Cost-Effective Deterministic Partitioning for Rapid Diagnosis in Scan-Based BIST

Ismet Bayraktaroglu and Alex Orailoglu
University of California, San Diego

Identifying fault-embedding scan cells is a significant challenge for fault diagnosis in scan-based BIST. Deterministic partitioning techniques provide cost-effective solutions to this problem. Both mathematical solutions and simulations on hardware implementations demonstrate the effectiveness of these techniques.

Designers use built-in self-test (BIST) in state-of-the-art chip designs to improve test quality and reduce the cost of test development and application. Despite such benefits, designers have not adopted BIST as the primary test methodology. Fault diagnosis in a BIST environment is problematic because only limited information is available in a compact signature like that produced with BIST.

Previous techniques have focused on extracting information hidden in the BIST signature, such as identification of fault-detecting test vectors.[1-3] However, the test signatures' highly compact nature, coupled with the inability to precisely model fault behavior, have confined such techniques to faults detected by only a few vectors.

For faults detected by more than a few vectors, aliasing problems prevent designers from extracting all the relevant information. In a scan-based BIST environment, fault-detecting vectors typically outnumber fault-embedding scan cells (those in which fault effects are manifested). However, such scan cells are still plentiful enough to thwart techniques for extracting diagnostic information directly from the BIST signature.

Consequently, researchers have attempted to gather more information by repeating the same test with different applications while adjusting signature analyzer parameters or observation outputs or by increasing signature register size.[4-9] In particular, partitioning-based diagnosis schemes[4,5,7] have been highly effective—and are possibly the only viable solution—for large industrial circuits.

Partitioning-based schemes provide effective fault diagnosis in a scan-based BIST environment, but generating the partitions using pseudorandom approaches makes it difficult to predict diagnosis results.[7] Both the linear feedback shift register (LFSR) parameters used for this partitioning and the faulty cell locations affect fault diagnosis time. Furthermore, the randomness of LFSR-generated partitions makes incorporating design-specific information into the diagnosis procedure difficult.

Deterministically partitioning scan cells eliminates these complications, but identifying cost-effective hardware implementations of

0740-7475/02/$17.00 © 2002 IEEE

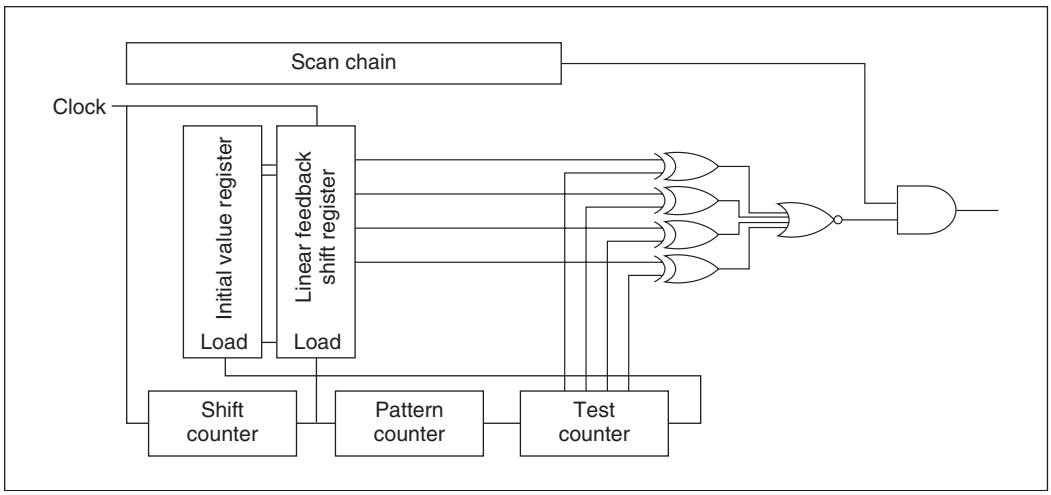


Figure 1. LFSR-based scan cell selector.[7]

such deterministic partitioning imposes a formidable challenge. In this article, we address such challenges by proving that deterministic-partitioning schemes are possible, and we describe their low-cost hardware implementations. Our proposed deterministic-partitioning approaches are capable of incorporating design and fault information, further reducing diagnosis time and enabling complex emerging fault models. We initially establish a quantitative measure for the partitions' diagnostic quality through correlation analysis, which indicates that a uniform overlap throughout the entire partitioning structure results in high-quality partitions. Cost-effectively generating such partitions in hardware requires regular, uniformly overlapping partitioning structures.

Partitioning-based diagnosis

A partitioning-based diagnosis scheme successively groups scan cells into a set of nonoverlapping partitions. Each set is a *partition group*. The test signatures corresponding to each partition yield valuable diagnostic information; each fault-free signature signals the absence of a fault in all cells within the corresponding partition. A single partition group, however, cannot adequately identify fault-embedding scan cells, because all cells in a partition that exhibits a failing signature are potential culprits. Consequently, test engineers must repeatedly apply the same test set but with differing partition elements. This process incrementally refines the candidate failures. Each application of the same test set, with repartitioned scan cells, also constitutes a partition group.

```
for each partition group c
   for each partition b in group c
     shift_counter = 0, LFSR = IVR
     while shift_counter != N
       if r bits of LFSR = b
         compact the current output
       shift_counter++
   IVR = LFSR
```

Figure 2. LFSR-based partitioning procedure.

Figure 1 depicts a possible scan-cell-partitioning hardware implementation using an LFSR and an initial value register (IVR).[7] To generate each partition in a particular partition group, the circuit loads the LFSR from the IVR. When the partition group is completely generated, the circuit updates the IVR with the LFSR's current state. The comparator (comprised of XOR and NOR gates) compares the test counter value to an arbitrarily selected set of the LFSR's *r* outputs and, when the value matches the outputs, compacts the corresponding scan cell output. For each partition, the test counter has a unique value, which distinguishes each partition in a partition group. Updating the IVR at the end of each partition group guarantees the distinctiveness of the partition groups because LFSR-generated sequences do not repeat.

