A Pseudo-Dynamic Comparator for Error Detection in Fault Tolerant Architectures

D. A. Tran, A. Virazel, A. Bosio, L. Dilillo,
P. Girard, A. Todri
LIRMM – University of Montpellier / CNRS
Montpellier, France
{tran, virazel, bosio, dilillo, girard, todri}@lirmm.fr

M. E. Imhof, H.-J. Wunderlich
Institute of Computer Architecture and Computer Engineering
University of Stuttgart, Germany
{imhof, wu}@iti.uni-stuttgart.de

***Abstract*—Although CMOS technology scaling offers many advantages, it suffers from robustness problem caused by hard, soft and timing errors. The robustness of future CMOS technology nodes must be improved and the use of fault tolerant architectures is probably the most viable solution. In this context, Duplication/Comparison scheme is widely used for error detection. Traditionally, this scheme uses a static comparator structure that detects hard error. However, it is not effective for soft and timing errors detection due to the possible masking of glitches by the comparator itself. To solve this problem, we propose a pseudo-dynamic comparator architecture that combines a dynamic CMOS transition detector and a static comparator. Experimental results show that the proposed comparator detects not only hard errors but also small glitches related to soft and timing errors. Moreover, its dynamic characteristics allow reducing the power consumption while keeping an equivalent silicon area compared to a static comparator. This study is the first step towards a full fault tolerant approach targeting robustness improvement of CMOS logic circuits.**

Keywords-robustness; soft error; timing error; fault tolerance; duplication; comparison; power consumption.

I. INTRODUCTION

Digital systems transitioned from specialized application areas to ubiquitous mass products. The robustness of electronic systems is one of the upcoming key requirements in many of these application areas, including safety critical applications and mass products [1]. While scaling supports the need for competitive mass products it also influences other functional and non-functional properties such as reliability, safety and robustness. Achieving and maintaining those properties is getting more critical with every introduced technology nodes as treating them on the physical level by adjusting process parameters during manufacturing is no longer feasible. Additional challenges are posed by feature sizes entering an area where simple inverters get susceptible to soft errors and single event effects.

Fault tolerant architectures are essential to reach the required robustness goals of future CMOS circuits and systems. These architectures are commonly used to tolerate on-line faults, i.e. faults that appear during normal system operations, irrespective of their transient of permanent nature [2]. However, it has been shown in [3, 4, 5] that these architectures could also tolerate permanent defects and hence, improve the manufacturing yield.

Fault tolerant techniques often rely on one of two basic concepts: Error Masking or Detection/Correction. While the first method masks possible error without giving any information about their occurrence, the second one employs a comparator to validate a circuit response by matching the outputs of redundant circuit copies (Duplication/Comparison) or to prove the absence of errors by validating a checksum against a golden reference [6,7].

The main idea of Duplication/Comparison consists of duplicating the logic circuit and adding a comparator to the circuit outputs. If an error affects one of the logic circuits, the comparator might indicate this problem depending on the error type. A hard error in one of the two circuits is always detected. Soft errors are often not detected due to two main reasons: 1) Glitches caused by soft and timing errors may be masked within the comparator, and 2) the observability of soft and timing errors is timing dependent, due to their transient nature.

Self-checking comparators and two-rail checkers are well known for their capability to detect faults present at inputs and internal nodes [16, 17, 18]. However, most of them target Stuck-at-fault only. In [16], authors claimed the detection of soft errors at comparator inputs, but only for large glitches that will not be filtered by logic gates.

Consequently, the comparator structure should integrate the ability to detect any transitions caused by soft and timing errors. A first approach utilizing a dedicated transition detector to capture these errors at the registers has been previous discussed in RAZOR II [8]. The approach can also be extended to detect errors at logic circuits by adding the transition detector to each circuit output, thereby introducing a high hardware overhead. In [9], the authors proposed to combine a one-bit parity compaction with a transition detector on its output. This solution is less costly in term of area overhead but only offers the detection of soft errors. Moreover, the problem of masked glitches at the compactor remains untreated.

