Efficient Testing of Clock Regenerator Circuits in Scan Designs

Rajesh Raina *raina@ibmoto.com*
Robert Bailey *robertb@ibmoto.com*
Charles Njinda *charles.njinda@amd.com*
Robert Molyneaux *bobmoly@ibmoto.com*
Charlie Beh *cbeh@ibmoto.com*

Motorola Inc.
Somerset Design Center
6300, Bridgepoint Pkwy #4
Austin, TX 78730

Advanced Micro Devices
M/S:45, One AMD Place
P.O. Box 3453
Sunnyvale, CA 94088

IBM Corporation
Somerset Design Center
6300, Bridgepoint Pkwy #4
Austin, TX 78730

Abstract

This paper describes the use of a high-level view (functional view) of a clock regenerator circuit for generating effective and inexpensive manufacturing tests. It is shown that the tests generated from the traditional, structural view add hardware overhead, increase design time and potentially lower effective yield when compared to the tests generated from the functional view. A test generation procedure is described and successfully used on a microprocessor design.

Index terms: Microprocessor Testing, Test Pattern Generation, Fault Simulation, Clocks, Clock Regenerators.

1. Introduction

Experience has shown that no single method can be used exclusively, for testing Microprocessors and similarly sized VLSI circuits [1]. Instead, a combination of novel and proven methods are employed to achieve testability goals in a cost-effective manner [2].

In this paper, we focus on the testability problem related to the clock regenerator circuits used in PowerPC^{TM} microprocessors. It is shown that the traditional, structural view of a clock regenerator provides a test solution that adds hardware overhead, increases design time and poses the potential for yield loss. We present a functional view and discuss novel procedures for testing faults in clock regenerator circuits, without incurring hardware overhead or increased design time. One such procedure, based on the functional view, is developed and used on a PowerPC microprocessor for generating manufacturing tests. This paper concludes with a discussion on areas of further study.

2. What is a Clock Regenerator ?

Clock distribution in circuits with millions of transistors is a challenging problem. The clock, typically supplied by one or more sources, has to reach every clocked element in the circuit with a known skew (perhaps identical). One popular method for effectively distributing the clock uses a carefully designed, skew-balanced distribution mesh called the H-tree. The clock is distributed over the H-tree and, at the node ends of the H-tree, the clock is regenerated to the desired drive strength with the use of skew-balanced buffer circuits called the clock regenerators [3,4].

Over time, clock regenerators have become much more than skew-balanced buffers. A typical clock regenerator is shown in Figure 1. Using H-tree distributed clock (gclk) as input, it generates the functional non-overlapping clocks c1 and c2, used in latch-based designs [5]. In addition, the clock regenerators are also used to selectively start and stop the clocks (using hold_ & hold_en inputs) for low-power applications. Furthermore, for testing, the regenerator is used for switching between scan clocks (aclk, c2) and functional clocks (c1, c2) using c1_test_ & scan_c1 inputs. Finally, the debug groups use the clock regenerators, via *waitr* input, to increase clock non-overlap in order to understand timing problems (such as *fastpaths*) on actual silicon.

3. Structural View

The focus of this paper is on the *waitr* (wait clock rise) signal. This signal is used to delay the rising edge of the c1 and c2 clocks by a fixed amount (300 ps) to aid in the debug of fastpath circuit problems. When *waitr*=1, gclk must go through one extra gate in order to generate the 0 --> 1 transition on c1 and c2. This may be readily verified by analyzing the clock regenerator in Figure 1. Figure 2 shows one portion of the clock regenerator circuit (in dotted circle) that is in the observe path for the *waitr* signal leading to the output c2. In an industry environment, where standard ATPG tools are used, the structural view will not be able to generate tests for the *waitr* signal because it is redundant. However, the *waitr* signal can be made testable by removing redundancy. Figure 3 shows the necessary modifications for removing redundancy in the circuit.

