by PAVR (Figure IV.9) and by SPTA (Figure IV.10). The patterns of Figure IV.8 do not belong to our data base, but were taken from Pavlidis (Pavlidis, 1981). We emphasize that the choice of the labelling technique in SPTA does not affect the skeletons produced. As we can see from Figure IV.9, PAVR produced skeletons with spurious tails and strokes of thickness of more than 1. This is not so for SPTA. In fact, we noticed that for most of the patterns in our database PAVR suffers from such drawbacks, whereas SPTA does not. However, both algorithms produced connected skeletons.

We also compared the reconstruction ability of SPTA to PAVR's. For SPTA with SILT, the reconstruction algorithm was identical to Pavlidis's (see Chapter II, Section II.10), whereas for SPTA with FILT the reconstruction technique was

an extension of Pavlidis's (see Chapter III, Section III.4). Testing on our database of 648 patterns, we observed the following: when the reconstructed pattern was superimposed on the original pattern and a point-to-point match was tested, PAVR always gave a 100% match. SPTA with SILT gave on an average a 94.1% match, and SPTA with FILT gave 98.6%. Figures IV.11 and IV.12 show the reconstructed patterns from the skeletons of Figure IV.10 using SPTA-SILT and SPTA-FILT, respectively. Since PAVR gave 100% point-to-point match, we do not show the reconstructed patterns, because they are identical to Figure IV.8. Although SPTA did not give 100% reconstruction but nearly 100%, it ensures that the reconstructed pattern retains the essential shape information of the original pattern. Table IV.3 shows that on an average the reconstruction time is 29.73 milliseconds with PAVR, 31.10 milliseconds with SPTA-SILT, and 51.49 milliseconds with SPTA-FILT. Thus, we see that PAVR is the fastest and also has perfect reconstruction ability. In fact, Pavlidis proves in (Pavlidis, 1981) that his algorithm would always give 100% reconstruction. But it is our contention that this 100% reconstruction by PAVR has been achieved at some cost. This cost is the presence of (a) spurious tails and (b) strokes of thickness greater than 1 in the skeletons of PAVR. By retaining spurious tails and thick strokes in the skeletons, PAVR is able to retain more dark points. But we believe that it is a generally accepted principle that the presence of spurious tails and thick

strokes in the skeletons diminishes the quality of the skeletons. Taking this into consideration, SPTA produces better skeletons than PAVR. Moreover, its reconstruction, though not perfect, is nearly perfect, with SPTA-FILT being slightly better than SPTA-SILT, noting however that SPTA-FILT is slower than SPTA-SILT.

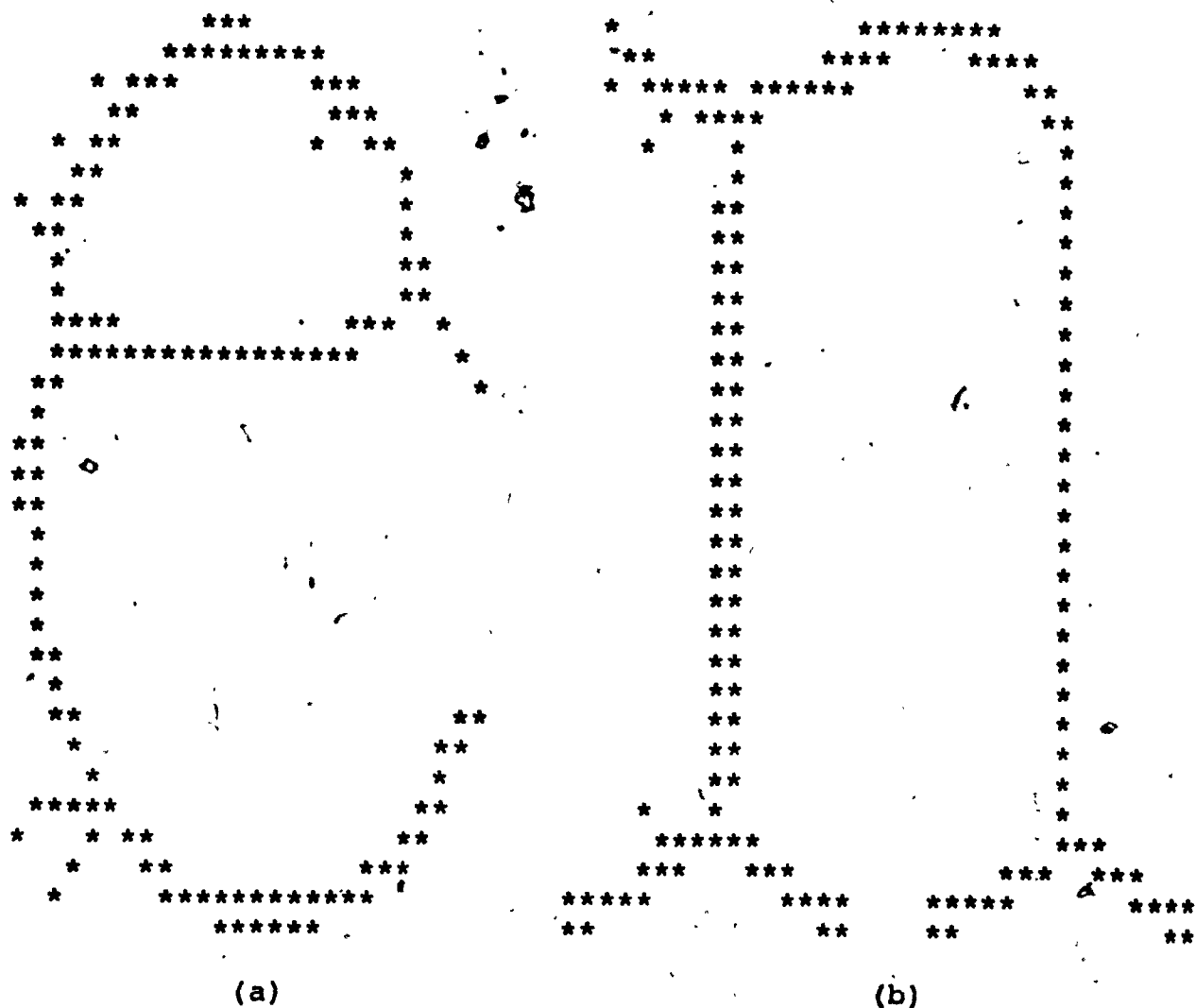


Figure IV.9. The skeletons of the patterns of Figure IV.8, obtained using Pavlidis's algorithm PAVR. Cpu-time required was 1037 ms to thin pattern (a), and 1280 ms to thin pattern (b).

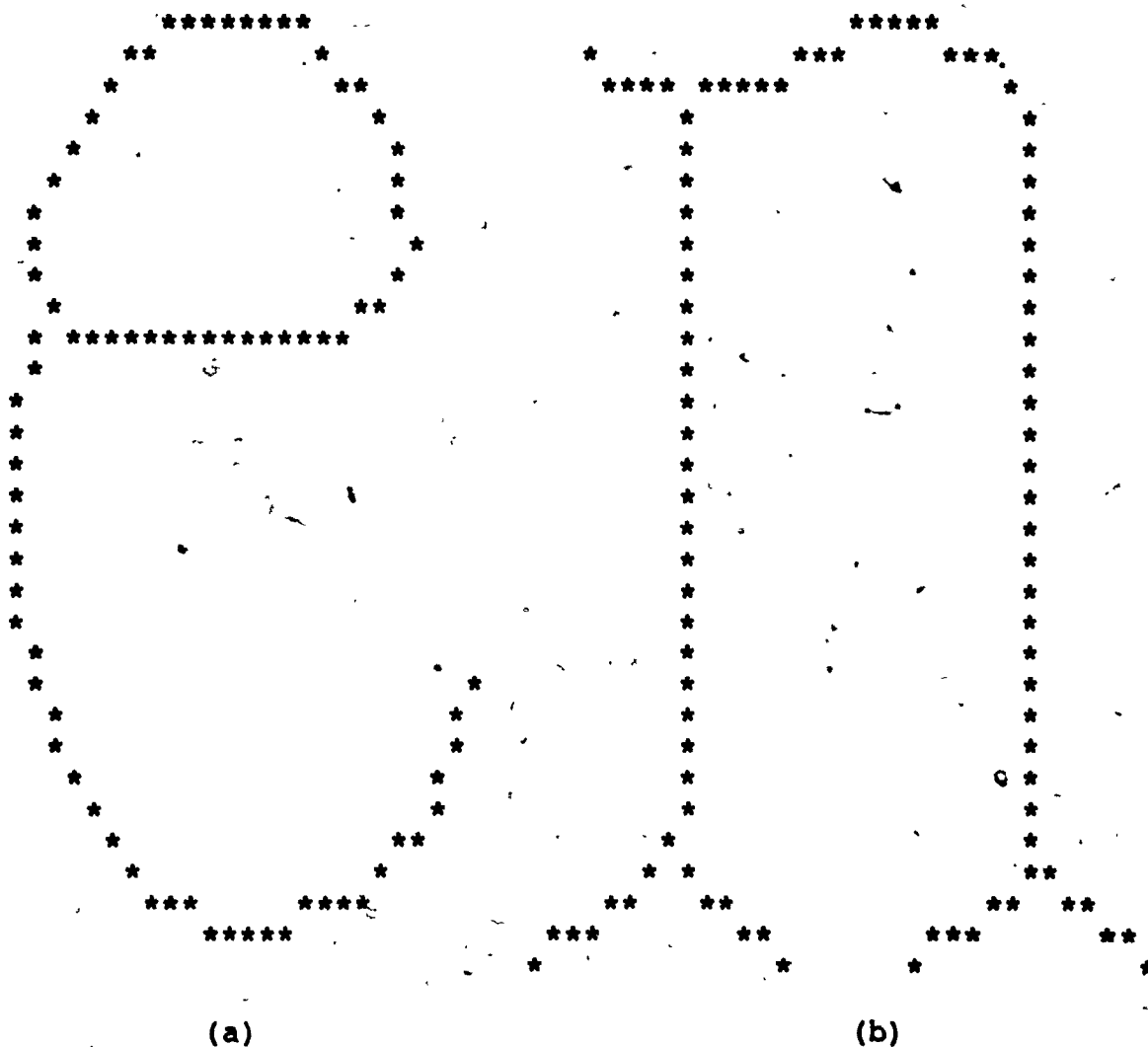


Figure IV.10. The skeletons of the patterns of Figure IV.8, obtained using the Safe-Point Thinning Algorithm. Cpu-time required to thin pattern (a) was 464 ms using SILT and 515 ms using FILT. Cpu-time required to thin pattern (b) was 584 ms using SILT and 648 ms using FILT. For a given pattern, SPTA-SILT and SPTA-FILT produced identical skeletons.

ALGORITHM USED	AVG CPU-TIME FOR SKELETO- NIZATION (*)	RECONST. CPU-TIME (*)	POINT-TO-POINT MATCH IN RECONSTRUCTION
PAVR	224.04 ms	29.73 ms	100.0 %
SPTA-SILT	122.17 ms	31.10 ms	94.1 %
SPTA-FILT	139.79 ms	51.49 ms	98.6 %

(*) These are the average cpu-times per pattern for our database.

ABBREVIATIONS USED

PAVR : Pavlidis's Algorithm
with Reconstruction Ability (Pavlidis, 1981)
SPTA-SILT : Safe-Point Thinning Algorithm
with Single Integer Labelling Technique
SPTA-FILT : Safe-Point Thinning Algorithm
with Four Integers Labelling Technique

TABLE IV.3.

