Pattern Recognition Pergamon Press 1973. Vol. 5, pp. 229–247. Printed in Great Britain

A Cellular Logic Array for Image Processing

M. J. B. DUFF, D. M. WATSON, T. J. FOUNTAIN and G. K. SHAW

Department of Physics,
University College, London, England

(*Received* 5 *September* 1972)

Abstract—A cellular logic image processor employing 192 cells in a 16 by 12 hexagonal array is described. The processor has been constructed and its performance assessed. The various classes of functions which can be implemented in the cellular array are discussed and sample programs explained in detail.

Cellular array Pattern recognition Image processing Parallel processing Binary images

I. INTRODUCTION

A. *Cellular logic arrays*

MANY authors have proposed parallel processing algorithms for image processing and pattern recognition, but the state-of-the-art in circuit component technology has only recently approached the point where the construction of large arrays of interconnected cellular logic elements is an economic proposition. It would seem appropriate to process two-dimensional data sets by using two-dimensional arrays of logic elements, thus matching the form of the processor to the form of the data, so that there is an incentive to explore the action of such arrays. However, this action can be simulated on conventional serial computers and the potential power of parallel systems can be assessed by studying the simulated performance. Nevertheless, precise simulation of large, complex arrays taking into account details of hardware properties, is by no means a simple task and tends to involve lengthy programs which are both expensive to develop and expensive to run.

The research program reported in this paper is an attempt to provide general purpose programmable cellular logic arrays which can be used for image processing studies and which will facilitate the design of specialized arrays for particular image processing applications. It is suggested that the use of such arrays will usually be more satisfactory than computer simulation, both from the point of view of matching technique to problem and from consideration of processing times and efficiency.

B. *Brief literature survey*

As has already been implied, most studies of proposed cellular logic systems for image processing have been purely theoretical, often employing a conventional serial computer for simulation of the parallel algorithms. In 1958 UNGER[1] proposed and, later, simulated[2] a square array with 36 by 36 elements and with nine memory registers in each cell. Possible instructions to each cell included left, right, up and down shift instructions, logical addition and multiplication between the cell accumulator, neighbouring cell accumulators, and the cell memories, transfers to memories, input/output, and a special "link" instruction for finding connected sets. Typical character recognition programs required of the order

300–500 instructions per recognition. Some of the features of Unger's proposals were incorporated in the SOLOMON computer built at the Westinghouse Electric Corporation, Baltimore, by GREGORY and MCREYNOLDS.[3] Many subsequent researchers have acknowledged inspiration from Unger's proposals, although others have sought rather to simulate parts of biological visual systems and have drawn their inspiration from neurophysiological work such as that carried out by HUBEL and WIESEL.[4] Simulated neural nets, and nets of neuron-like elements, have been investigated by many workers including ALEKSANDER[5] and BROWN, HALL and LAL.[6] An extension of these ideas into optical processing was proposed by HAWKINS and MUNSEY.[7] Theoretical studies of the properties of cellular arrays[8–14] have indicated their power and versatility in image processing for operations such as thinning and skeletonizing, image enhancement and feature extraction. MINNICK[15] has reviewed the use of cellular arrays for these and other tasks. But construction of hardware operating in a parallel manner has not often been attempted. MCCORMICK[16] constructed the ILLIAC III computer and LEVIALDI[17] has built parallel systems for shrinking and counting images of objects and for detection of certain image properties such as closed loops. A special purpose processor for use in the analysis of bubble chamber photographs was constructed by DUFF.[18]

C. *Tessellation and connectivity*

If an image is to be processed in a cellular logic array, it must first be divided into elements, or cells, each cell taking a value related to the average image density in the region of image represented in the cell. Very often, the cell will be assigned the logic value (1) or (0) depending on whether or not the average density exceeds a selected threshold level.

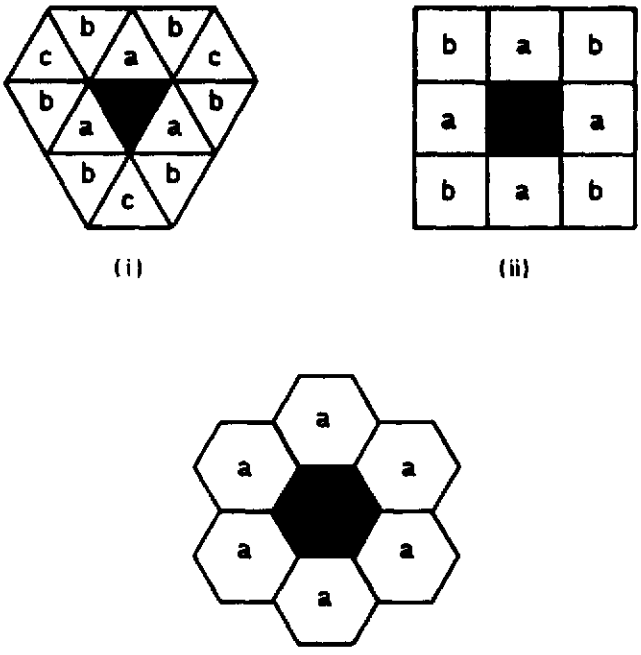


FIG. 1.

It should be noted at this point that the term "cell" is conventionally used to refer both to the small regions of the original image and to the cellular logic processors assigned to the particular regions of the image. It is usually obvious from the context which interpretation is appropriate each time the term is used.

Only three regular figures will close pack to completely cover an area. They are the equilateral triangle, the square and the hexagon (see Fig. 1).

Before undertaking hardware construction of any parallel processing device, it is necessary to determine the type of tessellation to be used, as this determines the type and degree of interconnection necessary. The detailed arguments concerning the relative merits of the possible types of cell are given in an appendix to this paper, but one of the primary advantages of the hexagonal cell can be seen with reference to Fig. 1. In the cases of triangular and square tessellation, (i) and (ii), neighbours of a given cell are of various types, viz. in some cases (cell type (a)) they have a common edge, and in others (cell types (b) and (c)) they share a common corner. This is clearly a less symmetrical and more complicated situation than that of (iii), hexagonal tessellation, where all cells share a common edge with the primary cell. It will be shown later (in the appendix) that hexagonal cells are also more efficient in terms of pattern connectivity, and, for these reasons were chosen for the initial pattern processing hardware system, Cellular Logic Image Processor, CLIP 2.

II. CLIP 2 CONSTRUCTION

A. *System outline*

The overall configuration of the CLIP 2 processing device is shown in Fig. 2. The operation may be described as follows. The input pattern is scanned by a flying light spot, digitized, and the digitized representation transmitted serially to the input memory plane and stored. It is then transmitted in parallel round a loop consisting of:

Input Memory Plane M_{in}
Processing Plane
Switch Matrix S_3
Either Memory Plane M_1 or Memory Plane M_{out}
Switch Matrix S_1
Either Input Memory Plane M_{in} or Processing Plane.

At the appropriate point in the loop the process can be halted and the processed pattern transmitted serially from M_{out} to the output display.

All parts of the system are controlled from the central block of control logic.

B. *The cellular array*

The cellular logic array for CLIP 2 consists of a 16 by 12 hexagonal array of 192 symmetric boolean operators. Each cell has two inputs: A_0 being the value of the input pattern at the cell and A_n which is the output from a NAND gate. The inputs to the NAND gate are the A'_1 outputs from the six neighbouring cells in the array. These two binary inputs are transformed by the cell into two independent binary outputs: A_1 and A'_1. The A_1 output is regarded as the processed pattern and the A'_1 output fans out into the interconnections with the six neighbouring cells. Figure 3 shows the schematic layout for the interconnection outputs from one cell (a seventh input to each gate is shown here and will be discussed later).

