*VLSI DESIGN*
1996, Vol. 4, No. 3, pp. 257–269
Reprints available directly from the publisher
Photocopying permitted by license only



# A Methodology for Testing Arbitrary Bilateral Bit-Level Systolic Arrays

S. BANDYOPADHYAY[a], A. SENGUPTA[b] and B. B. BHATTACHARYA[c]

[a]*School of Computer Science, University of Windsor, Windsor, Ontario, N9B 3P4, Canada;* [b]*Department of Computer Science, University of South Carolina, Columbia, SC, 29208, USA;* [c]*Electronics Unit, Indian Statistical Institute, Calcutta 700 035, India.*

In this paper, we discuss the controllability and observability issues in bilateral bit-level systolic arrays. We have introduced a new concept—'$S_j$-controllability in M steps', which is somewhat analogous to the concept of C-testability and refers to the fact that all the cells in the array can be set to the state $S_j$ in at most M steps after initialization. Systolic arrays where the value of M is independent of the length of the array are characterized. Our testing procedure is based on partitioning the array into several identical subarrays which allows us to apply a repetitive pattern of tests and propagate test outcome to the observable extremities so that every cell in the array is tested by a minimum sequence of tests. Based on this concept,we have developed a set of sufficient conditions for an arbitrary bilateral bit-level systolic array to be testable for single faults.

*Keywords:* Systolic arrays, faults, testing, C-testability, controllability and observability, bit-serial structure, AND-OR graphs, algorithms.

## I. INTRODUCTION

With recent advances in parallel algorithms and their cost effective implementation with VLSI architecture, systolic arrays [1] are now being widely used in many areas, e.g., matrix multiplication, graph algorithms, signal and image processing, pattern matching to name a few. In many applications, bit- level systolic systems provide more advantage over their parallel counterparts and are preferred for (i) simplicity of their structure, (ii) low chip area requirement and (iii) suitability of CAD Implementation [4,6,7]. The well known concepts of C-testability [3]and pI-testability [15] originally evolved to provide efficient test scheme for iterative logic arrays (ILA) [13], can also be adopted for some systolic arrays because of their repetitive structure. C-testability refers to the existence of a constant number of test vectors independent of the length of the array, whereas pI-testability means that the array is partitionable into subarrays having identical test outputs. The basic idea behind an efficient test process and built-in self test is to apply a complete functional test to every cell, so that instead of testing one cell at a time, many of them are tested simultaneously.

The testability of iterative logic arrays and techniques for inducing C- testability have been studied extensively in the past [3,12]. C-testability of a generalized tree structure was presented in [8] and a testing strategy for one and two-dimensional combinational ILA have been discussed in [3,9] and C- testability of two-dimensional combinational ILA was

considered in [10]. Methods for testing unilateral systolic arrays can be found in [5], [16] and for bilateral arrays in [2,11]. Necessary and sufficient conditions for C-testability and design for testability for both unilateral and bilateral systolic arrays appeared in [14]. A set of sufficiency conditions for testing bilateral arrays of combinational cells have been described in [9], [17].

In any testing procedure, we have two phases: controllability and observability. If a systolic array is controllable, then we may apply any desired combination of inputs to any arbitrary cell. This is a significant problem in many arrays because some of the inputs to a cell are generated by neighboring cells. Thus controlling the inputs to any cell, in general, is not trivial. Once we have applied a certain set of inputs to a cell, we must be able to infer the outputs of the cell from the outputs produced by the array. If an array is observable, this is possible. In this paper, we consider arbitrary bilateral bit-level systolic arrays and discuss a new approach to controllability and observability in such systems. We have introduced a new concept of '$S_i$-controllability in M steps' and show its application to testing an array regardless of its size. We show how an arbitrary array can be decomposed into a set of identical subarrays so that testing of them can be suitably overlapped. In section II, we introduce our model of systolic array and fault model. We have discussed the test procedure in section III. We have given the sufficiency conditions for controllability in section IV. Once the conditions for controllability have been satisfied, we have to check for observability. We have discussed this problem in section V.

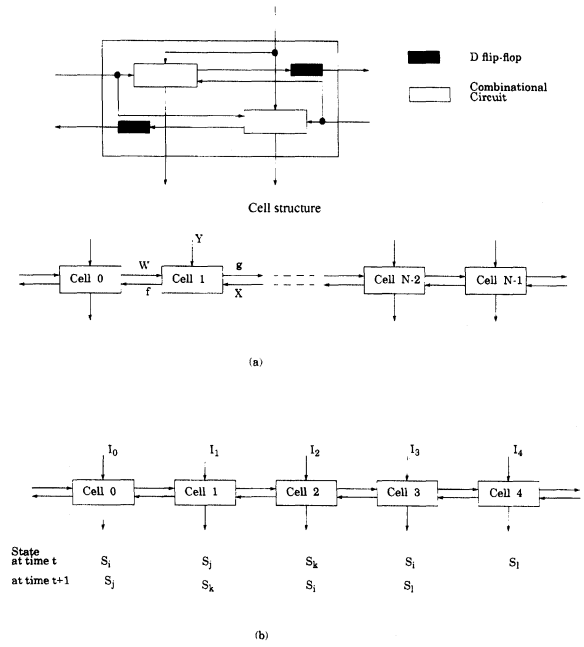



FIGURE 1 (a) A bilateral systolic array of length N (b) A bilateral systolic array with shifting of states

## II. MODEL OF BILATERAL BIT-LEVEL SYSTOLIC ARRAY

We will use a one-dimensional, bilateral, bit-level systolic array as shown in Fig. 1a. This model is similar to that in [9] and N will denote the length of the array, N being relatively large. Each cell receives multiple inputs from both the left and right neighbors and also from the top. To illustrate the key concepts, we assume that every cell has an input coming from the left, one from the right and at most one input coming from the top. Each cell has two outputs, one out of the left side and one out of the right side.

If we number the cells from the left as 0,1,2,..., N $-$ 1, then the left (right) output of the $i^{th}$ cell is the right (left) input of the $i - 1^{th}$ ($i + 1^{th}$) cell, for all i, $1 \leq i \leq N - 1$. The two cells on the two ends of the array, often play a very important role since the left (right) input to the $0^{th}$ ($N - 1^{th}$) cell is accessible to the user and hence, is directly controllable. Moreover, *only the extreme left and right outputs are directly observable*. The vertical outputs from each cell, if available, increase observability. To show how bilateral arrays may be tested, we assume that such outputs are not available. In our discussions, we have assumed that each cell has one vertical input which is however, directly controllable from outside. Generalizing the structure to accommodate more inputs (outputs) from any side is straightforward.