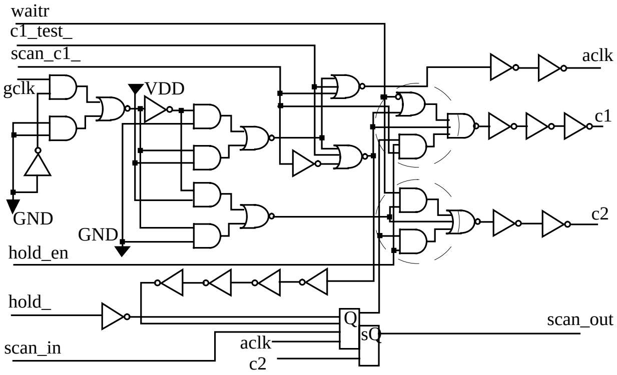


Figure 1: The clock regenerator circuit

DAC 97, Anaheim, California
(c) 1997 ACM 0-89791-920-3/97/06 ..$3.50

Motorola is a trademark of Motorola Inc. IBM, PowerPC, PowerPC 604e and RS/6000 are trademarks of International Business Machines Corporation.

With the modification, the redundancy in the circuit is removed and the clock regenerator circuit becomes fully testable. This may be considered a traditional solution to overcome the testability shortcoming with respect to the *waitr* faults. This solution requires one extra gate per instance of *waitr* signal usage. In addition, the test signal needs to be globally distributed to all the clock regenerators. While the addition of test hardware will not cause the clocks to slow, it will increase the time required to design the clock regenerators.

With the proper costs paid for removing testability shortcomings in the clock regenerator, in terms of extra hardware, extra routing and increased design time, it may be tempting to put this issue to rest. In the following sections, it is shown however, that the testability solution based on the structural view of the circuit is not an optimal one.

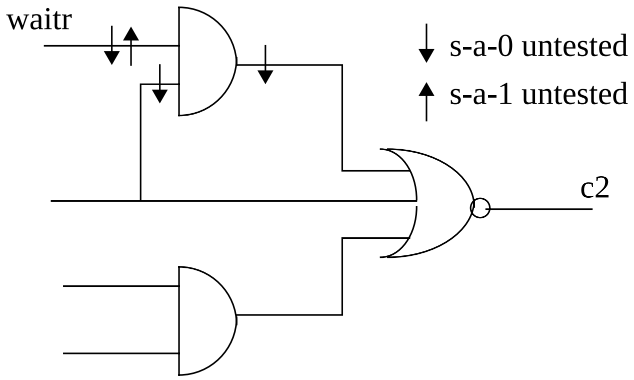


Figure 2: Portion of clock regenerator circuit showing untested faults

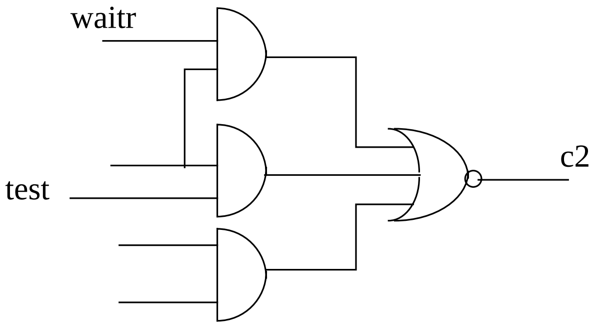


Figure 3: Modifications to the clock regenerator making it fully testable

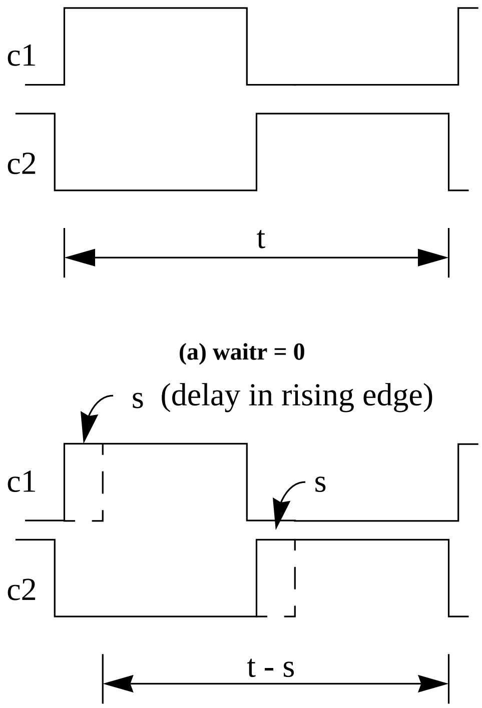
4. Functional view

A functional view helps us understand that stuck-at faults on the *waitr* signal lines do not result in a logical stuck-at fault at the clock regenerator outputs. A **s-a-0 fault** on the *waitr* signal effectively disables the *waitr* capability at the clock regenerator output. A chip user is expected to use the chip with the signal permanently set to a value of 0, since *waitr* is a purely debug feature, not visible to the chip users. Furthermore, it may be argued that a manufacturing test that tests for the s-a-0 fault may be throwing away otherwise functional chips. A **s-a-1 fault** on a *waitr* signal does, however, result in a delay fault at the clock regenerator output.

