2014 17th Euromicro Conference on Digital System Design

Probabilistic Gate Level Fault Modeling for Near and Sub-Threshold CMOS Circuits

Alexandru Amaricai Sergiu Nimara Oana Boncalo
University Politehnica Timisoara
Timisoara, Romania
{alexandru.amaricai, oana.boncalo} @cs.upt.ro

Jiaoyan Chen Emanuel Popovici
University College Cork
Cork, Ireland
{chenj, e.popovici} @ucc.ie

***Abstract*—This paper presents gate level delay dependent probabilistic fault models for CMOS circuits operating at sub-threshold and near-threshold supply voltages. A bottom-up approach has been employed: SPICE simulations have been used to derive higher level error models implemented using Verilog HDL. HSPICE Monte-Carlo simulations show that the delay dependent probabilistic nature of these faults is due to the process-voltage-temperature (PVT) variations which affect the circuits operating at very low supply voltages. For gate level error analysis, mutant based simulated fault injection (SFI) techniques have been employed for combinational netlist reliability analysis. Four types of gate level fault models, with different accuracies, are proposed. Our findings show that the proposed SFI method presents a 2X-5X simulation time overhead compared to the simulation of the gold circuit; with respect to SPICE analysis, the proposed method requires three orders of magnitude less simulation time.**

Keywords—simulated fault injection, probabilistic CMOS, sub-threshold circuits

I. INTRODUCTION

Energy consumption represents one of the major issues in deep sub-micron CMOS technologies. An important drawback for these technologies is represented by the high leakage currents; therefore, static power becomes the dominant component in the overall power consumption. The most effective way to reduce circuit's power is represented by supply voltage (Vdd) reduction. This reduces both the static and the dynamic components. Recent trends have shown dramatic reduction of the supply voltage, to near-threshold and even sub-threshold regions of operation [1][2]. However, such dramatic Vdd decrease imposes two challenges: low performance and low reliability. Regarding the former, reduced supply voltage coupled with aggressively scaled down transistors means that digital circuits become less resilient to small noises in the supply and ground lines, temperature and process variations. These PVT variations may induce probabilistic behavior of logic gates, which provide the correct output in the desired time with a probability less than 1 [3].

This paper deals with the analysis of the behavior of circuits operating in the very low supply voltage region under PVT variations. We have performed SPICE Monte-Carlo simulations (similar to the one in [4]) of logic gates in order to extract gate level fault behavior. The simulations have been carried out for different supply voltages in the sub-threshold and near-threshold regions (0.25V, 0.3V, 0.35 V) and different temperatures (25°C, 50°C, 75°C). Voltage and process variations (thickness of oxide and threshold voltage) have been considered using a Gaussian distribution. Based on the SPICE simulation results, we have derived four probabilistic fault models with different accuracies. The less accurate is represented by a simple probabilistic function of the gate's output; the most accurate considers different probabilities for each switching type at the gate's input.

We have developed generic gate level fault injection techniques in Verilog HDL for each of the four fault models. We have chosen Verilog HDL, as it allows reliability analysis for gate level net-lists generated by open-source synthesis tools, such as ABC [5]. The probabilities of gate failure are generated based on three parameters: temperature, supply voltage and desired gate delay. We have performed several simulation campaigns for ripple carry adders and carry select adders (in normal configuration and triple modular redundancy configuration for increased noise resilience). The performed simulation campaigns show the flexibility of the proposed technique: each logic gate can be injected according to a desired probability (dependent on the gate delay and voltage constraints).

This paper is organized as follows: Section II presents the circuit level analysis of logic gates operating at sub and near-threshold voltages under PVT variations, as well as the extracted probabilistic fault models; the proposed HDL based simulated fault injection (SFI) methodology for gate level net-list is depicted in Section III, while Section IV details the gate level simulation campaigns; the last section presents some concluding remarks.

II. DELAY DEPENDENT FAULT MODELS IN SUB-POWERED CMOS CIRCUITS

A. Circuit-Level Analysis

We have performed HSPICE simulations for NOT, NAND, AND, Majority Voters and XOR gates, in 45 nm CMOS technology. However, any basic (such as NOR) or complex logic gate can be analyzed using the same methodology. We have simulated these circuits under different supply voltage and temperature variations. The considered supply voltages are 0.25 V, 0.3 V and 0.35 V. The selected temperatures are 25ºC, 50ºC and 75ºC. The threshold voltages in the MOS transistors SPICE models are -0.302 V for PMOS and 0.322 V for NMOS. Hence, our analysis covers both the sub-threshold and near-threshold operation regions [11].

