2012-12-17

Noise-Immune Digital Circuit Design Based on Probabilistic Models

Tangim, Golam

UNIVERSITY OF CALGARY

Noise-Immune Digital Circuit Design Based on Probabilistic Models

by

Golam Tangim

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTERS OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA

December, 2012

Abstract

This thesis focuses on the logic design architectures that are inherently tolerant to noise. It aims at achieving noise-tolerance using Markov Random Field (MRF). This model considers inputs and outputs of a circuit as random variables and the function is evaluated via correct network states, which maximize the joint probability distribution of those variables. In implementation, this translates in creating a feedback that reinforces the correct states. Such circuits are simulated using the 16nm predictive CMOS technology model in SPICE.

This thesis proposes the implementation of MRF using Binary decision Diagrams (BDDs) where such circuits are realized on bi-directional switches. To accomplish the stability, we introduce BDD with feedback, called Cyclic BDD. The proposed designs are oriented at hardware implementation on the BDD based wrap-gate nanowire architectures. The comparison of the proposed design against conventional probabilistic models, in terms of performance, shows lesser power dissipation with the minimum area overhead.

Acknowledgments

I am extremely thankful to my supervisor Dr. Svetlana Yanushkevich, for the immense support and guidance. Her generous inspirations and thoughtful guidelines have helped me to become an independent thinker. I also thank my co-supervisor, Dr. V. P. Shmerko for his constructive suggestions and feedbacks throughout the years of my Master's.

Special thanks to the Information and Communication Technologies (ICT) Recruitment Scholarship and Alberta Innovates Technology Future (AITF) research grant for the financial support to conduct this research.

I'm grateful to all the people around me in Calgary, who never made me feel away from home, especially my friends.

Last but not least, a particular thanks to my parents and fiancee for their constant support and inspiration throughout the whole time.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nanotechnology is probably, as a phenomenon, the single most important new emerging force in technology.

-**Charlie Harris**, CEO, Harris & Harris Group

In the past decade, the semiconductor chip fabrication industry has been experiencing a steady downscaling of device dimension, from submicron to nano domain. At the same time, constant size downscaling and increased packing density cause unreliable or probabilistic behaviour of the device due to the intrinsic noise [1], defects [2], dopant concentration [3, 5] and tunneling effect [4]. The electro-chemo-mechanical phenomena in the nanoscale integration are accountable for the simultaneous increase in the device output noise and soft-errors [6].

Due to the unavoidable physical limit of semiconductors, the gate length is predicted to stop near $L_G \sim 5nm$ with the supply voltage being just $V_{dd} \sim 0.3V$ [7]. This Ultra Large Scale Integration (ULSI) with a minimum supply will unquestionably add up to the device noise and will cause its probabilistic behavior. As shown in Figure 1.1, a simple binary CMOS inverter, designed using *16nm Predictive Technology Model (PTM)*, will yield to noisy output and render ineffective with the introduction of $5db$ SNR noisy input signal (circuit driving voltage is $0.3V$). Note that this noise is an Additive White Gaussian noise (AWG) modeled using Gaussian process. Given the noise free signal y, the overall noisy signal can be is expressed as $Y = y + e$, where, e is signal noise.

In the sub threshold region, the intrinsic device noises dominate which are due to thermal noise, capacitive electrostatics, magnetic interferences and threshold variation.

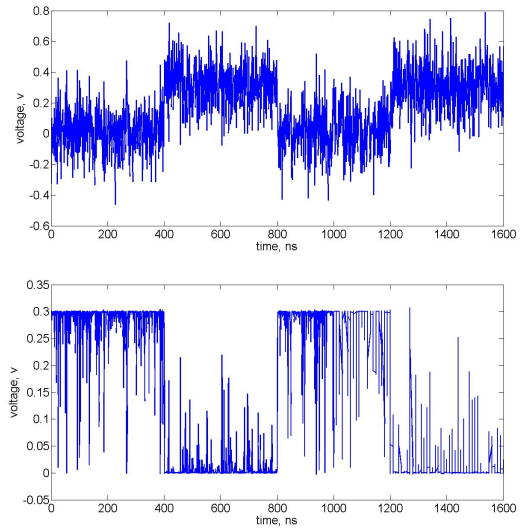


Figure 1.1: Noise condition of CMOS based Binary Inverter @ 16nm design for 5db SNR input

With low supply voltage, the reduced noise margin and non-uniform behavior of the circuits, the computation shifts from deterministic towards probabilistic paradigm.

The solution to the problem of noise margin and power consumption took an interesting turn as Palem's group [13, 14, 15] introduced the idea of considering the in-system noise as a resource, other than an obstacle. They also developed the algorithms to perform probabilistic switching and to use the switches for low power computation.

Examples of models, satisfying the requirements of dealing with noisy environment, include adaptive signal processing [16] and recurrent neural networks [17, 18]. Most of these models utilize statistical data [16, 19]. The previous states of a system are used to generate a statistically reliable, or correct, current states. A recent development of such model is based on the Markov Random Field (MRF). In hardware implementation, the state correction is accomplished by using a feedback. Therefore, the MRF-based models are combinational networks of logic gates with feedback [21], which can be re-structured as latches [22].

The silicon industry has found a way out of the physical limitation problem, by developing the following technologies: Carbon Nanotube Field-Effect-Transistors (CNFET) [28], Quantum Cellular Automata (QCA) [29], Single Electron Tunneling (SET) devices [30], Molecular Devices [31], Silicon Nanowire [32] and Wrap Gate Nanowires [33]. These technologies not only provide the way to overcome the barrier of the semiconductor physical limit, but also are stochastic in nature. The inherent stochastic behavior of these techniques makes them suitable for the probabilistic computing approach [34]. Even though the development of small gate level circuits is reported, large scale circuits are yet to be designed using these techniques. The most advanced circuits in this regard are based on CNFET and Wrap-Gate Nanowire based designs.

This dissertation aims at providing a noise immune circuit design technique that can be readily implementable using emerging technology. For this purpose, the inherently probabilistic Markov Random Field (MRF) model is used. The circuit design strategy is developed around the idea of Binary Decision Diagram (BDD) mapped onto the wrap gate nanowire structure reported in [33]. The thesis also proposes a probabilistic circuit design approach based on MRF for the design of non-binary or multivalued noise-immune circuits.

1.1 Thesis Contribution

The key contributions of this thesis are as follows.

Investigation of noise scenarios in nano-scaled CMOS. This thesis investigates the nano-scaled CMOS designs of elementary gates at the 16nm technology on SPICE. For the simulation, the Predictive Technology Model (PTM) model from Berkeley is used (available at http://ptm.asu.edu/). The elementary universal gates (NAND, NOR and NOT) were designed and simulated in the presence of

noisy input, and several parameters (SNR, BER and KLD) were introduced for the evaluation of gate reliability.

Introduction of novel MRF based circuit design framework. The dissertation reviews the idea of a probabilistic framework based on Markov Random Field (MRF) [46], first proposed by Bahar et. al. [56], and proposes a novel design methodology. The gates, designed based on MRF, are made stable using a reinforcer, feeding the correct value back to input from output. The thesis implements the proposed design on the CMOS based transistors, and compares the reliability of the elementary level gates with that of the other designs.

Implementation of MRF based circuit on Decision Diagram. This thesis proposes the complete novel idea of Cyclic-Binary Decision Diagram (Cyclic-BDD), using multiplexers. Mapping the MRF model onto a BDD is a precursor towards implementation on wrap-gate nano-wire based structure [33]. Not only the elementary gates, but also the large scale circuits, such as the MNCN'91 benchmarks are modeled on SPICE.

Extension of probabilistic circuit design towards the multivalued domain. The thesis extends the MRF based idea of binary circuit design to the multivalued domain. The dissertation introduces Arithmetic transformation method [48] for the calculation of the output probability of the multivalued gates (e.g. ternary gate). The proposed ternary logic gates are implemented, and the reliability is compared against the conventional CMOS based multivalued gate design.

The BDD implementation and its noise immunity, presented in this thesis, is not significantly different from MRF on CMOS [47], but the main difference is that the used data structure (BDD) is alternative to traditional, which allows to create noise-robust BDD nanodevices such as wrap-gate nanowire circuits.

The research presented in this dissertation has appeared in publications [1-7]. Sections 2.3 and 2.4 in Chapter 2, describe the spectral transformation methods, proposed for the calculation of the clique energy for Markov Random Field model, it is published in papers [2] and [7]. The Cyclic BDD based architecture introduced in Chapter 3 was published in [5] and [6]. Chapter 4 is covered by publication [4]. The probabilistic nature of multivalued (ternary) gates, introduced in the Chapter 5 is published in [2] and [7]. The rest of Chapter 5 is covered by publication [3]. The publication [1] also covers Chapter 3 and 4.

[1] S. Yanushkevich, S. Kasai, G.Tangim, T. Mohamed, V. Shmerko, Noise-immune and Fault-tolerant Models for Nanoscale Logic Devices, in press, *Morgan & Claypool Publishers*, USA, January 2013

[2] G. Tangim, S. N. Yanushkevich, S. E. Lyshevski, "Noise Immune Digital IC Design based on Shared Cyclic BDDs", *50th ACM/IEEE Design, Automation Conference (DAC)*, Austin, Texas, June 2013 (submitted)

[3] G.Tangim, S.N. Yanushkevich, S. Kasai, V.P. Shmerko, Multivalued Fault-tolerant Logic Design Using Cyclic Decision Diagram Techniques, *IEEE 43rd International Symposium on Multiple-Valued Logic (ISMVL)*, Toyama, Japan, May 2013 (submitted)

[4] G. Tangim, T. Mohamed, S. N. Yanushkevich, S. E. Lyshevski, "Comparison of Noise Tolerant Architectures of Logic Gates for Nanoscaled CMOS" *2nd International conference on High Performance Computing (HPC)*, vol. 2, pp.66-73, October, 2012

[5] S.N. Yanushkevich, G.Tangim, S. Kasai, S. E. Lyshevsky, V.P. Shmerko, "Design of nanoelectronic ICs: Noise-tolerant logic based on cyclic BDD," *Nanotechnology (IEEE-NANO), 2012 12th IEEE Conference on* , vol., no., pp.1-5, 20-23 Aug. 2012

[6] S. N. Yanushkevich, A. H. Tran, G. Tangim, V. P. Shmerko, E. N. Zaitseva and V. Levashenko, The EXOR Gate Under Uncertainty: A Case Study, *Facta Univ. Ser.: Elec.*

Energ., vol. 24, No. 3, December 2011, pp. 451-482

[7] S. N. Yanushkevich, V. P. Shmerko, A. Abbasinasab, G. Tangim; , "Probabilistic Gates and Belief Networks", *20th International Workshop on Post-Binary ULSI Systems*, Tuusula, Finland, May 2011.

1.2 Thesis Organization

This dissertation consists of the following chapters.

Chapter 1: Introduction, provides an introduction on the probabilistic computing method in nanoscale with the motivation and history of the works. It also discusses the contributions of this research and outlines the organization of the thesis.

Chapter 2: Background and Theoretical Basis. The basics of switching theory, noise scenario in nanoscale circuits and the corresponding architectural design methodologies for reliable circuit design are covered in this chapter. This chapter presents a detailed discussion on Markov Random Field (MRF) for the proposed probabilistic circuit design. The existing design methods are discussed along with the spectral transformation methods such as the Arithmetic, Haar and Walsh transformation.

Chapter 3: Nanoscaled IC Design based on Markov Random Field. The existing design techniques related to probabilistic design, are considered and a new design technique of reinforcement is introduced. The idea of Cyclic Binary Decision Diagram (Cyclic BDD) is introduced, and theoretical explanation with practical simulation results are provided.

Chapter 4: Probabilistic Model of Logic Networks Using Shared Cyclic BDDs. As a new idea of probabilistic gate design is introduced in the previous chapter, this chapter focuses on building larger scale circuits, other than elementary logic gates.

For the design of large scale reliable circuits with the minimum power consumption, the idea of Shared Cyclic Binary Decision Diagram (Shared Cyclic BDD) is introduced. This chapter outlines the design methodology of a binary 2-bit adder circuit, along with the MCNC'91 benchmarks and also does a systematic comparison on reliability and other circuit performance parameters against the existing modeling techniques.

Chapter 5: Noise Tolerant Modeling of Multivalued Gates based on MRF. This chapter extends the idea of reliable circuit design to the multivalued domain. The energy function is extended to design the MRF based reliable Ternary Inverter. The MRF based ternary inverter is compared with the conventional CMOS based ternary inverter, in terms of the performance.

Chapter 6: Conclusion. This chapter concludes the thesis with the summary of the contributions and future scopes of related research and development.

Chapter 2

Background and Theoretical Basis

Deep submicron and nano-scale circuits are highly susceptible to noise. As described in the previous chapter, the intrinsic probabilistic behavior and reduced noise margin due to lower supply voltage causes the output errors. This chapter discusses about the noise influence on errors in digital circuits. The approach to tolerate noise, including one the thesis focuses on, Markov Random Field (MRF), are introduced.

2.1 Switching theory in logic networks

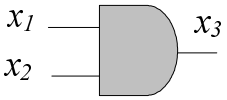
Switching algebra is the theoretical foundation of circuit design and decision diagram technique. The Boolean algebra is denoted by the set $\beta = \{\alpha; +, \times; \bar{}\,; 0, 1\}$, where α is the set of input variables. It also includes the two output elements 0 and 1, two binary operations $+$, and $\times$ and an unary operation $\bar{}$ [82].

A special case of Boolean algebra with two elements is called *switching algebra*. In switching algebra, an n-variable function $f : \beta^n \rightarrow \beta$ is called a *switching function*. A switching function of n variables is a discrete function where

$$f : \{0, 1\}^n \rightarrow \{0, 1\}$$

where $\{0, 1\}^n$ denotes the n-fold Cartesian product of 0 and 1 that is the set of all binary *n-tuples*. Each *n-tuple*, mapped to 1 by the function, is a minterm of the function. For example, the figure 2.1(a) shows a two input binary AND gate with its truth table 2.1(b). Being a binary system, the inputs can take on values 0 and 1. The possible input combinations will be a set of $2^2 = 4$ different tuples, that is {00, 01, 10, 11}. Each of the

input combination can be mapped to the range of the output being {0, 1}, as shown in the truth table. Given x_1 and x_2 being the inputs and output being x_3, x_1x_2 is the only minterm of the AND gate.



(a)

x_1	x_2	x_3
0	0	0
0	1	0
1	0	0
1	1	1

(b)

Figure 2.1: Binary two input AND gate (a) and its truth table (b).

2.2 Noise in nanoscale circuits

With the downscaling of device dimension towards deep submicron and nano scale, different sorts of noises and errors occur in the circuits. To tackle the noise scenario in nanoscaled digital circuits, a proper model of noise and error is needed with particular design approach based on the models. This section describes the noise and error sources at nanoscale, along with the available architectural noise-tolerant design methodologies.

2.2.1 Sources of noise and error

In nanoscale devices, the errors can occur for a number of reasons, starting from fabrication defects to intrinsic device characteristics. A few of the sources that are mostly responsible for circuit failures are given below.

Thermal noise. As the nanoscaled circuits usually operate only at the very low temperatures, the slight change in temperature can cause the circuit output deviations. The thermal variation during operation is responsible for radical rise in the output variance [50].

Leakage noise. This noise became an important contributor as soon as the device gate length scaled below 0.25μ. Even in the device OFF stage, the current leaks through the circuit to ground, degrading the voltage at the nodes. This creates the attenuation in the output voltage level of the circuit. It is considered to be a major cause of nano-circuit noise [43].

Supply noise. As the capacitors and resistors of the circuit demand equal current through the power grid, the components further away from the source get a reduced supply, which results in timing problem and functional failure. Furthermore, lowering the supply voltage in modern technologies causes a reduction in noise margin, and in proportion to that, a reduction in signal-to-noise ratio.

Crosstalk noise. As the number of devices inside the chip and closely laid interconnections increase, the capacitive coupling among the neighboring wires increases. Being in a close physical proximity, the grid switching characteristics get affected due to the raised capacitive coupling [44].

Miscellaneous. Other than the device-level noises, other deviations exist at molecular or the atomic level. Tunneling noise, shot noise [45], threshold variation [21] and hot electron effect are some of the sources of noise in the nanoscaled circuits. They make the circuits more vulnerable to errors, compared to the microelectronic technology.

2.2.2 Architectural model for error and noise tolerance

In order to combat noise and make noise-tolerant circuit design, an acceptable modeling technique is required. The reliability of circuits can be achieved through architectural level design [12], device-level methodology [11], error correcting codes for fault detection and correction [35], system learning using neuromorphic models [18] or Bayesian learning. The architectural reliability mainly focuses on special circuit configuration and design

for fault and noise tolerance, as well as on the transistor level. The recent techniques developed to achieve circuit reliability are listed below.

Circuit redundancy is one the mostly used circuit design schemes for reliability, where the circuit elements are replicated N times and passed through a majority gate for the final decision. Here, N is generally an odd number. This technique is called *N-Tuple Modular Redundancy*. A popular variation of this technique is called *Triple Modular Redundancy (TMR)* [10]where, $N = 3$. Other variations include *Cascaded Triple Modular Redundancy* [37], *Triple Interwoven Redundancy* [38] etc. Another circuit redundancy technique is the *Error Correcting Codes* [35], where the error is detected at the receiving end, and corrected accordingly.

Markov Random Field (MRF) proposes a probabilistic method of circuit design. Rather than traditional circuits, the deployed elements are random in nature, and use reinforcement to only yield output for the correct states [43, 47]. The output state combinations, having the highest probability of occurrence, are thought to be the correct logic levels.

Crossbar architecture is another reliable design technique proposed by Chen *et al.* [40]. This is an online testing method, where the defective devices and the interconnections systematically go through a Triple Modular Redundancy testing.

Ensemble Dependent Matrix (EDM) considers alternate elements of the circuit with the same functionality [41]. In the process, the element with the lowest Bit Error Rate (BER) is chosen among the same functional blocks to produce the least BER for the whole circuit.

As this dissertation focuses on the reliable circuit design on the architectural level, based on the principles of Markov Random Field, the next section of this chapter presents

an elaborate discussion on the theory and implementation methodology using Markov Random Field.

2.3 Markov Random Field (MRF)

The basic definitions of Markov Random Field and its application for implementation of logic functions are discussed in this section.

2.3.1 Basic definitions

The fundamental definition for the MRF is based on [49], and the applied results are reported in [47, 22, 66]. In terminology, we follow contemporary textbooks on probability and statistic for engineers, such as [81].

Let $X = (x_1, x_2, \ldots, x_n)$ be a set of **random** variables. Each variable, x_i, $i = 1, 2, \ldots n$, takes a finite set of states, for example, two states $x_i \in \{0, 1\}$ (binary variable), or m states $x_i \in \{0, 1, \ldots, m-1\}$ (m-valued variable). Consider the undirected graph $G(V, E)$, where elements of V and E are called **nodes** and **links**, respectively. In $G(V, E)$, each node $v_i \in V$ corresponds to a random variable x_i. A link in E between nodes v and u is specified by an unordered pair (u, v), where v and u are called **neighbors**. This graph represents a logic network defined on a library of gates. The interconnected neighbors in a graph produce a **clique**.

Formal notion of the MRF model. The MRF model of a logic gate is acceptable for hardware implementation and also convenient for probabilistic analysis of noisy logic gates and networks [56, 47, 22, 66]. The MRF is characterized by a *Gibbs joint probability distribution*:

$$p(X = x) = \frac{1}{Z} \exp\left(\frac{E(x)}{kT}\right) \tag{2.1}$$

where Z is a scaling factor (to normalize the total probability to 1), $E(x)$ is an *energy* function, and kT is a control parameter.

Logic function embedding. A logic function is embedded into the MRF model using an energy function. First, this function must be represented in the form of its *compatibility truth table* (instead of traditional truth table). The compatibility truth table represents a compatibility function defined as any algebraic form (instead of logic one), for example, arithmetic, Walsh, or Haar form [74]. Given a logic gate, its compatibility function is the energy function.

The MRF-based software models. Probabilistic analysis of logic gates, or networks of gates, is provided by computing a joint probability distribution, or marginal distributions (2.1). No strong limitations on the model for such analysis exist. However, the arithmetic sum of the function's minterms is preferable for the hardware implementation.

The MRF-based hardware models. For hardware implementation, the maximum-likelihood computing strategy is realized using feedback circuits. The energy function of a logic is defined as an arithmetic sum of its minterms. The search for the correct solution is implemented by reinforcement. This idea was implemented in [47] through designing a device called a *reinforcer*, using a logic array with multiple feedbacks. Note that the reinforcer is simpler than a neuromorphic model, which is based on threshold elements [18]. Hence, there is no notion of a correct logic state in the MRF model, as the states are determined using the Gibbs probability distribution of signals. The correct states are those, which maximize the joint probability distribution of random variables. The process is thoroughly explained below.