Experimental results of the three skeletonization and reconstruction algorithms. 648 patterns were processed by each algorithm.

IV.4. Excessive erosions in the skeletonization algorithms

We further tested all 16 algorithms and SPTA for excessive erosion, by using the slanting stroke of width 2 of Figure IV.3a. The three algorithms BEUO, SRVH and ZHAN caused excessive erosion; i.e., the stroke was iteratively deleted. For all the other algorithms, no excessive erosion was noticed in thinning the stroke. We also tested the algorithms on a set of geometric patterns, often used to show what thinning algorithms can achieve (Davies et al., 1981). In Figure IV.13a we show a specimen of such a pattern. The skeletons of Figures IV.13b, IV.13c and IV.13d were obtained by SPTA, ARCL and SR2S, respectively. The algorithm BELA gave a skeleton similar to the one of Figure IV.13b. Algorithms BEUO, BEUS, HILD, MAYA, PAVB, PAVR and ZHAN gave skeletons similar to the one shown in Figure IV.13c. All the other algorithms (PAVC, SRVH, SR4S, TM4F, TM4E and TM2E) reduced the pattern to a small set of dark points at the centre of the pattern, as shown in Figure IV.13d. Clearly, these were cases where the shape of the original pattern was no longer retained.

Thus we can conclude that our SPTA has experimentally been shown to be the fastest of the 18 algorithms we have tested, it produces good quality skeletons, and it has a strong reconstruction ability.

a. A specimen geometric pattern.

b. The skeleton obtained using SPTA. A similar skeleton was obtained using BELA..

c. The skeleton obtained using ARCL. Similar skeletons were obtained using BEUO, BEUS, HILD, MAYA, PAVB PAVR and ZHAN.

d. The skeleton obtained using SR2S. Similar skeletons were obtained using PAVC, SRVH, SR4S, TM4F and TM2E.

- 124 -

Chapter V

A Multiprocessor Network for Skeletonization Algorithms

In this chapter, we propose a multiprocessing network for skeletonization algorithms, called SKELNET. We describe the functional structure of SKELNET and its hardware design. Because of a lack of appropriate equipment at Concordia University, Montreal, the network was not implemented but simply simulated with a software program. We describe in this chapter the software simulation of SKELNET. Later, we give the experimental results of the simulation of the network.

V.1. Definition of Sequential and Parallel Skeletonization Algorithms

Reviewing the published literature, we found two kinds of skeletonization algorithms: the sequential skeletonization algorithms, and the parallel skeletonization algorithms. Amongst the sequential skeletonization algorithms (referred to as SQA) are HILD (Hilditch, 1969), MAYA (Ma, 1983), SRVH (Stefanelli et al., 1971) and SPTA (Naccache et al., 1984b) described in Chapters II and III. In these algorithms, the status that a dark point takes during any pass i over the pattern, depends on the status of the 8-neighbours of p during the same pass. Amongst the parallel skeletonization algorithms (referred to as PRA), are

all the other algorithms reviewed in Chapter II. In these algorithms, the status that a dark point p takes during any pass i over the pattern, depends on the status that its 8-neighbours had at the end of (previous) pass $i-1$. Although SQAs and PRAs are "mathematically equivalent" (i.e., they all aim at reducing the pattern to its medial axis (Rosenfeld et al., 1966)), the type of computer required to implement them is not the same. SQAs are designed to be implemented on sequential computers. Ideally, PRAs would need computers having one processor per point of the pattern, all the points being processed simultaneously during the same pass. However, such multiprocessor systems may become very expensive. Furthermore, we have seen in Chapter IV that some sequential algorithms (e.g., HILD, SPTA) give good skeletons, and should not be rejected at the cost of PRAs operating on expensive multiprocessor systems.

During recent years, with the fast development of array processors and pipe-line computers (Evans, 1982) and of VLSI technology, many researchers (Duff, 1976; Hanaki and Temma, 1982; Sternberg, 1982) have tried to develop parallel processing systems for PRAs. Unfortunately, the large amount of processors implied makes these systems uneconomic, and cannot be afforded by small industries and clinics involved in Image Processing applications. We thus propose a simple and inexpensive multiprocessing system, called

SKELNET, for skeletonization of binary patterns. As we shall see later in this chapter, this network can be generalized to other domains of Image Processing. Below, we give the functional description of SKELNET, followed by the experimental results of its software simulation.

V.2. The fundamental requirements of the multiprocessor network

Essentially, there is a need for a system where a pass i over the pattern could start before waiting for pass $i-1$ to finish. Since the status a dark point p takes during a pass i depends on the status of its 8-neighbours during the same pass, it should suffice for each processor of the point to lock the access to p and to its 8-neighbours from any other processor. That is, if a processor P is testing for deletion a point p at location (r,c) (see Figure IV.5), then no other processor can simultaneously test for deletion any point in the area $A(r,c) = ([r-1,r+1],[c-1,c+1])$; i.e., any of the 8-neighbours of p .

However, the task is not as easy as it may appear at a first glance. Indeed, suppose that a processor P wants to test a dark point p at location (r,c) . In order to do so, the processor P may have to examine all the 8-neighbours of p , n_0 through n_7 . Now suppose that, at the same time, another processor P' wants to test one of the dark

8-neighbours of p , say n_2 . In order to do so, the processor P' may have to examine all the 8-neighbours of n_2 . Thus, the following situation will occur: the processor P would try to lock the area $A(r,c)$, where $n_2 \in A(r,c)$; simultaneously, the processor P' would try to lock the area $A(r-1,c)$, where $p \in A(r-1,c)$. Thus, the two processors P and P' would try to lock simultaneously the non-disjoint areas $A(r,c)$ and $A(r-1,c)$. This situation could produce a deadlock. That is, each processor P and P' would wait indefinitely for each other to unlock the area which is under its control.

Moreover, in SQAs, the scanning sequence is important. Indeed, two fundamental situations may occur:

- 1- a processor P examines a point p before a processor P' examines any of the 8-neighbours of p ;
- 2- a processor P' examines any of the 8-neighbours of a point p before a processor P examines p .

Thus, the skeletons obtained in the first situation may not be the same as the ones obtained in the second situation. We say that SQAs are time-dependent; i.e., it may happen that a repeated skeletonization of the same pattern, with one given skeletonization algorithm, may not always lead to the same skeleton.

In order to prevent deadlocks and to avoid getting into an unnecessarily complicated system, we have set up the

following rules:

Rule 1: Instead of locking the area $A(r,c)$, a processor shall lock the area $Br = ([r-1, r+1], [1, WIDTH])$, where $WIDTH$ is the maximum width of the input pattern. In other words, a processor shall lock all three rows $r-1$, r and $r+1$ of the input pattern.

Rule 2: If a processor P starts scanning the pattern before a processor P' , then the processor P' shall always remain behind P . In other words, each processor is assigned a fixed priority with respect to the others.

The consequent of Rule 1 and Rule 2 is that the processor P' shall never skip ahead of P . Thus, P' shall always remain at least two rows behind the processor P . Abiding by these rules, we designed the network which is shown in Figure V.1. This figure shows the block-level architecture of SKELNET. We give below a detailed explanation of this block-level architecture.

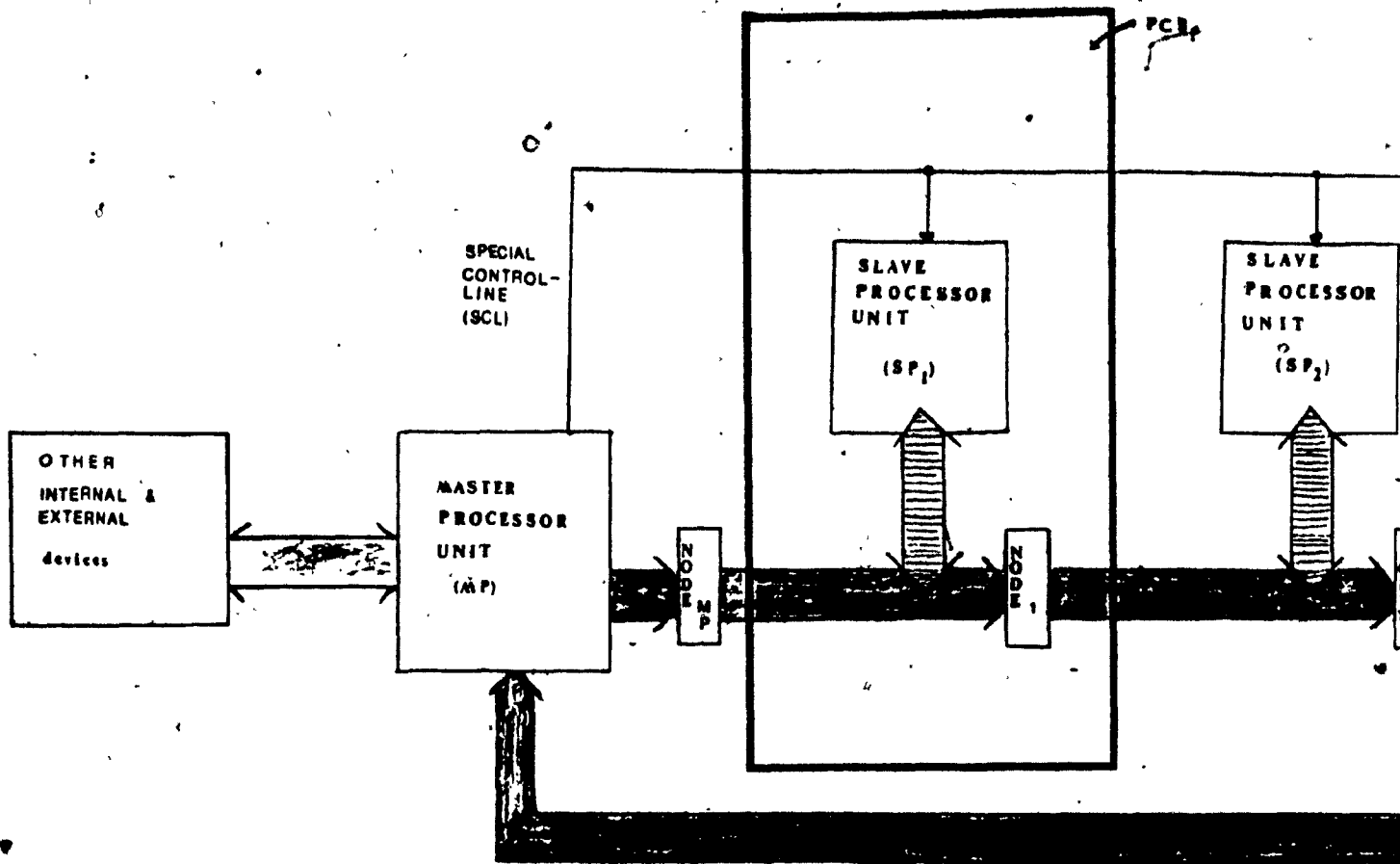


Figure V.1. The block-diagram of SKELNET.

The dark arrow shows the pipe-line.

