

# Ordering Storage Elements in a Single Scan Chain\*

Rajesh Gupta

IBM East Fishkill  
Zip 3A1/306, Route 52  
Hopewell Junction, NY 12533

Melvin A. Breuer

Electrical Engineering-Systems  
University of Southern California  
Los Angeles, CA 90089

## Abstract

In serial scan designs, particularly those tested in a partitioned manner, the circuit test time is influenced by the ordering of the storage elements in the scan chain. This paper describes a procedure for constructing a single serial scan chain with the objective of minimizing the overall test time. It uses a polynomial-time algorithm which results in an ordering of the storage elements along with an indication of the degree of optimality of the solution.

## 1 Introduction

Scan design [1, 2] can greatly reduce the cost of test pattern generation for general sequential circuits. However, it constrains test data (both input patterns and output results) to be shifted in and out of the circuit in a serial manner, typically through a single *scan chain* formed by the circuit storage elements. The high test time caused by the need for serial shifting of test data can be reduced by ordering the storage elements in the scan chain such that those that need to be accessed more frequently are closer to the scan-in/scan-out pins. This reduces the average time for applying tests to the circuit.

This paper describes the approach to single scan chain ordering used in the SIESTA design-for-testability methodology [3]. The problem of building a single scan chain can be stated as follows: given a circuit, the test length (number of test vectors) for each circuit partition, and the set of scan flip-flops (FFs), determine the ordering of the FFs in the scan chain such that the test time for the resulting design is minimized. The impact of routing considerations on the ordering of FFs is ignored in this discussion.

## 2 Background

In this paper we use the term *scan design* to mean *full scan design*; i.e., all storage elements in the circuit under consideration are assumed to be made scannable and included in the scan chain. The single scan chain design approach presented here is also applicable to certain *partial scan designs* [4].

\*This work was supported in part by the Defense Advanced Research Projects Agency and monitored by the Office of Naval Research under Contract No. N00014-87-K-0861, and in part by the Semiconductor Research Corporation under Contract No. 88-DP-075.

## 2.1 Circuit Model

In a full scan design of a synchronous sequential circuit, the scan chain essentially makes the combinational portions of the circuit fully accessible in test mode. This combinational logic can be subdivided into kernels, each essentially being a *maximal region of connected combinational logic* (sometimes known as a *cloud*). All inputs/outputs of a given kernel are connected to either circuit primary I/O or to scan FFs. Thus any scan design can be modeled as a set of disjoint combinational kernels, each having a subset of scan FFs feeding its inputs and another subset of scan FFs reading its outputs. Note that by the definition of kernels above, a scan FF may drive no more than one kernel and receive outputs from no more than one kernel. Because the kernels are disjoint, they can be tested independently.

## 2.2 Overlapped Test Scheme

The problem of ordering the scan FFs for minimum test time is dependent on the manner in which tests are applied to a circuit. We assume that the *overlapped test scheme* [2, Section 9.9.1] is used. This scheme is described briefly below.

Figure 1 shows a circuit example with kernels *A*, *B* and *C*, with test lengths indicated. From the definition of kernels it follows that test vectors for *A*, *B* and *C* could be applied at the same time simply by merging them and shifting the appropriate bit vector into the scan chain. For simplicity assume that each of the five scan registers  $R_1, \dots, R_5$  consists of one FF.

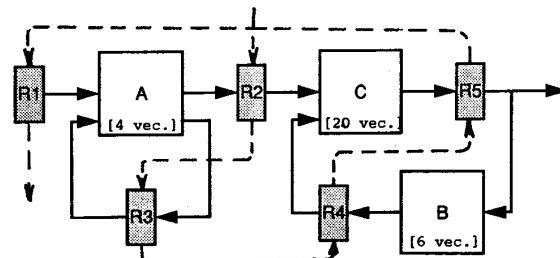


Figure 1: Example of single scan chain.

In the overlapped test scheme [2], the test sets are applied to the set of kernels in a number of *test sessions* equal to the number of kernels, as summarized in Table 1. In each session, vectors are applied to those

kernels currently under test until the test set for some kernel is exhausted.