Definition 1. *Given the compatibility truth vector* $\mathbf{U}$ *of a Boolean function* f *of* n *variables, the* ***arithmetic spectrum*** *of zero polarity,* $A = (a_1, a_2, \ldots, a_n)$ *is calculated using*

the arithmetic transform [74]:

$$\mathbf{A} \;=\; \mathbf{A}_{2^n} \cdot \mathbf{U}, \tag{2.2}$$

where matrix $\mathbf{A}_{2^n}$ *is formed as follows*

$$\mathbf{A}_{2^n} \;=\; \bigotimes_{j=1}^{n} \mathbf{A}_{2^j}, \quad \mathbf{A}_{2^1} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

Definition 2. ***Arithmetic form*** *of* r*-input,* $x_1, x_2, \ldots, x_r$*, logic function* f*, given by spectral coefficients,* $a_1, a_2, \ldots, a_n$ *is defined by the polynomial:*

$$f \;=\; \sum_{i=0}^{2^r-1} a_i \cdot (x_1^{i_1} \cdots \; x_r^{i_r}), \tag{2.3}$$

where i_j *is the* j*-th bit* $1, 2, \ldots, r$*, in the binary representation of the index* $i = i_1 i_2 \ldots i_r$*;* $x_j^{i_j} = 1$ *if* $i_j = 0$*, and* $x_j^{i_j} = x_j$ *if* $i_j = 1$*.*

Example 1. *Arithmetic spectrum (representation) of a 3-input logic gate (*$n = 3$*), implementing a function* $f(x_1, x_2, x_3)$*, is defined by Equation (2.3) as follows:* $f = a_0 + a_1 x_3 + a_2 x_2 + a_3(x_2 x_3) + a_4 x_1 + a_5(x_1 x_3) + a_6(x_1 x_2) + a_7(x_1 x_2 x_3)$*.*

Example 2. *Given the compatibility truth vector* $\mathbf{U}$ *of the 2-input AND gate,* $f = x_1 x_2$*, a clique potential is calculated using arithmetic transform (2.2):*

$$\mathbf{A} = \mathbf{A}_{2^3} \cdot \mathbf{U} = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ \hline -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{array}\right] \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{array}\right] = \left[\begin{array}{r} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 2 \end{array}\right]$$

Thus, the vector of spectral coefficients of the clique potential is $\boldsymbol{A} = [1\ -1\ 0\ 0\ 0\ 0\ -1\ 2]^T$*. The energy function in algebraic form is defined by Equation (2.3):*

$$E(v) = 1 - v_3 - v_1 v_2 + 2 v_1 v_2 v_3 \tag{2.4}$$

2.3.2 Algorithm for synthesis of MRF models

Consider a 2-input logic gate L of a function f, which is described by a single clique $c = \{v_1, v_2, v_3\}$ (complete graph) with energy function $E(x) = V_c(x)$. The algorithm for designing the MRF model of this gate is as follows.

```
Algorithm 1:  The MRF model design for a 2-input logic gate
Input: (a) Graph model {v1, v2, v3} (complete graph) and (b)
    function of gate L
Output: An MRF model of L gate
1.  Form the compatibility truth table
2.  Calculate the energy function, E(v), using Fourier-like
    transform, in particular, arithmetic transform (2.2).
3.  Specify Gibbs distribution (2.1) by substituting energy
    function, E(v), into it.
4.  Calculate the marginal probability distribution of the
    output node.
```

Designing the MRF model for the AND gate is illustrated by the following example.

Example 3. *Let L be the two-input binary AND gate. Its graphical model is a complete graph with three nodes, v_1, v_2 and v_3 where v_1 and v_2 are inputs and v_3 is the output that complete the clique $\{v_1, v_2, v_3\}$. Its compatibility truth table (step 1) is calculated as follows:*

V_1	V_2	V_3	U	*Comment*
0	*0*	*0*	*1*	
0	*0*	*1*	*0*	*Undesirable*
0	*1*	*0*	*1*	
0	*1*	*1*	*0*	*Undesirable*
1	*0*	*0*	*1*	
1	*0*	*1*	*0*	*Undesirable*
1	*1*	*0*	*0*	*Undesirable*
1	*1*	*1*	*1*	

Clique potential, calculated in Example 2, is equal to energy function, $E(v)$. Gibbs distribution is specified by substituting $E(v)$ (2.4) into (2.1):

$$p(v_1, v_2, v_3) = \frac{1}{Z} \exp\left(\frac{1 - v_3 - v_1 v_2 + 2 v_1 v_2 v_3}{kT}\right) \tag{2.5}$$

Marginal distributions are obtained by summing over all possible states of v_1:

$$\begin{aligned} p(v_2, v_3) &= \frac{1}{Z_1} \sum_{v_1 \in 0,1} \exp\left(\frac{1 - v_3 - v_1 v_2 + 2 v_1 v_2 v_3}{kT}\right) \\ &= \frac{1}{Z_1}\left[\exp\left(\frac{1 - v_3}{kT}\right) + \exp\left(\frac{1 - v_3 - v_2 + 2 v_2 v_3}{kT}\right)\right] \end{aligned}$$

Finally, marginal probability distribution of v_3 is obtained by summing over all possible states of v_2:

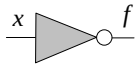
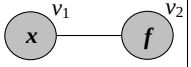
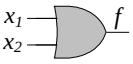
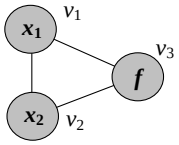
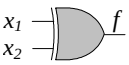
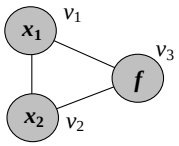
$$\begin{aligned} p(v_3) &= \frac{1}{Z_2} \sum_{v_2 \in 0,1} \left[\exp\left(\frac{1 - v_3}{kT}\right) + \exp\left(\frac{1 - v_3 - v_2 + 2 v_2 v_3}{kT}\right)\right] \\ &= \frac{1}{Z_2} \left[3 \exp\left(\frac{1 - v_3}{kT}\right) + \exp\left(\frac{v_3}{kT}\right)\right] \end{aligned} \tag{2.6}$$

Example 4. *The MRF design for binary NOT, OR and EXOR gate is shown in Table 2.1. Assume $p(v_1 = 1) = p(v_1 = 0)$ for arithmetic representation of clique potential.*

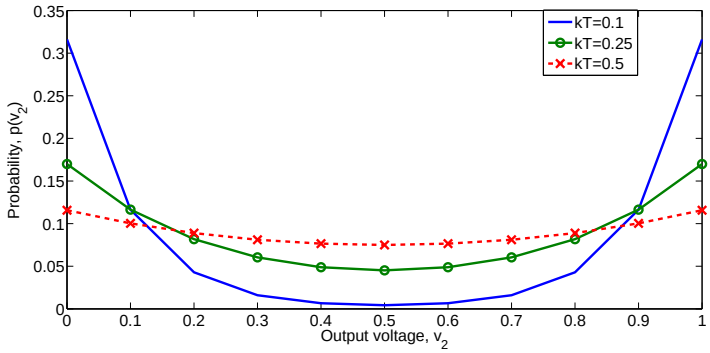
Then, after marginalization with respect to v_2, MRF model is represented as follows for a NOT gate with input v_1 and output v_2:

$$p(v_2) = \frac{\exp(\frac{v_2}{kT}) + \exp(\frac{1-v_2}{kT})}{2(1+\exp(\frac{1}{kT}))} \tag{2.7}$$

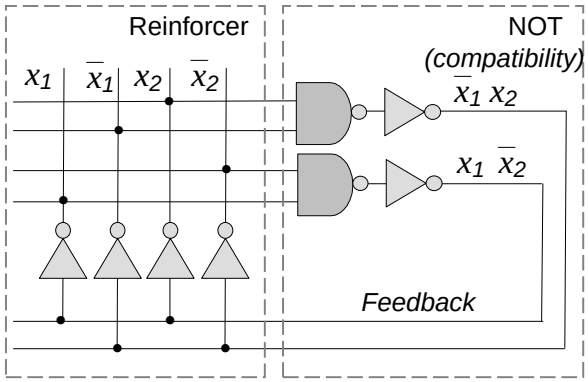
Table 2.1: Components of the MRF model of binary gates.

Gate	Graph model (clique)	Compatibility truth table	Energy function (clique) representation	
			Arithmetic spectrum	Walsh spectrum
 $f = \overline{x}$		v_1 v_2 U 0 0 0 **0 1 1** **1 0 1** 1 1 0	$\mathbf{U} = [0\ 1\ 1\ 0]^T$ $\mathbf{A} = [0\ 1\ 1\ -2]^T$ $E(v) = v_1 + v_2 - 2v_1v_2$	$\mathbf{U} = [0\ 1\ 1\ 0]^T$ $\mathbf{W} = {}^1/_4 \times [2\ 0\ 0\ -2]^T$ $E(v) = {}^1/_4 \times [2 - 2(-1)^{v_1+v_2}]$
 $f = x_1 \vee x_2$		v_1 v_2 v_3 U **0 0 0 1** 0 0 1 0 0 1 0 0 **0 1 1 1** 1 0 0 0 **1 0 1 1** 1 1 0 0 **1 1 1 1**	$\mathbf{U} = [1\ 0\ 0\ 1\ 0\ 1\ 0\ 1]^T$ $\mathbf{A} = [1\ -1\ -1\ 2\ -1\ 2\ 1\ -2]^T$ $E(v) = 1 - v_1 - v_2 - v_3 + v_1v_2 + 2v_1v_3 + 2v_2v_3 - 2v_1v_2v_3$	$\mathbf{U} = [1\ 0\ 0\ 1\ 0\ 1\ 0\ 1]^T$ $\mathbf{W} = {}^1/_8 \times [4\ -2\ 0\ 2\ 0\ 2\ 0\ 2]^T$ $E(v) = {}^1/_8 \times [4 - 2(-1)^{v_3} + 2(-1)^{v_2+v_3} + 2(-1)^{v_1+v_3} + 2(-1)^{v_1+v_2+v_3}]$
 $f = x_1 \oplus x_2$		v_1 v_2 v_3 U **0 0 0 1** 0 0 1 0 0 1 0 0 **0 1 1 1** 1 0 0 0 **1 0 1 1** **1 1 0 1** 1 1 1 0	$\mathbf{U} = [1\ 0\ 0\ 1\ 0\ 1\ 1\ 0]^T$ $\mathbf{A} = [1\ -1\ -1\ 2\ -1\ 2\ 2\ -4]^T$ $E(v) = 1 - v_1 - v_2 - v_3 + 2v_1v_2 + 2v_1v_3 + 2v_2v_3 - 4v_1v_2v_3$	$\mathbf{U} = [1\ 0\ 0\ 1\ 0\ 1\ 1\ 0]^T$ $\mathbf{W} = {}^1/_8 \times [4\ 0\ 0\ 0\ 0\ 0\ 0\ 4]^T$ $E(v) = {}^1/_8 \times [4 + 4(-1)^{v_1+v_2+v_3}]$

The output probability distribution of a Binary Inverter along with the MRF based circuit implementation from [47], is shown in the Figure 2.2. The equal concentration of output probability towards the logic 0 and 1 are equally distributed for the inputs, being equally likely to be 0 and 1.



(*a*)



(*b*)

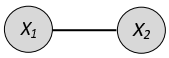
Figure 2.2: MRF model of a binary inverter: Gibbs distribution of the output [47] (a), and its implementation using feedback [56] (b).

2.4 Embedding logic function using MRF

The MRF model of a logic gate is acceptable for hardware implementation [47, 22, 66], and useful for probabilistic analysis [56].To embed the logic function into the MRF model, (a)the logic function must be presented in the form of compatibility truth table (instead of traditional truth table), and (b) the compatibility function must be represented by the arithmetic expression (instead of logic one). In terms of the MRF model, this expression is called the *energy function*, $E(x)$.

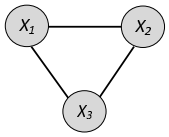
The graphical representation of the MRF for the logic function NOT ($x_2 = \overline{x_1}$) and

EXOR function ($x_3 = x_1 \oplus x_2$), along with their compatibility truth tables, are shown in Fig. 2.3, where the valid state combinations correspond to the value 1 of the compatibility function, (**U**), and to the value 0, otherwise.



x_1	x_2	**U**
0	0	0
0	1	1
1	0	1
1	1	0

(a)



x_1	x_2	x_3	**U**
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

(b)

Figure 2.3: MRF network and compatibility truth table for NOT gate (a) and two-input EXOR gate (b).

In the MRF model, the compatibility function U corresponds to the real valued function called energy function, $E(x)$; it is defined as an arithmetic sum of the valid minterms, also called an arithmetic polynomial [48]. For example, the compatibility function for NOT and EXOR (Fig.2.3) is $U_{NOT} = x_1\overline{x}_2 + \overline{x}_1 x_2$ and $U_{EXOR} = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1 x_2 x_3 + x_1\overline{x}_2 x_3 + x_1 x_2\overline{x}_3$, respectively. In general form, the compatibility function, U, of an n-input function $f = x_{n+1}$ of n variables $x_1, .., x_n$ can be written as follows:

$$\begin{aligned} U &= u_{0...00}x_1^0...x_n^0x_{n+1}^0 + u_{0...01}x_1^0...x_n^0x_{n+1}^1 + \cdots + u_{1...11}x_1^1...x_n^1x_{n+1}^1 \\ &= \texttt{SumMaxterm} \times x_{n+1}^0 + \texttt{SumMinterm} \times x_{n+1}^1 \end{aligned}$$

where $u_{0...00}, ...u_{1...11}$ are the components of the truth table of U, $x^0 = \overline{x}$ and $x^1 = x$, `SumMaxterm` and `SumMinterm` are the arithmetic sums of the maxterms (the variable assignments at the function value 0) and minterms (the variable assignments at the function value 1), respectively. For example, for NOT function, $U_{NOT} = x_1\overline{x}_2 + \overline{x}_1 x_2 =$

$x_1\overline{f} + \overline{x}_1 f$ (Fig. 2.3a);

By analogy, for EXOR function (Fig. 2.3b),

$$\begin{aligned} U_{EXOR} &= (\overline{x}_1\overline{x}_2 + x_1x_2)\overline{x}_3 + (\overline{x}_1x_2 + x_1\overline{x}_2)x_3 \\ &= (\overline{x}_1\overline{x}_2 \vee x_1x_2)\overline{x}_3 + (\overline{x}_1x_2 \vee x_1\overline{x}_2)x_3 = (\overline{x_1 \oplus x_2})\overline{f} + (x_1 \oplus x_2)f \end{aligned}$$

Therefore, the compatibility function preserves the states of the MRF via this twofold representation: a combination of maxterms (the minterms, corresponding to the values 0 of the function f), which are true when the function is false (hence, the product of this sum and $\overline{f}$), and the minterms, which are true when the function f is true. This corresponds to an implementation, which is a circuit with a feedback: both the variable-dependent implementation of the function and its complement are reinforced via a feedback. It preserves the previous (correct) state of the system, due to delay on the line (envisioned in 2.2(b) [47] as a chain of inverters). Note that both the previous and current states corresponds to the same combination of the inputs, x_1, ..., x_n, and the output $f = x_{n+1}$. If there is a fault on a line, such that the system "deviates" from the correct state, the reinforcement mechanism corrects the invalid state by forcing the system to correct state. Note that the reinforcement in the form of a positive feedback is ensured from the action results; the more rewarding the task, the greater the probability is that it will be selected again. Such reinforcement is a well-known computing paradigm in adaptive computing and artificial intelligence [60].

2.5 Performance Metrics

In the presence of noise, the gate output, Y, can be written as $Y = y + e$, where y is the noise-free output (desired signal); e is signal noise.

To evaluate performance of various models, the following metrics were used in our study:

Signal-to-Noise Ratio

Signal-to-Noise Ratio (SNR), measured in decibel (dB), is defined as follows:

$$\text{SNR} = 10\log_{10}\frac{\sigma_y^2}{\sigma_e^2} \ \ (\text{dB}), \tag{2.8}$$

where σ_y^2 and σ_e^2 are the variances of the desired signal y and the noise e, respectively.

Example 5. *Modeling with* $\text{SNR} = 10$ *dB means that the signal instantaneous power is 10 times higher than the noise power. Given an input signal of 300mV, the noise modeling under* $\text{SNR} = 5$ *dB assumes the noise variance of 14.3 mV. As the SNR of the input signal rises up to 10 dB, the signal variance reduces to 4.5 mV.*

Bit-Error Rate

Bit-Error Rate (BER) is the fraction of information bits in error; it is defined as follows:

$$\text{BER} = \frac{\texttt{Number of errors}}{\texttt{Total number of bits}}$$

The BER value of a circuit output represents the relative number of bit-errors due to the noise, interference or distortion. It is an unitless performance measure and can be expressed in percentage also.

Example 6. *Given a 300 mV input signal and* $\text{SNR} = 5$ *dB, the BER for the conventional CMOS NAND gate is* $\text{BER} = 0.0478$*, simulated for 16 nm technology. In circuit theory, it corresponds to 47 bit flips out of a total of 1000 bits.*

Kullback-Leibler Distance

The Kullback-Leibler Distance (KLD) is the measure of discrepancy between the probability distribution of the noise-free output p_y (ideal discrete signal) and the probability

distribution of the noisy output p_Y (real discrete signal) [77]. The KLD is defined as the average with respect to the probabilities p_y, scaled by the logarithmic difference between the probabilities p_y and p_Y[77]: For multivalued signal, the number of summed term rises equally with the rise in the number of states.

$$\mathrm{KLD} = \sum_{\text{States } y} p_y \ln \frac{p_y}{p_Y}$$

In the physical notion, the Kullback-Leibler Distance is the measure of the information lost when Y is used to approximate y. In information theory, KLD corresponds to the required extra bit per datum, to convey the same information as for the true distribution y, while the practical distribution for approximation is Y. The outcome of a high KLD value interprets as the high variance in the considered Y distribution, making the output distribution highly probable to cause bit-flips, hence, errors.

Example 7. *The 300 mV input signal for* SNR $= 5$ *dB results in* KLD $= 2.1714$ *for regular CMOS design @ 16nm, meaning that, 2.1714 extra bit per datum is required, to convey the same information as for the true distribution compared to approximation; it also means that chance of bit flips increases about twice.*

We compared the various gates and technologies in terms of the BER and KLD, given the same SNR. The goal of noise-tolerant design can be formulated in terms of these measurement. The physical notion of BER is the relative amount of error in the system and for KLD, it is the variance in the output signal. The lower the BER and KLD, the better the system performance. So, for a given level of SNR, we aim at achieving the performance of a model such that

$$\mathrm{BER}_{\mathrm{Proposed}} < \mathrm{BER}_{\mathrm{Conventional}}$$

or/and

$$\mathrm{KLD}_{\mathrm{Proposed}} < \mathrm{KLD}_{\mathrm{Conventional}}$$

2.6 Summary

This chapter introduces the reliable circuit design techniques from the architectural point of view. The elaborated theoretical discussion on the contemporary probabilistic design methodology based on Markov Random field is also presented in this chapter.

Chapter 3

Nanoscaled IC Design Based on Markov Random Field

3.1 Introduction

Noise-tolerance is a distinct approach to handle noise in nanoscaled ICs. Noise mitigation requires complex temperature assessment and consequent circuitry redundancy. Noise-tolerance is understood as the property of a device where the decision making process does not get compromised even in the presence of noisy input signals. Noise-immune design, design based on error correcting codes are thought to be some of the design examples [27]. In particular, in [17, 18], neuromorphic models based on Hopfield networks with Boltzmann updating rules were studied. These models exhibit noise tolerance. However, depending on the signal-to-noise ratio, thousands iterations are required in order to ensure the stability of these networks with corresponding states and outputs.

This work is concerned with noise-tolerance using Markov Random Field (MRF). The MRF, unlike the Hopfield model, does not require time-redundancy which is a multi iteration process of stabilization. It considers inputs and outputs of circuits as random variables. There is no notion of a correct logic state in the MRF model, as the states are determined using the probability distribution of signals. The correct states are those, which maximize the joint probability distribution of random variables. In circuit implementation, it is achieved via a feedback, which reinforces the most probabilistically correct state and reduces the error of incorrect (less probable) outputs.

In [47], an MRF model of logic gates is proposed using a reinforcement and updated by means of a feedback. The CMOS implementation of the modified MRF model was reported in [22]. Instead of using logic gates, this thesis work employs bidirectional

switches as the proposed design solution. Such circuits are often modeled using the Shannon expansion with the corresponding graph-based implementation, called a Binary Decision Diagram (BDD). To implement the premise of feedback, a new type of decision diagram is introduced , namely, Cyclic BDD.

3.2 Cyclic BDD

Binary decision diagram

An n-level Binary Decision Tree (BDT) represents a logic function of n variables. A BDT is a rooted directed graph with two types of vertices: non-terminal nodes (on levels 1 to n of the tree) and terminal nodes, corresponding to the function values. A BDD corresponds to a representation of a discrete function by means of the Shannon expansion. They are easily mapped to technology because the layout of a circuit is directly determined by the shape of the BDD and each node corresponds to a 1-to-2 demultiplexer (DEMUX) or a 2-to-1 multiplexer (MUX). An example of a BDD-based nanoscale structure is a wrap-gate nanowire network, in which a single or few electrons are injected into the root node, and correspondingly, registered at one of the terminal nodes. These nanowire structures for implementing logic functions have been fabricated at the Research Center for Integrated Quantum Electronics at Hokkaido University [33].

Cyclic binary decision diagram

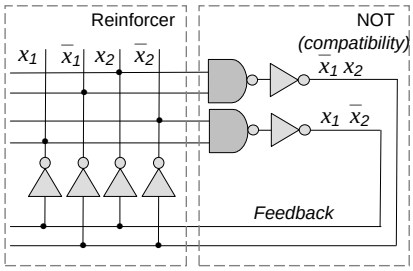
To realize a feedback in a BDD, we introduce a Cyclic BDD [66] by analogy with a cyclic graph. In the proposed cyclic BDD, the output of the root node is fed back to the "select" inputs of the nodes, using a reinforcement function, which can also be implemented by a BDD. In hardware implementation, a BDD node corresponds to a DEMUX, or MUX, because the terminal nodes store the truth-table and the output of

the function is evaluated by assigning the "select" inputs of the MUX. The first approach is called "top-to-bottom", while the second is referred to as the "bottom-to-top" design.

The cyclic BDDs can be implemented using either approach. The output of the network, which implements the gate, is fed back to the "select" inputs of the network via a reinforcement part, which can also be implemented using DEMUXes or MUXes.

3.3 Noise-tolerant NOT gate model

Using an appropriate compatibility and reinforcement function, as described in the previous chapter, an arbitrary logic gate can be implemented with a minimal redundancy. Consider the NOT function, for which an MRF model with feedback was proposed in [47] (Fig. 3.1).



(a)

x_1	x_2	**U**	**R**
0	0	0	1
0	1	$\overline{x}_1 x_2$	1
1	0	$x_1 \overline{x}_2$	0
1	1	0	0

(b)

Figure 3.1: Noise-tolerant logic NOT gate based on MRF model [47] (a) and its compatibility table (b).

