Error Prediction based on Concurrent Self-test and Reduced Slack Time

Valentin Gherman, Julien Massas, Samuel Evain, Stéphane Chevobbe, Yannick Bonhomme
CEA, LIST, Embedded Systems Reliability Laboratory,
Point Courrier 94, 91191 Gif-sur-Yvette CEDEX, FRANCE
firstname.lastname@cea.fr

***Abstract*-Small circuit defects occurred during manufacturing and/or enhanced/induced by various aging mechanisms represent a serious challenge in advanced scaled CMOS technologies. These defects initially manifest as small delay faults that may evolve in time and exceed the slack time in the clock cycle period. Periodic tests performed with reduced slack time provide a low-cost solution that allows to predict failures induced by slowly evolving delay faults. Unfortunately, such tests have limited fault coverage and fault detection latency.**

Here, we introduce a way to complement or completely replace the periodic testing with reduced slack time. Delay control structures are proposed to enable arbitrarily small parts of the monitored component to switch fast between a normal operating mode and a degraded mode characterized by a smaller slack time. Only two or three additional transistors are needed for each flip-flop in the monitored logic. Micro-architectural support for a concurrent self-test of pipelined logic that takes benefit of the introduced degraded mode is presented as well. Test stimuli are produced on the fly by the last two valid operations executed before each stall cycle. Test result evaluation is facilitated by the replication of the last valid operation during a stall cycle. Protection against transient faults can be achieved if each operation is replicated via stall cycle insertion.

I. INTRODUCTION

For decades, continuous scaling of CMOS manufacturing processes has been a way to renew the generations of integrated circuits. Unfortunately, this trend brought the emergence of greater reliability concerns as higher substrate temperatures, current densities and electrical fields lead to more pronounced circuit degradation over time [6][10][20][30][36].

In order to preserve system functionality, a temporal guard-band, also called slack time, of about 10-20% of the clock cycle period is currently used [1]. In advanced scaled CMOS technologies, increasingly large temporal guard-bands are required by worst case degradation [6]. In the same time, the filtering of the weak circuits immediately after their manufacturing based on burn-in testing becomes less effective due to reduced acceleration factors and insufficient fault coverage [13]. Especially, the gap between normal and burn-in test conditions is reduced by thermal run-away effects that manifest as a positive feedback loop between temperature and leakage current [19].

Periodic self-test is a complementary technique that enables the detection of aging-induced defects [16][18][29]. Concurrent self-test enables a component [28] or a system [12] to test itself concurrently with its normal operation. In this way, fault detection latency and fault coverage may be improved as the tests can be applied more often and under more realistic workload and environmental conditions. Since most of the target defects manifest as a small timing degradation which may worsen over time [9], failure prediction can be achieved if the tests are performed in a *degraded mode* where the slack time [16][29] and/or the power supply voltage [8] are reduced as compared with a normal operating mode. The test rate and the difference between the circuit parameters in test and normal operating modes need to be adapted such that defects that induced no error during a test campaign have no time to evolve and provoke an error until the next test campaign [9].

In this paper, we present an approach to improve the fault coverage and the fault detection latency of the self-test performed in degraded mode. *Delay control structures* (DCSs) are proposed to offer (a) more flexibility for the selection of the degraded zones and (b) fast switching between degraded and normal operating modes. The latency of a circuit can be adjusted with a quantifiable amount if a DCS is applied to each sequential element at the end of the logic paths in the considered circuit. Nevertheless, the hardware overhead is limited as a DCS contains only two or three transistors.

Another achievement of this paper is a micro-architectural support for the concurrent self-test of in-order pipelines based on the opportunistic re-execution of operations during naturally occurred stall cycles. The last valid operation before a stall cycle is re-executed at the pipeline stage where the stall cycle has occurred and the replicated operation is let to propagate through the pipeline. In each pipeline stage, the last two valid operations executed before such a stall cycle provide a pair of delay test patterns. In this way, a large number and diversity of operations provide test patterns potentially detecting faults undetectable with the available periodic tests. The results of the replicated operations can be compared at the outputs of each pipeline stage or only at the pipeline outputs in order to reduce the hardware overhead. Error prediction becomes possible if the first version of each replicated operation is executed in degraded mode. This can be achieved with the help of the introduced DCSs that enable a high temporal granularity for the selection of the degraded mode. The execution of a few benchmark programs on a small processor proved that important fractions of transition delay faults [35] can be tested concurrently with the program execution.

A special mode can be easily enabled in which each operation is re-executed over the full pipeline length in order to detect potential failures induced by transient and delay faults.

State-of-the-art approaches to aging monitoring are discussed in Section II. Section III gives the definitions of some observables used to describe the temporal behaviour of flip-flops. Several DCS designs are introduced in Section IV. Section V presents micro-architectural support for a DCS-based monitoring of in-order pipelines. The paper achievements are summarized in Section VI.

