Concurrent Test for Digital Linear Systems

Ismet Bayraktaroglu, *Student Member, IEEE,* and Alex Orailoglu, *Member, IEEE*

***Abstract*—Invariant-based concurrent test schemes can provide economical solutions to the problem of concurrent error detection. An invariant-based concurrent error-detection scheme for linear digital systems is proposed. The cost of concurrent error-detection hardware is appreciably reduced due to utilization of a time-extended invariant, which extends the error-checking computation over time and, thus, reduces hardware requirements. Error-detection capabilities of the scheme proposed in this work are analyzed and conditions on the implementation for achieving complete fault coverage are outlined. Implementations fulfilling such conditions have been shown through experiments to provide 100% concurrent fault detection.**

***Index Terms*—Accumulation-based invariants, concurrent test, digital linear systems, online test.**

I. Introduction

VERY LARGE scale integration (VLSI) technology has evolved to a level where large systems, previously implemented as printed circuit boards with discrete components, are integrated into a single integrated circuit (IC). As industry continues to push the limits of VLSI technology, reliability of such systems becomes one of the primary concerns. While manufacturing test provides a relatively easy way of eliminating defective ICs, thus, virtually preventing any defective IC from reaching consumers, it cannot completely eliminate reliability concerns. Interestingly, aggressive new chip design techniques frequently adversely effect chip reliability during functional operation.

Linear digital systems have been widely utilized in signal processing applications during the last two decades. Mission-critical applications, such as military, satellite, and medical applications, have employed every means available at the expense of considerable cost in order to improve reliability by tolerating fault occurrences at field. Less critical consumer applications, on the other hand, have ignored fault tolerance as well as concurrent fault detection as competition removed margins necessary for including fault tolerance.

The increasing design sizes and densities of IC chips as a result of improvements in VLSI manufacturing and fabrication are shifting the predominant IC cost to design aspects. While larger chips help decrease the proportional cost of error-handling hardware, such reductions are not as of yet sufficient to induce a widespread adoption of error-handling hardware in consumer applications. Yet consumer applications, while benefitting from embedded error handling, do not pose as stringent requirements as traditional fault tolerance niches such as medical, space, and others. While medical applications necessitate typically instantaneous error detection and, frequently, error recovery, it may be perfectly sufficient for consumer applications to provide the same capabilities, albeit with some latency. The ability to incorporate such reduced latency in error handling can, in turn, set the basis of mathematical analyses and consequent schemes, which at extremely low-cost can attain complete concurrent fault coverage within latencies of less than human response times.

Various techniques have been developed for concurrent testing of digital systems. Among these, algorithm-based error-detection approaches are well suited for digital-signal processing applications as such systems can be usually represented by a simple algorithm. In [7], Huang and Abraham present an algorithm-based fault detection scheme for matrix operations, which relies on checksum codes for fault detection. As floating or fixed-point implementations of such operations cause problems for checksum codes, [10] proposes real-number codes as an alternative. In [1], an algebraic model of algorithm-based fault-tolerant schemes is developed to prove fault detection and correction capabilities of such schemes.

Error-detection and correction schemes for fast Fourier transform networks and processors are developed in [5], [8], [15], and [17]. Fault detection capabilities of algorithm-based techniques are further extended in [11] to QR factorization. While a dependence graph-based approach is introduced in [16] for algorithm-based fault-tolerant systems, design of fault-tolerant linear digital state variable systems is investigated in [3]. In [4], techniques developed in [3] are extended to nonlinear systems by linearizing such systems in a time-varying manner. In [14], a general fault-tolerant synthesis scheme for linear operations is proposed.

All of the previously proposed schemes utilize either error-detection codes or an invariant property of the system so as to detect faults online. Analysis of such schemes indicates that 100% fault detection for single *unit faults* is possible; gate-level coverage may and typically does differ. Numerical inaccuracies reduce further the effectiveness of such schemes for faults with low-magnitude effects. The hardware overhead incurred by the previously proposed schemes may further be reduced in order to enable their acceptance in consumer electronics, if error-detection latency is introduced.

In this paper, we propose concurrent error-detection techniques that are low-cost and provide high fault coverage within a short latency for linear digital systems. The proposed scheme detects the existence of faults in the circuit by utilizing the average behavior of the system, as opposed to previously proposed schemes that utilize the instantaneous behavior of the system.

Manuscript received January 23, 2000; revised November 18, 2000. This work was supported by the Hughes Space and Communications and by the University of California Micro Program. This paper was recommended by Associate Editor K.-T. Cheng.

The authors are with the Computer Science and Engineering Department, University of California at San Diego, La Jolla, CA 92093 USA (e-mail: ibayrakt@cs.ucsd.edu; alex@cs.ucsd.edu).

Publisher Item Identifier S 0278-0070(01)06883-X.

The short latency in fault detection is more than compensated through drastically reduced overhead and complete fault coverage of all stuck-at faults, impervious even to the confounding effects of numerical inaccuracies.

We present an invariant for concurrent fault detection in linear digital systems and outline conditions on the implementation in order to attain very high fault coverage and at the same time tolerate numerical inaccuracies. Additionally, implementation strategies to fulfill such conditions are provided. Simulation results for the systems implemented by fulfilling the outlined conditions indicate that high concurrent fault coverage is achievable within small area overhead.

The hardware added for concurrent fault detection can also be shared with an offline built-in self-test (BIST) solution. An offline BIST solution that employs arithmetic pattern generators is applied to one of the linear systems implemented in this work. Fault-simulation results for BIST hardware indicate that over 99% fault coverage is achievable with small incremental area overhead.

A brief description of the type of digital systems that this work addresses together with the assumptions utilized in this paper are provided in Section II. While Section III provides a brief introduction to time-extended invariant-based fault-detection approaches, Section IV provides a derivation of the invariant equations and possible implementation styles for concurrent error-detection hardware. In Sections V and VI, implementation strategies for tolerating numerical inaccuracies and ensuring fault effect accumulation are outlined, respectively. Section VII provides experimental results and Section VIII presents conclusions.

II. Assumptions

In this paper, a concurrent detection technique for linear digital systems is proposed. The scheme is based on checking the average behavior of a system. Checking variations from average behavior, while providing very high fault coverages under certain design constraints, introduces latency to fault detection. A fault is detected only after a number of observations, as opposed to fault secure systems in which a fault is detected as soon as it is observed at the output.

During verification of the proposed scheme, a single stuck-at fault model is utilized. While the outlined design constraints provide 100% fault coverage under the fault model, the proposed scheme is not limited to detection of single faults. Even multiple faults are also detected unless they perfectly compensate. The latency for such detection of multiple faults varies depending on the phase relation among fault effects.

The following general form for digital systems is utilized in this paper:

$$y[n] - \sum_{k=1}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]. \qquad (1)$$

Systems described by (1), depending on the coefficients, may have feedback loops. A system with feedback (i.e., $a_k \neq 0, \exists k$) generates an infinite impulse response (IIR). Systems without feedback, on the other hand, generate finite impulse responses (FIRs). The proposed scheme is applicable to both FIR and IIR systems. Additionally, there exist various implementation forms of such systems, which include direct form, transposed form, parallel, and serial implementations [12]. While the proposed scheme can be proven to work with all implementation forms, we experiment in this paper with the transposed direct-form implementation, depicted in Fig. 1.