Figure 2 is a pseudocode segment of the

```
for each partition group c
   for each partition b in c
     shift_counter = 0, i = 0
     while shift_counter != N
       if P(c, b, i) =
            π(shift_counter)
         compact the current output
         i++
       shift_counter++
```

Figure 3. Canonical form of the partitioning procedure.

diagnosis procedure for LFSR-generated partitions. In this pseudocode, N and b correspond to the scan chain's size and test counter's value. The pseudocode is specific to LFSR-based partitions with scan cells indexed by scan order.

Figure 3 shows a canonical form, with generalized cell selection logic and scan cell identifiers, for the partitioning-based scheme. In this canonical formulation, $P(c, b, i)$ indicates the numeric identifier of a scan cell in location i, partition b, and partition group c. The permutation function, π, provides distinct numeric identifiers for the scan cells.

Diagnostic quality of partitions

Although partitions inside a given partition group do not overlap, partitions belonging to distinct partition groups overlap in various numbers of scan cells, depending on how the partitions were generated. Overlap between partitions increases diagnostic time because it reduces the information available if both partitions happen to be fault free. A partitioning scheme polluted by an excessive number of highly overlapping partitions unnecessarily lengthens the diagnosis procedure.

We can analytically determine the amount of overlap for two partitions in distinct partition groups for any partitioning scheme in which a group's partitions are disjoint and cover all scan cells in the scan chain. A partition's overlap with the union of all partitions in a distinct group is equal to the partition size, because the partition group covers all N scan elements. The expected partition size is N/B, where B is the number of partitions in a partition group. The expected overlap of one partition in the group is the expected partition size times the probability of selecting that partition: $(N/B)(1/B) = N/B^2$.

Because the expected overlap value is identical for any partition generation scheme, the cause of possibly differing partition quality is found in higher-order measures such as overlap deviations. We convert overlap deviations into a single quality measure by calculating the root mean square (RMS) of the deviations. Although we can make deterministic partitions exhibit an RMS of 0 as long as the partition overlap uniformly equals the expected overlap value, we can show that the RMS for LFSR-generated partitions is nonzero and varying.

We compute the RMS value for partitionings generated by all primitive polynomials of degree 14 and 15. (There are 756 primitive polynomials of degree 14, and 1,800 of degree 15.) Figure 4 provides the expected overlap and the RMS deviation in overlap values for scan cell length and partition counts that result in an expected overlap of 4, 8, and 16 scan cells. The overlap among partitions significantly varies from the expected value with RMS deviation ranging from 25% to 50% of the expected overlap. However, the RMS deviation varies only slightly depending on the LFSR width.

The RMS deviations depicted in Figure 4 constitute the average deviation value for all primitive polynomials of corresponding degree. Although RMS values indicate significant overlap deviations, it is unclear whether such deviations translate into diagnostic time degradations. Figure 5 shows the results of our correlation analysis to validate the RMS measure's diagnostic quality.

For each simulation parameter (the number of cells, the number of partitions, and the LFSR size) in Figure 4, we simulated the diagnosis procedure to determine the expected fault diagnosis time for each primitive polynomial. Furthermore, we calculated the correlation between the RMS overlap deviation and the expected fault diagnosis time. The results depicted in Figure 5 clearly show a significant, positive correlation between the expected fault diagnosis time and the RMS partition quality measure we've proposed.

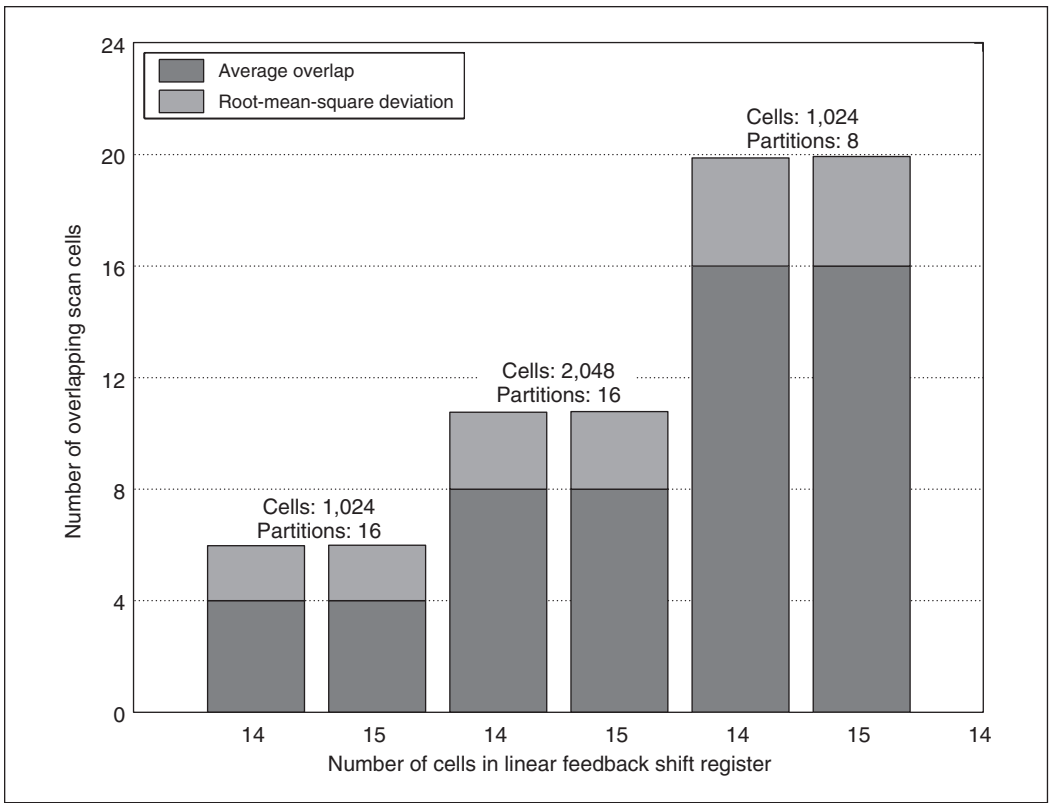


Figure 4. RMS deviation of partition overlaps.