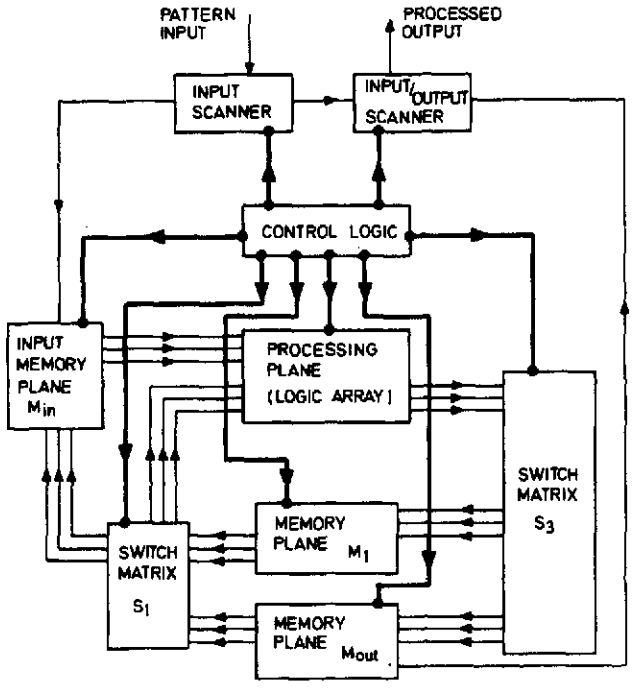


FIG. 2. Layout and routing of CLIP 2.

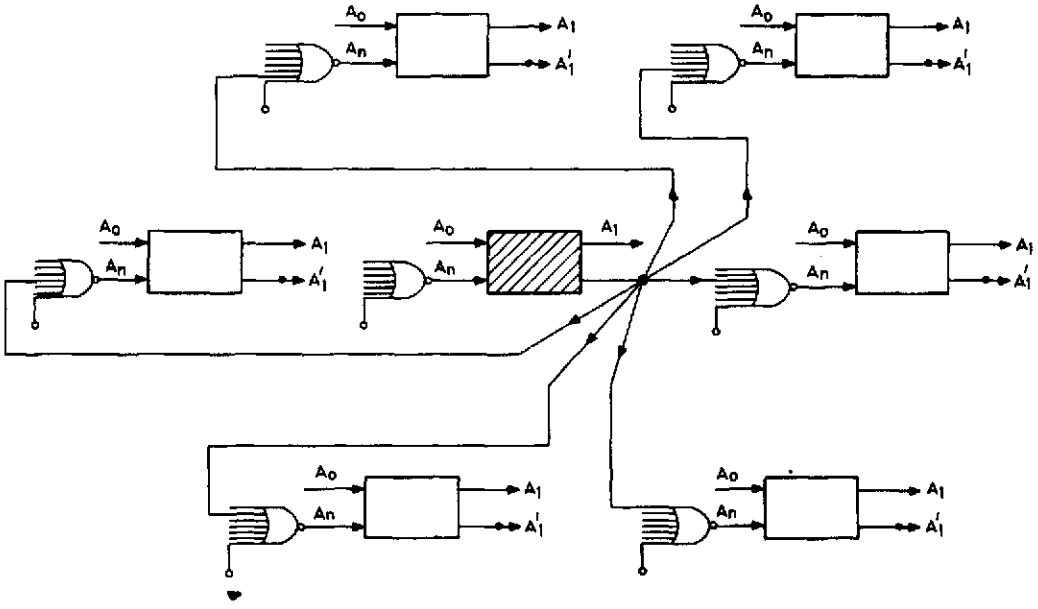


FIG. 3. CLIP 2 interconnection scheme.

The internal logical design of the CLIP 2 cell is shown in Fig. 4.
The boolean expressions for the two cell outputs A_1 and A'_1 are:

$$A_1 = \overline{(p \cdot A_0 \cdot A_n)} \cdot \overline{(q \cdot \bar{A}_0 \cdot A_n)} \cdot \overline{(r \cdot A_0 \cdot \bar{A}_n)} \cdot \overline{(s \cdot \bar{A}_0 \cdot \bar{A}_n)}$$

$$A'_1 = \overline{(l \cdot A_0 \cdot A_n)} \cdot \overline{(m \cdot \bar{A}_0 \cdot A_n)} \cdot \overline{(n \cdot A_0 \cdot \bar{A}_n)} \cdot \overline{(o \cdot \bar{A}_0 \cdot \bar{A}_n)}$$

By selecting appropriate binary values for the eight control lines l to s, A_1 and A'_1 outputs can be chosen for each of the four possible input states of the pair of inputs A_0 and A_n.

Availability of integrated circuit gates has resulted in a cell implementation which includes several inversions. Figure 5 is an equivalent logic circuit of the system which has been rearranged so as to simplify the description of the cellular logic. The only inversion outside the boolean cell (shown framed with double lines) applies to transfers between the output and input memories, M_{out} and M_{in}.

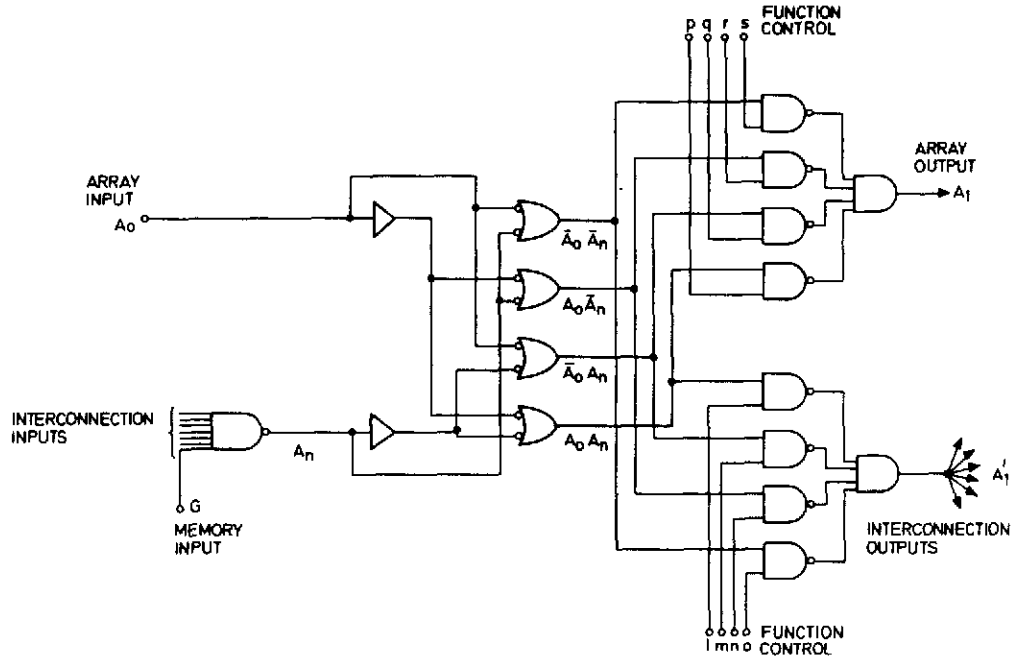


FIG. 4. Circuit diagram of one CLIP 2 processor cell.