In our approach, a NOT gate is represented as a BDD, realizing a compatibility function of two variables:

$$U(x_1, x_2) = x_1 \oplus x_2,$$

where x_1 is the input; $x_2 = \overline{x_1}$ is the output of the gate, as shown in Fig. 3.2. Each node

S of the BDD implements a Shannon expansion:

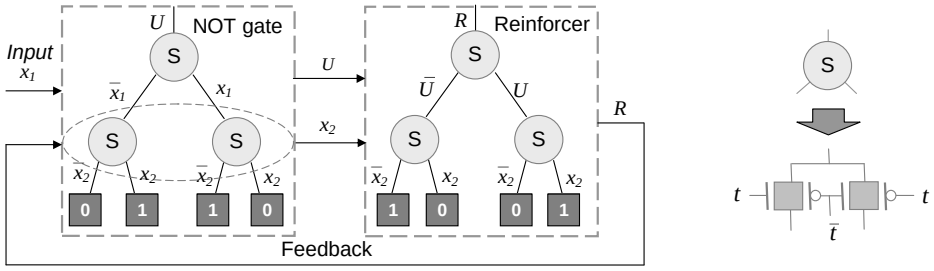
$$f = \overline{x}_i f_0 \vee x_i f_1,$$

where $f_0 = f_{x_i=0}$ and $f_1 = f_{x_i=1}$ are the left and right outgoing branches, respectively.

The second BDD implements the reinforcement function,

$$R(x_2, U) = \overline{x_2 \oplus U}$$

Note that the compatibility function varies, depending on the gate function, while the reinforcer function is formed by XNOR of the output and the compatibility function.



(a) (b)

Figure 3.2: (a) The cyclic BDD which implements the MRF model of a NOT gate consists of two connected BDDs: the left BDD implements the compatibility function $U(x_1, x_2) = x_1 \oplus x_2$; the right BDD corresponds to the reinforcer function $R(x_2, U) = \overline{x_2 \oplus U}$, as shown in its truth table; (b) Equivalent transistor circuit of a single DEMUX.

3.3.1 Experiments

Technology

The proposed cyclic BDD model was simulated using SPICE and the 16-nm Berkeley CMOS technology model (http://ptm.asu.edu/), with the gate driving voltage being $V_{DD} = 0.3\ V$ and the temperature at $27°C$. The CMOS operates in the below threshold region, where the threshold voltages for the NMOS and PMOS are 0.4797 V and

$-0.4312\ V$, respectively. When the supply voltage is lower than the threshold voltage of the transistor, the following noise effects arise:

(a) The noise margin reduction; the noise margin gets reduced due to the lower supply voltage and the thermal noise adversely affects the device noise scenario by upsetting events

(b) Electromagnetic coupling; accounts for the crosstalk noise as the closely laid interconnects cause the propagating signals in the adjacent wires to get distorted

(c) Hot-electron effects; where the electron, due to high temperature, gets the kinetic energy to overcome a potential barrier and thus causing a permanent change in the device switching characteristic

(d) Threshold variations; the random dopant concentration causes an non-uniform behavior in the connected transistors and accounts for undesired switching variation

A cyclic BDD implementation of the NOT gate is given in Fig. 3.3. The similar implementations are derived for the two-input elementary logic gates, such as NAND, NOR, AND, OR and EXOR.

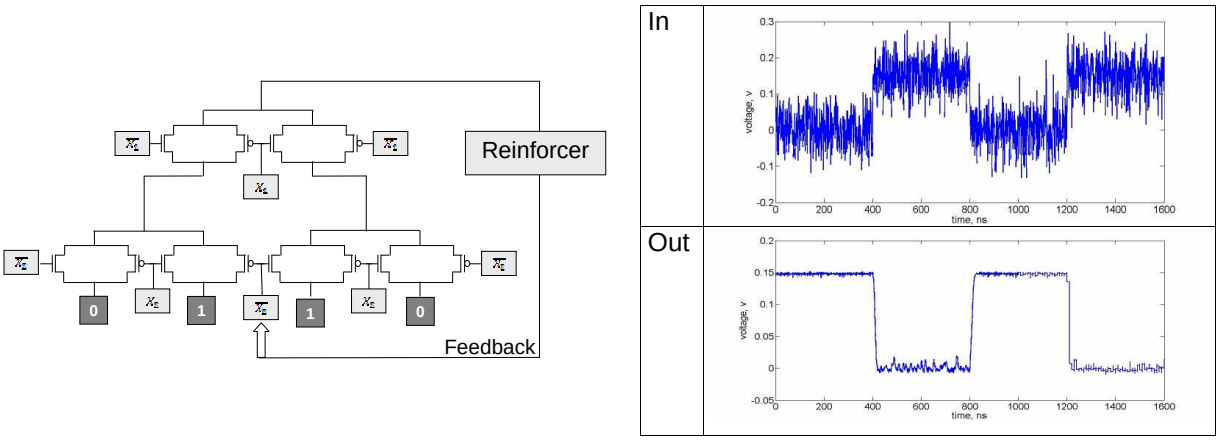
Experiment: Comparison of the NOT gate models

The comparison of three noise-tolerant models of the NOT gate are given in Table 3.1 for various levels of input noise (SNR $=3-12$ dB) for the 16-nm CMOS technology:

(a) Conventional CMOS design,

(b) MRF model [47] and

(c) The proposed cyclic BDD model.



(a) (b)

Figure 3.3: (a) CMOS implementation of a cyclic BDD for a NOT logic gate; (b) Fragment of the simulation results (noisy input and noise-tolerant output) at the subthreshold supply voltage (b).

Results and findings

1. The reported performance of a cyclic BDD network for a NOT gate, by modeling noise of subthreshold supply voltage while varying the SNR from 3 to 12 dB SNR, shows that the KLD of the cyclic BDDs ranges between 0.651 to 0.096, as opposed to 1.576 and 1.373 for the CMOS gate-level networks. The MRF model, proposed in [47], outperforms the proposed model, with the KLD varying from 0.432 to 0.012, respectively.

2. In terms of BER, the cyclic BDDs outperform the CMOS and MRF model [47] for lower SNR (3 to 5 dB). However, for higher SNR, all models perform comparably.

3.4 Noise-tolerant NAND gate modeling

In the MRF-based model, each input or output is assumed as a random variable (node in graphical representation), which value varies within the range between 0 V (logic 0) and

Table 3.1: Comparison of Noise-Tolerant NOT Gate Models, Measured in KLD (BER) for Various Levels of SNR at 16-nm CMOS Technology

SNR	Conventional	MRF [47]	**Cyclic BDD**
	C o m p a r i s o n for NOT gate		
3	1.576 (0.115)	0.432 (0.045)	**0.651 (0.059)**
5	1.571 (0.062)	0.231 (0.017)	**0.455 (0.042)**
7	1.569 (0.03)	0.116 (0.005)	**0.417 (0.010)**
9	1.459 (0.009)	0.034 (0.005)	**0.298 (0.006)**
10	1.455 (0.005)	0.029 (0.005)	**0.182 (0.005)**
12	1.373 (0.002)	0.012 (0.004)	**0.096 (0.005)**

V_{DD} (logic 1). That is, instead of a **correct** logic signal (0 or 1), the MRF model operates with the **probability** of correct logic signal. Given the observed logic signal, **correct logic values are those that maximize the joint probability distribution of all the logic variables**. The probability of state at a given node can be determined by marginalizing. Marginalizing the distribution 2.1, with respect to variable x_i, we obtain

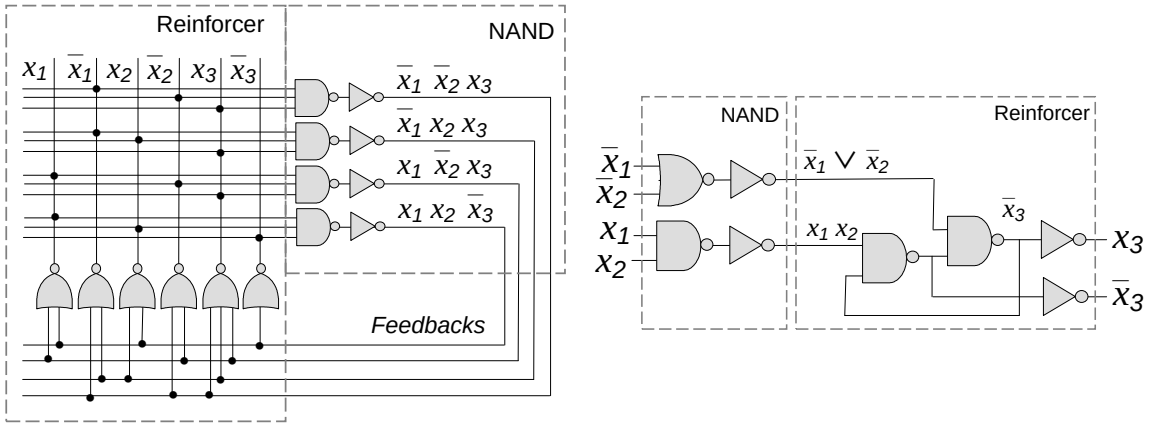
$$p(x_i|X - x_i) = \frac{1}{Z}\sum_c exp^{U(x_c)/kT},$$

this implies the following implementations of the model: (a) The feedback-based combinational MRF model [47]; (b) The latch-based MRF model [22]; (c) The cyclic BDD-based MRF model [66].

3.4.1 Gate-level networks

The combinational MRF model with a feedback, called the reinforcer, have been proposed in [47]. It was shown that this device can handle soft errors and single-event upsets. Further restructuring of the same architecture led to the hardware implementation based on the latch based structures [22].

Design examples of a NAND logic gate from [47] and [22] are given in Fig. 3.4.



(a) (b)

Figure 3.4: (a) MRF-based architecture with feedback of noise-tolerant logic NAND gate [47]; (b) latch-based architecture with feedback of noise-tolerant logic NAND gate [22].

3.4.2 Cyclic BDD based design

Figure 3.5 shows the NAND gate representation using regular non-reduced BDD and a pass-gate CMOS transistor-level implementation of one node, or multiplexer.

In the cyclic BDD [66], the output of the root node is fed back to the "select" inputs of the nodes using a reinforcement function. This function can be implemented by another BDD. The output of the network, which implements the gate, is fed back to the "select" inputs of the network through a reinforcement part, which can be implemented using multiplexers.

The BDD-based MRF model of a NAND gate using two cyclic BDDs is presented in Figure 3.6. The first BDD implements compatibility function $U(x_1, x_2, x_3) = x_1x_2 \oplus x_3$, where x_1 and x_2 are the inputs and x_3 is the output of NAND gate, that is, $x_3 = \overline{x_1x_2}$. The second BDD is the reinforcer, implementing the function $R(x_3, U) = \overline{x_3 \oplus U}$.

Figure 3.7 demonstrates the efficiency of the cyclic BDD implementation of the MRF model for a NAND gate. For both noisy inputs (Fig.3.7a,b), the regular CMOS gate-level implementation produces a noisy output (Fig.3.7c), while the cyclic BDD-based

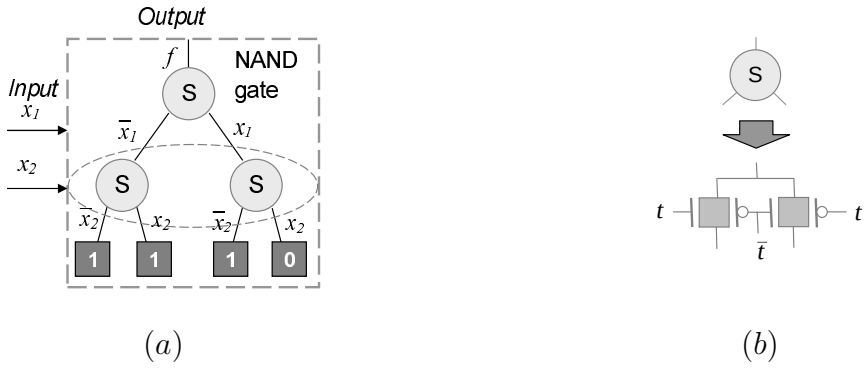


Figure 3.5: (a) Complete BDD for a NAND gate; (b) BDD node implementation on pass-gate transistors.

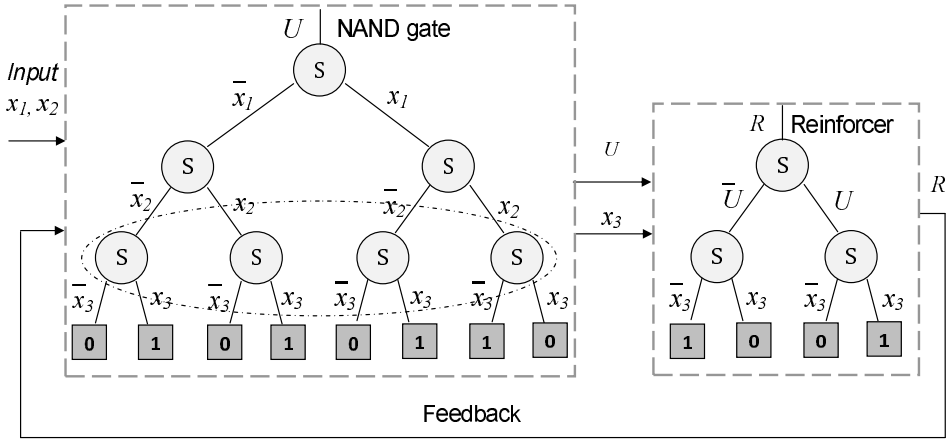


Figure 3.6: The cyclic BDD-based architecture of noise-tolerant NAND logic gate, consisting of two connected BDDs. The left BDD implements the compatibility function $U(x_1, x_2, x_3) = x_1 x_2 \oplus x_3$, and the right BDD corresponds to the reinforcer function $R(x_3, U) = \overline{x_3 \oplus U}$ as shown in its truth table.

implementation of the MRF model of the same gate demonstrates the superior noise attenuation performance (Fig.3.7d).

3.4.3 Experiments

The designed cyclic BDD model was simulated using PSPICE and the 16-nm Berkeley CMOS technology model (http://ptm.asu.edu/). The gate driving voltage is $V_{DD} = 0.3\ V$, and the temperature is $27°C$. The CMOS operates in the below threshold region, where the threshold voltages for the NMOS and PMOS are $0.4797\ V$ and $-0.4312\ V$, respectively. The cyclic BDD of a NAND gate (Fig. 3.5) is implemented using the 16-nm Berkeley CMOS technology model. The corresponding transistor level circuit is shown in Figure 3.8.

Experiment I: Performance comparison of two-input NAND gate models

The goal of this experiment was to compare the performance of the following two-input noise-tolerant logic gate models:

(a) Standard (conventional) CMOS design,

(b) MRF model [47],

(c) MRF model [22],

(d) MRF model on cyclic BDD [66],

A fragment of our experiment is given in Table 3.3 for various levels of noise measured in terms of SNR when $SNR = 3, 5, 7, 9, 10, 12$ (dB). The output was evaluated using two metrics, which are the BER and KLD. The proposed MRF-BDD based design shows promising results compared to the two different MRF designs and outperforms the CMOS based design in every aspect.

Input 1

Input 2

CMOS output

BDD output

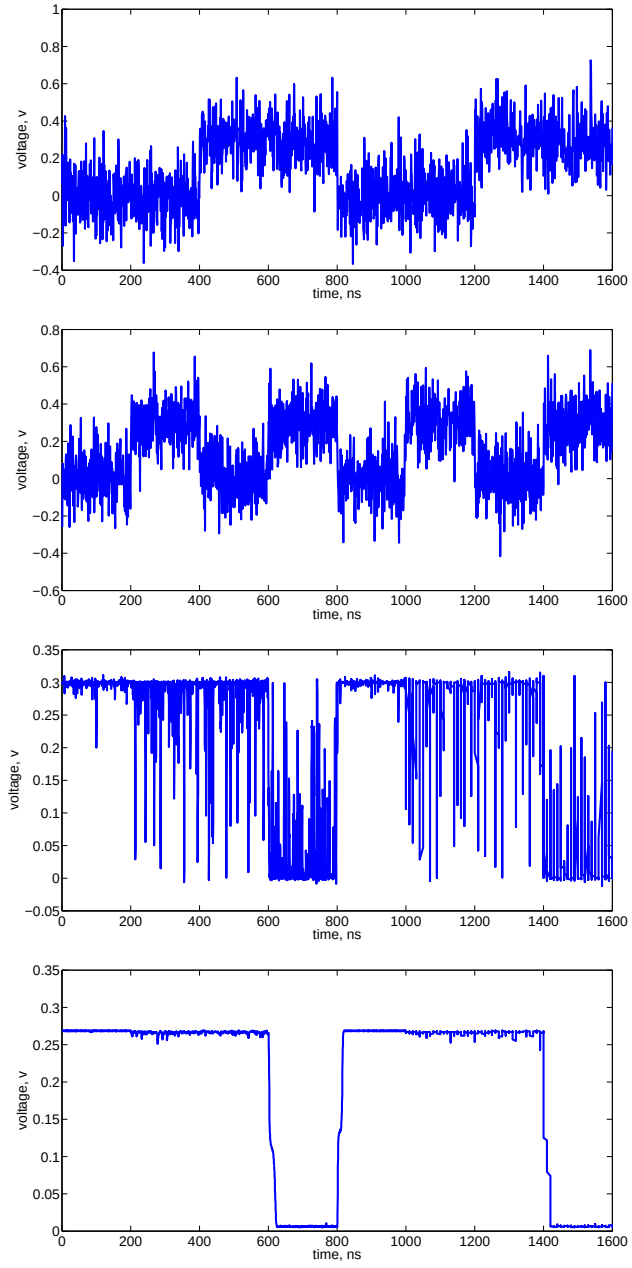


Figure 3.7: Fragment of the simulation results at the subthreshold supply voltage for the NAND gate: The inputs and corresponding output for CMOS and BDD based design.

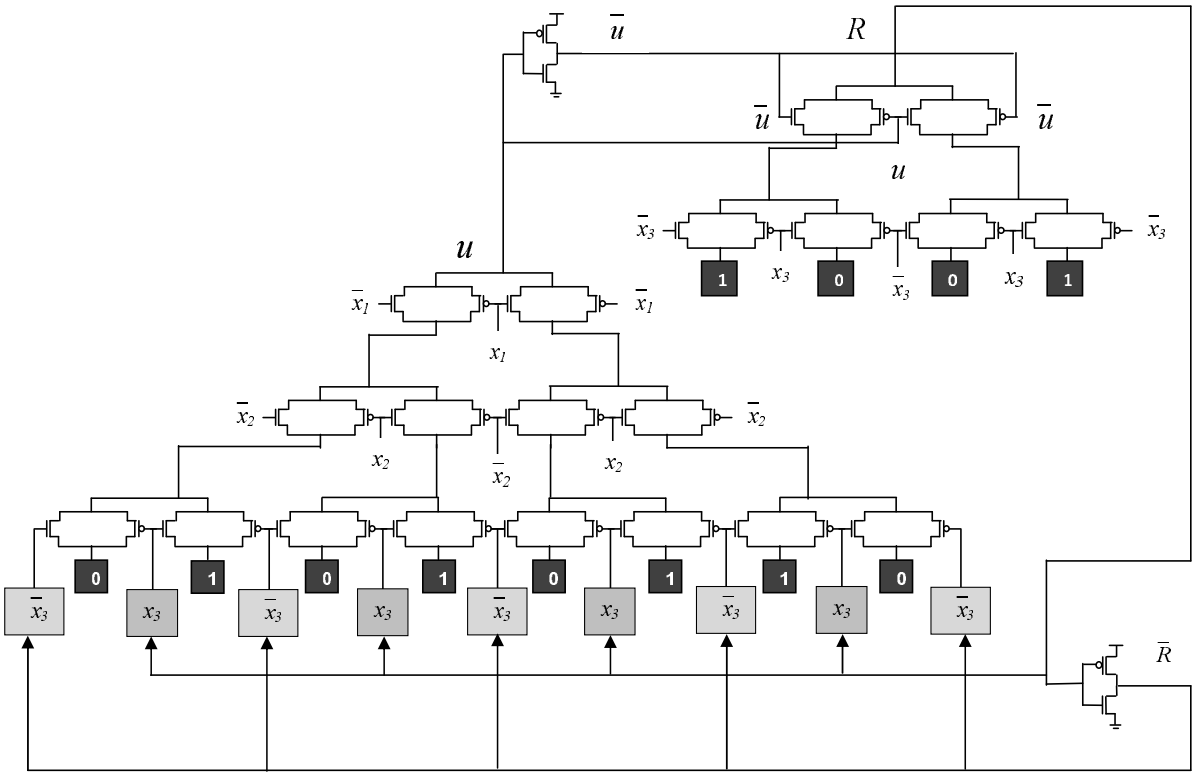


Figure 3.8: The 16-nm Berkeley predictive CMOS implementation of cyclic BDT for NAND logic gate based MRF model.

Experiment II: Performance comparison of elementary logic gate models

The goal of this experiment is to compare the performance of various logic gate models based on cyclic BDDs. Table 3.3 and Fig. 3.9 illustrate the performance of each model, measured using the KLD and BER metrics.

Results and findings

1. In the worst-case scenario, when $SNR = 3$dB, the vulnerability of the proposed cyclic BDDs, started with the most vulnerable gates, are as follows:

Table 3.2: Comparison of noise-tolerant NAND gate models measured using the KLD and BER metrics for various levels of SNRs simulated at 16nm.

SNR	Conventional CMOS		MRF model [47]		MRF model [22]		MRF-BDD [66]	
	KLD	BER	KLD	BER	KLD	BER	KLD	BER
3	2.2144	0.1028	1.2714	0.0275	0.8618	0.0463	1.2463	0.0356
5	2.1714	0.0478	0.969	0.0157	0.5076	0.0177	0.9130	0.0169
7	1.9847	0.0160	0.7225	0.0074	0.1759	0.0079	0.5659	0.0121
9	1.9224	0.0069	0.4091	0.0067	0.0321	0.0054	0.2031	0.0093
10	1.6038	0.0048	0.1400	0.0052	0.0248	0.0055	0.1235	0.0067
12	1.4000	0.0027	0.096	0.0043	0.0073	0.0051	0.1031	0.0053

BER metric: EXOR OR AND NOT NOR NAND

KLD metric: EXOR OR AND NOR NAND NOT

The best noise tolerance is demonstrated by the two-input logic NAND, NOR and single input NOT gate models.

