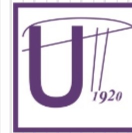
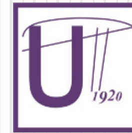


Gate Level HDL Simulated Fault Injection for Probabilistic Combinational Circuits

Oana Boncalo*, Sergiu Nimara*, Alexandru Amaricai*, Jiaoyan Chenº, Emanuel Popoviciº

* "Politehnica" University of Timişoara

º University College Cork



Research reported in this presentation was supported by the Seventh Framework Programme of the European Union, under Grant Agreement number 309129 (**i-RISC** project)

Outline

1. Goals of simulated fault injection (SFI) techniques
2. Introduction to SFI
3. SPICE analysis of sub-threshold CMOS circuits
4. Fault models
5. SFI for probabilistic CMOS circuits
6. Simulation results
7. Conclusion

Goals

- To develop error models at logic and gate level
- Develop gate level simulated fault injection (SFI) methodology
- Reliability assessment of gate-level netlists

Introduction to Simulated Fault Injection (SFI) (1)

- Reliability assessment -> can be performed using: analytical techniques, simulations or experiments on the physical devices
- Analytical techniques -> lowest cost and well suited in the early stages of the design process, but low fault modeling capability
- Experiments performed on the physical devices lead to the most accurate results
- Fault Injection -> a validation technique of the dependability of fault tolerant systems (FTSs)
- The observation of the system's behaviour in presence of faults is induced explicitly by the introduction of faults in the system

Introduction to Simulated Fault Injection (SFI) (2)

- Three main categories of fault injection techniques:
  - physical or Hardware Implemented Fault Injection - HWIFI
  - Software Implemented Fault Injection - SWIFI
  - simulation-based
- Simulation-based fault injection: useful for evaluating the dependability of a system during the design phase
- Offers both high observability and controllability of the modeled components
- Two main categories of HDL-based fault injection:
  - simulator commands (signals and variables)
  - HDL code modification
    - **Saboteurs**
    - **Mutants**

Introduction to Simulated Fault Injection (SFI) (3)

- Simulator commands -> the value or timing of the signals and variables of the model is modified using the commands of the simulator at simulation time
- A saboteur -> a special component which alters the value or timing characteristics of one or more signals when the fault is injected
  - During normal operation, a saboteur component remains inactive
- A mutant -> a component description which replaces the correct architecture of a module
  - While inactive, a mutant behaves like the original component; when activated it behaves like the component in the presence of faults

SPICE Analysis of Sub-threshold Circuits (1)

- Monte Carlo simulations set-up:

  - three basic gates (AND, NAND and inverter) are simulated in HSPICE, using 45 nm technology models

  - three types of variations involved: voltage supply variation, process variation and temperature variation

  - three values for Vdd: 0.35 V, 0.30 V and 0.25 V

  - three temperature values: 25 ^{0}C, 50 ^{0}C and 75 ^{0}C

  - Gaussian distribution with standard deviation 0.05 V and sigma 1

  - process variation:

  - oxide thickness (TOX)
  - threshold voltage (VTH)

SPICE Analysis of Sub-threshold Circuits (2)

- Monte Carlo simulations set-up:

  - TOXn = 1.14e-09 ; TOXp = 1.26e-09

  - VTHn = 0.322 V; VTHp = - 0.302 V

  - TOX: Gaussian distribution: 10% deviation; sigma = 3

  - VTH: Gaussian distribution: 0.05V deviation; sigma=3

  - rise and fall time of the inputs are set to 0.1 ns

  - for the inverter gate, the width of PMOS (0.2 μm) is twice the width of NMOS (0.1 μm)

  - each gate has four identical devices as output load

SPICE Analysis of Sub-threshold Circuits (3)

- Probability of correctness for Vdd = 0.25 V

0.25VDD									
Input_switch	Temperature	delay/0	1E-10	3E-10	5E-10	7E-10	1,00E-09	1,5E-09	1,9E-09
00_11	25C	0	0,009039	0,247043	0,488252	0,64938	0,793488	0,907189	0,948449
00_11	50C	0	0,006004	0,234005	0,489239	0,661235	0,811623	0,923658	0,961041
00_11	75C	0	0,004156	0,224828	0,493611	0,675318	0,829946	0,938126	0,9711
01_11 (same as 10_11)	25C	0	0,016273	0,29818	0,542073	0,694999	0,826089	0,925129	0,959642
01_11	50C	0	0,012527	0,291255	0,548389	0,709054	0,843092	0,938709	0,969462
01_11	75C	0	0,00938	0,284166	0,555454	0,724048	0,860252	0,951135	0,977751
11_00	25C	0	0,370218	0,662184	0,766742	0,823006	0,872236	0,916216	0,936506
11_00	50C	0	0,361091	0,658165	0,765144	0,822671	0,872889	0,917545	0,93802
11_00	75C	0	0,350345	0,655327	0,765783	0,824998	0,876391	0,921612	0,94207
11_10 (same as 11_01)	25C	0	0,185981	0,490779	0,622806	0,698601	0,768155	0,833826	0,865851
11_10	50C	0	0,1736	0,482106	0,618468	0,696998	0,769028	0,836793	0,869646
11_10	75C	0	0,165591	0,478669	0,619058	0,699971	0,774002	0,843219	0,876485

SPICE Analysis of Sub-threshold Circuits (4)