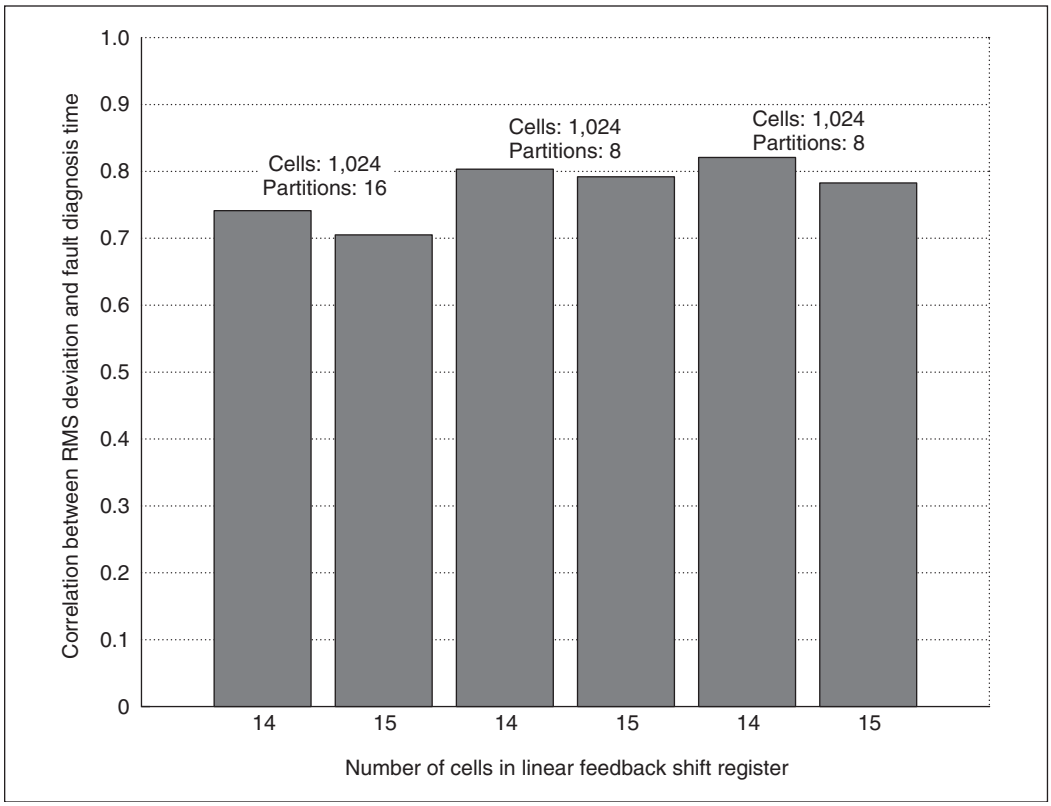


Figure 5. Correlation between RMS deviation and expected fault diagnosis time.

Table 1. Deterministic partitioning of a 25-cell scan chain. (Bold numbers identify the overlap with partition 0 in group 0.)

Partition group	Partition 0					Partition 1					Partition 2					Partition 3					Partition 4				
0	**0**	**1**	**2**	**3**	**4**	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	**0**	6	12	18	24	5	11	17	23	**4**	10	16	22	**3**	9	15	21	**2**	8	14	20	**1**	7	13	19
2	**0**	11	22	8	19	5	16	**2**	13	24	10	21	7	18	**4**	15	**1**	12	23	9	20	6	17	**3**	14
3	**0**	16	7	23	14	5	21	12	**3**	19	10	**1**	17	8	24	15	6	22	13	**4**	20	11	**2**	18	9
4	**0**	21	17	13	9	5	**1**	22	18	14	10	6	**2**	23	19	15	11	7	**3**	24	20	16	12	8	**4**

Constructing deterministic partitions

Reduced partition overlap reduces diagnosis time. Accordingly, as only deterministic partitioning can minimize overlap variation, we can generate deterministic partitions that achieve an RMS value of 0 and so are far better than LFSR-based schemes at aiding fault diagnosis.

A partitioning with 0 RMS overlap deviation must have uniformly minimal overlap with all partitions, strictly constraining partition structure. If the number of partitions in a partition group equals the partition size, S, where $S = N/B$, a partition must have a single overlap to each partition inside every other partition group. This constraint sharply limits the number of attainable partition groups, making their identification and construction complex.

Mathematical foundation

Table 1 illustrates a deterministic partitioning that satisfies the aforementioned constraints for a scan chain size of 25 cells. The scan chain comprises five partition groups, each consisting of five partitions. Each row in Table 1 corresponds to a partition group with five partitions. We can easily verify that the number of overlapping cells for these partitions consistently equals 1 for partitions in distinct partition groups and 0 for partitions in the same group. We can trace the minimal overlap property, at least for a single partition, by observing in bold all the elements of the first partition of the first partition group in this table.

Although such partitions have minimal overlap, generating them in hardware is difficult. The relationships across groups yield the recurrence relation in equation 1. Equation 2 denotes a simple definition of the initial partition group. In these and the following equations, assume that c, b, and i vary between 0 and $S - 1$ (in Table 1, $S = 5$).

$$P(c+1, b, i) = P(c, [b+i] \bmod S, i) \quad (1)$$

$$P(0, b, i) = bS + i \quad (2)$$

Although these equations are straightforward, generating the partitions directly through these equations requires high area overhead because of recursion. A direct implementation of a recurrence relation requires that the previous result be stored. A simple hardware implementation for the partition groups in Table 1 requires a solution of the recurrence relations in equations 1 and 2. Recursive application of equation 1's recurrence relation, starting from the initial partition group of equation 2, results in the following nonrecursive partition equation:

$$P(c, b, i) = ([ci + b] \bmod S)S + i \quad (3)$$

The existence of partitions with minimal overlap, as Table 1 shows, is not coincidental. The following theorem shows that partition groups generated by equation 3 always exhibit the minimal overlap of 1 for any prime number S.

Theorem: Partitions generated by equation 3 have an overlap of 1 whenever they are in distinct partition groups, and have no overlap whenever they are in the same partition group (for S prime; and c, b, and i each less than S).

Proof: For the elements of two partitions, $P(c_1, b_1, i_1)$ and $P(c_2, b_2, i_2)$, to overlap, the following equality must be satisfied:

$$P(c_1, b_1, i_1) = P(c_2, b_2, i_2) \quad (4)$$

Table 2. Quotient-uniform-deterministic partitioning of a 25-cell scan chain.