We have performed Monte-Carlo simulations consisting of 10.000 runs for each parameter. Both supply voltage variations and process variations have been considered. For supply voltage variations, a Gaussian distribution with 0.05 V deviation and sigma 1 has been selected. Whereas, for process variations, two sets of parameters have been considered: threshold voltage and oxide thickness. For threshold voltage a Gaussian distribution with 0.05 V deviation and sigma 1 has been used, while for oxide thickness Gaussian distribution with 10% deviation and sigma 3 has been employed. Each gate allows four identical gates as output loads. For the inputs, the rise and fall times are 0.1 ns. Delay has been measured as the time gap between input cross half of the supply voltage and output cross half of Vdd [11].

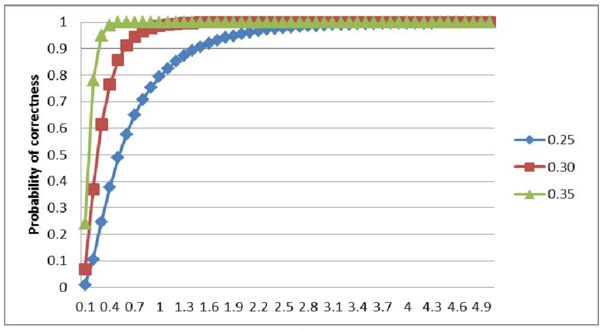
The results extracted for the 2-input NAND gate are exemplified in Fig. 1. It can be noted that the probability of a correct output is dependent to the considered gate delay (a larger delay results in greater probability of a correct switch), the type of switching at the gate inputs and supply voltage. Similar results have been obtained for the other considered gates. For our study we have considered certain assumptions (i.e. the input transition from 11 to 10 is considered the same as 11 to 01; skew at input signals has not been taken into account) that help simplify the analysis without major expected impact on the probabilistic nature of switching for sub-powered CMOS logic gates behavior. Furthermore, our results show that very large gate delays (more than 5 ns for a gate) yield a correct output. In addition to this, there is a wide range of applications which do not require 100% correct results. For these, correctness can be trade-off for better power consumption (due to aggressive voltage scaling) and performance (by considering small gate delays). In these cases, probabilistic reliability analysis should be performed in order to estimate the overall probability of correctness for a circuit for given performance and voltage constraints.

B. *Probabilistic Logic Level Fault Models*

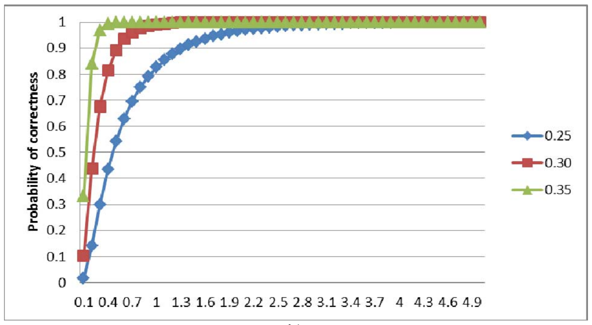
The simulations described in the previous section have shown the probabilistic nature of the gate switching for sub-powered CMOS circuits. Depending on the targeted accuracy, the following fault models have been derived:

1. **Gate output probabilistic model (GOP)** – For this model, the gate has the correct output with a given probability. The faulty output may appear due to two factors: the inability of the gate to switch in a given amount of time and a bit-flip of the output (similar to the single event upsets). The undesired bit-flips may have negligible effects on the overall circuit output, due to its limited propagation; we expect that in sub-threshold and near-threshold regions the propagation of glitches associated to the bit-flips is minimized by high gate delays associated with the very low supply voltages.

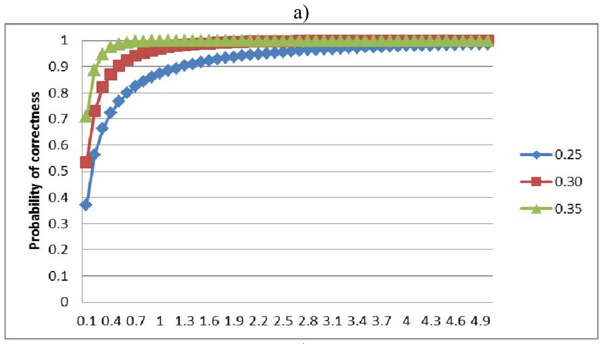
2. **Gate output switching probabilistic model (GOS)** – For this model, the logic gate switches correctly with a probability, dependent on the supply voltage, temperature and considered gate delay.