- Probability of correctness for Vdd = 0.35 V

0.35VDD

Input_switch	Temperature	delay/0	1E-10	3E-10	5E-10	7E-10	1E-09	1,5E-09	1,9E-09
00_11	25C	0	0,240751	0,950672	0,997717	0,999891	0,999999	1	1
00_11	50C	0	0,187746	0,939243	0,997111	0,999862	0,999998	1	1
00_11	75C	0	0,142369	0,926457	0,996442	0,999832	0,999998	1	1
01_11 (same as 10_11)	25C	0	0,331791	0,968701	0,998817	0,999953	1	1	1
01_11	50C	0	0,287438	0,956573	0,997913	0,999895	0,999999	1	1
01_11	75C	0	0,221855	0,952137	0,998059	0,99992	0,999999	1	1
11_00	25C	0	0,709231	0,948198	0,985864	0,995481	0,999052	0,999914	0,999986
11_00	50C	0	0,658966	0,916904	0,969871	0,987346	0,996043	0,999309	0,999812
11_00	75C	0	0,423807	0,8112	0,917329	0,959035	0,983865	0,99594	0,99853
11_10 (same as 11_01)	25C	0	0,444745	0,78272	0,885311	0,93188	0,965074	0,986545	0,9932
11_10	50C	0	0,434425	0,795007	0,900068	0,945023	0,974772	0,991868	0,996425
11_10	75C	0	0,423807	0,8112	0,917329	0,959035	0,983865	0,99594	0,99853

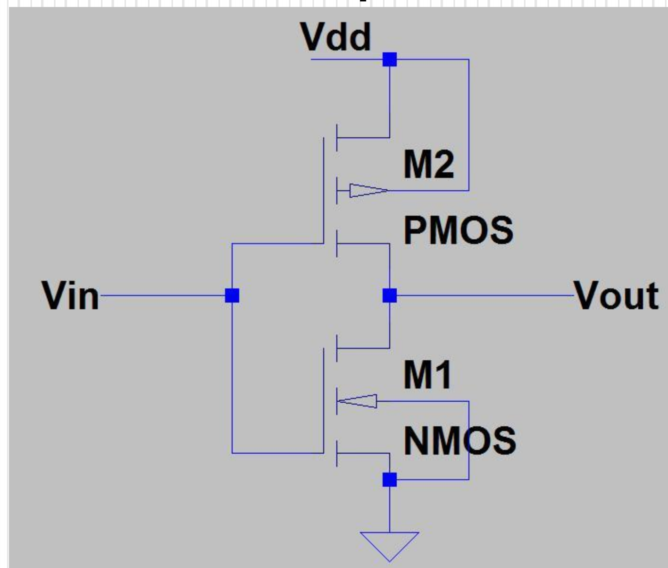
Fault Models (1)

- Fault model 1

  - the output of the logic gate is bit-flipped with a probability *p*

  - *p* is a function of three parameters: supply voltage, temperature and delay

  - the altered output can occur at any time, regardless of the binary values applied at the inputs or the previous value of the output



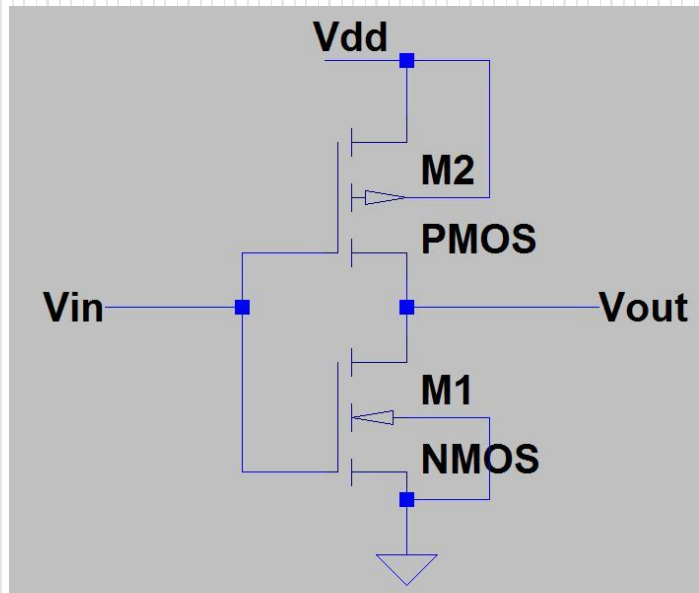
Fault Models (2)

- Fault model 2

  - the logic gate doesn't switch correctly

  - in the situation when the output of the gate must switch to the converse value, the output may be affected by a fault, with a probability *p*

  - an output switch detection is performed



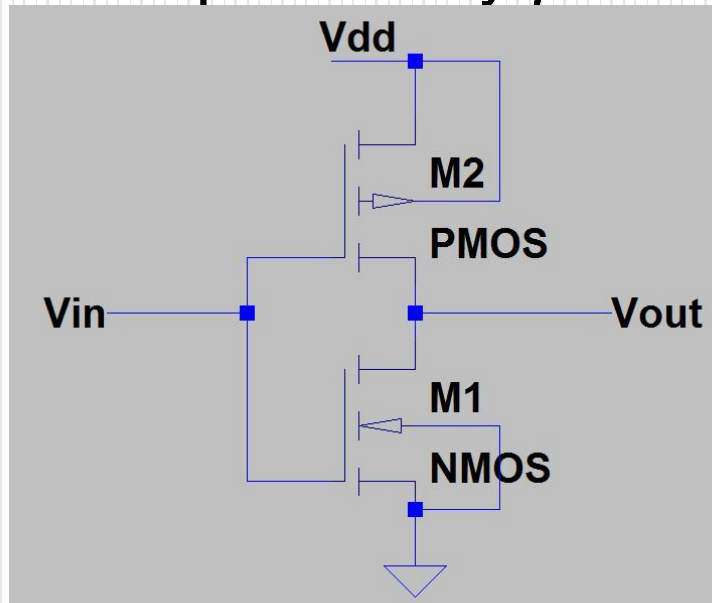
Fault Models (3)

- Fault model 3

  - different switching characteristics are considered

  - takes into account the type of output switching: from 0 to 1, or from 1 to 0

  - for the 0->1 transition we use a probabiltiy *p1* and for the 1->0 transition a probability *p2*



Fault Models (4)

