Optimal Periodic Testing of Intermittent Faults In Embedded Pipelined Processor Applications

N. Kranitis [1] A. Merentitis [1] N. Laoutaris [1] G. Theodorou [1]
A. Paschalis [1] D. Gizopoulos [2] C. Halatsis [1]
[1] *Dept. of Informatics & Telecom. Univ. of Athens, Greece* – nkran@di.uoa.gr
[2] *Dept. of Informatics, Univ. of Piraeus, Greece*

Abstract

Today's nanometer technology trends have a very negative impact on the reliability of semiconductor products. Intermittent faults constitute the largest part of reliability failures that are manifested in the field during the semiconductor product operation. Since Software-Based Self-Test (SBST) has been proposed as an effective strategy for on-line testing of processors integrated in non-safety critical low-cost embedded system applications, optimal test period specification is becoming increasingly challenging.

In this paper we first introduce a reliability analysis for optimal periodic testing of intermittent faults that minimizes the test cost incurred based on a two-state Markov model for the probabilistic modeling of intermittent faults. Then, we present for the first time an enhanced SBST strategy for on-line testing of complex pipelined embedded processors. Finally, we demonstrate the effectiveness of the proposed optimal periodic SBST strategy by applying it to a fully-pipelined RISC embedded processor and providing experimental results.

1. Introduction

Current semiconductor fabrication processes for nanometer technology have a very negative impact to manufacturing yield and in-field reliability. The reliability faults manifested in the field during the life cycle of semiconductor products are classified in the following three types. *Permanent faults* reflect irreversible physical changes. Improvements in semiconductor design and manufacturing have significantly decreased the occurrence of permanent faults. *Intermittent faults* appear repeatedly at the same location causing errors in bursts when they are active, because of unstable or marginal hardware mainly due to process variations and manufacturing residuals. *Transient faults* (also known as "soft-errors") appear irregularly at various locations and last short time. These faults are induced by temporary environmental conditions such as neutron and alpha particles, power supply and interconnect noise, electromagnetic interference and electrostatic discharge.

Concurrent error detection, self-checking, time and space redundancy mechanisms that significantly increase system cost [1] are not suitable for non safety-critical low-cost embedded system applications. In such embedded applications detection of intermittent operational faults, that cause errors in bursts only when they are active and may precede the occurrence of permanent faults, is much more important than detection of transient operational faults that appear once and last short time.

On-line periodic testing is a non-concurrent test strategy, well suited to such embedded systems since it provides low-cost detection of permanent and intermittent faults with very high probability, that trades off between fault detection latency and performance overhead.

On-line periodic testing of deeply embedded processors in complex low-cost and non safety-critical embedded systems is becoming increasingly challenging. Hardware-based built-in self-test (BIST) techniques for on-line periodic testing provide excellent test quality and at-speed testing at the cost of increased hardware overhead, high power consumption and manual extensive design changes. Software-based self-test (SBST) in several variations on [2]-[9] has been proposed as a very promising approach for at-speed processor testing using low-cost structural testers. SBST is a non-intrusive approach that utilizes processor resources and Instruction Set Architecture (ISA) to perform test generation, test application and test response evaluation. Recently, the use of low-cost SBST techniques for on-line periodic testing of embedded processors has been proposed in [8] as an effective alternative solution to hardware-based BIST.

In this paper first we present a reliability analysis of intermittent faults (also including permanent faults) using probabilistic modeling. For on-line periodic testing of embedded processors, we introduce a novel cost function in order to minimize the test cost incurred by the execution of SBST programs and achieve high detection probability. Then, we present for the first time an enhanced SBST strategy for on-line periodic testing of embedded pipelined processors with more advanced ISA than the publicly available benchmarks used so far (e.g. [2],[3],[4],[7],[8]) and demonstrate the effectiveness by applying it to a fully-pipelined RISC embedded processor.