The effect of *waitr* signal on the c1 and c2 clocks is shown in Figure 4. When *waitr* = 0, the non-overlapping clocks operate normally at a frequency f = 1/t. It may be noted that the time period, in latch based designs, is defined as the time elapsed from the rising edge of c1 clock to the falling edge of c2 clock. When *waitr* = 1, the rising edge of both the c1 and c2 clocks is delayed by a fixed amount of 's' time units.

Let us consider one of many cases where *waitr* signal s-a-1 might have an effect on the functional operation of a circuit. Figure 5 depicts a sequential circuit design based on latches. Let the time taken for a value to propagate from 'd1' to 'd2' be 'd' time units. Therefore, for correct operation, the time period 't' should be greater than or equal to 'd'. When *waitr* is s-a-1, the rising edge of c1 clock is delayed by 's' time units. Thus, the effective time period is shortened by 's' time units. Therefore, in the presence of *waitr* s-a-1 fault the following conditional statements can be made:

If **(t-s) < d <= t** ; then the fault causes circuit failure at 1/t frequency of operation.
If **d <= (t-s)** ; then the fault does not cause circuit failure at 1/t frequency of operation.



(a) waitr = 0

(b) waitr = 1

Figure 4: Two-phase non-overlapping clocking (a) waitr = 0 (b) waitr = 1

Therefore, only the *waitr* s-a-1 faults that cause circuit failure at the specified frequency of operation are of interest from a manufacturing standpoint. We will refer such faults as *critical waitr* faults.

It may be noted that for this circuit example, the delay in the rising edge of c2 clock did not affect circuit functionality at the normal frequency of operation. This is because the critical path started on a c1 clock. In latch based designs, there are cases where critical paths start on c2 clocks - in which case the delay in the rise of c2 clock shortens the effective time period. The analysis is however similar.

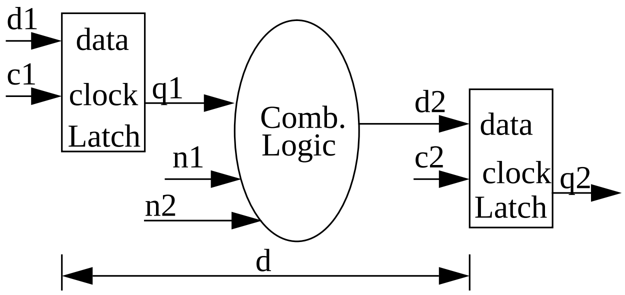


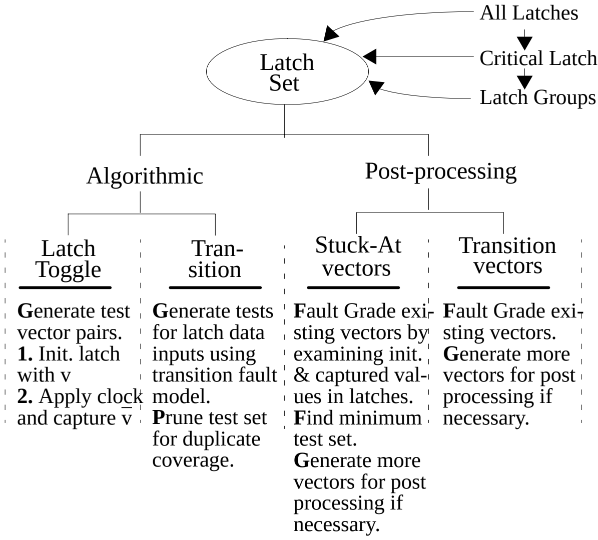
Figure 5: A sequential circuit design based on latches

5. High-level test generation for clock regenerators

From the previous discussion, to test for a critical *waitr* fault, it is necessary and sufficient to test the at-speed latching capability of the corresponding clock regenerator output. In other words, if a latch is initialized with value ‘0’ (or ‘1’) and an application of clock pulse, at the frequency of operation, sets the latch to value ‘1’ (or ‘0’), then this vector pair constitutes a test for a critical *waitr* fault in the clock regenerator that is sourcing the clock pulse.