The main idea of this work is to combine a dynamic transition detector and a static comparator in order to detect all aforementioned errors (hard, soft and timing) during a user-defined detection window. The static part of the comparator architecture attains the detection of hard errors while the dynamic part obtains the detection of soft and timing errors. Moreover, this "pseudo-dynamic" comparator consumes less power in comparison to a static comparator with an equivalent silicon area. Finally, compared to the "full dynamic" comparator proposed in [14], our architecture does

978-1-4673-1074-1/12/$31.00 ©2012 IEEE

not suffer from logic level problem caused by PMOS transistors implemented in the pull-down network [15].

The presented pseudo-dynamic comparator incorporates the following properties:

- **Dynamic behavior:** It is able to detect and flag glitches during a comparison window, making it applicable for the detection of timing and soft errors.
- **Increased sensitivity:** Compared to a static comparator smaller glitches are detected.
- **Area:** The overhead is comparable to a static comparator.
- **Power:** The power consumption is reduced as nonessential transitions outside the comparison window are filtered.

The remainder of this paper is organized as follows. Section II provides the fundamentals of the Duplication/Comparison scheme. Section III presents the pseudo-dynamic comparator architecture that embeds the transition detection. Experimental results in terms of detection capabilities, area overhead and power consumption are discussed in Section IV.

II. FUNDAMENTALS OF DUPLICATION/COMPARISON

Duplication/Comparison is an error detection method widely used in fault tolerant architectures. It consists in duplicating the Logic Circuit (LC). The two LCs can be identical or different but realizing the same logic functions. The same input vector, *Input* feeds both LCs. Their output vectors, *A* and *B* are compared to determine the presence of any errors. In a complete fault tolerant architecture, the comparison signal *Comp* is used to activate the error correction flow. Figure 1 illustrates the basic scheme of the Duplication/Comparison scheme.

The comparator performs two stages: local comparison and global comparison. The local comparison verifies each LC's output pairs by using XOR functions. The global comparison provides an accumulation of all local comparisons with the help of an OR/AND tree. This stage is necessary when a global comparison result is required by next stages, such as in Detection/Correction scheme where a single output of the Detection stage triggers the correction mechanism at errors occurrences.

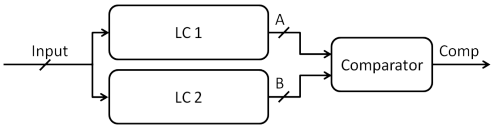

Figure 1. Duplication/Comparison principle.

The difference comparator (d_comparator, Figure 2) is mostly used in a fault tolerance context to verify that the two output vectors are different. The first logic level consists of XOR gates implementing the local comparison while the accumulation into a single bit signal is achieved by an OR-tree.

The stability of the comparator output *Comp* depends on many parameters. Since the two LCs are not physically equivalent (process variations), *Comp* is not stable during the LC computation time (max propagation delay t_{LC} of LCs). In addition, the comparator itself has a propagation delay t_{COMP} during which glitches may occur. Consequently, *Comp* starts at a stable state, result of the previous comparison, to reach another stable state, result of the new comparison. In between, *Comp* is unstable. The use of the *Comp* signal must be strictly restricted to its stable state. Consequently, *Comp* must be used only during a stable period called "comparison window", after $t_{LC} + t_{COMP}$ and before the next computation.

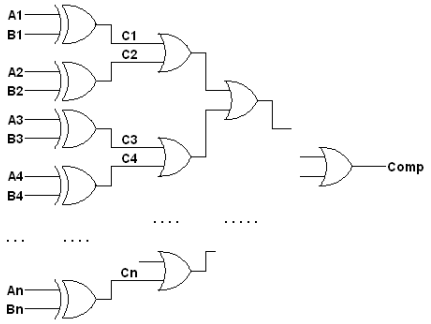

Figure 2. Static d_comparator structure.