The equivalent boolean expressions for the redefined cell are:

$$A_1 = \overline{(p \cdot \bar{A}_0 \cdot A_n)} \cdot \overline{(q \cdot A_0 \cdot A_n)} \cdot \overline{(r \cdot \bar{A}_0 \cdot \bar{A}_n)} \cdot \overline{(s \cdot A_0 \cdot \bar{A}_n)}$$

$$A'_1 = (l \cdot \bar{A}_0 \cdot A_n) + (m \cdot A_0 \cdot A_n) + (n \cdot \bar{A}_0 \cdot \bar{A}_n) + (o \cdot A_0 \cdot \bar{A}_n).$$

For the remainder of this paper, all descriptions refer to the equivalent (simplified) circuit. Nevertheless, the equivalent circuit behaviour is identical to that of the original circuit which appears in the hardware. Each cell (excluding the memories) is constructed using five TTL integrated circuit chips.

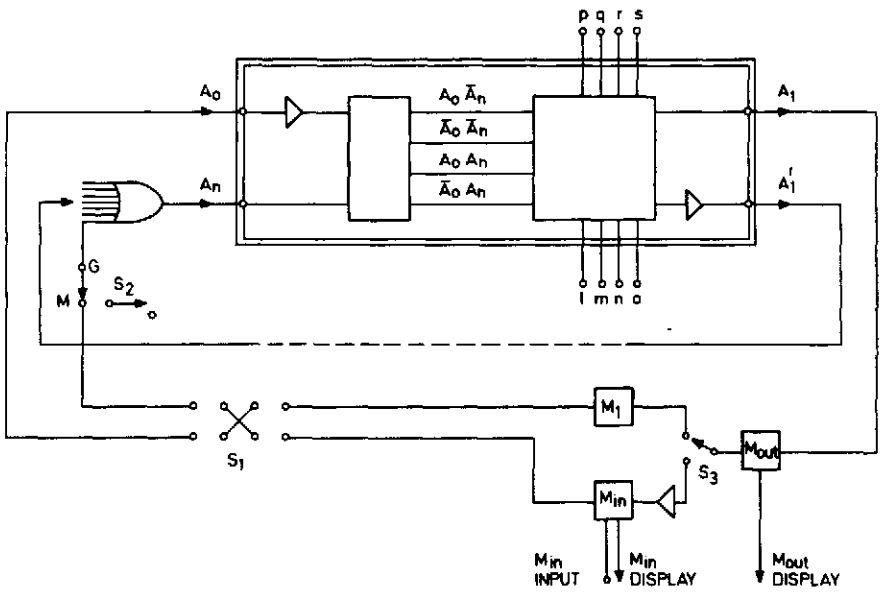


FIG. 5. Simplified logic diagram.

C. *The complete system*

1. *Pattern input.* The input pattern A_0 is stored in a 192 bit shift register M_{in} as a sequence of logic (1) and logic (0) states (representing black and white parts respectively of the original pattern). The shift register is loaded serially and can be cycled to provide a serial read out. In addition, each bit can be accessed in parallel so that the entire pattern is presented to the cellular array by clocking the register contents out onto 192 individual leads, one going to each cell in the array (see Fig. 2). A flying spot scanner is used to load the shift register with a thresholded image. The stored pattern can be modified by means of a simply constructed light pen operating on a CRT display obtained by cycling the shift register. The pen can be used in a similar manner to construct and load patterns *ab initio*. The CRT display is a dot display in hexagonal array format, the contents of M_{in} being used to modulate the spot brightness, so that a (1) appears as a bright spot and a (0) as a faint spot. The display uses one beam of a double beam oscilloscope.

2. *Pattern output.* The output pattern A_1 appears on the 192 output leads from the array and is stored in a parallel input shift register M_{out} on receiving a suitably timed clock pulse. The contents of the register can be displayed serially using a second beam of the double beam oscilloscope which provides the M_{in} display. Thus the input and output patterns are displayed one above the other on the same CRT face.

3. *Pattern storage.* Since both M_{in} and M_{out} are used during pattern processing, neither can provide storage for a pattern whilst another pattern is being processed. A third 192 bit memory M_1 is used for this purpose. Transfer from M_{out} to M_1 and from M_{out} to M_{in} are parallel operations.

4. *Pattern processing.* The CLIP 2 system is controlled by means of 12 bit word instructions. These are of two types: LOAD instructions are used to cycle the input and output

memories, for loading and display of their contents, and for transferring the contents of M_{out} into either or both of M_{in} and M_1; PROCESS instructions set the values of the control lines l to s, the position of the routing switches S_1 (so that either the contents of M_1 or of M_{in} can appear at the array as A_0), the position of the gate switches S_2 (which allow the contents of either M_1 or M_{in} to enter the interconnection gates along with the six interconnection leads from the neighbouring cells), and set up (1) or (0) on the interconnection leads which extend outside the 16 by 12 cell array.

Up to 32 instruction words can be stored in a 12 bit wide, 32 bit long shift register, so that sequences of instructions can be executed by CLIP 2. The control circuitry is responsible for supplying suitable clock pulses to operate the system. The instruction store is loaded either from twelve push buttons or from punched tape and can be read out onto tape in order to store programs which may be required again.

III. CLIP 2 OPERATION

A. *Programming CLIP 2*

Since it is possible to connect the outputs from a memory storing a pattern into the interconnection gates by means of the leads G (see Figs. 4 and 5), four types of operation can be executed.

These are (1) Pattern processing in which G is set to (0), (2) Pattern comparison in which G is set to M and the control lines l to o set A'_1 at zero, (3) Labelled pattern processing which employs active interconnections as well as G being set to M, and (4) Instruction programs, with sequences of any of the three types of individual instruction.

1. *Pattern processing.* Considering the first class of operations, it is required that each cell shall produce a predetermined (by means of the control lines l to s) pair of binary outputs A_1 and A'_1 for each of the four possible pairs of binary inputs A_0 and A_n.

For example, for a particular function the cell might be required to implement the truth table in Table 1. This table indicates that white cells ($A_0 = (0)$) output a (1) into the interconnections with neighbours and that only black cells ($A_0 = (1)$) receiving a (1) input from

TABLE 1

A_0	A_n	A'_1	A_1
0	0	1	0
0	1	1	0
1	0	0	0
1	1	0	1

a neighbour ($A_n = (1)$) appear as (1) in the output pattern A_1. Thus the A_1 output is (1) for all black cells which have at least one white neighbour.

The cell is designed so as to allow all mappings between the binary outputs and inputs; the columns A'_1 and A_1 can assume any combinations of (1) and (0) values. This implies[19] the possibility of 256 different truth tables, since the number of these is

$$\theta = (N)^M$$

where M is the number of distinct input states and N the number of distinct output states, both equalling 4 here. It should be noted that the eight binary control lines can be set to implement these 256 truth tables. The general form of these truth tables is shown in Table 2.

TABLE 2

A_0	A_n	A_1'	A_1
0	0	n	$\bar{r}$
0	1	l	$\bar{p}$
1	0	o	$\bar{s}$
1	1	m	$\bar{q}$

From the point of view of pattern processing, one further control line, which sets the spare interconnections at the array margins at either (1) or (0), must be taken into account. This doubles the number of possible states of the system so that 512 functions can be listed.

A detailed study of these functions shows them to be conveniently subdivided into five categories or orders:

ZERO ORDER— the A_1 outputs are entirely (0) or (1) and independent of the A_0 pattern structure.