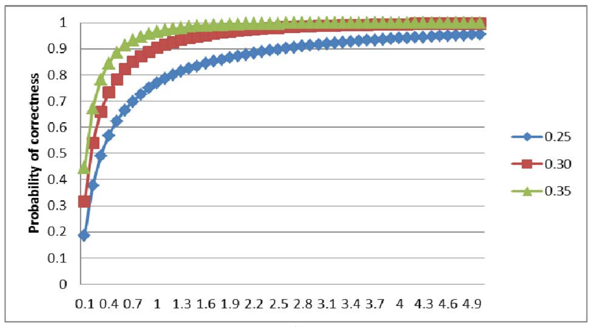
a)



b)



c)



d)

Fig. 1 – Probability of correctness function of considered gate delay (expressed in ns) for 2-input NAND gate for Vdd={0.25V, 0.3V and 0.35V) for 25ºC (a) – for 00 to 11 input switch (b) – for 01 or 10 to 11 input switch (c) – for 11 to 00 input switch and (d) – for 11 to 01 or 10 input switch

Similar graphs have been extracted for NOT, 2-input AND, 3-input Majority Voting, 2-input XOR gates for 25ºC, 50ºC and 75ºC.

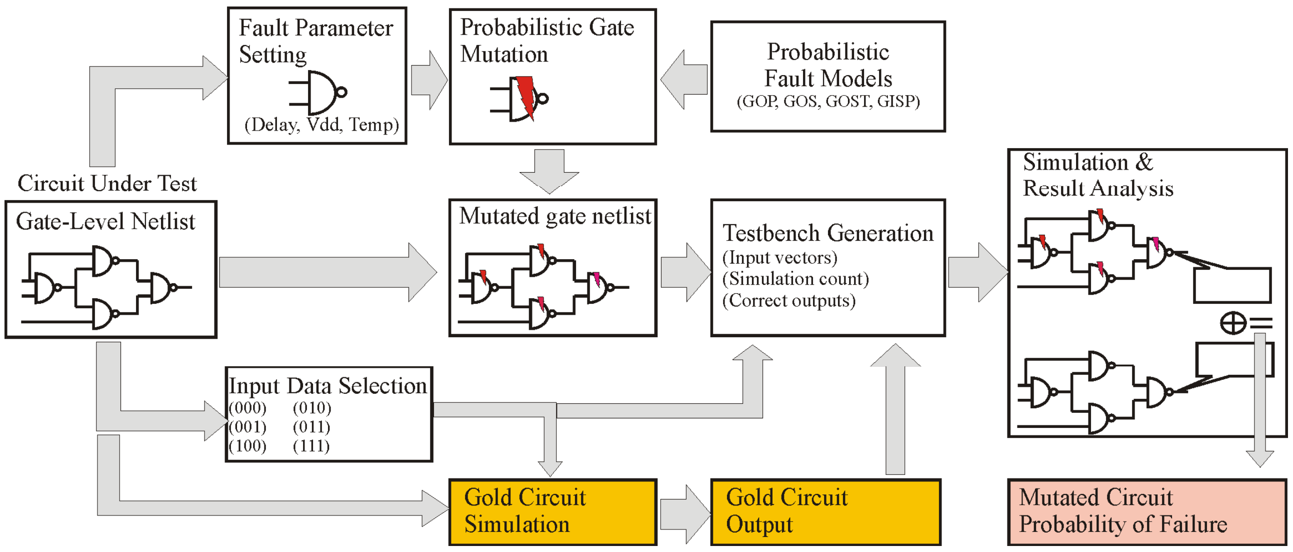


Fig. 2 – Simulated fault injection process for probabilistic error analysis

3. **Gate output switching type probabilistic model (GOST)** – Compared to the previous model, different probabilities for 0-to-1 and 1-to-0 gate switching are considered. The different probabilities for each type of switching are a consequence of the un-balanced NMOS and PMOS stages found in many logic gates. For balanced NMOS and PMOS stages, this fault model is equivalent to GOS.
4. **Gate input switching probabilistic model (GISP)** – For this model, we have different probabilities for each of the input switching combination. It is justified by the fact that each gate input determines the turn-on/turn-off of a pair of NMOS/PMOS gate transistors. For example, in a 2-input NAND gate, the transition 11-to-01 determines the opening of a single PMOS transistor, while the transition 11-to-00 determines the opening of both PMOS transistors. For the inverter gate, this model is equivalent to the previous fault model.