- Fault model 4

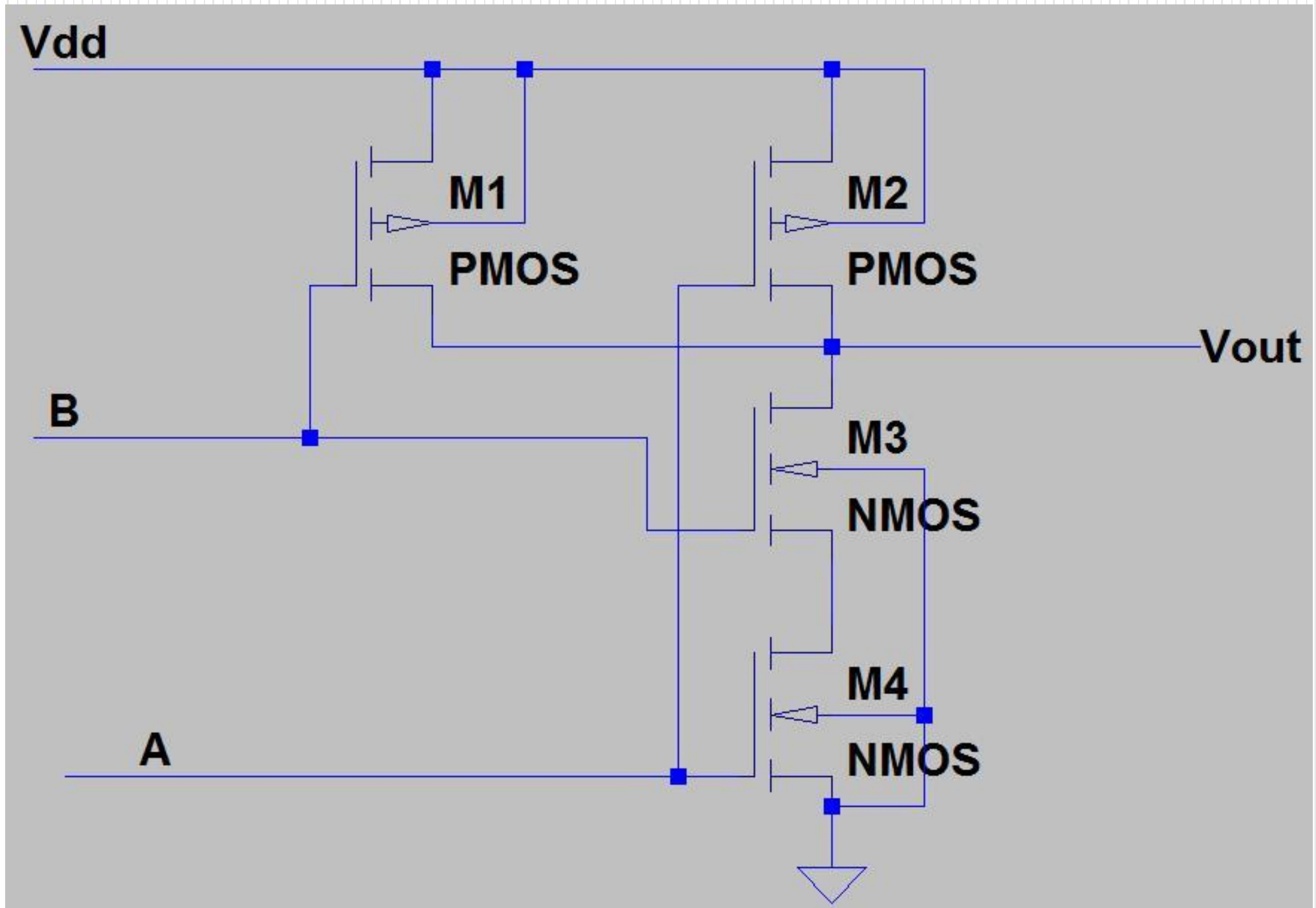
  - it's input dependent

  - the previous and the actual values of the inputs are compared and if the transition of the inputs also determines a transition of the output of the gates, a failure occurs with a certain probability

  - four distinct situations when the transition of the inputs determines the transition of the output

  - four distinct probabilities are calculated, each one as a function of the transition that occured at the inputs, the supply voltage, temperature and delay