FIRST ORDER— the A_1 output is either equal to the A_0 INPUT or is equal to its binary COMPLEMENT.

SECOND ORDER—the A_1 output is a function of the A_0 INPUT to each cell, and of its IMMEDIATE NEIGHBOURS.

THIRD ORDER— the A_1 output is a function of the A_0 INPUT to each cell and of the MARGIN CONNECTIVITY of its NEIGHBOURS.

FOURTH ORDER—the A_1 output is UNDETERMINED due to logical inconsistency in the cell function.

(A margin connected cell has been defined[19] as a cell which connects, via cells of similar value, through to a cell on the margin of the array. Two cells are said to be connected when they have the same value and are neighbours.)

Although 512 functions can be implemented, the number of distinct operations is much smaller. Many of the truth tables result in identical pattern processing. Table 3 lists the numbers of distinct operations possible using the functions in each order.

TABLE 3

Order	Functions	Distinct operations
0	136	2
1	136	2
2	48	24
3	24	24
4	168	0
Total	512	52

Further examination of these operations reveals that the result of each operation is to output as (1) in A_1 those parts of the input image A_0 which exhibit one or more of a list of 14 topological properties. Dismissing the zero, first and fourth orders as trivial, these properties or descriptions are;

SECOND ORDER—(a) BLACK NOISE,* (b) BLACK EDGE,
(c) BLACK CORE,† (d) WHITE NOISE,
(e) WHITE EDGE, (f) WHITE CORE.

THIRD ORDER— (g) MARGIN CONNECTED BLACK CELLS.
(h) MARGIN CONNECTED WHITE CELLS.
(i) BLACK CELLS NOT MARGIN CONNECTED.
(j) WHITE CELLS NOT MARGIN CONNECTED.
(k) BLACK NEIGHBOURS OF (h).
(l) WHITE NEIGHBOURS OF (g).
(m) BLACK CELLS NOT NEIGHBOURS OF (h).
(n) WHITE CELLS NOT NEIGHBOURS OF (g).

* "Noise" implies either an isolated black cell (with no black neighbours) or an isolated white cell.

† "Core" is used to refer to black cells with no white neighbours and white cells with no black neighbours.

A particular truth table will select a particular combination of these features and output as (1) those cells exhibiting the features. For example, the truth table in Table 1 selects cells of type (a) and (b), and is therefore a second order function. Figure 6 shows how these descriptions apply to a simple pattern.

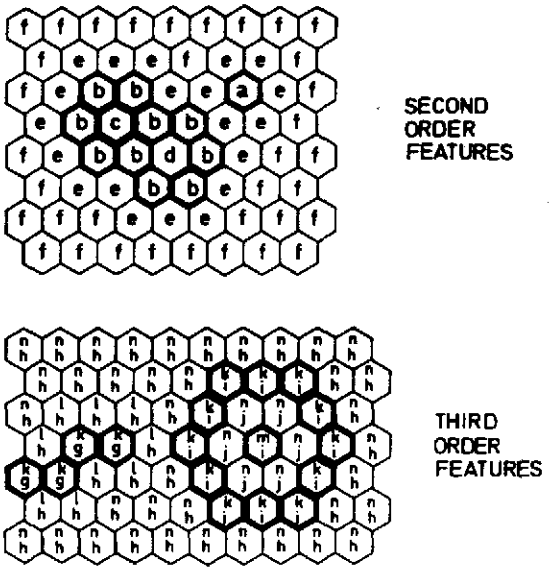


FIG. 6. Classification of cell types.

Some cells exhibit more than one property; this can be seen in the labelling of Fig. 6. All possible overlapping of properties is shown in the VENN diagram in Fig. 7 which maps the seven states describing black cells in A_0. The diagram shows the seven states to be dependent on four basic properties: the presence of black or white neighbours, margin connectivity of the cell itself and the margin connectivity of any white neighbours. The five orders of function are characterized primarily by the nature of the relationships governing the interconnection outputs A'_1. Zero and first order functions do not require interconnections (although particular interconnection functions can also give zero or first order outputs). Second order functions result when A'_1 is either A_0 or $\bar{A}_0$; in other words,

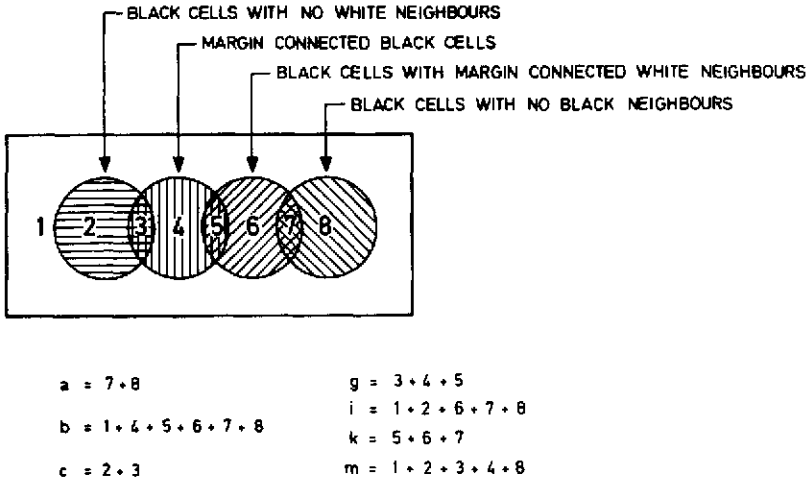


FIG. 7. Venn diagram of cell states.

the interconnection output depends only on the cell value. Third order functions result when A'_1 is equal to A_n for black cells and (0) for white cells, or vice versa. It is convenient to regard the cell as a switch which is either open or closed, depending on the cell value, and which regulates the transmission of the A_n interconnection signal. The unstable or undetermined output resulting from fourth order functions occurs when the value of the interconnection signal is inverted as it passes through a cell, for either black or white cells, or both. A pair of such cells which are neighbours can result in an inconsistent logical state.

Since time must elapse for the interconnection signals to propagate through the array, either to neighbours, as in the case of second order functions, or to some variable distance which depends on the pattern structure, as in the case of third order functions, it is necessary to allow a delay between entering A_0 from M_{in} and clocking out A_1 into M_{out}. The maximum propagation time τ for an array of dimensions L and M is approximately

$$\tau = \tfrac{1}{2}(M+1)Ld$$

where $L > M$ and both L and M are even, and where d is the propagation delay between and within consecutive cells.

2. *Pattern comparison.* Pattern P_1 is loaded into M_{in} and Pattern P_2 loaded into M_1. The two patterns can now be combined into A_1 by setting the switches S_2 so as to connect M_1 into the interconnection gates. At the same time the interconnection outputs are inactivated by zeroing the control lines l to o. Table 4 shows the form of the truth tables which are obtained for particular values of the control lines p to s. For example, the INCLUSIVE OR of P_1 and P_2 will appear in A_1 if r is set at (1) and the remaining control lines at (0).

3. *Labelled pattern processing.* If a pattern is stored in M_1, and M_1 is connected into the interconnection gates at G, then black cells in this pattern have the effect of making

TABLE 4

P_1	P_2	A_1
0	0	$\bar{r}$
0	1	$\bar{p}$
1	0	$\bar{s}$
1	1	$\bar{q}$

corresponding cells in the A_0 input behave as though they were margin cells with the margin interconnections set at (1). Such cells behave as sources of the interconnection signal for second and, more particularly, third order functions. Suppose, for example, that printed text is being scanned and the image A_0 appears at the array via M_{in}. By entering a small black line or cross into M_1, it is possible to label cells in the character which is centrally placed in A_0 so that the output contains only this character. To do this a third order function is chosen in which black cells which receive a (1) at A_n pass on a (1) at A_1', and only for these cells does A_1 take the value (1).

