# Defect Tolerance Inspired by Artificial Evolution

Asbjoern Djupdal and Pauline C. Haddow
IDI, NTNU, Norway
{djupdal,pauline}@idi.ntnu.no

## Abstract

*Defect densities in integrated circuits are expected to increase as the semiconductor feature size decreases. Some form of transistor level defect tolerance is, therefore, desirable to reduce this increasing production challenge. Series and parallel replication of transistors can be applied to a circuit for tolerating stuck-open and stuck-closed transistors. The circuit is, however, still damaged by gate/drain and gate/source shorts.*

*This paper applies an evolutionary algorithm to evolve a circuit tolerant to any single short between two transistor terminals. The evolved circuit is then analysed and a general defect tolerance technique is formed based on the evolved circuit. Applying the new technique to a circuit results in tolerance to any single stuck-open, stuck-closed, gate/drain shorted or gate/source shorted transistor. A Monte Carlo experiment compares the reliability of the new technique applied to a NAND gate with other redundant NAND gate implementations.*

## 1 Introduction and Motivation

As the semiconductor feature size decreases and the number of transistors on a single chip increases, one of the growing challenges facing the electronic design community is faulty behaviour [7]. If defects are expected to occur in a digital circuit, defect tolerance — the ability to function correctly in the presence of defective components, may be achieved by incorporating redundancy (additional resources) in some form. These additional resources may be in the form of additional hardware, in which case it is called *hardware redundancy* [8]. One form of hardware redundancy is *static* hardware redundancy. Static redundancy involves introducing extra components in a way that masks faults, thus without any need to detect and repair the defects.

There are several causes of defects, and defects may appear in different parts of an integrated circuit. This paper concentrates on transistor defects. A defective transistor may be modelled in several ways [2]. This paper considers stuck-open and stuck-closed, as well as any short between two of the terminals of a transistor. A *stuck-open* transistor is a transistor that is never conducting, no matter what gate voltage is applied. A *stuck-closed* transistor is, on the other hand, always conducting.

Redundant hardware may be introduced at different levels. At the system level, one of the most popular redundancy techniques is *Triple Modular Redundancy* (TMR) [9]. Three equal modules calculate the same function and a voter outputs the majority output. TMR may also be applied at a more fine grained level where each module is a smaller part of the complete system and where a cascade of TMR subsystems make up the complete system.

Defects may occur in any part of the system, including the voter. One disadvantage of TMR is the need for a perfect working voter or, if a perfect voter is not likely, triplicating the voter itself. The need for a voter makes TMR only practical when each of the modules are large compared to the voter. For TMR to function, the probability of having a functioning module must be more than 0.5. If the expected defect density of the circuit is high, the modules must be small to safeguard that the probability of working is more than 0.5. If the defect density is high enough, TMR is no longer suitable because each module must be so small that the voter will be dominating in terms of susceptibility to defects.

A gate level alternative to TMR is *interwoven logic* [12]. Interwoven logic involves constructing the network of logic gates in such a way such it masks defects. Defect masking is achieved by quadrupling every gate in the system and connecting the gates in a specific way so as to avoid the need for a voter. The lack of a voter makes interwoven logic useful at higher defect densities than TMR.

When the expected defect density is so high that it is probable that a large amount of the digital gates are defective, gate level techniques, like interwoven logic, fail to mask all the defects. This makes it useful to introduce redundancy at the transistor level i.e. introducing redundant transistors when implementing the basic logic gates. Redundancy at the transistor level would help the systems reliability by providing robust gates. To get even higher relia-

978-0-7695-3170-0/08 $25.00 © 2008 IEEE
DOI 10.1109/ISVLSI.2008.77

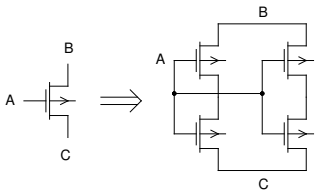



**Figure 1. Series and parallel technique**

bility, these robust gates could be applied together with gate level or system level redundancy techniques. A further benefit of introducing redundancy at the transistor level is to be able to exploit some non-digital properties of the transistor.