The output of each cell at time T + 1 is determined solely by the inputs to the cell at time T. In other words, if W, X and Y represent the inputs from the left, right and top respectively at time T, then the left output at time T + 1 is f(W,X,Y) where f is some boolean function of W, X, Y. Similarly, the right output at time T + 1 is g(W,X,Y). Such outputs may be realized by two combinational circuits, both with inputs W, X and Y and generating f(W,X,Y) and g(W,X,Y) respectively. The outputs of these combinational circuits are stored in two latches giving the required functional behavior. Due to the fact that the output of a cell at time T + 1 is completely determined by the inputs to the cell at time T, it is convenient to view a cell as having 8 states corresponding to the different input combinations of W, X and Y (i.e., 000, 001,..., 111) to the cell. We will call these states as $S_0$, $S_1$,..., $S_7$ respectively. As an example, if a cell is in state $S_5$, then the current inputs W,X and Y are 1,0 and 1 respectively. We define two sets F and G of states where F (G) consists of those states for which f (g) is 1. We will use $\overline{F}$ ($\overline{G}$) to denote sets of states for which f (g) is 0.

We assume the existence of a single clock synchronizing the operation of the array so that we currently exclude the possibility of data moving at different rates along different buses. We assume that the systolic array has a control line to globally reset all the latches of the cells in the array so that when the testing process starts, the left and right inputs to all cells, except the cells at the two ends, are 0.

### A. Fault Model

We will consider permanent logical faults and assume that no more than one cell is faulty at a time. Faults are assumed to affect only the truth table of cell functions and should not increase the number of states in the latches. Moreover, the effect of faults in a cell is combinational in nature, so that, if a cell is faulty, applying a single appropriate test vector (not a sequence of vectors) to the cell is enough to generate an output from the cell which is different from the expected output. Since the number of inputs to a cell, in a bit-level systolic array, is usually small, we adopt a functional verification approach. In other words, we will apply all input combinations exhaustively to every cell in the array and observe test outcomes at the two extremities. Testing is limited to fault detection only.

### B. Comparison with Other Existing Models

In this model, the existence of two signal streams flowing in opposite directions complicates the scenario considerably. Due to the fact that every cell has latched outputs, going in different directions, the temporal behavior of such a system is potentially very complex. The bilateral model studied in [14] assumed vertical outputs with each cell and a canonic cell design. Such a design implies that the signals moving left are *not influenced* by the right moving signals and vice-versa and thus methods used in unilateral arrays may be used. Researchers have observed that controllability and observability in bit- level systolic arrays are poor [4]. Testing of arrays with counter-flow of data is considered in [2], but no general algorithmic approach applicable to an arbitrary bilateral array is suggested. The arrays studied in [9], [17]are assumed to satisfy very restrictive cell functionality.

### C. Difficulty in Testing Bilateral Arrays

The concept of C-testability has been widely used in the past for testing unilateral arrays of combinational and sequential cells,and special type of bilateral arrays [2–5], [8–10, 14]. As observed in [2], no systolic array will ever be strictly C-testable due to delays in observing test outcome resulting from latency and initialization. In a bilateral systolic array, the test problem is inherently more complex. We will illustrate this with an example.

Consider a bilateral systolic array of arbitrary length as shown in Fig. 1b. Since the number of possible states in a cell is usually less compared to the length of the array, by pigeon-hole principle, at any instant of time, at least two cells must be in the same

state say, $S_i$ for any input pattern. To illustrate this with an example, let $S_i$,$S_j$,$S_k$, denote the states of 3 consecutive cells and suppose $S_i$ reappears at the next cell as in Fig. 1b. For efficient test scheme one needs to shift the pattern of states at the next clock pulse (see Fig. 1b), which then allows us to apply a circular pattern of inputs [2]. It is now easy to verify that this could only happen if in a cell either $f(T + 1) = X(T)$ or $g(T + 1) = Y(T)$. Thus, either the left or the right going signal should be a simple delay function and hence cannot be applicable to a general array. The example considered in [2] indeed exhibits this simple property. The two types of systolic arrays cited in [9] for FIR filtering and matrix multiplication also satisfy this restriction.

## III. TEST PROCEDURE

In order to test whether a systolic array of N cells is working properly, each cell in the array should be tested using every possible bit configuration. In other words, we must test the output of the i-th cell for each of the eight states $S_0$, $S_1$,..., $S_7$ and verify that the outputs of the i-th cell are correct, for all i, $0 \leq i \leq N - 1$.

To test whether the i-th cell of a systolic array of length N generates correct outputs when it is in state $S_j$, we need a *controllability* phase followed by an *observability* phase. In the controllability phase, we have to set the i-th cell to state $S_j$. Then, in the observability phase, we have to ensure that the left and right outputs of the i-th cell 'propagate' in the left and right directions and generate outputs at the left and right end of the array in such a way that a faulty left (right) output from the i-th cell generates an incorrect output from the left (right) end of the array. We have discussed the controllability phase, where the i-th cell is to be set to state $S_j$. Then we will briefly discuss the observability stage where the output of the i-th stage will be inferred from the outputs of the right and left extremities of the array after requisite delays for the signal to propagate from the i-th cell to the two ends of the array.

*Remark 1:* We assume that there is a global reset line which initially resets every latch in an array to 0. Therefore, depending on the vertical input to a cell, the initial state of all cells excepting the two lying at extremities (i.e., cell 0 and cell $N - 1$), can be only $S_0$ or $S_1$. Since the horizontal inputs to the boundary cells are also directly accessible, the initial state of cell 0 ($N - 1$) can be either of $S_0$,$S_1$,$S_4$,$S_5$, ($S_0$,$S_1$,$S_2$,$S_3$).

### A. Controllability in Bilateral Systolic Arrays

Given a state $S_j$ and a positive integer M, a systolic array of length N will be called $S_j$-*controllable* in M *steps* if for any designated cell, after resetting the latches of all the cells, it is possible to derive a sequence of inputs at time $t_0$,$t_1$,$t_2$,...,$t_{M-1}$ so that at time $t_M$, the designated cell is in state $S_j$ for the first time. There are systolic arrays where we can find a constant value of M (*independent of the length* of the array). Consider an array of length 5 whose cells are defined in Fig. 2a. State $S_7$ is controllable in 2 steps as shown in Fig. 2b. It is easy to verify that in such an

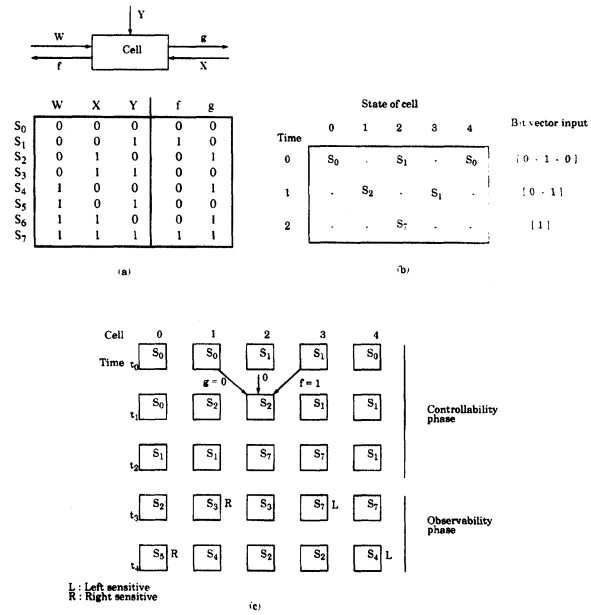



FIGURE 2 (a) Cell definition (b) Controllability of state $S_7$ (c) Observability of state $S_7$

array, M is always 2 for state $S_7$ even if the size of the array increases arbitrarily. For some arrays, a state $S_i$ may not be controllable at all. In many cases, the value of M for a given state may depend on N.

