COST EFFECTIVE DIGITAL FILTER DESIGN FOR CONCURRENT TEST

Ismet Bayraktaroglu and Alex Orailoglu

Computer Science & Engineering Department
University of California, San Diego
La Jolla, CA 92093
ibayrakt,alex@cs.ucsd.edu

ABSTRACT

Invariant-based concurrent test schemes can provide economical solutions to the problem of concurrent test of digital filters. Design methodologies for digital filters ensuring concurrent testability are outlined. Experimental results confirm through fault simulation 100% fault coverage within area cost comparable to that of DfT for off-line test and with error detection latency well below human response times.

1. INTRODUCTION

Attaining digital filter designs that can provide 100% fault coverage during concurrent testing in a cost-effective manner is highly challenging. While full information duplication largely meets the challenge, it does so at inordinate cost. Schemes such as parity [11] reduce the cost of implementation, yet such cost reduction is achieved inherently by sacrificing portions of fault coverage. A hasty examination may lead to the conclusion that achieving concurrent-testability for every possible fault at anything less than full computational duplication would be unattainable. Indeed current invariant-based schemes consistently sacrifice a significant portion of fault-coverage [7]. We show in this paper that for digital filters utilization of a compact functional invariant coupled with rigorous mathematical and experimental analysis can provide 100% fault coverage in concurrent testing.

Achieving 100% fault coverage in the case of digital filters necessitates not only the identification of a functional invariant but furthermore requires attention to a number of practical, implementation aspects. Fixed-point computations are prone to numerical inaccuracy effects; unless the calculation in question is identically duplicated, the stringent requirements of a functional invariant comparison would produce nothing but false alarms. While a strict equality checking invariant comparison can be made to work for time-extended computations, prevalent in digital filters, it results in a shift of the time window for the computation of the invariant. The pipelined structure of the computation and sharing of intermediate results can be duplicated in the invariant to account for the time shift, but only at high cost. Alternative invariant approaches, such as tolerance-based ones, need to be examined to handle time-extended computations.

While reliance on tolerance-based invariants resolves a number of problems, it introduces in turn a number of further challenges that need to be examined. Since one can no longer rely on a clear-cut, instantaneous invariant comparison, issues both of fault capture and of latency arise. Fault capture necessitates computation of a precise tolerance, which results neither in false alarms nor in inordinate latency. A fault occurrence that results in a consistently unidirectional fault effect accumulation violates the invariant rapidly, while an equally bidirectional fault effect accumulation will result in such faults never violating the invariant. Design approaches that ensure that every possible fault possesses such monotonicity of fault effect accumulation introduce new perspectives on test-oriented digital filter design. Ensuring that such fault effect accumulation is preserved even under typical perturbations imposed by numerical round-off schemes and that furthermore faults in round-off hardware themselves can be completely covered adds to a list of requirements that need to be achieved in order to provide a truly cost-effective, completely concurrent testable design for digital filters. As latency is unavoidable in tolerance-based schemes, latencies that are inversely proportional to fault magnitude effect would be additionally highly desirable, as they ensure that large magnitude faults, possibly capable of destabilizing the overall system, are the ones most rapidly caught.

In this paper, we examine the issue of providing complete concurrent fault coverage for digital filters in a cost-effective manner. We propose a time-extended, cost-effective invariant and outline tolerance computations for capturing all possible stuck-at faults. We outline design issues for ensuring complete concurrent testability and indicate appropriate design approaches to ensure no fault capture loss. Experimental results confirm the outlined theoretical treatment.

2. PREVIOUS WORK

Concurrent test approaches can be grouped in two main categories: test vector and invariant-based approaches. Test vector based approaches, such as the input vector monitoring technique introduced in [10], require observation of a predetermined set of test vectors during normal operation of a combinational circuit. The set of vectors could either be compact ATPG patterns, or the complete input space of the circuit as in [10]. In both cases, the time required to observe all the vectors may be significant and area overheads incurred for either storage of ATPG patterns or recording of the observed pattern sequences may be high.

