Delay Defect Screening for a 2.16GHz SPARC64 Microprocessor

Noriyuki Ito, Akira Kanuma, Daisuke Maruyama, Hitoshi Yamanaka, Tsuyoshi Mochizuki, Osamu Sugawara, Chihiro Endoh, Masahiro Yanagida, Takeshi Kono, Yutaka Isoda, Kazunobu Adachi, Takahisa Hiraide[1], Shigeru Nagasawa, Yaroku Sugiyama, Eizo Ninoi

Fujitsu Limited, [1]Fujitsu laboratory
4-1-1 Kamikodanaka, Nakahara-ku,
Kawasaki, 211-8588, Japan

ito.noriyuki@jp.fujitsu.com

ABSTRACT

This paper presents a case-study of delay defect screening applied to Fujitsu 2.16GHz SPARC64 microprocessor. A non-robust delay test is used while each test vector is compacted to detect multiple transition faults in a standard scan-based design targeting a stuck-at fault test. Our test technique applied to a microprocessor designed with 6M gate logic, 4MB level 2 cache, and 239K latches, achieves 90% coverage using 3,103 test vectors. We estimate the distribution of the delay of paths covered by our delay test. We also show the effectiveness of our method by discussing the correlation between the screening result and the actual number of delay defects.

Keywords: Microprocessor, screening, delay defect, at-speed

1. Microprocessor Overview

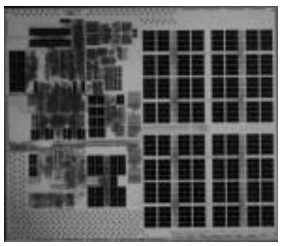

Figure 1. Fujitsu SPARC64 Microprocessor

As the successor to the 1.3GHz SPARC64 microprocessor [1,2] designed in 130 nm technology, we have developed a 2.16GHz SPARC64 microprocessor [4] in 90 nm technology. The chip die image is shown in Figure 1. The right-hand area is the level 2 cache, and the left-hand area is the logic. We used a hierarchical incremental custom design methodology supported mainly by in-house CAD tools [3] which have been customized for high performance microprocessor design. Our SPARC64 microprocessors have the same reliability, availability, and serviceability (RAS) functions as the mainframe; they are employed in the UNIX servers used for a wide range of applications including the mission critical ones. The specification of our 2.16GHz SPARC64 microprocessor is as follows:

Process: 90nm, Cu metallization, 10 metal layers
Frequency: 2.16GHz
Die size: 18.46mm x 15.94mm
Transistor count: 400M
Level 2 on-chip cache: 4MB
I/O signals count: 279
Power dissipation: less than 65W

For testability purposes, scan chains are designed to scan data latches. Scan chains are selected and controlled by a test port controlling macro (TPCM). For the memory, a high speed test based on BIST is used in addition to the test through scan chains. For the delay test performed by two consecutive at-speed clock pulses, a 2-pulse generator is built in the chip. These design-for-testability circuits are verified based on logic simulation by a CAD tool to check testability design rules. The features of these testability circuits are as follows:

Scan clock: 2-phased clocks
Scan chain count: 16
Scan latch count: 238,620
Additional circuits: TPCM, 2-pulse generator, RBIST

The rest of this paper is organized as follows. In Section 2, the test flow from chip manufacturing to shipment of server products in which the microprocessor is used is overviewed. Section 3 presents the general concept of a delay test and our algorithm of test vector generation applied to a 2.16GHz SPARC64 microprocessor design. In Section 4, we show vector generation results, and compare the results with the results of using a robust test method. Finally, we conclude the paper in Section 5.