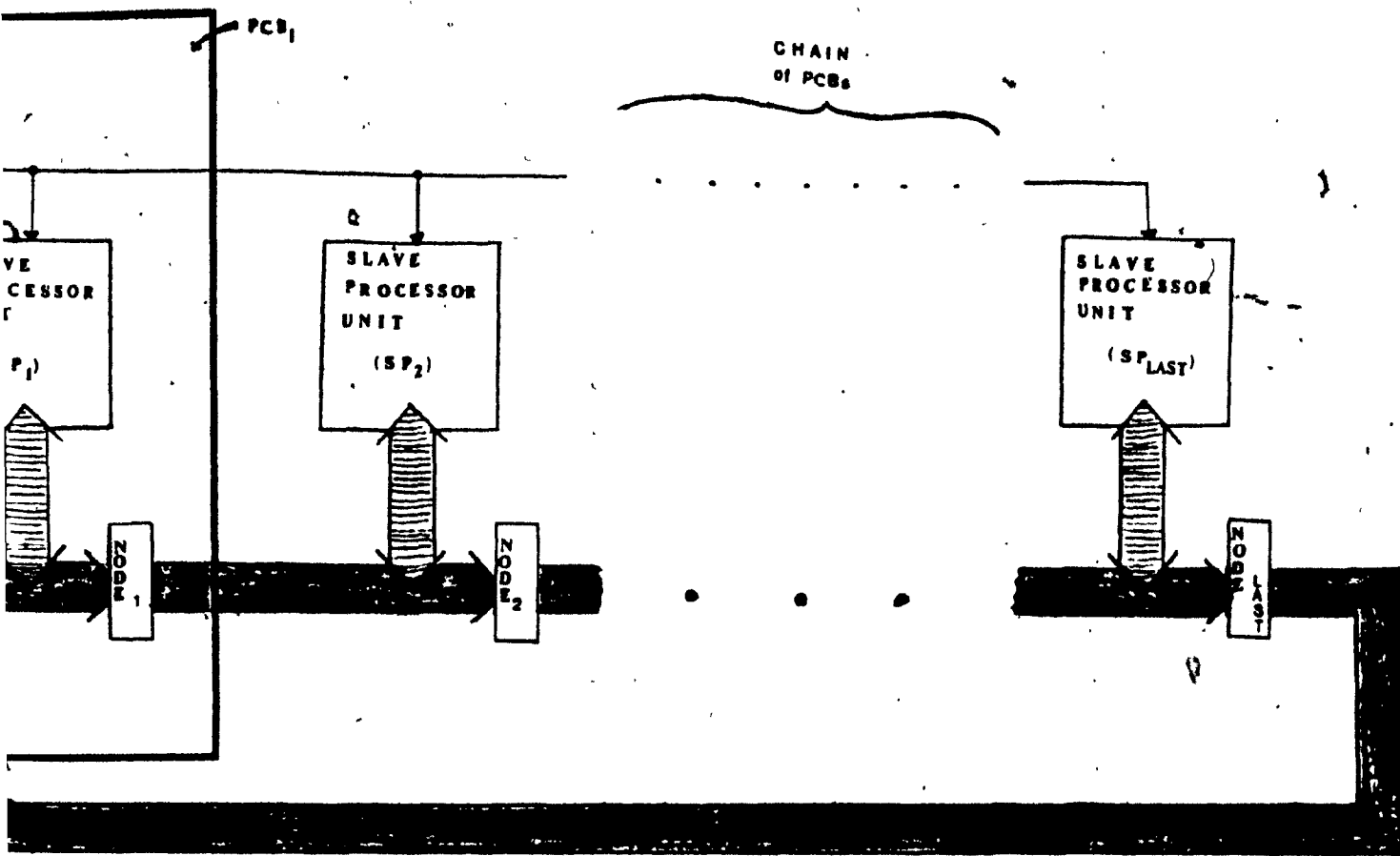


diagram of SKELNET.

now show the pipe-line.

V.3. The block-level architecture of SKELNET

In the discussion which follows, we will consider a theoretical skeletonization algorithm, consisting of only one scan per pass. Later, we will generalize the case to algorithms requiring more than one scan per pass.

The microprocessor network SKELNET essentially consists of a Master Processor unit (MP) and of a set of Slave Processor units (SPs). The MP drives the whole skeletonization process. The SPs perform the skeletonization process itself.

The MP consists of the following major components:

1. A Micro-Processor Unit (MPU).
2. Random Access Memory (RAM), large enough to store the entire input pattern and some local variables.
3. Programmable Read-Only Memory (PROM), storing the main Operating System, the power-up procedures and system diagnostics, and the executable program driving the skeletonization process.

The Master Processor, which is also part of the main working-station, receives the input pattern from an appropriate device (i.e., a scanner, a disk, or any external device), and stores it into its own Random Access Memory. Then, the MP sends the data (i.e., the pattern) to a linear chain of Slave Processors. The role of each SP is to

process the data. The data is cascaded to all the SPs through a pipe-line (shown in dark in Figure V.1). This pipe-line consists of a linear chain of nodes (NODE), where each node is a transit stage for the data. The pipe-line is circular; that is, after being processed by the last SP of the chain, the data is sent back to the MP. Moreover, the Master Processor is also connected to all SPs by a special control-line, SCL. This SCL is used for initialization purpose of all SPs.

In SKELNET, a Slave Processor consists of the following major components:

1. A Micro-Processor Unit (MPU).
2. Random Access Memory (RAM), capable of storing 3 rows of the input pattern, some local variables and a local flag called the Scan-Flag (or SFLAG).
3. Programmable Read-Only Memory (PROM), capable of storing the executable program of the skeletonization process itself.

Each SP is part of a Printed-Circuit Board (PCB). A PCB consists of the following functional components:

1. One Slave Processor (SP).
2. One node of the pipe-line (NODE).
3. A hardware interface connecting the node to the SP on one hand, and to the preceding node of the chain on the other hand.

Moreover, the PCBs are sequentially numbered 1,2,3,... Thus, PCB_j represents the PCB at position j in the linear chain, where $j=LAST$ for the last PCB of that chain. Each component of the PCB is labelled with the number of the PCB. Thus, each SP "knows" its position number j in the chain.

V.4. The skeletonization process of SKELNET

In what follows, we assume that all the processor units have been reset through a common switch, and that the power-up procedure has already taken place. Again, we emphasize that SP_1 receives the data from the MP, and that SP_{LAST} sends it to the MP, after processing.

Initially, the MP reads into its RAM the pattern to be skeletonized and triggers the control-line, thus informing all the SPs to perform an initialization operation, of the components of their PCB, namely: set SFLAG to FALSE, set the pass number i of the skeletonization process to the value of j , initialize some local variables in the RAM, and prepare to process the data to be received. The MP also sends to all SPs the following information, as a command-line: the number of rows of the pattern (ROWCOUNT), and the number of slaves in the chain (LAST). Then, the MP starts sending one row of the pattern at a time to the first node of the pipe-line. This row is cascaded through all the nodes of the pipe-line, abiding by the following sequence of

operations in every PCB_j :

- 1- The Slave Processor SP_j receives the rows from $NODE_{j-1}$ and stores them in RAM_j ;
- 2- Upon reception of 3 rows ($r-1$, r and $r+1$), SP_j performs pass i of the skeletonization process over row r ;
- 3- SP_j sends row $r-1$ to $NODE_j$;
- 4- SP_j shifts up rows r and $r+1$. This operation can be done by relabelling row r as row $r-1$ and row $r+1$ as row r ;
- 5- SP_j decrements ROWCOUNT by 1;
- 6- SP_j awaits for the next row to be received from $NODE_{j-1}$. When received, SP_j stores the row in RAM_j as row $r+1$, and performs pass i of the skeletonization process over the newly labelled row r ;
- 7- SP_j repeats the steps 3 through 6 until ROWCOUNT = 0; i.e., until one pass of the skeletonization process over the pattern is terminated.

During any one pass, if a Slave Processor SP_j flags any dark point p of the rows it processes, then $SFLAG_j$ is set to TRUE. The MP sends all the rows of the pattern as a continuous stream of data. When SP_{LAST} finishes processing one row r , the PCB_{LAST} requests an interrupt from MP. An interrupt service routine is then enabled, and MP stores back the processed row r into its RAM. Thus, the whole pattern which had been sent by the MP is received back as processed by the SPs. Since each SP has performed one pass over the pattern and since there are 'LAST' number of SPs, the MP receives back the pattern as thinned after 'LAST'

passes.

We now examine the termination criterion of the skeletonization process. When all the rows are received back from PCB_{LAST} , the Slave Processor SP_{LAST} sends to the MP the value of $SFLAG_{LAST}$. If the Scan-Flag is FALSE, then it means that no points have been flagged by SP_{LAST} ; thus, the skeletonization process over the pattern stops. If the Scan-Flag is TRUE, then it means that more passes over the pattern are still required; thus, the whole set of operations is repeated again with, this time, each Slave Processor SP performing the pass $i = i + LAST$.

We can see from the above that the two rules established earlier are met:

- (1) The area Br is locked by each PCB , and this by the hardware structure of SKELNET (Rule 1);
- (2) All SP s being set up in a linear chain, each SP is automatically assigned a fixed priority over the next SP of the chain (Rule 2).

The network described above can be applied on most skeletonization algorithms. For instance, for the Safe-Point Thinning Algorithm (SPTA) discussed in Chapter III, each Slave Processor would perform one scan over the pattern instead of one pass. If SP_j performs Scan 1 of pass i , then SP_{j+1} would perform Scan 2 of the same pass.

Therefore, Scan 2 will always be 2 rows behind Scan 1 (Rule 2). Thus, in SPTA the termination criterion is tested as follows: instead of receiving the value of SFLAG from PCB_{LAST} only, the MP receives SFLAG from both PCB_{LAST} and PCB_{LAST-1}. The value of SFLAG_{LAST} thus indicates if Scan 2 has to be initiated, and SFLAG_{LAST-1} indicates if Scan 1 has to be initiated. As mentioned in Chapter III, it may happen that Scan 1 is not initiated while Scan 2 is still initiated. In this case, it becomes necessary to by-pass the Slave Processors which are dedicated to performing Scan 1. Thus, before sending again the stream of data, the MP informs through a command-line all the SPs dedicated to Scan 1, to temporarily disconnect from the pipe-line. This command-line is sent through the pipe-line. Upon reception of such a command, the SP_j will disconnect itself from the pipe-line. As a result, the flow of data will be directly transferred from NODE_{j-1} to NODE_j, without being interfaced to SP_j.

Now that we have explained the functional structure of SKELNET, we give below its pin-description shown in Figure V.2. This figure shows the components of one PCB. It is generalized to all PCBs.