In the case of an invariant-based approach, the invariant could be the function itself as in duplication [5], or a linear approximation of a non-linear function [2]. In the case of high performance designs that are implemented by residue

This work is being supported by Hughes Space and Communications and the UC Micro Program.

number systems, a redundant residue [3] can be utilized to provide an invariant. In designs that implement a certain algorithm, such as FFT or QR factorization, an invariant property of the algorithm can be utilized. In [7], for example, *Parseval's theorem* [8] is utilized for the FFT algorithm so as to achieve concurrent detection of faults.

3. ON-LINE TEST IMPLEMENTATION

In this work, the DC gain [8] of a digital filter is utilized as an invariant in order to provide on-line fault detection. In an on-line test implementation various properties of invariants, such as computational complexity and fault detection capabilities, play an important role. While an invariant that can be simply evaluated results in low area overhead, invariants with no fault masking provide high levels of fault coverage.

Evaluation of the DC gain of a filter can be performed off-line from the filter coefficients and on-line from the inputs and outputs of the filter, thus providing a way to ensure the continued well-being of the system. The functionality of an FIR filter is defined by the following equation.

$$y[n] = \sum_{k=0}^{M} h_k x[n-k] \tag{1}$$

In this equation, the h_k's constitute the coefficients that govern the output characteristics of the filter. While the DC gain of the filter can be computed as $\sum_{k=0}^{M} h_k$, on-line computation of the DC gain can be performed by taking the infinite summation of both sides of equation 1, and then manipulating the resultant equation.

$$\sum_{k=0}^{M} h[k] = \frac{\sum_{n=-\infty}^{+\infty} y[n]}{\sum_{n=-\infty}^{+\infty} x[n]} \tag{2}$$

While equation 2 provides an exact estimation of the DC gain, in reality, estimation of the DC gain from a limited set of patterns introduces inaccuracy. The relation between the DC gain computed on-line and off-line is given by

$$\sum_{n=0}^{N} y[n] = \sum_{k=0}^{M} h_k \sum_{n=0}^{N} x[n] + \mathcal{T} \tag{3}$$

where the tolerance, $\mathcal{T}$, is equal to

$$\sum_{k=0}^{M} h_k \sum_{n=-k}^{-1} x[n] - \sum_{k=0}^{M} h_k \sum_{n=N-k+1}^{N} x[n] \tag{4}$$

Equation 3 provides a simple relation between the accumulated input and the output of a digital filter within a small tolerance range. A tight upper bound for the tolerance is obtained, necessary for latency reduction, by rearranging equation 4

$$\mathcal{T}_{max} = 2x_{max} \sum_{n=1}^{M} \left| \sum_{k=n}^{M} h_k \right| \tag{5}$$

where x_{max} denotes the maximum input signal magnitude. $\mathcal{T}_{max}$ depends only on filter coefficients and the maximum input magnitude. Equation 3 can then be utilized in the following inequality form for on-line checking of the filter circuitry.

$$\left| \sum_{n=0}^{N} \left(y[n] - x[n] \sum_{k=0}^{M} h_k \right) \right| \leq \mathcal{T}_{max} \tag{6}$$

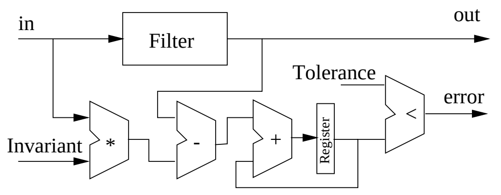

Figure 1: On-line test implementation

This equation ensures against false alarms if one assumes infinite precision arithmetic; nonetheless, bias introduced by numerical inaccuracies may in time cause the system to produce false alarms in practice. Consistent accumulation of fault effects so as to exceed the threshold value, $\mathcal{T}_{max}$, needs to be ensured. We present a solution to both of these problems in the following section.