978-3-9810801-7-9/DATE11/©2011 EDAA

II. Related Work

Fault-tolerant architectures and concurrent error-detection schemes that ensure systematic protection against transient and/or permanent faults require significant amounts of hardware redundancy [2][4][7][20][24][22][32].

While the transient faults are characterized by a high unpredictability [5][15], permanent faults induced by various aging mechanisms are expected to produce a gradual degradation of the circuit performance [9]. Consequently, aging models were proposed to estimate the wear-out of non-critical systems, which do not require protection against transient faults, based on prior knowledge of the executed applications [30]. Unfortunately, the accuracy of the aging models is severely limited by unknown environmental conditions, manufacturing variations, unavailable technology information and insufficiently understood physics of some aging mechanisms [25].

A different approach is to use sensors, such as inverter chains or tuneable replica circuits that are slightly slower than the critical paths in the monitored components [17][33][34]. Sensors and adaptive circuits require a substantial calibration time per die that can lead to increased testing costs. Such approaches may be suitable for sensing alterations of circuit latency due to uniform aging or voltage and temperature variations, but they are inappropriate for monitoring random defects that can provoke infant mortality or non-uniform aging.

Periodic functional self-tests performed in degraded mode provide a low-cost approach to predict aging-induced failures [16][29]. In order to better cope with the inaccuracy of the aging models [25], self-tests performed concurrently with the normal operation may be used to improve the fault detection latency and the failure prediction capability [12][28]. A way to avoid the impact on system performance is to execute these tests during idle periods. The higher the temporal granularity at which these idle periods can be exploited, the better the fault prediction and the fault detection latencies are [28]. Unfortunately, opportunities to perform fine grain self-testing in degraded mode are restricted by the limited spatial granularity of the voltage-frequency islands and the relatively high latency of their control infrastructure. This problem is addressed in the *BulletProof* approach where circuit components are tested concurrently in degraded mode with the help of an additional clocking tree [28]. Our DCSs enable degraded mode self-testing with high temporal and spatial granularities without requiring a supplementary clock tree.

Concerning the utilization of stall cycles for the opportunistic re-execution of operations in pipelined logic, at least two approaches need to be mentioned. Instruction re-execution during course-grain idle periods induced by high latency memory accesses has been proposed in the *Introspection* approach to enable transient fault detection [26]. This approach requires large storage buffers and rather complex control logic. In the *BulletProof* approach, deterministic test patterns for stuck-at faults are applied during stall cycles [28]. Unfortunately, the stuck-at fault model is not coherent with the degraded mode testing whose focus is to uncover small delay faults. Consequently, the error prediction potential is not used and the system is provided with a check-pointing scheme implemented at the cache memory level. Despite the use of deterministic stuck-at test patterns, the resulting pairs of delay test patterns are still random.

In our approach, the replication of operations facilitates the application of test patterns generated on the fly by the monitored logic itself. No hardware overhead is required for the storage of test patterns and test response signatures. Even if only naturally occurred stall cycles are used, a large number and variety of test stimuli will be applied before an emerging delay fault, which is detectable in degraded mode testing, could affect the normal operating mode.

III. Terminology

Flip-flops are essential elements in synchronous circuit design. Their temporal behaviour can be characterised by the setup time τsetup, the hold time τhold and the clock-to-Q delay τ_{clk2Q}. τ_{setup} relates to the time interval before the active clock edge during which the data at the flip-flop input (D) must be valid for reliable latching. Similarly, τ_{hold} represents the time that the data input must be held stable after the active clock edge. τ_{clk2Q} is defined as the time interval between the active clock edge and the arrival of the valid new data at the flip-flop output (Q). τ_{setup} and τ_{hold} depend strongly on each other: τ_{setup} decreases/increases as τhold increases/decreases. Traditionally, the pairs of τsetup and τ_{hold} are selected such that τ_{clk2Q} increases by 10% with respect to its value when the input data is stable long before and after the active clock edge.

IV. Delay Control Structures (DCSs)

This section introduces DCSs in order to control the size of the slack time τ_{slack} usually included in the clock cycle period T_{clk} of synchronous circuits to account for circuit aging and for process, voltage and temperature variations. Fig. 1 (a) presents the main subdivisions of T_{clk} which besides τ_{slack} also includes the latencies of the combinational logic, $\tau_{combinational}$, and of the sequential logic, τ_{setup} and τ_{clk2Q}. Fig. 1 (b) illustrates a different configuration of the clock cycle period in which τ_{setup} and τ_{clk2Q} are extended to the detriment of τ'_{slack} ($\Delta\tau_{slack}=\tau_{slack}-\tau'_{slack}=\Delta\tau_{setup}+\Delta\tau_{clk2Q}$). Consequently, emerging delay faults induced by aging defects will produce an error in case (b) before they could affect the circuit in case (a).