A general transistor level redundancy technique is shown in figure 1. Originally described by Moore and Shannon [11] for relays, the technique provides redundant transistors in series for tolerating stuck-closed defects and redundant transistors in parallel to tolerate stuck-open defects. Combining these, as in figure 1, results in tolerance to both stuck-open and stuck-closed faults.

When considering only stuck-open and stuck-closed defective transistors at high defect rates, the series–parallel technique results in more reliable circuits than TMR. Tolerance to shorts between the transistor gate and source or drain is, however, in general not tolerated by the series-parallel technique. A new transistor level redundancy technique is needed to tolerate gate shorts. Further, the series–parallel technique is area inefficient in that it quadruples the area needed for implementing a circuit.

The main objective of this paper is to provide tolerance to gate/drain and gate/source shorts through a new transistor level redundancy technique. When trying to find new redundancy techniques it is important to free oneself from the constraints brought upon us by thinking in the way of traditional redundancy techniques. The way one thinks when designing circuits is influenced by the way that one is taught electronics, designed electronics and the tools used in the design process. One way of freeing oneself from these human and design automated constraints is to search for ideas using some sort of heuristic search process. One such process is that of evolutionary algorithms [4]. The application of evolutionary algorithms to the design of hardware is termed evolvable hardware (EHW) [13]. Previous work [1] proved that it is possible to tune an evolutionary algorithm to create useful hardware redundancy at the transistor level.

This paper builds on the work in [1] and applies an evolutionary algorithm to find a circuit tolerant to gate/source and gate/drain shorts. The evolved circuit then forms the basis for a new redundancy technique. The paper starts in section 2 with a discussion of different aspects regarding evolution of transistor level redundancy. Section 3 explains the evolutionary experiment and provides an analysis of the best evolved circuit. A new redundancy technique based on the evolved circuit is presented in section 4 and several redundant NAND implementations are compared with respect to reliability in section 5. The paper concludes in section 6.

## 2 Evolving Redundant Circuits

The approach taken for evolution of redundant circuits in this paper follows the technique outlined in [1]. When fitness is to be evaluated, the circuit is tested repeatedly with different injected faults. Fault injection during fitness evaluation provides a means to calculate a reliability metric for the circuit. The quality of the evolved redundancy depends on the way fault injection is performed and how the reliability metric is included in the fitness function. This section presents the approach taken for the evolutionary experiment in section 3.

### 2.1 Measuring Functionality

The output of a circuit is defined to be *true* if having a value larger than $\frac{V_{dd}}{2}$. If the output is less than $\frac{V_{dd}}{2}$, it is defined to be *false*. One functionality metric, $f_{bool}$, simply states whether the circuit has correct Boolean output for all possible input combinations ($f_{bool} = 1$) or not ($f_{bool} = 0$).

$f_{bool}$ is an important functionality metric when reliability is to be determined. However, $f_{bool}$ might be too coarse grained when evolving towards a specific functionality. Fitness should therefore include a functionality metric that represents functionality in terms of how close the circuit's output voltage is to the desired output voltage. The main functionality metric for this paper, $f_{rms}$, is based on the Root-Mean-Square (RMS) error between the simulated output and the ideal output, for all $n$ possible input values $i$.

$$f_{rms} = 1 - \sqrt{\frac{\sum_i^n (sim(i) - ideal(i))^2}{n}} \qquad (1)$$

### 2.2 Fault Models

A *fault scenario* is one possible configuration of faulty transistors for a given circuit. A *fault model* dictates how the fault scenarios can be constructed and how probable the different fault scenarios are of occurring. Two fault models are considered in this work: *the transistor reliability model* and the *single fault model*.

In the transistor reliability model, each transistor has a certain probability of failing and each transistor fails independently of each other. If a fault scenario for the transistor reliability model is to be created, each transistor in the circuit is tested against a random number generator and selected to be faulty or not, based on a chosen fault rate.