Partition group	Partition 0	Partition 1	Partition 2	Partition 3	Partition 4
0	0 5 10 15 20	1 6 11 16 21	2 7 12 17 22	3 8 13 18 23	4 9 14 19 24
1	0 6 12 18 24	1 7 13 19 20	2 8 14 15 21	3 9 10 16 22	4 5 11 17 23
2	0 7 14 16 23	1 8 10 17 24	2 9 11 18 20	3 5 12 19 21	4 6 13 15 22
3	0 8 11 19 22	1 9 12 15 23	2 5 13 16 24	3 6 14 17 20	4 7 10 18 21
4	0 9 13 17 21	1 5 14 18 22	2 6 10 19 23	3 7 11 15 24	4 8 12 16 20

Because $|i_1 - i_2| < S$, equation 4 holds if $i_1 = i_2 = i$. Therefore, the overlap condition reduces to

$$(c_1 - c_2)i \overset{\text{mod}\,S}{=} b_2 - b_1 \quad (5)$$

For partitions in different partition groups ($c_1 \neq c_2$) we need to analyze only two cases:

- $b_1 = b_2$ (there is a unique solution, $i = 0$); and
- $b_1 \neq b_2$ (a unique solution exists for S prime).

For partitions inside the same partition group ($c_1 = c_2$), overlap necessitates equality of b_1 and b_2; a partition can only overlap with itself inside a partition group.

We have shown that partitions within distinct partition groups have an overlap of 1, and that partitions within the same group do not overlap. Consequently, the partitions generated by equation 3 fulfill the minimal overlap requirement and have 0 RMS overlap deviation.

Monotonic partitions

We can simplify the hardware implementation corresponding to the pseudocode in Figure 3 by choosing π as the identity function. However, this selection necessitates that function $P(c, b, i)$ be monotonic in parameter i, because the scan cells can be accessed only sequentially in a regular scan shift operation. Table 1 shows that not all partitions are monotonic.

Because ordering numeric identifiers inside a partition has no restrictions, we can easily reorder them to satisfy the monotonicity requirement. However, arbitrarily reordering these identifiers destroys the regular recurrence relations. To generate partitions with low-cost hardware implementation, we investigate the structure of partitions generated by equation 3, and, with that structure, provide an alternative partitioning for which the increasing order property holds. Determining the structure of partitions generated by equation 3 requires grouping the scan cell identifiers according to their remainder and quotient to S.

All partitions in Table 1 except those in the first row include one element from each of the remainder and quotient groups. Partition elements are always composed of elements of the remainder groups 0, 1, 2, …, $S - 1$. We can generate elements of the first partition of each group by selecting an element from the quotient groups in increments of 0, 1, 2, …, $S - 1$. We can generate elements of the higher numbered partitions by rotating the quotient selection sequence to the right. Besides the partition in Table 1, all such partitions generated by equation 3, for S prime, exhibit the same regular structure. We call this class of partitions *remainder uniform partitions*.

From these observations, we can build partitions with the monotonically increasing order property by interchanging the quotient and the remainder in the partition structure. We call this new class of partitions *quotient uniform partitions*. Table 2 shows an example, for $S = 5$. A comparison of Tables 1 and 2 reveals that, except for the first group, both partitionings are identical, yet they order partition groups and partitions differently. Consequently, in both tables, the total number of partition groups satisfying the minimal overlap property is 6. Indeed, exhaustive computer simulations for S up to 7 indicate that for a scan size of S^2 there are at most $S + 1$ partition groups satisfying the minimal overlap criterion.

We obtain the partition equation for the quotient uniform partitions by interchanging the terms corresponding to the quotient group, $(ci + b)$, and remainder group, i, in equation 3:

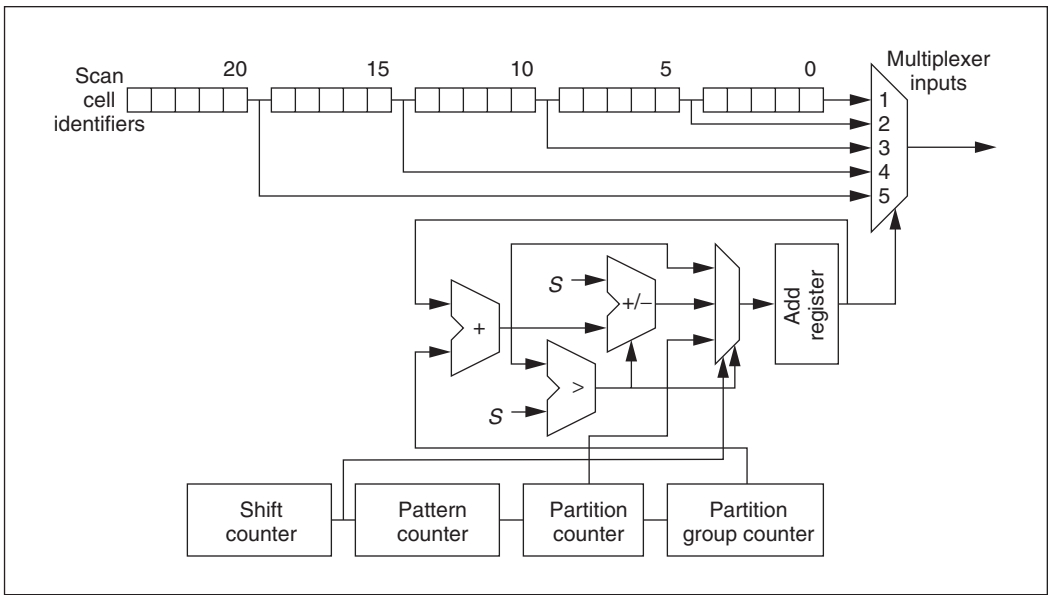


Figure 6. Tapped partitioning.

$$P(c, b, i) = iS + (ci + b) \bmod S \quad (6)$$

Generalization

We have shown the existence of deterministic partitioning schemes with uniform overlap and monotonicity, although for prime partition sizes. The primality constraint can be eased for partition size because the proof of the uniform overlap theorem uses the constraint solely to guarantee the uniqueness of the inverse of the term $c_1 - c_2$. With a modulo arithmetic, the integers modulo a prime number constitute a finite field—a Galois field denoted $GF(s)$—which guarantees that every field element except 0 has only one inverse. We can thus achieve uniformly overlapping monotonic partitions for any finite field in equation 6. Furthermore, $(ci + b)$ constitutes the sole term needing evaluation in the finite field. Consequently, with the following generic partition equation, we can generate uniformly overlapping monotonic partitions:

$$P(c, b, i) = iS + (c \otimes i) \oplus b \quad (7)$$

In equation 7, $\otimes$ and $\oplus$ denote the two finite-field arithmetic operators. One possible choice for the finite field that provides partition sizes equal to those generated by pseudorandom partitioning is the extension fields of $GF(2)$.