The DCSs presented in Fig. 2 enable a fast switching between the two operating modes in Fig. 1. In each DCS, a current-starved transistor provides a high latency connection between the power supply and a standard circuit element, e.g. a standard cell. The principle of the current-starved transistor is usually employed in the implementation of delay looked loops (DLL) [21]. The DCSs also contain a parallel large-width transistor that can be switched-on when a low latency connection to the power supply is needed.

The DCSs of types (a) and (c) provide high and low latency

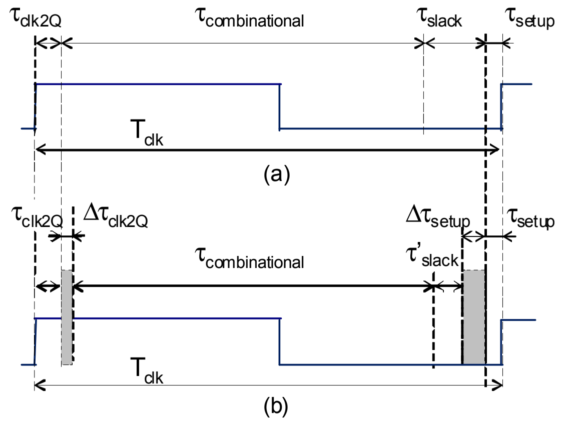

Fig. 1. Subdivisions of the clock cycle period in normal (a) and degraded (b) operating modes.

connections to Vdd depending on whether the test signal T is asserted high or low, respectively. Similarly, the DCSs of types (b) and (d) allow the selection of high or low latency connections to Vss based on the value of the inverted test signal nT (nT = not T). The Vdd' and Vss' nodes can be connected to any internal node of a standard circuit element that otherwise would have been connected to Vdd or Vss, respectively. Despite the fact that these DCSs can be used with both sequential and combinational logic, only their application to the sequential standard cells will be considered. This choice is motivated by the fact that acting on sequential elements may offer a low-cost and generic approach to control the slack time of all logic paths in a pipelined design.

The DCSs presented in Fig. 2 have been applied to the flip-flop shown in Fig. 3. If connected to a single NAND gate, a single DCS can affect the propagation time of either rising or falling transitions. DCSs of types (a) and (c) can slow down the falling transitions at the input of a NAND gate while the DCSs of types (b) and (d) can affect only the rising transitions. Consequently, if a single DCS is used, it should be applied to several NAND gates inside the considered flip-flop such that both rising and falling transitions at the flip-flop input can be delayed.

Table I reports the impact of the proposed DCSs on the setup time, τ_{setup}, and the total latency, $\tau_{D2Q}=\tau_{setup}+\tau_{clk2Q}$, of the flip-flop in Fig. 3 implemented in a 32 nm technology [37]. The first column indicates which of the DCSs presented in Fig. 2 has been used. The second column gives the indices of the NAND gate to which DCSs have been applied as shown in Fig. 2. In order to increase the impact on τ_{setup} and prevent the degradation of the signal integrity at the flip-flop's Q-output, the DCSs have not been applied to the NAND gates 5 and 6. The purpose of this condition is to limit the DCS effect to the flip-flop alone and keep the speed of the rest of the circuit unaffected. The third column indicates whether the DCSs are activated or not. In the remaining columns, two pairs (τ_{setup}, τ_{D2Q}) are reported for falling and rising transitions at the flip-flop input. In the first pair, the setup time τ_{setup} follows the traditional definition as mentioned in Section III. The second pair ($\tau_{setup\text{-}min}$, $\tau_{D2Q\text{-}min}$) corresponds to a definition of the setup time that enables a better evaluation of the DCS impact on the flip-flop latency. According to this new definition, $\tau_{setup\text{-}min}$ is defined as the data arrival time τ_{D2clk} for which the flip-flop latency τ_{D2Q} is minimal. Fig. 4 illustrates that the minimum of τ_{D2Q} occurs for a data arrival time τ_{D2clk} that is smaller than the setup time given by the traditional definition mentioned in Section III. Table I shows that the second setup time definition provides sensibly lower τ_{D2Q} values when the flip-flop in Fig. 3 is affected by DCSs. The gap between the two setup times is much smaller in the absence of DCSs.

Despite their very small transistor count, the proposed DCSs can provide a sensible increase of $\tau_{setup\text{-}min}$ and $\tau_{D2Q\text{-}min}$ for the considered flip-flop. With the DCSs of types (a) and (b), $\tau_{setup\text{-}min}$ is raised by up to 5 times and $\tau_{D2Q\text{-}min}$ by a factor between 1.5 and 2. The DCSs of types (c) and (d) have fewer transistors but a larger impact on the flip-flop parameters. $\tau_{setup\text{-}min}$ may be increased by up to 13 times and $\tau_{D2Q\text{-}min}$ by a factor between 3 and 5. The large impact on the flip-flop latency should not be a concern if τ_{D2Q} is extended to the