4. IMPACT OF ROUND-OFF SCHEME

We model fault effects as a multiplication of the fault effect magnitude, F_{eff}, by the activation probability of that particular fault, P_{act}. The errors due to the round-off scheme can be modeled as their average bias effect per pattern on the output, R_{err}. Therefore, the sufficient condition for the on-line checking hardware to indicate an error is:

$$NR_{err} + NF_{eff}P_{act} > \mathcal{T}_{max} \tag{7}$$

The fault indicator may raise a false alarm due to numerical inaccuracies, as they will accumulate to rapidly exceed $\mathcal{T}_{max}$. In the case of a non-zero R_{err}, false alarms can be eliminated by compensating for the bias effect by modifying the tolerance to $\mathcal{T}_{max} + NR_{err}$. The horizontal lines in figure 2a show the original tolerance, and the inclined lines represent the modified tolerance. However, in the case of a modified tolerance, depending on the input distribution, the faults with either positive or negative effects (negative effect in the case shown) will not be detected whenever the race against numerical inaccuracies is lost. In order for a fault to be invariably detected in the case of a non-zero R_{err}, the following condition needs to be satisfied:

$$F_{eff}P_{act} > R_{err} \tag{8}$$

This equation mostly affects faults in the least significant bits. The faults in the least significant bits can be assumed to have $\frac{1}{4}$ activation probability on the average. Assuming that the effect of numerical inaccuracies is at the order of the least significant bit's magnitude level, the faults at the filter affect the output by $2^i R_{err}$, depending on their bit position, i. These assumptions lead to the conclusion that unless the round-off error averages to zero, a coverage loss at the least significant two to three bits of the filter is to be expected.

Figure 2a shows the average behavior of an invariant checking hardware when truncation is employed in the filter implementation. Truncation errors accumulate over time to produce an undesired bias. There exist various proposed approaches for reducing bias through simple rounding schemes [4]. However, while the approaches introduced in [4] reduce bias appreciably, this will only delay onset but will not obviate the occurrence of a false alarm. A round-to-nearest-odd (or equivalently to nearest-even) scheme [9], on the other hand, introduces zero-bias, as the results of our implementation attest in figure 2b. Such a round-off

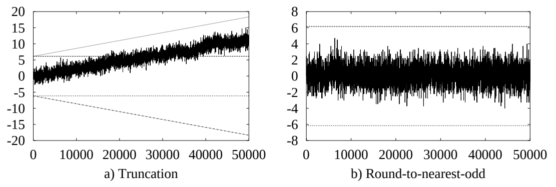

Figure 2: Round-off behavior

scheme is necessary in tolerance-based concurrent test approaches for digital filters as zero-bias will allow detection of all faults by eliminating the need for bias compensation.

5. MONOTONIC FAULT EFFECTS

In the absence of numerical inaccuracies, equation 7 suggests that as long as the fault effects are accumulated monotonically[1], the faults, regardless of their magnitude, will be detected. The magnitude of each individual fault effect impacts solely the latency of detection in that case.

Digital filters are implemented by adders, constant multipliers and registers. Multipliers are constructed from a set of shift-add operations. The linearity of digital filters ensures that fault effects within the adders and registers are propagated to the outputs monotonically. Therefore, guaranteeing monotonicity of the faults within the adders and registers guarantees overall monotonicity. The monotonicity of faults at the registers can be shown easily. We provide an analysis of the adder faults only, thus completing the monotonicity analysis for the overall filter.

We illustrate the analysis by applying it to ripple carry adders; analogous treatment can be applied [1] in the case of high speed adders, such as carry look-ahead adders. We analyze ripple carry adders at the full adder decomposition level. It can be seen by examining the functionality of the adder that faults at the inputs (x, y, and c_i) and at the outputs (c_o and sum) behave monotonically. Faults internal to full adder cells, on the other hand, may behave non-monotonically depending on the implementation style. Figure 3a provides a visualization of the error accumulation produced by the complete set of all possible stuck-at faults of a standard full-adder implementation with XOR gates. In this figure, the x-axis shows the number of patterns applied and the y-axis denotes accumulated fault effects. 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 on-line error detection. A full-adder implementation in which XOR gates are replaced with $AND/OR/NOT$ gate equivalents, on the other hand, produces an improved monotonic faulty behavior. Such an implementation has only 3 non-monotonic faults.