2. There is no difference, in terms of the BER metric, for all gates, if $SNR = 9$ to 12 dB. However, in terms of the KLD, NAND and NOT gates demonstrate better noise tolerance.

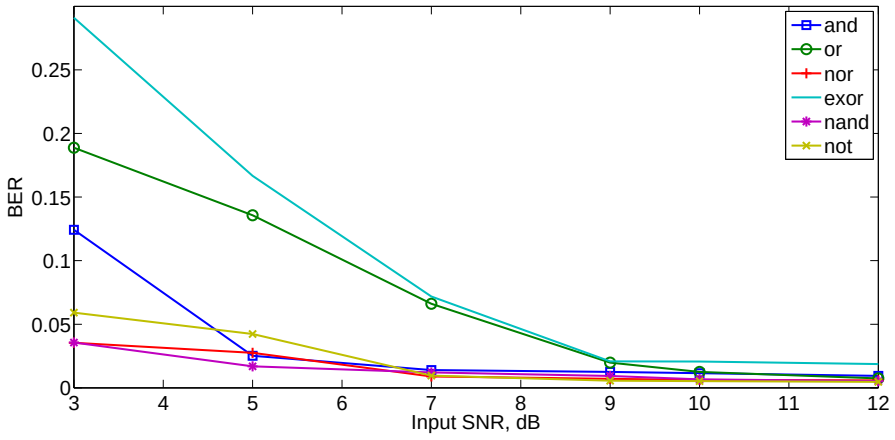
3. Compared with the regular CMOS implementation, the noise-tolerance of the cyclic BDDs, measured in terms of the BER metric, is from 2 to 3 times better, in the worst-case scenario when $SNR = 3$. For example, the cyclic BDD model of the NAND gate is 2.9 times more noise-tolerant, in terms of BER (0.036 for BDD versus 0.103 for CMOS), if $SNR = 3$dB.

3.5 Summary

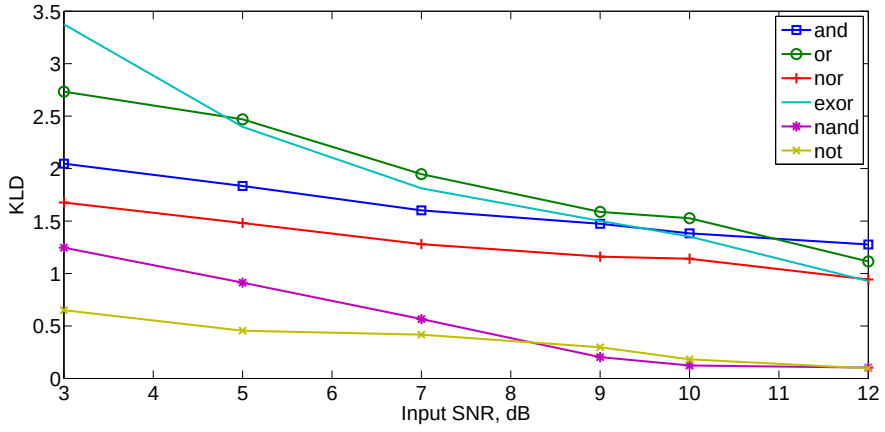
The major contribution of this chapter is the new noise and fault-tolerant designs of nanoscale ICs using the Cyclic BDD. The proposed feedback schemes to realize the MRF

Table 3.3: Noise Tolerance of Cyclic BDD Models of the Two-input Logic Gates, measured in terms of KLD and BER for various levels of SNR, simulated for 16-nm CMOS technology.

SNR	AND		OR		NOR		EXOR		NAND		NOT	
	KLD	BER	KLD	BER	KLD	BER	KLD	BER	KLD	BER	KLD	BER
3	2.0469	0.1243	2.7334	0.1888	1.677	0.0354	3.3746	0.291	1.2463	0.0356	0.6509	0.0591
5	1.8343	0.0252	2.4694	0.1357	1.481	0.0275	2.398	0.1667	0.913	0.0169	0.455	0.0423
7	1.6016	0.014	1.947	0.0661	1.2801	0.0089	1.8119	0.0718	0.5659	0.0121	0.4171	0.0096
9	1.4743	0.0125	1.5877	0.0198	1.1609	0.0071	1.5017	0.0208	0.2031	0.0093	0.2975	0.0056
10	1.3832	0.0115	1.5273	0.0125	1.1407	0.0062	1.3521	0.0207	0.1235	0.0067	0.1821	0.0053
12	1.2765	0.0094	1.1148	0.0074	0.9431	0.0059	0.9285	0.0187	0.1031	0.0053	0.0964	0.0047



(a)



(b)

Figure 3.9: (a) Comparison of noise-tolerance of the cyclic BDD models of elementary logic gates, measured in terms of BER metric; (b) KLD metric with respect to the input SNR for the 16-nm CMOS technology.

models of logic circuits are compatible with CAD, physical layout and implementation of ICs on nanoscaled micro and nano-electronic devices. For example, the quantum-effect resonant-tunneling devices, photonic devices, wrap-gate nanowire BDD circuits [33], resistor-logic demultiplexers [51] as well as other solutions are the potential physical embodiment of the model. The proposed design suits the noise and fault-tolerant BDD-centric networks, because the robust cyclic BBDs may correct and accommodate soft technological and operational errors. Another extension of the MRF models and cyclic BDDs is their capacity to map 2D onto the 3D nanoscale structures [17].

This chapter compared three different implementations of the noise tolerant MRF model of logic gates using the SNR, KLD, and BER metrics. All these implementations showed superior immunity to noise compared to the conventional CMOS implementation.

Chapter 4

Probabilistic Model of Logic Networks Using Shared Cyclic BDDs

4.1 Introduction

This chapter pursues the goal of designing networks of logic gates based on the MRF model, suitable for implementation on available technology such as wrap-gate nanowire BDD networks [33]. In chapter 3, a realization of the MRF model of elementary logic gates using so-called *cyclic BDD* was introduced.

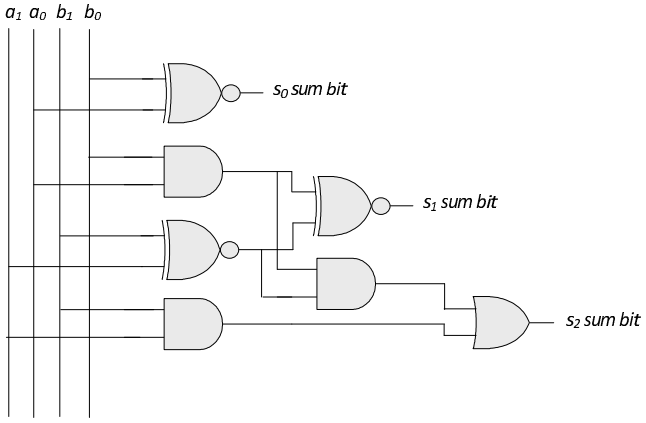
This chapter studies the efficiency of implementing the BDD for logic networks with many outputs, called shared BDDs [65], using the MRF model. It extends the shared BDD by adding feedbacks, thus forming the *cyclic shared BDDs*. It compares area, power and delay of the CMOS models (traditional and MRF-based from [47]) of a 2-bit adder, against the CMOS MRF model implemented by the shared cyclic BDD. It also introduces the result of experiments with large benchmarks from MCNC'91 set and shows that the shared cyclic BDDs demonstrate lesser total area and power consumption, while providing superior noise immunity.

4.2 MRF model of a two-bit adder using a Shared cyclic BDD

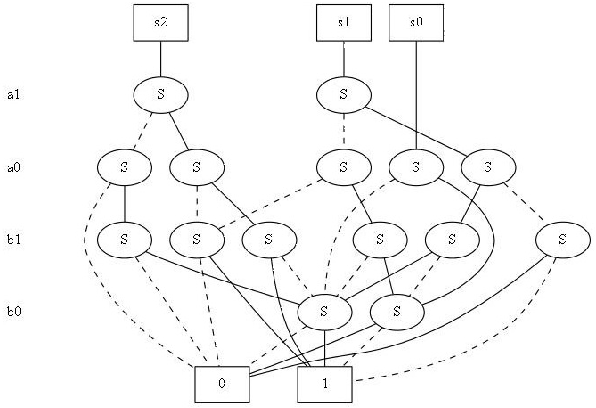
4.2.1 Shared BDD

Shared BDDs were introduced in [65]. These BDDs are suitable for multiple-output circuits. For example, shared BDDs were used in [33] for design of a two-bit ALU with 2 outputs on a wrap-gate nanowire BDD network. Fig. 4.1 shows a shared BDD of a

two-bit adder.



(a)



(b)

Figure 4.1: A two-bit adder (a) and its implementation using a shared BDD (b).

4.2.2 Shared cyclic BDD

A shared cyclic BDD for a two-bit adder is designed by adding a reinforcer block to each output (Fig. 4.2). Note, that the switches are not shown, but assumed, in this figure. The only modification necessary to form a cyclic BDD from a regular shared BDD is the shifting of the output variable node to the top of the tree (as the root node), other than

being on the bottom-most level. The reinforcer block remains the same as the regular Cyclic BDD, and feeds the output back to the corresponding nodes of the shared BDD.

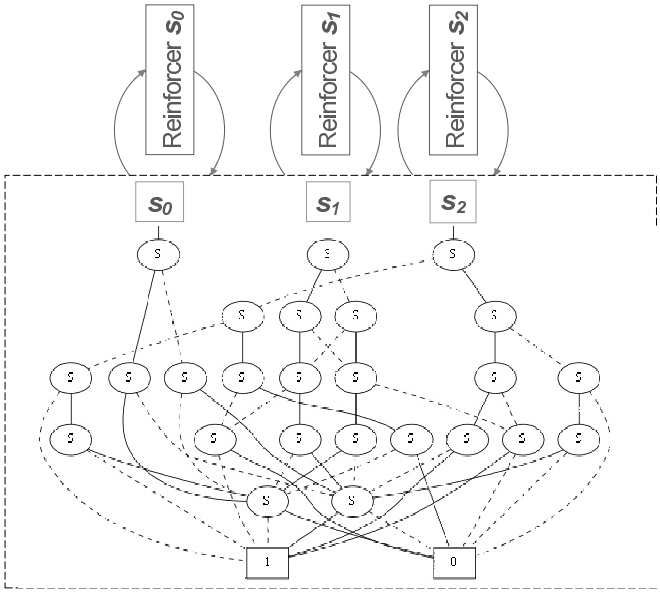


Figure 4.2: Noise-tolerant two-bit adder based on implementation of the MRF model by cyclic shared BDD.

4.3 Experiments

The MRF models of the multi-output switching functions based on the shared cyclic BDDs were simulated using SPICE and the 16-nm Berkeley CMOS technology model (http://ptm.asu.edu/), with the gate driving voltage being $V_{DD} = 0.3\ V$ and the operating temperature being room temperature, $27°C$. The transistors operated in the below threshold region, where the threshold voltages for the NMOS and PMOS were $0.4797\ V$ and $-0.4312\ V$, respectively. For shared BDD design, the CUDD package was used (http://vlsi.colorado.edu/ fabio/CUDD/).

We aimed at evaluating:

1. The circuit's behavior in the presence of noise, compared to the conventional CMOS

design of a two-bit adder. Since the previous studies were limited to two-input, single-output logic gates, for the 4-input, 3-output circuit, we expected significant decrease in BER and KLD values while increasing the SNR.

2. The power dissipation, delay and the number of transistors of our design compared to the known CMOS designs.

3. The feasibility of the MRF model implemented using the shared cyclic BDD for more complicated circuits.

Experiment 1: Comparisons in KLD and BER metrics for various levels of SNR

In Table 4.1, the comparison of conventional CMOS design of the 2-bit adder and noise-tolerant design of this adder based on the MRF model implemented by shared cyclic BDD using the 16-nm CMOS technology is introduced. In this experiment, the input noise was changed from SNR = 3 dB to SNR = 12 dB.

Table 4.1: Comparison of conventional 2-bit adder and MRF model of 2-bit adder (implemented using a shared cyclic BDD) in terms of KLDs and BERs for various levels of SNR, simulated at 16nm CMOS technology.

Input SNR	Conventional CMOS 2-bit adder						MRF model of 2-bit adder (shared cyclic BDD)					
	Output s_2		Output s_1		Output s_0		Output s_2		Output s_1		Output s_0	
	BER	KLD	BER	KLD	BER	KLD	BER	KLD	BER	KLD	BER	KLD
3	0.0716	0.7521	0.4399	2.4752	0.2334	2.3501	0.0801	0.2204	0.1456	1.4876	0.1465	1.6016
5	0.0538	0.5909	0.2542	2.1278	0.1502	2.2451	0.0238	0.079	0.047	1.2274	0.0573	0.8786
7	0.043	0.3749	0.1032	1.7949	0.0982	1.9549	0.0122	0.0292	0.0217	0.6658	0.0219	0.4712
9	0.042	0.3585	0.0783	1.6631	0.0973	1.6072	0.0091	0.0087	0.0146	0.3512	0.0122	0.4296
10	0.0419	0.3277	0.0682	1.484	0.0929	1.5012	0.0079	0.0073	0.017o	0.2894	0.0113	0.2713
12	0.0362	0.1849	0.059	1.3391	0.0907	1.3681	0.0051	0.0073	0.0159	0.2097	0.0106	0.2277

Results and findings

As expected, the MRF model of a 2-bit adder, implemented by a shared cyclic BDD, outperforms the conventional CMOS design. In particular, the improvement in BER is

around 50% to 80%, and for the KLD values, the improvement is around 40% to 90%. This represents the reduction in the output errors and also the decreases the variance in the signal that would cause bit-flips. This noise-tolerant effect is comparable with the results, reported for logic gates and circuits in [47, 22].

Fig. 4.3 illustrates strong noise-immune property of the proposed shared cyclic BDD implementation of the MRF model, compared with a conventional design, given the fixed SNR value 7db for the input signal.

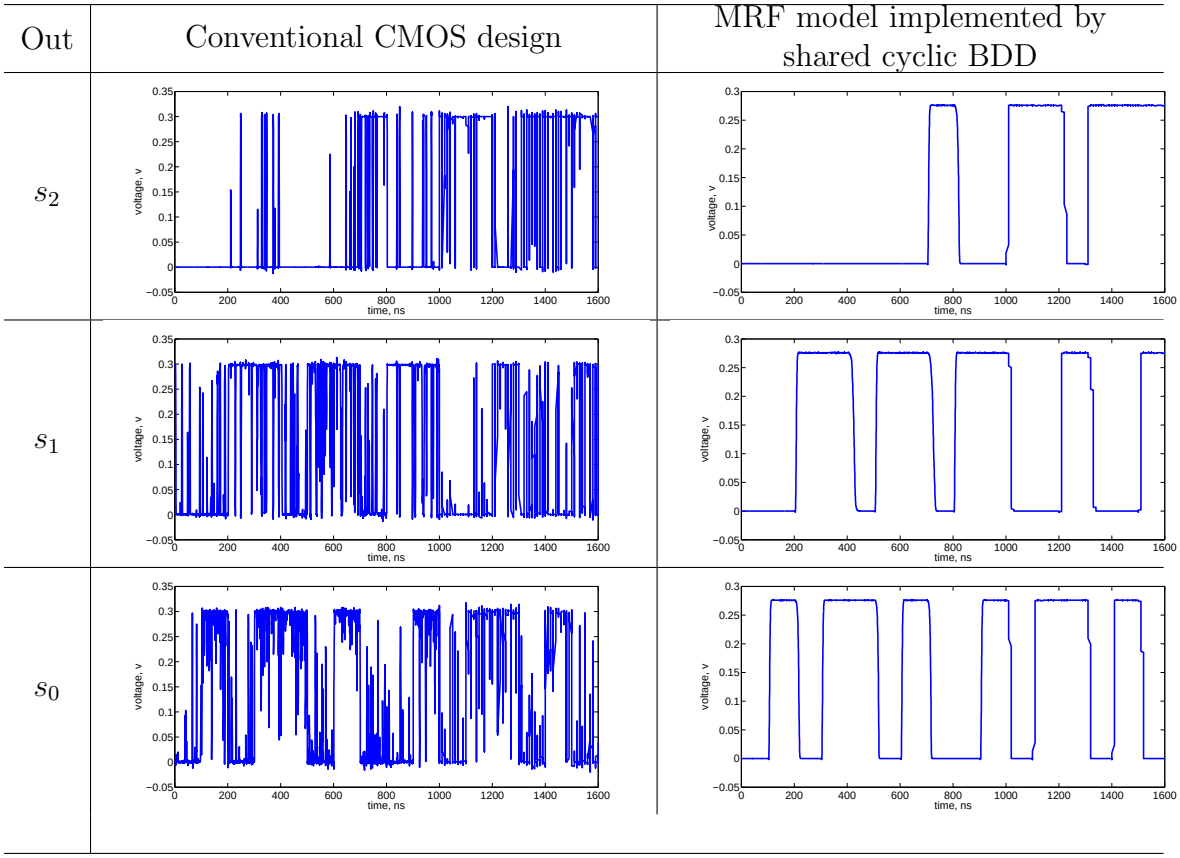


Figure 4.3: Noise output signals s_0, s_1, and s_2 of the 2-bit adder for input SNR value 7db for conventional CMOS design (left), and the same noise-free signals in CMOS MRF model, implemented using a shared cyclic BDD (right).

Experiment 2: Area, Power and Delay

In Table 4.2, the power dissipation, delay and the number of transistors are compared for both CMOS design and the MRF model implementations, reported in [47, 22, 66]. The operating voltage for the CMOS based design is $0.7V$ as indicated by the ITRS roadmap [7]. The BDD based noise-tolerant models were operated in the subthreshold region with the transistor driving voltage being $0.3V$.

Table 4.2: Performance comparison of conventional CMOS design of a 2-bit adder and its noise-tolerant designs based on various implementations of the MRF model reported in [47, 22, 66], and the proposed shared cyclic BDD implementation.

Parameter	**CMOS design**	**MRF design[47]**	**MRF design [22]**	**MRF design [66]**	**Proposed design**
Supply voltage (v_{dd})	0.7	0.3	0.3	0.3	0.3
No. of transistors	60	420	196	176	140
Power dissipation (nW)	0.151	0.042	0.034	0.039	0.032
Output delay avg. (nS)	3	16.4	11.1	21.2	18.3

Results and findings

The presented study of the MRF model, implemented using the shared cyclic BDDs, was aimed at providing the feasibility of the noise-tolerant designs for the nanowire structures, such as the existing wrap-gate nanowire BDD structures [33], as currently those structures do not possess any noise immunity. As reported below, the simulated delay and power dissipation of the BDD-based structures are comparable with the known CMOS MRF models and in certain cases outperform the conventional design.

Experiment 3: Simulation of MCNC'91 benchmark circuits

It is known, that shared BDDs are preferable for large circuits [65]. In order to study effectiveness of the shared cyclic BDDs in the implementation of the MRF model, we conduct experiments with MCNC'91 benchmarks (Table 4.3). The conventional CMOS designs were simulated for the $V_{dd} = 0.7V$ and MRF designs for $V_{dd} = 0.3V$. Table 4.3 contains the MCNC'91 benchmark circuits specifications (the number inputs and

outputs), the number of transistors, # tran and power dissipation in nW for various designs.

Results and findings

The simulation demonstrated about 4.4 - 4.9 times increase in area for MRF model described in [47], compared to the regular (not noise-immune) CMOS implementation of the same circuits, and 1.02 - 1.53 times increase in area for the shared cyclic BDD. The proposed circuit design approach shows a huge improvement over the power dissipation of the total circuit. The proposed design consumes about 3 times less power than the MRF based model proposed in [47] and 15 times less power than the conventional CMOS design. The whole dissipation would increase rather rapidly for the reliability of CMOS circuit with the introduction of approaches like Triple Modular Redundancy (TMR). Whereas, the proposed Shared Cyclic BDD approach doesn't need any additional circuitry for the reliability.

Table 4.3: Comparison of the number of transistors and power dissipation required for conventional CMOS implementation, MRF model [47], and the proposed design for the MCNC'91 circuits.

MCNC'91 Benchmarks			**CMOS design,** $V_{dd} = 0.7V$		**MRF based design [47],** $V_{dd} = 0.3V$		**Proposed design,** $V_{dd} = 0.3V$	
Circuits	Input	Output	# tran	Power (nW)	# tran	Power (nW)	# tran	Power (nW)
5xp1	7	10	568	1.431	2756	0.487	608	0.138
alu4	14	8	6928	17.458	33416	5.915	7166	1.626
con1	7	2	78	0.196	356	0.063	120	0.027
ex5	8	63	5448	13.728	25964	4.596	6108	1.386
cordic	23	2	604	1.522	2612	0.462	720	0.163
misex1	8	7	356	0.897	1700	0.300	386	0.087
rd53	5	3	236	0.594	1012	0.179	284	0.064
squar5	5	8	346	0.871	1532	0.271	356	0.080

4.4 Summary

This chapter proposes a novel probabilistic model of logic networks based on Shared Cyclic BDDs. These structures realize the MRF based maximum state probability principle discussed in previous chapter.

The chapter considers the Shared Cyclic BDDs, for practical circuit, 2-bit Binary adder circuit, as a design example. Furthermore, the benchmark circuits from MCNC'91 set were also simulated and tested using the proposed design against the contemporary methodologies. The analysis of layout area overhead and power dissipation shows the promising results for probabilistic design.

Chapter 5

Noise Immune Multivalued Logic Design

5.1 Introduction

Ternary, or 3-valued, inverter is a basic element for ternary storage devices. Ternary logic gates are known for advantages over their binary counterparts, such as increasing transmitted information, decreasing the number of connections, power dissipation, delay, and the complexity of circuits. The multivalued domain inherits these significant attributes from digital world. A Multivalued (MVL) function f of n variables $x_1, x_2, x_3, ..., x_n$ is defined as

$$f = \begin{cases} f(x_1, x_2, ..., x_n) \\ \\ \quad f : K^n \rightarrow K \end{cases}$$

where the set $K = 0, 1, ..., K-1$ is the the k-valued domain, and k is the radix of the logic. For the value of $K = 2$, the function becomes a Binary switching function.

