Digital Optical Iterative Pattern Recognizer

S P Kozaltis
C S Kischuk

Digital optical iterative pattern recognizer

S.P. Kozaitis

Florida Institute of Technology
Department of Electrical and Computer Engineering
150 W. University Blvd.
Melbourne, Florida 32901

C.S. Kischuk

Wayne State University
Department of Electrical and Computer Engineering
Detroit, Michigan 48202

ABSTRACT

An optical pattern recognizer which classifies an input sequence into one of two classes is described. The language of regular expressions is used to identify a set of input sequences to be recognized. The system is realized without memory elements by using an optical iterative array implemented with the method of symbolic substitution. The hardware must be able to implement arbitrary and multiple substitution rules for the general design of a pattern recognizer.

1. INTRODUCTION

Although the interconnection capabilities of optics show promise for an optical computing system, the lack of optical memory elements often inhibits an implementation. It is well known in electronic systems, that many identical combinational computing elements can perform the same function as a sequential machine with memory elements[1]. Identical computing cells with regular interconnects are called iterative arrays. Iterative arrays are well-suited to optical implementation because of the regular interconnects needed and the superior fan-in and fan-out capabilities of optics.

Sequential machines which classify patterns into one of two classes are investigated and implemented in an optical iterative array. By using regular interconnects and optical combinational logic, optical sequential machines can be designed without memory elements. Logic operations are distributed over many elements of a large array with the method of symbolic substitution. In this fashion, the advantages of optics can be applied to sequential machines.

2. ITERATIVE ARRAYS

An iterative array is an array made up of identical computing cells with regular interconnects[2]. The cells are usually either combinational or sequential logic networks and may be connected in different types of ways[3]. Furthermore, iterative arrays may be designed in several different topologies according to the data flow, such as, unilateral, bilateral or multi-dimensional.

A one-dimensional unilateral iterative array is shown in figure 1. The inputs to the array are labled X1, X2,...Xn. All inputs are applied simultaneously. The outputs of the array are Z1, Z2,...Zn, which are available simultaneously. The variable n refers to the cell number. In this example, there are two intercell connections which carry data between cells. The m variables, Y0n-Ymn of each cell, indicate the state of the nth cell. The functions within the cells and the exchange of data between cells can be performed using symbolic substitution.

3. SYMBOLIC SUBSTITUTION

Symbolic substitution is a method for performing a type of spatial logic[4-7]. Both the value of a pixel and its location are important. A spatial arrangement of pixels is mapped to a different arrangement according to a set of substitution rules. The execution of arithmetic operations is especially interesting since the substitution rules may be performed in parallel. The representation of a binary one and zero are typically the complementary patterns shown in figure 2, while the two dark cells represent no data. Operations are performed by applying the appropriate substitution rules of the function of interest.

Since the substitution rules are a kind of spatial truth table, they may implement arbitrary switching functions[8]. By using higher-ordered rules, the number of substitution steps may be reduced[8]. Each cell of an iterative array may perform a function defined by a set of substitution rules. By distributing a substitution rule over an array such as with a spatial light modulator (SLM), operations common to iterative arrays can be performed.

4. MODELING OF RECOGNIZER

A set of strings or input sequences which are to be recognized can be described by regular expressions[9]. After a set has been described, it is an easy matter to convert a regular expression to a transition graph or state diagram. From the state diagram, an iterative array can be designed. As an example, an input sequence consisting of two consecutive 0's and any combination of 1's and 0's before or after the two consecutive 0's will be recognized. The sequences 01010, 01111111110 and 010110 will not be recognized and will produce a 0 output. The sequences 001111010, 11001, and 011001 will be recognized and produce a 1 output. Using the language of regular expressions[9], the sequence F to be recognized may be written as:

$$F=(0+1)^{*}00(0+1)^{*} \qquad (1)$$

The terms in parentheses indicate that any combination of 0's and 1's (including the empty set) may precede or follow two consecutive 0's. Any function which can be expressed in terms of regular expressions can be described by a state diagram.

Inputs to the recognizer enter state A of the state diagram which is shown in figure 3. If the next input in the sequence is either a 0 or a 1, the next state will be B. If any input sequence is applied which is does not contain two consecutive 0's, then the recognizer will remain in state A or B. The λ indicates that a transition from one state to another occurs without an input applied. When two consecutive 0's are in the input sequence, the recognizer will go to state D. Whenever the state D is reached, a 1 is sent to the output, otherwise the output is 0. Through the λ transition, the recognizer starts over again recognizing the proper pattern. Traditionally, the state diagram can be implemented by a sequential circuit with combinational logic and memory elements according to digital logic theory.

5. RULE EXTRACTION

The sequential circuit described in figure 3 may be implemented with an iterative array of combinational logic networks. Generally when considering a one-dimensional array, the network within each cell is the same as the combinational network designed for a sequential circuit[1]. The number of cells is often greater than or equal to the number of states in the state diagram. The number of cells is equal to the length of the sequences being examined.