In the single fault model, a circuit can have exactly one fault at any time and any single fault scenario is equally

probable. One and only one of the transistors are selected to fail for any given fault scenario.

### 2.3 Failing Transistors

A transistor may fail in several ways. In this paper, several types of transistor defects are considered: Stuck-open transistors are permanently off and are modelled by removing the transistor from the SPICE netlist. Stuck-closed transistors are permanently on and are modelled by shorting the source and drain with a $1\Omega$ resistor. In addition, there may be a short between gate/drain or gate/source which both are modelled with a $1\Omega$ resistor shorting the respective transistor terminals.

### 2.4 Measuring Reliability

A reliability metric indicates how well a circuit functions in the presence of faults. Reliability may be measured by testing the circuit against a number of randomly selected fault scenarios. The possible fault scenarios depend on the chosen fault model. The $R_{trad}$ metric, which is used in this paper, is the percentage of these tests where the circuit is fully functioning ($f_{bool} = 1$). When $R_{trad}$ is applied with the single fault model, it is named $R_{trad\_single}$. When applied with the transistor reliability model, the metric is named $R_{trad\_trans}$. $R_{trad\_single}$ may be calculated exactly by testing all possible single faults.

$R_{trad\_trans}$ may be estimated using a Monte Carlo simulation. A Monte Carlo simulation is too time consuming during evolution. Instead, equation (2) is applied during evolution to estimate $R_{trad\_trans}$, see [3]. $x0$ and $x1$ are the probabilities for having zero and one defective transistor, respectively, in a randomly chosen fault scenario. Equation (2) underestimates $R_{trad\_trans}$ because multiple defects are not considered.

$$R_{trad\_trans} = x0 \cdot f_{bool} + x1 \cdot R_{trad\_single} \quad (2)$$

### 2.5 Fitness Function

Earlier work on evolving transistor level redundancy [1] achieved best results when evolving the circuits in two phases. First generate redundancy using an $R_{trad\_single}$ based fitness function. As concluded in [3], an $R_{trad\_single}$ based fitness function is better suited for generating redundancy than an $R_{trad\_trans}$ based fitness function.

Phase one typically generates very bloated circuits. The evolved circuit is, therefore, optimised in a second evolutionary phase using an $R_{trad\_trans}$ based fitness function. $R_{trad\_trans}$ is much less forgiving for transistors without any real purpose.

The following two fitness functions, $f_1$ and $f_2$, are applied in this paper for phase one (equation (3)) and phase two (equation (4)):

$$f_1 = k_1 f_{rms} + k_2 \hat{f}_{rms} + k_3 R_{trad\_single} + k_4 f_{bool} \quad (3)$$

$$f_2 = k_1 f_{rms} + k_2 \hat{f}_{rms} + k_3 R_{trad\_trans} + k_4 f_{bool} \quad (4)$$

The first component, $f_{rms}$, is just for a single test with no defective transistors and is included to encourage high gain circuits. The second component, $\hat{f}_{rms}$, represents the average $f_{rms}$ after having tested the circuit for all single faults. The second component is included to make sure the circuit performs as well as possible, also when there are defective transistors. The third component is the reliability metric and the fourth component, $f_{bool}$ is to make sure a working circuit is always rewarded more than a non-working circuit.

## 3 Evolving a Circuit Tolerant to Gate Shorts

An evolutionary experiment is performed where evolution is steered towards creating redundancy tackling gate/drain and gate/source shorts. The purpose is to generate at least one example that illustrates how evolution tolerates a gate short in a circuit. This section explains the experimental setup and provides an analysis of the best circuit that resulted from the experiment.

### 3.1 Experimental Setup

To keep the size small and complexity (and thus the evolution time) low, the chosen target functionality for the evolutionary experiment is a digital inverter.