3-9810801-0-6/DATE06 © 2006 EDAA

2. Reliability analysis

2.1 Test model

We consider a two-state continuous-parameter Markov model [8], [10], [11] for the intermittent faults. Let λ and μ denote the rates of leaving state 0 (operating) and state 1 (fault), respectively, and let also S_t denote the state at time t. For this model, the time period during which the embedded processor stays at the operating (state 0) or the faulty (state 1) time has an exponential distribution with mean time the parameters $1/\lambda$ and $1/\mu$, respectively. Thus, the transition probabilities $P_{i,j}(t)$ from state i at time t_0 to state j at time t_0+t are:

$$P_{0,1}(t) = \frac{\lambda}{\lambda+\mu}\left(1 - e^{-(\lambda+\mu)t}\right)$$

$$P_{0,0}(t) = 1 - P_{0,1}(t)$$

$$P_{1,0}(t) = \frac{\mu}{\lambda+\mu}\left(1 - e^{-(\lambda+\mu)t}\right)$$

$$P_{1,1}(t) = 1 - P_{1,0}(t)$$

The steady-state probabilities of being at each one of the two possible states upon an arbitrary observation instant, are $\pi_0=\mu/(\lambda+\mu)$ and $\pi_1=\lambda/(\lambda+\mu)$ respectively.

2.2 Cost function derivation

The embedded processor that operates under this intermittent fault is used for the processing of an endless stream of jobs, with each job requiring a processing time B. If the fault becomes active during the duration of a job it corrupts the final result of the processing thus making it useless and potentially dangerous.

Consider now a SBST program for the detection of the aforementioned intermittent faults. Each test is assumed to have a finite duration D. During the execution of a test: (i) the processing of the current job is halted; (ii) the intermittent fault remains at the state that it was at the beginning of the test (i.e., we assume that the two-state model "freezes" for the duration of the test). We assume a perfect detection capability for the faults covered by the SBST program and, thus, at the completion of the test we always detect a fault if one is active at the beginning of the test. When a fault is detected, the results of the processing are discarded and the job re-enters processing from the beginning. The time diagram (Figure 1) shows the test period T, the test duration D, the processing time lost due to a flawed results R and the number of tests performed N under the given period and job duration B.

We are focusing on a periodic testing strategy in which the nth test is executed at time $t_n = n \cdot T$, $1 \le n \le N$, where T is the testing period and $N = \lfloor B/T \rfloor$ is the total number of tests performed during a single job (for simplicity we disregard the floor function in the sequel). The presented reliability analysis does not require the test duration D to be much less than the test period T, as it was required in previous works [8], [10], [11].

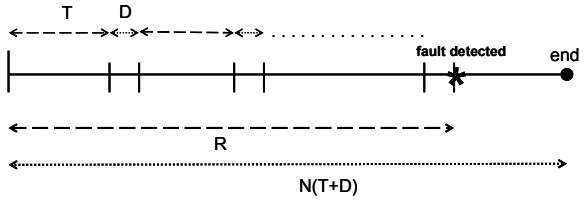

Figure 1 Time diagram

Let p_d denote the probability of detecting a fault with a single test; $p_{fd}(n)$ the probability of detecting the fault (for the first time) with the nth test and p_{fd} the probability of detecting the fault with any one of the N tests.

For the above intermittent fault model and testing strategy, we have:

- $p_d = P_{0,1}(T) = \frac{\lambda}{\lambda+\mu}\left(1 - e^{-(\lambda+\mu)T}\right)$
- $p_{fd}(n) = (1-p_d)^{n-1} p_d = P_{0,0}(T)^{n-1} \cdot P_{0,1}(T)$
- $p_{fd} = \sum_{n=1}^{N} p_{fd}(n) = 1 - P_{0,0}(T)^{\frac{B}{T}}$