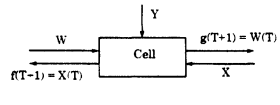
The cell definition of a systolic array which is *not* $S_7$-controllable for a *fixed value* of M is shown in Fig. 3a. Notice that the array is $S_j$-controllable for all j, $0 \leq j \leq 7$, but the number of clock pulses (inputs) necessary to set any given cell to state $S_j$ may be N, in the worst case. There are also situations where a systolic array is not $S_j$-controllable in general. However, we may set any cell to state $S_i$ only if the size of the array does not exceed a certain limit. This happens due to the influence of the boundary cells, as pointed out in Remark 1. An example is shown in Fig. 3b. All cells in such an array can be set to $S_7$ only if its length does not exceed 3.

We now explore some useful properties of an array where it is possible to set the i-th cell to state $S_j$. Theorems 1 and 2 refer to an array which is $S_j$-controllable in M steps.

THEOREM 1: *The process of setting the i-th cell to state $S_j$ involves M consecutive cells on each side of the i-th cell, thus a total of 2M + 1 cells.*

*Proof:* Follows directly from the fact that each cell derives its left (right) input from its immediate left (right) neighbor. Therefore, each step involves exactly two more cells on the two extremities.

From this theorem, we note that, in the process of setting a cell to state $S_j$ in M steps, we don't need to care for the inputs to cells which are more than M cells away from the i-th cell. Further, at time $t_k$, the number of cells involved in the process is (M − k) on each side of the i-th cell.




(a)

| | W | X | Y | f | g |
|---|---|---|---|---|---|
| $S_0$ | 0 | 0 | 0 | 0 | 0 |
| $S_1$ | 0 | 0 | 1 | 0 | 0 |
| $S_2$ | 0 | 1 | 0 | 0 | 1 |
| $S_3$ | 0 | 1 | 1 | 0 | 1 |
| $S_4$ | 1 | 0 | 0 | 1 | 1 |
| $S_5$ | 1 | 0 | 1 | 1 | 1 |
| $S_6$ | 1 | 1 | 0 | 0 | 1 |
| $S_7$ | 1 | 1 | 1 | 0 | 1 |

(b)

FIGURE 3 Cell functions of a bilateral systolic array

In other words, in an array of size 2M + 1, at time $t_0$, we need to specify inputs to cells 0,2,..., 2M. At time $t_1$, we have to specify the inputs to cells 1,3,..., 2M − 1 and so on. Thus, at time $t_k$, we only need to specify the inputs to cells, k + 2,..., 2M − k. The inputs to cells k + 1, k + 3,... are not relevant and will be indicated by a dash (−). This specification of inputs is represented by a *bit vector.*

For instance, in Fig. 2b, the bit vector inputs at time $t_0$,$t_1$,$t_2$ are [0 − 1 − 0], [0 − 1], [1].

The following corollary now describes the necessary and sufficient condition for having $S_j$-controllability in constant steps for some state $S_j$ regardless of the length (N) of an array.

COROLLARY 1: *Let a systolic array A be $S_j$-controllable in M steps. Then the value of M will be independent of N iff after initializing M + 1 alternate cells labelled i − M, (i − M + 2),..., i,..., (i + 2),..., (i + M − 2), (i + 2M), for some i ≥ M, to states $S_0$ or $S_1$, one can eventually set cell i to $S_j$* in M steps.

*Proof:* Follows directly from Theorem 1 and Remark 1.

*Remark 2:* If in a cell f = 0 = g for states $S_0$ and $S_1$, then an array of such cells will never be $S_j$-controllable for all $j \neq 0, 1$.

THEOREM 2: *In a systolic array A of length N which is $S_j$-controllable in M steps, where N ≥ 2M + 1, it is always possible to set simultaneously* $\left\lfloor \frac{N+M}{2M+1} \right\rfloor$ *cells to state $S_j$.*

*Proof:* First, we consider an array of N cells where N = r(2M + 1), for some r ≥ 1. We label them as cell 0, 1, 2,..., 2rM + r − 1. We divide the array, logically, into subarrays each of length 2M + 1. At time $t_k$, $0 \leq k \leq M - 1$, we apply the same bit vector input for time $t_k$ to each of these r logical subarrays. If we do so, cell M of every subarray will be set to state $S_j$ at time $t_M$. We therefore have succeeded in setting r cells to state $S_j$ at time $t_M$ in an array containing r(2M + 1) cells.

In the general case, the array A may not satisfy the condition N = r(2M + 1). Let us consider an array A

with N cells where N = r(2M + 1) − q for some q < 2M + 1. Suppose we take another array A1 of length r(2M + 1). When array A1 receives appropriate inputs at time $t_0,\ldots, t_{M-1}$, r cells are in state $S_j$ at time $t_M$. The N-th cell of A1 plays a role only at times $t_0$, $t_1,\ldots, t_{q-1}$. During this time, the N-th cell generates an output f which becomes input X for the (N − 1)-th cell. Let these inputs be $I_0, I_1,\ldots, I_{q-1}$. Consider a situation where the vertical inputs of cells 0, 1,..., N − 1 of A are made identical to the vertical inputs to A1 at all times $t_0, t_1,\ldots, t_M$ and the input X of the rightmost cell of A is $I_0, I_1,\ldots, I_{q-1}$ at times $t_0, t_1,\ldots, t_{q-1}$. Clearly, the states of the first N − 1 cells of A1 and the corresponding cells of A will be identical at times $t_0, t_1,\ldots, t_M$. Therefore, depending on whether or not N mod(2m + 1) < M, we will be able to get r − 1 or r cells set to state $S_j$ at time $t_M$.

THEOREM 3: *In a systolic array A of N cells, we may simultaneously set the i-th cell (for any i, 0 ≤ i ≤ 2M) of every subarray of (2M + 1) cells to state $S_j$.*

*Proof:* Consider an array A1 with N + (M − i) cells. We have seen in Theorem 2 how cell M of every subarray of A1 may be simultaneously set to state $S_j$. Let the left inputs received by cell (M − i) of A1 be $I_0, I_1,\ldots$, at times $t_0, t_1,\ldots$If we make the vertical inputs to cell m + (M − i) of array A1 and cell m of array A identical for all m, 0 ≤ m ≤ N, and feed inputs $I_0, I_1,\ldots$to the left input of A. Therefore, we have succeeded in shifting the sequence of states in A1 by (M − i) cells.