Session	Kernels tested	Vectors	Drivers	Receivers
1	A, B, C	4	R1... R5	R2... R5
2	B, C	2	R2, R4, R5	R4, R5
3	C	14	R2, R4	R5

Table 1: Summary of test sessions.

In a given session, the registers that supply test data to/receive test results from the kernels currently under test are called **drivers/receivers**, respectively, for that session (see Table 1). To minimize the duration of the session, each vector is shifted in, while the previous test result is being shifted out, using the minimum possible number of clock cycles needed to access the drivers/receivers. This number of clock cycles is known as the **chain cycle** for the current session. For the scan chain ordering shown in Figure 1, it is easily verified (by referring to Table 1) that the chain cycle values for the three sessions are 5, 4 and 3, respectively. (See Table 2, first row.)

The duration of each test session  $i$ , in clock cycles, is equal to  $n_i(cc_i + 1)$ , where  $n_i$  is the number of vectors applied,  $cc_i$  is the chain cycle in session  $i$ , and the “+1” term accounts for loading test results into the scan chain. In addition, a certain number of clock cycles are taken up in flushing out the last test result at the end of the test, and also in synchronizing between test sessions with different chain cycles; the total number of such clock cycles equals the length of the chain. Thus the overall test time for the circuit is  $4(5 + 1) + 2(4 + 1) + 14(3 + 1) + 5 = 95$  clock cycles.

### 2.3 Scan Chain Ordering

When the overlapped test scheme is used, the test time depends on the ordering of the FFs in the scan chain. Table 2 shows the test time for different orderings. The variation in test time illustrates the significance of the scan chain design problem: to determine an ordering of scan FFs that leads to minimum test time. For a circuit with  $n$  FFs, the number of possible orderings is  $(n!)$ , which increases roughly exponentially with  $n$ . Thus an efficient way of searching for a desirable ordering is required.

Ordering	Chain cycles	Test time
(R2, R3, R4, R5, R1)	(5, 4, 3)	95
(R5, R4, R3, R2, R1)	(5, 5, 5)	125
(R1, R2, R3, R4, R5)	(5, 5, 4)	111

Table 2: Test time in clock cycles for different scan chain orderings.

Bhawmik [5] proposes a heuristic method for ordering scan registers which is based on counting the number of “roles” played by each scan register, i.e., the number of kernels that are tested using the register. Registers playing more “roles” are placed nearer to the ends of the scan chain. However, this technique does not take into account the relative test lengths of the kernels; this factor can have a significant impact on the “goodness” of a particular ordering.

In Section 3 we derive upper and lower bounds on the test time, which will be useful in the search for optimal orderings in Section 4.

## 3 Bounds on Test Time

Since the maximum chain cycle in any session is equal to the number of scan FFs, an upper bound on test time is  $TL_{max}(N+1) + N$ , where  $TL_{max}$  is the highest test length among the kernels and  $N$  is the total number of scan FFs. The second row of Table 2 illustrates a design that meets the upper bound.

Let us now consider the other end of the spectrum, namely orderings that lead to the lowest test time under the overlapped test scheme. The overall test time consists of the sum of the individual session test times. In each session a fixed number of vectors is applied to a fixed set of kernels. Only the *chain cycle* depends on the scan chain ordering. In general to minimize the overall test time, the chain cycles of the individual sessions must be minimized. However, an ordering that minimizes the chain cycle in one session need not minimize the chain cycle in the other sessions. We will derive a lower bound on the test time by determining the lowest possible duration of *each session separately*, and summing these values.

For the circuit of Figure 1, referring to Table 1, all five FFs serve as drivers during the first session. Hence irrespective of the scan chain ordering, a chain cycle of 5 will be required to supply patterns to all the drivers. This is called the **minimum session cycle (MSC)** of the first session. Thus the time needed to apply the 4 patterns in the first session is always  $4 \times (5 + 1) = 24$  clock cycles.