2. Delay Defect and Chip Screening

Delay defect is a defect that does not influence the functionality of a circuit but changes the speed of the circuit. The cause of this type of defect is mainly an excessive delay due to a high resistance by an open wire or interconnect via, or due to a large capacitance caused by bridging short circuits between wires [5]. Since this kind of delay defect is fatal in high performance microprocessors, it has to be detected as early as possible after manufacturing. Figure 2 shows our test flow from chip manufacturing to shipment of server products. At the end of wafer fabrication, a go/no-go test is performed for each chip using a set of functional test vectors. For good chips, a delay defect screening is performed. Chips that pass all these tests are shipped to a server product factory. In the server product factory, the chips are packaged after an acceptance test such as external inspection is performed. Then, a burn-in test is applied to induce time and stress dependent failures by applying thermal and electrical stresses. Next in a *speed binning* test, which determines the maximum functional operating speed of each chip,

0-7803-9451-8/06/$20.00

a test program is loaded into the memory of the chip. Then, speed binning at several frequencies is performed by executing the test program on the chip. After the maximum speed is determined for each chip, they are used inside units. Then, a unit test is performed under a high temperature and supply voltage. After the unit test, each system is configured according to a customer's requirement and the server is shipped to the customer after running a test for several days. In this paper, we refer to the test performed after the burn-in test as the system test.

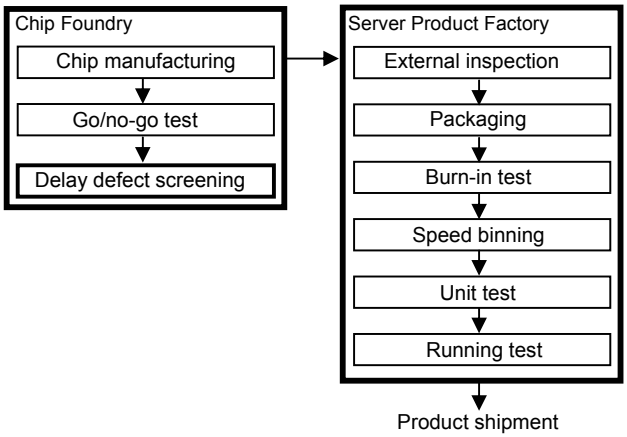

Figure 2. Test Flow of Server Products

In our test flow, we can reduce delay defects found after packaging by screening which is done through performing a delay test at wafer-level. Since the packaging cost is wasted if a defect is found after packaging, this screening is very important to reduce the manufacturing cost. In the delay test at the wafer-level, a set of two consecutive at-speed pulses is applied for each pair of test vectors at the same or a slower speed than the production specified speed. The important thing before adding a delay test step to the test flow is to confirm that a go/no-go result by a delay test correlates well with the delay defect found in a system test. To the best of our knowledge, few analyses on the correlation between delay test results and actual delay defects have been reported [6]. In [6,7] to confirm the correlation, chips failing the delay test are tested again at a lower speed. In our experiment, we confirmed it using the data of chips that failed the speed binning. If there is a good correlation and there is a possibility of reducing the manufacturing cost, the screening by the delay test is applied. Table 1 shows four possible cases depending on the results of the delay test at chip-level and the system test performed after packaging the chip. Case A is when a chip passes both delay and system tests. Case B is when a chip is *under-killed* by the delay test. The chip passes the delay test but fails a system test. Since the delay defect is found in a system test after packaging, the packaging cost is wasted. Case C is when a chip is *over-killed* by the delay test. The chip fails the delay test but passes the system test. In this case, screening regards a good chip that passes a system test as a defective chip. This means that good chips are wasted. Case D is when a chip fails both delay and system tests. By screening out chips with delay defects, there is no waste of manufacturing costs for packaging and the system test. When the delay test is used for screening, ideally cases B and C should not happen. The reason some chips are under-killed (Case B) can be the low coverage achieved for critical paths and the low frequency of the clock applied for the delay test. The reason some chips are over-killed (Case C) is the excessive usage of test vectors that test functionally untestable paths [6,7]. A functionally untestable path is a path not activated by any combination of instructions in a microprocessor.