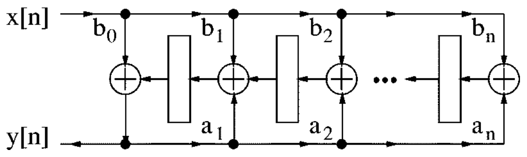


Fig. 1. Transposed direct-form implementation.

III. Invariant-Based Approaches

While systems may implement rather complex functionality, a simple invariant of such systems can be always found. Existence of such an invariant, while providing a means of concurrently checking the system, does not guarantee desired levels of fault detection capability. Depending on how thoroughly the invariant checks the functionality, the fault-coverage level may vary drastically. Though determination of a simple invariant is quite easy, determination of one that provides online checking of the complete functionality is highly challenging and usually requires modifications to the implementation.

Invariants usually work by checking the result of the calculations in groups instead of checking the functionality of each calculation separately. As long as occurrence of errors at any time is limited to one, such checking procedures can detect all possible irredundant faults. Such group comparisons reduce the checking hardware by reducing the amount of calculation required at each time step. The calculations are usually grouped together only if they occur in the same time interval. Invariants that can be determined from a set of data available in only one clock cycle are called *instantaneous* invariants in this paper. Calculations, as shown in this paper, can also be grouped even if they do not happen in the same time frame. Instead of checking correctness of the functionality at each time step, a checking scheme extended to a number of time steps can still detect all faults that may have occurred at any of the time steps by checking the average behavior of the system. Checking the average behavior of the system in a time-extended manner further reduces the area overhead required for concurrent error detection.

Instantaneous invariants, when applied to sequential circuits, require access to the internal states of the system, which in turn increases the hardware overhead. Additionally, such access necessitates information about the implementation of the system. Consequently, the online checking hardware needs to be modified, increasing the cost of testing, whenever the circuit implementation is modified without changing the functionality; such a modification may even be a simple retiming transformation. A time-extended invariant with an associated tolerance, on the other hand, can be checked by observing solely the inputs and outputs of the system. Such invariants work independently of the implementation and can provide low-cost concurrent error-detection capabilities.

Even though the invariant checking hardware does not depend on the implementation style for time-extended invariants, fault-coverage levels of such implementations depend on the implementation style. In this paper, we analyze effects of implementation style on fault coverage and provide conditions on the implementation in order to achieve online fault detection without missing even a single fault. Fault detection latency of the proposed scheme is also analyzed and the correlation between detection latency and fault magnitude effect is shown. While the detection latency is quite long for faults with very low magnitudes, the impact of such faults on the signal-to-noise ratio (SNR) is shown to be less than that of numerical inaccuracies. Additionally, faults with large magnitude effects are shown to be detected within a very short time period.

IV. Implementation

We introduce the invariant utilized in this paper and its implementation by showing its derivation and application for linear systems. A well-known invariant property of linear systems is their direct current (dc) behavior. Such an invariant can be employed as an online testing capability if it can be utilized to estimate the relation between a set of input and output patterns. Verification of the relation between the inputs and outputs within a tolerance range will reveal the existence of faults in the circuit. Evaluation of the value of the dc gain of a system, the invariant $\mathcal{I}$, can be performed from its coefficients

$$\mathcal{I} = \frac{\sum_{k=0}^{M} b_k}{1 - \sum_{k=1}^{N} a_k}. \tag{2}$$

The invariant can be utilized to check the relation between the inputs and outputs as follows:

$$\mathcal{I} \times \sum_{n=-\infty}^{+\infty} x[n] - \sum_{n=-\infty}^{+\infty} y[n] = 0. \tag{3}$$

Such a relation, though accurate, cannot be utilized online as it would imply infinite error-detection latency. Though a relation over a finite set of inputs and outputs is feasible for online test, utilization of a limited set of inputs and outputs will introduce inaccuracy to the relation. The inaccuracy of the relation consequently introduces latency to fault detection. The relation between a set of C patterns is given by

$$\mathcal{I} \times \sum_{n=0}^{C-1} x[n] - \sum_{n=0}^{C-1} y[n] = \mathcal{T} \tag{4}$$

with the exact value of the tolerance given by (5) at the bottom of the page.

$\mathcal{T}$ depends on the system coefficients and the input to the system. While $\mathcal{T}$ is time-varying, by rearranging (5), a time-invariant upper bound on $\mathcal{T}$ can be found [see (6) at the bottom of the page].

In (6), $x_{\max}$ and $y_{\max}$ denote the maximum input and output signal magnitudes, respectively. While (6) provides an upper bound on $\mathcal{T}$, it is loose due to the assumption of independent output variance in determining the upper bound. The output depends on the input and cannot be arbitrarily chosen while determining a tight upper bound on $\mathcal{T}$.

A tight upper bound can be found in the case of FIR systems. The result attained for FIR systems can also be utilized in the case of IIR systems if the impulse response of IIR systems is truncated so as to convert it to an FIR system. When the number of responses employed is large enough, the effect of truncating the impulse response becomes negligible.

In the case of FIR systems, $a_k = 0, \forall k$. Therefore, (5) simplifies to

$$\sum_{k=0}^{M} b_k \left(\sum_{n=C-k}^{C-1} x[n] - \sum_{n=-k}^{-1} x[n] \right). \tag{7}$$

In this case, a tight upper bound can be found by rearranging (7) as

$$\mathcal{T}_{\max} = 2x_{\max} \sum_{n=1}^{M} \left| \sum_{k=n}^{M} b_k \right|. \tag{8}$$

Since for FIR systems, the filter coefficients are equal to the impulse response of the system, the truncated impulse response for IIR systems can be substituted for the filter coefficients b_k in (8). Online estimation of the invariant together with the tight

$$\mathcal{T} = \frac{\sum_{k=0}^{M} b_k \left(\sum_{n=C-k}^{C-1} x[n] - \sum_{n=-k}^{-1} x[n] \right) + \sum_{k=1}^{N} a_k \left(\sum_{n=C-k}^{C-1} y[n] - \sum_{n=-k}^{-1} y[n] \right)}{1 - \sum_{k=1}^{N} a_k} \tag{5}$$

$$\mathcal{T}_{\max} = \frac{2x_{\max} \sum_{n=1}^{M} \left| \sum_{k=n}^{M} b_k \right| + 2y_{\max} \sum_{n=1}^{N} \left| \sum_{k=n}^{N} a_k \right|}{1 - \sum_{k=1}^{N} a_k} \tag{6}$$

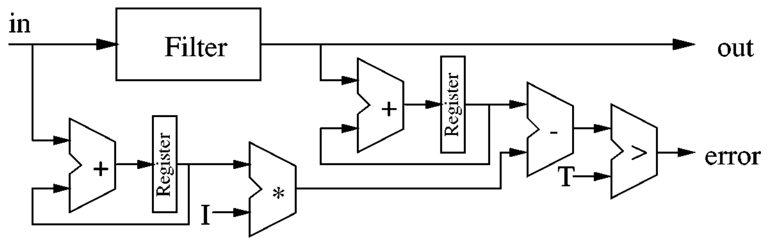


Fig. 2. Two-accumulator implementation.