5.1 Various Test Generation Approaches

There are several approaches towards test generation for critical *waitr* faults. The easiest procedure may result in a large test vector set while the most rigorous procedure may result in an optimal test vector set. The various test generation approaches are outlined in Figure 6.



Generate test vector pairs. **1.** Init. latch with v **2.** Apply clock and capture $\bar{v}$

Generate tests for latch data inputs using transition fault model. **P**rune test set for duplicate coverage.

Fault Grade existing vectors by examining init. & captured values in latches. **F**ind minimum test set. **G**enerate more vectors for post processing if necessary.

Fault Grade existing vectors. **G**enerate more vectors for post processing if necessary.

Figure 6: Various Test Generation Approaches

Selecting a Latch Set:
The choice of a particular latch set, from all the latches in a design, is important in generating a small test vector set. Latches clocked from clock regenerators that do not have critical *waitr* faults need not be considered. This is because, as described in Section 4, such faults do not cause functional failure at the specified frequency of operation. We define a “latch group” as a collection of latches that are clocked by the same clock regenerator. The latches clocked by clock regenerators, that have critical *waitr* faults, can be pruned further such that the resulting latch set contains only one latch from every latch group with critical *waitr* faults.

While these steps can reduce the latch set considerably, timing information is required to select critical latches, and design netlist processing is needed to select a representative latch from each latch group. In the simplest form, the latch set to be considered for test generation could very well be all latches in the design. An overkill with the potential for producing a large test vector set ... but simple.

Having established a working latch set, test generation itself could be done traditionally using an Algorithmic (& Automatic) Test Pattern Generation (ATPG) approach [6] or by Post Processing existing vectors that were generated from stuck-at or transition fault models.

Test Generation - Algorithmic:
To generate specific test for critical *waitr* faults, it is possible to generate vector pairs such that the initial and captured (after at-speed clock application) values, in latches from the latch set, are opposite and the captured value is observed at a Primary Output. We call this “latch toggle” coverage. The drawback is that no commercial ATPG tool specifically performs this procedure, and hence it must be developed in-house and integrated with the design methodology. However, this approach has the greatest potential for generating a compact test vector set.

ATPG based on the transition fault model is provided by several commercial ATPG tools. Tests can be generated for faults on the data inputs of latches in the latch set. However, the commercial tool may attempt to generate tests for both 0-->1 and 1-->0 transitions when only either transition (i.e., latch toggle) is required. The resulting test set can be pruned further to eliminate such duplicate coverage.

Test Generation - via post-processing:
Test generation via post-processing of test vectors simply means that we already have generated a test vector set for stuck-at faults (or transition faults) ... now let us see how many critical *waitr* faults can also be detected with all or part of the existing set. This is useful and even desirable when algorithmic test generation is not available and/or there is no room for extra test vectors in tester memory. The critical *waitr* faults that remain undetected after post-processing may need algorithmically generated test vectors. Alternately, one may generate additional stuck-at (or transition) fault vectors and post-process them as well to obtain additional coverage for critical *waitr* faults.

When post processing existing stuck-at fault test vectors, a small test vector set is desired for testing critical *waitr* faults. In scan based designs (especially LSSD) the entire stuck-at fault test vector set cannot be applied at-speed as a substitute for a small set that targets only the critical *waitr* faults. This is because of two reasons. Firstly, the stuck-at fault test vector set may be testing for delay defects in many non-functional paths (called *false paths*) when administered at-speed. These paths make themselves visible in scan-mode and can never be invoked during functional operation - a problem especially widespread in scan based designs. Secondly, the at-speed application of test vectors for testing critical *waitr* faults requires careful and cumbersome tester control.

However, if transition fault test vectors are available, post-processing is required only to determine *coverage* for critical *waitr* faults; a minimal test set is not required. Of course, it is assumed that the ATPG environment responsible for generating transition fault test vectors has the capability of distinguishing between functional and false paths, and hence generates tests only for the functional paths.

From the various approaches outlined above, we describe a test generation procedure based on post processing of stuck-at fault test vectors - the most commonly available test vectors for industry designs. The procedure also needs to select the smallest set of test vectors that provides maximum coverage because these vectors are required to be applied at functional speed. This procedure strikes a balance between development effort and test set compactness. Although our focus is on scan based designs, the techniques described in this paper can be extended to non-scan designs.