To make sure the evolved circuits are able to drive a representative load, the output of the circuit under test is connected to a chain of two inverters. SPICE Piece Wise Linear (PWL) voltage sources drive inverters that again drive the circuit inputs. Isolating the inputs with inverters is important because perfect voltage sources connected directly to the circuit inputs are not representative when an injected fault results in a short between input and either $V_{dd}$ or $V_{ss}$. When the functionality of a circuit is to be tested, all possible input transitions are tested in turn by setting the PWL voltage sources to correspond to the input transition to be tested. A transient analysis of the test setup is then performed in the BSD licensed SPICE simulator *ngspice* [5].

The circuit output is measured after inputs have been stable for 50ns. Circuit components allowed are nMOS and pMOS transistors. The V1.0 BPTM 22nm CMOS transistor models [14] are applied with allowed transistor sizes from 30nm to 1000nm. Supply voltage $V_{dd} = 1V$. Feedback loops are not allowed, but several transistors may drive the same wire.

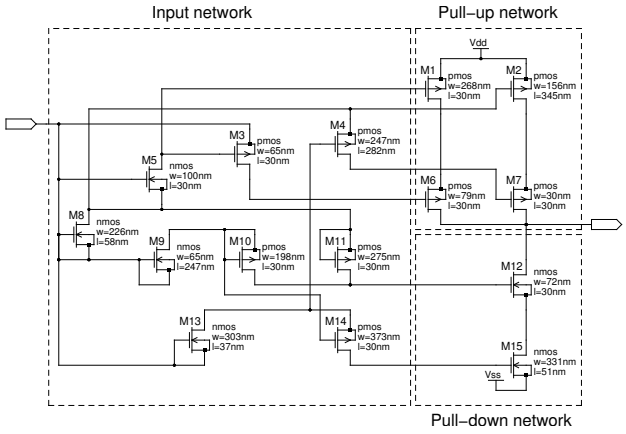



**Figure 2. Evolved defect tolerant inverter**

**Table 1. Characteristics of inverters**

| Property | Fig. 2 | Fig. 4 |
|---|---|---|
| size | 15 transistors | 24 transistors |
| $f_{rms}$ | 0.998929 | 0.997970 |
| $\hat{f}_{rms}$ | 0.978471 | 0.981983 |
| $R_{trad\_single}$ | 1.000000 | 1.000000 |
| $R_{trad\_trans} \vert R_t = 0.99$ | 0.965700 | 0.992000 |

A representation resembling Cartesian Genetic Programming [10] is applied, with the modification that in a gene, both inputs and the output of the component are explicitly defined, in addition to component type (nMOS or pMOS transistor) and transistor dimensions. A (1+4) evolutionary strategy is applied with mutation rate 0.1. Mutation is applied independently for each information block (inputs, output, type, sizes) inside each gene in the genome.

Evolution may create the circuit from a maximum of 50 transistors and 55 nets for each circuit. A net is an internal wire in the circuit, including inputs and output. Fault scenarios are created employing the following defect types: drain-source short (stuck-closed), gate-drain short and gate-source short. $R_{trad\_trans}$, a component in the fitness function for evolution phase two, can only be found given a certain transistor reliability. The transistor reliability applied in this experiment is 0.99. Coefficients used for the fitness functions for both evolution phases are $k_1 = k_2 = k_3 = 0.2$ and $k_4 = 0.4$. The high value for $k_4$ is there to favour fully functioning circuits over non-functioning circuits.

## 3.2 Analysis of Best Evolved Inverter

The best circuit found by evolution is shown in figure 2. The evolved inverter is fully functional and different metrics for the inverter are shown in table 1. As seen by the $R_{trad\_single}$ metric in table 1, the inverter is tolerant to all possible single gate/drain, gate/source and source/drain shorts on any transistor present in the circuit. As such, the circuit in figure 2 is suited for further analysis regarding how to tolerate gate/source and gate/drain shorts.