In a fault-free context, *Comp* is constantly at logic-0 during the comparison window. If a hard error occurs, the problem is detectable since *Comp* will change to a logic-1. For soft and timing errors, the detection may be effective since glitches are observable at *Comp* signal during the comparison window. However, small glitches can be masked by the comparator itself, which makes the detection impossible. Fault-free and faulty (hard, soft and timing errors) cases are highlighted in Figure 3. Consequently, the comparator structure must be modified in order to detect any transitions caused by soft and timing errors.

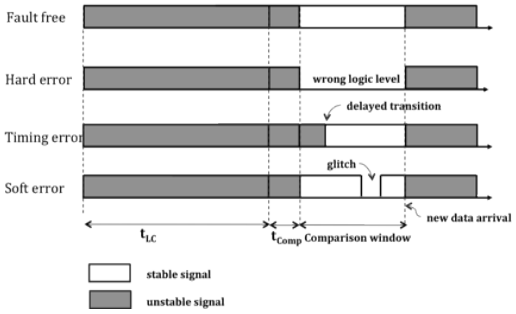

Figure 3. Comparison signal waveforms.

III. THE PSEUDO-DYNAMIC COMPARATOR ARCHITECTURE

The objective of this work is to replace the static comparator in the Duplication/Comparison paradigm in order to detect any transition happening within the comparison window.

The presented pseudo-dynamic comparator checks the two input vectors during a user-defined time window. In the fault free case and in presence of hard errors both input vectors are stable throughout the comparison window. In these cases the pseudo-dynamic comparator provides the same function as a static comparator: It returns a logic-0 if two input vectors are equal and a logic-1 otherwise. In

presence of timing or soft errors the behavior of the pseudo-dynamic comparator differs. While the static comparator propagates transitions and glitches from its inputs to its output, the pseudo-dynamic comparator will generate exactly one transition from logic-0 to logic-1 indicating that a difference occurred. The output stays stable at logic-1 until the pseudo-dynamic comparator is reset at the end of the comparison window. In addition, the pseudo-dynamic comparator filters all transitions happening before or after the comparison window, thereby masking all transitions happening during t_{LC}.

Different possibilities exist in order to implement a pseudo-dynamic comparator with the aforementioned characteristics:

- A dynamic transition detector, as described in [8], could be added to the output of a traditional static comparator. While improving the soft error tolerance capability drawbacks are expected with respect to the other properties. As the detector is added at the end of the accumulation phase glitches are still masked within the previous logic levels. In addition all transitions, whether within the comparison window or not, propagate throughout the complete structure and lead to a power consumption comparable to a static comparator.
- Adding transition detectors at LC outputs eliminates the masking of glitches within the comparator but results in an increased area overhead.

The pseudo-dynamic comparator presented here implements the dynamic behavior by combining the transition detection and the accumulation stage within the comparator. The first logic level of the accumulation stage is replaced by dynamic CMOS logic resulting in all of the above-mentioned properties.

Section III.A presents a dynamic OR (DOR) gate combining the static OR function needed in the accumulation stage with the ability to detect and flag transitions. Section III.B depicts the complete architecture of the pseudo-dynamic comparator and explains its advantages over a traditional static comparator.

A. Dynamic OR gate architecture

Figure 4 shows the schematic of a dynamic OR gate with 4 logic inputs. It consists of 9 transistors and is controlled by a detection clock signal *DC* and a *reset* signal.

When the reset signal is at logic-0, node *N* is pre-charged to VDD and thus node *Z* is kept at logic-0. During the evaluation phase, both *reset* signal and digital clock (*DC*) signal are at logic-1. T1 is off while T2 and T3 are on. During this phase, if all signals *C1*, *C2*, *C3* and *C4* are at logic-0 then T4, T5, T6 and T7 are all off. Thus, no discharge current path exists and node *N* is kept at logic-1 while node *Z* is at logic-0. If at least one of the four logic inputs, *C1* for example, turns to logic-1 during the evaluation phase, a current path will be formed and node *N* will be pulled down which makes node *Z* to switch to logic-1. Note that, once node *N* is discharged, it will remain at logic-0 until the next time we reset the DOR gate with the reset signal.