Table 1. Correlation between Delay Test and System Test

		Delay test	
		Pass	Fail
System test after packaging	Pass	Case A: Real pass No loss	Case C: Over-kill Loss
	Fail	Case B: Under-kill Loss	Case D: Real fail No loss

When the delay test is applied, the clock is set to the same frequency or a slower frequency than the target frequency of the chip. According to the frequency of the clock in the delay test, the percentages of chips categorized into cases A, B, C, and D change. In general, when the frequency increases, the percentages of cases A and B decrease and the percentages of cases C and D increase. Here, let N_A, N_B, N_C, and N_D be the numbers of chips categorized into cases A, B, C, and D, respectively. Further, let UP, PC, STC , and DTC be the unit price, the packaging cost, the system test cost, and the delay test cost respectively. When comparing the case the delay test is applied with the case the delay test is not applied, the loss of the manufacturing cost ΔLMC is expressed as,

$$\Delta LMC = N_D \cdot (PC + STC) - N_C \cdot UP - \sum_{i=A,B,C,D} N_i \cdot DTC \qquad (1)$$

Applying the delay test is beneficial only if ΔLMC is positive. The value of ΔLMC depends on the ability of the delay test to detect actual delay defects and the frequency of the clock used in the test. If the ability of the delay test is fixed, it is possible to use a frequency which maximizes ΔLMC .

3. Delay Test

In a standard scan design, all latches are connected into a single or multiple scan chains through which the values of latches can be loaded and unloaded. In the delay test for a standard scan design, the test vector is loaded and the result is unloaded through scan chains. To generate test vectors for the delay test, *transition fault model* [8] or *path delay fault model* [9] is used. In the delay test based on the path delay model, paths to be tested are selected from critical paths. A path which causes an incorrect operation of a chip due to a small excessive delay can be tested with a high probability. However, the test coverage for all delay faults in the entire chip is typically low because only a small number of paths are tested; the reason is the limitations on test generation and application times. On the other hand, the coverage for all delay faults in the entire chip is higher if a delay test based on the transition fault model is used. Unfortunately the delay test may not test critical paths with a

high probability. Since the coverage for all delay faults in the entire chip is very important if the delay test is used to screen out actual delay defects, we select the transition fault model instead of the path delay fault model to achieve a high coverage. In the transition fault model, *slow-to-rise* and *slow-to-fall* delay faults are assumed at all input and output pins of each gate in a circuit. A slow-to-rise fault is a fault that makes a rising transition slow, while a slow-to-fall fault is a fault that makes a falling transition slow. In this fault model, it is assumed the delay increase due to a defect is large enough to make a chip operate incorrectly at the desired clock speed. The delay test for detecting a transition fault is done using a pair of test vectors $\langle V_1, V_2 \rangle$. The delay test is performed using this vector pair and two consecutive at-speed clock pulses. The first test vector V_1 is loaded into launching latches through scan chains. Then, the first clock pulse is issued so that the launching latches capture outputs of the combinational circuit. The outputs of launching latches become inputs to the combinational circuit to generate a transition. The generated transition is rising for a slow-to-rise fault and falling for a slow-to-fall fault. The second at-speed clock pulse is issued to capture the generated transition at capturing latches. When a value captured by a capturing latch is the same as the value after a transition, there is no actual delay defect at the assumed location. When the captured value is the same as the value before a transition, there is an actual delay defect at the assumed location. Figure 3 shows the sequence of the abovementioned delay test. In our delay test, the second test vector is loaded into launching latches through the combinational circuit by issuing the first clock to latches. This technique is called *functional justification*. There are two other techniques to load the second test vector, *enhanced scan* and *skewed load*. [10] compares these three techniques. We use functional justification because it needs no additional hardware. Functional justification is less likely to test functionally untestable paths than other two techniques [10,11]. Thus, the possibility of over-kill in the functional justification is least among the three techniques.

