Technical Report CS73006-R

Microprogrammable Cellular Automata

by

J. C. Muzio

and

T. C. Wesselkamper

Authors' addresses: J. C. Muzio, Dept. of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada; T. C. Wesselkamper, Dept. of Computer Science, Virginia Polytechnical Institute and State University, Blacksburg, Virginia, U.S.A.

Errata

page 7, line 4 should read:

$$(pq + pr + qs) \;\Big|\; t(p + r + \bar{s}) + qr\bar{s}t,$$

page 7, line 6 should read:

AJAKpqAKprKqsKtApArNsKqKrKNst,

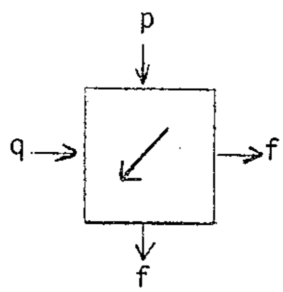
Abstract

This paper reports research into cellular automata with two binary inputs, two binary outputs, and an octal control variable. A set of control variables is chosen and it is shown that any function of three variables can be realized by a 2 x 2 array of cells, any function of four variables by a 2 x 6 array of cells. A construction based on the Shannon Decomposition Theorem is given for the realization of functions of more than four variables. The existence of a more efficient construction is conjectured. A definition of the circuit defining the cell is given as well as an implementation using NAND gates. A practical configuration of the cells is suggested and fault correction is discussed.

This paper presents, for the most part, a statement of new results with only a sketch of the proofs of those results. At this time elegant proof methods are lacking and, as a result, the proofs involve consideration of a large number of cases. An extensive bibliography on cellular automata is found in [2].

1. Fundamental Results.

We consider a cell with two binary inputs and two identical binary outputs. In addition the cell possesses an octal control input. This control input determines the function defined by the cell.

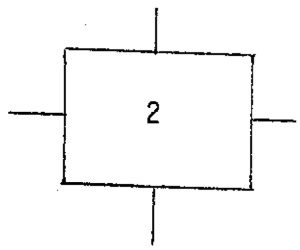


We say that Fpq = <abcd> whenever F00 = a, F01 = b, F10 = c, and F11 = d. The eight functions defined by the eight control values are:

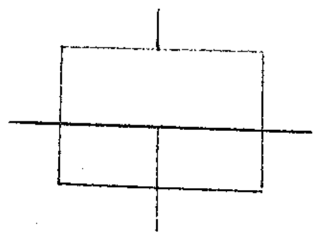
Control	Function	
0	Kpq = <0001>	(and)
1	Dpq = <1110>	(nand)
2	q = <0101>	(left input)
3	Jpq = <0110>	(nonequivalence)
4	p = <0011>	(top input)
5	Cpq = <1101>	(implication)
6	Apq = <0111>	(or)
7	Xpq = <1000>	(nor)

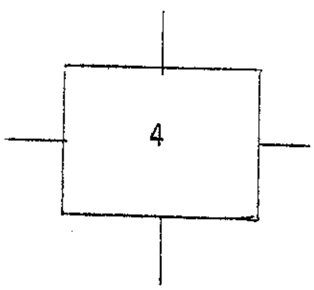
Hereafter we refer to the set of variables defined in this table as the set of control variables.

It is convenient to use the following schematic representations for the cell when the control variable is 2 or 4:

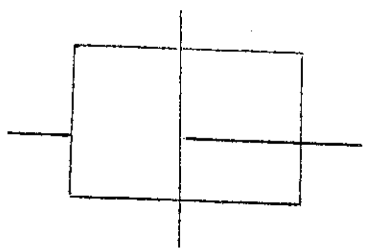


we represent as:



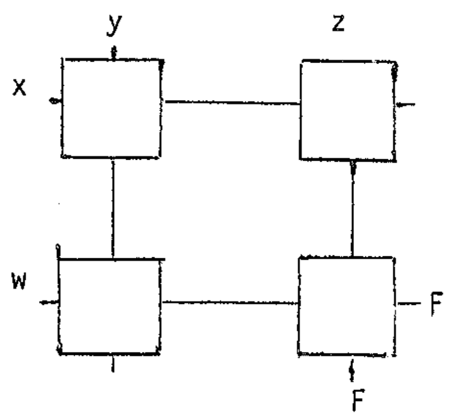


we represent as:



The selection of the set of control variables is, to some degree, the result of experiment. The functions K, D, J, A, and X are reasonably natural choices and these functions have the advantage of being commutative. Very early it became clear that it is necessary to include p and q. C was chosen after it was determined that our first two choices (Np and Epq) were insufficient to prove Theorem 1. The function Cpq may be replaced by Bpq = Cqp. Cpq can be replaced by neither Lpq = NCpq nor Mpq = NBpq.