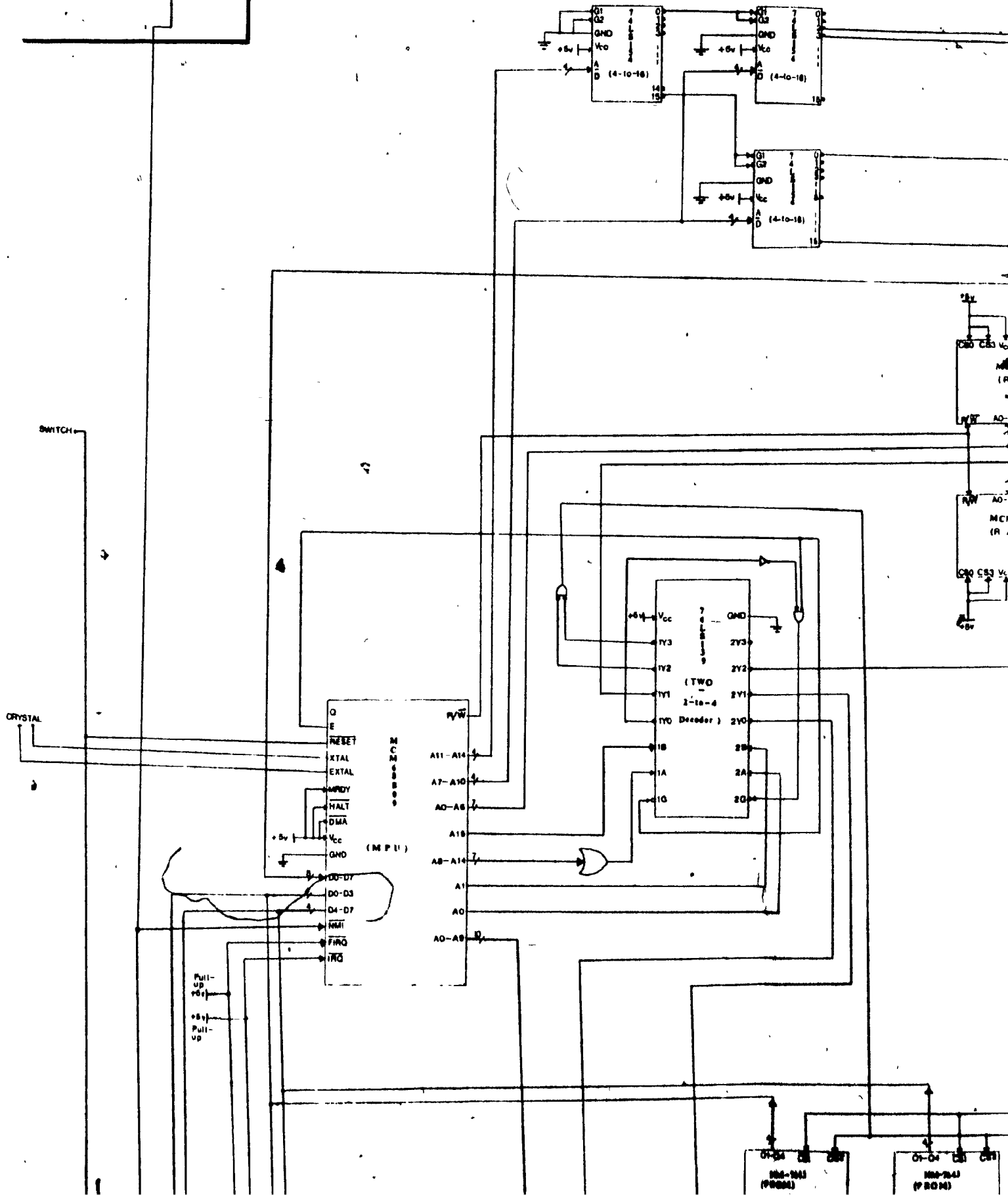
V.5. The pin-description of SKELNET

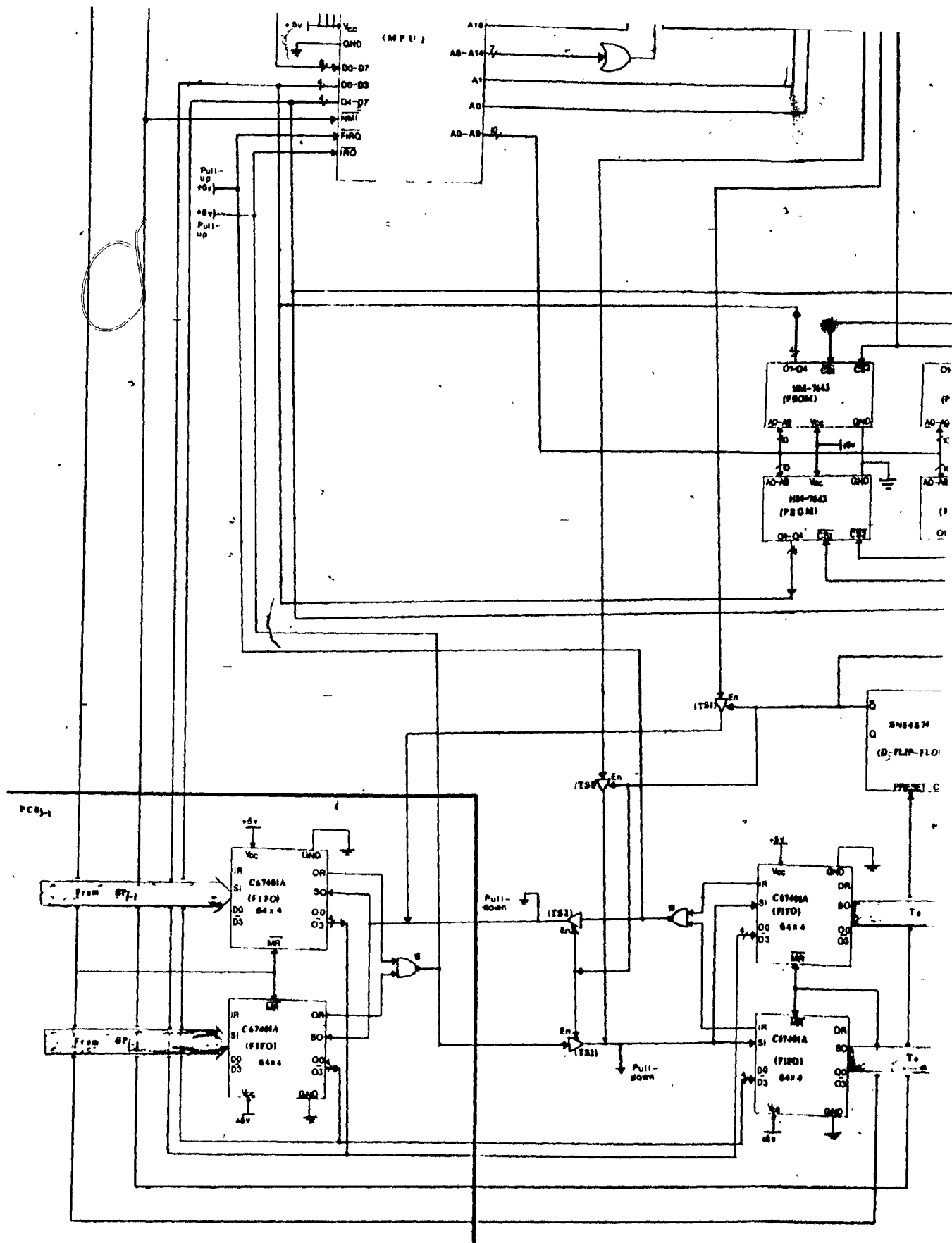
Below we give the pin-description of the Master

10F

MASTER PROCESSOR
UNIT

74LS123
(MSV)
Q Q





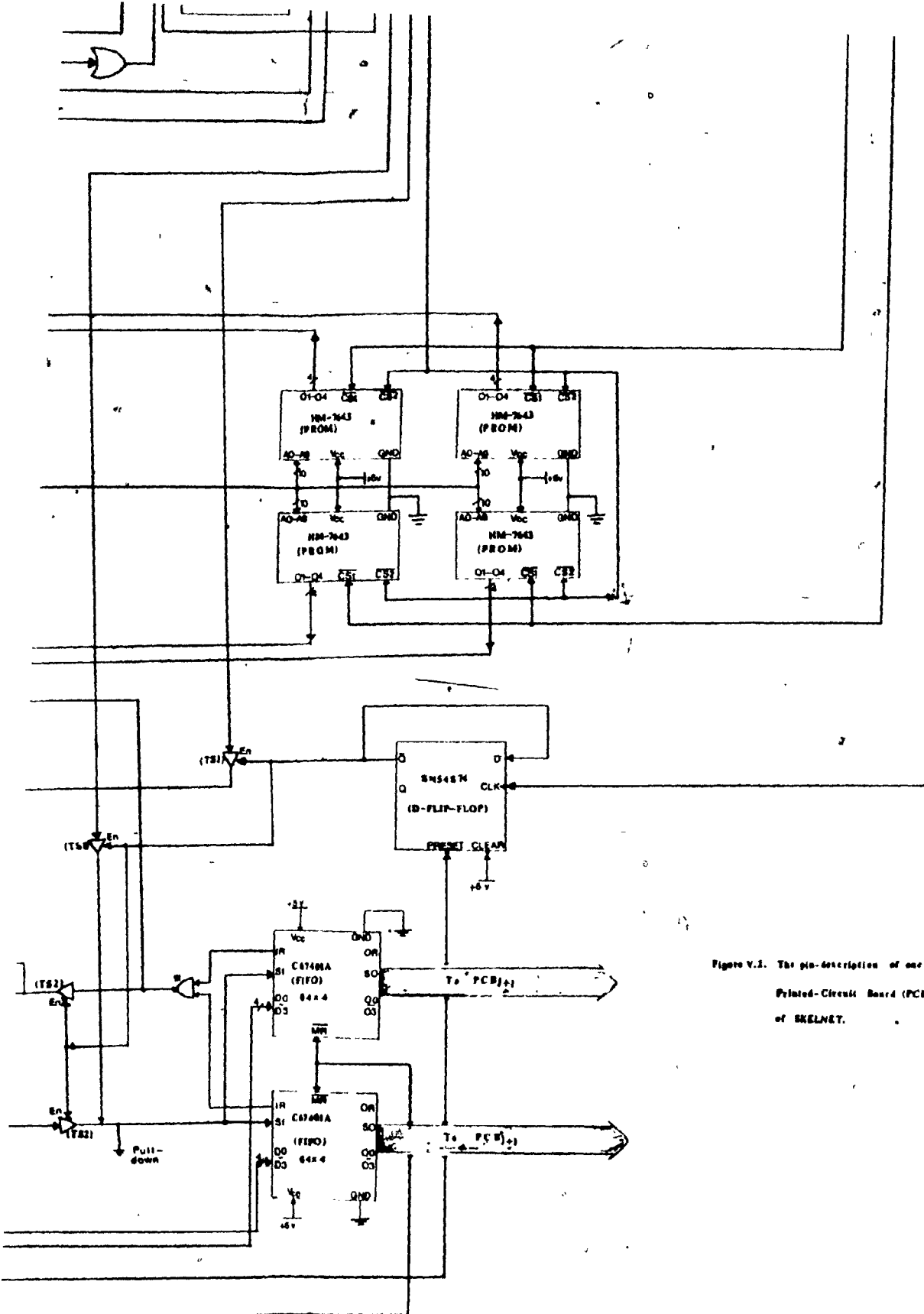


Figure V.1. The pin-description of one
Printed-Circuit Board (PCB)
of SKELNET.

Processor unit, followed by the pin-description of the Slave Processor units. For detailed descriptions of the chips used in SKELNET, we refer the reader to the pertinent data books (Fletcher, 1980; Motorola Data Manual, 1981; TTL Data Book, 1981). In Section V.6, we explain how the different units interact. The network SKELNET has been designed to accept 64x64 binary patterns, and the description is given with respect to this constraint. However, SKELNET can be generalized to accept larger patterns.

Master Processor unit

The MP is part of the main working-station of the system. We describe below only the components that play a key-role in SKELNET.

The microprocessor unit of the MP is an 8-bit 6809 chip (numbered MC68B09), whose frequency is 2 MHz. An 8 KBytes Random Access Memory (RAM) unit is required: 4 KBytes are required to store the pattern, the other 4 KBytes being used as working storage. A 4 KBytes Programmable Read-Only Memory (PROM) unit is also required: it contains the executable program to drive the skeletonization process. The MP also contains a retriggerable monostable multivibrator (74LS122). This multivibrator triggers the special control-line SCL of SKELNET. It generates a pulse, programmed by the selection of external resistance and

capacitance values. The output pulse can be triggered either on a rising edge (pin Q of the multivibrator in Figure IV.2) or on a falling edge (pin \bar{Q}). For a reason explained in the description of the Slave Processor unit, the width of the pulse must be adjusted to approximately 430ns.

Slave Processor unit

The pin-description of the SP is shown in Figure IV.2. It also shows the connection of the SP_j to the MP, and a segment of the pipe-line with its interface to the SP, namely: NODE_j of SP_j and NODE_{j-1}.