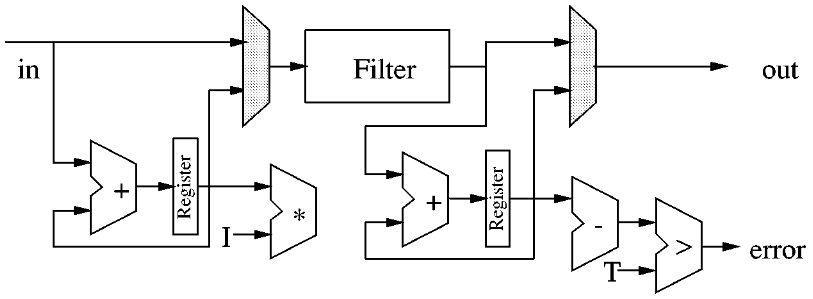


Fig. 3. Online and offline implementation.

upper bound on $\mathcal{T}$ can be employed for error detection as in the following form:

$$\left|\mathcal{I}\sum_{n=0}^{C-1} x_n - \sum_{n=0}^{C-1} y_n\right| \le \mathcal{T}_{\max}. \tag{9}$$

Equation (9) guarantees that the invariant relation between the input and output is continuously satisfied. The invariant is violated if a fault occurrence causes the output to deviate from its correct value. As a fault continues to be activated, the effect of the fault will be accumulated together with the output. If such an error accumulation happens to be *monotonic*, the accumulated fault effect will eventually exceed $\mathcal{T}_{\max}$. An analysis of fault accumulation and conditions imposed on the implementation style are outlined in Section VI.

Accumulation of fault effects is governed by two parameters: the average effect of a fault on the output F_{eff} and the activation probability of the fault P_{act}. The activation probability of a fault depends on the position of the fault and the input distribution. In general, the activation probability of faults at the least significant bits is higher than the activation probability of faults at the most significant bits [6]. The average fault effect on the output depends both on the behavior of the fault within the embedding register–transfer (RT) block and the data flow graph of the linear system. An analysis of the fault effects on the output is outlined in detail in Section VI. As the accumulated fault effect can be determined as the multiplication of P_{act} and F_{eff}, the condition for an error indication in the face of a fault can be given by

$$\left|\mathcal{I}\sum_{n=0}^{C-1} x_n - \sum_{n=0}^{C-1} y_n + CP_{\text{act}}F_{\text{eff}}\right| > \mathcal{T}_{\max}. \tag{10}$$

While detection of an error depends on the input distribution, a sufficient condition for detection of a fault, independent of the input distribution, can be shown to be

$$CP_{\text{act}}|F_{\text{eff}}| > 2\mathcal{T}_{\max}. \tag{11}$$

As long as the fault effects are accumulated in a monotonic manner, they will be detected independently of fault effect magnitudes or activation probabilities. Even in the case of nonmonotonicity, a small bias in one direction will allow detection of such faults, albeit for larger values of C. Yet this idealized mathematical analysis can be impacted by numerical inaccuracies. If the implementation utilizes roundoff schemes in order to keep the bitwidth of the system constant, numerical inaccuracies may accumulate over time to cause false alarms. Effects of numerical inaccuracies are examined in Section V.

A. Implementation Styles for Online Checking

While (9) can be implemented directly in the form shown, such an implementation necessitates two accumulator structures: one for the inputs and one for the outputs. Additionally, a multiplier is required in order to multiply the accumulated inputs with the invariant and a subtractor is necessitated in order to find the difference of the accumulated outputs from the product of the invariant times the accumulated inputs. Finally, a comparator is required in order to compare the resultant difference to the maximum tolerance level. Such an implementation is depicted in Fig. 2.

The accumulators added for concurrent error checking can be utilized for offline manufacturing testing [2]. The input accumulator can be converted to an arithmetic pattern generator with an additional multiplexer. The output accumulator, on the other hand, can be converted to an arithmetic response compactor with an additional multiplexer. Therefore, the two multiplexers, depicted shaded in Fig. 3, are enough to convert the concurrent test hardware into an offline BIST hardware. The downside of this implementation is that depending on the input distribution, the accumulated input and output may grow beyond the capacity of the accumulators. Therefore, the accumulators have to be constructed in such a way that they do not overflow until the pattern count is large enough so as to detect faults with small magnitude

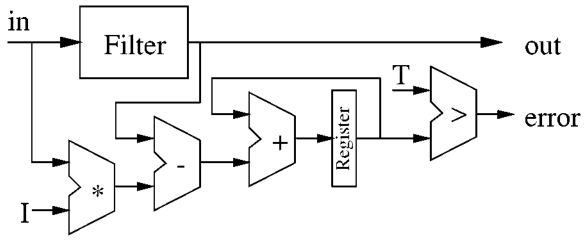


Fig. 4. One-accumulator implementation.

effects. Such a requirement imposes an unnecessarily high area overhead in the case of a nonzero mean input distribution.

Equation (9) can be modified in order to reduce the number of accumulators required. The implementation of the modified form, given in (12), requires only one accumulator, thus, significantly reducing the area requirement

$$\left|\sum_{n=0}^{C-1}(\mathcal{I}x_n - y_n)\right| \le \mathcal{T}_{\max}. \tag{12}$$

The one accumulator implementation is depicted in Fig. 4.

Additionally, in one accumulator implementations, the accumulated difference does not grow beyond the level of $\mathcal{T}_{\max}$ unless there indeed is a fault. However, the multiplication with the invariant has to be performed in full precision in order to prevent error accumulation. While in the previous implementation such a requirement was not imposed, the bitwidth of the accumulator was larger and the multiplication operation with truncation necessitated a comparable area. The drawback of the second implementation is that the concurrent error-detection hardware cannot be shared with an offline BIST scheme as proposed in [2], which depends on arithmetic pattern generators.

V. Numerical Inaccuracies

Errors due to the roundoff scheme can be modeled as their average bias effect per pattern on the output R_{err}. Therefore, the condition for the online checking hardware to indicate an error for a set of C patterns becomes

$$\left|\mathcal{I}\sum_{n=0}^{C-1}x_n - \sum_{n=0}^{C-1}y_n + CF_{\text{eff}}P_{\text{act}} + CR_{\text{err}}\right| > \mathcal{T}_{\max}. \tag{13}$$

The sufficient condition, independent of input distribution, thus becomes

$$C|F_{\text{eff}}|P_{\text{act}} + CR_{\text{err}} > 2\mathcal{T}_{\max}. \tag{14}$$

The fault indicator may raise a false alarm due to numerical inaccuracies, as they will accumulate to rapidly exceed $\mathcal{T}_{\max}$. Fig. 5 shows the case where truncation errors accumulate and exceed the tolerance indicated as horizontal lines in the figure. In this figure, the horizontal axis indicates the number of patterns and the vertical axis corresponds to the term inside the absolute value at the left-hand side of the inequality in (9). In the case of a nonzero R_{err}, false alarms can be eliminated by compensating for the bias effect by modifying the tolerance to $\mathcal{T}_{\max} + CR_{\text{err}}$. The inclined lines in Fig. 5 represent the modified tolerance. However, in the case of a modified tolerance, depending on the input distribution, either the faults with posi-

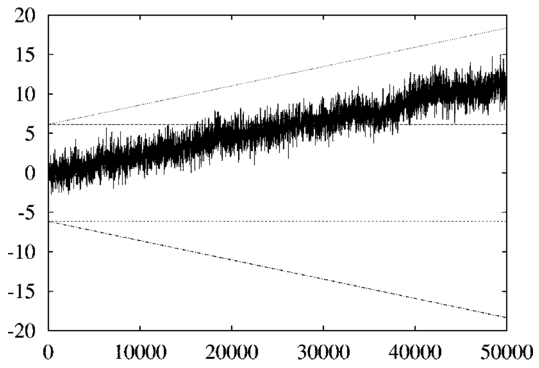


Fig. 5. Roundoff behavior: truncation.