The probability in these models can be expressed either as a constant (for a given delay, voltage supply and temperature) or as a probability density function (a delay dependent function).

III. Gate Level Simulated Fault Injection

HDL Simulated Fault Injection (SFI) is a powerful tool to analyze the circuit behavior in the presence of faults [2][7][8]. SFI can be applied as soon as a system model is available in the design phase. A key ingredient for deriving good results from the fault injection process is the fault model. This techniques have been used to study the effect of permanent faults [2][8][9], to transient faults [8][9][10] that result in single event upsets and single event transients for simple to more complex circuits (e.g. Leon3 microprocessor [10], commercial microcontrollers [8]). SFI techniques have been classified in two main categories [2]: approaches that don't require any code instrumentation (i.e. simulator commands and scripts [7]) and those that require modifications of the HDL code (i.e. mutants and saboteur techniques [2][8]). The techniques based on code intervention either alter the characteristics of the signals from a structural description (i.e. *saboteurs*), or replace a component description with one that has a faulty behavior (i.e. *mutants*). Although simulator commands do not require code intervention, they are dependent on the simulator environment capabilities. Furthermore, unless a tool such as the one presented in [7] is available, the applicability of this technique for large circuits is cumbersome. Typically a SFI campaign consists of three phases [2][7][8]: set-up phase (i.e. select fault models, fault locations, number of runs, do required changes to the simulation model and or prepare simulation scripts), simulation phase (when the actual simulation takes place), and data processing/analysis phase (reliability estimates are derived and the system behavior in the presence of noise characterized; typically the simulation results are compared against a gold circuit, that is a circuit that is non-faulty).

The proposed SFI methodology comprises two major phases: set-up phase and simulation and data analysis phase. The setup phase comprises the following operations:

1. **Fault parameter settings** – the three parameters (gate delay, Vdd and temperature) are set for each gate; the gate delay parameter can be extracted from the gate level netlist (i.e. gates on the critical path may have tighter delay constraints); different Vdd and temperature may be considered for different gates.
2. **Probabilistic gate mutation** – based on the gate fault parameters and the fault model, the corresponding mutation is selected for each gate; the mutated circuit netlist is created;
3. **Input data selection** – due to data dependence of the GOS, GOST and GISP fault models, circuit input data represents a very important aspect in the reliability analysis

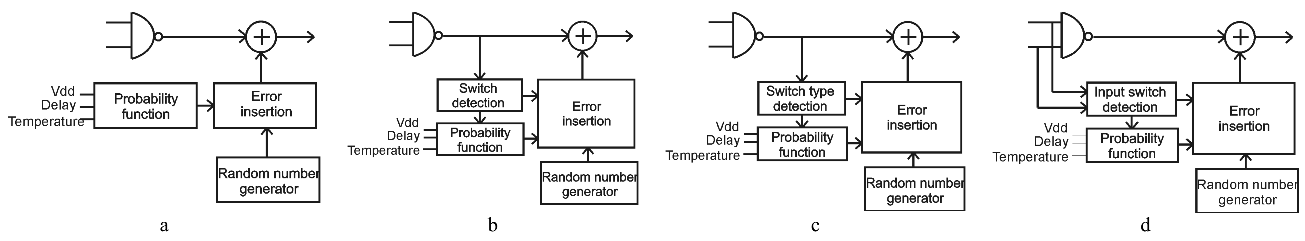


Fig. 3 – Mutant architecture of NAND gates with the four fault models
(a – GOP, b – GOS, c – GOSP, d – GISP)