The microprocessor unit of the SP is an 8-bit 6809 chip (MC68B09), whose frequency is 2 MHz. A two 2-to-4 line decoder (74LS139) is used for chip selection, according to the following mapping (the addresses are given in hexadecimal):

1. Address 0000 to 00FF for interfaces.
2. Address 0100 to 7FFF for Random Access Memory.
3. Address 8000 to FFFF for Read-Only Memory.

Thus, there are 32K addresses possible for interfaces and RAM altogether, and 32K addresses for PROM. The selection of the memory chips is done through a tree of 4-to-16 line decoders (74LS154). For all the decoders, the inputs are asserted high and the outputs are asserted low. The inputs

of the root of the tree are the 4 high order bits of A7-A14. The output lines are used to select one child of the root. The inputs of each child are the 4 low order bits of A7-A14. The output lines of the children are used to select any memory chip. Thus, up to 256 chips can be selected using this tree structure.

In SKELNET, there are two 128x8-bit RAMs (MCM68B10), whose maximum access time is 250ns. Thus, the 256 bytes of RAM can store three rows of the input pattern (192 bytes), and the local variables (64bytes). The addresses of these two RAMs range from 0100 to 01FF.

There are four 1Kx4-bit PROMs (HM-7643), whose maximum access time is 70 ns. The PROM contains the object code of the skeletonization process, as well as the interrupt service routines. The addresses of these two pairs of PROM range from F800 to FFFF.

One node of the pipe-line consists of two 64x4-bit First-In First-Out (FIFO) memory chips (C67401A) in parallel, whose internal frequency is 15 MHz. Each FIFO has:

- 1- four lines for data input (Dx) and four lines for data output (Ox), with $x = 0, 1, 2, 3$.
- 2- an Input Ready (IR) line: when asserted high, it informs the MPU that it is ready for data input;

- 3- a Shift In (SI) line: when asserted high, it requests from FIFO to receive data;
- 4- an Output Ready (OR) line: when asserted high, it informs the MPU that it is ready for data output;
- 5- a Shift Out (SO) line: when asserted high, it requests from the FIFO to send data;

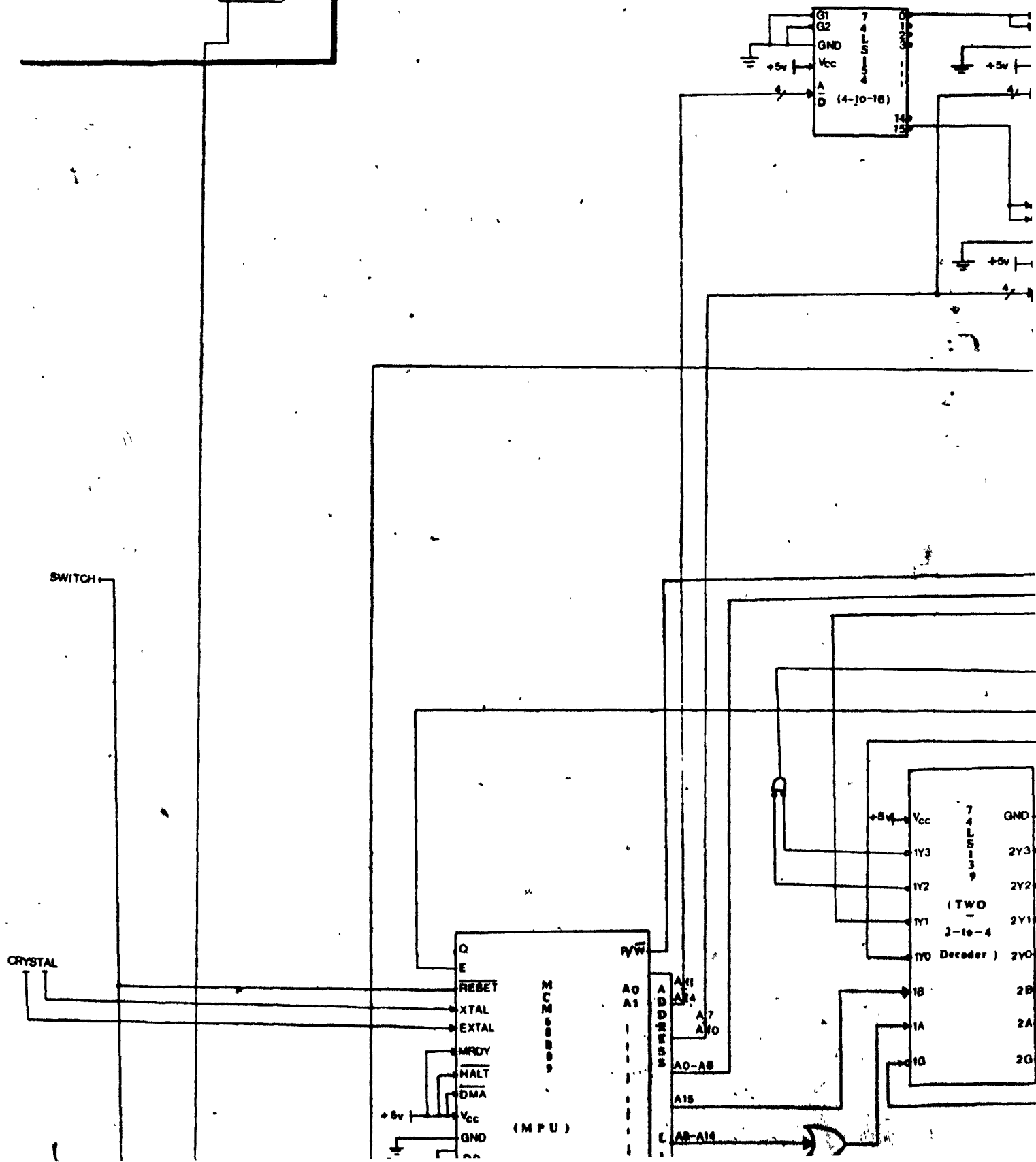
The NANDed IR lines of NODE_j , with an open-collector (represented by a '*') and a pull-up resistor, are connected to the maskable interrupt line $\overline{\text{FIRQ}}$ of the 6809 (vector interrupt at memory location FFF6 and FFF7). The 7401 chip is a set of such NAND gates with open-collector but without pull-up. Similarly, the OR lines of NODE_{j-1} are connected to the maskable interrupt line $\overline{\text{IRQ}}$ of the 6809 (vector interrupt at memory location FFF8 and FFF9). In Figure V.2, the accesses to SI and SO are memory mapped to addresses 0000 and 0001, respectively. The interface between the pipe-line and the Slave Processor has also a D-Flip-Flop (SN54LS74), memory mapped at address 0002, and a Tri-State Buffer Logic (TSBL). The purpose of the TSBL is to allow the pipe-line to be "disconnected" from the Slave Processor when requested (e.g., as in SPTA when Scan 2 has to be initiated while Scan 1 has not). The TSBL consists of (a) two tri-state buffers TS1, where output is disabled (high impedance) when the enable En is low (SN74LS126A); (b) two tri-state buffers TS2, where output is disabled when En is high (SN74LS125A); and (c) two pull-down resistors.

1 OF

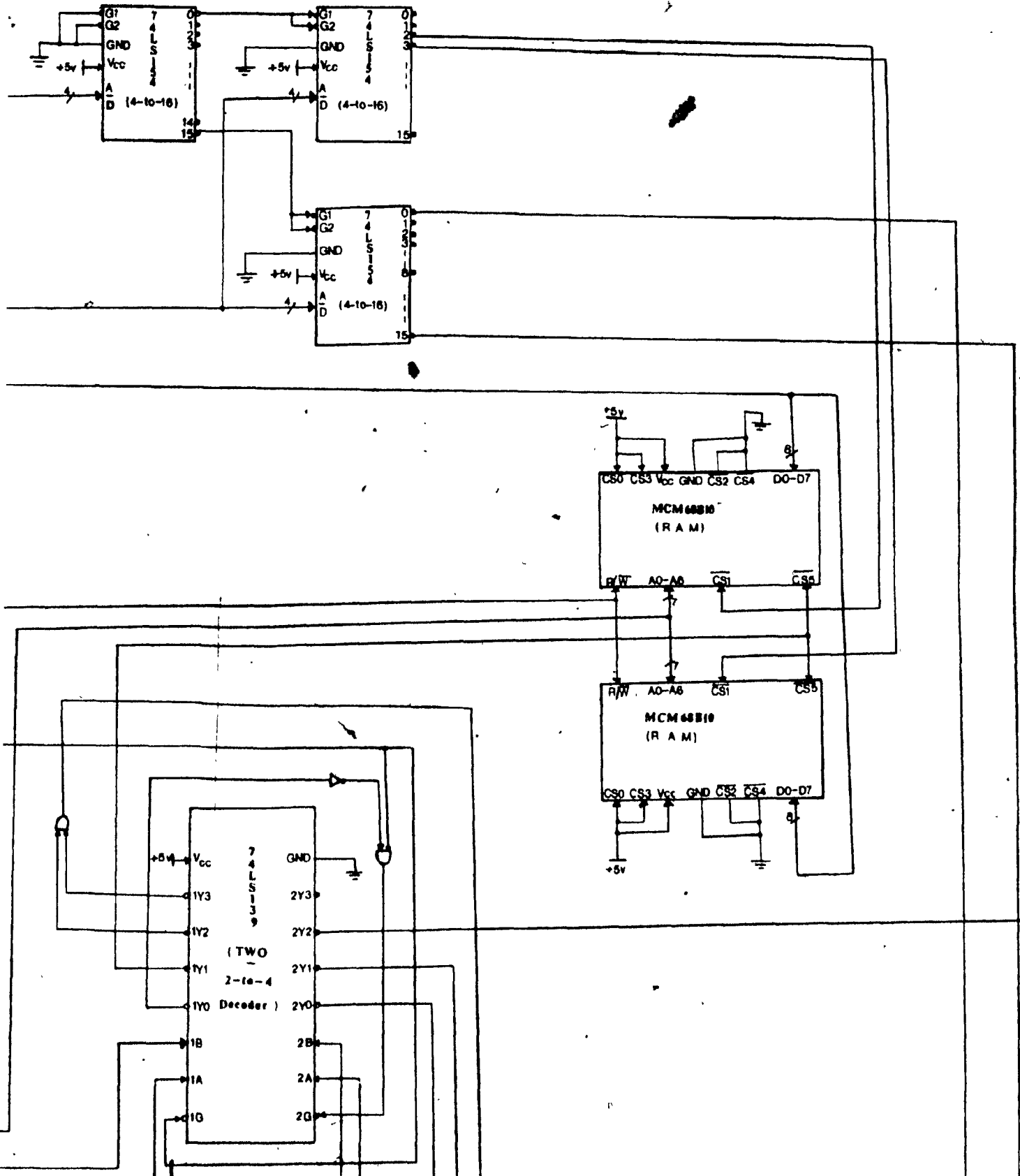
**MASTER PROCESSOR
UNIT**

74LS122
(MSV)