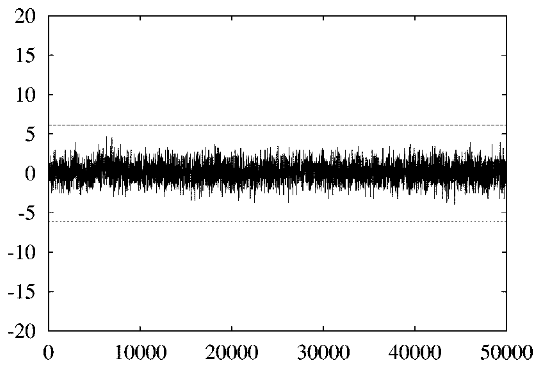


Fig. 6. Roundoff behavior: round-to-nearest odd.

tive or negative effects (negative effect in the case shown) will not be detected whenever they lose the race against numerical inaccuracies. In order for a fault to be invariably detected in the case of a nonzero R_{err}, the following condition needs to be satisfied:

$$|F_{\text{eff}}|P_{\text{act}} > R_{\text{err}} \tag{15}$$

This equation mostly affects faults at the least significant bits. Each fault in the circuit has an effect of $2^i R_{\text{err}}$ on the output depending on its bit position i. Assuming that the average activation probability of the faults at the least significant bits is 1/4 and that the effect of numerical inaccuracies is of the order of the magnitude level of the least significant bit, a coverage loss at the least significant 2–3 bits of the circuit will occur, unless the roundoff error averages to zero.

R_{err} can be reduced to zero if a nonbiased roundoff scheme is utilized, such as round-to-nearest odd (even). Fig. 6 empirically confirms the fact that such rounding schemes introduce no bias. While rounding introduces area overhead to the system, schemes with nonzero bias due to numerical inaccuracies necessitate larger accumulator bitwidths in order to accommodate accumulated inaccuracy bias. While the overheads due to both options can be analyzed and fault coverage may be traded off for reduced overhead, a zero bias is also desirable for system implementations and may already be part of the original implementation.

Even in the case of such a nonbiased roundoff scheme, the probability of error accumulation to an arbitrary level is not exactly zero. Ignoring such occurrences will introduce a possibility of false alarms to fault detection. Therefore, the effect that is due to distribution of the roundoff errors needs to be analyzed and the tolerance needs to be augmented in order to reduce occurrence of false alarms practically to zero.

Analysis of a fixed-point implementation of a linear system under the rounding scheme indicates that the average noise power at the output due to roundoff errors is $(N+M+1)(2^{-2B}/12)$, where B is the number of bits utilized in the implementation, excluding the sign bit. While roundoff errors have a uniform distribution in each operation ranging from $-(2^{-B}/2)$ to $2^{-B}/2$, the distribution of the accumulated errors converges to a Gaussian distribution. The parameters μ and σ of the Gaussian distribution for a set of C patterns can be approximated as

$$\mu = 0, \quad \sigma = \frac{\sqrt{C(N+M+1)}2^{-B}}{\sqrt{12}}. \tag{16}$$

In a Gaussian distribution, the probability of occurrence of a result over 6σ can be ignored. In the case of full precision arithmetic, the faults at the least significant bit positions require

$$\mathcal{L} = \frac{\mathcal{T}_{\max}2^B}{P_{\text{act}}} \tag{17}$$

patterns in order to be detected. However, numerical inaccuracies can cause false alarms if their effect is not taken into account. The tolerance $\mathcal{T}_{\max}$ needs to be augmented in order to prevent false alarms as

$$\mathcal{T}'_{\max} = \mathcal{T}_{\max} + 6\sigma. \tag{18}$$

Consequently, the number of patterns required for detection becomes

$$\mathcal{L}' = \frac{\left(\mathcal{T}_{\max} + 6\frac{\sqrt{\mathcal{L}'(N+M+1)}2^{-B}}{\sqrt{12}}\right)2^B}{P_{\text{act}}}. \tag{19}$$

The solution of $\mathcal{L}'$ from (19) is

$$\mathcal{L}' = \mathcal{L} + \sqrt{\mathcal{L}}\frac{\sqrt{3(N+M+1)}}{P_{\text{act}}}\sqrt{\left(1+\frac{3(N+M+1)}{4P_{\text{act}}\mathcal{T}_{\max}2^B}\right)} + \frac{3(N+M+1)}{2P_{\text{act}}^2}. \tag{20}$$

A simple approximate solution of $\mathcal{L}'$, using the fact that $(N+M+1) \ll \mathcal{T}_{\max}2^B P_{\text{act}}$, can be attained

$$\mathcal{L}' \approx \mathcal{L} + \sqrt{\mathcal{L}}\frac{\sqrt{3(N+M+1)}}{P_{\text{act}}}. \tag{21}$$

The same approximate solution can be attained by replacing $\mathcal{L}'$ on the right-hand side of (19) with $\mathcal{L}$. While modification of the tolerance due to numerical inaccuracies increases detection latency, the effect is quite small. In the case of 14-bit fixed-point implementation, the latency for the faults at the least significant bits increases only by 10%. While the increase in latency is not significant, including such analysis for determination of the maximum tolerance level eliminates any false alarm possibility. Since the analysis requires only the noise level introduced by a specific implementation, the tolerance can be augmented for various implementations easily as the noise level introduced by roundoff schemes for most implementation styles is available in the literature.

VI. Monotonicity

Detection of faults in the proposed scheme depends on fault-effect accumulation. If the effect of a fault on the output is always in the same direction, the effect of successive activations of the same fault will be accumulated at the output. A fault is said to be *monotonic* if it always changes the output in the same direction. Assuming that the response of a fault-free circuit to an input x is $g(x)$, if the fault is *monotonic*, the output of the faulty circuit $g_f(x)$ has to satisfy the following condition:

$$(g_f(x) \geq g(x)\ \forall x) \vee (g_f(x) \leq g(x)\ \forall x). \tag{22}$$

The equality case in (22) holds when the current inputs do not activate the faults or when the fault effect is not accumulated to the output. As such a definition is valid for combinational circuits only, the definition needs to be extended to sequential circuits. For sequential circuits, the inputs (outputs) cannot be defined at each time step separately; they need to be defined over time with each value associated with a time stamp, such as $x[n]$ $(y[n])$. In that case, the requirement becomes

$$(y_f[n] \geq y[n]\ \forall x[n]) \vee (y_f[n] \leq y[n]\ \forall x[n]). \tag{23}$$

Even though the condition in (23) is a straightforward extension of (22), most circuits will not satisfy such a strong requirement. While this requirement is sufficient for fault-effect accumulation, it is not necessary. A relaxed requirement for monotonicity of faults in sequential circuits, as it is used in this paper, can be defined as

$$\left(\sum y_f[n] \geq \sum y[n]!\forall x[n]\right) \vee \left(\sum y_f[n] \leq \sum y[n]\forall x[n]\right). \tag{24}$$

Enforcing such a condition for all the faults in a circuit may be computationally complex. If the circuit hierarchy is utilized, complexity could be reduced. The following conditions when applied at two levels of circuit hierarchy are enough to guarantee monotonicity of fault, as defined by (24).

1) Effect of a fault is propagated monotonically from the output of the RT component embedding the fault to the output.
2) Effect of a fault in an RT component is propagated monotonically to the output of the RT component embedding it.