Theorem 1: If Fpqr is a function of three binary variables then Fpqr may be realized by a 2 x 2 array of cells with inputs w, x, y, z, where w, x, y, z $\in$ {p, q, r}. Thus:

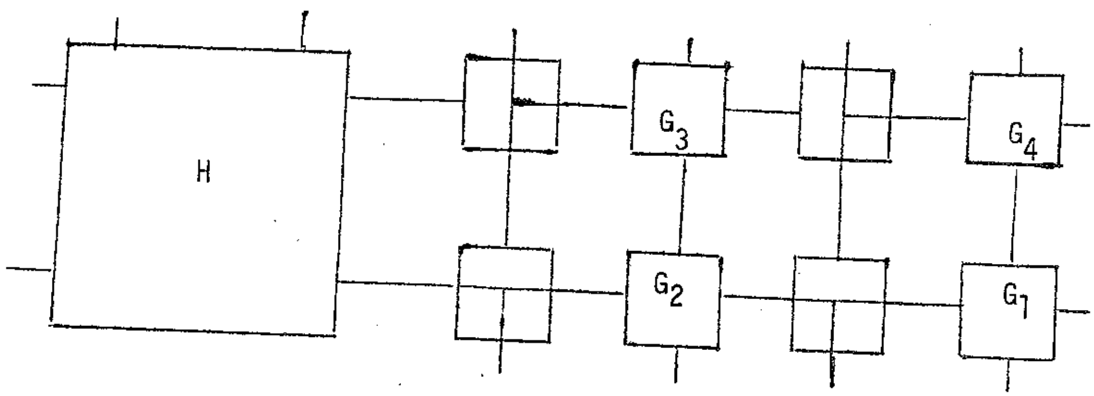


The proof consists in dividing the 256 functions of three variables into thirteen classes and showing that each class may be represented by a fixed substitution of p, q, and r for w, x, y, and z.

Theorem 2 can be used to obtain a realization of all four place binary functions.

Theorem 2: If Fwxyz is a function of four binary variables then there exist functions H, G_1, G_2, G_3, and G_4 such that H is a function of three variables and G_1, G_2, G_3, and G_4 are elements of the set of control variables and $Fwxyz = G_1G_2Hk_1k_2k_3G_3k_4k_5G_4k_6k_7$, where $k_i \in \{w, x, y, z, 0, 1\}$, for $1 \leq i \leq 7$.

The proof of the theorem depends upon looking at 222 classes and also showing that the use of inverted inputs is unnecessary [3]. Since H may be realized by a 2 x 2 array of cells, we have F realized by the 2 x 6 array of cells:



It is possible to extend this result to functions of more than four variables using Shannon's Decomposition Theorem, that is, by using the fact that if F is a function of the variables $x_1, x_2, \ldots, x_n$, then we may write:

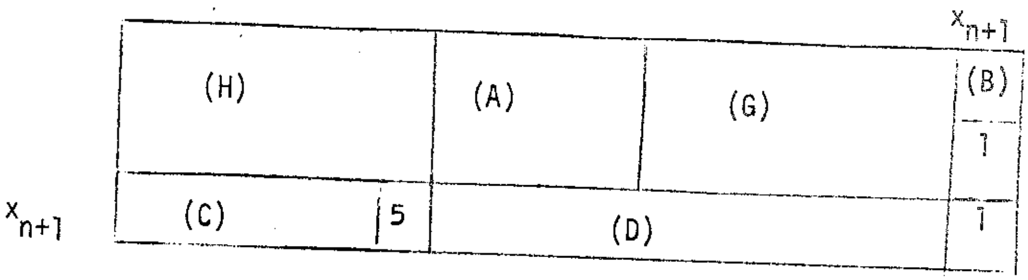
$$\begin{aligned} Fx_1 \ldots x_n &= x_n Fx_1 \ldots x_{n-1}1 + \overline{x_n} Fx_1 \ldots x_{n-1}0 \\ &= x_n Gx_1 \ldots x_{n-1} + \overline{x_n} Hx_1 \ldots x_{n-1}. \end{aligned}$$

We write this in parenthesis-free notation and note that it may take an alternate form:

F = AKxGKNxH = DCHxDxG, where G and H are functions.