As will be discussed later, interesting results can be obtained when the pattern in M_1 is the result of a preliminary processing of the original input image.

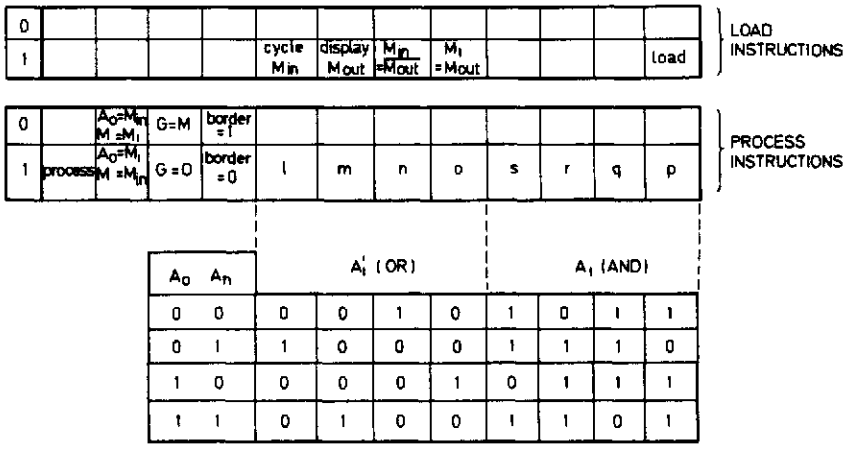
4. *Instruction programs.* The instruction set which can be employed in CLIP 2 programs is limited to one microprogrammable instruction for loading and displaying the contents of the memories, and one whose parameters determine the cell function to be implemented and the data sources. The instruction set is tabulated in Table 5.

The instruction word is either a LOAD instruction or a PROCESS instruction and each will comprise a maximum or four microinstructions as indicated in the table. The structure of the control words is shown in Fig. 8 which also shows the relationship between the process instruction word bits and the resulting truth table for the cell function.

The control circuit is arranged so that either individual word instructions or else the complete 32 word sequence can be executed. The cycle can be repeated indefinitely if required. Unused positions in the 32 long sequence are skipped.

TABLE 5

Instruction	Description	
Cycle M_{in}	Cycle the M_{in} register so that it can be loaded from the scanner or light pen if required (depending on control switch positions) and display its contents	Load instructions
Display M_{out}	Display the contents of M_{out}	
$M_{in} = \overline{M_{out}}$	Set M_{in} to equal the complement of M_{out}	
$M_1 = M_{out}$	Set M_1 to equal M_{out}	
Switch	Reverse, by means of S_1, the usual connections ($A_0 = M_{in}$ and $M = M_1$) so that for this instruction only $A_0 = M_1$ and $M = M_{in}$	Process instructions
$G = M$	Reverse, by means of S_2, the usual connections ($G = 0$) so that for this instruction only $G = M$.	
$B = 1$ (border = 1)	Connect all spare margin interconnection to (1) instead of the usual (0) connection.	
$M_{out} = \text{Function}(M_{in}, M_1)$	Set the control lines so that the required cell function is implemented and clock the processed pattern from the array into M_{out}.	



A_0	A_n								
0	0	0	0	1	0	1	0	1	1
0	1	1	0	0	0	1	1	1	0
1	0	0	0	0	1	0	1	1	1
1	1	0	1	0	0	1	1	0	1

FIG. 8. Control word structure.

B. *Performance*

1. *Process times.* The internal propagation delay within one cell is approximately 30 ns and delays for propagation to neighbouring cells are negligible in comparison. Thus zero and first order functions require 30 ns and second order functions, which involve processing in two consecutive cells or sets of adjacent cells, require about 60 ns. In the CLIP 2 16 by 12 array, the maximum propagation delay τ is calculated, from the formula $\tau = \frac{1}{2}(M+1)Ld$, to be 3.12 μs (putting $L = 16$, $M = 12$ and $d = 30$ ns). However, maximum length paths would rarely be encountered in any image being processed so that a 1 μs delay, allowing propagation through 33 consecutive cells, should be quite adequate. Table 6 summaries

TABLE 6

Function order	Process time (ns)
0	30
1	30
2	60
3	1000

these times. The times taken to load an image into M_{in}, or to display the contents of M_{in} or M_{out}, depend on the scanner and display system, and are not the concern of this investigation. Should these times become limiting in a real-time application of a parallel processing system, then presumably the serial scanning and displaying devices would be replaced by parallel devices.

2. *Sample programs.* CLIP 2 is a parallel processing computer intended for image processing; as with any computer, it is not sensible or even possible to try to list all the programs that can be run on it. In order to indicate the nature of the programs which have been written so far, a few selected programs will be described in detail.

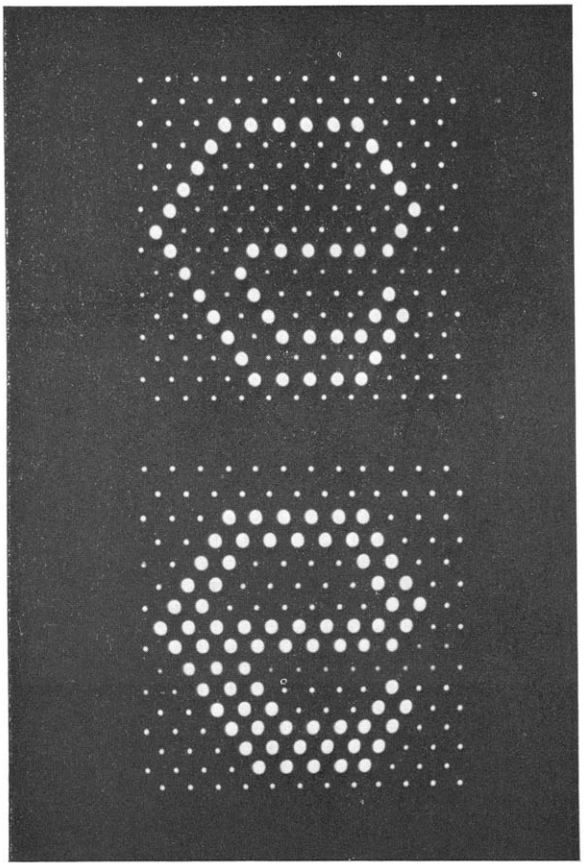


FIG. 9. Input and output display for "figure outside edge" program.