In a nutshell, our approach is to try to determine, for each state $S_0, S_1,\ldots, S_7$, a value for M. If,for state $S_j$, we fail then it is not possible to test the cells of a systolic array of an arbitrary size for state $S_j$. Otherwise, if the array is $S_j$-controllable in M steps, we obtain the bit input vectors for time $t_0, t_1,\ldots, t_{M-1}$, $t_M$. We logically divide a given systolic array of N cells into subarrays of 2M + 1 cells each and it is possible for us to set cell i of all subarrays simultaneously to state $S_j$ in M steps, for any value of i, 0 ≤ i ≤ 2M. In order to set every cell to state $S_j$, we need at most 2M + 1 separate operations.

### B. Observability in bilateral arrays

In a control phase let cell i be set to state $S_j$ at the end of the control phase. Observation phase starts from the following clock pulse. Let $t_0, t_1, t_2,\ldots, t_k,\ldots$be successive clock pulses in the observation phase. At time $t_0$, the ith cell, currently in state $S_j$ will produce an output in the left (right) direction. This is the right (left) input to the i − 1 (i + 1) th cell of the array at time $t_0$. Therefore the state of cell i − 1 (i + 1) at time $t_0$ is partially determined by the output of cell i in state $S_j$ at the end of the control phase. Let the ith cell be faulty so that the left (right) output of the ith cell, set at the end of the control phase to state $S_j$, is different from the expected value. This means that the state of cell i − 1 (i + 1) at time $t_0$ will be different from the expected state. In order to propagate the faulty output of the ith cell to the right/left end of the array, the left (right) output of cell i − 1 (i + 1) at time $t_1$ must be different from the expected value. Since the only observable outputs of the array are the left and right ends of the array, we have to propagate the incorrect left (right) output of the ith cell to the left (right) extremity of the array. Thus, in the observation phase, we must allow sufficient time for the outputs of cell i, set to $S_j$, to ultimately generate an output at the right and left extremities of the array.

Consider a cell with left input $W_k$, right input $X_k$ and vertical input $Y_k$. Let $S_k$ be the state corresponding to the input combination $W_k$, $X_k$, $Y_k$. Let $S_l$ correspond to the input combination $\overline{W}_k$, $X_k$, $Y_k$ where $\overline{W}_k$ is 0 (1) if $W_k$ is 1 (0).

A cell will be called to be in a *left sensitive state* $S_k$ if the right outputs for states $S_k$ and $S_l$ are different. We define *right sensitive state* in an identical way. Clearly, if $S_k$ is a left (right) sensitive state, $S_l$ is also a left (right) sensitive state. As an example, for the cell defined in Fig. 2a, states $S_0$,$S_5$,$S_3$ and $S_7$ are left sensitive and $S_1$,$S_3$,$S_5$ and $S_7$ are right sensitive.

In the observability phase, to propagate any change in the outputs generated by cell i, to the right (left) end of the array, we have to make sure that, at time $t_0$, the state of the i + 1 (i − 1)-th cell is in a left (right) sensitive mode. At time $t_1$, the state of the i + 2 (i − 2)-th cell is in a left (right) sensitive mode and so on, so that, at time $t_{t-1}$, the state of the i + t (i − t)-th cell is left (right) sensitive.

*Example:* Consider an array of 5 cells where each

cell is described in Fig. 2a. It is possible to apply appropriate inputs at successive clock pulses so that we get a configuration as shown in Fig. 2c. Cell 2 is set to state $S_7$ at the end of controllability phase. The observability phase is from $t_1$ to $t_2$. At time $t_3$, the outputs from the two extremities of the array can be observed to infer the correctness of cell 2.

To explain our approach to observability, we only discuss the case where the $M^{th}$ cell of each subarray of length 2M + 1 has been set to state $S_j$ at the end of the controllability phase. The general case where the kth cell is set to state $S_j$ at the end of the controllability phase requires practically identical treatment.

In order to verify that the outputs generated by cells which are set to state $S_j$, in an array of r(2M + 1) cells are correct, we divide the array into logical subarrays of size 2M + 1 each. We consider a situation where we have already set cell M of all subarrays to state $S_j$ in M steps. Ideally, it should be possible to infer the outputs f and g of the Mth cell of each subarray in the same operation. This is the case we discuss in this paper. In the cases where this is not possible, we may still be able to infer f in one control phase followed by an observation phase for f. In a separate cycle, we may again have a control phase and an observation phase for g. The procedure for having separate observation phases for f and g is quite similar and will be omitted. As discussed above, in order to ensure that any error in f (g) in cell M is propagated to the observable left (right) extremity of the array, we must ensure that, at time $t_0$, cell M − 1 (M + 1) should be in a right (left) sensitive state. Continuing in a similar way, at time $t_{M-1}$, cell 0 (2M) of each subarray should be in a right (left) sensitive state. In the following clock pulse, output f (g) of cell 0 (2M) of each subarray becomes the input X (W) of the last (first) cell of the preceding (following) subarray and so on.

It may be readily verified that such conditions may be satisfied iff the t-th cell of each subarray, $0 \leq t \leq 2M$, is in a left or right-sensitive state at time T according to the Table I, where $1 \leq i \leq M$, $1 \leq j \leq N$.

We now define a *sensitivity matrix* (SM) with 2M + 1 rows and 2M + 1 columns, which conveniently represents the sensitivity of different cells in a subarray of length 2M + 1 at various points in time. The rows and columns are numbered 0, 1,..., 2M. The (i,j)-th entry denotes the sensitivity of j-th cell at time $t_t$ denoting don't care, left, right and simultaneously left and right sensitivity respectively.

TABLE I Sensitivity of cells at different positions and time

| Time (T) | Cell Position | State* |
|---|---|---|
| i | M + i | Left sensitive |
| i | M − i | Right sensitive |
| M + j | (2M + 1 − j) mod (2M + 1) | Right sensitive |
| M + j | (j − 1) mod (2M + 1) | Left sensitive |

*Example:* For M = 2, the matrix is shown in Fig. 4a. Notice that at time $t_5$, cell 2 of every subarray of length 5 should be in left as well as in right sensitive state if we want to infer both outputs f and g simultaneously (see Figs. 2c & 4b).

We note that if an array A of N cells, where N = r(2M + 1) − q and $0 \leq q \leq 2M$, is $S_j$-controllable in M steps, then we can set one cell in every subarray of 2M + 1 cells in one control phase. For observability, we have to wait for N cycles. Thus we need M + N cycles. This process has to be repeated 2M + 1 times so that all the 2M + 1 cells of each subarray is set to state $S_j$ and the outputs are observed. Thus the total time required is O((M + N)M). One important conclusion is that, in order to minimize the total time required for testing whether all cells of the array give correct response in state $S_j$, we have to search for the minimum possible value of M.