<u>Theorem 3</u>: If $n \geq 4$, then $Fx_1 \ldots x_n$ may be realized by an array of $(n - 2) \times (10 \cdot 2^{n-4} - n)$ cells.

The theorem is proved by induction. The basis of the induction is Theorem 2. The induction step is shown, for $Fx_1 \ldots x_n x_{n+1} = x_{n+1} Gx_1 \ldots x_n + \overline{x_{n+1}} Hx_1 \ldots x_n$.



where (H) is the $(n - 2) \times (10 \cdot 2^{n-4} - n)$ cells required to realize $Hx_1 \ldots x_n$;

(G) is the $(n - 2) \times (10 \cdot 2^{n-4} - n)$ cells required to realize $Gx_1 \ldots x_n$;

(A) is the $(n - 2) \times (n - 2)$ array having 4 as control variable on and above the secondary diagonal and having 2 as control variable below the secondary diagonal (the effect of this is to take the input variables from the top to the side);

(B) is a column of cells with each control variable 4;

(C) and (D) are rows of cells with each control variable 2.

F is actually being evaluated as DCHxDxG. It could also be evaluated as KCxGAxH.

The size of the array above is:

$$(n - 1) \times \{n - 2 + 2(10 \cdot 2^{n-4} - n) + 1\}$$
$$= (n - 1) \times (10 \cdot 2^{n-3} - n - 1),$$

which gives the result.

The extension of Theorem 3 does not appear to be very efficient in that following the realization of a function of three variables by a 2×6 array, as few as four of the control variables (1, 2, 4, 5) are sufficient to achieve all further extensions. We suggest the following:

<u>Conjecture</u>: If $n \geq 4$, then $Fx_1 \ldots x_n$ may be expressed in the form:

$$Fx_1 \ldots x_n = G_1 G_2 H y_1 \ldots y_{n-1} G_3 v_1 \ldots v_{n-2} G_4 w_1 \ldots w_{n-2},$$

where all y_i, v_j, $w_k \in \{x_1, \ldots, x_n, 0, 1\}$, and G_1 and G_2 are elements of the set of control variables, and G_3 and G_4 are functions of n-2 variables, and H is a function of n-1 variables.

If this conjecture is true, it gives an improved situation with regard to the size of the array of cells necessary to realize a function of n variables. For small values of n we would have:

n	Theorem 3	Conjecture
3	2 x 2	2 x 2
4	2 x 6	2 x 6
5	3 x 15	3 x 14
6	4 x 34	3 x 30
7	5 x 73	4 x 64

2. Implementation.

Supposing that the octal control input to a cell is realized by an ordered triple of binary inputs (which we denote by r, s, and t) and denoting exclusive-or by a vertical stroke, the cell used in this paper is defined by:

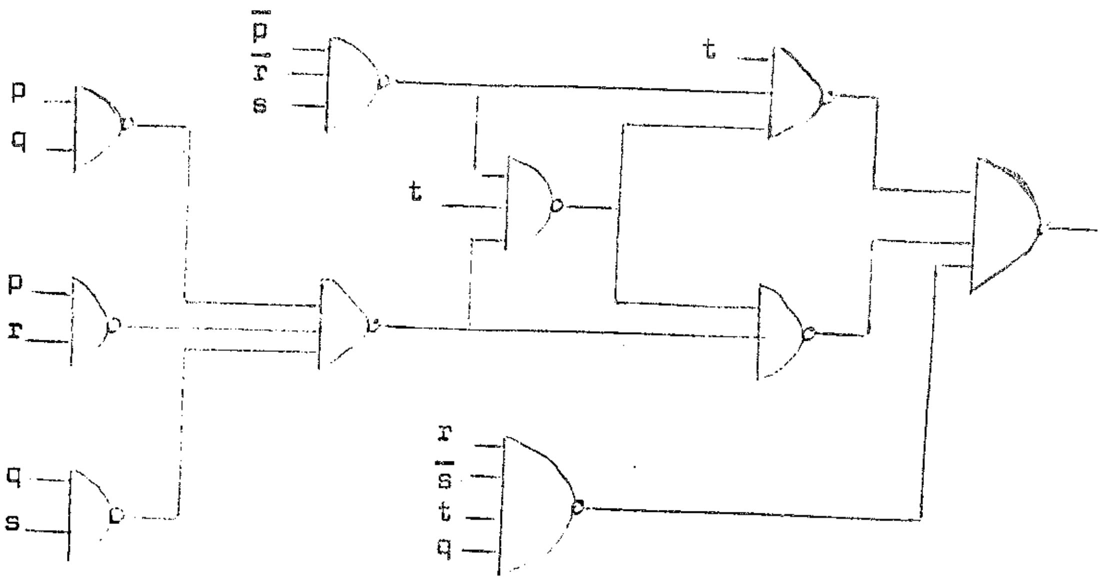
$$(pq + pr + qs) \quad t(p + r + \bar{s}) + qr\bar{s}t,$$