Let q denote the probability that an existing fault becomes active (any number of times) during the time interval [0, B] that is required for finishing a single job. Then, the complementary probability, i.e., for the fault to stay inactive during the job although it exists, is $1 - q = P\{E_1 \cap E_2\}$, where E_1 is the event that the fault is inactive at $t = 0$ and E_2 is the event that the fault remains inactive during the interval [0, B]. The probability for the fault to stay inactive during the job can be re-written as $1 - q = P\{E_1\} \cdot P\{E_2|E_1\}$. The probability of the event E_1 is equal to the probability of the event ($S_0 = 0$), which is given by π_0, the steady-state probability of being at the operating state. The probability of the event $E_2|E_1$ is equal to $e^{-\lambda B}$. Thus, the probability for the fault to stay inactive during the job is $1 - q = \pi_0 e^{-\lambda B}$. Consequently the probability that the fault becomes active during the time interval [0, B] is $q = 1 - \pi_0 e^{-\lambda B}$.

Let d denote the detection probability that an existing fault becomes active (any number of times) during the time interval [0, B] and is detected with anyone of the N tests. Thus, the detection probability is $d = q \cdot p_{fd}$. In practice, it is required that d must be greater than $1-\varepsilon$, where ε is a prespecified value (e.g. $\varepsilon = 10^{-5}$).

Let us consider the case that the fault becomes active one or more times during the processing of a job thereby corrupting the final result and making it useless. Let R denote the amount of processing time wasted in this case:

$$R = \begin{cases} n \cdot (T+D), & \text{if the fault is detected by the } n\text{th test}, 1 \le n \le B/T \\ N \cdot (T+D), & \text{if the fault stays undetected till the end of the job} \end{cases} \quad (1)$$

The first case amounts to the total processing time that is wasted when a fault is detected prior to the completion of the ongoing job, thus causing the discard of the partial result and the restart of the job. The second case amounts to the total processing time that is effectively wasted by producing a flawed result when errors occur but remain undetected. Using Eq. (1) the expected value $E\{R\}$ of R, can be written as follows:

$$E\{R\} = B/T \cdot (T+D) \cdot (1 - p_{fd}) + \sum_{n=1}^{B/T} n \cdot (T+D) \cdot p_{fd}(n)$$

In the following we define the *cost function C(T)* that expresses the cost incurred when selecting a periodic test strategy with period T.

$$C(T) = (1-q) \cdot N \cdot D + q \cdot E\{R\} \quad (2)$$

The first part of $C(T)$ amounts to processing time unnecessarily wasted in running tests, when in fact no faults have occurred (an event of probability $(1-q)$). This part of the cost function favors the selection of large T's. The second part of $C(T)$ amounts to the processing time wasted when faults occur (an event of probability q). If the fault is detected by the periodic test procedure, then no additional time is wasted in processing a result that is already flawed. If the detection fails, then the maximum penalty is incurred, which is equal to the entire duration of the processing. Thus, the second part of $C(T)$ favors the selection of small T's (i.e., it favors frequent tests).

The optimal test period T^* can be derived from the root of the following equation

$$\frac{dC(T)}{dT} = 0 \quad (3)$$

Equation (3) does not have a simple closed-form solution as it includes complex combined polynomial and exponential forms of T. Thus, it can only be solved numerically for an exact solution, using for example Newton's method.

2.3 Numerical and case study examples

Let us consider the following device parameters as a numerical example: λ = 0.01 ms^{-1} and μ = 0.1 ms^{-1}. The embedded application job processing time is B=1000ms and the duration of the SBST test program is D=0.3ms (according to experimental results of Section 3 that follows). The cost function $C(T)$ for the above device parameters is depicted in Figure 2. The optimal test period is T^*=2.24ms while the total number of tests performed is N=446.

Let us also consider different device parameters, with a range of the ratio μ/λ between 0 and 10. Intermittent fault modeling requires $\mu > \lambda$ while the corner case μ=0 represents a permanent fault.