Even though the number of non-monotonic faults is reduced in the latter implementation, there still exist faults with such behavior. Further analysis specific to particular adder types can be utilized to provide forms with embedded monotonicity as in [1]. In more general terms, we show in this paper that implementations with only monotonic fault effects can always be guaranteed, by examining 2-level implementations. As full adders are functionally monotonic, the following proof implies the existence of monotonic implementations of full adders.

[1]A *monotonic fault* always increases or always decreases the output of the circuit whenever it is activated.

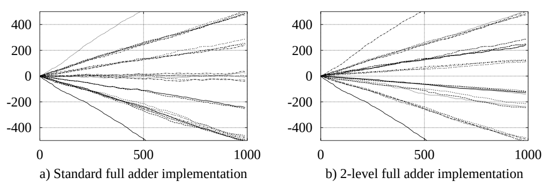

Figure 3: Accumulated fault effects of a full-adder

Theorem: All possible faults in any 2-level implementation of a monotonic function are monotonic.

Proof: In a 2-level AND/OR implementation, there exist only three categories of faults: faults at the inverters, faults at the inputs and output 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 and output of the AND gates:** Stuck-at-0 faults essentially eliminate the minterm implemented by that AND gate and thus always reduce the output from 1 to 0. Stuck-at-1 faults at the inputs expand a minterm by eliminating the literal; a stuck-at-1 fault at the output forces the function to a tautology. In this case the output is strictly increased.
2. **Faults at the inputs and output of the OR gate:** Stuck-at-1 faults force the function to a tautology, thus strictly increasing the output. Stuck-at-0 faults at the inputs have the same effect as stuck-at-0 faults at the outputs of AND gates. The stuck-at-0 fault at the output of the OR gate forces the function always to 0, thus strictly reducing the output.
3. **Inverter faults:** The effect of these faults is the same as having stuck-at-0/1 faults at the inputs of multiple AND gates at the same time, and therefore still monotonic.

While monotonicity of the faults in each of the above three categories is thus shown, faults affecting multiple stems of the same line with opposing polarities can possibly still violate monotonicity. Yet as multiple stem faults in 2-level logic can strictly occur at the inputs of AND gates and consequently only at the inputs of the adder, the functionality of the adder guarantees that the impact of this type of faults on the output is monotonic. The same analysis can be applied to OR/AND implementations *mutatis mutandis.* ■

Figure 3b shows the outcome of accumulated errors produced by a 2-level full adder implementation. As it can easily be seen, all faults produce a non-zero mean average. The difference in 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 $\frac{1}{8}$ to $\frac{1}{2}$.

6. RESULTS

Two 13-tap FIR filters, one low pass and one high pass, including the on-line checking hardware, have been implemented using ripple carry adders, composed of 2-level full-adder cells. A fractional number system of the form 1.9 (1 for the sign and 9 bits for the fraction) was selected for the input of low pass filter. For the output of the filter, a 1.13 form was utilized. In the case of the high pass filter, the input and the output were selected to be of the form 1.7

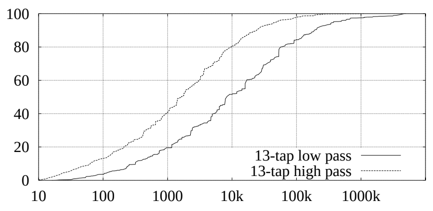

Figure 4: Fault simulation results

and 1.11, respectively. A round-to-nearest-odd scheme was employed in order to eliminate undesired bias produced by numerical inaccuracies in both cases.

The hardware cost of on-line checking hardware is independent of the size of the filter and depends strictly on the hardware requirements for the multiplier used to multiply the input with the invariant. In the case of a low pass filter, the invariant is around 1 whereas the invariant is almost 0 in the case of a high pass filter. Though the hardware cost may vary depending on the actual value of the invariant, it is expected to be less for a high pass filter. The following table presents the implementation details for both filters. Both filters have been synthesized from a data flow graph description and optimized at the gate level by a set of tools developed in C during this work. Fault simulations have been performed by HOPE [6]. The flip-flops, denoted as FF in the table, are assumed to be equivalent to 4 gates in calculating overhead percentiles.