In the second session, there is one “pure driver”, R2, and two “driver-receivers”, R4 and R5. It can easily be seen that the MSC is 4; this is achieved if R2 is placed near the scan-in pin, and R4, R5 are placed near the middle of the chain (i.e., within the middle three positions). The corresponding session duration is  $2 \times (4 + 1) = 10$ .

In the third session, there are two “pure drivers” and one “pure receiver”. Hence the MSC is 2, which is achieved by placing the two drivers in the first two positions near the scan-in pin, and placing the receiver in one of the last two positions near the scan-out pin. This results in a session duration of  $14 \times (2 + 1) = 42$ .

In general, in session  $i$ , the minimum session cycle  $MSC_i$  can be computed as follows. Let  $d_i$ ,  $r_i$ , and  $c_i$  be the number of pure drivers, pure receivers, and driver-receivers, respectively. Then

$$MSC_i = \begin{cases} \max(d_i, r_i) & \text{if } c_i = 0; \\ \max(d_i, r_i, \lceil \frac{N-c_i}{2} \rceil) + c_i & \text{otherwise.} \end{cases}$$

It can be verified that for our example,  $MSC_1 = 5$ ,  $MSC_2 = 4$ , and  $MSC_3 = 2$ .

Note that for each session its MSC can be achieved by one or more scan chain orderings. However, there may or may not be any ordering that simultaneously achieves all session MSC’s. In any case, the sum of the session durations yields a lower bound on the overall test time. In our example this value is  $24 + 10 + 42 + 5 = 81$  clock cycles.

In the following section a method will be described for checking whether or not the lower bound can be achieved by some ordering.

## 4 Single Chain Algorithm

If a chain ordering could be found that simultaneously minimized each session test time to its lower bound, then the resulting overall test time would meet the overall lower bound, and the ordering would be optimal. The approach taken in our algorithm is to characterize the set of orderings that locally optimizes each individual session. Then, if the intersection of these sets of orderings is nonempty, any ordering in the intersection is an optimal ordering.

In the following discussion the scan chain is treated as consisting of  $N$  slots. Each slot is essentially a location in which a scan FF may be placed. The slots are numbered 1 through  $N$ , in order from the scan-in pin to the scan-out pin. The objective of scan chain ordering is to determine a mapping of scan FFs to slots.

### 4.1 Flip-Flop Position Ranges

Consider the example circuit of Figure 1. As we have derived in the previous section, for the first test session  $MSC_1$  is 5, i.e., five clock cycles are used for shifting in each test vector. Since there are exactly 5 scan FFs  $R_1 \dots R_5$ , they may be placed in any order. We characterize this by the statement

$$\forall Ri, Range_1(Ri) = [1, 5],$$

where  $[a, b]$  denotes the set of consecutive integers from  $a$  to  $b$ , both included, and  $Range_1(Ri)$  represents the range of slots in which the FF  $Ri$  must be placed to ensure that session 1 has the minimum chain cycle.

In the second test session  $MSC_2 = 4$ . This implies that all driver FFs must lie within the first 4 slots  $[1, 4]$  and all receiver FFs must lie within the last 4 slots  $[2, 5]$ . This translates to

$$Range_2(R2) = [1, 4];$$

$$Range_2(R4) = Range_2(R5) = [1, 4] \cap [2, 5] = [2, 4].$$

Note that because  $R4$  and  $R5$  are driver-receivers, they must satisfy the conditions on both drivers and receivers. FFs not listed are idle in this session and have range  $[1, 5]$ .

Finally, in the third test session  $MSC_3 = 2$ . This corresponds to

$$Range_3(R2) = Range_3(R4) = [1, 2];$$

$$Range_3(R5) = [4, 5].$$

The ranges for the various scan FFs over the three test sessions are summarized in Table 3.