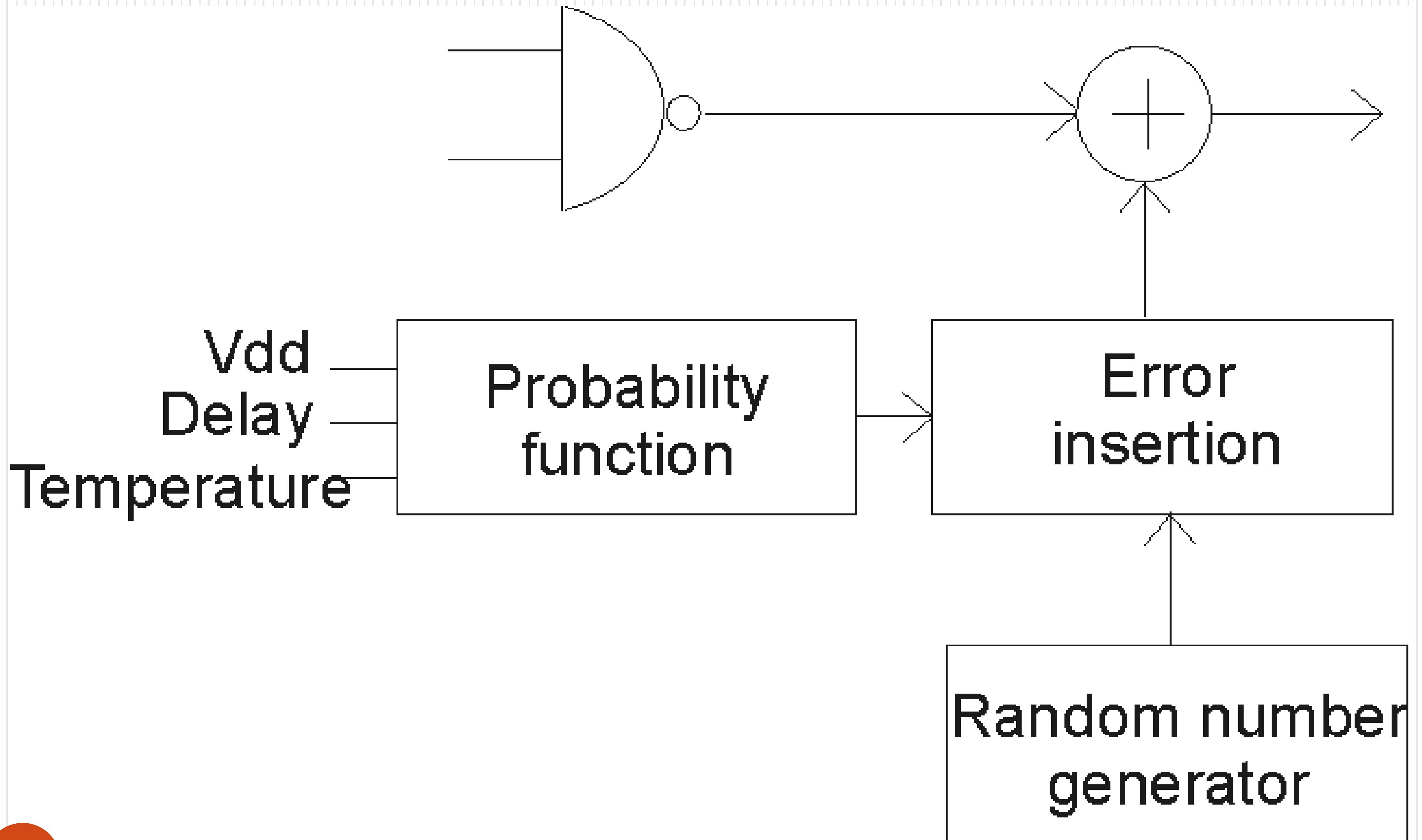
The Structure of the 2-inputs NAND Gate



Simulated Fault Injection for Probabilistic Circuits

- Generic gate level fault injection techniques were developed in Verilog HDL
- The reason for chosing Verilog: we desire to perform reliability analysis in gate level net-lists generated by open-source synthesis tools, such as ABC
- The reason for chosing mutants instead of saboteurs: the faults occur at component level, not at signal level

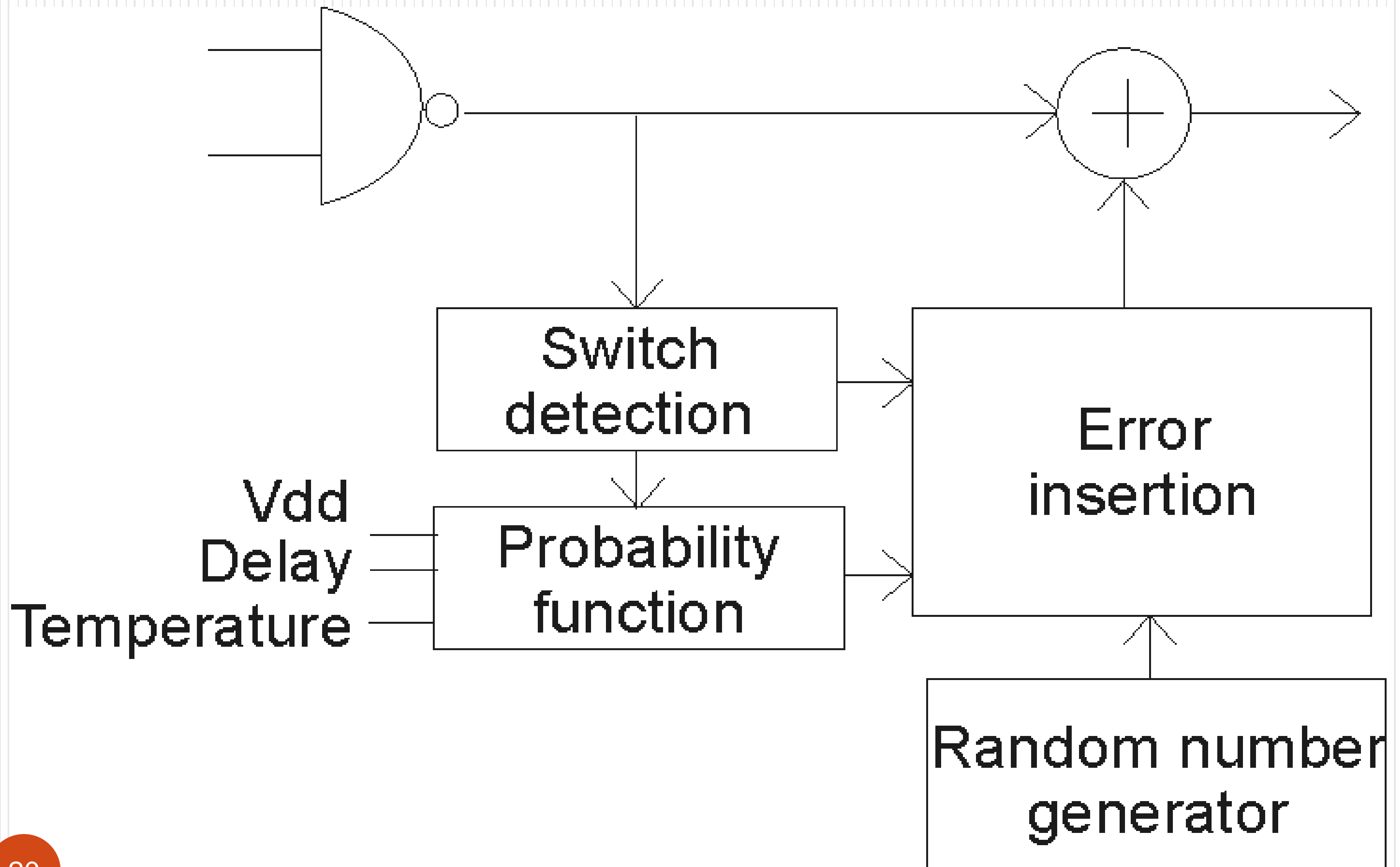
Mutant Fault Model 1



Pseudocode of Mutant 1

```
module gate_output_probability
    // generate random number
    rand_nr = randomNumberGenerator();
    // calculate the gate failure probability
    PTF = errorModel1(vdd,delay,temp);
    // failure condition
    fail = generate_probabilistic_failure(rand_nr,PTF);
    output = fail ? (incorrect_op) : (correct_op);
```