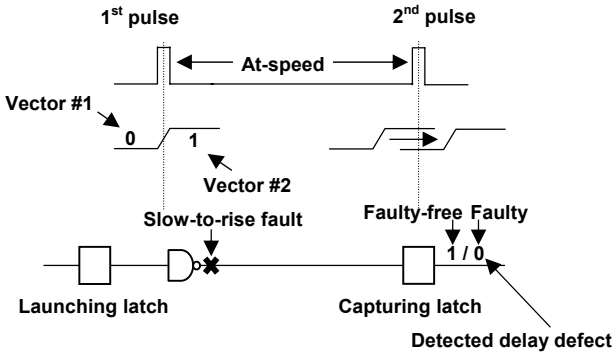

Figure 3. Delay Test by Two At-speed Clock Pulses

3.1 Design for Testability Circuits

Our delay test based on a standard scan design needs no additional hardware except for a circuit to generate two at-speed clock pulses. In this generator, a 2-pulse extractor and a selector are added after a PLL circuit as shown in Figure 4. The 2-pulse extractor is a circuit to extract two consecutive pulses from the PLL output and the selector selects the PLL output or the 2-pulse extractor output according to the test mode signal that controls the selector. The selected clock pulses are supplied to the system clock distribution circuit.

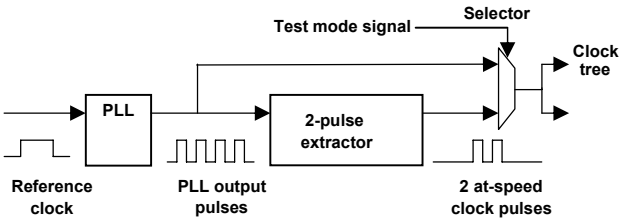

Figure 4. The Circuit Used to Generate Two At-speed Clock Pulses

3.2 Test Vector Generation

To generate test vectors for the delay test, our ATPG tool for stuck-at faults is enhanced in two areas. The first is to handle two consecutive time-frames corresponding to time frames before and after the first at-speed clock pulse. By this enhancement, the ATPG tool can generate a set of two test vectors by processing two time frames in the reverse order (i.e., t and t-1). In the original ATPG, expression (2) is applied only as a test generation condition to detect a stuck-at fault on line p. In the enhanced ATPG, expression (3) is added to generate a transition. Expression (2) corresponds to time frame t and expression (3) corresponds to time frame t-1. Our enhanced ATPG processes expression (2) before expression (3).

$$\text{state}(t,p) = T/F \quad (2)$$

$$\text{state}(t\text{-}1,p) = F/F \quad (3)$$

Here, the term state(t,p) in the left hand side of expression (2) denotes the state of line p at time frame t. The T/F in the right hand side represents value T in the fault-free circuit and value F in the faulty circuit. A slow-to-rise fault takes T=1 and F=0, and a slow-to-fall fault takes T=0 and F=1. The term state(t-1,p) in the left hand side of expression (3) denotes the state of line p at time frame t-1. The notation F/F in the right side represents value F in both the fault-free and faulty circuits. A slow-to-rise fault takes F=0 and a slow-to-fall fault takes F=1. In the second enhancement, a feature to generate multiple transitions is added to our ATPG. This is done by applying a dynamic compaction technique at time frame t-1 to detect as many faults as possible using a single test vector pair. Table 2 shows the comparison of off-path values in robust, non-robust, and our tests. A delay test is called *robust* if it can detect the target delay fault independent of other delay faults. On the other hand, it is called *non-robust* if the delay test for the target delay fault is invalidated by other delay faults. In a robust test, off-path values at time frame t-1 are set to the values they had at time frame t so that other delay faults do not invalidate the delay test. Since invalidation is permitted in a non-robust test, off-path values at time frame t-1 can take any arbitrary value. Our method [12] tries to generate as many transitions as possible by selecting different values for off-path signals at time frames t-1 and t. We denote these off-path values by A0 and A1 (Table 2) to distinguish them from don't care (X) used in the non-robust test. A0 (A1) means the value 0 (1) is selected if possible. When a rising transition on an on-path input of an AND or a NAND gate is propagated, the off-path input can be set to A0 (see Figure5). When