[*facing page* 240]

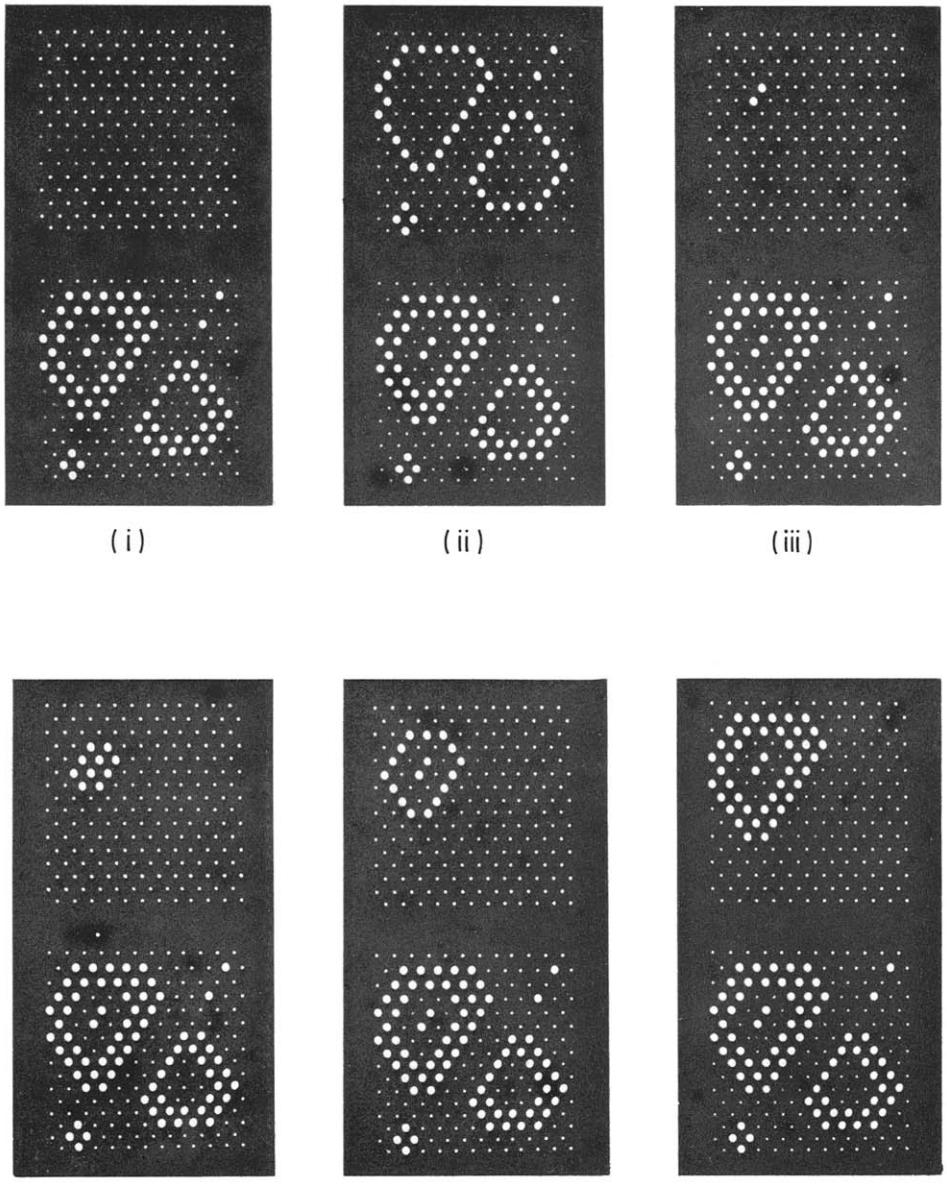
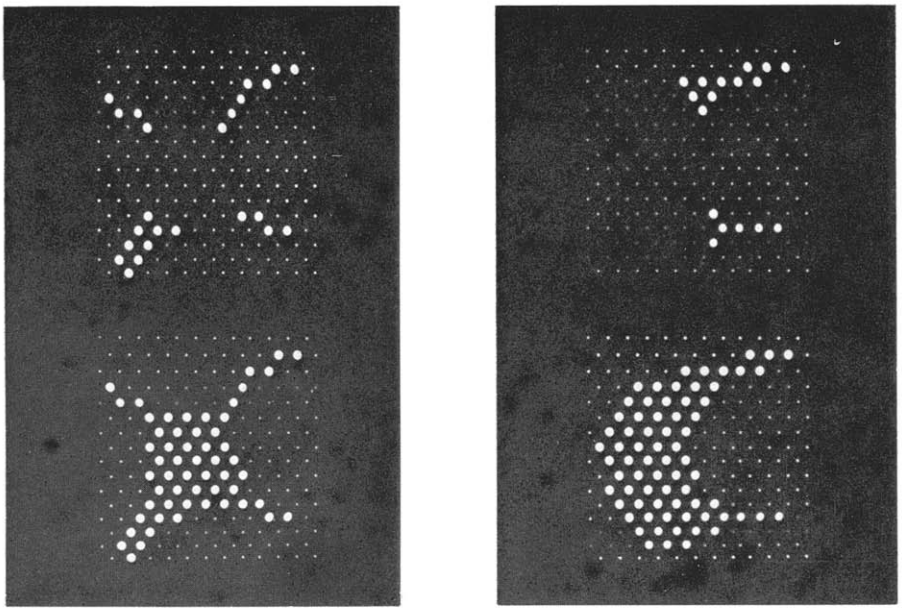


FIG. 10. Process steps in "nucleated cells" program.



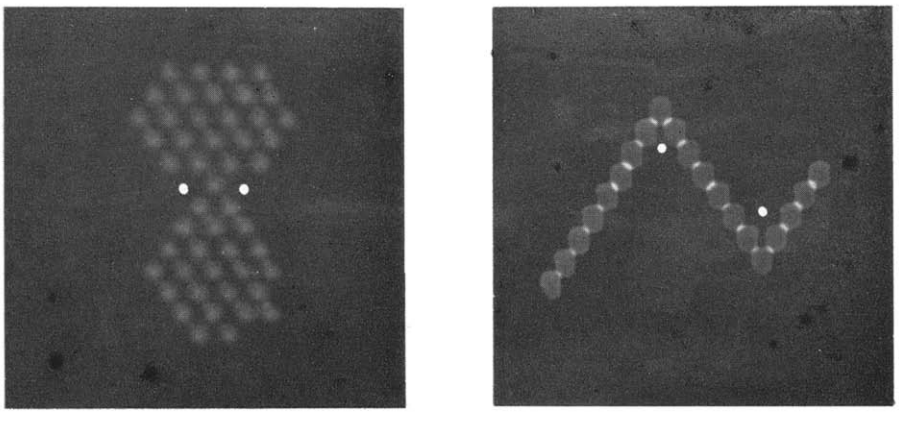


FIG. 11. Various processing examples.

(i) Figure outside edges (contours). This program comprises only two instructions: the first loads an image into M_{in} and displays it, whilst simultaneously displaying the contents of M_{out}; the second is a process instruction.

(1) CYCLE/DISPLAY (M_{in}, M_{out})

(2) $B = 1$, M_{out} = OUTER EDGE (M_{in})

The function truth table is shown in Table 7(a). All margin cells receive a (1) on their spare interconnection inputs. White cells at the margin on receiving a (1), pass a (1) through the cell to neighbours as A_1'. The A_1 output is (1) only for black cells which receive a (1). These are neighbours of margin connected white cells or black cells in the margin.

TABLE 7

A_0	A_n	A_1'	A_1	A_1'	A_1	A_1'	A_1'	A_1'	A_1	A_1'	A_1
0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	1	0	0	1	0
1	0	0	0	0	1	1	1	0	0	0	1
1	1	0	1	1	0	1	1	1	1	0	0
Function		Outer edge		F_1		Expand		F_2		Thin	
		(a)		(b)		(c)		(d)		(e)	

The two instructions can be repeated indefinitely permitting real-time contour finding in a scanned input. A typical display is shown in Fig. 9.