This chapter reviews the previously reported results on the subject as follows. The CMOS ternary logic is discussed in [80]. The ternary NOT gate for building SRAM devices was studied in [79].

A noise-tolerant model of a ternary inverter is an extension of the MRF models of logic gates developed in [47, 22], and also the results based on previously discussed cyclic Binary Decision Diagrams (BDDs) [66].

5.2 Ternary Inverter and MIN gate

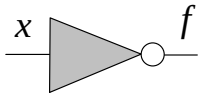
As the simplest definition for a multivalued function can be $f : \{0, 1, ..., m-1\}^n \rightarrow \{0, 1, ..., m-1\}$, $f : \{0, 1, 2\}^n \rightarrow \{0, 1, 2\}$ is the definition for Ternary logic function where the value of states, $m = 3$.

The inversion operation is specified by $\overline{x} = (m-1) - x$, where m is the number of states and x is the variable. Similarly the ternary inverter can be presented by the equation $\overline{x} = 2 - x$, as for ternary, the value of $m = 3$.

The MIN function is a two input single output ternary function defined by the following equation,

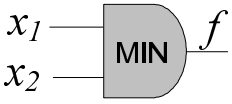
$$MIN(x_1, x_2) = \begin{cases} x_2, \ if \ x_1 \geq x_2 \\ \\ x_1 \ otherwise \end{cases}$$

For n variables, the equation can be written as $\text{MIN}(x_1, x_2, ..., x_n) = x_1 \wedge x_2 \wedge .... \wedge x_n$. The Figure 5.1 shows symbols of a ternary inverter and a ternary MIN gate along with their truth tables.



x_1	f
0	2
1	1
2	0

(a)



x_1	x_2	f
0	0	0
0	1	0
0	2	0
1	0	0
1	1	1
1	2	1
2	0	0
2	1	1
2	2	2

(b)

Figure 5.1: Gate symbol and truth table for a ternary inverter (a) and two-input ternary MIN gate (b).

5.3 Design of a noise-tolerant ternary inverter

Consider the probabilistic analysis of noise-tolerant ternary inverter, using previously discussed MRF model. Its truth table is given in Table 5.1. Using vector **U**, we find the compatibility function in arithmetic form [74]:

$$f(v) = \frac{1}{2}(-v_2 + v_2^2 - v_1 + 11v_1v_2 - 6v_2^2v_1 + v_1^2 - 6v_1^2v_2 + 3v_1^2v_2^2) \tag{5.1}$$

Embedding this function into the model (2.1) corresponds to the following equivalence: $f(v) = E(v)$. After substitution $E(v)$ into model (2.1), we obtain the Gibbs joint probability distribution: $p(v_1, v_2)$. Summing over all possible values of the variable v_1 eliminates this variable, leaving the marginal probability distribution of the output v_2:

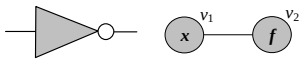
$$\begin{aligned} p(v_2) &= \frac{1}{Z_1} \sum_{v_1 \in 0,1,2} \exp\left(-\frac{E(v)}{kT}\right) = \frac{1}{Z_1} \left[\exp\left(\frac{-2v_2 + 2v_2^2}{4kT}\right)\right. \\ &+ \left. 2\exp\left(\frac{8v_2 - 4v_2^2}{4kT}\right) + 2\exp\left(\frac{4 - 6v_2 + 2v_2^2}{4kT}\right)\right] \end{aligned}$$

Family of distributions $p(v_2)$ for various parameters kT is given in Figure 5.2. We observe that output values are grouped around logic values 0, 1, and 2 with equal probability (three picks); this is because it is assumed that the input is equally likely to be 0, 1, or 2. So, it is a reasonable behavior model, that is, the Gibbs distribution provides an acceptable probabilistic approximation of real uncertainty, caused by noise. With the increased value of kT, a decrease in the terminal probabilities are observed while the probabilities of the intermediate voltage levels rise. The modeling proposes increased amount of variance in the output signal with the increment of the system temperature.

In this CMOS model, a high-resistance transmission gate is placed between a low-resistance inverter and 0.3V supply, in order to produce a middle voltage level (Fig. 5.3). That is, the threshold voltage of transistors $P2$ and $N2$ is half of the supply voltage. Note

Table 5.1: Design of a noise-tolerant ternary inverter based on the MRF model.

Gate **Graph-model** **Compatibility truth table**



x	$\overline{\mathbf{x}}$
0	2
1	1
2	0

v_1	v_2	U	Minterm
0	0	0	Undesirable
0	1	0	Undesirable
0	2	1	$v_1^0 v_2^2$
1	0	0	Undesirable
1	1	1	$v_1^1 v_2^1$
1	2	0	Undesirable
2	0	1	$v_1^2 v_2^0$
2	1	0	Undesirable
2	2	0	Undesirable

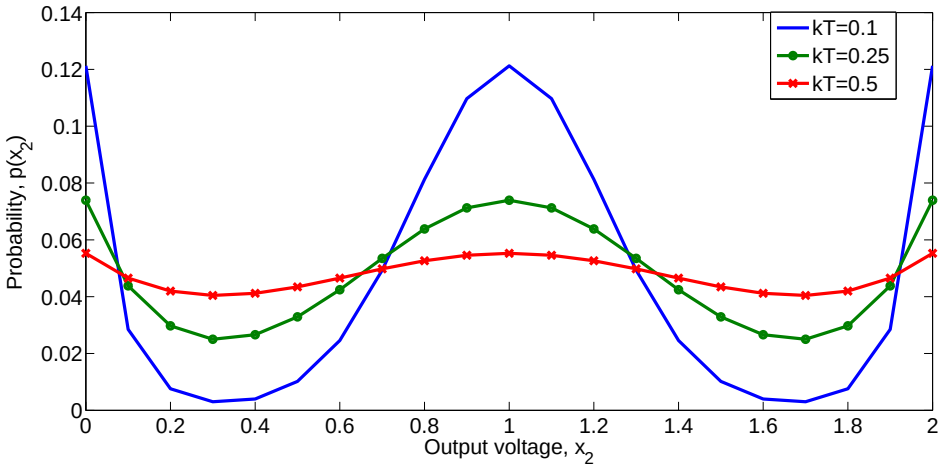


Figure 5.2: Output voltage probability distribution for a ternary inverter

that both $P2$ and $N2$ are in the subthreshold region for the middle state "1". A typical problem of this CMOS implementation (and the other known ternary gates, for example, [80]), is that the noise margins of the voltage transfer characteristic are decreased. This is a well-known effect, observed in many multi-valued circuit realizations, that for the fixed values of the highest and lowest voltages, noise-tolerance of a circuit with more logic levels is worse, compared with binary analogs.

Arithmetic SOP form of the compatibility function is derived from the compatibility

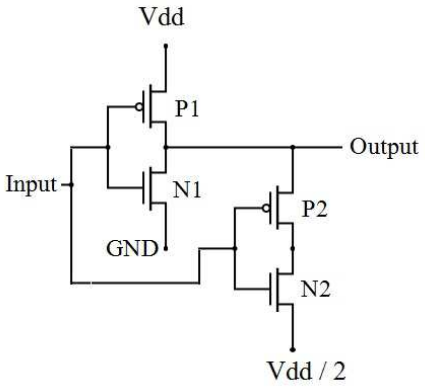


Figure 5.3: A conventional CMOS ternary inverter [79].

truth table for three valid state combinations {02,11,20} as follows

$$\mathbf{U} = \frac{1}{4}(v_1^0 v_2^2 + v_1^1 v_2^1 + v_1^2 v_2^0) \tag{5.2}$$

where the literal operation is defined as $y^x = 2$ if $x = y$ and $y^x = 0$ otherwise, $x, y \in \{0, 1, 2\}$.

To implement an MRF model that embodies the above SOP form (5.2) and uses maximum likelihood principle and multiple feedback, we need a conventional ternary inverter and a conventional two-input ternary MIN-NOT gate. A ternary inverter is shown in Fig. 5.3 [79]. The inverter has two storage nodes, v_1 and v_2, that can take three different logic values, 0, 1 and 2. The logic value 0 is represented by the voltage $0V$, and logic levels 1 and 2 correspond to $0.15V$ and $0.3V$, respectively. For the input value of *logic 0*, the both the circuits stay inactive and *logic 2* is propagated to the output. For the introduction of *logic 1* in the input, the first circuit stays inactive but the second circuit turns on to provide the middle logic level. The similar operation can be described for the *logic 2* input.

Ternary two-input MIN-NOT gate, $\overline{\text{MIN}}(x_1, x_2)$, is shown in Fig. 5.4. In this circuit, $P1$ and $P2$ are the p-channel enhancement transistors, whereas $N1$ and $N2$ are n-channel enhancement transistors. The threshold voltage, V_T, and supply voltage, V_{dd}, are specified

by the following conditions: $0 < |V_T| < |V_{dd}|$ for $P2$ and $N2$ and $|V_{dd}| < |V_T| < |2V_{dd}|$ for $P1$ and $N1$.

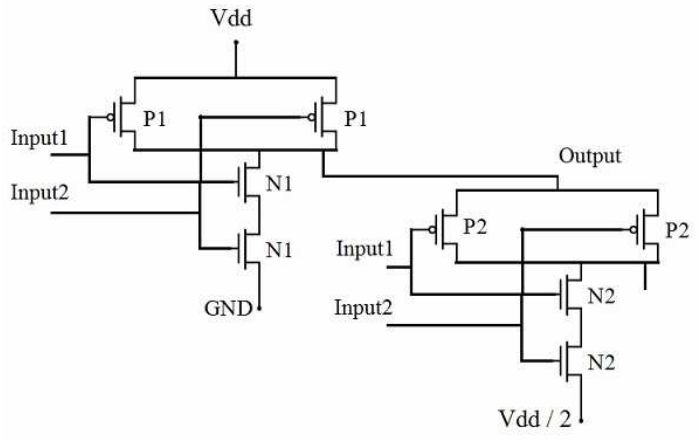


Figure 5.4: Conventional CMOS ternary two-input MIN-NOT gate.

The circuit for noise-tolerant ternary inverter is shown in Fig. 5.5. Different state combination of the storage nodes ensures the propagation of the conditional probability within the circuit. Assigning the highest probability to the valid state combination is ensured by the reinforcement feedback. The feedback forces the circuit to return to the state combinations, for which the clique energy function is true, thus making the system tolerant to incorrect output due to noise. For example, suppose the nodes, v_1 and v_2 takes on the logic values 0 and 2 respectively. Getting fed from cyclic inverters [74] the inputs of the MRF based circuit gets all the possible states. As for the connection, the first Ternary MIN-NOT gate stays inactive and feeds logic level 0 to corresponding input nodes. The second and third MIN-NOT gate outputs are logic 1 and logic 2 which are consistent with the storage node values. For the other valid state combinations between the two storage nodes, the stability of the network can be defined accordingly.

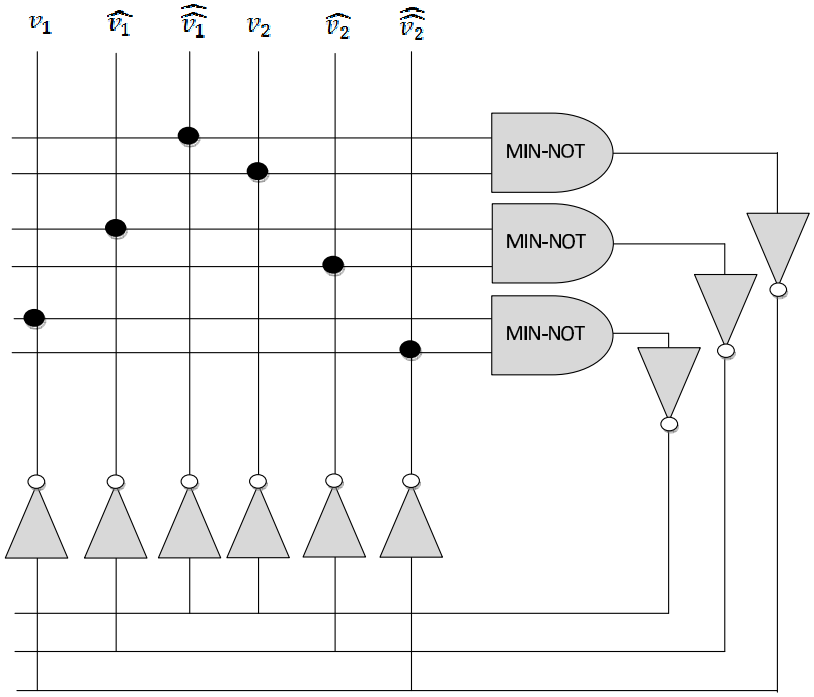


Figure 5.5: A ternary noise-tolerant inverter based on MRF model.

5.4 Experiments

The experimental study involves measurement of the noise-tolerance characteristics of the proposed MRF-based model of a ternary inverter in various metrics, and also comparison of the latter against the conventional CMOS design of the inverter.

Technology

The proposed noise tolerant model was simulated using SPICE and the 16-nm Berkeley CMOS technology model (http://ptm.asu.edu/), with the gate driving voltage being $V_{DD} = 0.3\ V$ at the room temperature, $27°C$. CMOS models with two different threshold voltages were used for the ternary experiments. The CMOSs denoted as P1 and N1 are the regular PMOS and NMOS models with threshold voltages $0.4797\ V$ and $-0.4312\ V$ respectively. The P2 and N2 models represent the PMOS and NMOS with threshold

voltages being $-0.17\ V$ and $0.17\ V$ respectively.

Set 1 of experiments: comparison in SNR, KLD and BER metrics

Table 5.2 contains the results of comparison of conventional CMOS design of a ternary inverter [79] and the proposed noise-tolerant design. The latter is based on the MRF model, implemented using the 16-nm CMOS technology. In this experiment, the input SNR was varied between 9 dB 18 dB.

Table 5.2: Comparison of the ternary conventional and noise-tolerant inverters.

Input SNR	**Conventional CMOS design [79]**		**CMOS MRF-based design**	
	BER	KLD	BER	KLD
9	0.2484	3.9198	0.0861	1.7438
10	0.2301	3.8697	0.0364	1.498
12	0.2103	3.717	0.0191	1.2048
14	0.1824	3.6329	0.0164	1.0628
16	0.1726	3.5169	0.0102	0.7508
18	0.1493	3.3761	0.01	0.5543

Results and findings

As expected, the MRF model of a ternary inverter outperforms the conventional CMOS design. In particular, the improvement in BER is 60% - 94% and for the KLD values, the improvement is 55% - 83%. Fig. 5.6 illustrates strong noise-immunity of the proposed noise-tolerant implementation of the MRF model, compared to a conventional design, given the fixed SNR value 10dB of the input signal.

The distribution of the output voltage probability for the proposed noise-tolerant ternary inverter is measured, given the input signal SNR of 9dB and 16dB (Fig. 5.7). The observation, that random variations of voltage are concentrated around $0V$, $0.15V$ and $0.3V$, corresponding to the logical values “0”, “1”, and “2”, respectively. Hence, both the theoretical MRF model with Gibbs distribution (Table 5.1) and the measured

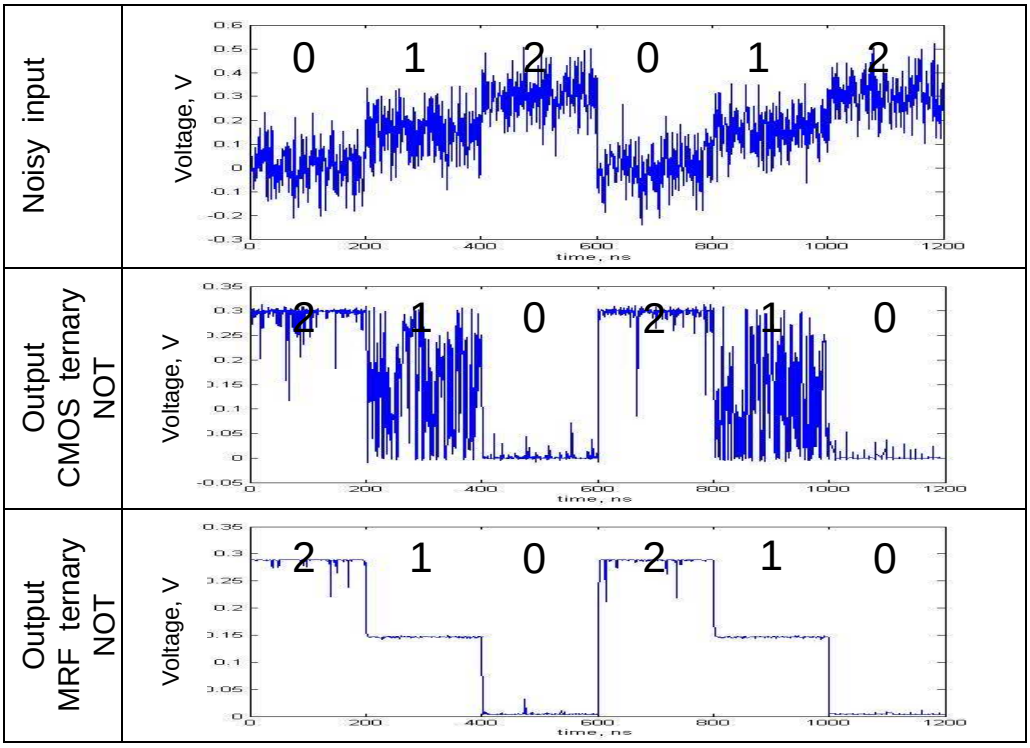


Figure 5.6: The output of a conventional ternary CMOS NOT gate, and its noise-tolerant MRF-based model, given the noisy input signals.

probabilistic distribution at the output of CMOS circuit, which operates accordingly to the maximum likelihood principle, exhibit a very close resemblance.

Set 2 of experiments: comparison in area, power and delay

In Table 5.3, the power dissipation, delay and the number of transistors are shown for both CMOS and MRF model of the inverter. The operating voltage for the CMOS based design is $0.7V$, as indicated by the ITRS roadmap [7]. The noise-tolerant ternary inverter operated in the subthreshold region, with the transistor driving voltages being $0.3V$ and $0.15V$, correspondingly.

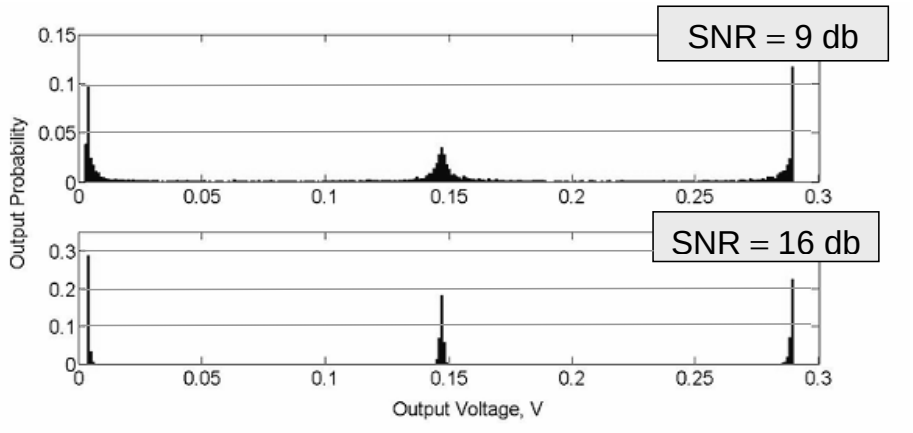


Figure 5.7: Output voltage probability distribution of a noise-tolerant ternary inverter.

Table 5.3: Comparison of CMOS conventional and noise-tolerant ternary inverter.

Parameter	**Conventional design**	**MRF based design**
Supply voltage (V_{dd}) (V)	0.7/0.35	0.3/0.15
Number of transistor	4	56
Power dissipation(nW)	0.378	0.048
Output delay (nS)	0.3334	0.6667

Results and findings

Even though the number of transistor increases 14 times compared to the conventional CMOS based design, the power dissipation does not increase linearly, given the small driving voltages. Instead, 87% reduction in power consumption is observed, on average. At the same time, the average delay for the MRF based ternary circuits is lesser, compared to the binary ones with the similar models from [47, 66].

5.5 Summary

The main result of this work is a simulated experimental evidence that MRF is an appropriate model of a ternary NOT logic gate with high noise-tolerant properties. The

theoretical contribution is an extension of the MRF model of binary logic gates and networks developed in [56, 47, 22] towards a noise-tolerant ternary NOT gate. The preliminary results show, that the generalization of the MRF model toward the two-input ternary logic gates is straightforward as well. The obtained results can be useful for advanced future technologies, characterized by high level of noise due to nanoscale phenomena.

Chapter 6

Conclusion and Future Works

"Nanotechnology is the base technology of an industrial revolution in the 21st century. Those who control nanotechnology will lead the industry."

-**Michiharu Nakamura**, Executive VP at Hitachi

6.1 Summary

As electronics technology migrates from micron and sub-micron towards the nano dimension, the question of downscaled circuit reliability arises. The device intrinsic noise becomes much severe with the minimized supply voltage for the low-power applications. This dissertation addresses this question, and answers it through the proposed probabilistic noise-immune design architecture.

A novel structure on Binary Decision Diagram is proposed and implemented on SPICE as a noise immune design method that can be readily implemented on wrap-gate nano wires. Though, not significantly different from the models of Markov Random Field (MRF) based design on CMOS gates [47], this thesis takes a completely different data structure and proposes a different reinforcer routing for robust noise-tolerant computing. The reliable circuit design method in the multivalued domain is also addressed in this thesis. The key contributions of this thesis are as follows:

- It proposes the implementation of the Markov Random Field based probabilistic model of logic gates, using novel Binary Decision Diagrams (BDD) with feedback. The choice of BDD structure is motivated by the fact that they can be implemented on the wrap-gate nano-wires [33]

- It studies the MRF based model not only for the elementary logic gates but also a network of gates. For that, another version of the Cyclic BDD is proposed called Shared Cyclic BDD.

- It performs the simulation of the proposed circuits in the SPICE environment, using the 16nm predictive transistor model from Berkeley.