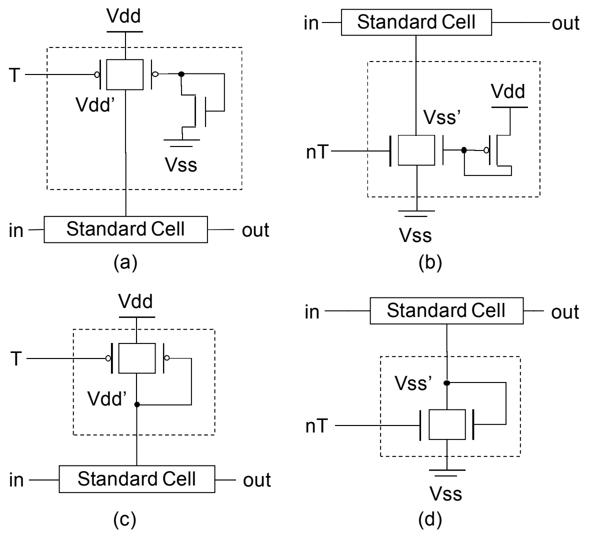

Fig. 2. Delay control structures (DCSs).

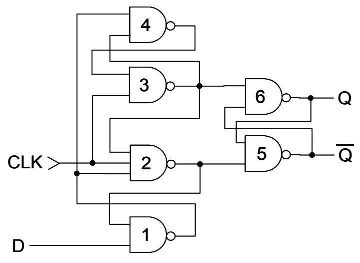

Fig. 3. Positive-edge-triggered D flip-flop.

TABLE I
DCSs IMPACT ON τ_{setup} AND τ_{D2Q} OF THE POSITIVE-EDGE-TRIGGERED FLIP-FLOP IN FIG. 3 (τ_{hold} = 10ps, 27°C)

DCS type	NAND gates connected to DCS	DCS on/off	rising transition at the D-input				falling transition at the D-input			
			traditional definition		minimal D2Q time		traditional definition		minimal D2Q time	
			τ_{setup} [ps]	τ_{D2Q} [ps]	τ_{setup_min} [ps]	τ_{D2Q_min} [ps]	τ_{setup} [ps]	τ_{D2Q} [ps]	τ_{setup_min} [ps]	τ_{D2Q_min} [ps]
no DCS	-	-	11	30	**8**	**29**	16	30	**9**	**27**
DCS (a)	1, 2, 3, 4	On	92	122	**25**	**76**	135	154	**16**	**64**
		Off	13	34	**9**	**33**	23	38	**10**	**32**
DCS (b)	1, 4	On	49	68	**21**	**50**	64	78	**33**	**56**
		Off	19	38	**13**	**36**	29	43	**15**	**36**
DCS (a)&(b)	1	On	57	76	**41**	**68**	76	90	**44**	**67**
		Off	14	33	**8**	**32**	31	45	**17**	**38**
DCS (c)	1, 2	On	157	178	**121**	**151**	130	194	**26**	**111**
		Off	18	37	**9**	**32**	18	33	**9**	**29**
DCS (d)	1, 4	On	160	201	**44**	**122**	236	256	**184**	**209**
		Off	21	40	**13**	**37**	31	45	**17**	**38**
DCS (c)&(d)	1	On	120	141	**61**	**99**	213	233	**114**	**142**
		Off	14	33	**8**	**32**	36	50	**20**	**40**

detriment of the slack time τ_{slack} as illustrated in Fig. 1.

For uniformly aging circuits in which the latencies of all signal paths are progressively increasing, the difference between the total latencies $\tau_{D2Q\text{-}min}$ in degraded and normal operating modes gives the temporal detection window $\Delta\tau_{slack}= \Delta\tau_{D2Q\text{-}min}=\Delta\tau_{setup\text{-}min}+\Delta\tau_{clk2Q\text{-}min}$ described in Fig. 1.

For isolated defects such as early life failure candidate transistors [9], the determination of an equivalent $\Delta\tau_{slack}$ is more difficult since an increase of flip-flop latencies due to a setup time violation in a pipeline stage can be tolerated by the temporal margins in the subsequent pipeline stage. Fig. 5 shows the curves that describe the dependency of the flip-flop latencies τ_{clk2Q} on the data arrival time τ_{D2clk} in degraded and normal operating modes. If these curves could be reduced to each other only by shifting them along the X-axis, then the equivalent $\Delta\tau_{slack}$ would correspond to the amplitude of this shifting. As the shapes of the two curves are different, we approximate the lower bound of the equivalent $\Delta\tau_{slack}$ with the sum of the following time intervals illustrated in Fig. 5:

- $\Delta\tau_{clk2Q\text{-}min}$ is the minimal additional latency of the flip-flop in degraded mode when the D-input signal is stable long before and after the active clock edge. $\Delta\tau_{clk2Q\text{-}min}$ corresponds to the minimal additional amount of time with which new valid signals arrive at the input of a pipeline stage in degraded mode.
- $\Delta\tau_{D2clk\text{-}min}$ is the minimal difference between the signal arrival times at the flip-flop's D-input such that the same latency τ_{clk2Q} is produced in degraded and normal operating modes. As shown in Fig. 5, it comes out that $\Delta\tau_{D2clk\text{-}min}$ corresponds to the arrival times τ_{D2clk} for which τ_{clk2Q} is infinite or the flip-flop's Q-output becomes meta-stable.