Examination of the former RT level condition shows that in the case of FIR systems, a fault effect, unless it is in the primary inputs, will impact the output of the system during a single time step only. The linearity of such systems, therefore, guarantees that the fault effect will be propagated to the output monotonically, as defined by (22). In the case of IIR systems, a fault impacts the output during more than one clock cycle as the system has IIR. Using the modified monotonicity requirements of (24), it can be shown that the monotonicity requirement for linear systems transforms to the following condition:

$$\frac{1}{1-\sum_{k=1}^{N} a_k} \neq 0. \tag{25}$$

This condition is trivially satisfied as the left-hand side of the equation cannot equal zero unless at least one of the coefficients is ∞. Nonetheless, the closer the left-hand side approaches zero, the larger the latency for fault detection. The latter concern can be alleviated by the observation that a linear system is stable as long as the poles of the system lie within the unit circle [12], thus guaranteeing that $|1-\sum_{k=1}^{N} a_k| < 1$. Therefore, for practical, thus, necessarily stable linear system implementations, the first monotonicity condition does not pose any restrictions on the implementation.

The condition imposed on the gate-level implementation necessitates detailed analysis of the gate-level implementations of RT components that are utilized in digital linear systems. Such systems possess multiplication, addition, and delay components only. Multiplications in linear systems are constant and are usually implemented through a set of shift-add operations. Delay components are quite simple and can easily be shown to have monotonic implementations. As the main components of such systems are adders and subtractors, we investigate the existence of monotonic implementations for adders and subtractors. Since adders and subtractors are quite similar in functionality and implementation, in this paper, only the analysis for adders is outlined. Before analyzing various adder implementations, we prove the existence of *monotonic* implementations of monotonic logic functions in the following subsection.

A. Monotonicity Analysis of Two-Level Logic

In this section, we prove that all two-level AND/OR implementations of monotonic logic functions are *monotonic*.

Theorem: All two-level AND/OR implementations of monotonic logic functions are *monotonic*.

Proof: In a two-level AND/OR implementation, there exist only five categories of faults: faults at the inverters, faults at the inputs and outputs of AND gates, and faults at the inputs and output of the OR gate. We first show that the faults in each of these categories are monotonic.

1) *Faults at the Inputs of the AND Gates:* Stuck-at-zero faults essentially eliminate the minterm implemented by that AND gate and, thus, always reduce the output from one to zero. Stuck-at-one faults eliminate the literal and, thus, expand a minterm by always increasing the output from zero to one.
2) *Faults at the Outputs of the AND Gates:* The above argument for the stuck-at-zero faults holds in this case as well. Stuck-at-one faults force the function to a tautology, thus, strictly increasing the output.
3) *Faults at the Inputs of the OR Gate:* Stuck-at-one faults force the function to a tautology, thus, strictly increasing the output. Stuck-at-zero faults have the same effect as stuck-at-zero faults at the outputs of AND gates.
4) *Faults at the Outputs of the OR Gate:* The stuck-at-zero (one) fault at the output of the OR gate forces the function always to zero (one), thus, strictly reducing (increasing) the output.
5) *Inverter Faults:* The effects of these faults are the same as having stuck-at-zero/one faults at the inputs of multiple AND gates at the same time and, therefore, still monotonic.

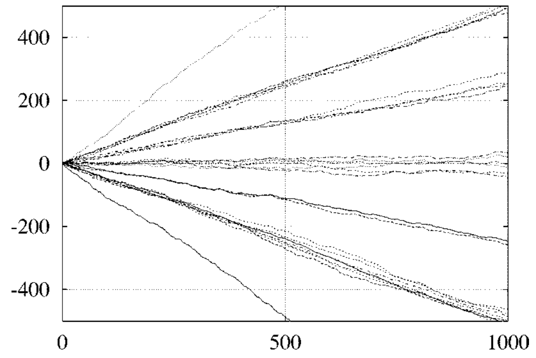


Fig. 7. Average behavior of full-adder faults.

While monotonicity of the faults in each of the above five categories is, thus, shown, faults affecting multiple stems of the same line with opposing polarities can possibly still violate monotonicity of fault effects. Yet as multiple stem faults in two-level logic can strictly occur at the inputs of AND gates and, consequently, only at the inputs of the function, the monotonicity requirement in the logic function guarantees that the impact of this type of faults on the output is monotonic. ■

The above proof *guarantees* the existence of monotonic implementations of adders as they are linear. Yet the higher cost of two-level implementations of adders forces us to search for monotonic implementations, which have area comparable to standard implementations. The following sections provide such analysis for ripple-carry and carry-look-ahead adders.

B. Ripple-Carry Adders

Ripple-carry adders can be decomposed into full-adder cells. The fault effects propagate from the outputs of full adder cells to the outputs of the ripple-carry adder monotonically as the adders are linear. Therefore, guaranteeing monotonicity of full adders will guarantee monotonicity of ripple-carry adders.

The faults internal to full-adder cells may behave nonmonotonically depending on the implementation. Fig. 7 provides a visualization of the error accumulation produced by the complete set of all possible stuck-at faults of a standard full-adder implementation, depicted in Fig. 9(a). While some faults produce an effect that averages to zero, others produce a monotonic effect that is accumulated over time and, thus, provide a means of online error detection.

A full-adder implementation with AND/OR/NOT gates, shown in Fig. 9(b), produces improved monotonic faulty behavior. The implementation shown in Fig. 9(b) has only three nonmonotonic faults $(nx@1, ny@1, (x \oplus y)@1)$.

The analysis of the faults $(ny@1, nx@1)$ shows that both of these faults require the same two patterns $(x = 1, y = 1, c_i = 0/1)$. Whenever the carry input is "0," the output of the adder increases from 2 ("10") to 3 ("11"); similarly whenever the carry input is "1," the adder output decreases from 3 ("11") to 2 ("10"). Thus, these faults produce *equi-bidirectional*[1] effects. These two faults manifest themselves to the $(x \oplus y)$ logic as if the ("11") input is mapped to ("10") or ("01"), depending on the fault. If the same case could be made for the carry logic, the output would have been reduced regardless of the carry input. However, the carry propagate logic is not affected by these faults, and therefore bidirectionality ensues.

Modification of the logic so as to allow propagation of these fault effects to the carry output by implementing the carry propagate (xy) logic as $(x'y' + x \oplus y)'$, shown in Fig. 9(c), moves these faults to the monotonic fault list. Nonetheless, there does exist a fault $(x \oplus y)@1$ that such a modification does not resolve. The proposed modification reduces the remaining percentage of nonmonotonic faults to less than 2.5%.

In implementations that are similar to the standard-full adder implementation, the nonmonotonic fault cannot be transformed to a monotonic one. The fault $(x \oplus y)@1$ behaves nonmonotonically due to the late introduction of the carry input to the circuit. For the input combinations of $x = 1, y = 1, c = 0$ and $x = 1, y = 1, c = 1$, the *sum* output increases and diminishes, respectively, under the existence of the fault $(x \oplus y)@1$ and the *carry* output is always one independent of the fault occurrence. Therefore, the overall output of a full-adder behaves nonmonotonically under the existence of the fault $(x \oplus y)@1$ independent of implementation.

In a ripple-carry adder implementation, the inputs are initially added in parallel by the input stage of full-adders and then the carry is propagated through the carry chain. Introduction of the carry input to the full-adder circuit at the same level as primary inputs, though it would eliminate all nonmonotonic faults, would increase the latency of the addition operation due to the increased length of the path that the carry has to propagate through the carry chain. Therefore, eliminating the last nonmonotonic fault $(x \oplus y)@1$ without introducing significant area or delay overhead is not possible.