5.2 A Test Generation Procedure

Selecting a Latch Set:

For each clock regenerator, a set of representative latches are selected. The latches selected have the tightest timing margin (obtained from the timing report for the design). Based on this, the following assumptions are made:

Assumption 1: All the representative latches connected to the same clock regenerator (*i.e.*, latches of the same latch group) must experience a similar effect due to s-a-1 fault on the *waitr* signal. This implies if one of the latches fails to capture data all the other latches in the set will also fail to capture data.

Assumption 2: All the representative latches must be in a scan chain.

Assumption 3: The generated test patterns will sensitize the slowest path leading to a representative scan latch.

Based on Assumption 1, it is sufficient to exercise only one of the representative latches, from each latch group, to be able to test for a *waitr* s-a-1 fault. Assumption 2 enables us to initialize and observe the values of the representative latches. Assumption 3 simplifies test generation, but limits this work to a “gross-delay” fault model. The task of identifying the slowest path leading to a representative scan latch requires timing information as well as analysis capability similar to that utilized in “path-delay” test generation [7,8].

Test generation via post-processing:

Our procedure for generating test vectors for *waitr* s-a-1 faults in clock regenerators is based on post processing stuck-at fault test patterns generated from an ATPG program. Vectors capable of detecting *waitr* s-a-1 faults must satisfy the following: *If a latch is initialized during scan-in with a value v, the clock must capture the value $\bar{v}$ for the fault to be observed*.

Clock Regenerator	Latches
R1	L1, L2
R2	L3, L4
R3	L5, L6
R4	L7, L8

TABLE 1. Clock regenerators Vs Latches

Consider an example circuit with 8 representative scan latches and 4 clock regenerators. Table 1 shows the representative latches that are controlled by each clock regenerator. The initialization and captured value for each scan latch for a possible set of test vectors is illustrated in Table 2. The test vectors that can be used to test the *waitr* s-a-1 fault for each clock regenerator is presented in Table 3. From the data in Table 3, vectors {1,4} or {3,4} are all that is needed for testing all clock regenerators in the circuit.

A typical PowerPC microprocessor chip may use several hundred clock regenerators. For example, the PowerPC 604e^{TM} microprocessor contains 625 clock regenerators and about 15000 scan latches [9]. Over 5000 scan vectors are required to test the device.

Test vector #	Attribute	L1	L2	L3	L4	L5	L6	L7	L8
1	Initialization	0	0	0	0	0	0	0	0
	Capture	1	0	0	1	0	0	0	0
2	Initialization	1	1	1	0	1	1	1	1
	Capture	0	0	1	0	1	0	1	1
3	Initialization	0	1	0	1	0	1	0	1
	Capture	0	0	1	0	0	1	0	1
4	Initialization	1	0	1	0	1	0	1	0
	Capture	1	0	1	0	0	1	0	0
5	Initialization	1	1	1	1	1	1	1	1
	Capture	1	1	1	1	1	1	1	1

TABLE 2. Test Vectors

.

Clock Regenerator	Test Vectors
R1	1, 2, 3
R2	1, 3
R3	2, 4
R4	4

TABLE 3. Test Vectors for waitr s-a-1 faults for each clock regenerator

Procedure for post-processing scan vectors to find a minimal test vector set:

1. Partition the scan latches into a set of latch groups, S; where S = {G_1, G_2, G_3,..., G_n}; n is the number of clock regens. Each group G_i contains those scan latches whose capture clocks are derived from the same clock regenerator.
2. Let T be the set of test vectors. For each $t_i \in$ T, determine all $G_j \in$ S that are covered by t_i. A group G_j is covered by a test vector t_i, if at least for one of the representative latches, $l_k \in G_j$, the initialization value is different from the captured value.
3. Find the minimum cover.

For the example circuit in Table 1, S = {G_1, G_2, G_3, G_4}. G_1 = {L_1, L_2}, G_2 = {L_3, L_4}, G_3 = {L_5, L_6} and G_4 = {L_7, L_8}. Vector 1 covers {G_1, G_2}, Vector 2 covers {G_1, G_3}, Vector 3 covers {G_1, G_2} and vector 4 covers {G_3, G_4}. Thus the minimum cover is vector set {1, 4} or {3, 4}.