- It compares the proposed probabilistic models as the Bit Error rate, Kullback-Liebler Distance, power, area and delay are used as the comparison parameters for the experiments.

- It introduces the noise-immune design method in the multivalued domain, as it extends the idea of Markov Random Field to the Ternary valued circuits for reliable design. The circuit for a ternary inverter is designed based on MRF principle and compared against the CMOS based design in noisy environment.

6.2 Future work

Both Markov Random Field and Hopfield models are based on Gibbs sampler algorithm [57, 49, 61, 62] (excellent introduction to these models can be find in textbook [60]). These models are characterized by (a) the same mechanism of embedding logic function via compatibility function, and (b) maximum-likelihood computing strategy or Gibbs sampler. Their combination working together may constitute an efficient hardware implementation, using iterative adaptive scheme via a multiple feedback loop network.

The most urgent plan is to test the proposed model against the implementation on the existing nanowire networks [33], which is being pursued in collaboration with the Research Center for Quantum Information Sciences at Hokkaido University.

There are several open problems in design of this new type of noise-tolerant devices,

such as:

1. Sum of minterms, $\sum_j \text{Minterm}_j$, is the simplest arithmetic form of logic functions (binary and multivalued) to represent the clique energy function in the MRF model. The effectiveness of other arithmetic forms, such as arithmetic, Walsh, and Haar spectrum [74, 48] has not been studied yet.

2. There are other architectures of the MRF model implementations.

3. Application of the MRF model for the multivalued logic functions can be useful for predictive technologies.

Another open research area is the study of the equivalence between three basic probabilistic models, which can be established using the Gibbs distribution or Gibbs sampling method, Bayesian network, Boltzmann machine, and MRF [18, 64]. It follows from this similarity that we can expect similar numerical results while using various models. However, the techniques for achieving these results, algorithms, behavioral characteristics and hardware implementation are different. This may become an useful development towards another nanoscale architecture, neuromorphic arrays [83].

Bibliography

[1] Kish, L. B., End of Moores law: thermal (noise) death of integration in micro and nano electronics, *Physics Letters A*, vol. 305, pp. 144149, Dec. 2002.

[2] Ron M Roth et al, Defect-tolerant demultiplexer circuits based on threshold logic and coding, *Nanotechnology*, Volume 20, Number 13, 2009

[3] Burnett, D., Higman, J., Hoefler, A., Li, C., and Kuhn, P., Variation in natural threshold voltage of NVM circuits due to dopant fluctuations and its impact on reliability, *Int. Electron Devices Meeting*, pp. 529532, Dec. 2002.

[4] Scholten, A.J.; Tiemeijer, L.F.; van Langevelde, R.; Havens, R.J.; Zegers-van Duijnhoven, A.T.A.; Venezia, V.C.; , "Noise modeling for RF CMOS circuit simulation,"*Electron Devices, IEEE Transactions on* , vol.50, no.3, pp. 618- 632, March 2003

[5] M.A. Breuer, S.K. Gupta and T. M. Mak, Defect and Error tolerance in the presence of massive number of defects, *IEEE Des. Test*, 21:216-227, May 2004

[6] R. W. Keyes, Fundamental limits of Silicon Technology, *In Proceedings of the IEEE*, volume 89, pages 227-239, IEEE Press, 2001.

[7] ITRS, http://www.itrs.net/reports.html, 2012.

[8] G. Cheng, P. Siles, F. Bi, C. Cen, D. Bogorin, C. Bark, C. Folkman, J. Park, C. Eom, G. Ribeiro, J. Levy. Sketched oxide single-electron transistor. *Nature Nanotechnology*, 2011

[9] Mahmoodi-Meimand H. Low power, robust, and high performance circuit design in nano-scale CMOS. *United States – Indiana: Purdue University*; 2005

[10] Wakerly, J.F. , Microcomputer reliability improvement using triple-modular redundancy, *Proceedings of the IEEE*, Volume:64 Issue:6, June 1976.

[11] Pelloie, J. L., Using SOI to achieve low-power consumption in digital, in *Proc. SOI Conf.*, pp. 1417, Oct. 2005.

[12] Markovic, D., Stojanovic, V., Nikolic, B., Horowitz, M. A., and Brodersen, R. W., Methods for true energy-performance optimization, *IEEE J. Solid-State Circuits*, vol. 39, no. 8, pp. 12821293, 2004.

[13] Palem, K. V., Proof as experiment:probabilistic algorithms from a thermodynamic perspective, in *Proc. Int. Symposium on Verification (Theory and Practice)*, pp. 524 547, June 2003.

[14] Palem, K. V., Energy aware computing through probabilistic switching: A study of limits, *IEEE Trans. Computer*, vol. 54, pp. 11231137, Sept. 2005.

[15] Korkmaz, P.; Akgul, B.E.S.; Palem, K.V.; , "Energy, Performance, and Probability Tradeoffs for Energy-Efficient Probabilistic CMOS Circuits," *Circuits and Systems I: Regular Papers, IEEE Transactions on* , vol.55, no.8, pp.2249-2262, Sept. 2008

[16] G. V. Varatkar, S. Narayanan, N. R. Shanbhag, and D. L. Jones, Stochastic networked computation, *IEEE Trans. VLSI Systems*, vol. 18, no.10, pp. 1421–1432, 2010.

[17] S. E. Lyshevski, V. P. Shmerko, M. A. Lyshevski, and S. N. Yanushkevich, Neuronal processing, reconfigurable neural networks and stochastic computing, *Proc. 8th IEEE Conf. Nanotechnology*, Arlington, TX, pp. 717–720, 2008.

[18] A. H. Tran, S. Yanushkevich, S. Lyshevski and V. Shmerko, Design of neuromorphic

logic networks and fault-tolerant computing, *Proc. 11th IEEE Int. Conf. Nanotechnology,* Portland, pp. 457–462, 2011.

[19] S. E. Lyshevski, S. N. Yanushkevich, V. P. Shmerko, and V. Geurkov, Computing Paradigms for Logic Nanocells. *J. Computational and Theoretical Nanoscience*, vol. 5, pp. 2377–2395, 2008.

[20] Nepal, K.; Bahar, R.I.; Mundy, J.; Patterson, W.R.; Zaslavsky, A.; , "Designing logic circuits for probabilistic computation in the presence of noise," *Design Automation Conference, 2005. Proceedings. 42nd* , vol., no., pp. 485- 490, 13-17 June 2005

[21] K. Nepal et al., Designing Logic Circuits for Probabilistic Computation in the Presence of Noise, *Proc. 42nd Design Automation Conf.* , ACM Press, 2005, pp. 485–490.

[22] I-C. Wey, Y-G. Chen, C.-H. Yu, et. al. Design and implementation of cost-effective probabilistic-based noise-talerant VLSI circuits, *IEEE Trans. Circuits and Systems-I*, vol. 56, no.11, pp. 2411–2424, 2009.

[23] Kaeslin, Hubert (2008), Digital Integrated Circuit Design, from VLSI Architectures to CMOS Fabrication, *Cambridge University Press*, section 14.2.

[24] B. Rajendran et al, CMOS transistor processing compatible with monolithic 3D integration, *Proceedings of the VLSI Multi Level Interconnect Conference*, 2005, Page(s): 76-82.

[25] B. Rajendran, "Sequential 3D IC Fabrication: Challenges and Prospects,"in *IEEE Trans. on Electron Devices*, 2010.

[26] M. Fuechsle, J. Miwa, S. Mahapatra, H. Ryu, S. Lee, O. Warschkow, L. Hollenberg, G. Klimeck and M. Simmons, A single-atom transistor, *Nature Nanotechnology 7*, 242246 (2012).

[27] Lei Wang; Shanbhag, N.R.; , "Noise-tolerant dynamic circuit design," *Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on* , vol.1, no., pp.549-552 vol.1, Jul 1999

[28] Jie Deng., et al., "Carbon Nanotube Transistor Circuits: Circuit-Level Performance Benchmarking and Design Options for Living with Imperfections", *In Proc. of IEEE ISSCC*, 2007.

[29] P.D. Lent, C.S. Tougaw, A device architecture for computing with quantum dots, In *proceedings of IEEE*, volume 85, pages 541-557, Press, 1997

[30] Likharev, K.K.; , "Single-electron devices and their applications," *Proceedings of the IEEE* , vol.87, no.4, pp.606-632, Apr 1999

[31] M. Stan, P. Franzon, S. Goldstein, J. Lach, and M. Ziegler, Molecular Electronics: From Devices and Interconnect to Circuits and Architecture, *Proceedings of the IEEE*, 91(11), November, 2003

[32] M. Mongillo, P. Spathis, G. Katsaros, P. Gentile, S. Franceschi, Multifunctional Devices and Logic Gates With Undoped Silicon Nanowires, arxiv.org/abs/1208.1465

[33] H. Hasegawa, S. Kasai, and T. Sato, Hexagonal BDD quantum circuit approach for ultra-low power III-V quantum LSls, *IEICE Trans. Electron.*, vol. ES7-C, no.11, pp. 1757–1768, 2004.

[34] J. S. Yedidia, W. T.Freeman and Y. Weiss. (2001). Understanding Belief Propagation and Its Generalizations. published as chapter 8 of 'Exploring Articial Intelligence in the New Millennium eds. G. Lakemeyer and B. Nebel, pp. 239-269, Morgan Kaufmann 2003

[35] Kuekes P J, RobinettW, Seroussi G andWilliams, Defect-tolerant demultiplexers for nano-electronics constructed from error-correcting codes *Appl. Phys. A* 80, 2005

[36] T. Rejimon, K. Lingasubramanian, S. Bhanja, Probabilistic error modeling for nano-domain logic circuits, *IEEE Trans VLSI*, 17 (1) (2009), pp. 5565

[37] Hamamatsu, M.; Tsuchiya, T.; Kikuno, T.; , "On the Reliability of Cascaded TMR Systems," Dependable Computing (PRDC), *2010 IEEE 16th Pacific Rim International Symposium on* , vol., no., pp.184-190, 13-15 Dec. 2010

[38] Han, J.; Jonker, P.; , "From massively parallel image processors to fault-tolerant nanocomputers," *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on* , vol.3, no., pp. 2- 7 Vol.3, 23-26 Aug. 2004

[39] M.G. Karpovsky, R.S. Stankovic, and J.T. Astola, *Spectral Logic and its Applications for the Design of Digital Devices,* Wiley, NJ, 2008.

[40] Chen He; Jacome, M.F.; de Veciana, G.; , "A reconfiguration-based defect-tolerant design paradigm for nanotechnologies," *Design & Test of Computers, IEEE* , vol.22, no.4, pp. 316- 326, July-Aug. 2005

[41] H. Rao; J. Chen; C. Yu; W. Ang; I. Wey; A. Wu; H. Zhao; , "Ensemble Dependent Matrix Methodology for Probabilistic-Based Fault-tolerant Nanoscale Circuit Design," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on* , vol., no., pp.1803-1806, 27-30 May 2007

[42] J. E. Besag, Spatial interaction and the statistical analysis of lattice systems, *J. Roy. Stat. Soc.,* series B, vol. 36, no. 3, 1974, pp. 192–236.

[43] R. I. Bahar, H. Y. Song, K. Nepal, J. Grodstein, Symbolic Failure Analysis of Complex CMOS Circuits due to Excessive Leakage Current and Charge Sharing,

IEEE Transactions on Computer-Aided Design of Integrated Circuits, Vol. 24, No. 5, April 2005, pp. 502-515

[44] R. Gandikota, Crosstalk Noise Analysis for Nano-Meter VLSI Circuits (Doctoral Dissertation), *University of Michigan, Ann Arbor*, 2009

[45] Y. Chen, M. Ventra, Shot noise in nanoscale conductors from first principles, *Phys. Rev. B*, vol. 67, Issue. 15, 2003.

[46] S. Z. Li, Markov Random Field Modeling in Computer Vision, *Springer-Verlag New York, Inc. Secaucus*, NJ, USA 1995

[47] K. Nepal, R. I. Bahar, J. Mundy, W. R. Patterson, and A. Zaslavsky, Designing nanoscale logic circuits based on Markov random fields, *J. Electronic Testing: Theory and Applications*, vol. 23, pp. 255–266, 2007.

[48] S. N. Yanushkevich, D. M. Miller, V. P. Shmerko, and R. S. Stanković, *Decision Diagram Techniques for Micro- and Nanoelectronic Design Handbook*, CRC Press, Taylor & Francis Group, Boca Raton, FL, 2006.

[49] S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, 1984.

[50] Keunwoo Kim; Das, K.K.; Joshi, R.V.; Ching-Te Chuang; , "Nanoscale CMOS Circuit Leakage Power Reduction by Double-Gate Device," *Low Power Electronics and Design, Proceedings of the 2004 International Symposium on* , vol., no., pp.102-107, 11-11 Aug. 2004

[51] P. J. Kuekes, W. Robinett, and R. S. Williams, Defect tolerance in resistor-logic demultiplexers for nanoelectronics, *Nanotechnology*, vol. 17, pp. 2466–2474, 2006.

[52] H. Astola, S. Stanković, and J. T. Astola, Error-correcting decision diagrams, *Proc. 3rd Workshop on Information Theoretic Methods in Science and Engineering*, Tampere, Finland, Aug., 2010.

[53] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1997.

[54] T. Mohamed, S. N. Yanushkevich, and S. Kasai, Fault-Tolerant Nanowire BDD circuits, *The First International Workshop on Physics and Computing in nanoscale Photonics and Materials*, Orleans, France, September 2012.

[55] V. P. Shmerko, S. N. Yanushkevich, and S. E. Lyshevski, *Computer Arithmetics for Nanoelectronics*, Taylor & Francis/CRC Press, Boca Raton, FL, 2009.

[56] R.I. Bahar, J. Chen, and J. Mundy, A probabilistic-based design for nanoscale computation. In: S. Shukla, and R.I. Bahar, Eds. *Nano, Quantum and Molecular Computing: Implications to High Level Design and Validation*, Kluwer, 2004.

[57] H. Derin and P. A. Kelly, Discrete-index Markov-type random process, *Proceedings of the IEEE*, vol. 77, no. 10, pp. 1485–1510, 1989.

[58] E. Fredkin and T. Toffoli, Conservative logic, *Int. J. Theoretical Physics,*, vol. 21, Nos. 3/4, pp. 219-253, 1982.

[59] B. J. Frey and N. Jojic, A comparison of algorithms for inference and learning in probabilistic graphical models, *IEEE Trans. Pattern Analysis and Machine Intelligance*, vol. 27, no. 9, pp. 1392-1416, 2005.

[60] S. Haykin, *Neural Networks and Leaning Machines*, third edition, Pearson Education, Upper Saddle River, NY, 2009.

[61] G. E. Hinton, Deterministic Boltzmann machine learning performs steepest descent in weight-space, *Neural Computation*, vol. 1, pp. 143–150, 1989.

[62] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of National Academy of Sciences*, USA, vol. 79, pp. 2554–2558, 1982.

[63] A. Lin, N. Patil, H. Wei, S. Mitra and H.-S.P. Wong, A Metallic-CNT-tolerant design methodology for carbon nanotube VLSI: Concepts and experimental demonstration, *IEEE Trans. Electron Devices*, 2009.

[64] S. E. Lyshevski, S. N. Yanushkevich, V. P. Shmerko, et. al., Computing paradigms for logic nanocells. *J. Computational and Theoretical Nanoscience*, vol. 5, pp. 2377–2395, 2008.

[65] S. Minato, N. Ishiura and S. Yajima, Shared binary decision diagram with attributed edges for efficient Boolean function manipulation, *Proc. 27th ACM/IEEE Design Automation Conf.*, 1990.

[66] S. N. Yanushkevich, G. Tangim, S. Kasai, S. E. Lyshevski, V. P. Shmerko, Design of nanoelectronic ICs: Noise-tolerant logic based on cyclic BDD, *Proc. IEEE Nanotechnology Conf.*, Birmingham, UK, August 2012.

[67] Seiya Kasai, Tatsuya Nakamura and Yuta Shiratori Multiple path switching device utilizing size-controlled nano-Schottky wrap gates for MDD-based logic circuits. *Journal of Multiple-Valued Logic and Soft Computing* vol.13(3), 267-277, 2007

[68] R. Keyes, "The evolution of digital electronics towards VLSI," *IEEE J. Solid-State Circuits*, vol. SC-14, pp. 193-201, Apr. 1979.

[69] K.C. Smith, The prospects for multivalued logic: A technology and applications view, *IEEE Trans. Comput.*, vol. C-30, pp. 619-634, Sept. 1981.

[70] R. Mariani, F. Pessolano, , and R. Saletti: 'A new CMOS ternary logic design for low-power low-voltage circuits', *PATMOS, Louvain-la-Neuve*, Belgium, September 1997

[71] S. Lin, Y-B Kim, and F. Lombardi, A novel CNTFET-based ternary logic gate design, In *IEEE International Midwest Symposium on Circuits and Systems*, pp.435-438, 2009.

[72] S. Lin, Y-B Kim, and F. Lombardi, Design of a Ternary Memory Cell Using CNTFETs, *IEEE Transactions on Nanotechnology*, vol. 11, no. 5, september 2012

[73] Z. Kamar and K. Nepal, Noise margin-optimized ternary CMOS SRAM delay and sizing characteristics, in *Proc. IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2010, pp. 801804.

[74] V. P. Shmerko, S. N. Yanushkevich, and S. E. Lyshevski, *Computer Arithmetics for Nanoelectronics*, Taylor & Francis/CRC Press, Boca Raton, FL, 2009.

[75] V. Shmerko, S. Yanushkevich, V. Levashenko, and I. Bondar, Techniques of Computing Logic Derivatives for MVL Functions, *Proc IEEE. 26th Int. Symp. Multiple-Valued Logic*, Spain, pp. 267-272,1996.

[76] Balla, A. Antoniou, Low power dissipation MOS ternary logic family, *IEEE J. Solid-State Circuits*, vol. SC-19, pp. 739-749, 1984.

[77] S. Kullback, *Information Theory and Statistics*, Gloucester, MA, Peter Smith, 1968.

[78] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, Oxford University Press, New York, 2004.

[79] Z. Kamar and K. Nepal, Noise margin-optimized ternary CMOS SRAM delay and sizing characteristics, *Proc. IEEE Int. Midwest Symp. Circuits Syst.*, pp. 801–804, 2010.

[80] X.W. Wu and F.P. Prosser, CMOS ternary logic circuits, *IEE Proceedings*, vol. 137, Pt. G, no. I, pp. 20–27, 1990

[81] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*, 5th ed., Willey, 2011.

[82] S. Yanushkevich, D. Miller, V. Shmerko, R. Stankovich, Decisin Diagram Techniques for Micro and NanoElectronic Design, *Taylor & Francis*, 2006

[83] Trel , Lee JH, Ma X, Likharev KK., Neuromorphic architectures for nanoelectronic circuits, *International Journal of Circuit Theory and Applications* 2004; 32:277302.

Appendix A

MATLAB Codes for Performance Evaluation

A.1 Noisy Signal Generation

```
close all;
    clear all; fileID=fopen('value.txt','wt');
    n = 1:1600;
    x = [zeros(1,400),ones(1,400)];
    x1=[x,x];
    x1=x1*0.3;
    y1 = awgn(x1,3,'measured');
    figure(1)
    plot(n,y1);
    xlabel('time, ns')
    ylabel('voltage, v')


    fprintf(fileID,'%12.4f)n',x1);
```

A.2 Calculation of BER and KLD

```
close all;
    clear all;
    [a1,b1] = textread('nand_cmos_3.txt','%f %f');
    c1=a1.*1000000000;
    figure(1)
    plot(c1,b1);
    xlabel('time, ns')
    ylabel('voltage, v')

    [a2,b2] = textread('nand_cmos_5.txt','%f %f');
    c2=a1.*1000000000;
    figure(2)
    plot(c2,b2);
    xlabel('time, ns')
    ylabel('voltage, v')

    [a3,b3] = textread('nand_cmos_9.txt','%f %f');
    c3=a1.*1000000000;
    figure(3)
    plot(c3,b3);
    xlabel('time, ns')
    ylabel('voltage, v')

    [a4,b4] = textread('nand_cmos_12.txt','%f %f');
    c4=a1.*1000000000;
    figure(4)
    plot(c4,b4);
    xlabel('time, ns')
    ylabel('voltage, v')

    [a5,b5] = textread('nand_cmos_10.txt','%f %f');
    c5=a1.*1000000000;
    figure(5)
```

```
plot(c5,b5);
xlabel('time, ns')
ylabel('voltage, v')

[a6,b6] = textread('nand_cmos_12.txt','%f %f');
c6=a1.*1000000000;
figure(6)
plot(c6,b6);
xlabel('time, ns')
ylabel('voltage, v')

x = [ones(1,2649),zeros(1,884),ones(1,2648),zeros(1,884)];
figure(11)
plot(c1,x);
b6_avg=(max(b6)+min(b6))/2;
for i=1:length(b6)
if b6(i)>=b6_avg;
b6(i)=1;
else b6(i)=0;
end
end
figure(8)
plot(c1,b6);
y=0;
for i=1:length(b6)
if b6(i)==x(i);
y==y;
else y=y+1;
end
end
BER=y/7065

d_1 =-0.08:0.001:0.35;
[count_1 bins_1] = hist(b1,d_1);
count_1_nor=count_1/sum(count_1);
figure(5)
bar(bins_1,count_1_nor)
d_2 =-0.08:0.001:0.35;
[count_2 bins_2] = hist(b2,d_2);
count_2_nor=count_2/sum(count_2);
figure(6)
bar(bins_2,count_2_nor)
d_3 =-0.08:0.001:0.35;
[count_3 bins_3] = hist(b3,d_3);
count_3_nor=count_3/sum(count_3);
figure(7)
bar(bins_3,count_3_nor)
d_4 =-0.08:0.001:0.35;
[count_4 bins_4] = hist(b4,d_4);
count_4_nor=count_4/sum(count_4);
figure(8)
bar(bins_4,count_4_nor)
d_5 =-0.08:0.001:0.35;
[count_5 bins_5] = hist(b5,d_5);
count_5_nor=count_5/sum(count_5);
figure(9)
bar(bins_5,count_5_nor)
d_6 =-0.08:0.001:0.35;
[count_6 bins_6] = hist(b6,d_6);
count_6_nor=count_6/sum(count_6);
figure(10)
bar(bins_6,count_6_nor)
```