(ii) Objects containing other objects (Nucleated cells). To avoid confusion in this section array cells are referred to as "elements" and the word "cell" reserved for biological cells. This program is designed to display those parts of the input image which are closed loops enclosing black elements (separately or in a group) which do not touch the closed loop. A microscope slide containing some cells with nuclei and some without would present this situation; the program would display the nucleated cells and reject the others (also rejecting solid black specks outside the cell walls). The program is a continuously cycled sequence of 10 instruction, as follows:

(1) CYCLE/DISPLAY (M_{in}, M_{out})

(2) $B = 1$, M_{out} = OUTER EDGE (M_{in})

(3) $M_1 = M_{out}$

(4) $G = M$, $M_{out} = F_1(M_{in}, M_1)$

(5) $M_1 = M_{out}$

(6) Switch, M_{out} = EXPAND (M_1)

(7) $M_1 = M_{out}$

(8) $G = M$, M_{out} = OUTER EDGE (M_{in}, M_1)

(9) $M_1 = M_{out}$

(10) $G = M$, $M_{out} = F_2(M_{in}, M_1)$

Figure 10 illustrates the steps in this process. Instruction (2) locates all contours and (3) transfers them to M_1. Function F_1 in instruction (4) treats these stored contours as labels and removes all parts of the original image in M_{in} which connect through to the labelled array elements; this implies the removal of all black elements external to the cells and all cell walls, leaving only "nuclei." The nuclei are transferred to M_1 and expanded by one neighbour set by instruction (6). The expanded nuclei are transferred to M_1 and instruction (8) uses the nuclei as labels in the original image, so that a signal propagates through the white

elements enclosed by the nucleated cell walls. The "outer edge" thus detected is now the inside of the cell walls. These are transferred to M_1 together with the expanded nuclei and the final instruction uses function F_2 to select those parts of the original image which connect to elements coincident with the labels stored in M_1, in other words: the complete nucleated cells. The truth tables for the various functions used are given in Table 7(a)–(d).

(ii) Projections on well shaped objects. It is sometimes important to be able to extract the "hard core" of an object which has thin projections from its central region (for example, dendritic structures on neurons). This can be achieved by a program which thins the figure twice and expands it twice, and then compares the new and the original figures. The parts of the figure which have disappeared in this process are the projections.

The program has 12 instructions as follows:

(1) CYCLE/DISPLAY (M_{in}, M_{out})
(2) $M_{out} = M_{in}$
(3) $M_1 = M_{out}$
(4) Switch, $B = 1$, M_{out} = THIN (M_1)
(5) $M_1 = M_{out}$
(6) Switch, $B = 1$, M_{out} = THIN (M_1)
(7) $M_1 = M_{out}$
(8) Switch, M_{out} = EXPAND (M_1)
(9) $M_1 = M_{out}$
(10) Switch, M_{out} = EXPAND (M_1)
(11) $M_1 = M_{out}$
(12) $M_{out} = M_{in} \cdot \overline{M}_1$

The function THIN is shown in Table 7(e). The result of applying the program to two images is shown in Fig. 11. Two further output displays, one from a program detecting CONCAVITIES, and the other from a LINE BENDS program, are also shown in this figure.

IV. CONCLUSIONS

A. *Future developments*

Clip 2 has been constructed as an operational hardware system whilst maintaining a full awareness of its limitations. Invaluable experience in designing, constructing and operating parallel systems has been obtained. The real time, interactive programs have acted as a spur to the development of new algorithms. Nevertheless, from a practical point of view, it is appreciated that the CLIP 2 cell and operating system is not sufficiently sophisticated to be of use in the solutions of real problems. CLIP 3 has been designed and is now being constructed and differs from CLIP 2 in the following features:

1. The basic cell is anisotropic in that each interconnection is separately gated into the cell. Thus a single instruction will make all the connections to neighbouring cells which are displaced in a particular direction from each cell in the array.
2. The interconnection NAND gate is replaced by a variable threshold gate.
3. The array can be switched from hexagonal to square architecture, with 4- or 8-connectivity (this is discussed in detail in the Appendix).
4. The number of bits of storage is increased from 3 to 18/cell.
5. A conventional serial computer will act as controller for the array, bringing all array functions under software control.

6. Although the same array size of 16 by 12 cells will be maintained, techniques for scanning this array across much larger fields will be developed.

It is hoped that CLIP 3 will be fully operational in the early Spring of 1973. Later in the year, it is intended to design a large scale integrated circuit implementing those parts of the CLIP 3 cell which prove valuable, incorporating these cells in a larger array which will be known as CLIP 4. The same operating system as will be used for CLIP 3 will be adapted for CLIP 4.

B. *Summary*

It has been shown that parallel processing hardware is likely to prove satisfactory for real time image processing, both from the point of view of cost (the CLIP 2 system cost $2000 USA and one man year) and speed of processing. Although CLIP 2 is insufficiently complex for use in applications, experience gained with CLIP 2 has pointed to worthwhile improvements for inclusion in later designs (CLIP 3 and CLIP 4) and has provided a useful tool as an aid to the development of parallel processing algorithms. It remains to be seen whether the confidence that is now felt in these techniques bears fruit in the research program now being conducted.

SUMMARY

This paper describes CLIP 2, a cellular logic image processor, which has been designed and constructed at University College London. A 16 by 12 array of hexagonally connected cells, each with connections to its immediate six neighbours, receives a binary pattern on its 192 input wires. The pattern is transferred in parallel from a serially loaded shift register fed from a flying spot scanner. Each cell, which is an assembly of logic gates contained in five TTL integrated circuits, is programmable to provide two independent binary outputs which are boolean functions of the two inputs to the cell, the one input being the pattern and the other being obtained by taking the NAND of the interconnection outputs from neighbouring cells. During any particular process, every cell in the array is set to perform an identical function.

The interconnections allow propagation of information within the array, so that it is possible for all parts of the input image to contribute to all parts of the output image.

The processing times in the array are very short; functions which do not propagate are completed in about 30 ns whereas on an array of this size, propagating functions all reach a final state in about 3 μs, even for very complex patterns. CLIP 2 can be programmed to perform series of instructions and some of these programs which are of interest in image processing are described in detail. The cells are interconnected symmetrically and isotropically so that image processing algorithms requiring directional operations (rather than topological operations) are not achievable. A more complex array is now being developed (CLIP 3) and its significant differences from CLIP 2 are discussed.

REFERENCES

1. S. H. UNGER, A computer orientated toward spatial problems, *Proc. IRE* **46** (10), 1744 (1958).
2. S. H. UNGER, Pattern detection and recognition, *Proc. IRE* **47** (10), 1737 (1959).
3. J. GREGORY and R. MCREYNOLDS, The SOLOMON computer, *Trans IEEE* **EC-12** (6), 774 (1963).