The sums $\Delta\tau_{clk2Q\text{-}min}+\Delta\tau_{D2clk\text{-}min}$ are reported in Table II for the DCSs in Fig. 2 applied to the flip-flop in Fig. 3. The DCSs of types (c) and (d) produce an important equivalent $\Delta\tau_{slack}$. When used alone, the DCSs of types (a) and (b) provide a very small equivalent $\Delta\tau_{slack}$. A low equivalent temporal detection window $\Delta\tau_{slack}$ can be compensated by frequent concurrent self-test as offered by the monitoring scheme presented in the following section.

V. Monitoring of In-Order Pipelines

The DCSs may be used to perform degraded mode testing during the stall cycles which occur from time to time in certain pipeline stages and propagate to the outputs of in-order pipelines like bubbles in a glass of sparkling water. In-order pipelines can be found in interconnection and computing components, such as networks-on-a-chip or processor cores.

A low cost concurrent test solution that also keeps the switching activity at a low level consists of repeating the last valid operation executed before a stall cycle. Such a situation is illustrated in Fig. 6 where a stall cycle occurs after the operation O_3 that can be a valid processor instruction in the case of a processor pipeline. The re-execution of an operation can be ensured by preserving the state of the buffer that drives the pipeline stage where a stall cycle is about to occur. In Fig. 6, this is the case of the buffer at the pipeline input.

The pair of operations O_2 and O_3 provides an *initialization* pattern and an *activation* pattern for the delay faults in the traversed pipeline stages. In order to allow error prediction, the results of the first instance of O_3, noted with O'_3, must be captured by flip-flops in degraded mode. The re-execution of an operation offers the possibility to re-evaluate the pipeline

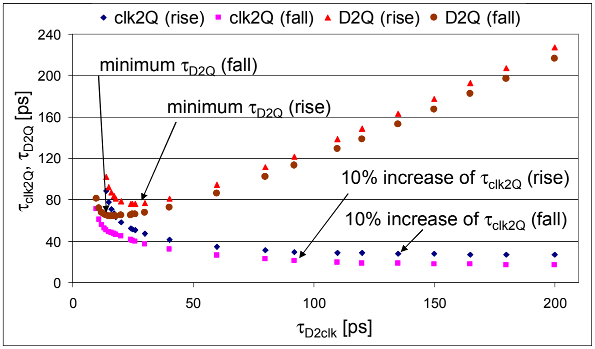

Fig. 4. Flip-flop latencies depending on the data arrival time τ_{D2clk} for a D flip-flop affected by a DCS of type (a).

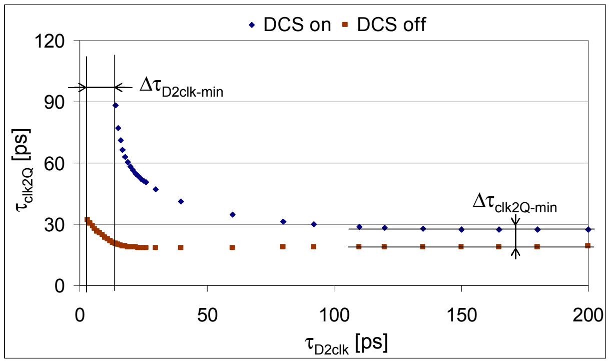

Fig. 5. Latency dependency on the data arrival time τ_{D2clk} for a D flip-flop affected by a DCS of type (a).

TABLE II
DCS-Induced Temporal Detection Windows $\Delta\tau_{slack}$

$\Delta\tau_{slack}$	DCS (a)	DCS (b)	DCS (a) & (b)	DCS (c)	DCS (d)	DCS (c) &(d)
Rise [ps]	9	1	23	84	30	41
Fall [ps]	5	7	18	45	126	77

response to the *activation* test pattern. As this re-evaluation is performed at subsequent clock cycles, delay faults can be detected even in the absence of degraded mode testing despite the execution on the same hardware resources.

If an error occurs during the execution of the first instance of a replicated instruction, this error will be propagated up to a pipeline output if it is not logically masked. In Fig. 6, a fault is activated during the execution of O'_3 in the second pipeline stage and the resulting error, indicated by the symbol 'X', is propagated to a pipeline output. If the results produced by different instances of a replicated operation are compared only at the pipeline outputs, the buffers at the pipeline outputs need to be duplicated. This enables to store the outcome of a replicated operation and to compare it with the results of the subsequent replicas.