Appendix B

SPICE codes for System Design

B.1 CMOS based NAND design

```
* CMOS NAND @ 16nm
    VDD 1 0 0.3
    V1 2 0 PWL .INCLUDE input1.txt
    V2 3 0 PWL .INCLUDE input2.txt
    MQ1 1 2 4 1 P1 (L=16nm)
    MQ2 1 3 4 1 P1 (L=16nm)
    MQ3 4 2 5 5 N1 (L=16nm)
    MQ4 5 3 0 0 N1 (L=16nm)
    .model N1 nmos level = 14
    +version = 4.6.5 binunit = 1 paramchk= 1 mobmod = 0 +capmod = 2 igcmod = 1 igbmod = 1 geomod = 1 +diomod
= 1 rdsmod = 0 rbodymod= 1 rgatemod= 1 +permod = 1 acnqsmod= 0 trnqsmod= 0 +tnom = 27 toxe = 9.5e-010 toxp =
7e-010 toxm = 9.5e-010 +dtox = 2.5e-010 epsrox = 3.9 wint = 5e-009 lint = 1.45e-009 +ll = 0 wl = 0 lln = 1 wln
= 1 +lw = 0 ww = 0 lwn = 1 wwn = 1 +lwl = 0 wwl = 0 xpart = 0 toxref = 9.5e-010 +xl = -6.5e-9 +vth0 = 0.47965
k1 = 0.4 k2 = 0 k3 = 0 +k3b = 0 w0 = 2.5e-006 dvt0 = 1 dvt1 = 2 +dvt2 = 0 dvt0w = 0 dvt1w = 0 dvt2w = 0 +dsub
= 0.1 minv = 0.05 voffl = 0 dvtp0 = 1e-011 +dvtp1 = 0.1 lpe0 = 0 lpeb = 0 xj = 5e-009 +ngate = 1e+023 ndep =
7e+018 nsd = 2e+020 phin = 0 +cdsc = 0 cdscb = 0 cdscd = 0 cit = 0 +voff = -0.13 nfactor = 2.3 eta0 = 0.0032
etab = 0 +vfb = -0.55 u0 = 0.03 ua = 6e-010 ub = 1.2e-018 +uc = 0 vsat = 290000 a0 = 1 ags = 0 +a1 = 0 a2 =
1 b0 = 0 b1 = 0 +keta = 0.04 dwg = 0 dwb = 0 pclm = 0.02 +pdiblc1 = 0.001 pdiblc2 = 0.001 pdiblcb = -0.005 drout
= 0.5 +pvag = 1e-020 delta = 0.01 pscbe1 = 8.14e+008 pscbe2 = 1e-007 +fprout = 0.2 pdits = 0.01 pditsd = 0.23
pditsl = 2300000 +rsh = 5 rdsw = 140 rsw = 75 rdw = 75 +rdswmin = 0 rdwmin = 0 rswmin = 0 prwg = 0 +prwb = 0
wr = 1 alpha0 = 0.074 alpha1 = 0.005 +beta0 = 30 agidl = 0.0002 bgidl = 2.1e+009 cgidl = 0.0002 +egidl = 0.8
aigbacc = 0.012 bigbacc = 0.0028 cigbacc = 0.002 +nigbacc = 1 aigbinv = 0.014 bigbinv = 0.004 cigbinv = 0.004
+eigbinv = 1.1 nigbinv = 3 aigc = 0.0213 bigc = 0.0025889 +cigc = 0.002 aigsd = 0.0213 bigsd = 0.0025889 cigsd
= 0.002 +nigc = 1 poxedge = 1 pigcd = 1 ntox = 1 +xrcrg1 = 12 xrcrg2 = 5 +cgso = 5e-011 cgdo = 5e-011 cgbo =
2.56e-011 cgdl = 2.653e-010 +cgsl = 2.653e-010 ckappas = 0.03 ckappad = 0.03 acde = 1 +moin = 15 noff = 0.9
voffcv = 0.02 +kt1 = -0.11 kt1l = 0 kt2 = 0.022 ute = -1.5 +ua1 = 4.31e-009 ub1 = 7.61e-018 uc1 = -5.6e-011
prt = 0 +at = 33000 +fnoimod = 1 tnoimod = 0 +jss = 0.0001 jsws = 1e-011 jswgs = 1e-010 njs = 1 +ijthsfwd= 0.01
ijthsrev= 0.001 bvs = 10 xjbvs = 1 +jsd = 0.0001 jswd = 1e-011 jswgd = 1e-010 njd = 1 +ijthdfwd= 0.01 ijthdrev=
0.001 bvd = 10 xjbvd = 1 +pbs = 1 cjs = 0.0005 mjs = 0.5 pbsws = 1 +cjsws = 5e-010 mjsws = 0.33 pbswgs = 1 cjswgs
= 3e-010 +mjswgs = 0.33 pbd = 1 cjd = 0.0005 mjd = 0.5 +pbswd = 1 cjswd = 5e-010 mjswd = 0.33 pbswgd = 1 +cjswgd
= 5e-010 mjswgd = 0.33 tpb = 0.005 tcj = 0.001 +tpbsw = 0.005 tcjsw = 0.001 tpbswg = 0.005 tcjswg = 0.001 +xtis
= 3 xtid = 3 +dmcg = 0 dmci = 0 dmdg = 0 dmcgt = 0 +dwj = 0 xgw = 0 xgl = 0 +rshg = 0.4 gbmin = 1e-010 rbpb
= 5 rbpd = 15 +rbps = 15 rbdb = 15 rbsb = 15 ngcon = 1
    .model P1 pmos level = 14
    +version = 4.6.5 binunit = 1 paramchk= 1 mobmod = 0 +capmod = 2 igcmod = 1 igbmod = 1 geomod = 1 +diomod
= 1 rdsmod = 0 rbodymod= 1 rgatemod= 1 +permod = 1 acnqsmod= 0 trnqsmod= 0 +tnom = 27 toxe = 1e-009 toxp = 7e-010
toxm = 1e-009 +dtox = 3e-010 epsrox = 3.9 wint = 5e-009 lint = 1.45e-009 +ll = 0 wl = 0 lln = 1 wln = 1 +lw
= 0 ww = 0 lwn = 1 wwn = 1 +lwl = 0 wwl = 0 xpart = 0 toxref = 1e-009 +xl = -6.5e-9 +vth0 = -0.43121 k1 = 0.4
k2 = -0.01 k3 = 0 +k3b = 0 w0 = 2.5e-006 dvt0 = 1 dvt1 = 2 +dvt2 = -0.032 dvt0w = 0 dvt1w = 0 dvt2w = 0 +dsub
= 0.1 minv = 0.05 voffl = 0 dvtp0 = 1e-011 +dvtp1 = 0.05 lpe0 = 0 lpeb = 0 xj = 5e-009 +ngate = 1e+023 ndep
= 5.5e+018 nsd = 2e+020 phin = 0 +cdsc = 0 cdscb = 0 cdscd = 0 cit = 0 +voff = -0.126 nfactor = 2.1 eta0 = 0.0032
etab = 0 +vfb = 0.55 u0 = 0.006 ua = 2e-009 ub = 5e-019 +uc = 0 vsat = 250000 a0 = 1 ags = 1e-020 +a1 = 0 a2
= 1 b0 = 0 b1 = 0 +keta = -0.047 dwg = 0 dwb = 0 pclm = 0.12 +pdiblc1 = 0.001 pdiblc2 = 0.001 pdiblcb = 3.4e-008
drout = 0.56 +pvag = 1e-020 delta = 0.01 pscbe1 = 1.2e+009 pscbe2 = 8.0472e-007 +fprout = 0.2 pdits = 0.08 pditsd
= 0.23 pditsl = 2300000 +rsh = 5 rdsw = 140 rsw = 70 rdw = 70 +rdswmin = 0 rdwmin = 0 rswmin = 0 prwg = 0 +prwb
= 0 wr = 1 alpha0 = 0.074 alpha1 = 0.005 +beta0 = 30 agidl = 0.0002 bgidl = 2.1e+009 cgidl = 0.0002 +egidl =
0.8 aigbacc = 0.012 bigbacc = 0.0028 cigbacc = 0.002 +nigbacc = 1 aigbinv = 0.014 bigbinv = 0.004 cigbinv =
0.004 +eigbinv = 1.1 nigbinv = 3 aigc = 0.0213 bigc = 0.0025889 +cigc = 0.002 aigsd = 0.0213 bigsd = 0.0025889
cigsd = 0.002 +nigc = 1 poxedge = 1 pigcd = 1 ntox = 1 +xrcrg1 = 12 xrcrg2 = 5 +cgso = 5e-011 cgdo = 5e-011
cgbo = 2.56e-011 cgdl = 2.653e-010 +cgsl = 2.653e-010 ckappas = 0.03 ckappad = 0.03 acde = 1 +moin = 15 noff
= 0.9 voffcv = 0.02 +kt1 = -0.11 kt1l = 0 kt2 = 0.022 ute = -1.5 +ua1 = 4.31e-009 ub1 = 7.61e-018 uc1 = -5.6e-011
prt = 0 +at = 33000 +fnoimod = 1 tnoimod = 0 +jss = 0.0001 jsws = 1e-011 jswgs = 1e-010 njs = 1 +ijthsfwd= 0.01
ijthsrev= 0.001 bvs = 10 xjbvs = 1 +jsd = 0.0001 jswd = 1e-011 jswgd = 1e-010 njd = 1 +ijthdfwd= 0.01 ijthdrev=
```

```
0.001 bvd = 10 xjbvd = 1 +pbs = 1 cjs = 0.0005 mjs = 0.5 pbsws = 1 +cjsws = 5e-010 mjsws = 0.33 pbswgs = 1 cjswgs
= 3e-010 +mjswgs = 0.33 pbd = 1 cjd = 0.0005 mjd = 0.5 +pbswd = 1 cjswd = 5e-010 mjswd = 0.33 pbswgd = 1 +cjswgd
= 5e-010 mjswgd = 0.33 tpb = 0.005 tcj = 0.001 +tpbsw = 0.005 tcjsw = 0.001 tpbswg = 0.005 tcjswg = 0.001 +xtis
= 3 xtid = 3 +dmcg = 0 dmci = 0 dmdg = 0 dmcgt = 0 +dwj = 0 xgw = 0 xgl = 0 +rshg = 0.4 gbmin = 1e-010 rbpb
= 5 rbpd = 15 +rbps = 15 rbdb = 15 rbsb = 15 ngcon = 1
    .OPTION TEMP=27
    .TRAN 1ns 1600ns 0s 1ns
    .END
```

B.2 MRF based design from [47]

```
* inverter design using the switching theory in the inputs VDD 1 0 0.3
    V1 2 0 PWL .INCLUDE input1.txt
    V2 3 0 PWL .INCLUDE input2.txt
    .SUBCKT not_bin vcc n_in n_out
    MQ1 vcc n_in n_out vcc P1 (L=16nm)
    MQ2 n_out n_in 0 0 N1 (L=16nm)
    .ENDS not_bin
    .SUBCKT nand vcc n_in1 n_in2 n_out
    MQ1 vcc n_in1 n_out vcc P1 (L=16nm)
    MQ2 vcc n_in2 n_out vcc P1 (L=16nm)
    MQ3 n_out n_in1 20 20 N1 (L=16nm)
    MQ4 20 n_in2 0 0 N1 (L=16nm)
    .ENDS nand
    .SUBCKT switch vcc n n_bar n_in1 n_in2 n_out
    MQ1 n_in1 n n_out n_out N1 (L=16nm)
    MQ2 n_in1 n_bar n_out n_in1 P1 (L=16nm)
    MQ3 n_in2 n_bar n_out n_out N1 (L=16nm)
    MQ4 n_in2 n n_out n_in2 P1 (L=16nm)
    .ENDS switch
    x1 1 4 15 not_bin
    x2 1 2 13 not_bin
    x3 1 4 15 12 2 3 switch
    x4 1 4 15 14 13 5 switch
    x5 1 3 7 8 nand
    x6 1 8 9 not_bin
    x7 1 5 6 10 nand
    x8 1 10 11 not_bin
    x9 1 9 14 not_bin
    x10 1 9 6 not_bin
    x11 1 11 7 not_bin
    x12 1 11 12 not_bin
    .model N1 nmos level = 14
    +version = 4.6.5 binunit = 1 paramchk= 1 mobmod = 0 +capmod = 2 igcmod = 1 igbmod = 1 geomod = 1 +diomod
= 1 rdsmod = 0 rbodymod= 1 rgatemod= 1 +permod = 1 acnqsmod= 0 trnqsmod= 0 +tnom = 27 toxe = 9.5e-010 toxp =
7e-010 toxm = 9.5e-010 +dtox = 2.5e-010 epsrox = 3.9 wint = 5e-009 lint = 1.45e-009 +ll = 0 wl = 0 lln = 1 wln
= 1 +lw = 0 ww = 0 lwn = 1 wwn = 1 +lwl = 0 wwl = 0 xpart = 0 toxref = 9.5e-010 +xl = -6.5e-9 +vth0 = 0.47965
k1 = 0.4 k2 = 0 k3 = 0 +k3b = 0 w0 = 2.5e-006 dvt0 = 1 dvt1 = 2 +dvt2 = 0 dvt0w = 0 dvt1w = 0 dvt2w = 0 +dsub
= 0.1 minv = 0.05 voffl = 0 dvtp0 = 1e-011 +dvtp1 = 0.1 lpe0 = 0 lpeb = 0 xj = 5e-009 +ngate = 1e+023 ndep =
7e+018 nsd = 2e+020 phin = 0 +cdsc = 0 cdscb = 0 cdscd = 0 cit = 0 +voff = -0.13 nfactor = 2.3 eta0 = 0.0032
etab = 0 +vfb = -0.55 u0 = 0.03 ua = 6e-010 ub = 1.2e-018 +uc = 0 vsat = 290000 a0 = 1 ags = 0 +a1 = 0 a2 =
1 b0 = 0 b1 = 0 +keta = 0.04 dwg = 0 dwb = 0 pclm = 0.02 +pdiblc1 = 0.001 pdiblc2 = 0.001 pdiblcb = -0.005 drout
= 0.5 +pvag = 1e-020 delta = 0.01 pscbe1 = 8.14e+008 pscbe2 = 1e-007 +fprout = 0.2 pdits = 0.01 pditsd = 0.23
pditsl = 2300000 +rsh = 5 rdsw = 140 rsw = 75 rdw = 75 +rdswmin = 0 rdwmin = 0 rswmin = 0 prwg = 0 +prwb = 0
wr = 1 alpha0 = 0.074 alpha1 = 0.005 +beta0 = 30 agidl = 0.0002 bgidl = 2.1e+009 cgidl = 0.0002 +egidl = 0.8
aigbacc = 0.012 bigbacc = 0.0028 cigbacc = 0.002 +nigbacc = 1 aigbinv = 0.014 bigbinv = 0.004 cigbinv = 0.004
+eigbinv = 1.1 nigbinv = 3 aigc = 0.0213 bigc = 0.0025889 +cigc = 0.002 aigsd = 0.0213 bigsd = 0.0025889 cigsd
= 0.002 +nigc = 1 poxedge = 1 pigcd = 1 ntox = 1 +xrcrg1 = 12 xrcrg2 = 5 +cgso = 5e-011 cgdo = 5e-011 cgbo =
2.56e-011 cgdl = 2.653e-010 +cgsl = 2.653e-010 ckappas = 0.03 ckappad = 0.03 acde = 1 +moin = 15 noff = 0.9
voffcv = 0.02 +kt1 = -0.11 kt1l = 0 kt2 = 0.022 ute = -1.5 +ua1 = 4.31e-009 ub1 = 7.61e-018 uc1 = -5.6e-011
prt = 0 +at = 33000 +fnoimod = 1 tnoimod = 0 +jss = 0.0001 jsws = 1e-011 jswgs = 1e-010 njs = 1 +ijthsfwd= 0.01
ijthsrev= 0.001 bvs = 10 xjbvs = 1 +jsd = 0.0001 jswd = 1e-011 jswgd = 1e-010 njd = 1 +ijthdfwd= 0.01 ijthdrev=
0.001 bvd = 10 xjbvd = 1 +pbs = 1 cjs = 0.0005 mjs = 0.5 pbsws = 1 +cjsws = 5e-010 mjsws = 0.33 pbswgs = 1 cjswgs
```

```
= 3e-010 +mjswgs = 0.33 pbd = 1 cjd = 0.0005 mjd = 0.5 +pbswd = 1 cjswd = 5e-010 mjswd = 0.33 pbswgd = 1 +cjswgd
= 5e-010 mjswgd = 0.33 tpb = 0.005 tcj = 0.001 +tpbsw = 0.005 tcjsw = 0.001 tpbswg = 0.005 tcjswg = 0.001 +xtis
= 3 xtid = 3 +dmcg = 0 dmci = 0 dmdg = 0 dmcgt = 0 +dwj = 0 xgw = 0 xgl = 0 +rshg = 0.4 gbmin = 1e-010 rbpb
= 5 rbpd = 15 +rbps = 15 rbdb = 15 rbsb = 15 ngcon = 1
    .model P1 pmos level = 14
    +version = 4.6.5 binunit = 1 paramchk= 1 mobmod = 0 +capmod = 2 igcmod = 1 igbmod = 1 geomod = 1 +diomod
= 1 rdsmod = 0 rbodymod= 1 rgatemod= 1 +permod = 1 acnqsmod= 0 trnqsmod= 0 +tnom = 27 toxe = 1e-009 toxp = 7e-010
toxm = 1e-009 +dtox = 3e-010 epsrox = 3.9 wint = 5e-009 lint = 1.45e-009 +ll = 0 wl = 0 lln = 1 wln = 1 +lw
= 0 ww = 0 lwn = 1 wwn = 1 +lwl = 0 wwl = 0 xpart = 0 toxref = 1e-009 +xl = -6.5e-9 +vth0 = -0.43121 k1 = 0.4
k2 = -0.01 k3 = 0 +k3b = 0 w0 = 2.5e-006 dvt0 = 1 dvt1 = 2 +dvt2 = -0.032 dvt0w = 0 dvt1w = 0 dvt2w = 0 +dsub
= 0.1 minv = 0.05 voffl = 0 dvtp0 = 1e-011 +dvtp1 = 0.05 lpe0 = 0 lpeb = 0 xj = 5e-009 +ngate = 1e+023 ndep
= 5.5e+018 nsd = 2e+020 phin = 0 +cdsc = 0 cdscb = 0 cdscd = 0 cit = 0 +voff = -0.126 nfactor = 2.1 eta0 = 0.0032
etab = 0 +vfb = 0.55 u0 = 0.006 ua = 2e-009 ub = 5e-019 +uc = 0 vsat = 250000 a0 = 1 ags = 1e-020 +a1 = 0 a2
= 1 b0 = 0 b1 = 0 +keta = -0.047 dwg = 0 dwb = 0 pclm = 0.12 +pdiblc1 = 0.001 pdiblc2 = 0.001 pdiblcb = 3.4e-008
drout = 0.56 +pvag = 1e-020 delta = 0.01 pscbe1 = 1.2e+009 pscbe2 = 8.0472e-007 +fprout = 0.2 pdits = 0.08 pditsd
= 0.23 pditsl = 2300000 +rsh = 5 rdsw = 140 rsw = 70 rdw = 70 +rdswmin = 0 rdwmin = 0 rswmin = 0 prwg = 0 +prwb
= 0 wr = 1 alpha0 = 0.074 alpha1 = 0.005 +beta0 = 30 agidl = 0.0002 bgidl = 2.1e+009 cgidl = 0.0002 +egidl =
0.8 aigbacc = 0.012 bigbacc = 0.0028 cigbacc = 0.002 +nigbacc = 1 aigbinv = 0.014 bigbinv = 0.004 cigbinv =
0.004 +eigbinv = 1.1 nigbinv = 3 aigc = 0.0213 bigc = 0.0025889 +cigc = 0.002 aigsd = 0.0213 bigsd = 0.0025889
cigsd = 0.002 +nigc = 1 poxedge = 1 pigcd = 1 ntox = 1 +xrcrg1 = 12 xrcrg2 = 5 +cgso = 5e-011 cgdo = 5e-011
cgbo = 2.56e-011 cgdl = 2.653e-010 +cgsl = 2.653e-010 ckappas = 0.03 ckappad = 0.03 acde = 1 +moin = 15 noff
= 0.9 voffcv = 0.02 +kt1 = -0.11 kt1l = 0 kt2 = 0.022 ute = -1.5 +ua1 = 4.31e-009 ub1 = 7.61e-018 uc1 = -5.6e-011
prt = 0 +at = 33000 +fnoimod = 1 tnoimod = 0 +jss = 0.0001 jsws = 1e-011 jswgs = 1e-010 njs = 1 +ijthsfwd= 0.01
ijthsrev= 0.001 bvs = 10 xjbvs = 1 +jsd = 0.0001 jswd = 1e-011 jswgd = 1e-010 njd = 1 +ijthdfwd= 0.01 ijthdrev=
0.001 bvd = 10 xjbvd = 1 +pbs = 1 cjs = 0.0005 mjs = 0.5 pbsws = 1 +cjsws = 5e-010 mjsws = 0.33 pbswgs = 1 cjswgs
= 3e-010 +mjswgs = 0.33 pbd = 1 cjd = 0.0005 mjd = 0.5 +pbswd = 1 cjswd = 5e-010 mjswd = 0.33 pbswgd = 1 +cjswgd
= 5e-010 mjswgd = 0.33 tpb = 0.005 tcj = 0.001 +tpbsw = 0.005 tcjsw = 0.001 tpbswg = 0.005 tcjswg = 0.001 +xtis
= 3 xtid = 3 +dmcg = 0 dmci = 0 dmdg = 0 dmcgt = 0 +dwj = 0 xgw = 0 xgl = 0 +rshg = 0.4 gbmin = 1e-010 rbpb
= 5 rbpd = 15 +rbps = 15 rbdb = 15 rbsb = 15 ngcon = 1
    .OPTION TEMP=27
    .TRAN 1ns 1600ns 0s 1ns
    .END
```