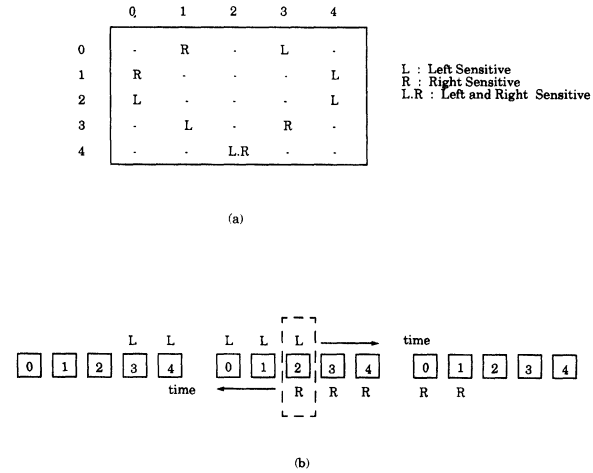



FIGURE 4 (a) Sensitivity matrix for a subarray of length 5 (b) Simultaneity of left and right sensitivity

## IV. SUFFICIENT CONDITIONS FOR CONTROLLABILITY

We have seen (in Theorem 1) that, if a systolic array is $S_j$ controllable in M steps, then, to set a specific cell to $S_j$, we need be concerned only with a subarray of length 2M + 1 cells, having M cells on either side of the specified cell. We number the cells in the subarray so that the specified cell is the cell in the middle (i.e., the $M^{th}$ cell). At time $t_0$, alternate cells of the subarray are in state $S_0$ or $S_1$. At times $t_1, t_2, \ldots, t_{M-1}$ the number of cells in the subarray, whose outputs determine $S_j$ in the $M^{th}$ cell, decreases by 2 with each clock—one at each end of the subarray. Fig. 5a depicts this situation for M = 2. (This simply shows the Fig. 2b in a different way.) We also note that at every clock pulse, only the states of alternate cells are of interest. The states of cells in which we are not interested will be represented by a dash (-). Thus, at time $t_{M-1}$, there are exact three cells in the subarray that we need be concerned with: the $(M-1)^{th}$ cell, $M^{th}$ cell and the $(M+1)^{th}$ cell (the state of the $M^{th}$ cell is of no interest).

Let state $S_j$ correspond to inputs $W_j$, $X_j$ and $Y_j$. Clearly, at time $t_M$, the input from the top to the $M^{th}$ cell must be $Y_j$. Let the states of the $(M-1)^{th}$ cell and the $(M+1)^{th}$ cell at time $t_{M-1}$ be $S_k$ and $S_m$ respectively. In order to generate $W_j$ at time $t_{M-1}$ as its right output, $S_k$ must be in $G$ ($\overline{G}$) if $W_j$ is 1 (0).

We will use the term $G_j$ to stand for $G$ ($\overline{G}$) if $W_j$ is 1 (0). We will use the term $F_j$ in a similar way so that $F_j$ stands for $F$ ($\overline{F}$) if $X_j$ is 1 (0). Thus $S_k \in G_j$ and $S_m \in F_j$.

A useful way to look at the notion of $S_j$ controllability is as follows. If $S_j$ is $S_0$ or $S_1$, then an array is $S_j$ controllable in 0 steps. Otherwise, a necessary condition for $S_j$ controllability is to have two states $S_k$ and $S_m$ such that $S_k \in G_j$ and $S_m \in F_j$ where the array is $S_k$ controllable in M − 1 steps and $S_m$ controllable in M − 1 steps (Fig. 5b).

The process of $S_j$ controllability may be depicted by an isosceles triangle with the vertex at the top representing the i-th cell in state $S_j$ at time $t_M$ (refer to Fig. 6). The base is of length 2M + 1 and is internally subdivided into 2M segments(segmentation is shown by short vertical lines dividing the base) so

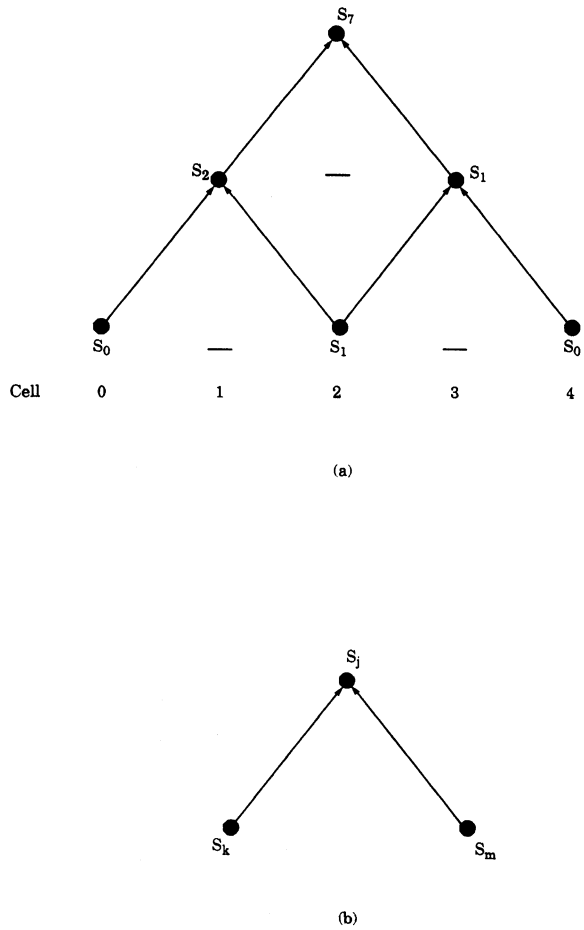



FIGURE 5 (a) Controllability of state $S_7$ (a different view of fig. 2b) (b)Controlling $S_j$

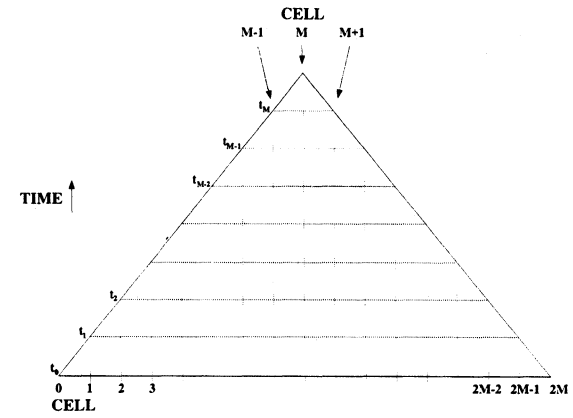



FIGURE 6 Representing controllability by an isosceles triangle

that each segment is of unit length. Every segment may be visualized as having a cell at each end. The base represents the states of all the 2M + 1 cells of the subarray at time $t_0$, where alternate cells of the array are either instate $S_0$ or in state $S_1$. Inside the triangle we visualise M − 1 equispaced lines, parallel to the base, of length 2M − 2, 2M − 4,..., 4, 2 (shown by dashed lines). Each of these lines is internally subdivided into segments of unit length each. The dashed line closest to the base represents the states of cells, at time $t_1$, in the subarray of 2M − 1 cells and so on, so that the dashed line closest to the vertex at the top represents states of 3 cells of interest at time $t_{M-1}$. That is, it depicts the situation that cells M − 1 and M + 1 are in states $S_k$ and $S_m$ respectively (the state of cell M is of no interest and will be represented by a —). Thus in general, at time $t_i$, we are only interested in alternate cells of a subarray whose length is 2(M − i) + 1, for all i, $0 \leq i \leq M$.