or, in parenthesis-free notation:

AXAKpqAKprKqsKtApArNsKqKrKNst,

where the octal numerals from the table of control variables of Section 1 are all replaced by their binary equivalents.

A possible NAND implementation of the cell consists of:



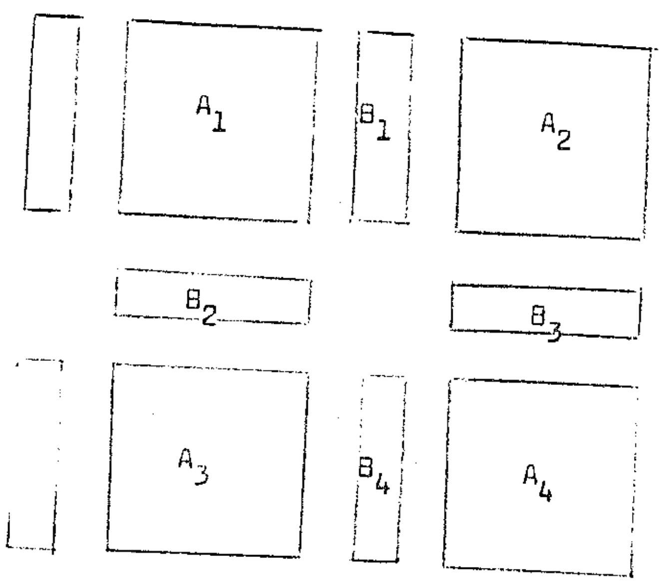
These cells might be arranged into a practical unit in the following way. The cells are combined into n x n arrays. Each pair of these arrays is separated by an n-bit register. Below the arrays are denoted A_i and the registers B_i. Each B_i is used for input and output to its neighboring arrays. The control of the B_i is such that, say for B_1:

(i) B_1 may accept output from A_1;

(ii) B_1 may be input for A_2;

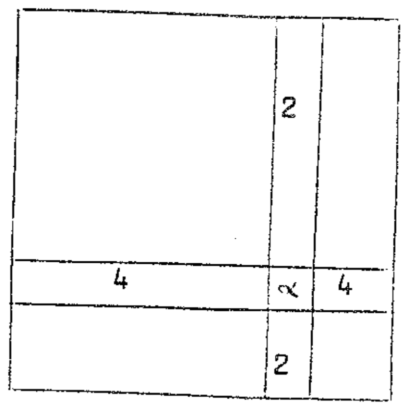
(iii) B_1 may be completely bypassed, A_1 being connected directly to A_2.

In this last mode an external supervisor may be performing input and output to B_1



The value of n actually chosen is a compromise to avoid the high cost and complexity arising from a small value for n, and to avoid inefficient use of the power of the cell arising from a large value of n. In current experimentation we are using n = 16.

In the event of a single cell failure within a block, the octal micro-program can easily reconfigure the array, resulting in the logical removal of one row and one column from the block.



Suppose that α has failed in the block shown. All the cells in the same column as α are given the control value 2. All the cells in the same row are given the control value 4. As seen, this effectively bypasses α in the same way in which Akers [1] does for his arrays. Consequently, there is not the loss of a block, but only a reduction in its useful size.

References

1. S. B. Akers, "A rectangular logic array", IEEE Trans., Vol. C-21, pp. 848-57 (1972).
2. A. Mukhopadhyay and H. S. Stone, "Cellular Logic", in Recent Advances in Switching Theory (ed. A. Mukhopadhyay), (New York: Academic Press, 1971)
3. M. A. Harrison, Introduction to Switching and Automata Theory, (New York: McGraw-Hill, 1965)