Mutant Fault Model 2

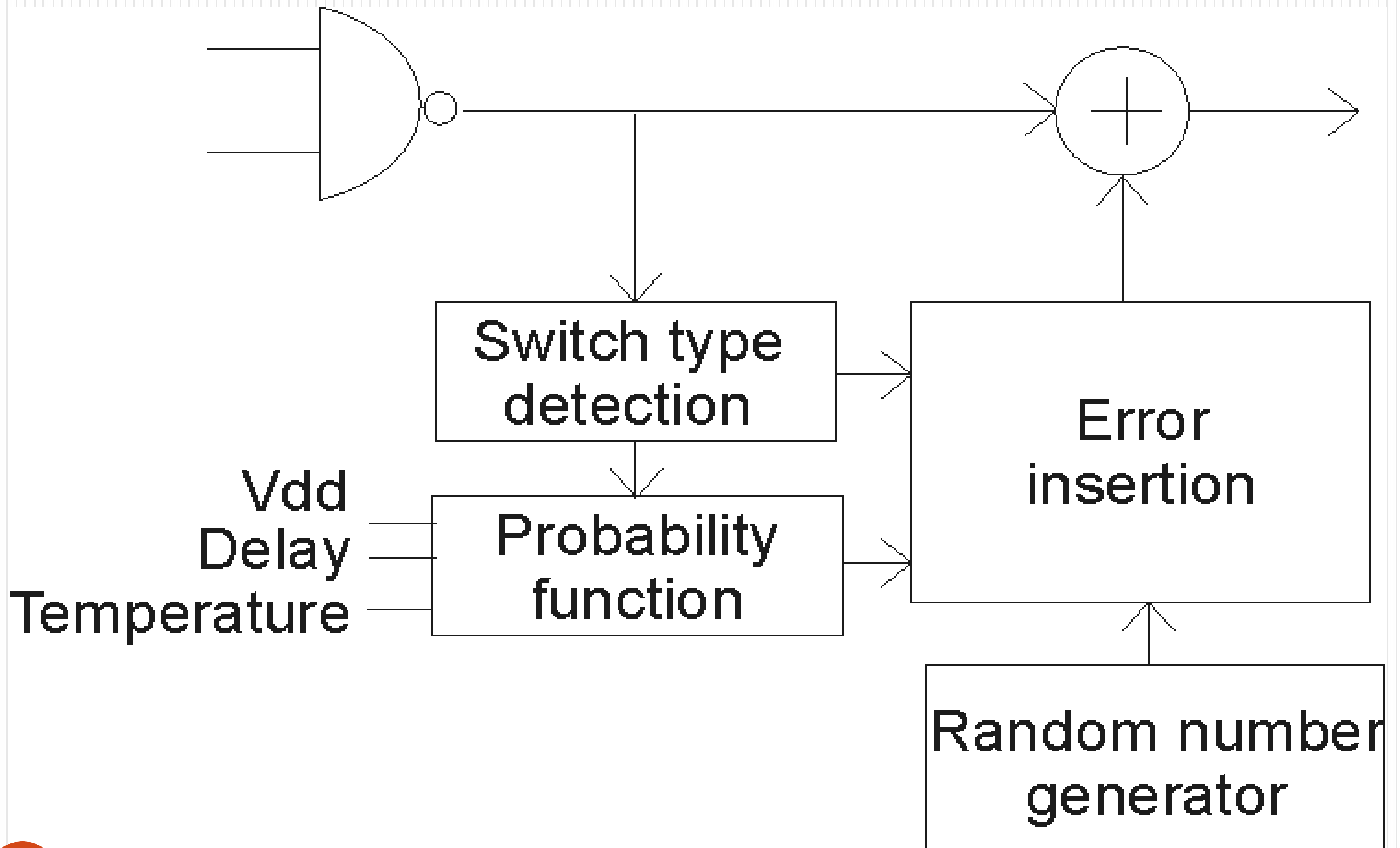


Pseudocode of Mutant 2

```
module gate_switching_probability

  if (old_output != new_output) then
      // generate random number
      rand_nr = randomNumberGenerator();
      // calculate the gate failure probability
      PTF = errorModel2(vdd,delay,temp);
      // failure condition
      fail = generate_probabilistic_failure(rand_nr, PTF);
      output = fail ? (incorrect_op) : (correct_op);
```