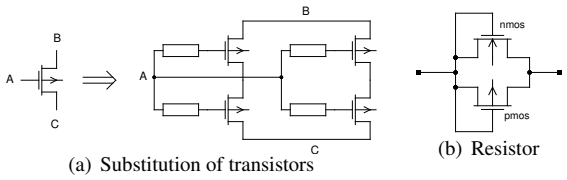
The standard CMOS inverter consists of a pull-up pMOS transistor and a pull-down nMOS transistor. The first step towards understanding the evolved circuit is to identify the corresponding pull-up and pull-down transistor networks. Transistors M1, M2, M6 and M7 in figure 2 represent the pull-up network, while transistors M12 and M15 represent the pull-down network.

The pull-up and pull-down structures are interesting by themselves. First, they show that evolution has introduced redundant transistors in series (M1/M6, M2/M7, M12/M15) to tolerate drain/source shorts (stuck-closed transistors). As the drain/source short defect was one of the defects injected during evolution, redundant transistors in series was expected.

However, evolution also introduced redundant transistors in parallel in the pull-up network (the M1–M6 chain is parallel to M2–M7), a structure known to tolerate stuck-open defective transistors. Stuck-open was *not* one of the defects injected during evolution, so why did evolution introduce these parallel structures? When there is a short between gate and source on transistor M1 or M2, the transistor is effectively stuck-open, resulting in the need for a parallel chain of transistors. The same reasoning applies for the pull-down network. However, instead of introducing parallelism in the pull-down network, evolution has relied on the output slowly discharging to the correct value. Unfortunately, such a solution is suboptimal because the delay will increase and the output will just barely reach a value less than $\frac{V_{dd}}{2}$.

The next step is to understand the purpose of the transistors in the input network i.e the transistors that connect the inverter input with the transistor gates in the pull-up and pull-down networks. None of these transistors are connected to $V_{dd}$ or $V_{ss}$, but are instead just passing on the inverter input. SPICE simulations showed that all nets in the input network are more or less degraded versions of the inverter input. It seems that evolution has tried to separate the inverter input from the pull-up and pull-down networks with a resistive circuit. To tolerate a short between, for example, the transistor M1 gate and source, the inverter input must be separated from the gate to avoid clamping the input to $V_{dd}$ and thus resulting in the inverter output stuck-at-0. If the input is separated from the shorted gate with a resistor, the result is a slightly degraded input signal whilst retaining correct output.

A resistor in a CMOS IC is usually formed by using an nMOS transistor with gate connected to $V_{dd}$. Evolution never introduced such resistors in the input network in figure 2 because those resistors are not themselves tolerant to gate/source and gate/drain shorts. Instead, evolution has created the input network without any connections to $V_{dd}$




(a) Substitution of transistors (b) Resistor

**Figure 3. New defect tolerance method**

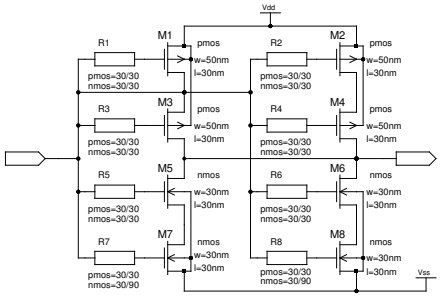
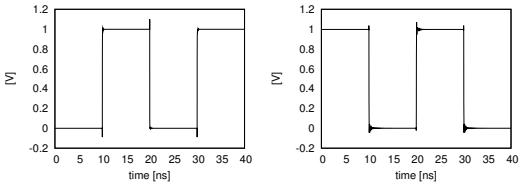
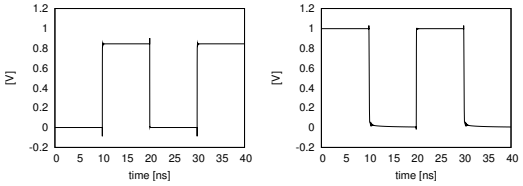



**Figure 4. Defect tolerant inverter**




(a) Input (b) Output

**Figure 5. Simulation of inverter in figure 4**




(a) Input (b) Output

**Figure 6. Simulation of inverter in figure 4 (M7 gate to $V_{ss}$ shorted)**

or $V_{ss}$, thus avoiding the problem of gate shorts in the input network.