If the logical masking of the errors propagated through several pipeline stages is important, as it can happen in processor pipelines, the error verification can be done after each pipeline stage. An XOR gate can be used to compare the input and the output of each flip-flop at the pipeline stage outputs as shown in Fig. 7 [23]. Since the amplitude of the target delay faults is limited by the temporal detection window $\Delta\tau_{slack}$ which is lower than the flip-flop latency τ_{D2Q} in degraded mode ($\tau_{D2Q}=\Delta\tau_{slack}+\tau_{setup}+\tau_{clk2Q}$), a flip-flop input signal that arrives too late to be correctly captured is still faster than the erroneously captured signal at the flip-flop's output. Consequently, both signals are stable during the subsequent clock cycle if an operation is re-executed. In this case, no buffer duplication is required. The transistor overhead is about three times smaller as compared to RAZOR or stability checkers-based schemes [2][4][22].

Systematic re-execution of operations over the full pipeline can be achieved if stall cycles are inserted each time an operation entering the pipeline is not naturally followed by a stall cycle. Systematic re-execution enables the detection of potential failures induced by transient faults and makes the detection latency quasi-inexistent. As a consequence, the degraded mode is not required anymore. In the case of a pipelined processor core, systematic re-execution decrements the number of branch delay slots and halves the number of instructions that can be inserted in the remaining branch delay slots. This also means that the branch delay slots can be used for on-line monitoring besides performance improvement. Systematic re-execution of instructions in degraded operating mode can be used during software-based testing [18] to increase the fault observability and, implicitly, the fault coverage.

In order to allow proper recovery in case of detected errors, the replicated operations must release only verified values during non-reversible exterior accesses when data is written or updated. The results produced by the first instance of a replicated operation can only be used inside the monitored pipeline or for reversible exterior accesses, like read accesses to memory or the register file of a processor core. If recovery schemes based on micro-rollback are used to distinguish between transient and delay faults [31], the re-execution must not start from the failing operation but from the preceding operation which provides the initialization test pattern for a potential delay fault.

The management of the operation replication can be done with the help of a status bit associated to each pipeline buffer. An alternating encoding scheme as illustrated in Fig. 8 can protect the status bits against delay and transient faults [27]. In Fig. 8, only the buffer that drives an instance of a replicated operation except for the last instance, e.g. O'$_3$, has a status bit with the same value as the status bit as the previous buffer. Table III shows how the status bits can be used to indicate whether an operation is replicated and which version of a replicated operation is executed. This encoding works also when the replication of operations goes beyond duplication, which might be helpful for certain low power policies [14].

The encoding in Table III allows to (a) indicate which operation must produce identical results as the preceding operation, (b) select the results delivered at the pipeline outputs or forwarded to other operations in the pipeline and (c) generate the 'T' and 'nT' signals which control the DCSs applied to the pipeline flip-flops. In case of the operation executed by the first pipeline stage the *(n-1)th* status bit need to be replaced by the *valid* signal in Fig. 6 and Fig. 8. The control signals of the DCSs applied to the output buffer of the first pipeline stage depends only on the *valid* signal that indicates the occurrence of a stall cycle. The control signals 'T' and 'nT' must switch fast when a buffer goes from a normal to a degraded mode. For the opposite transition, an additional cycle is available which corresponds to the execution time of the second instance of a replicated operation. The proper timing of the 'T' and 'nT' signals can be verified with the stability checker introduced in [2]. Only one such detector per pipeline stage is sufficient.

In the case of pipelined interconnection systems where the propagation time of status bits from one buffer to the next one is comparable to the latency of a pipeline stage, the status bits in Fig. 8 can be assigned such that they arrive with one clock cycle earlier. This means that they are shifted with one position to the right. The status bits are not necessary in pipelined interconnection systems where the communicating components have a scheduling table that indicates when a replicated message slice should arrive or leave. Status bits are also not needed in bus-based interconnection systems where an existing central arbiter [3] can take the responsibility of message slice replication and degraded mode management.

A stall cycle management scheme based on alternating status bits was applied to an in-house processor core used for control applications. This core is an in-order RISC processor with 4 pipeline stages: fetch/decode, execute, memory and write-back. The stall cycles generated by the following events are considered for the re-execution of instructions:

- Fetch-misses induce a stall cycle in the first pipeline stage,
- *Nop* (no operation) instructions and taken branches generate a stall cycle in the second pipeline stage.

Benchmark programs from the MiBench suite [11] were simulated and each pair of consecutive instructions followed by a stall cycle has been transformed into pairs of test patterns for a collapsed version of the processor pipeline. Table IV reports the coverage of the transition delay faults [35] obtained

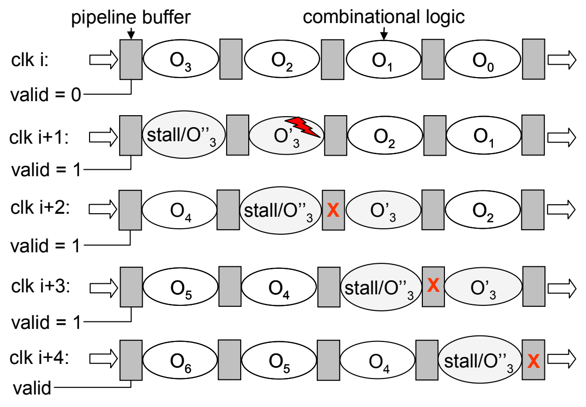

Fig. 6. Utilization of stall cycles for operation re-execution.