Hardware implementation of deterministic partitions

Although so far our analysis has focused largely on monotonic implementations, we imposed monotonicity because of the monotonicity constraint on the scan cell ordering. Under multiple scan chains and more complex control, we can drop this constraint on scan cell ordering, and thus enable nonmonotonic implementations. Here, we show tapped implementations for nonmonotonic partitions and outline cost-effective hardware implementations both for a prime number of partitions and for partition sizes corresponding to powers of two.

Nonmonotonic implementation

We can generate nonmonotonic partitions without modifying the regular scan shift operation, because each partition has an element from every remainder group in remainder uniform partitions. The partitions' nonmonotonicity necessitates nonsequential access to the scan chain. Tapping the scan chains so that each scan cell group between taps has the same quotient group can circumvent the problems associated with such nonsequential access. Figure 6 shows our proposed tap implementation, which directly mimics the partitions' structure. Access to the scan chain's internal points provides direct access to quo-

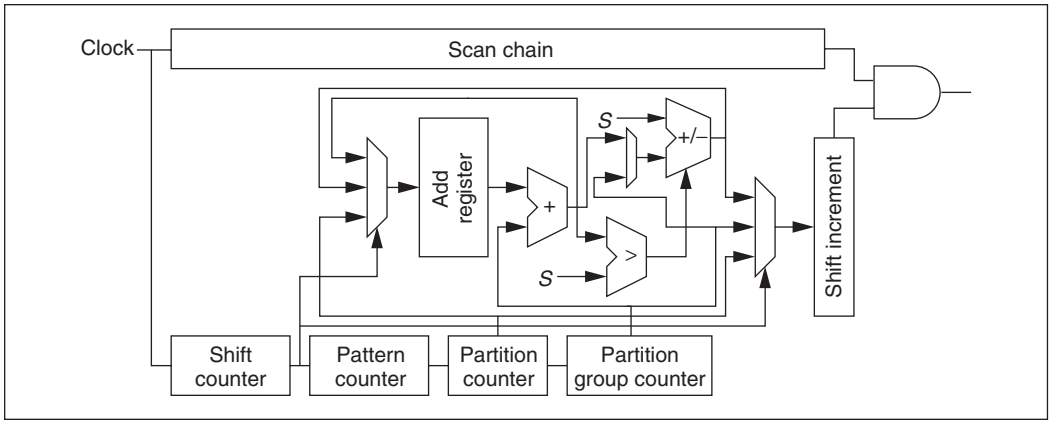


Figure 7. Successive-difference-based partitioning hardware.

tient groups, and the shifting process affords access to remainder groups. Because partitions have a single element from each remainder group, the output multiplexer can select the correct quotient group when necessary.

For example, for the third partition of the fourth partition group in Table 1, the multiplexer receives the select signals 3, 1, 4, 2, and 5. At the end of the first cycle, the multiplexer selects scan cell 10, which is in remainder group 0. After the shift operation, the content of scan cell 1, which is in remainder group 2, is shifted into scan cell 0, and the multiplexer selects it for compaction. Another shift operation shifts the content of scan cell 17 into scan cell 15, which the multiplexer consequently selects. After five cycles, all necessary scan cell outputs for the third partition have been compacted at the output.

If extra signature compactors are available, they can produce signatures in parallel for all partitions in a partition group by connecting the output of the last scan cell to the input of the first scan cell. The result is sharply reduced diagnosis time, despite some associated hardware overhead. Although it might seem that diagnosis time is further reduced by a factor of S, the reduction is actually only a factor of $S/2$. The reason is that the connection from the last to the first scan cell eliminates the capability of shifting in new patterns while shifting out results.

Although the tapped implementation lets us generate nonmonotonic partitions with seemingly low-cost hardware, the routing overhead, especially for a higher number of partitions, can be high. Therefore, we also address hardware implementation of monotonic partitions, which do not suffer from high routing overheads.

```
for each partition group c
   for each partition b in group c
     ShiftCounter = 0--
     Add = b, ShiftIncrement = b
     while ShiftCounter != N
       Until ShiftIncrement = 0
         ShiftIncrement--, ShiftCounter--
       compact the output of the current cell
       Add = Add + c
       If Add >= S
         Add = Add – S
         ShiftIncrement = c
       Else
         ShiftIncrement = c + S
```

Figure 8. Successive-difference-based partitioning procedure.

Monotonic implementation

The partitions generated with modulo arithmetic, defined by equation 6, can be generated in hardware via the generic pseudocode of Figure 3. This partition generation hardware does the $P(c, b, i)$ implementation directly by using two multipliers, two adders, and a modulo operator. We call this an *absolute-value-based* implementation. Figure 7, which shows an alternative, improved implementation using difference $D = P(c, b, i + 1) - P(c, b, i)$, corresponds to the pseudocode of Figure 8 (next page). D reduces to $c + S$ unless $[c(i + 1) + b]$ $mod\ S < (ci + b)\ mod\ S$, in which case it reduces to c. This *successive-difference-based* hardware implementation requires significantly fewer

```
for each partition group c
   for each partition b in group c
     ShiftCounter = 0
     s = S-1
     gf = b
     ci = c
     While ShiftCounter != 0
       ShiftIncrement--
       if gf == 0
         Compact current output
       if s == 0
         gf = ci + b
         s = S-1
         ci++
```

Figure 9. *GF*(2^k)-based partitioning procedure.