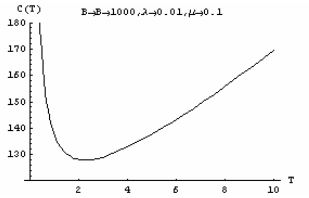

Figure 2 Cost of test and optimal test period

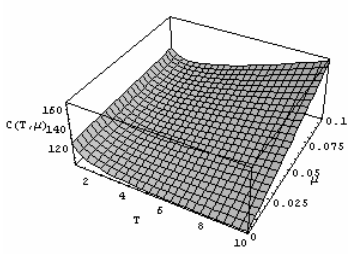

Figure 3 Cost of test as a function of T and μ

The proposed cost function $C(T)$ for the above device parameters is shown in Figure 3. It can be easily seen that if the mean time $(1/\mu)$ spent by the device in state 1 (fault state) becomes smaller, the optimal test period gets smaller as well. For the corner case μ=0 representing a permanent fault, the cost function leads to an optimal test period T^* with the largest value. In Table 1, we have calculated the optimal test period T^* for several different values of device parameter μ.

μ/λ	μ (ms^{-1})	Optimal test period T^*(ms)	Number of tests N
0	0	7.64	130
1	0.010	5.38	185
2.5	0.025	4.04	247
5	0.050	3.06	326
7.5	0.075	2.56	390
10	0.100	2.24	446

Table 1 Optimal T^* for different device parameters

Figure 4 depicts the detection probability d that an intermittent fault becomes active during the time interval [0, B] and is detected with any one of the N tests. It is clear that for all the device parameters considered and the optimal test periods of Table 1, there is very high detection probability.

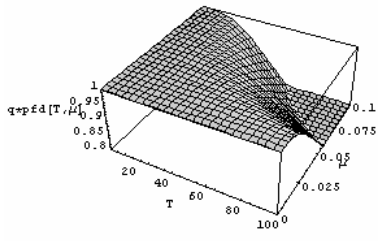

Figure 4 Detection probability in periodic testing

3. SBST strategy for on-line periodic pipelined processor testing

On-line periodic testing is performed in-field while the processor operates at its normal operational environment. Software-Based Self-Test (SBST) has been proposed as an effective strategy for on-line testing of processors integrated in non-safety critical low-cost embedded system applications [8]. The SBST program is resident in the cache memory of the embedded core and is executed at the processor's actual speed (at-speed testing). The above unique characteristics of on-line periodic testing impose several additional constraints in comparison to off-line manufacturing testing.

In Section 2, we introduced reliability analysis for optimal periodic testing of intermittent faults that minimizes the test cost incurred when running the SBST program based on probabilistic modeling of intermittent faults. In this Section, based on the component-based SBST methodology for off-line [4], [7] and on-line testing [8] we developed self-test routines that comply with the stringent characteristics of on-line testing. We present for the first time an enhanced SBST strategy for on-line testing of complex pipelined embedded processors. Moreover, we address successfully the new test challenges related to more advanced processor architectures than the publicly available processor benchmarks used so far (e.g. [2],[3],[4],[7],[8]).

3.1 Self-test routine development

On-line periodic testing is often applied on mobile platforms where power consumption is a critical factor. A study by Intel [12] shows that a considerable part of the total power dissipation concerns the memory system and specifically the cache hierarchy system. Furthermore, the system's performance specifications are also an important factor, thus special care must be taken so that the overall performance is not degraded overmuch due to the execution of the test routine. This means that the test routine should have the shortest possible test execution time (less than a quantum cycle).

In order to satisfy these requirements, self-test routines are developed in a way that they utilize the characteristics of temporal and spatial locality. According to this, test routines include compact loops that take advantage of temporal locality and data structured in arrays that exploit spatial locality. In addition, references to the data memory should be kept to a minimum. This was achieved by the use of a software MISR routine that performs compaction of the test responses so that only a single signature per component is written in the data cache, keeping cache references that result in cache miss overhead and increase in power consumption to a minimum.

Since on-line testing is performed in-field at the normal operational environment and under the restrictions of the operating system, on-line embedded processor testing is much more challenging than off-line testing. For example, only a limited part of memory hierarchy can be accessed in normal mode of operation, thus resulting in lower fault coverage in certain processor units like the instruction fetch unit and other address related units. However, these faults should be considered as functionally untestable. In addition, self-test code that minimizes references to memory hierarchy including stack memory imposes increased test challenges for certain processor units.