Once we determine that the array is $S_j$ controllable in M steps, and know the inputs to the subarray at times $T_0$, $T_1$,..., $T_M$, drawing the triangle is straightforward. Conversely, if we can generate the triangle, determining the inputs to the subarray at times $T_0$, $T_1$,..., $T_M$, is trivial. We will term such triangles as the *controllability triangle of height M*, rooted at $S_j$. The states of cell M − 1 (M + 1) i.e., $S_k(S_m)$ at time $t_{M-1}$ will be termed a *left (right) requirement for* $S_j$. For example, $S_2$ is a left requirement for $S_7$ in cells defined in Fig. 2a.

If an array is $S_j$ controllable for some M, the controllability triangle for $S_j$ may be recursively defined as follows :

i) if j = 0 or j = 1: the controllability triangle consists of the vertex $S_j$ alone.
ii) if j > 1: the controllability triangle consists of the vertex $S_j$ connected to two controllability triangles, each of height M − 1, rooted at $S_k$ and $S_m$ respectively Fig. 5b).

It is important to note that the controllability triangles rooted at $S_k$ and $S_m$ overlap unless both $S_k$ and $S_m \in \{S_0, S_1\}$. Also we note that whenever there is an overlap, there must exist a state $S_n$ *such that* $S_n$ is simultaneously a right requirement of $S_k$ and a left requirement of $S_m$. Thus the controllability triangle rooted at $S_n$ represents the portion of overlap between the triangles rooted at $S_k$ and $S_m$ Fig. 7). Let us consider a subarray A of length 2M + 1 where the successive cells from left to right are designated as $A_0$, $A_1$,..., $A_{2M}$. Our problem is to set the cell $A_M$ to $S_j$. The controllability triangle rooted at $S_k$ has a subarray A1, of length 2M − 1, as its base such that the leftmost cell of A1 is $A_0$. The controllability triangle rooted $S_m$ has a subarray A2, of length 2M − 1, as its base such that the rightmost cell of A2 is $A_{2M}$. The shaded portion represents the cells of A that are both in A1 and A2. These cells must have the same state in both A1 and A2.

In this section we will identify certain situations for controllability which may be tested readily. Since we are trying to develop sufficiency conditions, we will test whether all the states maybe controlled this way. The concept of controllability triangle presented above is helpful in characterizing such situations.

We will define the sufficiency conditions for $S_j$ controllability using Horn clauses [18]. Each clause is an expression in the following form:

*conclusion* ← *condition*$_0$, *condition*$_1$,..., *condition*$_{n-1}$

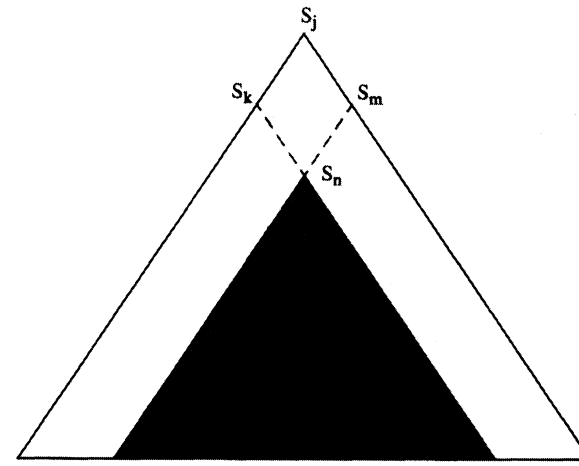



FIGURE 7 Controllability triangle when $j > l$, $S_k$ and/or $S_m \notin \{S_0, S_1\}$

The *conclusion* is true if *condition*$_i$ is true, for all i, $0 \leq i < n$. The *conclusion* as well as each of the *condition* must be atomic formulae of the form $P(t_0, t_1,\ldots, t_{M-1})$, where P is a predicate name and $t_k$ is called the k-th term. A term can be an arithmetic expression, a variable or a constant. A term can be a variable and we will use the convention that all terms with a name starting with an upper case alphabet stands for a variable. We will use a to stand for a term whose value is immaterial. All other terms are constants.

In the description below, we use two classes of predicates as discussed below:

Class (i) Predicates with two terms. We have two such predicates—*left_requirement* and *right_requirement*. The formula *left_requirement(A, B)(right_requirement(A, B))* is true if A and B represent two states where B is a left (right) requirement for A.

Class (ii) Predicates with four terms. We have five such predicates—*controllable*, *identical*, *left_different*, *right_different* and *left_right_different*. Each of these predicates has the form *predicate_name*($S_j$, *M*, *Left*, *Right*). If the atomic formula *controllable*($S_j$, *M*, *Left*, *Right*) is true, then it is possible to get the middle cell (i.e., the $M^{th}$ cell of a subarray of 2M + 1 cells) in state $S_j$ in M steps (note that we count M from 0). In this process, at the $(M - 1)^{th}$ step, cell M − 1 (M + 1) of the subarray will be in state *Left* (*Right*).

The Horn clauses for the sufficiency conditions for controllability are presented in the Table II. The proof of their correctness are presented in the following lemmas and Theorem 4.

LEMMA 1: *If identical($S_j$, M, Left, Right) is true for some $S_j$, M, Left, Right, then it is always possible to set the Mth cell to $S_j$ in M steps. In this process, at time $t_t$, all cells of interest in the subarray of 2(M − k) + 1 cells are in the same state, for all k, $0 \leq k \leq M - 1$.*

*Proof:* We will prove the lemma by induction. The lemma is trivially true for M = 0. Construction of the controllability triangle for $S_0$ *or for* $S_1$ is trivial because all the cells can be initially set to $S_0$ or $S_1$. Assume the lemma is true for some state *X* and for some *P* steps, that is, *identical(X, P, Left, Right)* is true for some *X*, *P*, *Left*, *Right* and it is possible to set the *P*th cell to *X* in *P* steps. In this process, at time $t_t$, all cells of interest in the subarray of 2(*P* − k) + 1 cells are in the same state $\sigma_k$ (say), for all k, $0 \leq k \leq P - 1$. Consider a subarray of 2*P* + 3 cells such that