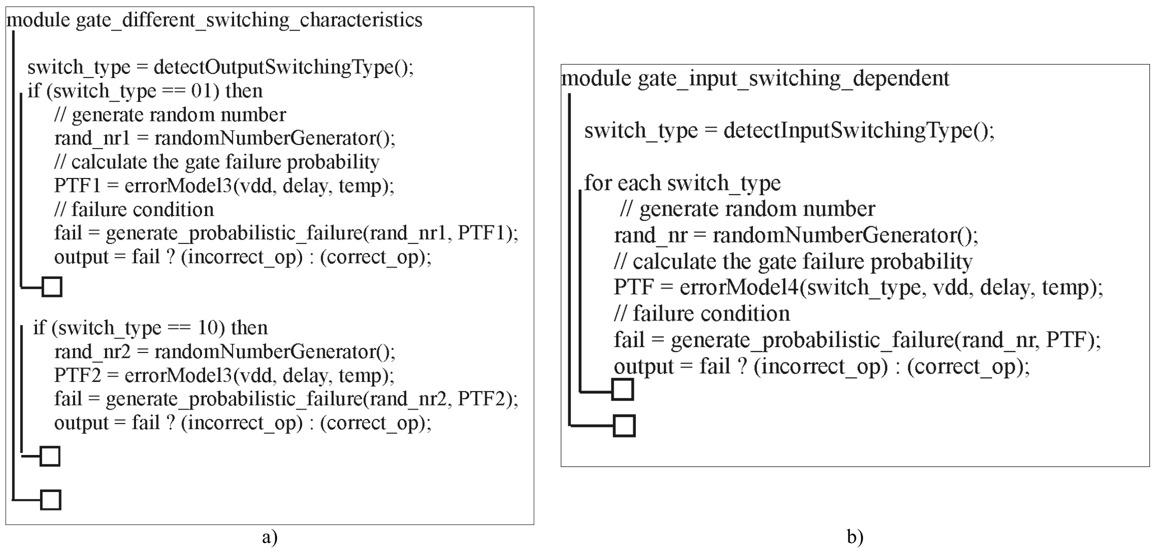
4. **Gold circuit simulation** – this simulation is performed in order to extract the correct outputs for the considered input data.
5. **Testbench generation** – input data, correct output and number of simulation are considered when developing the testbench module/entity which controls the simulation and analyzes the results.

The second phase comprises the actual simulation and result analysis. Because the fault models are probabilistic, a large number of simulations are required. The result analysis is performed almost simultaneously with the simulation: after simulating the circuit for each input, the output is compared with the gold circuit output provided for testbench generation. The SFI process is depicted in Fig. 2.

We have chosen the mutant technique for implementing probabilistic fault injection. A mutant architecture has been designed for each of the proposed fault models. Their construction is depicted in Fig. 3. We targeted increase flexibility for the proposed mutant architectures. Each mutated gate in the netlist is tuned by a different set of voltage, temperature and delay parameters. This represents a highly desirable feature, allowing the reliability analysis of a wide range of circuits such as circuits with un-balanced delay paths, circuits with multiple voltage islands and circuits with regions that heat up differently. The mutants have been implemented in Verilog HDL. We have chosen Verilog in order to analyze output provided by open-source logic synthesis tools, such as ABC. However, the proposed techniques can be easily extended to VHDL. Fig. 4 depicts the pseudo-code associated to the mutant modules which implement GOST and GISP fault models.

IV. SIMULATIONS

We have performed several simulation campaigns, detailed in Tables I, II and III. The simulations have been carried using Modelsim 10.05 SE commercial simulator on computer with Intel Core i5 at 2.4GHz and 4 GB of main memory with Windows 7 OS. A number of 16 000 test vectors have been applied for each campaign. Verilog testbenches are used in order to perform both the simulation phase and the result analysis phase. Result analysis is done by comparing the SFI results with those of the gold circuit computed previously. In our campaigns, we have extracted two types of results: the probability of failure for each bit of the result and the overall probability of failure for the entire circuit.



```
module gate_different_switching_characteristics

  switch_type = detectOutputSwitchingType();
  if (switch_type == 01) then
      // generate random number
      rand_nr1 = randomNumberGenerator();
      // calculate the gate failure probability
      PTF1 = errorModel3(vdd, delay, temp);
      // failure condition
      fail = generate_probabilistic_failure(rand_nr1, PTF1);
      output = fail ? (incorrect_op) : (correct_op);

  if (switch_type == 10) then
      rand_nr2 = randomNumberGenerator();
      PTF2 = errorModel3(vdd, delay, temp);
      fail = generate_probabilistic_failure(rand_nr2, PTF2);
      output = fail ? (incorrect_op) : (correct_op);
```

a)

```
module gate_input_switching_dependent

  switch_type = detectInputSwitchingType();

  for each switch_type
      // generate random number
      rand_nr = randomNumberGenerator();
      // calculate the gate failure probability
      PTF = errorModel4(switch_type, vdd, delay, temp);
      // failure condition
      fail = generate_probabilistic_failure(rand_nr, PTF);
      output = fail ? (incorrect_op) : (correct_op);
```

b)