A two-level implementation of the full-adders, though somewhat more expensive in terms of area, provides complete monotonicity for ripple-carry adder implementations. Fig. 8 shows the outcome of accumulated errors produced by a two-level full adder implementation. As it can easily be seen, all faults produce a nonzero mean average. The difference of the increase in the mean can be explained by the fact that the activation probabilities and magnitude effects of faults vary. With a uniform distribution at the inputs of an adder, the activation probability of faults varies between $1/8$ to $1/2$.

C. Carry-Look-Ahead Adders

The complex behavior of carry-look-ahead adders, compared to ripple-carry adders, suggests that there might be a higher number of nonmonotonic faults, as is substantiated by the following analysis. The faults causing nonmonotonicity are mainly due to the separation of carry generation and addition logic. The addition stage $(P_i \oplus c_i)$ includes an XOR functionality, which is nonmonotonic. In this implementation, there exist 61 faults with nonmonotonic behavior, out of which 49 display equi-bidirectional behavior. The logic affected by the equi-bidirectional behavior is shown in bold in Fig. 10.

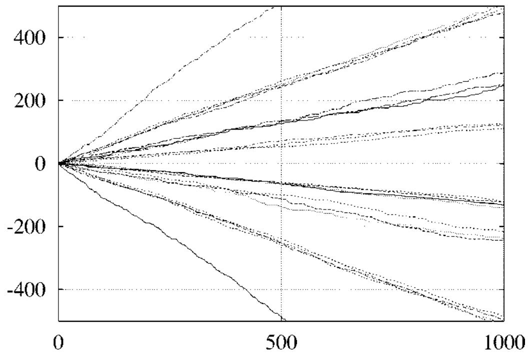


Fig. 8. Average behavior of full-adder faults.

We hereby introduce a carry-look-ahead adder implementation, wherein the functionality of P (propagate term) is slightly modified in order to skew the bidirectional effects of most faults

$$p_i = x + y \tag{26}$$

$$c_{i+1} = \mathbf{p_i c_i} + g_i \tag{27}$$

$$s_i = c_i p_i' + c_i' p_i g_i' + \mathbf{p_i c_i} g_i. \tag{28}$$

In this implementation, a part of the logic that generates c_{i+1}, i.e., $\mathbf{p_i c_i}$ is utilized in the implementation of s_i and, therefore, the bidirectional effect of the $(P_{i+1} \oplus c_{i+1})$ addition stage is skewed. By itself, this modification does provide a better biased faulty behavior. However, an even better solution is to intermix the original implementation of P terms with the proposed implementation such that P_0 and P_2 are implemented as $x \oplus y$ and P_1 and P_3 are implemented as $x + y$. The $(P_i \oplus c_i)$ logic is modified as in (28) for P_1 and P_3. In this implementation, 29 faults still have nonmonotonic behavior; however, only two faults ($P_0@1$ and $P_2@1$) still have equi-bidirectional behavior, resulting in a possible detection loss of 1.2%.

VII. Experimental Results

Initially, a 13-tap FIR system is designed in order to verify the fault detection capability of the proposed scheme. A fractional number system of the form 1.9 (1 for the sign and 9 bits for the fraction) was selected for the input and a 1.13 form was utilized for the output of the circuit. In this implementation, ripple-carry adders that are composed of standard adders are employed. The analysis in Section V indicates that some fault-coverage loss is to be expected for this design due to nonmonotonic faults. Simulation results shown in Fig. 11 indicate that the fault coverage exceeds 99.5%. However, some of the nonmonotonic faults in this implementation are not detected. Analysis of the magnitude effect of the faults that are detected indicates that faults with larger magnitude effects are detected earlier than faults with smaller magnitude effect. Nonmonotonic faults also produce a similar pattern. Nonmonotonic and bidirectional faults with large magnitude are eventually detected, as the probability of occurrence of short biased subsequences in an input sequence increases directly with the number of patterns. However, such a probability increase is not enough for detection of low magnitude nonmonotonic faults.

[1]Faults that produce positive and negative fault effects at the output of the component in the exact same number of input combinations are called equi-bidirectional; these are a subset of nonmonotonic faults.

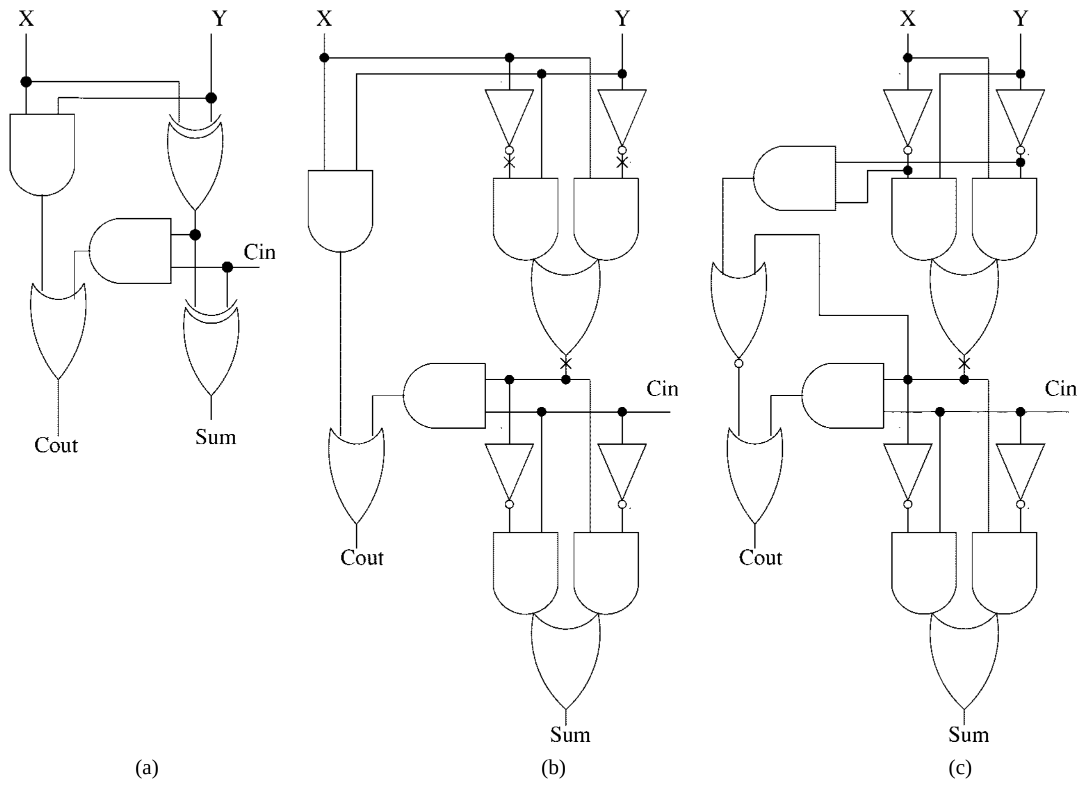


Fig. 9. Full-adder implementations. (a) Standard implementation. (b) AND/OR/NOT implementation. (c) Improved implementation.