a falling transition on an on-path input of an OR or a NOR gate is propagated, the off-path input can take A1. Since in other two possible cases no transition can propagate on an off-path if a transition propagates on an on-path, off-paths can be set to neither A0 nor A1. For A0 (A1), a test vector compaction algorithm assigns 0 (1) value to off-path inputs. More details on this can be found in [12]. In the non-robust delay test described before, it is not possible to analyze a specific path because other delay faults may invalidate the test for that specific path. On the other hand in our method, delay faults that invalidate a test are also detectable. For example, consider an on-path and an off-path input of an AND gate. A slow-to-rise delay fault on an on-path input may invalidate a slow-to-rise delay test of the other input and vice versa. If only one of the inputs has a slow-to-rise delay fault, then the defect is detectable. Therefore, our method can detect delay faults using a small number of test vectors.

Table 2 Off-path conditions in Delay Test

	On-path		Off-path					
			Our Method		Non-Robust		Robust	
Time frame	t-1	t	**t-1**	**t**	t-1	t	t-1	t
AND/NAND	0	1	**A0**	**1**	X	1	1	1
	1	0	**X**	**1**	X	1	1	1
OR/NOR	0	1	**X**	**0**	X	0	0	0
	1	0	**A1**	**0**	X	0	0	0

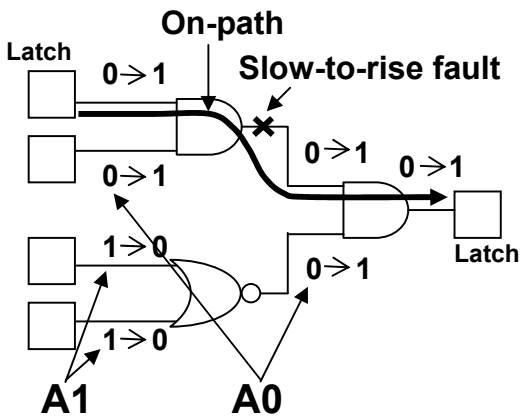

Figure 5. Generation of Multiple Transitions

4. Result

In this section, we summarize statistics such as the number of test vectors for each test item and the test vector generation time. Then, we present the followings: 1) a comparison of the number of test vectors and the test vector generation time for the robust test method and our method, 2) the delay distribution for paths covered by our delay test method, 3) the correlation between screening results and the actual delay defects.

4.1 Test Items and Generated Test Vectors

We have developed an in-house ATPG tool at Fujitsu and have been using it since the ECL mainframe era. The reason of developing an in-house tool is that an ATPG program needs to be optimized for design-for-testability circuits. All test vectors including the ones used for the delay test are generated by our ATPG program. Table 3 shows test vector generation results for a 2.16GHz microprocessor design described in Section 1. SCAN is a test for scan chains, FUNCTION is a functional test based on a stuck-at fault model, RBIST is a Built-In-Self-Test for memories, and DELAY is the delay test to screen out delay defects. We verify all test vectors from functional and timing viewpoints before they are applied to chips. The verification is done by an in-house test vector verification tool named VERIFIER. Using the delay data generated by a static timing analyzer in SDF format, VERIFIER verifies the expected values and timing by performing timing simulation. By this, it is possible to verify not only the expected values for input vectors but also the timing information reported by the static timing analyzer. Table 4 shows the CPU time needed for the verification.