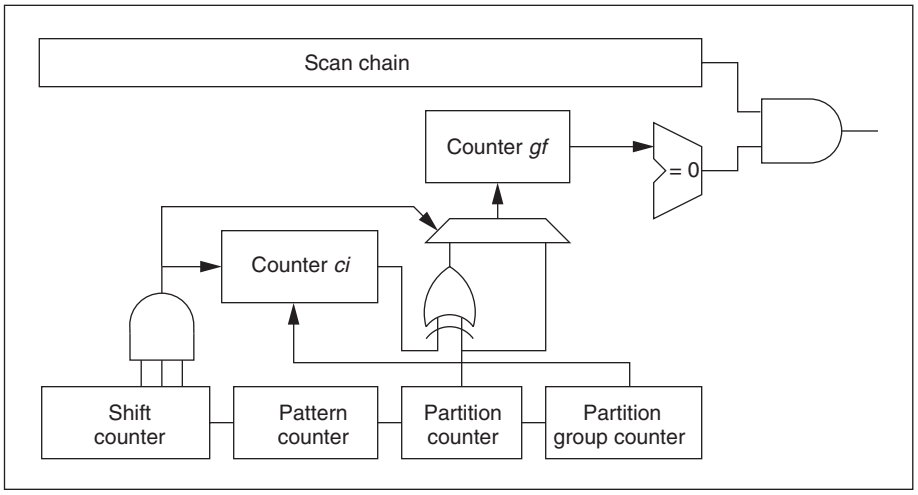


Figure 10. Hardware implementation of *GF*(2^k)-based partitions.

hardware components, compared to the absolute-value-based implementation.

In the modified implementation, both register **Add** and counter **ShiftIncrement** are set to the first element of the current partition, which is always equal to the partition number. Subsequently, register **Add** continuously holds the current remainder group of the current element in the partition. The current remainder group is updated whenever that element is reached in the scan chain. The circuit reloads the **ShiftIncrement** counter with the difference between $P(c, b, i+1)$ and $P(c, b, i)$ upon reaching 0. After the entire scan chain is shifted, the process restarts. At the end of the test application session, the signature is shifted out, the partition count is incremented, and the process repeated.

GF(2) implementation

We can implement, in both absolute-value- and successive-difference-based approaches, partitions that use the *GF*(2) extension fields. Here, however, we outline an implementation in which we use a scheme based on successive differences to calculate the term that is evaluated in binary arithmetic, *iS*; and we use an absolute-value-based approach to implement the term, $(c \otimes i) \oplus b$, that uses extension fields of *GF*(2).

The pseudocode in Figure 9 implements term iS through successive differences and directly implements term $(c \otimes i) \oplus b$. We calculate $c \otimes i$, however, by progressively employing an LFSR. We can generate the elements of the field *GF*(2^k) with a primitive polynomial of degree k starting from α, which is one of the primitive elements of *GF*(2^k), and successively multiplying it by α. Therefore, the elements of *GF*(2^k) can be ordered as 0, 1, α, α^2, …. Of course, once the degree of the elements equals k, the remainder with respect to the primitive polynomial must be evaluated. If we follow the same sequence during partition generation, term *ci* results in sequence 0, c, $c\alpha$, $c\alpha^2$, …. We evaluate this sequence by employing a k-bit LFSR with the same primitive polynomial that generates field *GF*(2^k) and with the initial seed of c. An XOR operation can simply add b to $c \otimes i$.

As illustrated in Figure 9, whenever counter s identifies the condition that scan cell *iS* is reached, the circuit initializes counter *gf* to $(c \otimes i) \oplus b$. Whereas counter s is a binary modulo 2^k counter, counter *gf* is implemented as an LFSR counting in reverse. We augment the reverse-counting LFSR such that it visits the *all-zeros* state after the state 0 … 01. Because counter s is a modulo 2^k counter, we can eliminate it from the circuitry by employing the shift counter's least significant k bits. Figure 10 depicts a hardware implementation corresponding to the pseudocode given in Figure 9.

Hardware overhead

The partitioning-based diagnosis schemes, whether pseudorandom or deterministic, require additional hardware atop the regular BIST for scan cell selection. Here we compare additional hardware requirements for LFSR-based partitioning[7] and for two deterministic-partitioning schemes based on modulo and finite-field arithmetic. The LFSR-based scheme employs one LFSR, one register, one log(S) bit comparator, and one log(N/S) bit counter. Although the sizes of the LFSR and the register are typically user defined, LFSRs of size 16 are satisfactory.[7] The deterministic-partitioning scheme based on modulo arithmetic uses one log(S) bit register, two [log(S) + 1] bit adders and comparators, two log(S) bit counters, one [log(S) + 1] bit counter, and three multiplexers. Finally, the partitioning scheme based on finite-field arithmetic exploits four log(S) bit LFSRs, one log(S) bit comparator and multiplexer, and a few AND and XOR gates. Whereas the modulo-arithmetic-based partitioning scheme has the highest hardware overhead, the finite-field-arithmetic-based partitioning scheme's hardware overhead is comparable to that for pseudorandom partitioning.

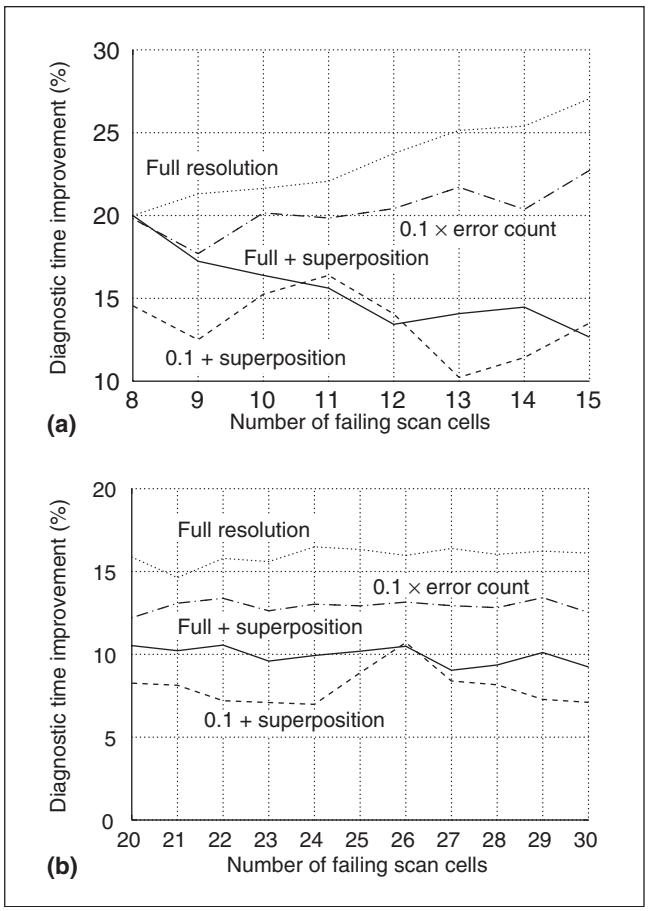


Figure 11. Diagnosis time improvements for $B = S = 17$ (a) and $B = S = 31$ (b).