FF	Session 1	Session 2	Session 3	Ideal range
$R_1$	[1, 5]	[1, 5]	[1, 5]	[1, 5]
$R_2$	[1, 5]	[1, 4]	[1, 2]	[1, 2]
$R_3$	[1, 5]	[1, 5]	[1, 5]	[1, 5]
$R_4$	[1, 5]	[2, 4]	[1, 2]	[2, 2]
$R_5$	[1, 5]	[2, 4]	[4, 5]	[4, 4]

Table 3: Flip-flop position ranges in different sessions.

### 4.2 Chaining Procedure

For each FF in Table 3, the rightmost column indicates the intersection of the ranges associated with different sessions. This is known as the **ideal range**. In other words, the ideal range for a given FF  $Ri$ , denoted by  $Range(Ri)$ , is  $\cap_j Range_j(Ri)$ . The significance of the ideal ranges is that if *every* FF can be placed in its ideal range, then it follows that the condition for *each* session to have its chain cycle equal to the MSC value is satisfied; hence the overall test time for this ordering would be equal to the lower bound computed in the previous section.

In this example, if the scan chain slots 1...5 are filled using the sequence of FFs ( $R_2, R_4, R_3, R_5, R_1$ ), it can easily be verified that every FF is indeed within its ideal range of slots, and furthermore that the overall test time is equal to the lower bound of 81 clock cycles. Comparing this test time with those in Table 2, the test time has been reduced from a worst case of 125 clock cycles to 81, a saving of 35%, simply by reordering the FFs. In this case we know that the solution is optimal because we have determined 81 to be a lower bound on test time for this circuit. An ordering that satisfies the ideal ranges (and therefore meets the lower bound) is called an **ideal solution**. Situations where an ideal solution does not exist are discussed in Section 4.3.

In the SIESTA design-for-testability methodology, the automatic ordering of the scan chain is carried out by the procedure `singleChain`. This procedure returns two items: (1) an ordering of the scan FFs, and (2) a confidence value between 0 and 1 that indicates the degree of optimality of the ordering, as judged by the procedure.

The procedure `singleChain` operates in two steps. In the first step, it attempts to find an ideal solution if one exists. This is done by transforming the problem to the well-known *maximum-size bipartite matching* problem [6], for which efficient polynomial-time algorithms exist. To do this, a bipartite graph  $(V, U, A)$  is constructed, where nodes  $V$  correspond to scan FFs, nodes  $U$  correspond to slots 1... $N$ , and arcs in  $A \equiv \{(Ri, j) \in V \times U \mid j \in Range(Ri)\}$  map each FF to slots in its ideal range. The maximum-size bipartite matching problem is to find a maximum-cardinality set  $M \subseteq A$  such that for any two arcs  $a, b \in M$ ,  $a$  and  $b$  have no node in common. This problem can be solved in  $O(\sqrt{|V|} \cdot |A|)$  time by mapping it in turn to the maximum network flow problem [6]. If the resulting matching is a *perfect matching*, i.e.,  $|M| = |V| = |U|$ , the corresponding ordering of the FFs is an ideal (optimal) solution.

If the maximum match is not a perfect match, the procedure `singleChain` moves on to the second step, which is to find a near-optimal solution. In this case the match  $M$  yields a partial placement of the FFs in the chain. Heuristics are used to place the remaining FFs in the vacant slots, each as close as possible to its ideal range. This is carried out in a greedy fashion by starting with FFs with empty or small ranges (i.e., more strongly constrained FFs) and proceeding to FFs with larger ranges. The confidence value returned is

$|M|/N$ , where  $|M|$  is the number of FFs lying within their ideal ranges and  $N$  is the total number of FFs. Note that even though the confidence level is less than 1 in such cases, it is possible for the solution obtained to be optimal (even though it does not meet the lower bound of Section 3).

### 4.3 Non-Ideal Solutions

There are two possible situations in which an ideal solution does not exist.

**Empty Ideal Range** In the circuit of Figure 2, the locally optimal ranges  $Range_1(R2)$  and  $Range_2(R2)$  are mutually exclusive, hence the ideal range of  $R2$  is empty. In this case the singleChain heuristic returns the ordering  $(R1, R2, R3)$ , with confidence level  $\frac{2}{3}$ .