Table 3. Test Items

Test	# Faults	# Vectors	Coverage	Time (Hours)
SCAN	9,059,216	14	99.9%	0.22
FUNCTION	21,803,669	2,014		14.15
RBIST	N/A	N/A	N/A	0.12
DELAY	9,750,387	3,103	90.0%	31.11

Table 4. Time Required for Verification and Test

Test	Verification (Hours)	Relative Verification Time
SCAN	101.02	2.9%
FUNCTION	16.80	0.5%
RBIST	3,346.04	96.1%
DELAY	16.24	0.5%

We generated test vectors using a Fujitsu 1.3GHz PRIMEPOWER, and we verified test vectors on IA servers with an Intel Pentium4 2.4GHz CPU. The verification time in Table 4 corresponds to the case when one CPU is used; we accelerated the verification by parallel execution on 12 CPUs. Overall, 2.9%, 0.5%, 96.1%, and 0.5% of the total time was used for the scan chain test, the function test, the memory test, and the delay test, respectively.

4.2 Comparison with Robust Delay Test

First we used the robust test to screen out delay defects, but it was not practical because of large generation time and large number of test vectors. Therefore, we compare our test with the robust test to show the effectiveness of our method. The left graph in Figure 6 shows the change of coverage with the number of test vectors. In our method, the coverage reaches 90% using only 3,103

test vectors. The robust test requires twice the number of test vectors to achieve the same coverage. The right graph in Figure 6 shows the change of coverage as a function of the execution time. In our method, the coverage reaches 90% in 31 hours, but it takes the robust test three times longer to reach the same coverage. Therefore, to screen out delay defects of microprocessors, using the robust test is not practical. Table 5 summarizes the above results. When performing our delay test, the maximum memory usage was about 5 GB.

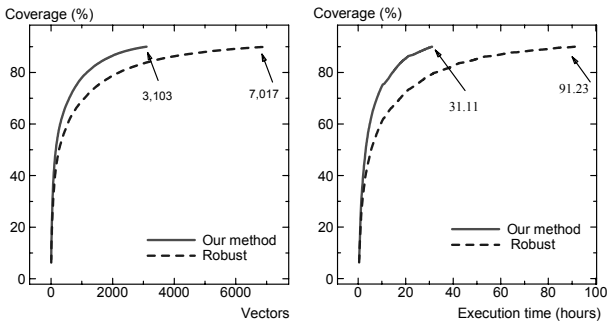

Figure 6. Change of Coverage of Delay Test

Table 5. Summary of delay test results

Test	# Faults	# Vectors	Coverage	Time (Hours)
Robust	9,749,914	7,017	90.0%	91.3
Our Method	9,750,387	3,103	90.0%	31.1

4.3 Delay of Paths Covered by Delay Test

The transition delay fault assumes an infinite excessive delay due to a defect. More than one path may pass location *x* where a delay fault is assumed. In a delay test based on a transition fault, the path with the maximum delay out of all paths passing location *x* is not necessarily tested by a test vector generated for this purpose. We estimate the distribution of the delay of paths covered by our delay test by VERIFIER described in Section 4.1. VERIFIER performs timing simulation using input vectors and delay information. Then simulated output values are compared against the expected values. In our verification tool, we can specify the time to compare the values (i.e., the strobe time). We performed the verification several times using different values for the strobe times; we used frequencies both lower and higher than the target frequency.

If the simulated values of latches are different than the expected values, the applied vector tests the delay of a path that is slower than the frequency corresponding to the strobe time. Therefore, we can get the delay distribution of paths tested by the applied test vectors from the number of latches in which simulated values are different than the expected ones. However, the exact number of paths is not known from the number of those latches. If the simulated value of a latch is different than its expected value, then some of the paths terminating at the latch are tested. For each test vector we count the number of latches whose simulated values are different than the expected values. Then, we calculate the sum of the number of latches for all test vectors. We plot the results in a graph (see Figure 7). The graph *Dist_DT* shows the delay distribution of paths covered by the delay test. For *Dist_DT* the vertical axis corresponds to the numbers of latches. Figure 7 also shows *Dist_STA*, the actual delay distribution of paths reported by the static timing analyzer. The vertical axis in this case corresponds to the number of paths.