Fig. 4 – Pseudo-code for GOST (a) and GISP (b) mutant architectures

TABLE I – SIMULATION RESULTS FOR 6-BIT RCA

Circuit	Fault Model	Delay (ns)	Vdd (V)	Sum Bits Failure Probabilities							Circuit Failure Prob.	Sim. Time (s)
				6	5	4	3	2	1	0		
6-bit RCA	GOS	1.5	0.35	1.01	2.34	3.05	2.88	2.74	1.90	1.66	10.16	1
			0.30	6.38	13.50	16.53	16.72	15.63	11.53	10.84	50.09	2
			0.25	22.63	39.07	42.90	42.28	43.79	37.58	35.80	90.16	2
	GOST	1.5	0.35	0.97	2.03	2.56	2.54	2.39	1.88	1.86	9.46	1
			0.30	5.23	13.19	14.59	15.19	14.14	10.48	12.54	49.53	1
			0.25	21.25	39.09	41.06	42.56	42.73	38.19	37.27	90.03	1
	GISP	1.5	0.35	0.99	2.94	3.25	3.44	3.31	2.51	2.60	12.96	1
			0.30	5.50	14.22	17.48	17.66	17.79	12.56	15.59	55.44	1
			0.25	20.08	39.29	41.18	43.59	43.74	41.29	39.79	90.48	1
6-bit RCA	GOS	Carry chain 1 Sum gates 1-4	0.35	3.34	7.27	7.66	6.88	5.76	3.21	3.44	20.44	1
			0.30	12.39	25.73	26.02	25.28	21.58	12.78	13.77	64.03	1
			0.25	35.61	48.42	45.68	46.68	46.16	38.21	45.53	92.41	2
	GOST	Carry chain 1 Sum gates 1-4	0.35	3.10	6.50	6.39	5.97	5.29	2.09	3.54	18.53	2
			0.30	11.26	24.14	24.21	24.16	21.49	10.78	14.91	62.16	2
			0.25	31.89	46.72	44.44	46.26	46.03	36.71	44.85	92.03	1
	GISP	Carry chain 1 Sum gates 1-4	0.35	3.11	7.73	8.41	8.26	6.76	3.06	4.75	22.73	1
			0.30	10.99	24.79	25.81	25.96	21.89	11.94	17.64	65.64	2
			0.25	33.38	46.11	45.63	46.84	47.01	39.16	46.66	92.31	1
Gold 6-bit RCA				-	-	-	-	-	-	-	-	0.5

TABLE II – SIMULATION RESULTS FOR 6-BIT CSEA

Circuit	Fault Model	Delay (ns)	Vdd (V)	Sum Bits Failure Probabilities							Circuit Failure Prob.	Sim. Time (s)
				6	5	4	3	2	1	0		
6-bit CSeA	GOS	1.5	0.35	1.39	3.13	3.69	3.99	2.91	1.79	1.66	12.17	1
			0.30	8.73	15.84	18.90	19.04	15.01	11.24	11.03	55.59	1
			0.25	29.81	39.71	42.14	44.28	42.28	37.73	36.83	91.09	2
	GOST	1.5	0.35	1.35	2.73	3.08	3.26	2.32	1.58	1.91	11.35	1
			0.30	6.96	14.23	16.51	18.29	14.34	11.28	12.99	54.36	1
			0.25	26.28	36.64	40.93	44.69	42.66	38.75	37.21	90.64	2
	GISP	1.5	0.35	1.14	3.23	4.04	3.98	3.10	2.29	2.56	14.13	1
			0.30	6.53	15.95	20.18	21.51	17.80	13.17	15.69	59.55	1
			0.25	26.09	39.94	41.50	45.06	45.11	40.64	39.85	91.70	1
6-bit CSeA	GOS	Carry chain 1 Sum gates 1-4 Mux 1.5	0.35	1.39	3.13	3.69	3.99	2.91	1.79	1.66	12.17	2
			0.30	8.73	15.84	18.90	19.04	15.01	11.24	11.03	55.59	1
			0.25	29.81	39.71	42.14	44.28	42.28	37.73	36.83	91.09	1
	GOST	Carry chain 1 Sum gates 1-4 Mux 1.5	0.35	1.35	2.73	3.08	3.26	2.32	1.58	1.91	11.35	1
			0.30	6.96	14.23	16.51	18.29	14.34	11.28	12.99	54.36	2
			0.25	26.28	36.64	40.93	44.69	42.66	38.75	37.21	90.64	2
	GISP	Carry chain 1 Sum gates 1-4 Mux 1.5	0.35	1.14	3.23	4.04	3.98	3.10	2.29	2.56	14.13	1
			0.30	6.53	15.95	20.18	21.51	17.80	13.17	15.69	59.55	2
			0.25	26.09	39.94	41.50	45.06	45.11	40.64	39.85	91.70	1
Gold 6-bit CSeA				-	-	-	-	-	-	-	-	0.5