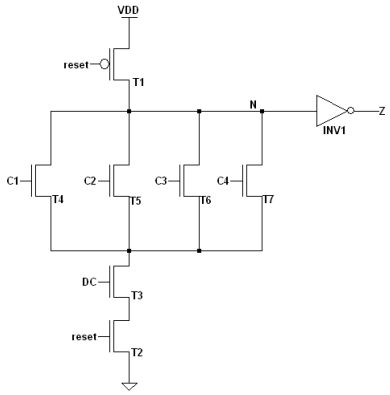

Figure 4. 4-input DOR gate (DOR4)

During the evaluation phase, dynamic CMOS logic suffers from leakage currents. Even if the pull-down network is off, these currents will discharge node *N*, which causes a false value at output *Z*. Therefore, we need a "keeper" to maintain *N* at logic-1 when the pull-down network is off. Moreover, this keeper must be weak enough so that when the NMOS logic is on, it can pull-down node *N* to logic-0. In [8], the authors proposed a "weak keeper" which is formed by a two inverters loop as shown in Figure 5a. However, in order to reduce the area overhead of the DOR, we decided to use a feedback transistor as presented in [10]. This structure is depicted in Figure 5b.

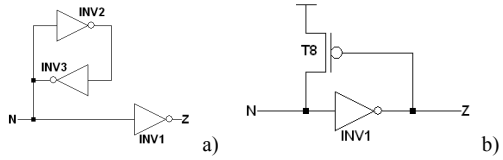

Figure 5. Keeper structures

B. Dynamic comparator architecture

Figure 6 shows a complete architecture of our pseudo-dynamic comparator. Similarly to a traditional static comparator as shown in Figure 2, the pseudo-dynamic comparator is composed of two stages: comparison and accumulation. The comparison stage consists of static 2-input XOR gates while the accumulation stage is modified from the one of the static comparator. Static OR gates, in the first layer of the static OR-tree, are replaced by DOR gates. The rest of the static OR-tree is left intact. Compared to the static comparator, the pseudo dynamic comparator has two more inputs, which control *reset* and *DC* inputs of the DOR gates.

As mentioned in previous sub-section, the output of the DOR gate is stable at logic-0 due to the keeper till the evaluation phase, which is controlled by the *DC* signal. Thus, *Comp* is constant logic-0. During the evaluation phase, if the two vectors *A* and *B* are stable and identical (fault-free case) then all signals *Ci* (i = 1..n) are also at logic-0 which maintain DOR outputs unchanged. Therefore, *Comp* will be constant logic-0, which means there is no error.

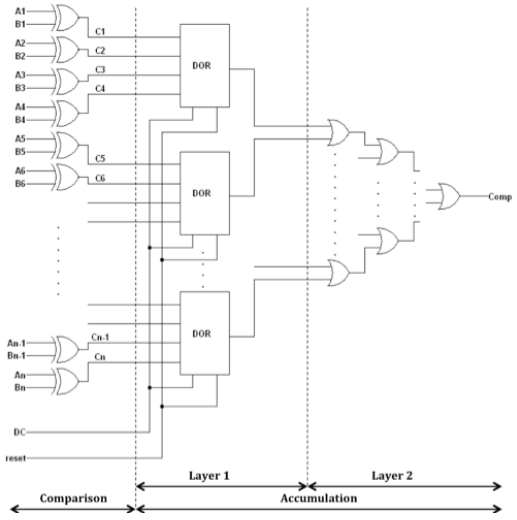

Figure 6. Pseudo dynamic comparator architecture.

Otherwise, if the two vectors *A* and *B* are not identical, there will be a constant logic-1 (presence of hard error) or glitches (presence of soft or timing error) at one of the *Ci* signals. At least one of the DOR outputs will then turn to logic-1, which makes *Comp* signal switch to logic-1 until the next *reset* signal is applied.