Simulation results

We ran a set of simulation experiments to verify the effectiveness of our proposed partitioning techniques. We compared the deterministic-partitioning techniques to the best pseudorandom partitioning techniques available: a pseudorandom partitioning with superposition,[5] and one with no superposition.[7] In comparing the deterministic-partitioning techniques to our earlier results with superposition,[5] we incorporated the superposition principle to conduct a fair comparison. We exploited the superposition principle in earlier research to generate composite partitions from already examined partitions. In turn, this increased the number of partitions without additional application of the test. An increased number of partitions boosts the information extracted without additional test time and so reduces diagnosis time.[5]

Improvements for $GF(s)$

We began our simulations for partitions using modulo arithmetic. In this case, we performed simulations on two prime number partitions, 17 and 31. A fair comparison suggests that the number of partitions be kept equal, yet the two compared schemes display conflicting requirements. We selected primes 17 and 31 because they are adjacent to the corresponding powers of 2 (16 and 32)—LFSR-based partitioning forces the number of partitions to be powers of 2—and because they bestow a slight advantage on alternating schemes.

Figure 11 shows the improved diagnosis time for 31 deterministically partitioned partitions compared to 32 pseudorandomly partitioned partitions, and for 17 deterministically partitioned partitions compared to 16 pseudorandomly partitioned partitions. The figure shows improvement for both full diagnostic resolution (all fault-embedding scan cells are exactly identified) and diagnostic resolution of 0.1 × error count. (The diagnosis resolution

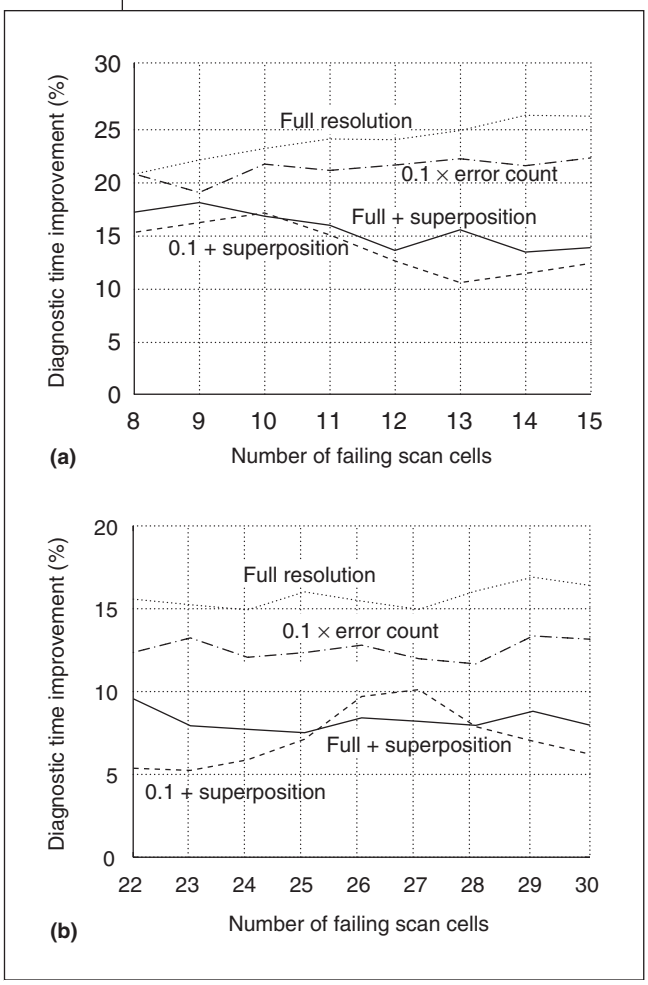


Figure 12. Diagnosis time improvements for *B* = *S* = 16 (a) and *B* = *S* = 32 (b).

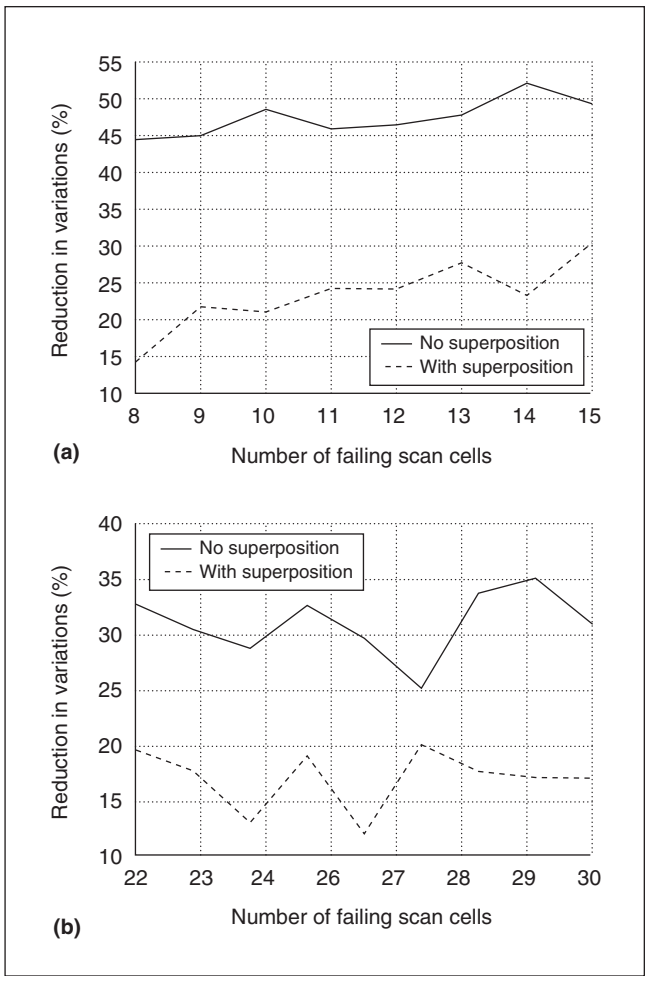


Figure 13. Improvements in diagnostic time variations for *B* = *S* = 16 (a) and *B* = *S* = 32 *(b).*

metric of 0.1 × error count, originally suggested by Rajski and Tyszer,[7] is the time at which many fault-free scan cells remain ambiguous and still reside in the candidate-failing scan cell set.)