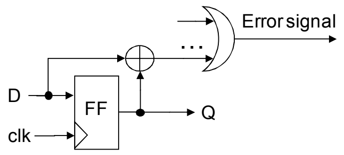

Fig. 7. Comparison of flip-flop inputs and outputs [23].

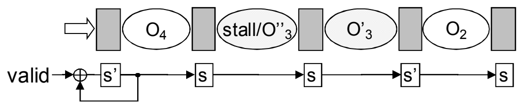

Fig. 8. Pipeline status bits (s∈ {0,1} and s' = not s).

TABLE III
UTILIZATION OF THE PIPELINE STATUS BITS (s∈ {0,1})

(n-1)th status bit	nth status bit	(n+1)th status bit	Type of operation in the nth pipeline stage
s' (*valid*=1)	s	s'	Non-replicated operation
s (*valid*=0)	s	s'	First version of a replicated operation
s (*valid*=0)	s	s	Intermediate version of a replicated operation
s' (*valid*=1)	s	s	Last version of a replicated operation

with Synopsys TetraMAX [38] for several fetch-miss frequencies. These fault coverage (FC) figures are relative to the FC achieved with a systematic re-execution of each instruction (100%) and an instruction result verification performed at the end of each pipeline stage. For each benchmark, the first line corresponds to a verification of the instruction result at the end of each pipeline stage (*stage*) while the second line reports the FC when the instruction results are observed only at the pipeline outputs (*pipeline*). It can be seen that the verification at the end of each pipeline stage can improve the FC by roughly 20%. Independently of the verification scheme, a fetch-miss frequency of 10% reduces the FC by only 10%. At lower fetch-miss frequencies, the influence on the FC is less pronounced for the *BasicMath* benchmark due to its higher instruction diversity.

An RTL description of the mentioned processor enhanced with a monitoring scheme that relies on the utilization of stall cycles was synthesized with Synopsys Design Compiler [38] and the 45nm TSMC standard cell library. The proposed monitoring scheme had no visible impact on the processor latency. For a wide range of target clock frequencies, an average hardware overhead of about 12% has been obtained. The hardware overhead is reduced to 5% when the verification of the instruction results is performed only at the pipeline outputs. The hardware overhead was exaggerated by the small size of the target processor which is below 10K (NAND2-equivalent) gates.

TABLE IV
DELAY-FAULT COVERAGE VS. MISS-FETCH FREQUENCY

Benchmark	Result checking	Fetch-miss frequency						
		100%	20%	10%	5%	2%	1%	0
StringSearch	/stage	100	93	90	74	72	67	51
	/pipeline	79	72	68	54	52	47	35
BasicMath	/stage	100	93	89	85	83	77	66
	/pipeline	81	74	70	66	63	59	50

VI. CONCLUSIONS

A new approach to degraded mode testing was presented that provides high flexibility for the selection of the degraded zones and low latency switching between degraded and normal operating modes. Delay control structures (DCSs) were introduced which enable to modify the latency of the monitored circuit with a quantifiable amount. A DCS contains only two or three transistors and a single DCS is sufficient for each flip-flop in the monitored circuit. A micro-architectural support for the concurrent self-test of in-order pipelined logic was proposed as well. This solution relies on the replication of the last valid operation executed before a stall cycle. Error prediction can be ensured if the first instance of each replicated operation is executed in degraded mode (with the DCS activated). Systematic re-execution of operations can be used to handle transient and delay faults. An alternating encoding scheme was proposed to protect the status bits associated to the pipeline buffers for the management of redundant executions. The application of this monitoring scheme to an in-order RISC processor required a small hardware overhead without any impact on performance. The simulation of a few benchmark programs showed that important fractions of transition delay faults can be tested concurrently with the program execution.

ACKNOWLEDGEMENTS

This work was carried out as part of the Catrene project "TOETS" [CT302] and was supported by the *French Ministry of the Economy, Industry and Employment*.