The state diagram in figure 3 is redrawn in figure 4 to show the state assingments. Each cell has one input Xi, and one output Zi. The two intercell connections describe the four states. State A is represented by Yi1=0, Yi2=0, state B by Yi1=0, Yi2=1, state C by Yi1=1, Yi2=0 and state D by Yi1=1, Yi2=1. The logical equations can be written as:

$$Zi = (yi1)(yi2)'(xi)' \qquad (2)$$

$$Yi1 = (yi1)'(xi)'+(yi2)'(xi)' \qquad (3)$$

$$Yi2 = (yi1)'(yi2)'(xi)+(yi1)(yi2)(xi) + (yi1)(yi2)'(xi)' \qquad (4)$$

The variable ymn refers to the input to a cell and the variable Ymn refers to the output of a cell, or equivalently, the present and next states respectively. The substitution rules to describe these equations are shown in figure 5. For this example, the inputs yi1, yi2 and xi are placed in the same column in different rows. The output Zi is placed in the bottom two rows in the same column as the inputs. The intercell outputs, Yi1 and Yi2 are placed one column to the left of the inputs. The inputs to the ith cell are referred to as yi1 and yi2 as indicated in figure 5.

The recognizer is implemented in figure 6. An example input sequence is shown along the top two rows. The system starts from state A at the leftmost column which is column 1. The first column is replaced according to one of the rules in figure 5. For each column, one of the eight rules is applied. The system must have the ability to implement multiple rules simultaneously. The remaining columns are replaced in the same fashion until the last column is reached. A one output is classified when an input string is determined to satisfy the state diagram.

6. CONCLUSION

Sequential functions may be implemented optically without the need for feedback or optical memeory elements. Bit serial algorithms which may be described by regular expressions may be implemented with this method. The hardware must be able to implement multiple substitution rules simultaneously. The regular interconnection pattern and higher-ordered rules are well suited to optics through the interconnection cabilities. Fault tolerance may be added by repetition of interconnects.

7. ACKNOWLEDGEMENT

This work began when both authors were at Wayne State University.

8. REFERENCES

1 Z. Kohavi, Switching and Finite Automata Theory, McGraw-Hill: New York (1978)

2 F.C. Hennie, Iterative Arrays of Logical Circuits, The MIT Press: Cambridge, Mass. (1961)

3 K. Hwang, F.A. Briggs, Computer Architecture and Parallel Processing, McGraw-Hill: New York (1984)

4 M.J. Murdocca, "Digital optical computing with one-rule cellular automata," Appl. Opt. 26, 682 (1987)

5 K-H. Brenner, A. Huang, N. Streibl, "Digital optical computing with symbolic substitution," Appl. Opt. 25, 3054 (1986)

6 P.A. Ramamoorthy, S. Anthony, "Optical modified signed adder using polarization-coded symbolic substitution," Opt. Eng. 26, 821 (1987)

7 R.P. Bocker, B.L. Drake, M.E. Lasher, T.B. Henderson, "Modified signed-digit addition and subtraction using optical symbolic substitution," Appl. Opt. 25, 2456 (1986)

8 S.P. Kozaitis, "Higher-ordered substitution rules," Opt. Comm. (to be published) (1988)

9 J.A. Brzozowski, "A survey of regular expressions and their applications," IRE Trans. Comp. EC-11, 324 (1962)

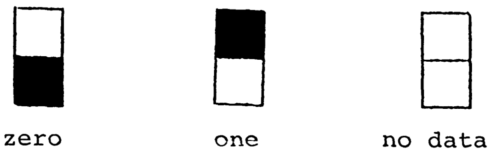
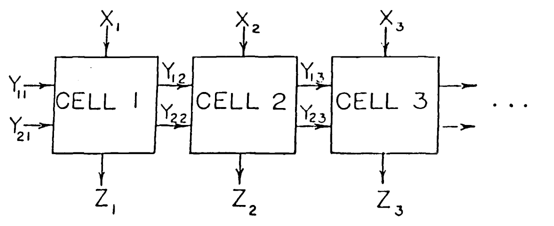
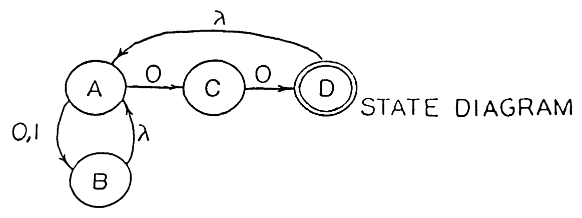


Figure 2 Coding of inputs



X_n - INPUT OF n-TH CELL
Y_{mn} - m-TH STATE VARIABLE OF n-TH CELL
Z_n - OUTPUT OF n-TH CELL

Figure 1 One-dimensional unilateral iterative array



present state	next state, z: x = 0	next state, z: x = 1
A	C,0	B,0
B	C,0	A,0
C	D,1	A,0
D	B,0	B,0