3.2 Self-test program enhancement

Modern embedded systems include complex high performance embedded processors that can handle real time applications with hard deadlines and increased throughput needs. Those requirements are satisfied by the use of advanced ISAs with characteristics like Instruction-Level Parallelism, system coprocessors support, etc. Multi-stage pipeline and exception support systems are very common in the embedded processors. Consequently, SBST should target those units in order to guarantee higher fault coverage and thus more reliable real time operation.

The goal of achieving high fault coverage in complex pipeline systems comprises several challenges that must be addressed. The unique character of the processor pipeline as a performance increase mechanism not visible to the assembly programmer, leads to reduced controllability and observability restrictions in some of its subcomponents. Specifically:

- Hazard Detection unit is a control component with limited visibility to assembly programmers resulting in reduced controllability and observability.
- Pipeline control is a control component that decodes the processor instruction at the instruction fetch pipeline stage producing the appropriate control signals for the subsequent pipeline stages. This implies limited observability.
- Pipeline registers are not directly visible to the assembly programmer. However, the Data part is much more testable than the Address part. Moreover, limited access to the memory hierarchy in normal mode operation imposed by the operating system, results in low fault coverage in the high order bits of the pipeline address registers. Faults in such pipeline register logic should be considered functionally untestable during on-line SBST. The same reasoning is valid for the instruction fetch unit.
- Forwarding logic implemented by forwarding multiplexers is not visible. Despite the fact that the controllability and observability is limited, forwarding multiplexers are functional components that can be targeted effectively by deterministic test routines that apply regular test patterns and guarantee very high fault coverage.

The role of the exception support system to interrupt the normal mode operation of a processor differentiates the challenges related to its on-line periodic testing.

Specifically:

- The part of the exception register that holds the address where the exception occurred and the exception controller, provides reduced controllability and observability.
- Exceptions that are generated by the ISA can be triggered by the SBST routine. On the other hand, exceptions triggered by external interrupts cannot be triggered by the SBST routine, thus resulting in part of the exception control to be SBST untestable.

We enhanced the on-line self-test program of [8] by adding self-test routines that target pipeline and exception logic using a deterministic approach for the functional units and a functional testing approach for the control units. Forwarding logic implemented by forwarding multiplexers is targeted by deterministic tests applied by proper instruction sequences and propagated to fully observable GPRs. The data part of the pipeline registers was targeted by deterministic test patterns. The address part was also targeted deterministically and special care was taken so that all address logic test responses are propagated to the processor data bus, since the data memory is the processor primary output where test response capturing can take place. The test routine targeting the exception register that holds the address where the exception occurred, follows a similar approach.

All the pipeline and exception control logic are targeted using standard verification-based functional testing techniques with test development performed at high-level as in [7]. Use of high-level RTL verification metrics supported by industry standard simulation tools like RTL statement, branch, condition and expression coverage, helps to improve verification manual effort. In many practical cases, verification-based test routines are developed at the design verification phase. Test code can be reused for the testing of control and hidden components with no additional manual effort involved or in the worst case substantially alleviating any manual self-test routine development effort.

4. Experimental results

In this section, the effectiveness of the SBST strategy for on-line periodic testing is demonstrated by its application to a complex fully pipelined processor. *Athena* is an in-house developed processor designed to fulfill the requirements of a fully functional complex processor benchmark. It is a 32-bit embedded RISC processor core that implements a 5-stage pipeline with hazard detection and forwarding mechanisms [13]. *Athena* implements the full MIPS-I ISA with the sole exception of the unaligned load and store operations that are patented. The processor core is enhanced with a fast parallel multiplier [14] and exception handling mechanisms that supports four types of exceptions: unknown command, external interrupt, ALU overflow and commands that cause exception (Syscall or Break in MIPS-I ISA). The RTL processor model was synthesized targeting a 0.18um technology library. Synthesis was optimized for area and the netlist gate-count was 26,432 gates with 1518 FFs. The design runs at a clock frequency of 89 MHz. A *Test Evaluation Framework* using commercial tools was used for VHDL synthesis, functional and fault simulation.