SFI has been applied to 6-bit ripple carry adders (RCA) and carry-select adders (CSeA). They have been implemented using only 2-input NAND gates. For these designs the number of gates is consistently higher (up to 4X) compared to the classic XOR/Majority Voter configuration of the adder. We have chosen 6-bit adders because they are building blocks for variable node units and check node units of LDPC decoders. The main goal of these simulation campaigns is to prove the flexibility of the proposed approach and to analyze the simulation overhead required in the reliability analysis based on fault injection, as well as the adders' behavior in the presence of noise.

TABLE III – SIMULATION RESULTS FOR 6-BIT RCA IN TMR CONFIGURATION

Circuit	Fault Model	Delay (ns)	Vdd (V)	Sum Bits Failure Probabilities							Circuit Failure Prob.	Sim. Time (s)
				6	5	4	3	2	1	0		
TMR 6-bit RCA	GOS	Carry chain 1 Sum gates 1-4 Voter 2	0.30	10.99	20.72	20.99	17.88	14.71	7.99	8.15	56.81	5
	GOST	Carry chain 1 Sum gates 1-4 Voter 2	0.30	12.46	20.33	19.84	17.83	14.46	8.33	9.87	57.09	4
	GISP	Carry chain 1 Sum gates 1-4 Voter 2	0.30	10.29	21.14	21.74	20.61	16.33	7.81	10.73	57.88	4
TMR 6-bit RCA	GOS	Carry chain 1 Sum gates 1-4 Voter 2	Adder 0.3 Voter 0.35	8.05	18.69	19.19	17.02	12.37	4.75	6.14	48.23	5
	GOST	Carry chain 1 Sum gates 1-4 Voter 2	Adder 0.3 Voter 0.35	8.83	17.53	17.71	16.21	11.95	4.83	6.11	46.39	4
	GISP	Carry chain 1 Sum gates 1-4 Voter 2	Adder 0.3 Voter 0.35	9.16	18.60	20.09	18.85	13.79	5.55	8.56	50.94	4
Gold TMR 6-bit RCA				-	-	-	-	-	-	-	-	1

In the first two simulation campaigns, the same delays and supply voltages have been considered for all the gates in the circuit. The next two simulation campaigns have considered a more realistic case: the gates on the critical path (the carry chain) have tighter delay constraints than the others. The last two SFI campaigns have simulated 6-bit ripple carry adders in triple modular redundancy (TMR) configuration. The first series has used probabilities derived for the same Vdd in the entire circuit. The second series of TMR campaigns has considered different supply voltages for the adders and the voters. This way, we have simulated the case of integrated circuits with multiple voltage islands, where critical operations in terms of reliability and performance are executed on islands with higher Vdd.

Regarding the results of the simulations, we observe that the CSeA configuration has better reliability with respect to the RCA. Furthermore, we observe that the TMR configuration (when all the circuit operates at the same Vdd) does not significantly improve the reliability of the RCA for the considered gate probabilities. Both the RCA and the RCA in TMR configuration at Vdd=0.3V are more likely to give an erroneous result (the probability of erroneous output is higher than 50%).

The results show that in 6-bit adders (with carry-out), the most error prone bits are the most significant three bits of the sum (bits 4,5, and 3). The most "resilient" bits are the least significant bits (bits 0 and 1) followed by the carry-out bit.

Simulation times are also depicted in Tables I, II and III. The last rows of these tables provide the simulation times required for the gold circuit. The proposed approach has a simulation overhead of 2-4 times higher compared to the gold circuit simulation, due to 2 factors: the fault injection mechanism and the result analysis which is also performed in the simulation campaigns (result analysis is not required for the gold circuit simulation). However, for small and medium circuits (of up to several thousand gates), simulation times are negligible (maximum 5s), despite the high number of input vectors (16 000).