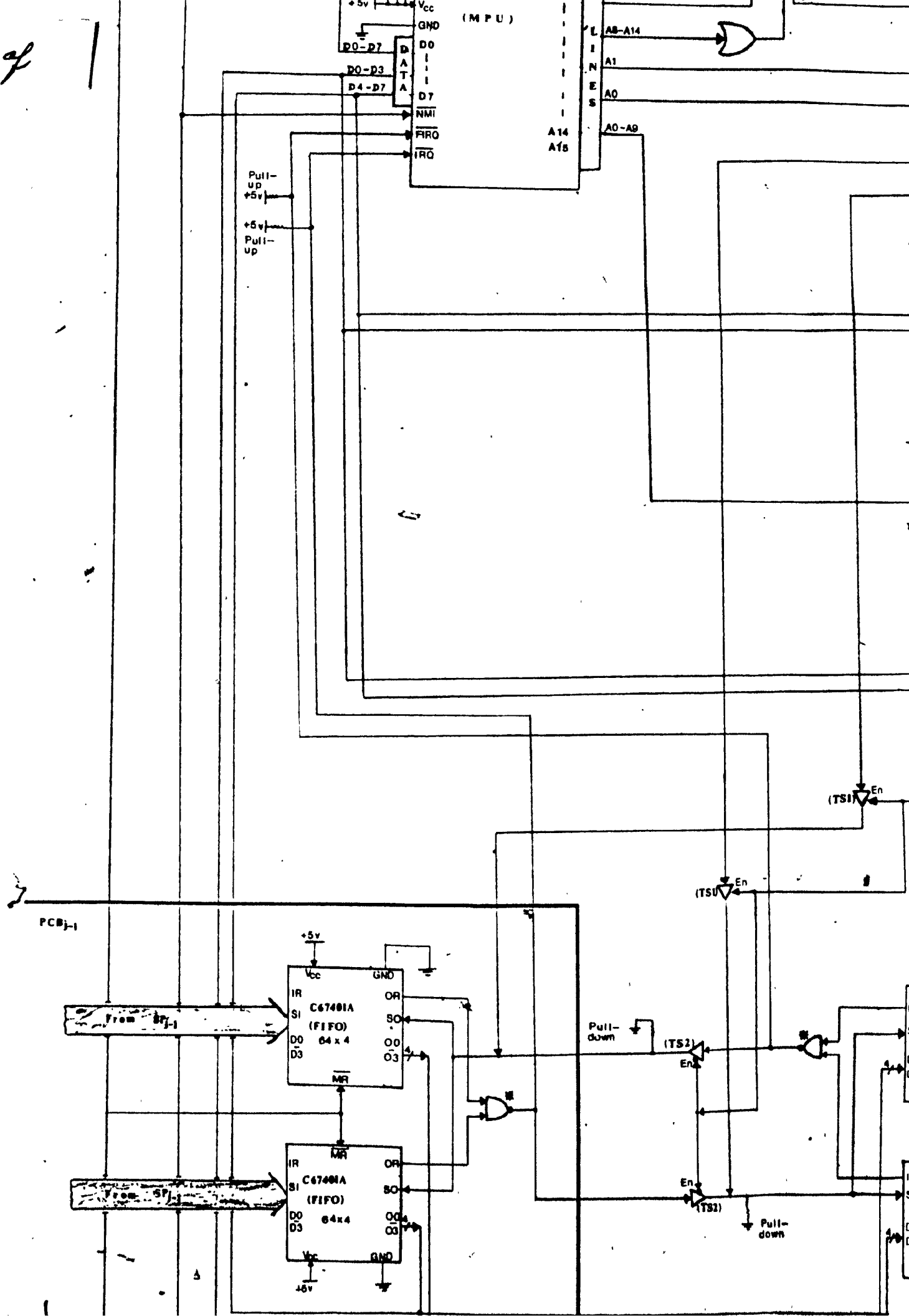
Q Q



24



1



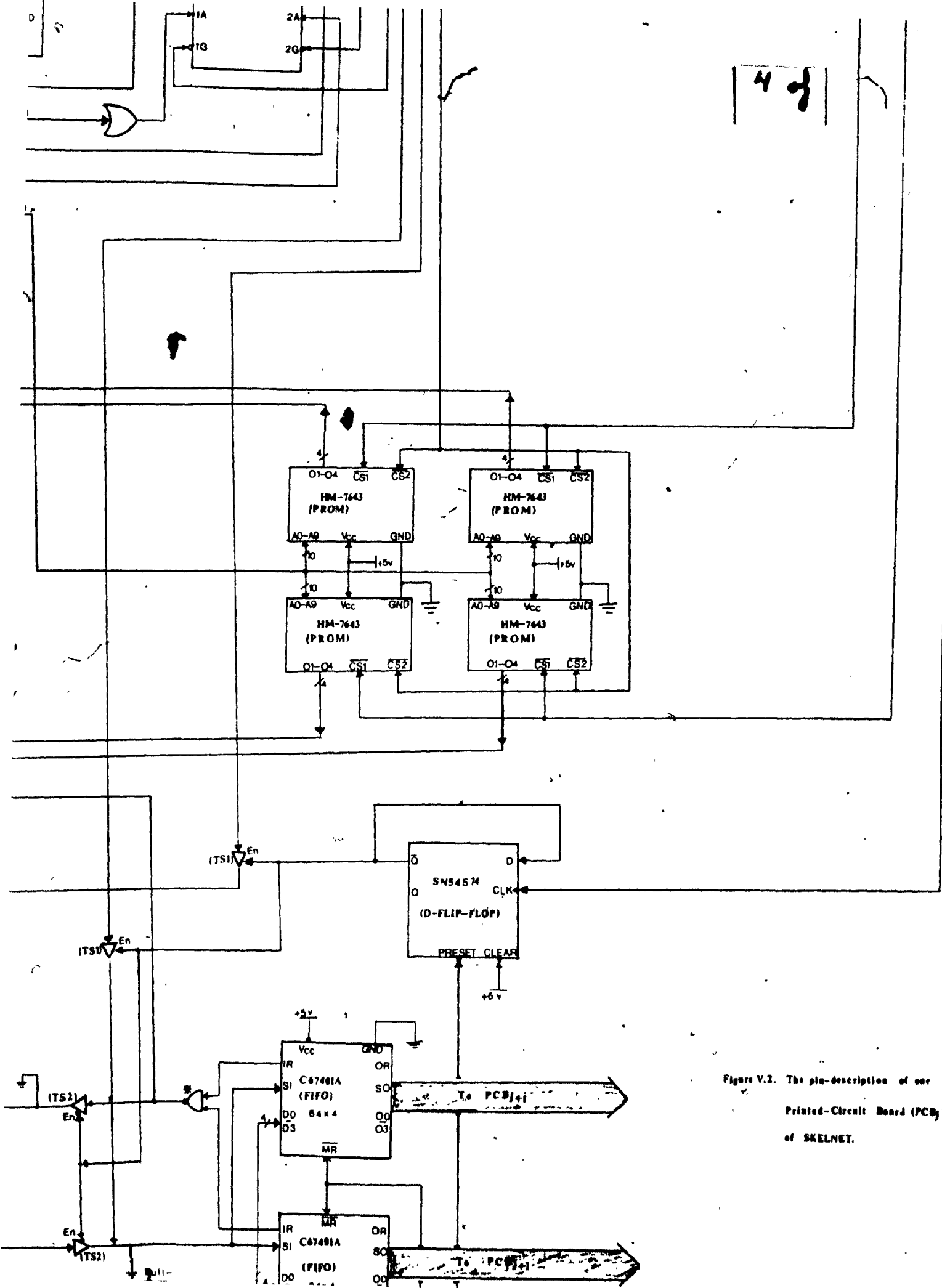
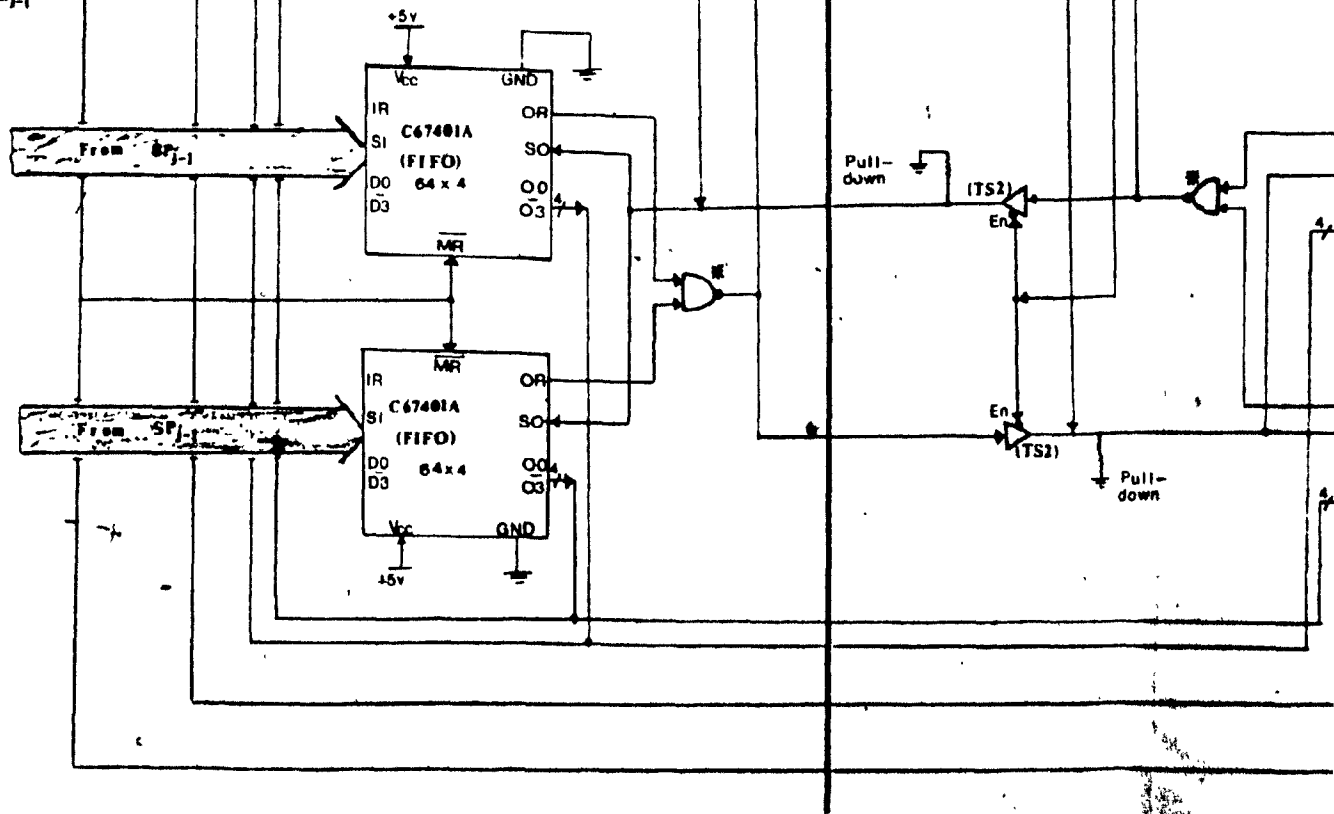


Figure V.2. The pin-description of one Printed-Circuit Board (PCB) of SKELNET.