## 4 A Generalised Defect Tolerance Technique

The analysis in section 3.2 can now be used as basis to form a new redundancy technique: (1) To allow for stuck-open and stuck-closed transistors, redundant transistors should be introduced to the pull-up and pull-down networks both in series and parallel, as in figure 1. (2) To tolerate gate/source and gate/drain shorts, the transistor gates in the pull-up and pull-down networks must be isolated from the inverter input using a defect tolerant resistor. A defect tolerant resistor can be formed using the transistor configuration in figure 3(b). Combining these two elements results in the technique summarised in figure 3.

To demonstrate how the technique summarised in figure 3 can be used to create a defect tolerant inverter, figure 4 shows the result after having applied the substitution in figure 3 to the standard CMOS inverter. The resistance of the resistors must be large enough to achieve isolation. For the circuit in figure 4, minimum sized transistor gates are suitable for most of the resistors, expect for R7 and R8 that should be sized for larger resistance to reduce the impact of a gate short to $V_{ss}$.

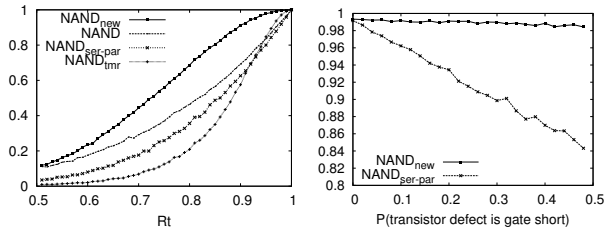
Figure 5 shows a SPICE simulation of the inverter in figure 4 when no transistors are defective. For this simulation and all later simulations in this paper, the V2.0 BPTM 22nm transistor models are applied. Figure 6 show the same simulations when transistor M7 has a short between gate and $V_{dd}$, one of the most damaging shorts. As seen in figure 6(a), the shorted gate pulls down the circuit input, but the isolating resistor R7 ensures this pull-down is so small that it does not result in incorrect output. Figure 6 shows only one fault scenario. Further simulations were performed to verify that the inverter is actually capable of tolerating all single transistor defects of types gate/source short, gate/drain short, stuck-open and stuck-closed. A summary of the characteristics of the gate is given in table 1.

## 5 Reliability Analysis

To investigate the quality of the proposed defect tolerance technique, the technique has been applied to a standard NAND gate. This new defect tolerant NAND-gate is called NAND$_{new}$ in this section and consists of 48 transistors. The reliability of NAND$_{new}$ is then compared with the reliability of three other NAND implementations: NAND (Standard 4 transistor CMOS NAND), NAND$_{ser-par}$ (a 16 transistor series–parallel version of CMOS NAND) and NAND$_{tmr}$ (a 48 transistor TMR version of NAND). To make the comparison with NAND$_{tmr}$ fair, the voter is triplicated to tolerate faults. Triplicated voters do, however, mean that the circuit will have three outputs instead of one and give NAND$_{tmr}$ the added advantage of only needing two of the three outputs to be correct. Each voter in NAND$_{tmr}$ consists of a mirrored adder [6] connected to an inverter.

### 5.1 Experimental Setup

The reliability metric $R_{trad\_trans}$ can be interpreted as the probability of having a correct output given a probabil-




(a) Comparing NANDs (b) Impact of gate-short

**Figure 7. Monte Carlo reliability simulations**

ity $R_t$ that a transistor is functional. $R_{trad\_trans}$ is therefore a more realistic reliability metric for a real circuit than $R_{trad\_single}$ and is thus chosen for the following experiments. It is further assumed that each transistor fails independently with a certain probability $(1 - R_t)$. When failing, the transistor may be stuck-open; stuck-closed; gate/source, gate/drain or source/drain shorted. Each defect type is equally probable.