The minimum covering problem is NP-complete. Our goal is to generate a small set of test vectors not necessarily minimum. The following heuristic can be used to obtain a quasi-minimal cover.

1. Find the set, X of essential vectors. *An essential vector is one which must be selected to cover a group*. That is, no other vector can be used to cover that group. Mark these vectors as selected and the groups as covered.
2. Determine all other groups covered by the essential vectors. Mark these additional groups as covered.
3. For the rest of the vectors P, determine the number of unmarked groups than can be covered by each vector. Let this number for each vector t_i, be m_i.
4. Select the vector with the largest m. Mark this vector as selected and mark all groups that are covered by this vector. Add this vector to the set X.
5. Repeat steps 3 and 4 until all groups are marked or no new groups can be covered by the test vectors left.
6. If all groups are covered, then X is the minimal set of test vectors required for at-speed testing of all clock regenerators. If however, all groups are not covered then additional test vectors need to be generated.

6. Results

Given the industry requirements, our approach was to use the simplest test generation procedure that could produce a test vector set of acceptable size. As a first step, postprocessing was done on a set of 500 scan test vectors that were generated by an ATPG tool (Fastscan [10]) for stuck-at fault testing of PowerPC 604e microprocessor chips. Our latch set comprised of all latches in the design. We could have selected only the critical latches (using timing information) and the representative latches from each group (by performing netlist analysis) - but these steps require extra work that was not budgeted for this study. Table 4 shows the cumulative fault coverage Vs the number of test vectors processed.

Vectors	Fault Coverage
1	32.91 %
2	49.04 %
10	72.71 %
50	81.45 %
100	90.95 %
250	95.81 %
500	98.49 %

Table 4: Fault Coverage Vs At-speed vectors

As a next step, this set of 500 vectors was processed further to order the vectors by their effectiveness in detecting *waitr* faults. The results from this step are shown in Table 5 where the 10 most effective vectors and the resulting fault coverage is shown vector-by-vector. The Table lists the vector order in the original set, its stand-alone fault coverage and the cumulative fault coverage. By selecting the most effective vectors, the fault coverage level from 500 vectors is achieved with 33 vectors. The post processing package was developed in Perl/C languages. Post processing was done on an IBM RS/6000^{TM} workstation.

It may be noted that for post processing, we considered all latches as part of the latch set - not just the latches affected by critical *waitr* faults or single representative latch from each latch group. Therefore, the untested faults (1.51%) may not be critical or may already be covered by a tested latch belonging to the same latch group. The fault coverage number thus represents the lower bound for actual coverage.

Ordered Vectors	Vector # (orig. set)	Individual Fault Cov.	Cumulative Fault Cov.
1	62	49.76 %	49.76 %
2	428	49.55 %	74.50 %
3	124	49.35 %	86.51 %
4	462	49.22 %	92.43 %
5	395	49.19 %	95.49 %
6	280	33.44 %	96.78 %
7	448	34.16 %	97.38 %
8	135	34.20 %	97.71 %
9	268	21.24 %	97.91 %
10	263	34.50 %	98.04 %
33	-	-	98.49 %

TABLE 5. Selecting the most effective vectors

Upon inspection of the untested latch toggle faults, it was found that 0.7% of the total were trap latches. On these scan latches, the data output is fed back to the data input and are used primarily for scan based silicon debug. Therefore, by design, these latches do not affect the critical *waitr* faults. This increases the lower bound for the fault coverage to 99.19%.

7. Discussion

The results are very encouraging. From the first post processing step, 100 vectors provide 90% coverage for the *waitr* signal faults. With 250 vectors, the coverage is over 95% and with 500 vectors, it is over 98%. The second post processing step, where the 500 vectors are ordered by effectiveness, we are able to achieve the same level of fault coverage (98.49%) with only 33 vectors. In fact 98% fault coverage is achieved with only 10 vectors.

Many digital circuits, most notably microprocessors, are also sorted by frequency of operation during testing. The chips that can function at a higher frequency of operation command a premium in price, while chips that can only function at a lower frequency are offered at a discounted price. In a manufacturing environment, tests based on a structural solution (Section 3) would have discarded chips with *waitr* s-a-0/s-a-1 which otherwise were functional. In addition, it would have discarded chips with *waitr* s-a-1 that could have operated at lower frequencies. Finally, the added test hardware and the test signal for the structural view solution would have added extra fault sites. From this discussion, it stands to reason that a structural approach would have resulted in an unnecessary yield loss.