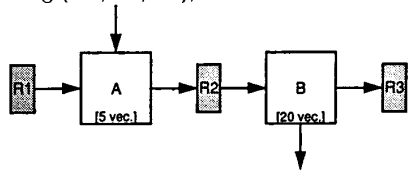


Figure 2: Empty ideal range for a register.

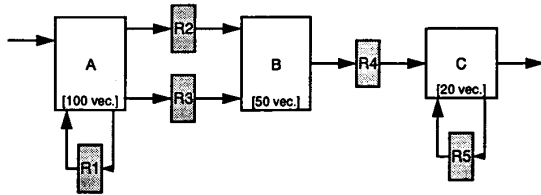


Figure 3: No perfect matching.

**No Perfect Matching** In the circuit of Figure 3, the ideal ranges for all registers are nonempty; however, no perfect matching exists. In this case the heuristic returns the ordering  $(R5, R3, R1, R2, R4)$ , where  $R3$  is outside its ideal range, hence the confidence level is 0.8.

## 5 Results

By using the scan chain ordering procedure described in this paper, and applying tests according to the overlapped scheme, the resulting test time is guaranteed to be equal to or less than that in the traditional testing scheme, in which all the logic in the circuit is tested in a single session with chain cycle equal to the length of the entire scan chain. A reduction in test time is generally achieved when the circuit has multiple kernels with varying test lengths; this is common in structured designs such as data path and signal processing circuits.

Table 4 shows the results of applying the chaining procedure to a single bit slice consisting of a single butterfly circuit, called BFLY, of a Viterbi decoder chip designed by the Jet Propulsion Laboratory.

The circuit could be partitioned into four independent combinational kernels, with test lengths 15, 13, 8 and 2. There were 28 FFs, all to be connected into a single scan chain.

The results are summarized in the first row of Table 4. The traditional test time refers to the test of the

circuit in a single session with the chain cycle equal to the length of the chain.

Circuit	Trad'l. test time	Optimal test time	%age saving	Soln. ideal?
BFLY	463	392	15.3%	Yes
BFLY'	463	398	14.0%	Yes
BFLY''	463	376	18.8%	Yes

Table 4: Flip-flop position ranges in different sessions.

The second and third rows show the results for variations on the circuit in which the test lengths of the three larger kernels are permuted among each other in different ways. The results show that for BFLY and its variations, a test time reduction in the range 14% to 18.8% can be obtained, by ordering the scan FFs in a different way. The ordering constraints may cause a tradeoff in area overhead; this can be minimized if the scan chain ordering is taken into account during early stages of the layout design. Note that an ideal solution (i.e., a perfect match of scan FFs to slots) is obtained for the BFLY circuit as well as for the variations shown, indicating that the solution must be optimal in each case.

## 6 Conclusion

A procedure for automatically constructing a single scan chain for a full scan design has been presented. The output of the procedure is an ordering of scan FFs as well as a number between 0 and 1 which indicates the degree of optimality of the solution. The approach described here is useful in minimizing the overall test time in full scan designs as well as in certain classes of partial scan designs [4].

## References

- [1] E. B. Eichelberger and T. W. Williams. A logic design structure for LSI testability. In *Proceedings, 14th Design Automation Conference*, pages 462-467, June 1977.
- [2] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design*. W. H. Freeman & Co., New York, 1990.
- [3] Rajesh Gupta. *Advanced Serial Scan Design for Testability*. PhD thesis, University of Southern California, Department of Electrical Engineering-Systems, 1991.
- [4] Rajesh Gupta, Rajiv Gupta, and M. A. Breuer. The BALLAST methodology for structured partial scan design. *IEEE Transactions on Computers*, 39(4):538-543, April 1990.
- [5] S. Bhawmik. *An Integrated CAD System for the Design of Testable VLSI Circuits*. PhD thesis, Indian Institute of Technology, Kharagpur, India, February 1988.
- [6] R. E. Tarjan. *Data Structures and Network Algorithms*. SIAM, Philadelphia, 1983.