### 5.2 Results

Monte Carlo simulations for different levels of transistor reliability $R_t$ have been performed on the different NAND implementations and the results are shown in figure 7(a). Each point in the graph represents the result after 10000 simulations with random fault scenarios for the given $R_t$.

As seen in figure 7(a), NAND$_{new}$ is the most reliable given the assumption that all defect types are equally probable. NAND$_{tmr}$ is slightly more reliable than plain NAND for $R_t \geq 0.95$. NAND$_{ser-par}$ is not suited at all for gate shorts.

The large number of extra transistors in the proposed redundancy technique might have a bad impact on the reliability if the probability of having shorted transistor gates is much less than stuck-open or stuck-closed defects. To investigate how the extra number of transistors affect reliability when the probability of having gate shorts is low, a new Monte Carlo experiment was performed. The reliability of NAND$_{new}$ is compared with NAND$_{ser-par}$ for different probabilities for having a gate/source or gate/drain short when a transistor is defective. $R_t = 0.96$. The results are shown in figure 7(b).

As seen in figure 7(b), NAND$_{new}$ is not affected by the defect type while the reliability of NAND$_{ser-par}$ decreases as the probability of having a gate short increases. When there are no gate shorts, the reliability of the two NAND implementations is about the same. As such, the large amount of area devoted to the isolation of inputs in the proposed technique does not seem to be of a disadvantage to reliability.

## 6 Conclusion and Further Work

An evolutionary experiment has been performed where an inverter was evolved for tolerance to shorted transistor terminals. Using an analysis of the evolved inverter as basis, a new redundancy technique was proposed. A NAND gate implemented applying the proposed technique is shown to perform well compared to other NAND gate implementations, with regard to reliability.

The proposed redundancy technique is an augmentation of the series-parallel technique and is therefore even less area efficient than the original series-parallel technique. Further work should look at more area efficient ways of achieving tolerance to gate shorts.

None of the experiments in this paper have considered shorts and opens in the metal layers implementing wiring. Defects in the metal layer is an important class of defects and although many metal defects will have an effect similar to one of the transistor defect types considered in this paper, further work should look at how defects in the metal layers affect reliability.

## References

[1] A. Djupdal and P. C. Haddow. Evolving efficient redundancy by exploiting the analogue nature of CMOS transistors. In *CIRAS*, 2007.

[2] J. Abraham and W. Fuchs. Fault and error models for VLSI. *Proc. IEEE*, 74(5):639–654, 1986.

[3] A. Djupdal and P. C. Haddow. Evolving redundant structures for reliable circuits – lessons learned. In *AHS*, pages 455–462, 2007.

[4] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[5] GEDA. Ngspice homepage. `http://ngspice.sourceforge.net/`, 2007.

[6] D. Hampel, K. J. Prost, and N. R. Scheinberg. Threshold logic using complementary MOS device, 1974. U.S. Patent 3 900 742.

[7] ITRS. Int. tech. roadmap for semicond. Technical report, ITRS, 2005.

[8] P. K. Lala. *Self-Checking and Fault Tolerant Digital Design*. Morgan Kaufmann, 2001.

[9] R. E. Lyons and W. Vanderkulk. The use of triple-modular redundancy to improve computer reliability. *IBM Journal*, pages 200–209, 1962.

[10] J. F. Miller and P. Thomson. Cartesian genetic programming. In *EuroGP*, pages 121–132, 2000.

[11] E. F. Moore and C. E. Shannon. Reliable circuits using less reliable relays. *J. Franklin Inst.*, pages 191–208, 1956.

[12] W. Pierce. *Failure-Tolerant Computer Design*. Academic Press, 1965.

[13] X. Yao and T. Higuchi. Promises and challenges of evolvable hardware. In *ICES*. Springer, 1996.

[14] W. Zhao and Y. Cao. New generation of predictive technology model for sub-45nm design exploration. In *ISQED*, pages 585–590, 2006.