The SPICE campaign consisting of 1000 simulations of a 6-bit ripple carry adder has lasted 1h 25 min, which is three orders of magnitude compared to the Verilog logic level analysis.

V. CONCLUSIONS

This paper addresses the gate level fault modeling of probabilistic errors which appear in sub-threshold and near-threshold CMOS circuits. The contributions of this paper are:

- Analysis of gate failure probability dependence on delay, supply voltage and temperature by means of SPICE simulations of Invert, NAND, AND, Majority Voting and XOR gates under PVT variations;
- Different accuracy gate level fault models for probabilistic sub-powered CMOS circuits
- Flexible mutant based SFI architectures for gate level descriptions of probabilistic sub and near-threshold circuits

Furthermore, we have performed several SFI simulation campaigns for several reliability assessments of gate level net-lists. In order to show the flexibility of the proposed approach for reliability analysis of sub-powered circuits, probability parameters (temperature, delay, voltage) have been varied according to the topology of the gate level description.

The proposed methodology introduces a reasonable overhead compared to the gold circuit simulation, due to fault injection on one hand and result analysis on the other hand. However, compared to a SPICE based analysis of medium complexity circuits, the logic level simulation requires three orders of magnitude less simulation time.

Although gate level analysis may prove slow for very large netlists (consisting of millions of gates), it represents a necessary step for deriving probabilistic fault models for RTL descriptions. As it has been shown in this paper, lower level analysis is used to provide the fault models for next level of abstraction. In this context, the proposed SFI methodology represents a valid option for reliability analysis of building blocks and fault model extraction for RTL descriptions.

ACKNOWLEDGMENTS

This work has been supported by the Seventh Framework Program of European Union under Grant Agreement 309129, project i-RISC – "Innovative Reliable Chip Design from Low Power Unreliable Components".

REFERENCES

[1] S. Jain, S. Khare, S. Yada et al., "A 280mV-to-1.2V wide-operating-range IA-32 processor in 32nm CMOS," 2012 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012, pp. 66-68

[2] V. De, "Near-Threshold Voltage design in nanoscale CMOS", Proc 2013 Design Automation & Test in Europe (DATE), pp 612

[3] P. Korkmaz, B.E.S Akgul, K. Palem, "Energy, Performance, and Probability Tradeoffs for Energy-Efficient Probabilistic CMOS Circuits" IEEE Trans. On Circuits and Systems I, vol. 55, Issue 8, 2008

[4] Merrett, M., Asenov, P., Yangang Wang and Zwolinski, M.,"Modelling circuit performance variations due to statistical variability: Monte Carlo static timing analysis" Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011, vol. 1, pp. 1-4, March 2011

[5] D. Mischenko, S. Chatarjee, R. Brayton, "DAG-Aware AIG Rewriting" Proc. Design Automation Conference (DAC), 2006

[6] E. Jenn et al., "Fault Injection into VHDL Models: The MEFISTO Tool". International Symposium on Fault Tolerant Computing, pp. 66-75, 1994.

[7] Weiguang Sheng, Liyi Xiao, Zhigang Mao, "An Automated Fault Injection Technique Based on VHDL Syntax Analysis and Stratified Sampling", 4th IEEE International Symposium on Electronic Design, Test & Applications, 2008.

[8] D.Gil, L.J. Saiz, J. Gracia, J.C. Baraza, P.J. Gil, "Injecting Intermittent Faults for the Dependability Validation of Commercial Microcontrollers", High Level Design Validation and Test Workshop, HLDVT '08, 2008.

[9] S. R. Seward and P. K. Lala, "Fault Injection for Verifying Testability at the VHDL Level". Proceedings of the International Test Conference, ITC'03, 2003.

[10] Wassim Mansour, Raoul Velazco, "SEU fault-injection in VHDL-based processors: a case study", Proceedings of the 2012 13th Latin American Test Workshop – LATW'12 , 2012.

[11] Jiaoyan Chen, Christian Spagnol, Satish Grandhi, Emanuel Popovici, Sorin Cotofana, Alexandru Amaricai, "Linear Compositional Delay Model for the Timing Analysis of Sub-Powered Combinational Circuits", Proc. International Symposium on VLSI (ISVLSI), 2014