STATE TABLE

Figure 3 State diagram of pattern recognizer

present state	next state, Z X 0	1
00	10,0	01,0
01	10,0	00,0
10	11,1	00,0
11	01,0	01,0

Figure 4 State table with state assignments

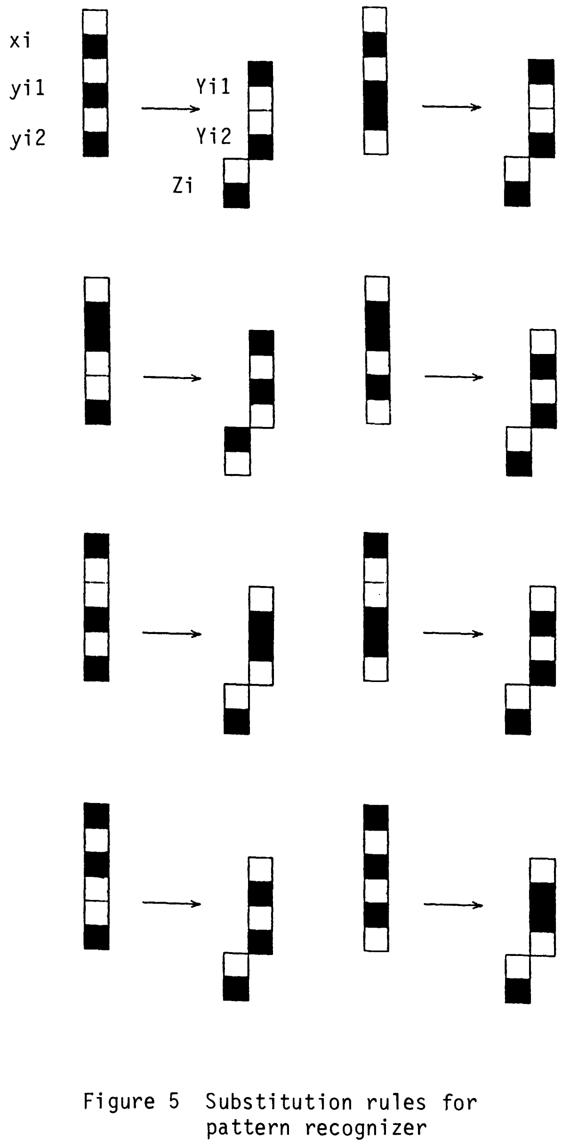


Figure 5 Substitution rules for pattern recognizer

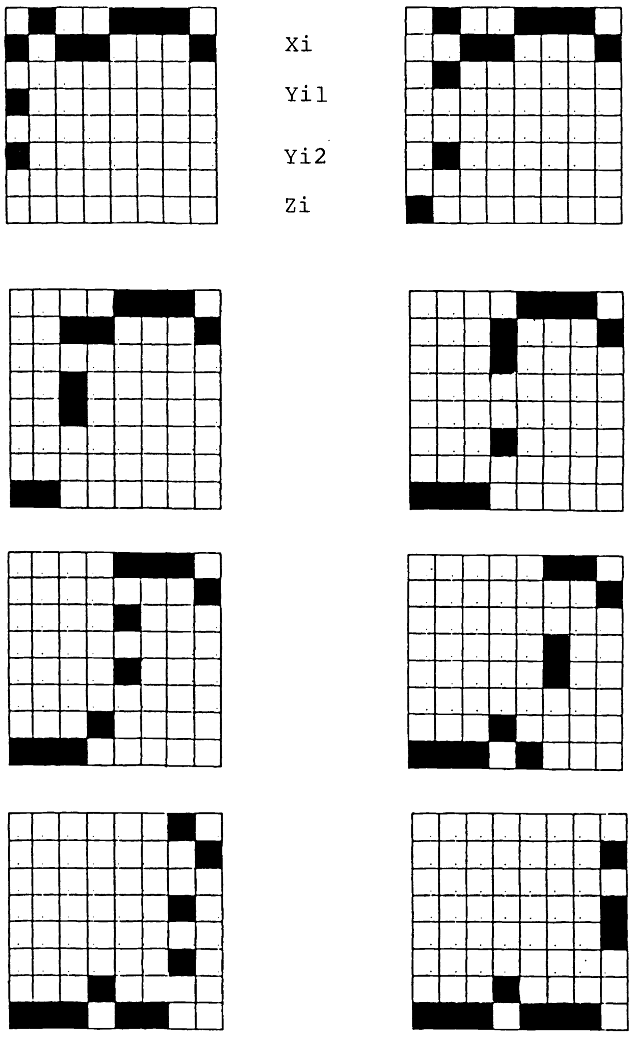


Figure 6 Array representation of pattern recognizer