Since the scale of the vertical axis is different for *Dist_DT* and *Dist_STA*, we cannot compare the two graphs directly to find out how many paths are covered by the delay test. Therefore, we enlarged *Dist_DT* graph vertically so that two graphs intersect at frequency *A*.

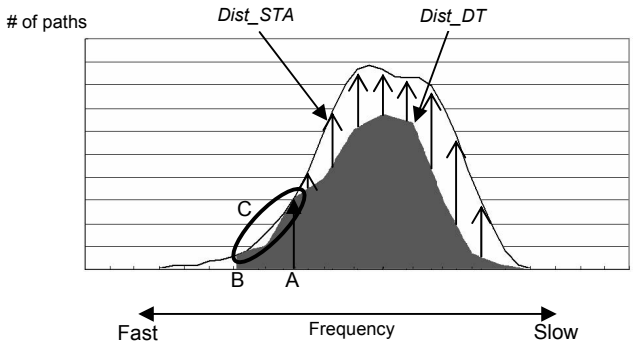

Figure 7. Delay Distribution of the Paths Covered by Delay Test

Since the fastest frequency at which VERIFIER can run is *B*, we collect data for frequencies less than it. Assume most paths with a speed between frequencies *A* and *B* in *Dist_STA* are almost fully covered by the delay test (see the circled area marked with *C*). On the other hand, *Dist_STA* is not fully covered by *Dist_DT* for frequencies faster than *A*. Therefore, we speculate paths with a speed faster than frequency *A* in *Dist_STA* are not fully covered by the delay test. However, we can see that the delay test can to some extent cover paths in each frequency according to the delay distribution *Dist_STA* although the coverage is smaller in low frequencies. Therefore, we can expect that screening by a delay test based on a transition fault model can also cover to some extent slower paths including critical paths. We report experimental results in Section 4.4.

4.4 Screening Results

We perform the screening for delay defects at 1.5GHz which is about 70% of the target frequency of the chip and at the normal operating voltage. The screening ratio for the total actual defective chips is 5.0%. In general, it is effective to detect delay defects at a lower voltage [13]. However, there is a risk to over-kill chips in a delay test at a lower voltage. Therefore, the important thing to do before actually applying a delay test to screening is to ensure that go/no-go results by a delay test correlate well with defective chips found by the system test. If the correlation between the delay test and the system test is confirmed, the screening for delay defects is applied if the value of ΔLMC in expression (1) is positive.

We apply the delay test at a lower voltage for chips that pass all functional tests as well as the delay test at the normal voltage. We set the clock frequency to 1.5GHz. Then, regardless of the result,

speed binning by the system test is performed on the chips. We used about 4,000 chips in our experiment. Table 6 shows the ratios of pass and fail in the system test for both outcomes of the delay test. Since the defects found in the system test are not only delay defects but also memory defects, we exclude chips that have a memory defect. From the table, we can see that 1.7% of the chips that passed the system test failed the delay test at a lower voltage. Further, 46.7% of the chips that failed the system test also failed the delay test at a lower voltage. This means the delay test may waste 1.7% of good chips that pass the system test. Furthermore, 46.7% of packaged chips that fail the system test can be eliminated through screening by a delay test. At the time of this paper submission, we do not have delay test results for frequencies different than 1.5GHz due to the limited tester resource. If the frequency applied in the delay test is changed, we believe it is possible to achieve a detection ratio higher than 46.7%. In any case, the decision whether to apply it to an actual screening must be done based on the value of ΔLMC in expression (1). Otherwise, there is a risk of wasting good chips.