Mutant Fault Model 3



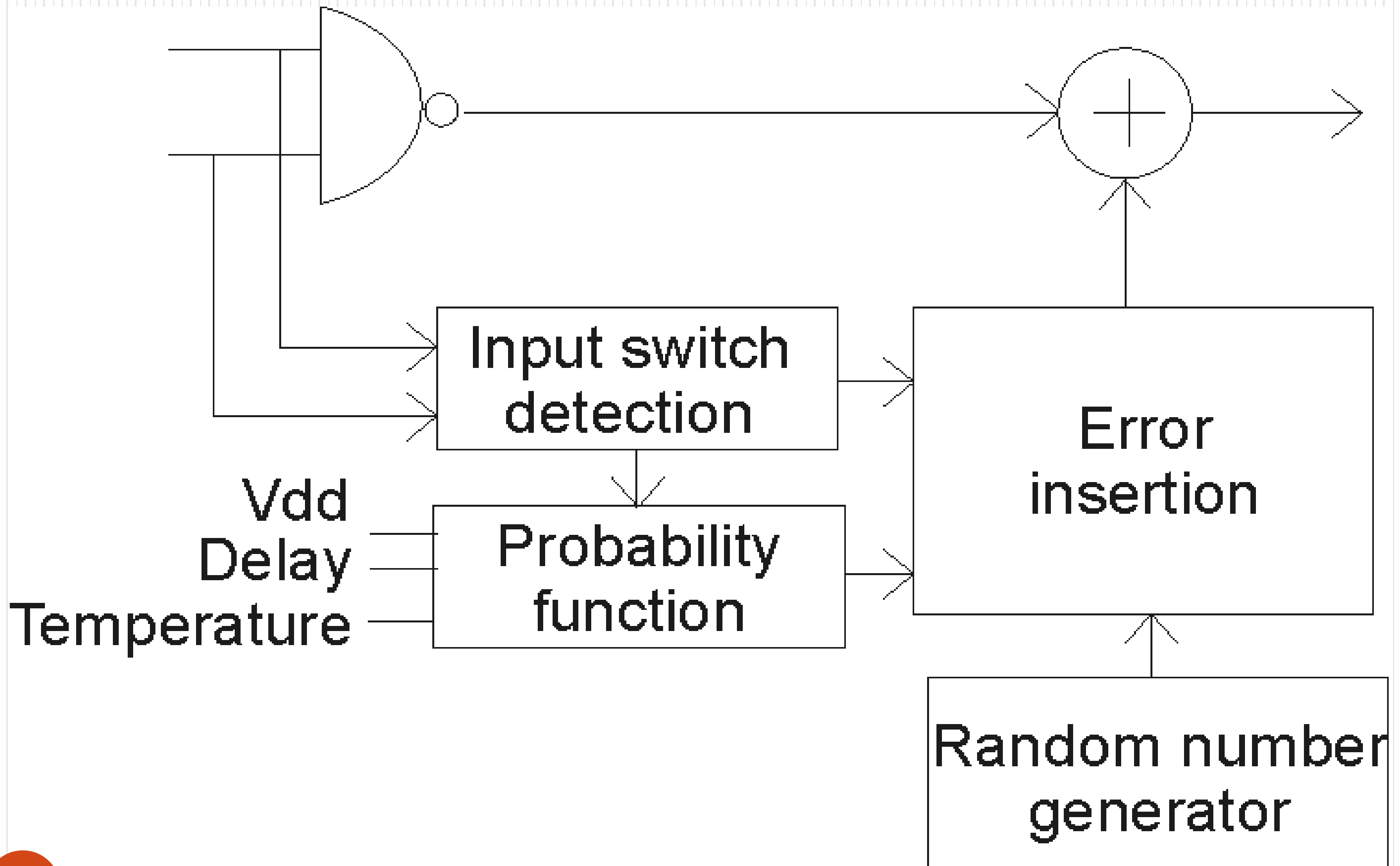
Pseudocode of Mutant 3

```
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);
```

Mutant Fault Model 4



Pseudocode of Mutant 4

```
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);
```

Pseudocode for Fault Injection Methodology

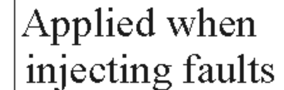
```
module majority_voting_v1(i, o);

  input [2:0] i;
  output o;
  wire wire1, wire2, wire 3, wire4, wire5, wire6;

 `ifdef FAULT_INJECTION
   nand_gate_input_switching_dependent gate1(.w1(i[2]), .w2(i[1]), .w_nand(wire1));
   nand_gate_input_switching_dependent gate2(.w1(i[2]), .w2(i[0]), .w_nand(wire2));
   nand_gate_input_switching_dependent gate3(.w1(i[1]), .w2(i[0]), .w_nand(wire3));
   nand3_gate_input_switching_dependent gate4(.w1(wire1), .w2(wire2), .w3(wire3), .w_nand(wire4));

  `else
    nand_gate gate1(.w1(i[2]), .w2(i[1]), .w_nand(wire1));
    nand_gate gate2(.w1(i[2]), .w2(i[0]), .w_nand(wire2));
    nand_gate gate3(.w1(i[1]), .w2(i[0]), .w_nand(wire3));
    nand3_gate gate4(.w1(wire1), .w2(wire2), .w3(wire3), .w_nand(wire4));

  `endif
```



Simulations Performed (1)

- Five simulation campaigns for different combinational circuits:

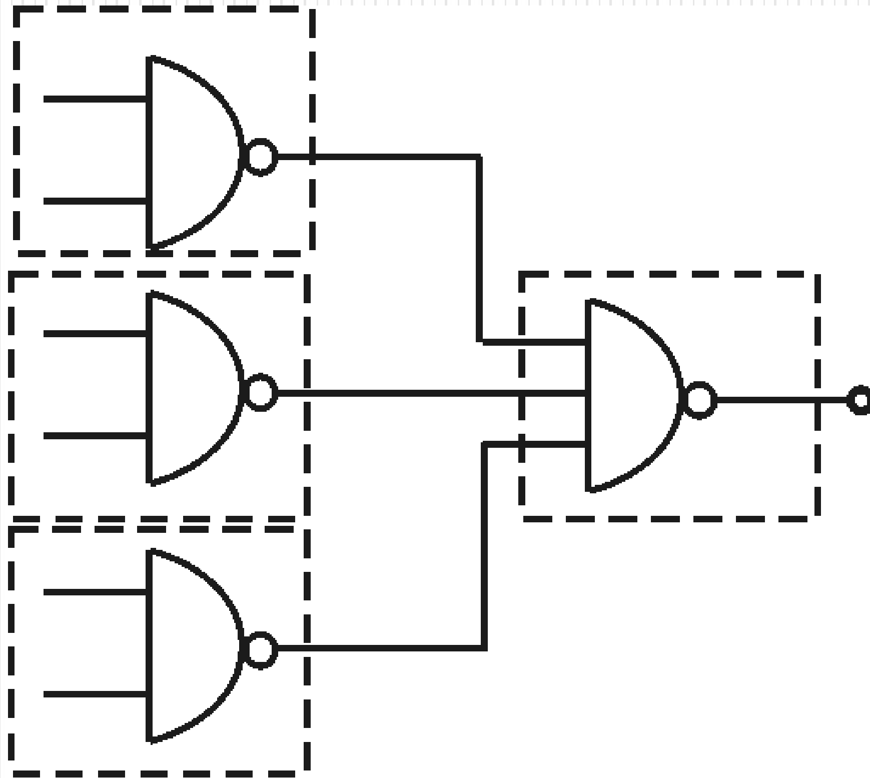
  - majority voting with equal / unequal paths