The structure of the pseudo-dynamic architecture is presented in Figure 6. Accumulation stage is active only during the comparison window. Moreover, in fault free conditions, which happen most of the time, DOR's gate outputs are at constant logic-0, which means that the Layer 2 of the Accumulation phase does not consume dynamic power. Therefore, beside the detection capability of soft and timing errors, the pseudo-dynamic comparator also provides power consumption reduction.

Finally, the advantage of having a comparison window is that the comparison signal *Comp* is at stable logic-0. Only in case of error, it will switch to logic-1 and remain at this level until the *reset* signal is applied. These characteristics make the interpretation of this signal easier when used in a complete Detection/Correction fault tolerant scheme.

IV. Experimental results

In order to evaluate our pseudo-dynamic comparator, we implemented it using the Nangate Open Cell Library (OCL, [11]), which contains standard cells for a 45nm technology specified by the Predictive Technology Model (PTM, [12]). In the following sub-sections we present several SPICE simulations to highlight the added functionality to the pseudo-dynamic comparator in contrast to the traditional static one. In order to prove the architecture concept, only typical conditions are taken into account at this first step of the work. Process variations will be studied in further papers. Then, the layout implementation is discussed.

A. Sensitivity

1) Glitches detection capability of DOR gate

In this sub-section, we verify the detection capability of the DOR gates with respect to input glitches during the evaluation phase. The DOR used in our simulations is represented in SPICE as the schematic shown in Figure 4 along with the "keeper" shown in Figure 5b.

In our simulations, the DOR is reset (logic-0 on *reset* input) at t_0=50ps and t_4=400ps while the evaluation phase (logic-1 on *DC* input) is set between t_1=100ps and t_3=350ps. The evaluation window is around 250ps. As the four inputs *C1*, *C2*, *C3* and *C4* of the DOR are symmetric, we only apply glitches at *C1* while *C2*, *C3* and *C4* are kept at logic-0. The glitches are applied at t_2=200ps (during the evaluation window). We simulate the output *Z* of the DOR gate in two cases based on the duration of the glitch (high level) as: 1) Δ1=40ps for a large and 2) Δ2=15ps for a small glitch. Both glitches have the rising and falling time of about 1ps. Simulation results are shown in Figure 7.

In Figure 7, waveforms of inputs, *reset* and *DC* are shown respectively as signals V(reset) and V(dc). Plot V(c1) presents the glitch at input *C1*. Output *Z* of the DOR gate is shown as V(z). We can observe that during the evaluation phase, when *C1* is at logic-0 between t_1 and t_2, *Z* is also at logic-0. When a large glitch appears at t_2, *Z* turns to logic-1 level, which means that the glitch was detected. Note that *Z* remains at logic-1 even when the large glitch has disappeared at time t_3. It only returns to low level when the DOR is reset at t_4.

In the second case shown in Figure 7, as the glitch is too small the output *Z* did not have time to completely turn to logic-1. When the small glitch disappears, *Z* returns to logic-0 because of the "keeper". In this simulation, the small glitch was not detected. These results clearly demonstrate that we cannot detect glitches of size smaller than the commutation time of the dynamic gate.

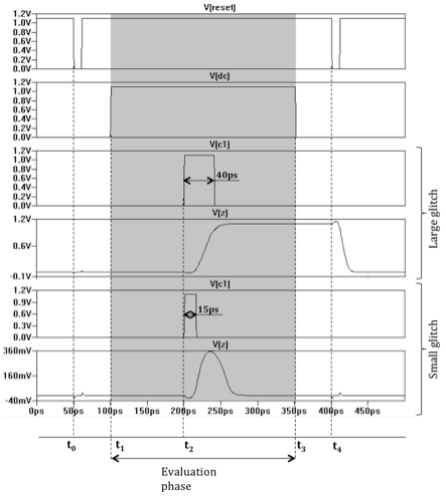

Figure 7. Glitches detection capability of DOR gate.