It may be noted that even though we have described a fairly rigorous procedure which offers the potential for generating a compact set of test vectors at a modest development cost (Section 5); we ended up using the simpler procedure (with very little development cost) because it provided a test vector set of acceptable size (Section 6). The simpler procedure, with negligible development costs, provides acceptable test coverage with 33 test vectors. The more rigorous procedure could have provided acceptable test coverage with far fewer test vectors. However, it turns out that 32, 50 or even upto 100 test vectors is not a capacity issue for microprocessor class designs - just yet! The lack of a practical issue provides little motivation to allocate resources for pursuing the problem of generating the most compact set of test vectors for critical *waitr* faults. The tables may turn rather quickly if such a class of faults increases and simpler procedures generate hundreds or even thousands of test vectors - an interesting industry rule nonetheless!

8. Conclusion

This paper describes a working procedure, and successfully integrates components of the procedure in the design flow, to generate efficient tests for faults in clock regenerator circuits of PowerPC 604e microprocessors.

With a simple real-design example, this paper has also attempted to illustrate the usefulness of functional view over the traditional structural view of a circuit in generating an effective and inexpensive test. In this example, a structural view would generate a test that would cost in terms of design hardware and design time; and also posed the potential for yield loss. A functional view, instead, helped generate a test without incurring hardware overhead, design time or yield loss.

Further study is needed to generalize the test characteristics of the example clock regenerator circuit that will help test engineers choose the right level of abstraction for specific circuits in order to develop effective and inexpensive tests.

Acknowledgments:

The authors would like to thank the paper reviewers for reviewing this work and offering corrections and useful suggestions. We also thank our colleagues Nandu Tendolkar, Phil Shephard, Magdy Abadir, Kofi Vida-Torku, Carol Pyron, George Joos and Ashu Razdan for providing valuable critique. Finally, our thanks goes to Carlos Gutierrez and Mark McDermott for providing management support in completing this work.

References:

1. Hunter C., E. K. Vida-Torku, J. LeBlanc, “Balancing Structured and Ad-Hoc Design for Test of the PowerPC 603 Microprocessor,” *Proc. of IEEE Intl. Test Conf.*, pp. 76-83, 1994.
2. Maxwell P., and R. Aitken, “Iddq Testing as a Component of a Test Suite: The need for several Fault Coverage Metrics,” J. *Electronic Testing: Theory and Applications*, pp. 305-316, 1992.
3. Ganguly S., & S. Hojat, “Clock Distribution Design & Verification for PowerPC Microprocessors,” *Proc. of Design Automation Conf.*, pp. 58-61, 1995.
4. Pullela S., N. Menezes, and L.T. Pileggi, “Skew and Delay Optimization for Reliable Buffered Clock Trees,” P*roc. IEEE Custom Integrated Circuits Conf.*, pp. 556-562, 1993.
5. M. Abramovici, M.A. Breuer and A.D. Friedman “Digital Systems Testing and Testable Design”, Computer Science Press, 1990.
6. Raina R., & T.N. Rajashekhara, “Automatic Test Pattern Generation - A general Approach,” *Proc. 1987 IEEE Southern Tier Technical Conf.*, SUNY Binghampton, pp. 132-138, April, 1987.
7. Smith G.L., “Model for Delay Faults Based Upon Paths,” *Proc. of IEEE Intl. Test Conf.*, pp. 342-349, 1985.
8. Pomeranz I., & S.M. Reddy, “On the Number of Tests to Detect All Path Delay Faults in Combination Logic Circuits,” I*EEE Trans. on Computers*, pp. 50-62, No. 45 (1), 1996.
9. Denman M., P. Anderson, and M. Snyder, “Design of the PowerPC 604e^{TM} Microprocessor,” *Proc. of IEEE Compcon*, pp. 126-131, 1996.
10. Raina R. , C. Njinda, R. Bailey, B. Long, C. Beh & R.F. Molyneaux, “Single CRAM Solution in Fastscan for a RAM with Different READ & WRITE Data Widths,” *Proc. of the 13th Mentor Graphics Users Group Intl. Conf.*; Oct. 21-24, 1996.