According to [7], [8] the processor units that have the highest priority for on-line periodic testing are the functional *Data-Visible Components* [8] (parallel multiplier, register file, ALU, shifter and sign extension unit), which occupy the 77.8% of the processor area. Targeting only these functional components results to acceptably high total fault coverage of 89.4% due to collateral coverage. The enhanced test program increases the fault coverage to 96.67% as a result of targeting explicitly the pipeline and exception logic. It should be clarified that a large number of the undetected faults are *functionally untestable* using processor instruction at the normal mode of in-field operation. However, currently there is no tool available for systematic extraction of such functionally untestable faults.

The derived self-test program for on-line periodic testing has the required stringent characteristics:

- Small code of only 3025 words, that takes advantage of temporal locality and spatial locality.
- A small number of only 42 memory data references (19 stores and 23 loads) that imposes very few data cache misses.
- A short CPU execution time of 23,337 clock cycles and an absolute time of 260us, which is much less than a quantum time cycle.

For every self-test routine, a test signature is derived after compaction of all responses by using a shared software MISR subroutine of only eight words. At the end of periodic testing, nine signatures one for every CUT plus one for the functional test are stored to data memory.

Table 2 presents the component gate count, self-test program statistics (program size in words, CPU clock cycles and data memory references – loads and stores), along with the achieved single stuck-at fault coverage and the percentage of the processor overall fault coverage which is missing from each of the processor components.

As in previous work [7], [8] the attained fault coverage for the processor functional units is almost complete. Deterministic tests, which utilize the inherent regularity that characterizes the architecture of several processor functional components, are applied by compact self-test routines with high structural test coverage for any gate-level implementation. This inherent regularity is not exploited either by pseudorandom test development or by ATPG-based test development approaches.

Processor components related to address logic, called *Address-Visible Components* [8] have reasonably high fault coverage, considering that the high order bits of the address bus are functionally untestable during periodic on-line testing.

Control components have fairly low fault coverage due to poor controllability and observability characteristics. However, these components are very small in comparison to the functional components, resulting in satisfactory fault coverage for the entire processor. Verification-based functional testing of these components was performed according to [7] by taking advantage of existing software platforms. Through re-usability, any manual self-test routine development effort is substantially alleviated. *Athena Processor Verification Suite (APVS)* was developed at the design phase of *Athena* and produces automatically verification-based functional test code.

The improvement in fault coverage achieved by the enhanced test program specifically in the pipeline parts is depicted in Table 3.

Pipeline Parts	[7], [8] FC (%)	Miss. FC over All (%)	Enhanced FC (%)	Miss. FC over All (%)
Pipe Control	64.7	0.44	83.2	0.32
Pipe Reg. Data	74.6	1.07	96.8	0.16
Pipe Reg. Addr.	62.9	0.41	78.8	0.25
Forward. MUXs	85.9	0.31	100.0	0.00
Hazard Detect.	54.6	0.23	92.5	0.08
Total Pipeline	73.5	2.46	92.9	0.56

Table 3 Fault coverage for pipeline components

5. Conclusions

The contribution of the paper is twofold: First, we have presented a reliability analysis of intermittent faults using probabilistic modeling and we have introduced a novel cost function in order to minimize the test cost incurred by the execution of SBST programs for on-line periodic testing of embedded processors. Secondly, we have presented for the first time an enhanced SBST strategy for on-line periodic testing of embedded processors with more advanced ISA than the publicly available processor benchmarks used so far (e.g. [2],[3],[4],[7],[8]). We have demonstrated the effectiveness of the proposed optimal periodic SBST strategy by applying it to a fully-pipelined RISC embedded processor and providing a set of experimental results.