B.3 Ternnary MRF based Inverter Design

```
* TERNARY MRF INVERTER @ 16nm
    VDD1 1 0 0.3
    VDD2 4 0 0.15
    V1 2 0 PWL .INCLUDE ternary_in.txt
    .SUBCKT inv_cyclic vcc1 vcc2 n_in n_out1 n_out2
    MQ1 vcc2 n_in n_out1 vcc2 P2 (L=16nm)
    MQ2 n_out1 n_in 0 0 N2 (L=16nm)
    MQ3 vcc1 n_in n_out2 vcc1 P1 (L=16nm)
    MQ4 n_out2 n_in vcc2 vcc2 N1 (L=16nm)
    .ENDS inv_cyclic
    .SUBCKT inv_ter vcc1 vcc2 n_in n_out
    MQ1 vcc1 n_in n_out vcc1 P1 (L=16nm)
    MQ2 n_out n_in 0 0 N1 (L=16nm)
    MQ3 n_out n_in 5 n_out P2 (L=16nm)
    MQ4 5 n_in vcc2 vcc2 N2 (L=16nm)
    .ENDS inv_ter
    .SUBCKT nand_ter vcc1 vcc2 n_in1 n_in2 n_out
    MQ1 vcc1 n_in1 n_out vcc1 P1 (L=16nm)
    MQ2 vcc1 n_in2 n_out vcc1 P1 (L=16nm)
    MQ3 n_out n_in1 5 5 N1 (L=16nm)
    MQ4 5 n_in2 0 0 N1 (L=16nm)
    MQ5 n_out n_in1 6 n_out P2 (L=16nm)
    MQ6 n_out n_in2 6 n_out P2 (L=16nm)
    MQ7 6 n_in1 7 7 N2 (L=16nm)
    MQ8 7 n_in2 vcc2 vcc2 N2 (L=16nm)
    .ENDS nand_ter
    x1 1 4 2 3 5 inv_cyclic
    x2 1 4 6 7 8 inv_cyclic
```

```
    x3 1 4 5 6 9 nand_ter
    x4 1 4 3 7 10 nand_ter
    x5 1 4 2 8 11 nand_ter
    x6 1 4 11 12 inv_ter
    x7 1 4 12 6 inv_ter
    x8 1 4 10 13 inv_ter
    x9 1 4 13 7 inv_ter
    x10 1 4 9 14 inv_ter
    x11 1 4 14 8 inv_ter
    *output node 6......
    .model N1 nmos level = 14
    +version = 4.6.5 binunit = 1 paramchk= 1 mobmod = 0 +capmod = 2 igcmod = 1 igbmod = 1 geomod = 1 +diomod
= 1 rdsmod = 0 rbodymod= 1 rgatemod= 1 +permod = 1 acnqsmod= 0 trnqsmod= 0 +tnom = 27 toxe = 9.5e-010 toxp =
7e-010 toxm = 9.5e-010 +dtox = 2.5e-010 epsrox = 3.9 wint = 5e-009 lint = 1.45e-009 +ll = 0 wl = 0 lln = 1 wln
= 1 +lw = 0 ww = 0 lwn = 1 wwn = 1 +lwl = 0 wwl = 0 xpart = 0 toxref = 9.5e-010 +xl = -6.5e-9 +vth0 = 0.47965
k1 = 0.4 k2 = 0 k3 = 0 +k3b = 0 w0 = 2.5e-006 dvt0 = 1 dvt1 = 2 +dvt2 = 0 dvt0w = 0 dvt1w = 0 dvt2w = 0 +dsub
= 0.1 minv = 0.05 voffl = 0 dvtp0 = 1e-011 +dvtp1 = 0.1 lpe0 = 0 lpeb = 0 xj = 5e-009 +ngate = 1e+023 ndep =
7e+018 nsd = 2e+020 phin = 0 +cdsc = 0 cdscb = 0 cdscd = 0 cit = 0 +voff = -0.13 nfactor = 2.3 eta0 = 0.0032
etab = 0 +vfb = -0.55 u0 = 0.03 ua = 6e-010 ub = 1.2e-018 +uc = 0 vsat = 290000 a0 = 1 ags = 0 +a1 = 0 a2 =
1 b0 = 0 b1 = 0 +keta = 0.04 dwg = 0 dwb = 0 pclm = 0.02 +pdiblc1 = 0.001 pdiblc2 = 0.001 pdiblcb = -0.005 drout
= 0.5 +pvag = 1e-020 delta = 0.01 pscbe1 = 8.14e+008 pscbe2 = 1e-007 +fprout = 0.2 pdits = 0.01 pditsd = 0.23
pditsl = 2300000 +rsh = 5 rdsw = 140 rsw = 75 rdw = 75 +rdswmin = 0 rdwmin = 0 rswmin = 0 prwg = 0 +prwb = 0
wr = 1 alpha0 = 0.074 alpha1 = 0.005 +beta0 = 30 agidl = 0.0002 bgidl = 2.1e+009 cgidl = 0.0002 +egidl = 0.8
aigbacc = 0.012 bigbacc = 0.0028 cigbacc = 0.002 +nigbacc = 1 aigbinv = 0.014 bigbinv = 0.004 cigbinv = 0.004
+eigbinv = 1.1 nigbinv = 3 aigc = 0.0213 bigc = 0.0025889 +cigc = 0.002 aigsd = 0.0213 bigsd = 0.0025889 cigsd
= 0.002 +nigc = 1 poxedge = 1 pigcd = 1 ntox = 1 +xrcrg1 = 12 xrcrg2 = 5 +cgso = 5e-011 cgdo = 5e-011 cgbo =
2.56e-011 cgdl = 2.653e-010 +cgsl = 2.653e-010 ckappas = 0.03 ckappad = 0.03 acde = 1 +moin = 15 noff = 0.9
voffcv = 0.02 +kt1 = -0.11 kt1l = 0 kt2 = 0.022 ute = -1.5 +ua1 = 4.31e-009 ub1 = 7.61e-018 uc1 = -5.6e-011
prt = 0 +at = 33000 +fnoimod = 1 tnoimod = 0 +jss = 0.0001 jsws = 1e-011 jswgs = 1e-010 njs = 1 +ijthsfwd= 0.01
ijthsrev= 0.001 bvs = 10 xjbvs = 1 +jsd = 0.0001 jswd = 1e-011 jswgd = 1e-010 njd = 1 +ijthdfwd= 0.01 ijthdrev=
0.001 bvd = 10 xjbvd = 1 +pbs = 1 cjs = 0.0005 mjs = 0.5 pbsws = 1 +cjsws = 5e-010 mjsws = 0.33 pbswgs = 1 cjswgs
= 3e-010 +mjswgs = 0.33 pbd = 1 cjd = 0.0005 mjd = 0.5 +pbswd = 1 cjswd = 5e-010 mjswd = 0.33 pbswgd = 1 +cjswgd
= 5e-010 mjswgd = 0.33 tpb = 0.005 tcj = 0.001 +tpbsw = 0.005 tcjsw = 0.001 tpbswg = 0.005 tcjswg = 0.001 +xtis
= 3 xtid = 3 +dmcg = 0 dmci = 0 dmdg = 0 dmcgt = 0 +dwj = 0 xgw = 0 xgl = 0 +rshg = 0.4 gbmin = 1e-010 rbpb
= 5 rbpd = 15 +rbps = 15 rbdb = 15 rbsb = 15 ngcon = 1
    .model P1 pmos level = 14
    +version = 4.6.5 binunit = 1 paramchk= 1 mobmod = 0 +capmod = 2 igcmod = 1 igbmod = 1 geomod = 1 +diomod
= 1 rdsmod = 0 rbodymod= 1 rgatemod= 1 +permod = 1 acnqsmod= 0 trnqsmod= 0 +tnom = 27 toxe = 1e-009 toxp = 7e-010
toxm = 1e-009 +dtox = 3e-010 epsrox = 3.9 wint = 5e-009 lint = 1.45e-009 +ll = 0 wl = 0 lln = 1 wln = 1 +lw
= 0 ww = 0 lwn = 1 wwn = 1 +lwl = 0 wwl = 0 xpart = 0 toxref = 1e-009 +xl = -6.5e-9 +vth0 = -0.43121 k1 = 0.4
k2 = -0.01 k3 = 0 +k3b = 0 w0 = 2.5e-006 dvt0 = 1 dvt1 = 2 +dvt2 = -0.032 dvt0w = 0 dvt1w = 0 dvt2w = 0 +dsub
= 0.1 minv = 0.05 voffl = 0 dvtp0 = 1e-011 +dvtp1 = 0.05 lpe0 = 0 lpeb = 0 xj = 5e-009 +ngate = 1e+023 ndep
= 5.5e+018 nsd = 2e+020 phin = 0 +cdsc = 0 cdscb = 0 cdscd = 0 cit = 0 +voff = -0.126 nfactor = 2.1 eta0 = 0.0032
etab = 0 +vfb = 0.55 u0 = 0.006 ua = 2e-009 ub = 5e-019 +uc = 0 vsat = 250000 a0 = 1 ags = 1e-020 +a1 = 0 a2
= 1 b0 = 0 b1 = 0 +keta = -0.047 dwg = 0 dwb = 0 pclm = 0.12 +pdiblc1 = 0.001 pdiblc2 = 0.001 pdiblcb = 3.4e-008
drout = 0.56 +pvag = 1e-020 delta = 0.01 pscbe1 = 1.2e+009 pscbe2 = 8.0472e-007 +fprout = 0.2 pdits = 0.08 pditsd
= 0.23 pditsl = 2300000 +rsh = 5 rdsw = 140 rsw = 70 rdw = 70 +rdswmin = 0 rdwmin = 0 rswmin = 0 prwg = 0 +prwb
= 0 wr = 1 alpha0 = 0.074 alpha1 = 0.005 +beta0 = 30 agidl = 0.0002 bgidl = 2.1e+009 cgidl = 0.0002 +egidl =
0.8 aigbacc = 0.012 bigbacc = 0.0028 cigbacc = 0.002 +nigbacc = 1 aigbinv = 0.014 bigbinv = 0.004 cigbinv =
0.004 +eigbinv = 1.1 nigbinv = 3 aigc = 0.0213 bigc = 0.0025889 +cigc = 0.002 aigsd = 0.0213 bigsd = 0.0025889
cigsd = 0.002 +nigc = 1 poxedge = 1 pigcd = 1 ntox = 1 +xrcrg1 = 12 xrcrg2 = 5 +cgso = 5e-011 cgdo = 5e-011
cgbo = 2.56e-011 cgdl = 2.653e-010 +cgsl = 2.653e-010 ckappas = 0.03 ckappad = 0.03 acde = 1 +moin = 15 noff
= 0.9 voffcv = 0.02 +kt1 = -0.11 kt1l = 0 kt2 = 0.022 ute = -1.5 +ua1 = 4.31e-009 ub1 = 7.61e-018 uc1 = -5.6e-011
prt = 0 +at = 33000 +fnoimod = 1 tnoimod = 0 +jss = 0.0001 jsws = 1e-011 jswgs = 1e-010 njs = 1 +ijthsfwd= 0.01
ijthsrev= 0.001 bvs = 10 xjbvs = 1 +jsd = 0.0001 jswd = 1e-011 jswgd = 1e-010 njd = 1 +ijthdfwd= 0.01 ijthdrev=
0.001 bvd = 10 xjbvd = 1 +pbs = 1 cjs = 0.0005 mjs = 0.5 pbsws = 1 +cjsws = 5e-010 mjsws = 0.33 pbswgs = 1 cjswgs
= 3e-010 +mjswgs = 0.33 pbd = 1 cjd = 0.0005 mjd = 0.5 +pbswd = 1 cjswd = 5e-010 mjswd = 0.33 pbswgd = 1 +cjswgd
= 5e-010 mjswgd = 0.33 tpb = 0.005 tcj = 0.001 +tpbsw = 0.005 tcjsw = 0.001 tpbswg = 0.005 tcjswg = 0.001 +xtis
= 3 xtid = 3 +dmcg = 0 dmci = 0 dmdg = 0 dmcgt = 0 +dwj = 0 xgw = 0 xgl = 0 +rshg = 0.4 gbmin = 1e-010 rbpb
= 5 rbpd = 15 +rbps = 15 rbdb = 15 rbsb = 15 ngcon = 1
    .model N2 nmos level = 14
    +version = 4.6.5 binunit = 1 paramchk= 1 mobmod = 0 +capmod = 2 igcmod = 1 igbmod = 1 geomod = 1 +diomod
= 1 rdsmod = 0 rbodymod= 1 rgatemod= 1 +permod = 1 acnqsmod= 0 trnqsmod= 0 +tnom = 27 toxe = 9.5e-010 toxp =
7e-010 toxm = 9.5e-010 +dtox = 2.5e-010 epsrox = 3.9 wint = 5e-009 lint = 1.45e-009 +ll = 0 wl = 0 lln = 1 wln
= 1 +lw = 0 ww = 0 lwn = 1 wwn = 1 +lwl = 0 wwl = 0 xpart = 0 toxref = 9.5e-010 +xl = -6.5e-9 +vth0 = 0.17 k1
= 0.4 k2 = 0 k3 = 0 +k3b = 0 w0 = 2.5e-006 dvt0 = 1 dvt1 = 2 +dvt2 = 0 dvt0w = 0 dvt1w = 0 dvt2w = 0 +dsub =
```

```
0.1 minv = 0.05 voffl = 0 dvtp0 = 1e-011 +dvtp1 = 0.1 lpe0 = 0 lpeb = 0 xj = 5e-009 +ngate = 1e+023 ndep = 7e+018
nsd = 2e+020 phin = 0 +cdsc = 0 cdscb = 0 cdscd = 0 cit = 0 +voff = -0.13 nfactor = 2.3 eta0 = 0.0032 etab =
0 +vfb = -0.55 u0 = 0.03 ua = 6e-010 ub = 1.2e-018 +uc = 0 vsat = 290000 a0 = 1 ags = 0 +a1 = 0 a2 = 1 b0 =
0 b1 = 0 +keta = 0.04 dwg = 0 dwb = 0 pclm = 0.02 +pdiblc1 = 0.001 pdiblc2 = 0.001 pdiblcb = -0.005 drout =
0.5 +pvag = 1e-020 delta = 0.01 pscbe1 = 8.14e+008 pscbe2 = 1e-007 +fprout = 0.2 pdits = 0.01 pditsd = 0.23
pditsl = 2300000 +rsh = 5 rdsw = 140 rsw = 75 rdw = 75 +rdswmin = 0 rdwmin = 0 rswmin = 0 prwg = 0 +prwb = 0
wr = 1 alpha0 = 0.074 alpha1 = 0.005 +beta0 = 30 agidl = 0.0002 bgidl = 2.1e+009 cgidl = 0.0002 +egidl = 0.8
aigbacc = 0.012 bigbacc = 0.0028 cigbacc = 0.002 +nigbacc = 1 aigbinv = 0.014 bigbinv = 0.004 cigbinv = 0.004
+eigbinv = 1.1 nigbinv = 3 aigc = 0.0213 bigc = 0.0025889 +cigc = 0.002 aigsd = 0.0213 bigsd = 0.0025889 cigsd
= 0.002 +nigc = 1 poxedge = 1 pigcd = 1 ntox = 1 +xrcrg1 = 12 xrcrg2 = 5 +cgso = 5e-011 cgdo = 5e-011 cgbo =
2.56e-011 cgdl = 2.653e-010 +cgsl = 2.653e-010 ckappas = 0.03 ckappad = 0.03 acde = 1 +moin = 15 noff = 0.9
voffcv = 0.02 +kt1 = -0.11 kt1l = 0 kt2 = 0.022 ute = -1.5 +ua1 = 4.31e-009 ub1 = 7.61e-018 uc1 = -5.6e-011
prt = 0 +at = 33000 +fnoimod = 1 tnoimod = 0 +jss = 0.0001 jsws = 1e-011 jswgs = 1e-010 njs = 1 +ijthsfwd= 0.01
ijthsrev= 0.001 bvs = 10 xjbvs = 1 +jsd = 0.0001 jswd = 1e-011 jswgd = 1e-010 njd = 1 +ijthdfwd= 0.01 ijthdrev=
0.001 bvd = 10 xjbvd = 1 +pbs = 1 cjs = 0.0005 mjs = 0.5 pbsws = 1 +cjsws = 5e-010 mjsws = 0.33 pbswgs = 1 cjswgs
= 3e-010 +mjswgs = 0.33 pbd = 1 cjd = 0.0005 mjd = 0.5 +pbswd = 1 cjswd = 5e-010 mjswd = 0.33 pbswgd = 1 +cjswgd
= 5e-010 mjswgd = 0.33 tpb = 0.005 tcj = 0.001 +tpbsw = 0.005 tcjsw = 0.001 tpbswg = 0.005 tcjswg = 0.001 +xtis
= 3 xtid = 3 +dmcg = 0 dmci = 0 dmdg = 0 dmcgt = 0 +dwj = 0 xgw = 0 xgl = 0 +rshg = 0.4 gbmin = 1e-010 rbpb
= 5 rbpd = 15 +rbps = 15 rbdb = 15 rbsb = 15 ngcon = 1
    .model P2 pmos level = 14
    +version = 4.6.5 binunit = 1 paramchk= 1 mobmod = 0 +capmod = 2 igcmod = 1 igbmod = 1 geomod = 1 +diomod
= 1 rdsmod = 0 rbodymod= 1 rgatemod= 1 +permod = 1 acnqsmod= 0 trnqsmod= 0 +tnom = 27 toxe = 1e-009 toxp = 7e-010
toxm = 1e-009 +dtox = 3e-010 epsrox = 3.9 wint = 5e-009 lint = 1.45e-009 +ll = 0 wl = 0 lln = 1 wln = 1 +lw
= 0 ww = 0 lwn = 1 wwn = 1 +lwl = 0 wwl = 0 xpart = 0 toxref = 1e-009 +xl = -6.5e-9 +vth0 = -0.17 k1 = 0.4 k2
= -0.01 k3 = 0 +k3b = 0 w0 = 2.5e-006 dvt0 = 1 dvt1 = 2 +dvt2 = -0.032 dvt0w = 0 dvt1w = 0 dvt2w = 0 +dsub =
0.1 minv = 0.05 voffl = 0 dvtp0 = 1e-011 +dvtp1 = 0.05 lpe0 = 0 lpeb = 0 xj = 5e-009 +ngate = 1e+023 ndep =
5.5e+018 nsd = 2e+020 phin = 0 +cdsc = 0 cdscb = 0 cdscd = 0 cit = 0 +voff = -0.126 nfactor = 2.1 eta0 = 0.0032
etab = 0 +vfb = 0.55 u0 = 0.006 ua = 2e-009 ub = 5e-019 +uc = 0 vsat = 250000 a0 = 1 ags = 1e-020 +a1 = 0 a2
= 1 b0 = 0 b1 = 0 +keta = -0.047 dwg = 0 dwb = 0 pclm = 0.12 +pdiblc1 = 0.001 pdiblc2 = 0.001 pdiblcb = 3.4e-008
drout = 0.56 +pvag = 1e-020 delta = 0.01 pscbe1 = 1.2e+009 pscbe2 = 8.0472e-007 +fprout = 0.2 pdits = 0.08 pditsd
= 0.23 pditsl = 2300000 +rsh = 5 rdsw = 140 rsw = 70 rdw = 70 +rdswmin = 0 rdwmin = 0 rswmin = 0 prwg = 0 +prwb
= 0 wr = 1 alpha0 = 0.074 alpha1 = 0.005 +beta0 = 30 agidl = 0.0002 bgidl = 2.1e+009 cgidl = 0.0002 +egidl =
0.8 aigbacc = 0.012 bigbacc = 0.0028 cigbacc = 0.002 +nigbacc = 1 aigbinv = 0.014 bigbinv = 0.004 cigbinv =
0.004 +eigbinv = 1.1 nigbinv = 3 aigc = 0.0213 bigc = 0.0025889 +cigc = 0.002 aigsd = 0.0213 bigsd = 0.0025889
cigsd = 0.002 +nigc = 1 poxedge = 1 pigcd = 1 ntox = 1 +xrcrg1 = 12 xrcrg2 = 5 +cgso = 5e-011 cgdo = 5e-011
cgbo = 2.56e-011 cgdl = 2.653e-010 +cgsl = 2.653e-010 ckappas = 0.03 ckappad = 0.03 acde = 1 +moin = 15 noff
= 0.9 voffcv = 0.02 +kt1 = -0.11 kt1l = 0 kt2 = 0.022 ute = -1.5 +ua1 = 4.31e-009 ub1 = 7.61e-018 uc1 = -5.6e-011
prt = 0 +at = 33000 +fnoimod = 1 tnoimod = 0 +jss = 0.0001 jsws = 1e-011 jswgs = 1e-010 njs = 1 +ijthsfwd= 0.01
ijthsrev= 0.001 bvs = 10 xjbvs = 1 +jsd = 0.0001 jswd = 1e-011 jswgd = 1e-010 njd = 1 +ijthdfwd= 0.01 ijthdrev=
0.001 bvd = 10 xjbvd = 1 +pbs = 1 cjs = 0.0005 mjs = 0.5 pbsws = 1 +cjsws = 5e-010 mjsws = 0.33 pbswgs = 1 cjswgs
= 3e-010 +mjswgs = 0.33 pbd = 1 cjd = 0.0005 mjd = 0.5 +pbswd = 1 cjswd = 5e-010 mjswd = 0.33 pbswgd = 1 +cjswgd
= 5e-010 mjswgd = 0.33 tpb = 0.005 tcj = 0.001 +tpbsw = 0.005 tcjsw = 0.001 tpbswg = 0.005 tcjswg = 0.001 +xtis
= 3 xtid = 3 +dmcg = 0 dmci = 0 dmdg = 0 dmcgt = 0 +dwj = 0 xgw = 0 xgl = 0 +rshg = 0.4 gbmin = 1e-010 rbpb
= 5 rbpd = 15 +rbps = 15 rbdb = 15 rbsb = 15 ngcon = 1

    .OPTION TEMP=27
    .TRAN 1ns 1200ns 0s 1ns


    .END
```