TABLE II Horn Clauses for Sufficiency Conditions

| | | |
|---|---|---|
| *controllable($S_j$, M, Left, Right)* | ← | *identical($S_j$, M, Left, Right).* |
| *controllable($S_j$, M, Left, Right)* | ← | *left_different($S_j$, M, Left, Right).* |
| *controllable($S_j$, M, Left, Right)* | ← | *right_different($S_j$, M, Left, Right).* |
| *controllable($S_j$, M, Left, Right)* | ← | *left_right_different($S_j$, M, Left, Right).* |
| *identical($S_0$, 0, _, _).* | | |
| *identical($S_1$, 0, _, _).* | | |
| *identical($S_j$, M, X, X)* | ← | *left_requirement($S_j$, X), right_requirement($S_j$, X), identical(X, M − 1, Y, Y).* |
| *left_different($S_j$, M, Left, Right)* | ← | *identical($S_j$, M, Left, Right).* |
| *left_different($S_j$, M, Left, Right)* | ← | *left_requirement($S_j$, Left), right_requirement($S_j$, Right), left_different(Left, M − 1, Leftnew, Middle), identical(Right, M − 1, Middle, Rightnew).* |
| *right_different($S_j$, M, Left, Right)* | ← | *identical($S_j$, M, Left, Right).* |
| *right_different($S_j$, M, Left, Right)* | ← | *left_requirement($S_j$, Left), right_requirement($S_j$, Right), identical(Left, M − 1, Leftnew, Middle), right_requirement(Right, M − 1, Middle, Rightnew).* |
| *left_right_different($S_j$, M, Left, Right)* | ← | *left_requirement($S_j$, Left), right_requirement($S_j$, Right), left_different(Left, M − 1, Leftnew, Middle), right_different(Right, M − 1, Middle, Rightnew).* |

at time $t_r$, $0 \leq r \leq P - 1$, the states of all cells of interest are in the same state $\sigma_k$. From our assumption, it is simple to see that the states of the $P$-th and the $P + 2$-th cell are both $X$. Evidently, if both *left_requirement($S_j$, X)* and *right_requirement($S_j$, X)* are true, then the $P + 1$-th cell can be set to the state $S_j$ in $P + 1$ steps. From the rule,if both *left_requirement($S_j$, X)*, *right_requirement($S_j$, X)* and *identical(X, P, Left, Right)* are true, then *identical($S_j$, P + 1, Left, Right)* is true. Thus the Lemma follows by standard induction argument.

LEMMA 2: *If left_different($S_j$, M, Left, Right) is true, for some $S_j$, M, Left, Right, then it is always possible to set the Mth cell to $S_j$ in M steps. In this process, at time $t_k$, all cells of interest in the subarray of 2(M − k) + 1 cells, except for the leftmost cell, are in the same state, for all k, $0 \leq k \leq M - 1$.*

*Proof:* If the predicate *left_different($S_j$, M, Left,* Right) becomes true by the first rule, then the Lemma follows directly from Lemma 1. If the predicate *left_different($S_j$, M, Left, Right)* becomes true by the second rule, then we prove the Lemma by induction. For M = 0, the Lemma is trivially true, because *left_different($S_j$, M, Left, Right)* can be true only by the first rule. Assume the Lemma is true for some $M = P$. Hence there is a state $X$ which is controllable in $P$ steps. We visualize a subarray A1 generating the state $X$ in cell $A_P$ at time $t_P$. At time $t_r$, the states of all the cells of interest in A1, except the leftmost cell, must be in the same state $\sigma_r$ (say) for all $r$, $0 \leq r \leq P - 1$. The subarray A2 (whose leftmost $2P - 1$ cells are the rightmost $2P - 1$ cells of A1) generates a state $Y$ in the cell $A_{P+2}$. Since $Y$ satisfies the rule for the predicate *identical*, at time $t_r$, all cells of A2 are in the same state $\lambda_r$ (say). From the rule, because *Middle* appears in both the predicate *identical* and *left_different*, $\sigma_r$ and $\lambda_r$ must be the same for all $r$. Evidently if $X$ is a *left_requirement* of $S_j$ and $Y$ is a *right_requirement* of $S_j$ then, the controllability triangle for $S_j$ can be formed with all the cells of interest being set to $\sigma_r$, excepting the leftmost cell, at time $t_r \leq r \leq P - 1$. At time $t_P$, the cells $A_P$ and $A_{P+2}$ will be at states $X$ and $Y$ respectively which can set the cell $A_{P+1}$ in state $S_j$. Hence the Lemma follows by induction.

LEMMA 3: *If right_different($S_j$, M, Left, Right) is true, for some $S_j$, M, Left,* Right, then it is always possible to set the *Mth cell to $S_j$ in M steps. In this process, at time $t_k$, all cells of interest in the subarray of 2(M + k) + 1 cells, except for the rightmost cell, are in the same state, for all k, $0 \leq k \leq M$.*

*Proof:* The proof is similar to that of Lemma 2.

LEMMA 4: *If left_right_different($S_j$, M, Left, Right) is true, for some $S_j$, M, Left,* Right, then it is always possible to set the *Mth cell to $S_j$ in M steps. In this process, at time $t_k$, all cells of interest in the subarray of 2(M − k) + 1 cells, except for the leftmost cell and the rightmost, are in the same state, for all k, $0 \leq k \leq M$.*

*Proof:* The proof is similar to that of Lemma 2. Note that the leftmost cell of the subarray A1 and the rightmost cell of the subarray A2 do not appear in the region of common cells in the subarray and hence there is no conflict if these cells are in states different from the others.

From the above Lemmas and the Horn clauses given in the Table II, the controllability of a bilateral systolic array is given by the following Theorem.

THEOREM 4: *Given an array, if the predicate controllable ($S_j$, M, Left, Right) as given by the clauses in Table II is true then the array is $S_j$-controllable in M steps.*

*Remark 3:* The set of sufficient condition described above ensures that a state $S_j$ is controllable in constant (M) steps regardless of the length of the array in contrast to those in [9] and [17] which do not guarantee controllability in constant time. However, our model assumes a global reset line whereas those in [9] and [17] do not.

*Example:* Consider an array composed of cells defined as in Fig. 2a. It might be readily verified that the following predicates are true, implying that all the states are controllable in at most two steps.

*identical*($S_0$, *0*, _, _)
*identical*($S_2$, *1*, $S_1$, $S_1$)
*identical*($S_4$, *2*, $S_2$, $S_2$)
*right_different*($S_0$, *1*, $S_1$, $S_0$)
*right_different*($S_6$, *1*, $S_2$, $S_1$)

*identical*($S_0$, *0*, _, _)
*identical*($S_3$, *1*, $S_1$, $S_1$)
*identical*($S_5$, *2*, $S_2$, $S_2$)
*right_different*($S_1$, *1*, $S_1$, $S_0$)
*right_different*($S_7$, *1*, $S_2$, $S_1$)

## V. ALGORITHM FOR CHECKING OBSERVABILITY

In this section, we describe an algorithm to check the observability of an array when the controllability criteria have been satisfied. In the process, we know the sequence of states and inputs preceding the appearance of $S_j$ in the array. The algorithm determines whether the outputs of cells which are set to state $S_j$ might be observed. If the algorithm is successful, then the array is controllable and observable so far as state $S_j$ is concerned.