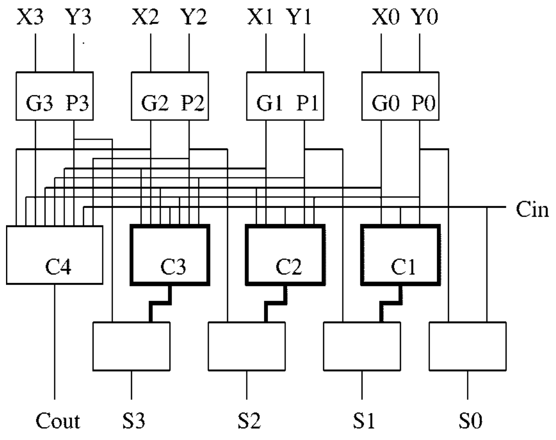


Fig. 10. Carry-look-ahead adder.

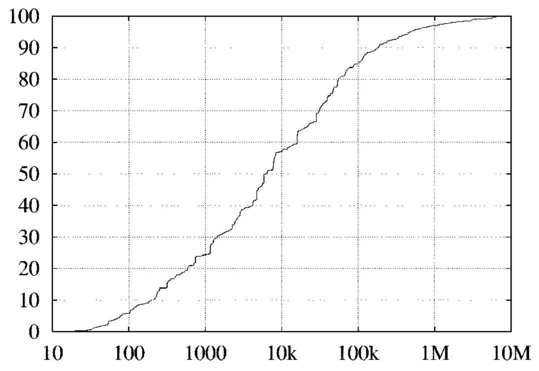


Fig. 11. Fault-simulation results for the FIR system with standard full-adder implementation.

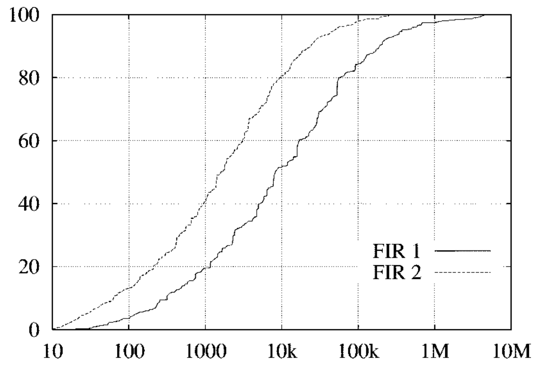


Fig. 12. Fault-simulation results for FIR systems with two-level full-adder implementation.

The same FIR design is implemented by utilizing monotonic full-adder cells in order to verify that the proposed scheme is capable of detecting all faults for *monotonic* implementations. The simulation results in this case indicate that 100% fault coverage is attained; not a single fault is missed.

Another *monotonic* FIR system is designed with inputs of the form 1.7 and outputs of the form 1.12 in order to validate the effect of the bitwidth of the filter on the fault detection latency, as determined by (17). Fault simulation is performed on this implementation and simulation results in Fig. 12 indicate that latency is reduced with the reduced filter bitwidth as predicted.

The capability of the online test scheme is so far validated on two FIR systems. In order to verify the capability on general IIR systems, an IIR system is implemented with monotonic adders as well. The same fractional number system as in the first FIR system, 1.9, is employed for this implementation. Fig. 13 indicates that near 100% coverage is attained for IIR circuits within 10 million patterns. Investigation of the fault detection loss indicates that all undetected faults reside in the multipliers that

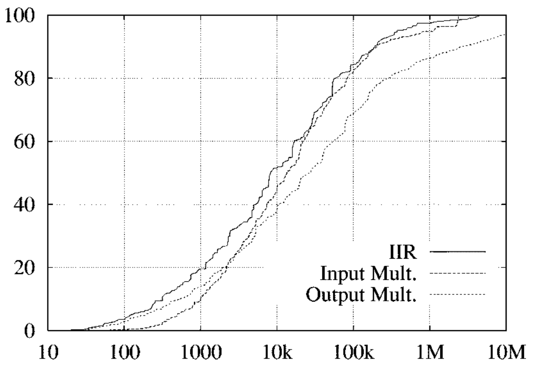


Fig. 13. Fault-simulation results for IIR system.

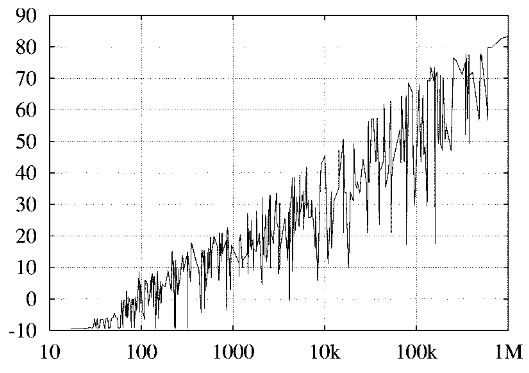


Fig. 14. SNR caused by the detected faults in FIR 1.

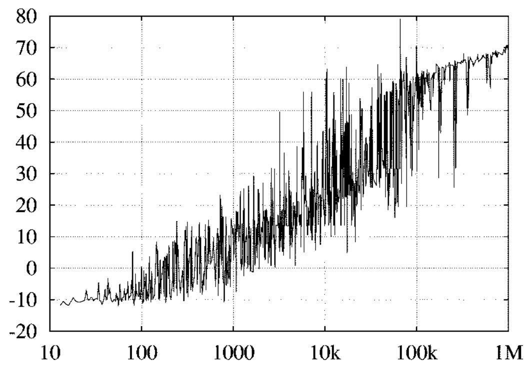


Fig. 15. SNR caused by the detected faults in IIR.

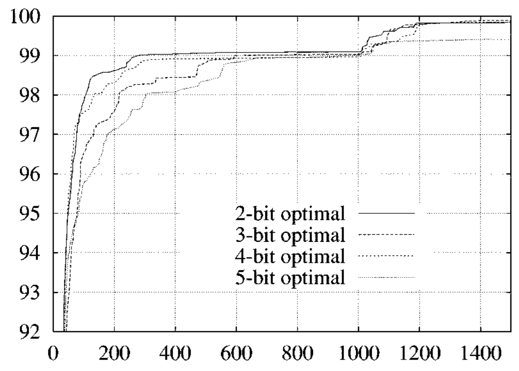


Fig. 16. Fault coverages for offline test.

TABLE I
GATE COUNTS FOR SYSTEM AND ERROR DETECTION HARDWARE

	Circuit		Test Overhead				
	Gates	FFs	Gates	FFs	%	$\mathcal{T}_{max}$	Patterns
FIR 1	4394	169	1096	24	23.5	6.15	846,000
FIR 2	3014	139	411	19	13.6	0.56	26,000
IIR	4434	67	876	24	20.7	1.90	250,000

multiply the output by the coefficients, a_k. Loss of fault detection in this case is caused by the reduced activation of faults, as the output of the circuits is not as random as the input. As the number of activations is reduced, accumulation of fault effects so as to violate the invariant takes longer.

The hardware cost of the online checking circuit is independent of the size of the circuit; it depends strictly on the hardware requirements for the multiplier used to multiply the input with the invariant. All circuits have been synthesized from a data flow graph description and optimized at the gate level by a set of tools developed in C in the course of this work. Fault simulations have been performed by HOPE [9]. Table I reports the number of gates utilized in the implementation of the linear systems and error-detection hardware. The flip-flops, denoted as FF in the table, are assumed to be equivalent to four gates in calculating overhead percentiles.

As can be seen in Fig. 12, 100% fault coverage is invariably achieved for FIR circuits with an area cost that ranges for small circuits between an eighth to a quarter of the basic design. As the area cost of the proposed concurrent test is a constant function of design size, typical circuit sizes of 64 taps and 25 000 gate equivalents exhibit approximate area costs of less than 5%.