	Filter		Test Overhead			
	Gates	FFs	Gates	FFs	%	$\mathcal{T}_{max}$
Low Pass	4394	169	1096	24	23.5	6.15
High Pass	3014	139	411	19	13.6	0.56

As can be seen in figure 4, 100% fault coverage is invariably achieved, with an area cost that ranges for small filters between an eighth to a quarter of the basic design. As the area cost of the proposed concurrent test is a constant function of the size of the design, typical filter sizes of 64 taps and 25000 gate equivalents can be expected to show area costs of less than 5%.

Fault simulation results confirm that exactly 100% fault coverage for both designs is obtained with no false alarm. The results indicate that even the faults inside the rounding logic are detected by this monotonic fault effect accumulation scheme.

The minimum fault effect for the low pass filter, excluding the rounding hardware, is at the least significant bit, 2^{-13}. The lowest fault activation probability for the least significant bit faults of a ripple carry adder implementation is 1/8 for a uniform input distribution. Therefore, in order to detect a fault at the least significant bits, the number of patterns has to exceed $\frac{\mathcal{T}_{max}}{F_{eff}P_{act}} \approx 400,000$. The simulation results in figure 4 show that at these levels of pattern count, the fault detection level already exceeds 95% for the low pass filter. Analysis of the remaining faults indicates that they are all in the rounding logic. This is due to the fact that the effect on the output of the bits rounded is less than the magnitude of the least significant bit of the output.

The results for the high pass filter can be interpreted similarly. The reduced bit-width of the filter increases the relative effect of faults at the least significant bits and thus reduces the detection latency by one order of magnitude.

7. CONCLUSION

The increasing prevalence of dsp-based electronic solutions expands the necessity for fully effective concurrent test, while at the same time forcing a search for cost-effective solutions applicable to consumer applications. The twin simultaneous requirements of complete fault coverage and cost effectiveness pose stringent challenges. Through detailed experimental and theoretical treatment, we identify possible design areas of concern and outline appropriate solutions for digital filters by utilizing a time-extended design invariant with its associated tolerance. We thus experimentally substantiate the efficacy of the proposed design methods on both low-pass and high-pass filters. 100% fault coverage is invariably achieved, with no false alarms as theoretically shown. Complete concurrent fault coverage coupled with low area overhead and associated guarantees of latency orders of magnitude below human response times make the proposed methods attractive for highly reliable electronic dsp-based applications.

REFERENCES

[1] I. Bayraktaroglu and A. Orailoglu. Design methodologies for concurrent test of digital filters. Technical Report CS99-636, University of California, San Diego, Dept. of CSE, November 1999.

[2] A. Chatterjee and R. K. Roy. Concurrent error detection in nonlinear digital circuits with applications to adaptive filters. In *IEEE International Conference on Computer Design*, pages 606–609, 1993.

[3] M. H. Etzel and W. K. Jenkins. Redundant residue number systems for error detection and correction in digital filters. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 28(5):538–545, October 1980.

[4] P. D. Fiore. Lazy rounding. In *IEEE Workshop on Signal Processing Systems*, pages 449–458, 1998.

[5] S. N. Hamilton and A. Orailoglu. Effective self-recovering ASIC design. *IEEE Design & Test of Computers*, 15(4):25–35, October 1998.

[6] H. K. Lee and D. S. Ha. HOPE: an efficient parallel fault simulator. In *IEEE Design Automation Conference*, pages 336–340, 1992.

[7] A. L. Narasimha Reddy and P. Banerjee. Algorithm-based fault detection for signal processing applications. *IEEE Trans. on Computers*, 39(10):1304–1308, October 1990.

[8] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing.* Prentice Hall, 1989.

[9] M. R. Santoro, G. Bewick, and M. A. Horowitz. Rounding algorithms for IEEE multipliers. In *IEEE Symposium on Computer Arithmetic*, pages 176–183, 1989.

[10] I. Voyiatzis, A. Paschalis, D. Nikolos, and C. Halatsis. R_CBIST: An effective RAM-based input vector monitoring concurrent BIST technique. In *International Test Conference*, pages 918–925, October 1998.

[11] J. F. Wakerly. *Error Detecting Codes, Self-Checking Circuits and Applications.* North-Holland, 1978.