Table 6. Screening Result

		Delay test		Total
		Pass	Fail	
System test after packaging	Pass	98.3%	1.7%	100%
	Fail	53.3%	46.7%	100%

5. Conclusions

We presented a screening technique based on the non-robust delay test model that can detect multiple transition faults simultaneously using a single pair of vectors. We showed the results of applying the delay test to screen 2.16GHz microprocessors. Our delay test based on the standard scan design can generate vectors within practical limitations on the number of test vectors and generation time and it can achieve 90% coverage. By applying this delay test to delay defect screening, we can screen out chips with defects at the normal voltage. The screening ratio at the normal voltage is 5.0% for the total actual defective chips. This screening can reduce the manufacturing cost by detecting delay defects before chips are packaged even if the applied frequency is slower than the target frequency. By applying the delay test at a lower voltage for chips that pass the delay test at a normal voltage, it is shown that 46.7% of chips that fail at the system test may be screened out before packaging.

Since our delay test described in this paper is based on the transition fault model, it does not need any delay information of gates or wires when the test vector is generated. Therefore, the coverage of critical paths of the circuit may not be enough. To increase the coverage, it is necessary to test more critical paths. For this, some test vectors generated based on the path delay model should be added to our test vectors. We are planning to enhance our ATPG tool to generate these test vectors using the delay information of gates and wires.

6. ACKNOWLEDGMENTS

We would like to thank Kaoru Kawamura and other members of CAD group at Fujitsu Labs. Ltd. as well as the members of Advanced CAD Technology group at Fujitsu Labs. of America for their helpful suggestions. We would also like to thank the members of Server CAD group and Technology group for supporting this development. Special thanks go to Yuji Oinaga for encouraging us to write this paper.

7. REFERENCES

[1] H. Ando, Y. Yoshida, et al, “A 1.3GHz Fifth Generation SPARC64 Microprocessor,” Proceedings International Solid-State Circuits Conference, pp. 246-247, 2003.

[2] H. Ando, Y. Yoshida, et al, “A 1.3GHz Fifth Generation SPARC64 Microprocessor,” Proceedings Design Automation Conference, pp. 702-705, 2003.

[3] N. Ito, H. Komatsu, et al, “A Physical Design Methodology for 1.3GHz SPARC64 Microprocessor,” Proceedings International Conference on Computer Design, pp. 204-210, 2003.

[4] A. Inoue, "SPARC64™ V/VI for Mission-Critical Servers," presented at Fall Processor Forum, 2004.

[5] K. Baker, G. Gronthoud, et al, “Defect-Based Delay Testing of Resistive Vias-Contacts A Critical Evaluation,” Proceedings International Test Conference, pp. 467-476, 1999.

[6] G. Vandling, "Modeling and Testing the GEKKO Microprocessor, an IBM PowerPC Derivative for Nintendo,” Proc. International Test Conference, pp.593-599, 2001.

[7] K. S. Kim, S. Mitra, et al, "Delay Defect Characteristics and Testing Strategies," IEEE Design & Test of Computers, vol.20, issue 5, pp.8-16, 2003.

[8] J. A. Waicukauski, et al, “Transition Fault Simulation by Parallel Pattern Single Fault Propagation,” Proceedings International Test Conference, pp. 542-549, 1986.

[9] G. L. Smith, “Model for Delay Faults Based on Paths,” Proceedings International Test Conference, pp. 342-349, 1985.

[10] J. Rearick, “Too Much Delay Fault coverage Is a Bad Thing,” Proceedings International Test Conference, pp. 624-633, 2001.

[11] W. C. Lai, A. Krstic, et al, "On Testing the Path Delay Faults of a Microprocessor Using its Instruction Set," Proceedings VLSI Test Symposium, pp.15-20, 2000.

[12] D. Maruyama, A. Kanuma, N. Ito, et al, “Detection of Multiple Transitions in Delay Fault Test of SPARC64 Microprocessor”, Proceedings International Conference on Computer Aided Design, pp. 893-898, 2004.

[13] P. Gillis, K. McCauley, et al, "Low Overhead Delay Testing of ASICs", Proceedings International Test Conference, pp. 534-542, 2004.