REFERENCES

[1] W. Abadeer, W. Ellis "Behavior of NBTI under AC dynamic circuit conditions," *Reliability Physics Symposium*, 2003.
[2] M. Agarwal, B. Paul, M. Zhang, S. Mitra "Circuit failure prediction and its Application to transistor aging," *VTS*, pp. 277-286, 2007.
[3] "AMBA™ Specification (Rev 2.0)," *ARM*.
[4] D. Blaauw et al. "Razor II: in situ error detection and correction for PVT and SER tolerance," *ISSCC,* pp. 400-401, 2008.
[5] R. Baumann "Soft errors in advanced computer systems" *IEEE Design and Test of Computers*, pp. 258-266, 2005.
[6] S. Borkar "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *Micro* 25, No. 6, pp. 10–16, 2005.
[7] K.A. Bowman et al. "Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance," *IEEE J. Solid-State Circuits*, pp. 49-63, Jan. 2009.
[8] J.T.-Y. Chang, E.J. McCluskey "Detecting delay flaws by very-low-voltage testing," *ITC*, pp. 367-376, 1996.
[9] T.W. Chen et al. "Gate-oxide early life failure prediction," *VTS*, 2008.
[10] C. Constantinescu "Trends and challenges in VLSI circuit reliability," *Micro* 23, Issue 4, pp. 14–19, July 2003.
[11] M.R. Guthaus et al. "MiBench: a free commercially representative embedded benchmark suite," *IEEE Workshop Workload Characterization*, pp. 3-14, 2001.
[12] I. Hiroaki, Y. Li, S. Mitra "VAST: Virtualization Assisted Concurrent Autonomous Self-Test, " *IEEE Intl. Test Conf.*, 2008.
[13] ITRS, available at *http://public/itrs.net/*
[14] H. M. Jacobson. "Improved clock-gating through transparent pipelining," *International Symposium of Low Power Electronics and Design*, 2004.
[15] T. Karnik et al. "Characterization of soft errors caused by single event upsets in CMOS processes," *IEEE Transactions on Dependable and Secure Computing*, Vol. 1, No. 2, April-June 2004, pp. 128-143.
[16] O. Khan, S. Kundu "A self-adaptive system architecture to address transistor aging," *DATE*, pp. 81-86, 2009.
[17] T. Kim et al. "Silicon Odometer: An On-chip Reliability Monitor for Measuring Frequency Degradation of Digital Circuits," *IEEE J. Solid-State Circuits*, pp. 874-880, Apr. 2008.
[18] N. Kranitis et al. "Optimal periodic testing of intermittent faults in embedded pipelined processor applications," *DATE,* pp. 65-70, 2006.
[19] M. Meterelliyoz, H. Mahmoodi, K. Roy "A leakage control system for thermal stability during burn-in test," *ITC*, 2005.
[20] S. Mitra et al. "Robust system design with built-in soft-error resilience," *Computer, Special Issue*, 38(2), pp. 43–52, 2005.
[21] M. Mota et al. "A four channel, self-calibrating, high resolution TDC," *IEEE International Conf. on Electronics, Circuits and Systems*, 1998.
[22] M. Nicolaidis "Time redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies," *VLSI Test Symposium (VTS)*, 1999.
[23] M. Nicolaidis "GRAAL: a new fault tolerant design paradigm for mitigating the flaws of deep nanometric technologies," *ITC*, pp. 1–10, 2007.
[24] M. Mueller et al. "RAS strategy for IBM S/390 G5 and G6," *IBM J. RES. DEVELOP*. Vol. 43 No. 5/6, 1999, pp. 875-888.
[25] J. Oliver et al. "Credit-based dynamic reliability management using online wearout detection," *Computing Frontiers*, pp. 139-148, 2008.
[26] M.K. Qureshi et al. "Microarchitecture-based introspection: A technique for transient-fault tolerance in microprocessors," *DSN*, 2005.
[27] D.A. Reynolds, G. Metze "Fault detection capabilities of alternating logic," *IEEE Trans. on Computers*, 27(12):1093–1098, December 1978.
[28] S. Shyam et al. "Ultra low-cost defect protection for microprocessor pipelines," *Architectural Support for Programming Languages and Operating Systems*, pp. 73-82, 2006.
[29] J.C. Smolens et al. "Detecting emerging wearout faults**,"** *Workshop on Silicon Errors in Logic - System Effects*, 2007.
[30] J. Srinivasan, S.V. Adve, P. Bose, J.A. Rivers "Lifetime reliability: toward an architectural solution," *Micro* 25(3), pp. 70-80, 2005.
[31] Y. Tamir et al. "High-Performance fault-tolerant VLSI systems using micro rollback," *Transactions on Computers* 39(4), pp. 548-554, 1990.
[32] Tandem Computers Incorporated "NonStop Himalaya range: K200, K2000, and K20000 servers," *NonStop servers product description*, 1995.
[33] J. Tschanz, et al., IEEE Symp. VLSI Circuits, pp. 112-113, 2009.
[34] A.K. Uht "Uniprocessor performance enhancement through adaptive clock frequency control," *SSGRR, 2003.*
[35] J.A. Waicukauski et al. "Transition Fault Simulation," *IEEE Design and Test of Computers*, Vol. 4, 1987, pp. 32–38.
[36] S. Zafar et al. "A model for negative bias temperature instability (NBTI) in oxide and high K PFETs," *VLSI Technology and Circuits*, 2004.
[37] *http://ptm.asu.edu/*
[38] *http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Pages/*