References

[1] M. Nicolaidis, Y. Zorian, "On-line Testing for VLSI – A Compendium of approaches", in *Journal of Electronic Testing: Theory and Applications*, v.12,1-2, 1998, pp. 7-20.

[2] J. Shen, J. Abraham, "Native mode functional test generation for processors with applications to self-test and design validation", in Proc. of *IEEE International Test Conference* 1998, pp. 990-999.

[3] L. Chen, S. Dey, "Software-Based Self-Testing Methodology for Processor Cores", *IEEE Transactions on CAD of Integrated Circuits and Systems*, vo.20, no.3, pp. 369-380, March 2001.

[4] N. Kranitis, D. Gizopoulos, A. Paschalis, Y. Zorian, "Instruction-Based Self-Testing of Processor Cores", in *Proc. of the IEEE VLSI Test Symposium* 2002, pp. 223-228

[5] P. Parvathala, K. Maneparambil, W. Lindsay, "FRITS – A Microprocessor Functional BIST Method", in *Proc. of the IEEE International Test Conference* 2002, pp. 590-598.

[6] L. Chen, S. Ravi, A. Raghunathan, S. Dey, "A Scalable Software-Based Self-Testing Methodology for Programmable Processors", in *Proc. of the Design Automation Conference (DAC)* 2003, pp. 548-553.

[7] N. Kranitis, A. Paschalis, D. Gizopoulos, G. Xenoulis, "Software-Based Self-Testing of Embedded Processors", *IEEE Transactions on Computers*, vol. 54, no. 4, pp. 461-475, April 2005.

[8] A. Paschalis, D. Gizopoulos, "Effective software-based self-test strategies for on-line periodic testing of embedded processors", *IEEE Transactions on CAD*, Vol. 24, no.1, pp. 88 – 99, Jan. 2005.

[9] M. Hatzimihail, M. Psarakis, G. Xenoulis, D. Gizopoulos, A. Paschalis, "Software-Based Self-Test for Pipelined Processors: A Case Study", in *Proc. of the Defect and Fault Tolerance in VLSI Systems (DFT'05),* 2005, pp. 535-543

[10] S.Y.H. Su, I. Koren, Y.K. Malaiya, "A Continuous-Parameter Markov Model and Detection Procedures for Intermittent Faults", *IEEE Transactions on Computers*, vol. c-27, no. 6, June 1978, pp. 567-570.

[11] T. Nakagawa, K. Yasui, "Optimal Testing-Policies for Intermittent Faults", *IEEE Transactions on Reliability*, vol. 38, no. 5, Dec. 1989, pp. 577-580.

[12] Intel "Mobile Power Guidelines 2000", Dec. 11, 1998

[13] D. Patterson, J. Hennessy, "*Computer Organization and Design: The Hardware/Software Interface*", MKP 1997.

[14] J. Pihl, E. Sand, Arithmetic Module Generator, http://www.fysel.ntnu.no/modgen

Component	Gate Count (gates)	[7], [8] Fault coverage (%)	New Fault coverage (%)	Size (words)	Clock Cycles	Data References	Missing FC over All (%)
Parallel Mult.	8,746	98.3	98.3	113	9,469	1	0.82
Register File	9,669	99.9	99.9	1,408	4,224	1	0.03
Shifter	861	100.0	100.0	190	721	1	0.00
ALU	579	98.2	98.4	178	1,533	1	0.06
Control Logic	457	62.1	76.6	596	5,492	8	0.44
Pipeline	3676	73.5	92.9	412	1,399	5	0.56
Exception Unit	574	31.2	83.3	31	244	4	0.09
Sign Extension Unit	176	100.0	100.0	68	131	17	0.00
Instruction Fetch Unit	1078	58.1	80.7	29	124	4	0.86
Remaining *(not target)*	616						0.47
Total	26,432	89.4	96.67	3,025	23,337	42	3.33

Table 2 Test statistics for processor targeted components