  - triple modular redundant adders

- For the majority voting with unequal paths, different delays were considered for gates on different paths
- For the triple modular redundant adders, different supply voltages were used for the adders and for the voter circuits

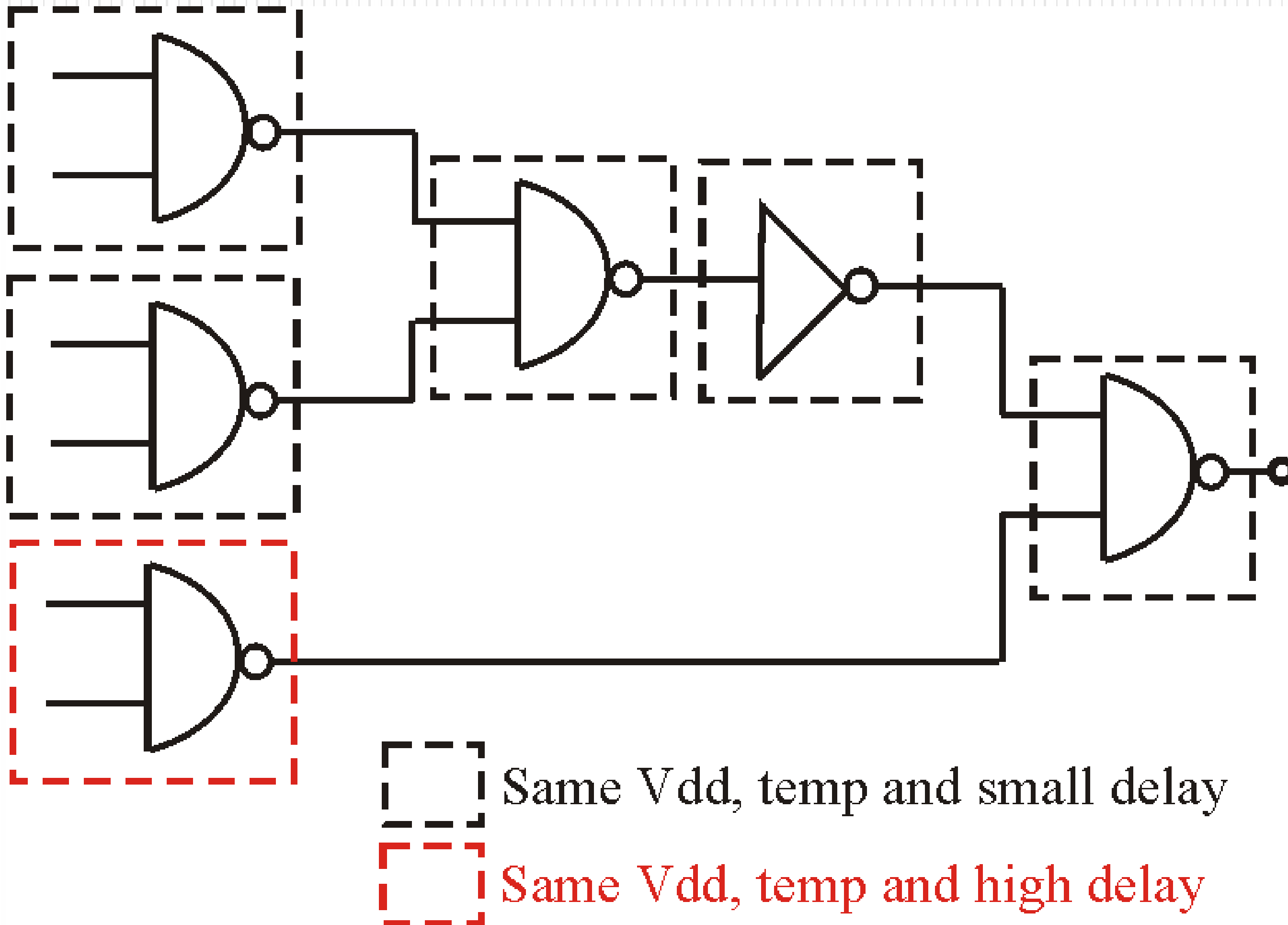
Simulations Performed (2)

- Majority voting with equal paths



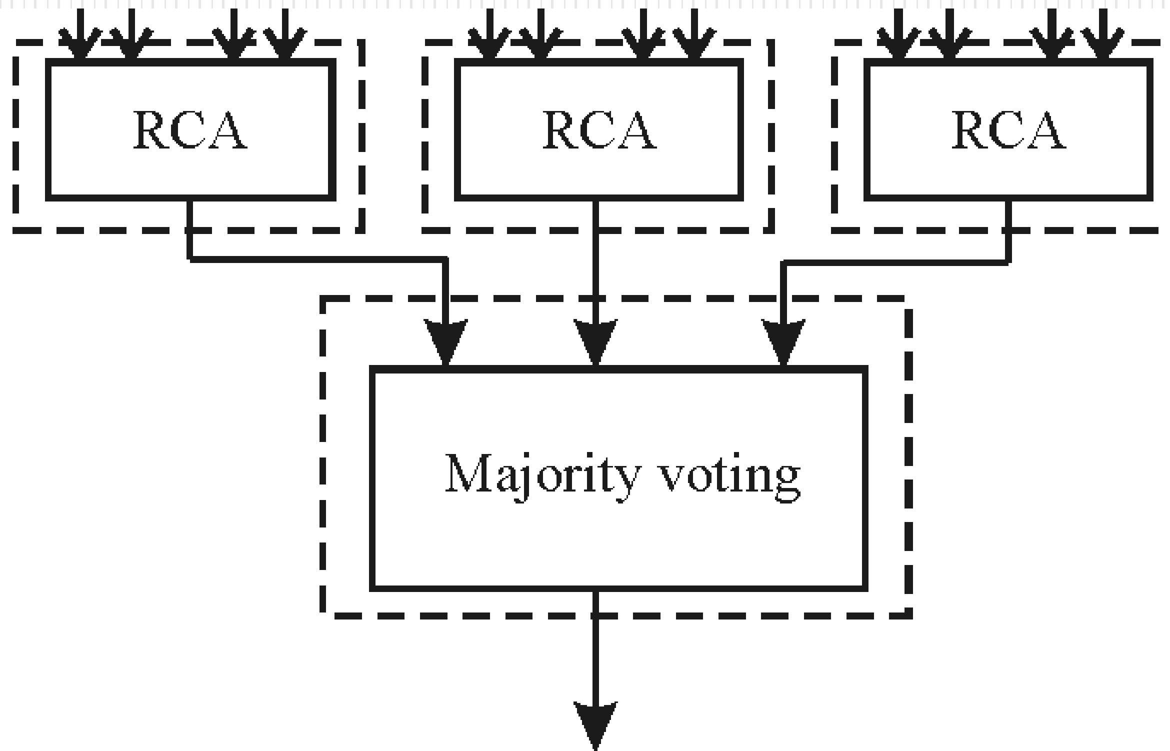
Simulations Performed (3)