PCB_{j-1} 

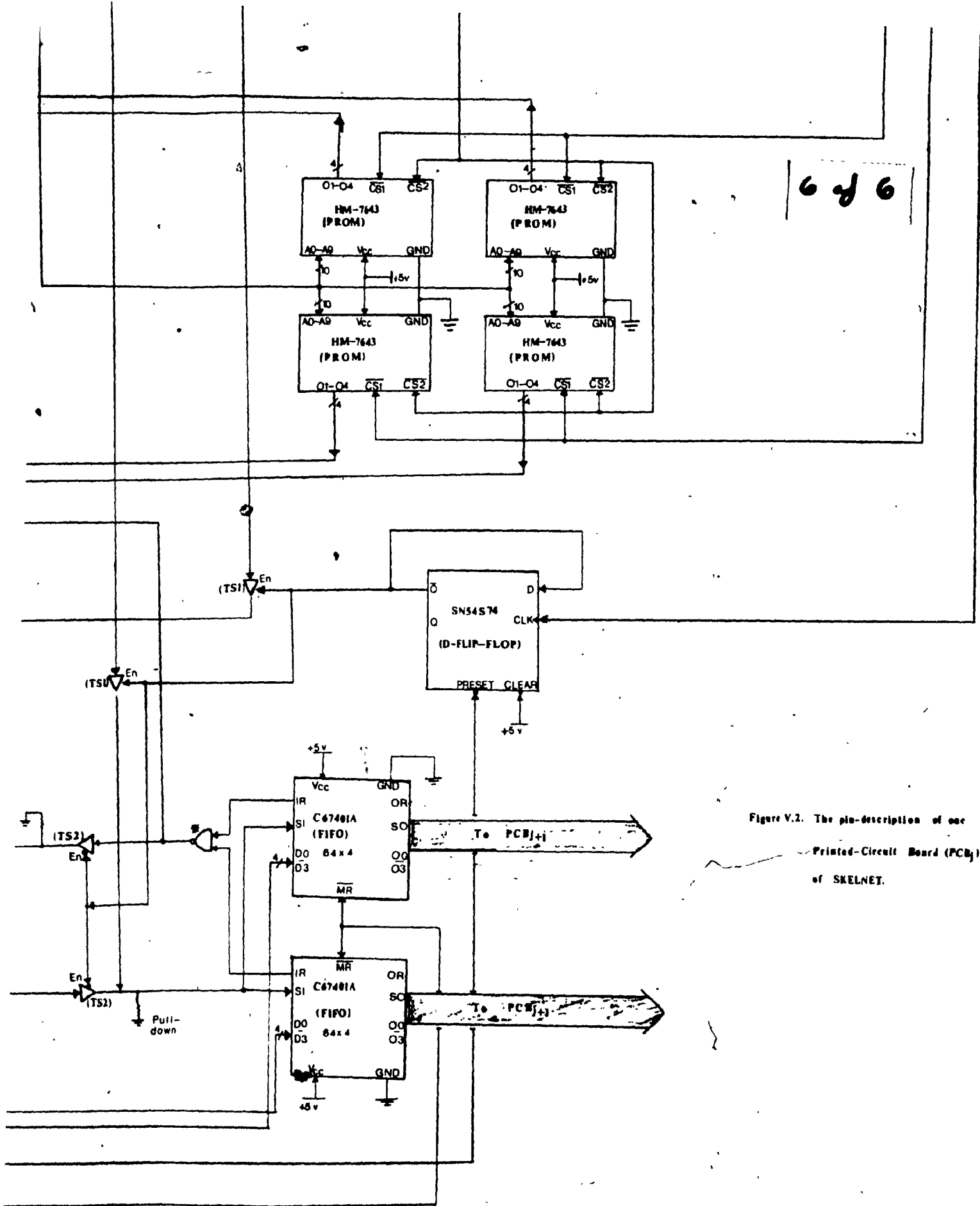


Figure V.2. The pin-description of one
Printed-Circuit Board (PCB)
of SKELNET.

V.6. Interaction between the processor units

After a master reset of the network by a common switch, the Slave Processors are ready for starting the skeletonization process. The \bar{Q} pin of the multivibrator of the MP is connected to the non-maskable interrupt line \overline{NMI} of each Slave Processor (vector interrupt at memory location FFFC and FFFD). The pulse width of the Q signal of the 6809 is 430ns. Since \overline{NMI} request is sampled on the falling edge of the signal Q of the 6809, and since \overline{NMI} is latched, one needs the width of the multivibrator pulse be long enough to be captured during the MPU's Q cycle; i.e., its width must be approximately 430ns. The same \bar{Q} pin of the multivibrator is also connected to the Preset pin of the D-Flip-Flop, asserted on the falling edge of the signal. Thus, the output Q of the D-Flip-Flop is set to 0, the output \bar{Q} being set to 1. In such a case, the enable line En of the two TS1 is high, and their outputs are not disabled. Simultaneously, the enable line En of the two TS2 is low, and their outputs are disabled. Thus, the pipe-line is connected to the SP.

During the skeletonization process of the SP over a row r of the input pattern, the two interrupt line \overline{IRQ} and \overline{FIRQ} are masked. When the SP is ready to receive data from $NODE_{j-1}$, the line \overline{IRQ} is unmasked. Upon an OR signal from $NODE_{j-1}$, the Slave Processor services the routine to receive

data from the node: the SP masks $\overline{\text{IRQ}}$, sends a SO signal to the node, reads in one byte of data (i.e., one point of the pattern) and stores it into RAM_j . Then SP unmask $\overline{\text{IRQ}}$, ready to receive another byte. Similarly, when the SP is ready to send data to NODE_j , the line $\overline{\text{FIRQ}}$ is unmasked. Upon an IR signal from NODE_j , the Slave Processor services the routine to send data to the node: the SP masks $\overline{\text{FIRQ}}$, sends a SI signal to the node and writes out one byte of data (i.e., one point of the pattern) from RAM_j . Then SP unmask $\overline{\text{FIRQ}}$, ready to send another byte.

Now, say a command-line sent by the MP requests the SP to be disconnected. Then, the MPU addresses line 0002, connected to the CLK pin (clock) of the D-Flip-Flop. Since the D-Flip-Flop is triggered on a falling edge of CLK, and since the output from the decoder is asserted low, then the D-Flip-Flop sets Q to 1. Thus, \overline{Q} becomes set to 0. In such a case, the enable line En of the two TS1 is low, and their outputs are disabled. Simultaneously, the enable line En of the two TS2 is low, and their outputs are no longer disabled. Thus, the pipe-line becomes disconnected from the SP. The data can thus flow directly from NODE_{j-1} to NODE_j . The use of the pull-down resistors becomes transparent here: their purpose is to avoid the creation of an "unknown" state on the pipe-line segment, when disconnecting from the SP or re-connecting to it.

Later, when a multivibrator pulse is sent by the MP, calling the SP for the processing of a new input pattern, the D-Flip-Flop is again preset, and the pipe-line is re-connected to the Slave Processor. The whole skeletonization process can thus be repeated.

V.7 Software simulation of SKELNET

In order to measure the speed-up that SKELNET would produce, and since we were unable to obtain the required equipment, we have simulated all the operations of the system by software. The simulation was based on the concepts used in PSIM, a discrete-event simulation package for Operating Systems, and available on the Cyber 835 at Concordia University, Montreal (Vaucher and Bratley, 1980). We are now going to show, in an algorithmic form, the various steps of the simulation. We are assuming that the MP has an input pattern ready to be skeletonized, and that all PCBs are reset.

The procedure MAIN simply calls for the two concurrent processes MASTER_OPS and SLAVES_OPS:

```

procedure MAIN;
begin
  MASTER_RESET;
  { Master Reset through the control-line }

  repeat
    COMMAND_LINE;
    { the Command-Line is sent by MP to all SPs:
      the first time with values ROWCOUNT and LAST,
      the subsequent times to inform the SPs whether they
      have to be disconnected from the pipe-line or not }

    cobegin
      MASTER_OPS; { MP's operations; shown below }
      SLAVES_OPS; { SP's operations; shown below }
    coend;

  until NO_MORE;
  { NO_MORE returns TRUE when the Scan-Flag of LAST
    is FALSE }

end; { MAIN }

```

```

process MASTER OPS;

var ROW, COLUMN : integer;

begin
    { Send all rows of the pattern to the chain }
    for all ROWs and COLUMNs of the pattern do
        begin
            repeat
                WAIT (IR, FIFOMP);
                SIGNAL (SI, FIFOMP);
                SEND (ROW, COLUMN);
            until one row is sent;
            QUEUE (ROW, ROWQ);
            { add ROW to the queue ROWQ of the rows sent
              to the chain }

            { On Interrupt Request from PCBLAST,
              call Service Routine }
            ON FIRQ do
                SERVICE_RECEPTION; { shown below }

        end;

        { All rows have been sent; now wait for all the
          processed rows to be received back }
        repeat
            SERVICE_RECEPTION
        until ROWQ queue is empty;

    end; {MASTER_OPS}

    {-----}

    procedure SERVICE_RECEPTION;

    var COLUMN : integer;

    begin
        repeat
            SIGNAL (SO, FIFOLAST);
            RECEIVE (ROWQ, COLUMN);
            { receive COLUMN of the first element ROWQ
              of the queue }
            WAIT (OR, FIFOLAST);
            until all COLUMNs are sent;
            DELETE (ROWQ, ROW);
            { remove ROW from the queue ROWQ }

        end; { SERVICE_RECEPTION }
    end;

```

```

process SLAVES_OPS;
var ROWSP, COLUMNSP : integer;
begin
  for all active Slave Processors SP of the chain do
    cobegin
      i := j; { initialize the pass number }
      repeat
        set ROWCOUNT to the number of rows to process;
        set SFLAG to FALSE;

        repeat
          { receive one row from preceding PCB of the chain }
          repeat
            WAIT (OR, FIFOj-1);
            SIGNAL (SO, FIFOj-1);
            RECEIVE (ROWSP, COLUMNSP);
            if a full ROWSP is received then
              ROWCOUNT := ROWCOUNT - 1;
          until RAM is full;

          SKELETONIZE (r, i);
          { apply pass-i of the skeletonization algorithm to
            row r of RAMj }

          { send one row r-1 to PCBj+1 }
          repeat
            WAIT (IR, FIFOj);
            SIGNAL (SI, FIFOj);
            SEND (r-1, COLUMNSP);
          until row r-1 is sent;

          RELABEL (rows of RAM);
          i := i + LAST;

        until ROWCOUNT = 0;

      until SFLAG = FALSE;
    coend;
  end; { SLAVE_OPS }

```

An Executive controls all the processes. It measures the time by calculating the number of units of time it took for SKELNET to accomplish the skeletonization of one pattern. One unit of time is defined to be one step of each process.

For instance, each procedure WAIT in SLAVES_OPS corresponds to one step. However, the procedure SKELETONIZE is itself divided into a sequence of steps: test the value of a point p, test the value of each of its 8-neighbours, modify the value of p if required. Thus, for every unit of time one step of the entire process is achieved. For ease of programming, hardware and software steps were given the same measure of time.

The simulation of SKELNET was experimentally tested using the Safe-Point Thinning Algorithm described in Chapter III, with its Single Integer Labelling Technique. The data base was the same as the one described in Chapter IV. The results obtained are shown in Table V.1. We can see from this table that with one pair of PCBs, the average time required to process one pattern is slightly higher than with no PCBs (i.e., when SPTA is run sequentially). The reason for that is the overhead due to the transmission of data through the chain. We notice that the speed up is optimal when 4 pairs of PCBs are used (speed up of 47%). Indeed, we saw in Chapter IV that the average number of passes required to process one pattern in SPTA is 3.31. Thus, we can say that optimal number (OPT) of SPs used should be:

$$\text{OPT} = \lceil \text{PASS} \rceil * \text{SCANS}$$

where $\lceil \text{PASS} \rceil$ stands for the ceiling value of the average number of passes required to skeletonize the patterns, and SCANS for the number of scans the algorithm requires per

pass, including the scan for deleting the flagged points from the pattern. For example, for SPTA, $\overline{\text{PASS}} = 4$ and $\text{SCANS} = 2$; that is, $\text{OPT} = 8$. Thus, one needs to know the average thickness of the input pattern prior to deciding on the number of PCBs to be used. However, we concede that the time measurement is not accurate. Indeed, having no other choice, we assumed in the simulation of SKELNET that a hardware step takes the same amount of time as a software step. In a real implementation of the network, the time for a hardware step would be much less. Thus, the speed up should be greater.