We have used Rajski and Tyszer's results[7] whenever available, as with some of the diagnosis results for the resolution metric 0.1 × error count; otherwise, our results came from implementing the outlined procedure.

Our results indicate that deterministic partitioning consistently reduces diagnosis time. The improvement is best for full diagnostic resolution, smaller partition sizes, and no superposition. Whereas pseudorandom partitioning easily exhibits good diagnostic resolution up to a point, full resolution benefits significantly from deterministic techniques.

Improvements for *GF*(2)

Additional simulation experiments performed on the partitions generated using the extension fields of *GF*(2), displayed similar improvements (as can be observed in Figure 12), affected by the same parameters: superposition, partition size, and number of errors. The number of partitions did not exactly match the pseudorandom case in previous experiments. But in the *GF*(2) simulations, the number of partitions exactly matched the number of pseudorandom partitions, providing an exact comparison.

Diagnostic time variations

We also conducted further experiments to compare the predictability of diagnosis procedures. We exploited the standard deviation of the diagnostic times as a measure of predict-

ability. Figure 13 shows that the variations in diagnosis time for the deterministic case are lower than those for the pseudorandom case. Because superposition effectively reduces the effect of varying overlaps, it also lessens improvements in diagnosis time predictability.

ALTHOUGH IMPLEMENTING deterministic partitioning in hardware is challenging, especially within low area overhead, the regular partition structures we've identified enable low-cost hardware implementations. Implementation regularity, associated reduction in hardware overhead, significant reduction in average diagnostic time, and imperviousness to diagnostic time deviations make deterministic partitioning a powerful new BIST-based diagnosis tool.

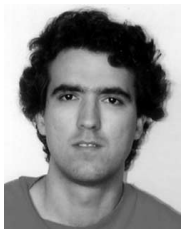
The importance of BIST-based diagnosis will undoubtedly increase as designers exploit its potential for incorporating design and fault effect information. We are investigating methodologies to incorporate such design and fault effect information in deterministic partitioning. ■

Acknowledgment

The work of Ismet Bayraktaroglu is supported by an IBM Graduate Fellowship.

■ References

1. R.C. Aitken and V.K. Agarwal, "A Diagnosis Method Using Pseudorandom Vectors without Intermediate Signatures," *Proc. Int'l Conf. Computer-Aided Design* (ICCAD 89), IEEE CS Press, Los Alamitos, Calif., 1989, pp. 574-580.
2. W.H. McAnney and J. Savir, "There Is Information in Faulty Signatures," *Proc. Int'l Test Conf.* (ITC 87), IEEE CS Press, Los Alamitos, Calif., 1987, pp. 630-636.
3. C.E. Stroud and T.R. Damarla, "Improving the Efficiency of Error Identification via Signature Analysis," *Proc. 13th IEEE VLSI Test Symp.* (VTS 95), IEEE CS Press, Los Alamitos, Calif., 1995, pp. 244-249.
4. I. Bayraktaroglu and A. Orailoglu, "Deterministic Partitioning Techniques for Fault Diagnosis in Scan-Based BIST," *Proc. Int'l Test Conf.* (ITC 00), IEEE CS Press, Los Alamitos, Calif., 2000, pp. 273-282.
5. I. Bayraktaroglu and A. Orailoglu, "Improved Fault Diagnosis in Scan-Based BIST via Superposition," *Proc. 37th Design Automation Conf.* (DAC 00), ACM Press, New York, 2000, pp. 55-58.
6. J. Ghosh-Dastidar, D. Das, and N.A. Touba, "Fault Diagnosis in Scan-Based BIST Using Both Time and Space Information," *Proc. Int'l Test Conf.* (ITC 99), IEEE CS Press, Los Alamitos, Calif., 1999, pp. 95-102.
7. J. Rajski and J. Tyszer, "Diagnosis of Scan Cells in BIST Environment," *IEEE Trans. Computers*, vol. 48, no. 7, July 1999, pp. 724-731.
8. J. Savir and W.H. McAnney, "Identification of Failing Tests with Cycling Registers," *Proc. Int'l Test Conf.* (ITC 88), IEEE CS Press, Los Alamitos, Calif., 1988, pp. 322-328.
9. Y. Wu and S.M.I. Adham, "Scan-Based BIST Fault Diagnosis," *IEEE Trans. Computer-Aided Design*, vol. 18, no. 2, Feb. 1999, pp. 203-211.



Ismet Bayraktaroglu is currently a member of the technical staff at Sun Microsystems. He performed the work discussed in this article while pursuing a PhD at the University of California, San Diego. His research interests include BIST, diagnosis of BIST designs, and concurrent test of DSPs. Bayraktaroglu has a BS and MS in electrical engineering from Bogazici University, Istanbul, Turkey, and a PhD in computer engineering from the University of California, San Diego.

Alex Orailoglu is a professor in the Computer Science and Engineering Department at the University of California, San Diego. His research interests include digital and analog test, fault-tolerant computing, computer-aided design, and embedded processors. Orailoglu has an SB in applied mathematics from Harvard University, and an MS and PhD in computer science from the University of Illinois, Urbana-Champaign. He is a member of the IEEE.

■ Direct questions and comments about this article to Alex Orailoglu, Computer Science and Engineering Dept., Univ. of California, San Diego, La Jolla, CA 92093-0114; alex@cs.ucsd.edu.