Table I also provides the values for $\mathcal{T}_{\max}$ and the worst case number of patterns, computed by (21), to detect faults at the least significant bit position of the filters, assuming that the fault activation probability is 1/8. As faults at the higher bit positions are detected earlier, the pattern counts provided in Table I are enough to detect all the faults in the filters excluding the faults in the rounding logic. As the activation probability of the faults in the rounding logic is significantly lower, their detection requires higher number of patterns. Average power analysis for the fault effects confirms the fact that only faults in the rounding logic remain undetected at the pattern counts provided in Table I.

An analysis of the average power of the fault effects indicates that as the number of patterns increases, the SNR of the undetected faults increases, as shown in Figs. 14 and 15. It is apparent from the figures that faults with large magnitude and higher activation rates are detected earlier. Fig. 14 also shows that an SNR comparable to the SNR of numerical inaccuracies is achieved at around 100 000 patterns for FIR 1.

While performing offline test, a test signal causes the filter inputs to be supplied from the arithmetic pattern generator. The initial seed and increment for the pattern generator are supplied from the circuit inputs. The outputs are compacted at the output accumulator and observed at the output at the end of the off line test session through the output multiplexer. The fault-simulation results for offline test are presented in Fig. 16 for four types of arithmetic pattern generators [13]. The first 1000 patterns are standard ABIST patterns. As can be seen in Fig. 16, the fault-coverage curves level off after 600 patterns. This is due to the fact that ABIST is not capable of activating faults at the sign bit of most adders. These faults usually require high-variance input patterns [6]. Therefore, a set of high variance patterns generated by the same offline hardware with an increment of "1111110111" is applied in order to detect the remaining sign bit faults. The fault-coverage levels, except for the 5-bit subspace optimal case, exceed 99.9%.

VIII. Conclusion

As increasing levels of electronic solutions permeate society, it becomes imperative to ensure that operational correctness be at least confirmed and any errors isolated, lest malfunctions avalanche through frequently highly interconnected systems of electronic solutions. Yet the high cost of even concurrent error detection through duplication-based mechanisms has confined the applicability of such techniques to only the most critical niches. Invariant-based approaches provide a compact way of ensuring the health of the system concurrently; yet frequently the invariants may have only a sparse coverage of the functional and certainly of the structural faults. Numerical inaccuracies furthermore complicate the applicability of such exact schemes, as they pose strict tradeoffs between false alarms and coverage.

We introduce in this paper an invariant-based scheme for complete concurrent fault identification for both cyclic and acyclic impulse response systems. Time-extended checking of the invariant enables detection of errors through accumulation, even at the least significant bits while round-off errors are prevented from introducing false alarms, as long as zero-bias schemes are utilized. Examinations of RT-level and gate-level designs of the linear systems can be utilized to identify design conditions that enable 100% concurrent fault coverage. Fortuitously, the hardware requirements of the proposed concurrent check can be shown to overlay highly popular pseudorandom test approaches, thus reducing the cost of complete concurrent and very high offline fault coverage to levels comparable to scan for typically large industrial designs. Experimental results for FIR and IIR systems confirm the mathematical analysis outlined in this paper.

References

[1] C. J. Anfinson and F. T. Luk, "A linear algebraic model of algorithm-based fault tolerance," *IEEE Trans. Comput.*, vol. 37, pp. 1599–1604, Dec. 1988.

[2] I. Bayraktaroglu and A. Orailoglu, "Unifying methodologies for high fault coverage concurrent and offline test," in *Proc. Int. Symp. Circuits and Systems*, May 2000, pp. 705–708.

[3] A. Chatterjee and M. A. d'Abreu, "The design of fault-tolerant linear digital state variable systems: Theory and techniques," *IEEE Trans. Comput.*, vol. 42, pp. 794–808, July 1993.

[4] A. Chatterjee and R. K. Roy, "Concurrent error detection in nonlinear digital circuits using time-freeze linearization," *IEEE Trans. Comput.*, vol. 46, pp. 1208–1218, Nov. 1997.

[5] Y. H. Choi and M. Malek, "A fault-tolerant FFT processor," *IEEE Trans. Comput.*, vol. 37, pp. 617–621, May 1988.

[6] L. Goodby and A. Orailoğlu, "Variance mismatch: Identifying random-test resistance in DSP datapaths," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, May 1996, pp. 3205–3208.

[7] K. H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Trans. Comput.*, vol. C-33, pp. 518–522, June 1984.

[8] J. Y. Jou and J. A. Abraham, "Fault-tolerant FFT networks," *IEEE Trans. Comput.*, vol. 37, pp. 548–561, May 1988.

[9] H. K. Lee and D. S. Ha, "HOPE: An efficient parallel fault simulator," in *Proc. IEEE Design Automation Conf.*, June 1992, pp. 336–340.

[10] V. S. S. Nair and J. A. Abraham, "Real-number codes for fault-tolerant matrix operations on processor arrays," *IEEE Trans. Comput.*, vol. 39, pp. 426–435, Apr. 1990.

[11] A. L. N. Reddy and P. Banerjee, "Algorithm-based fault detection for signal processing applications," *IEEE Trans. Comput.*, vol. 39, pp. 1304–1308, Oct. 1990.

[12] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

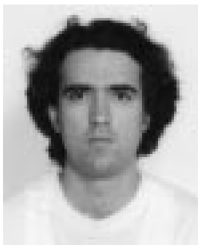
[13] J. Rajski and J. Tyszer, *Arithmetic Built-In Self-Test for Embedded Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[14] J.-L. Sung and G. R. Redinbo, "Algorithm based fault-tolerant synthesis for linear operations," *IEEE Trans. Comput.*, vol. 45, pp. 425–438, Apr. 1996.

[15] D. L. Tao and C. R. P. Hartmann, "A novel concurrent error detection scheme for FFT networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, pp. 198–221, Feb. 1993.

[16] B. Vinnakota and N. K. Jha, "A dependence graph-based approach to the design of algorithm-based fault-tolerant systems," in *Proc. Fault-Tolerant Computing Symp.*, June 1990, pp. 122–129.

[17] S.-J. Wang and N. K. Jha, "Algorithm-based fault tolerance for FFT networks," *IEEE Trans. Comput.*, vol. 43, pp. 849–854, July 1994.



Ismet Bayraktaroglu (S'98) received the B.S. and M.S. degrees in electrical engineering from Bogazici University, Istanbul, Turkey, in 1994 and 1996, respectively. He is currently working towards the Ph.D. degree in computer engineering at the University of California, San Diego.

His current research interests include built-in self-test (BIST), diagnosis of BIST designs, and concurrent test of digital signal processors.

Alex Orailoglu (M'84) received the S.B. degree (*cum laude*) in applied mathematics from Harvard University, Cambridge, MA, and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana-Champaign.

From 1983 to 1987, he was a Senior Member of Technical Staff at Gould Research Laboratories, Rolling Meadows, IL. In 1987, he joined the University of California, San Diego, where he is currently a Professor in the Computer Science and Engineering Department. His research interests include digital and analog test, fault-tolerant computing, computer-aided design, and embedded processors.

Prof. Orailoglu is a member of the IEEE Test Technology Technical Council (TTTC) Executive Committee and currently serves as Technical Activities Committee Chair and Planning Co-Chair of TTTC. He serves in numerous technical and organizing committees, including the International Test Conference and the VLSI Test Symposium, and has served as the Technical Program Chair of the 1998 High Level Design Validation and Test (HLDVT) Workshop and as the General Chair of HLDVT in 1999.