2) Static versus Pseudo-dynamic comparator

In this sub-section, we compare the glitches detection capability between a static comparator and a pseudo-dynamic comparator able to compare two 4-bit input vectors. The static comparator is implanted in SPICE using four 2-input XOR gates in the comparison stage and a 4-input OR gate in the accumulation stage. In the pseudo-dynamic

comparator, we replace the 4-input OR gate with the 4-inputs DOR gate presented in the last sub-section.

In the simulations, the pseudo-dynamic comparator is reset (logic-0 of *reset* input) at t_0=50ps and t_4=400ps while its comparison phase (logic-1 of *DC* input) is set between t_1=100ps and t_3=350ps. Both comparators are used to compare two input vectors *A[3:0]* and *B[3:0]*. Two input pairs (*A3*, *B3*) and (*A2*, *B2*) are kept at logic-0 while the pair (*A0*, *B0*) is kept at logic-1.

Figure 8 shows simulation results when we keep (*A1*, *B1*) at logic-0 and apply a glitch at *A1* at time t_2=200ps during Δ=70ps. Glitch rising and falling time is about 1ps. Waveforms of the inputs *reset* and *DC* of the pseudo-dynamic comparator are shown respectively as V(reset) and V(dc). Plot V(a1) and V(b1) presents the input signals applied at input pair (*A1*, *B1*) of both comparators. The output *Comp* of the static comparator and the one of the dynamic comparator are respectively shown in V(comp_s) and V(comp_pd).

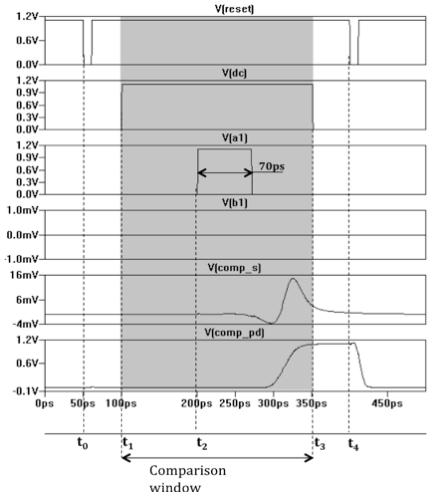

Figure 8. Static vs. Pseudo-dynamic comparator

Waveforms in Figure 8 show that the pseudo-dynamic comparator was able to detect the glitch of Δ=70ps while the static comparator filtered it. By making the glitch wider, we found that the pseudo-dynamic comparator could detect glitches of 64ps while the static comparator could only detect 86ps or larger glitches. Moreover, due to their un-symmetric internal structure, both comparators have glitch detection capability that depends on the glitch form. Simulations with *B1* and *A1* kept at logic-1 and glitch switching from logic-1 to logic-0 added to A1, reveal that the pseudo-dynamic comparator could detect glitches of 57ps wide. In the same conditions, the static comparator could only detect glitches larger than 80ps.

In Figure 8, we can also see that the comparison signal of the pseudo-dynamic comparator starts switching to logic-1 at t=300ps while the glitch appeared at t_2=200ps. In fact, this high delay is due to the additional delay added by the XOR gates of the comparator. Moreover, small glitches are also filtered out by these gates. In order to reduce the delay as well as to improve the sensitivity of DORs gates, we can make the Comparison stage (Figure 6) dynamic instead of the Accumulation stage. This can be done by using pseudo-dynamic XOR gates.

B. Area Overhead

In order to evaluate the feasibility of the proposed scheme, the 4-input dynamic OR (DOR4) was designed as a standard cell using a full custom design style. The Open Cell Library (OCL) is now used to compare the area of the DOR4 standard cell and a static 4-input OR. It contains standard cells for a 45nm technology specified by the Predictive Technology Model. The 4-input OR gate OR4 X1 of the OCL has a cell height of 1.4 µ m and a cell width of 1.14µm. Its area of 1.596µm2 is used as the baseline in the following.