We will now describe the algorithm for checking observability informally. Given a systolic array of length N, let us assume that at time $t_0$, we have succeeded in setting the M-th cell of every subarray of length 2M + 1 to state $S_j$ at the end of the controllability phase. Suppose our objective is to infer both the outputs f and g of cell M in each subarray in a single operation. To check whether this is possible, we construct a directed graph. Each node of this graph is represented by a vector of length 2M + 1, where the i-th component of the vector represents the state of the i-th cell of the subarray at time $t_p$. Two nodes $n_1$ and $n_2$ represented respectively by the vectors $[S_0, S_1,\ldots, S_{2M}]$ and $[S'_0\ S'_1,\ldots,S'_{2M}]$ will have a directed edge from $n_1$ to $n_2$ if for all i, $0 \leq i \leq 2M$, $S_i$ represents the state of cell i at time $t_p$ and $S'_i$ denotes the state of the same cell at time $t_{p+1}$ satisfying the sensitivity criterion as specified in the (p mod (2M + 1),i)-th entry in the sensitivity matrix SM.

The algorithm starts by creating the graph with several nodes which correspond to various valid patterns of states in cells 0,1...,2M at the end of controllability phase. Then it adds more vertices and directed edges at successive times in accordance to the above rule. The algorithm terminates whenever a directed cycle is found or when no further extension of the graph is possible. It is simple to see that whenever a directed cycle is found, the array is observable so far as the response of the state $S_j$ is concerned. On the other hand, if no further extension of the graph is possible, then an array of arbitrary length is not observable. In both cases, however, the algorithm terminates in finite time. If we want to infer f and g in isolation, the algorithm might be simply modified in a straight forward manner.

## VI. CONCLUSION

In this paper, we explore many interesting features of controllability and observability and present a new set of sufficient conditions for controllability and observability in an arbitrary bilateral bit-level systolic array. Though such arrays, in general, are difficult to test, the concept of $S_j$-controllability in M steps can be exploited to generate a test scheme for some of them regardless of the length of the array. A subsequent open problem is to propose a testable design for an arbitrary array so that all states become controllable inconstant steps, and observable. Extension of these techniques to 2-dimensional arrays will be presented in a future paper.

### *References*

[1] W.R. Moore, A.P.H. McCabe and R.B. Urquhart(Ed.), Systolic Arrays, *Adam Hilger*, Bristol, 1987.
[2] W.P. Marnane, W.R. Moore, H.M. Yassine, E. Gautrin, N. Burgess and A.P.H. McCabe, "Testing bit-level systolic arrays", *Proc. 1987 Int. Test Conf.*, pp. 906–914, Sept. 1987.
[3] T. Sridhar and J.P. Hayes, "Design of easily testable bit-sliced systems", *IEEE Trans. Comput.*, Vol. C-30, pp. 842–854, Nov. 1981.
[4] J.T. Scanlon and W.K. Fuchs, "A testing strategy for bit-serial arrays", *Digest of Technical Papers, Int. Conf. on Computer-Aided Design*, pp. 284–287, Santa Clara, Nov. 1986.
[5] W.R. Moore and V. Bawa, "Testability of a VLSI systolic array", *Proceedings, ESSCIRC-85*, pp. 271–276, France, Sept. 1985.
[6] N. Kanopoulos, "A bit-serial architecture for digital signal processing", *IEEE Trans. on Circuits and Systems*, Vol. CAS-32, pp. 289"291, March 1985.
[7] T.A. Davis, R.P. Kunda and W.K. Fuchs, "Testing of bit-serial multipliers", *Proc. IEEE Int. Conf. on Computer Design(ICCD)*, pp. 430–434, Oct. 1985.

[8] A. Chatterjee and J.A. Abraham, "C-testability for generalized tree structures with applications to Wallace trees and other circuits", *Digest of Technical Papers, Int. Conf. on Computer-Aided Design*, pp. 288–291, Santa Clara, Nov. 1986.
[9] A. Vergis and K. Steiglitz, "Testability conditions for bilateral arrays of combinational", *IEEE Trans. on Comput.*, Vol. C-35, pp. 13–22, Jan. 1986.
[10] H. Elhuni, A. Vergis and L. Kinney, "C-testability of two-dimensional iterative arrays", *IEEE Trans. on Computer-aided Design*, Vol. CAD-5, pp. 573–581, Oct. 1986.
[11] H. Elhuni and A. Vergis, "STV Testability, an approach to testing bilateral systolic arrays", *Proc. 24th Annual Allerton Conf.*, pp. 925–933, Oct. 1986.
[12] F.J.O. Dias, "Truth table verification of an iterative logic array", *IEEE Trans. on Comput.*, Vol. C-25, pp. 605–613, June 1976.
[13] F.C. Hennie, Finite State Models for Logical Machines, *John Wiley and Sons, Inc.*, 1968.
[14] S. Rawat and M.J. Irwin, "C-testability of unilateral and bilateral sequential arrays", *Proc. 1987 Int. Test Conf.*, pp. 181–188, Sept. 1987.
[15] E. Cerny and E.M. Aboulhamid, "Built-in testing of pI-testable iterative arrays", *Digest of Papers, FTCS-13*, Milano, Italy, 1983.
[16] R. Parthasarathy and S. M. Reddy, "A Testable design of iterative logic arrays", *IEEE Trans. on Comput.*, Vol. C-30, pp. 833–841, Nov. 1981.
[17] F. G. Gray and R. A. Thompson, "Fault detection in bilateral arrays of combinational cells" *IEEE Trans. on Comput.*, Vol. C-27, pp. 1206–1213, Dec. 1978.
[18] J. W. Floyd, Foundations of Logic Programming, *Springer-Verlag*, New York, 1984.

## Authors' Biographies

Subir Bandyopadhyay received the B.Tech., M. Tech. and Ph.D. degrees in Radio Physics and Electronics from the University of Calcutta in 1968, 1969 and 1975 respectively and the M.Math. degree from the University of Waterloo in 1975. Until 1982, he served as a Lecturer at the University of Calcutta, India and as an Associate Professor at the Indian Statistical Institute, Calcutta, India. He spent one year as a visiting fellow at the Queen's University of Belfast and another year as an Associate Professor at the University of Lethbridge, Alberta. At present, he is a Professor in the Computer Science Department at the University of Windsor, Ontario. His research interest includes System Diagnosis, Fault Tolerant Interconnection Network, VLSI System Architectures and Residue Number Systems.

Abhijit Sengupta received the B.Tech., M.Tech. and Ph.D. degrees in Radio Physics and Electronics from the University of Calcutta in 1970, 1971 and 1976 respectively. Until 1982, he was an Associate Professor in the Computer Science Unit at the Indian Statistical Institute, Calcutta, India. From 1982 to 1983, he served as a Visiting Professor in the Computer Science Department at the Western Illinois University. In 1983, he joined the Computer Science Department at the University of South Carolina, where currently he is a Professor. His research interest includes Fault Tolerant Computing, Parallel Architecture and Interconnection Networks, VLSI system architectures and Parallel Algorithms.

(The biography of B. B. Bhattacharya was not available)