4. D. H. HUBEL and T. N. WIESEL, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, *Trans. IEEE* **MIL-7**, 98 (1963).
5. I. ALEKSANDER and E. H. MAMDANI, Microcircuit learning nets: improved recognition by means of pattern feedback, *Electr. Letters* **4** (20), 425 (1968).
6. D. BROWN, M. HALL and S. LAL, Pattern transformation by neural nets, *J. Physiol.* **209**, 7P (1970).
7. J. K. HAWKINS and C. J. MUNSEY, A parallel computer organization and mechanizations, *Trans. IEEE* **EC-12** (3), 251 (1963).
8. E. S. DEUTSCH, Thinning algorithms on rectangular, hexagonal and triangular arrays, Univ. of Maryland Computer Sci. Center, Tech. Rpt. 70-115 (1970).
9. M. J. E. GOLAY, Hexagonal parallel pattern transformation, *Trans. IEEE* **C-18**, 733 (1969).
10. S. B. GRAY, Local properties of binary images in two dimensions, *Trans. IEEE* **C-20** (5), 551 (1971).
11. S. LEVIALDI, On shrinking binary patterns, *Communs Assoc. Comp. Mach.* **15** (1), 7 (1972).
12. K. PRESTON, JR., Feature extraction by Golay hexagonal pattern transforms, *Trans. IEEE* **C-20**, 1007 (1971).
13. A. ROSENFELD, Connectivity in digital pictures, *J. Assoc. Comp. Mach.* **17** (1), 146 (1970).
14. E. E. TRIENDL, Skeletonization of noisy handdrawn symbols using parallel operations, *Pattern Recognition* **2**, 215 (1970).
15. R. C. MINNICK, A survey of microcellular research, *J. Assoc. Comp. Mach.* **14** (2), 203 (1967).
16. B. H. MCCORMICK, The Illinois pattern recognition computer—ILLIAC III, *Trans. IEEE* **EC-12** (6), 791 (1963).
17. S. LEVIALDI, Parallel counting of binary patterns, *Electr. Letters* **6** (25), 798 (1970).
18. M. J. B. DUFF, B. M. JONES and L. J. TOWNSEND, Parallel processing pattern recognition system UCPR1, *Nucl. Instrum. Meth.* **52**, 284 (1967).
19. M. J. B. DUFF, Cellular logic and its significance in pattern recognition, AGARD Conf. Proc. No. 94 on Artificial Intelligence, 25-1 (1971).
20. M. J. B. DUFF, University College London Dept. of Physics, Int. Rpt. (1972).

APPENDIX

The relative merits of tessellation in the three alternative forms shown in Fig. 1 have been considered by ROSENFELD,[13] GRAY[10] and DEUTSCH.[8] Arguments usually treat two questions: is the topology in the original image faithfully represented in the tessellated image, and is the representation of detail in the tessellated image adequate for the image processing task which is to be performed.

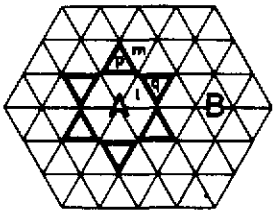
The neighbours of a cell are defined as those cells which touch the cell either at an edge or at a corner. The hexagonal array has the simplest structure; each cell has six neighbours all of which are the same distance from the central cell and all of which connect with the central cell by means of a common edge. The square array provides eight neighbours which are in two subsets. Those labelled (a) in Fig. 1(ii) connect to a common edge and those labelled (b) connect only at corners. The (b) neighbours are further from the central cell than are the (a) neighbours. The most complicated structure is the triangular array with the twelve neighbours each in one of three modes of connection and at three distances from the central cell. The three (a) cells form a common edge, the six (b) cells touch at a corner and are slightly further from the central cell, and the three (c) cells touch at a corner and are furthest away from the centre. Table 8 summarizes these characteristics.

If neighbouring cells are equal valued (i.e. both (1) or both (0), then they are said to be connected. In the hexagonal array, the definition presents no difficulties, but it is necessary to consider two types of connectivity in the square array (and three types in the triangular array). In the first type only the four (a) cells are regarded as neighbours and only these therefore connect with the central cell (see Fig. 1). In the second type, however, both the (a) and the (b) cells are treated as neighbours. Similar arguments apply to the triangular array. Unfortunately, both connectivity schemes present us with a paradox. A simple but informative explanation of the paradox can be given with reference to Fig. 12. In each case, a

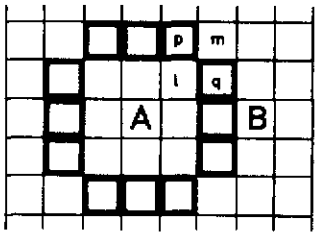
TABLE 8

Array	No. of neighbours (a)	(b)	(c)	Distance between neighbour and central cell (a)	(b)	(c)	Cell area
Triangular	3	6	3	$\frac{1}{\sqrt{3}}C$	C	$\frac{2}{\sqrt{3}}C$	$\frac{\sqrt{3}}{4}C^2$
Square	4	4	—	C	$\sqrt{2}C$	—	C^2
Hexagonal	6	—	—	$\sqrt{3}C$	—	—	$\frac{\sqrt{3}}{2}C^2$

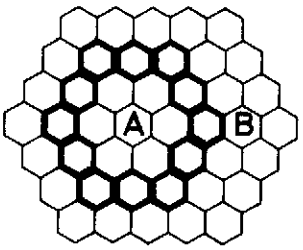
C = length of a side of the cell.



(i)



(ii)



(iii)

FIG. 12. Illustration of connectivity paradox.

ring-like figure is represented. If we assume cells such as those labelled p and q are connected, then the black figure is continuous and the figure is closed; regions A and B are therefore not connected. However, we must also assume that cells such as l and m are connected, since they are in the same neighbour relationship as p and q. This would imply that the white regions A and B are connected, contrary to the previous conclusion. Alternatively, if the corner-to-corner relationship is not regarded as a connection, then the black ring is discontinuous. But the paradox persists because now no cells in A connect with cells in B. This situation can be overcome by adopting different criteria for connection between white cells and between black cells, but this would appear to be an unnecessary complication since the paradoxical situation does not arise in the hexagonal array. The paradox can be stated more formally and analysed in detail in terms of the Euler theorem on polygonal networks.[10] Calculations have also been made to determine which of the square or the hexagonal arrays offers highest resolution for the same number of cells in a given image area.[20] A length parameter l is defined as the array resolution and a relation between l and the length of the side of an array cell sought. The constraints which define l are:

1. A straight black bar of width l is continuous in the array tessellation.
2. A straight white bar of width l is continuous in the array tessellation.
3. Black regions separated by a straight white bar of width l are not connected in the array tessellation.
4. White regions separated by a straight black bar of width l are not connected in the array tessellation.

In comparing the relative efficiencies of different arrays, it is reasonable to assume that each array will have the same number of cells. The cell size can then be determined by equating cell areas. Thus if the cell side lengths in the triangular, square and hexagonal arrays are t, S and h respectively, then the cell area is α

$$\alpha = \frac{\sqrt{3}}{4}t^2 = S^2 = \frac{\sqrt{3}3}{2}h^2.$$

TABLE 9

Tessellation	Black cell connectivity	White cell connectivity	$l/\sqrt{\alpha}$	ϕ
Square	4	4	$\sqrt{2} = 1.414$	0.5
	4	8	$\sqrt{5}-1 = 1.236$	0.382
	8	4	$\sqrt{5}-1 = 1.236$	0.618
	8	8	$\sqrt{2} = 1.414$	0.5
Hexagonal	6	6	$\left(\frac{2}{\sqrt{3}}\right)^{1/2} = 1.075$	0.5

As a simplification, it is assumed that the input image is two levelled, being black or white. If a black portion of the image obscures at least a fraction ϕ of the area of a cell, the cell output will be (1), otherwise it will be (0). The results of the calculations are shown in Table 9 where it can be seen that the ratio of l (the resolution) to $\sqrt{\alpha}$ is a minimum for the hexagonal array. Results are quoted for the four possible connection schemes for the square

array. In view of the high complexity of the triangular array and the unsatisfactory nature of its various connectivities, it has not been treated in detail here. It is sufficient to state that the triangular array appears to offer no advantages over either the square or hexagonal arrays.

In summary, it would appear that the hexagonal array is preferable to the square array with respect to both resolution and connectivity properties.