Figure 9 shows the layout of the dynamic OR standard cell. The transistors from Figure 4 are placed as follows: The small N-well in the upper left corner contains the pull-up transistor of the inverter, T1 as well as the feedback transistor T8. The Pwell at the bottom holds the pull-down transistor of the inverter together with the transistors T2 and T3. The right hand side implements the NMOS transistors T4-T7 used for the inputs.

The DOR4 standard cell was designed according to the design rules and electrical rules of the FreePDK process design kit [13]. The cell height is as in the OCL 1.4µm. Together with a width of 1.14µm the overall area results in 1.596µm2. The overhead to implement the proposed comparator at gate level consists of the DOR4 and the detection clock generator (DC). While the detection clock generation can be shared among all dynamic OR gates and synthesized using standard clock tree synthesis algorithms and tools more attention has to be paid for then dynamic part (Layer 1) of the comparator architecture. Compared to the OCL OR4 X1 the dynamic part of the comparator can be implemented without any area penalty while the static part of the comparator remains unchanged and is implemented using OCL standard cell ORs. The area of the detection clock generator is compared to the comparator considerable small.

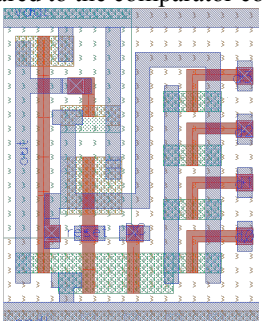

Figure 9. Layout of 4-input dynamic OR gate.

C. Power consumption

To determine and compare the power consumption of the comparators a case study for b05 from the ITC'99 benchmarks was carried out. According to Figure 1 two combinational copies of b05 were synthesized for the OCL. To account for a realistic timing variation at the comparators input signals LC1 was restricted to the use of primitive two input gates while LC2 used complex gates with more inputs.

This base circuit b is used as a baseline in the following. The base circuit is extended with a static comparator, the resulting circuit is called static (s). Attaching the proposed pseudo-dynamic comparator to LC1 and LC2 results in the dynamic (d) circuit.

All three circuits are then fed by a test bench applying a fully specified test set with 120 patterns in total, using 200ns per pattern. For the dynamic circuit the reset signal was pulled to logic-0 at 1ns for a duration of 1ns, the comparison window lasted from 5ns to 195ns. For all three circuits b, s and d, the following steps were performed:

- Compile circuit b, s, d to verilog, annotate gate delays.
- Read circuit and delay annotations into simulator.
- Simulate circuit and test bench, record switching activity.
- Perform cycle accurate timing analysis taking into account the circuit, delay annotations and switching activity.

TABLE I. AVERAGE POWER CONSUMPTION

Circuit	Average power (W)
base	1.163×10^{-05}
static	1.281×10^{-05}
dynamic	1.247×10^{-05}

Table I shows the calculated average power for all three circuits. The power consumption of the static comparator is: $P(static) = P(s) - P(b) = 1.18 \times 10{-}06$ W. The average power for the presented pseudo-dynamic comparator is significantly lower: $P(dynamic) = P(d) - P(b) = 0.84 \times 10{-}06$ W . Compared to the static comparator its power consumption amounts to only 71.186% (= $P(dynamic)/P(static)$). The reduction is achieved by strictly examining only transitions within the comparison window and ensuring that all differences are immediately stored due to the dynamic behavior. Moreover, in a fault free case, outputs of all DOR gates are stable at logic-0 which means that the OR-tree (Layer 2 of the Accumulation stage) does not consume dynamic power.

V. CONCLUSION

A new pseudo-dynamic comparator architecture has been presented that targets the error detection in a Duplication/Comparison context. This architecture combines a traditional comparator using static CMOS gates with a dynamic CMOS transition detector. While behaving like a static comparator in presence of hard errors the dynamic behavior enables the identification of timing and soft errors.

The performed analog simulations prove an increased sensitivity to glitches during a user-defined comparison window, thereby significantly reducing the internal switching activity while focusing the improved detection capabilities to the relevant period in time. Implementing the dynamic OR (DOR) as a new standard cell in a 45nm technology verifies the usability of the pseudo-dynamic comparator without area penalties. The preformed power simulations show a power reduction by nearly 30% compared a static comparator. In addition the presented comparator perfectly supports any fault tolerance scheme by eliminating the necessity to explicitly latch the comparator output.