Number of pairs of PCBs used	Average number of units of time required to process one pattern
NONE	5520 units
1	5756 units
2	4141 units
3	3416 units
4	2916 units
6	3113 units

TABLE V.1.

This table shows the average number of time-units required to process one pattern, in function of the number of pairs of PCBs used. The algorithm tested was SPTA with its Single Integer Labelling Technique (SILT). There were 648 patterns processed by each simulation.

V.7. Expandability of SKELNET

Whether a skeletonization algorithm needs larger fundamental windows for testing dark points for deletion (Stefanelli and Rosenfeld, 1971; Stentiford and Mortimer, 1983), SKELNET would need to look larger areas than the ones defined by Rule 1. This can be easily achieved by expanding the RAM of each SP. Indeed, there are almost 32K RAM addresses possible.

Moreover, should we need SKELNET to perform other tasks than skeletonization e.g., smoothing (Unger, 1959), or feature extractions and classification of the patterns (Gudesen, 1976; Ikeda, Yamamura, Mitamura, Fujiwara, Tominaga and Kiyono, 1981), then the programs of each SP's PROM can be expanded. Indeed, there are 32K PROM addresses possible, thus capable of storing relatively large machine-codes.

Furthermore, should we need adding-up new interfaces to each SP, enough address lines are left free for that purpose.

The network SKELNET can also be examined to program to handle gray-scale patterns (Salari and Siy, 1982).

Thus, our network SKELNET is inexpensive (at today's

rate -April 1984-, one PCB would cost approximately Can\$ 120), gives reasonably good speed-ups, and can be expanded to perform other tasks than skeletonization of binary patterns.

Chapter VI

Concluding Remarks

VI.1. General Conclusion

In this thesis, we gave a historical review of published literature on skeletonization algorithms for binary patterns (Chapter II). We have investigated 16 such algorithms and overviewed 10 other algorithms. We have also proposed, in Chapter III, another skeletonization algorithm, called the Safe-Point Thinning Algorithm (or SPTA). In this algorithm, we have demonstrated that a single test, called the safe-point test, can replace the end-point, break-point and excessive erosion tests. We have experimentally shown in Chapter IV the performance of the SPTA as compared to the 16 algorithms reviewed in Chapter II. The SPTA does not break the connectedness of the pattern, does not cause excessive erosion and does not alter the shape property of the original pattern. The proposed algorithm is also faster than all other 16 algorithms. Moreover, the SPTA has two labelling techniques: the Single Integer Labelling Technique (or SILT), and the Four Integers Labelling Technique (or FILT). The labelling techniques provide the algorithm with the reconstruction ability; that is, a pattern can be reconstructed, similar to the original pattern from which the skeleton was derived. The quality of reconstruction depends on the labelling technique used. In Chapter IV, we have also reviewed one smoothing technique and we have

proposed a modified version of it. Finally, in Chapter V, we have proposed a multiprocessing network for skeletonization algorithms, called SKELNET. We have described both the functions and the hardware of the network. We have also given the algorithmic description of a discrete-event simulator which was implemented to evaluate the performance of SKELNET. The simulation of SKELNET was tested using the proposed SPTA with its Single Integer Labelling Technique. We have also outlined the expandability of SKELNET.

VI.2. Future Work

The Safe-Point Thinning Algorithm can be generalized to multi-gray level patterns, by applying, for example, a distance function similar to the one used by Salari and Siy (1982).

Moreover, Hilditch (1983) has shown that it is possible to convert sequential algorithms into parallel ones, provided an appropriate set of fundamental windows be defined. We believe that such a set could be defined for SPTA, without however losing the safe-point concept. More work can be done in this direction, thus converting SPTA from a sequential to a parallel algorithm.

Concerning the proposed multiprocessor network SKELNET,

further simulations of algorithms other than SPTA can be performed, in order to evaluate the feasibility and definite advantages of such a system. Ideally, with adequate equipment, SKELNET can be implemented in hardware. We have already discussed in Chapter V the expandability of SKELNET. We believe that the network can be further refined such that as soon as any one Slave Processor finds out that no more passes are required over a pattern, this pattern is no longer cascaded through the entire chain.

We hope that the work described above will not remain ideas on paper. It will require time and appropriate equipment to further the work that has been described in this thesis.

References

Arcelli C., 1979, "A Condition For Digital Points Removal",
Signal Processing 1, pp 283-285.

Arcelli C. and di Baja G.S., 1981, "A Thinning Algorithm
Based On Prominence Detection", Pattern Recognition, Vol
13, No 3, pp 225-235.

Bel-Lan A. and Montoto L., 1981, "A Thinning Transform for
Digital Images", Signal Processing 3, pp 37-47.

Beun M., 1973, "A Flexible Method For Automatic Reading Of
Handwritten Numerals", Philips Technical Review, Vol 33,
No 4, pp 89-101 and pp 130-137.

Budak A., 1974, Passive And Active Network Analysis
Synthesis, Houghton-Mifflin, Boston.

Chaudhuri B.B., 1978, "A Simple Method Of Thinning", Journal
Instn Electronics and Telecom. Engrs, Vol 24, No 6, pp
264-265.

Davies E.R. and Plummer P.N., 1981, "Thinning Algorithms: A
Critique And A New Methodology", Pattern Recognition, Vol
14, Nos 1-6, pp 53-63.

Deutsch E.S., 1972, "Thinning Algorithms On Rectangular, Hexagonal and Triangular Arrays", Communications ACM, 15, pp 827-837.

Doyle W., 1960, "Recognition Of Sloppy Hand-Printed Characters", in Proc of West Joint Computer Conference, Vol 17, May 1960, pp 133-142.

Duff M.J.B., 1976, "CLIP 4: A Large Scale Integrated Circuit Array Parallel Processor", in International Joint Conference on Pattern Recognition, Colorado, California, 1976.

Duff M.J.B., 1978, "Review Of The CLIP Image Processing System", Proc National Computer Conference, 1978, pp 1055-1060.

Evans D.J., 1982, Parallel Processing Systems, Cambridge University Press, Cambridge, U.K., ed. Evans D.J.

Fletcher W.I., 1980, An Engineering Approach To Digital Design, Prentice-Hall Inc., Englewood Cliffs.

Gonzalez R.C. and Wintz P., 1977, Digital Image Processing, Addison-Wesley, Reading, Massachusetts.

Gudesen A., 1976, "A Quantitative Analysis Of Preprocessing

Techniques For The Recognition Of Hand-Printed Characters", Pattern Recognition, Vol 8, pp 219-227.

Hanaki S. and Temma T., 1982, "Template Controlled Image Processor (TIP) Project", in Multicomputers and Image Processing, by Preston K. Jr and Uhr L., Academic Press, pp 343-352.

Hilditch C.J., 1969, "Linear Skeletons From Square Cupboards", Machine Intelligence, Vol 4, pp 403-420.

Hilditch C.J., 1983, "Comparison Of Thinning Algorithms On A Parallel Processor", Image and Vision Computing, Vol 1, No 3, pp 115-132.

Ikeda K., Yamamura T., Mitamura Y., Fujiwara S., Tominaga Y., Kiyono T., 1981, "On-Line Recognition Of Hand-Written Characters Utilizing Positional And Stroke Vector Sequences", Pattern Recognition, Vol 13, No 3, pp 191-206.

Kohler R.J. and Howell H.K., 1963, "Photographic Image Enhancement By Superposition Of Multiple Images", Phot. Sci. Eng., Vol 7, No 4, pp 241-245.

Ma H., 1983, "A Comparative Study of Thinning Algorithms", Major Report, Department of Computer Science, Concordia

University, Montreal.

Moayer B. and Fu K.S., 1976, "A Tree System Approach For Fingerprint Pattern Recognition", IEEE Trans Comput., C-25, pp 262-275.

Motorola Microprocessors Data Manual, Motorola Inc., 1981.

Naccache N.J. and Shinghal R., 1984a, "An Investigation Into The Skeletonization Approach Of Hilditch", Pattern Recognition, in press.

Naccache N.J. and Shinghal R., 1984b, "SPTA: A Proposed Algorithm For Thinning Binary Patterns", IEEE Trans Systems, Man, Cybernetics, in press.

Pavlidis T., 1981, "A Flexible Parallel Thinning Algorithm", in IEEE Proc Conference on Pattern Recognition and Image Processing, August 1981, Dallas, pp 162-167.

Pavlidis T., 1982a, Algorithms For Graphics And Image Processing, Computer Science Press Inc., Rockville MD, pp 195-214.

Pavlidis T., 1982b, "An Asynchronous Thinning Algorithm", Computer Graphics and Image Processing, Vol 20, pp 133-157.

Rao C.V.K., Prasada B. and Sarma K.R., 1974, "An Automatic Fingerprint Classification System", in Proc 2nd International Joint Conference on Pattern Recognition, Copenhagen, Denmark, 1974, pp 180-184.

Rosenfeld A. and Pfaltz J.L., 1966, "Sequential Operations In Digital Picture Imaging", Journal of ACM, Vol 13, No 4, pp 471-494.

Rosenfeld A., 1975, "A Characterization Of Parallel Thinning Algorithms", Information and Control, 29, pp 286-291.

Rutovitz D., 1966, "Pattern Recognition", Journal Royal Statists. Soc., Vol 129, Series A, pp 504-530.

Salari E. and Siy P., 1982, "A Skeletonization Algorithm For Gray-Scale Pictures", in Proc. of the IEEE International Conf. on Cybernetics and Society, Seattle, Washington, October 1982, pp 459-462.

Saraga P. and Woollons D.J., 1968, "The Design Of Operators For Pattern Processing", in IEE NPL Conf. on Pattern Recognition, Teddington, 1968, pp 106-116.

Stefanelli R. and Rosenfeld A., 1971, "Some Parallel Thinning Algorithms For Digital Pictures", Journal of ACM, Vol 18, No 22, pp 255-264.

Stentiford F.W.M. and Mortimer R.G., 1983, "Some New Heuristics For Thinning Handprinted Characters For OCR", IEEE Trans Systems, Man, Cybernetics, Vol 13, No 1, pp 81-84.

Sternberg S.R., 1982, "Pipeline Architectures For Image Processing", in Multicomputers and Image Processing, by Preston K. Jr and Uhr L., Academic Press, pp 291-305.

Tamura H., 1978, "A Comparison Of Line Thinning Algorithms From Digital Geometry Viewpoint", in Proc 4th International Joint Conference on Pattern Recognition, Kyoto, Japan, 1978, pp 715-719.

The TTL Data Book for Design Engineers, Texas Instruments Inc., 1981.

Unger S.H., 1959, "Pattern Detection And Recognition", Proceedings of IRE, pp 1737-1752.

Vaucher J. and Bratley P., 1980, "La Programmation De Simulation", Departement d'Informatique et de Recherche Operationnelle, Univeriste de Montreal, Montreal, Chapter 6.

Weinberg L., 1962, Network Analysis. And Synthesis,

McGraw-Hill, New-York.

Yokoi S., 1973, "Topological Properties In Digitized Binary Pictures", Systems, Computers, Controls, Vol 4, No 6, pp 32-39.

Yokoi S., 1975, An Analysis Of Topological Features At Digitized Binary Pictures Using Local Features, Computer Graphics and Image Processing, Vol 4, pp 63-73.

Zhang T.Y. and Suen C.Y., 1984, "A Fast Parallel Algorithm For Thinning Digital Patterns", Communications of the ACM, Vol 27, No 3, pp 236-239.