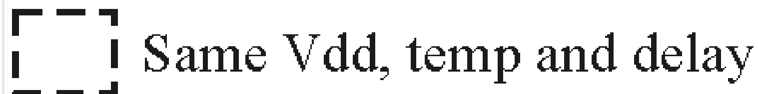
- Majority voting with unequal paths



Simulations Performed (4)

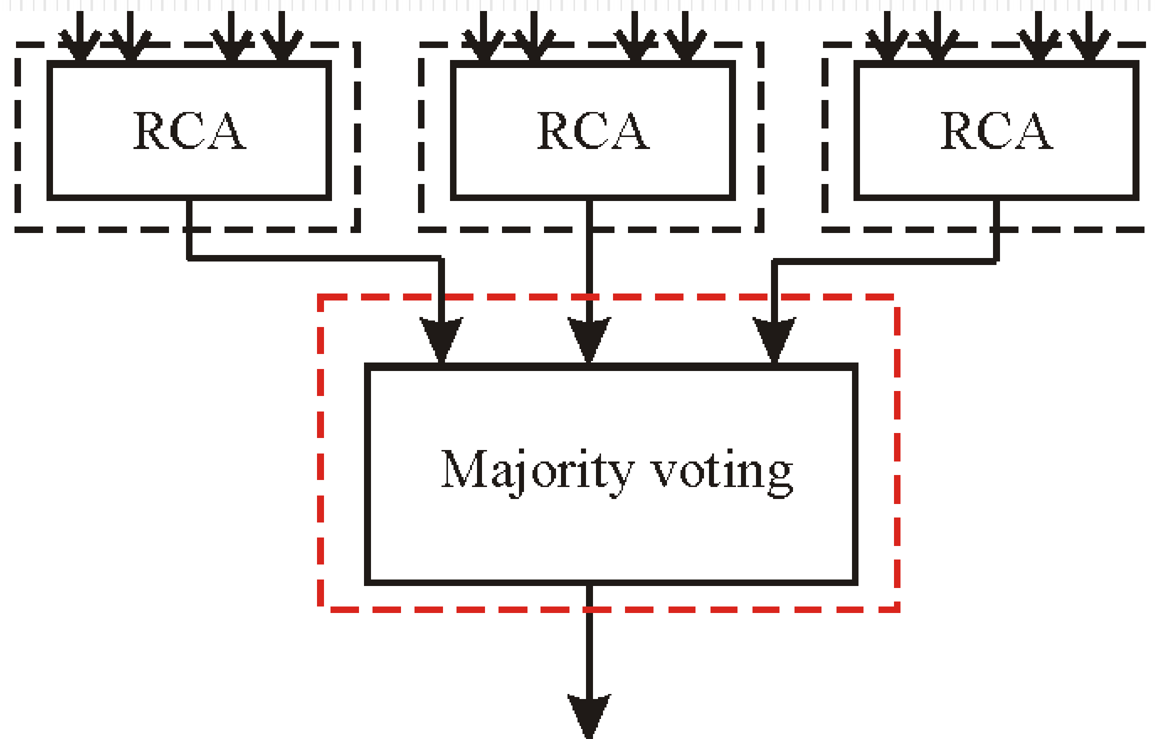
- Triple modular redundant adder with same supply voltage





Simulations Performed (5)

- Triple modular redundant adder with different supply voltages



Simulation Results (1)

Circuit under test	Error model	Vdd [V]	Temp [°C]	Delay [s]	Gate failure [%]	No of simulations	Probability of circuit failure [%]
Majority voting with equal paths	Model 2 - switching probability	0.35	50	5e-10	3.376	160000	5.1543
Majority voting with equal paths	Model 2 - switching probability	0.30	50	1.5e-09	1.6809	160000	2.7325
Majority voting with unequal paths	Model 2 - switching probability	0.35	50	upper path -> 5e-10; lower path -> 1.5e-09	upper path -> 3.376; lower path -> 0.2206	160000	7.0506
Majority voting with unequal paths	Model 2 - switching probability	0.25	50	upper path -> 5e-10; lower path -> 1.5e-09	Upper path -> 39.4691; lower path -> 9.5824	160000	42.3087

Simulation Results (2)

Circuit under test	Error model	Vdd [V]	Temp [^oC]	Delay [s]	Gate failure [%]	No of simulations	Probability of circuit failure [%]
2-bit adder with majority voting	Model 2 – switching probability	0.35	50	5e-10	3.376	80000	25.3675
2-bit adder with majority voting	Model 4 – input switching dependent	0.25	50	9e-10	22.8119 19.1749 14.0835 25.1028	32000	81.5687
2-bit adder with majority voting	Model 4 – input switching dependent	Adders-> 0.25; voters -> 0.35	50	9e-10	Voters: 0.0007 0.0006 0.5753 3.2338	32000	71.6718

Conclusions

- Four probabilistic fault models are proposed
- Verilog based SFI is performed for gate level descriptions
- Mutant based SFI is applied
- Five simulations campaigns have been performed
  - High flexibility of the proposed approach