VI. ACKNOWLEDMENT

This work was supported by the DFG project Realtest "Test and Reliability of Nano-Electronic Systems" (Wu245/5-2).

REFERENCES

[1] Semiconductor Industry Association (SIA), "International Technology Roadmap for Semiconductors (ITRS)", 2010.

[2] I. Koren and C. Krishna, "Fault Tolerant Systems", Morgan Kauffman Publisher, 2007.

[3] L. Fang and M. S. Hsiao, "Bilateral Testing of Nano-scale Fault-tolerant Circuits", in Proc. of IEEE Int. Symp. on Defect and Fault-Tolerance in VLSI Systems, pp. 309-317, 2006.

[4] J. Vial, A. Bosio, P. Girard, C. Landrault, S. Pravossoudovitch and A. Virazel, "Using TMR Architectures for Yield Improvement", in Proc. of Int. Symp. on Defect and Fault Tolerance in VLSI Systems, pp. 7-15, 2008.

[5] J. Vial, A. Virazel, A. Bosio, P. Girard, C. Landrault and S. Pravossoudovitch, "Is TMR Suitable for Yield Improvement?", IET Computers and Digital Techniques, vol. 3, No 6, pp. 581-592, November 2009.

[6] M. E. Imhof, H.-J. Wunderlich and C. G. Zoellin, "Integrating Scan Design and Soft Error Correction in Low-Power Applications", in Proc. of IEEE Int. On-Line Testing Symposium, pp.59-64, 2008.

[7] M. E. Imhof, H.-J. Wunderlich, "Soft error correction in embedded storage elements," in Proc. of IEEE Int. On-Line Testing Symposium, pp. 169-174, 2011.

[8] D. Blaauw et al. , "Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance", IEEE ISSCC, pp. 400-401, 2008.

[9] D. J. Palframan, N. S. Kim and M. H. Lipasti, "Time Redundant Parity for Low-Cost Transient Error Detection", in Proc. of IEEE Design, Automation & Test in Europe, pp. 1-6, March 2011.

[10] R.J. Baker, "CMOS Circuit Design- Layout and Simulation", IEEE Series on Microelectronic Systems, 3rd Edition.

[11] Nangate. 45nm Open Cell Library v1.3. http://www.nangate.com, 2009.

[12] W. Zhao and Y. Cao, "Predictive technology model for nano-CMOS design exploration", ACM Journal on Emerging Technologies in Computing Systems , Vol. 3, No. 1, 2007.

[13] J. Stine et al., "FreePDK: An Open-Source Variation-Aware Design Kit", IEEE Int. Conf. on Microelectronic Systems Education, pp. 173-174, 2007.

[14] C. C. Wang et al., "High Fan-in Dynamic CMOS Comparators with Low Transistor Count", IEEE Trans. on Circuits and Systems: Fundamental Theory and Applications, Vol. 50, No. 9, pp. 1216-1220, 2003.

[15] C. -Y. Kim and L. -S. Kim, "Low-power and high-performance equality comparator using pseudo-NMOS NAND gates", IEEE Electronics Letters, Vol. 40, No. 18, pp. 1100-1101, September 2004.

[16] M. Omaña, D. Rossi and C.Metra, "High Speed and Highly Testable Parallel Two-Rail Code Checker", in Proc. of IEEE Design, Automation and Test in Europe Conference, pp. 608-613, 2003.

[17] S. Kundu, E. S. Sogomonyan, M. Goessel and S. Tarnick, "Self-Checking Comparator with One Periodic Output", in IEEE Trans. on Computers, Vol. 45, No. 3, March 1996.

[18] J-C.Lo, "A Novel Area-Time Efficient Static CMOS Totally Self-Checking Comparator", IEEE Journal of Solid-State Circuits, Vol. 28, No. 2, February 1993.