# PROBABILISTIC CIRCUIT ARCHITECTURE USING STATISTICAL LEARNING

by

AMIR FAIZ BIN AB MALIK

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

## CERTIFICATION OF APPROVAL

**PROBABILISTIC CIRCUIT ARCHITECTURE USING STATISTICAL LEARNING**

by

Amir Faiz Bin Ab Malik

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

Dr. Vijanth Sagayan Asirvadam
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

December 2010

## CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

____________________________

Amir Faiz Bin Ab Malik

# ABSTRACT

As Si CMOS devices are scaled down into the nanometer (nm) range, they will soon reach their practical limits. Future nanodevices are more error-prone (due to structural and signal faults) than conventional CMOS devices. Moreover, the uncertainty of unreliable devices is common to all nanotechnologies. It is therefore important to model and characterize for example, the randomness and failures of logic states. In this report, the architecture for each basic logic gates, NOT, NAND, AND, NOR and OR using Markov Random Field (MRF) approach is presented. The scope of study includes probability and statistics with emphasis on MRF and Belief Propagation (BP), digital systems design and graph theory. The methodology of this project was first, literature research, and then followed by data evaluation. Next was to design probabilistic circuit using MRF approach on paper. After paper design, the circuit was simulated and verified using MATLAB. The result show that the MRF-implemented circuit is more complex than conventional circuit and is theoretically proven to be noise-tolerant. The main achievement of this project is the generalization of probabilistic circuit architecture. In other words, in mapping MRF into CMOS circuitry, one must fulfill two requirements; first bistable storage element for each logic state and second feedback network for belief propagation. For future work, it is recommended to demonstrate the signal fault tolerance of the architecture presented in this report as well as to extend the MRF approach to more complex circuits.

# ACKNOWLEDGEMENT

Firstly, I would like to express my utmost gratitude to my supervisor, Dr. Vijanth Sagayan Asirvadam for his guidance and constructive comments throughout this project. His feedback and support have made this project possible.

Lastly, I would like to thank my family and friends for their help and support during the whole period of this project.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| CMOS | Complementary Metal-Oxide-Semiconductor |
| ITRS | International Technology Roadmap for Semiconductors |
| MRF | Markov Random Field |
| BP | Belief Propagation |

# CHAPTER 1

# INTRODUCTION

## 1.1 Background of Study

Mainstream Si technology has relied on scaling down CMOS transistors following Moore's Law for decades. However, the International Technology Roadmap for Semiconductor (ITRS) predicts that the continued shrinkage will stop around 2015, due to unavoidable physical limits (with the ultimate transistor gate length near 10 nm). These ultimate transistors will be nanodevices in the true sense of the word. One example of nanotechnology is molecular electronics, which has been demonstrating the potential for unprecedented levels of device density, low-power computing, and high operating speed. However, many challenges remain for most nanotechnologies before they can replace conventional CMOS.

One of the challenges for example in designing nanoscale CMOS is signal error caused by noise. Since the faults are random and dynamic in nature, so the only way to deal with them is to design probabilistic circuits. A probabilistic circuit is an electrical circuit which their element values are defined by probability functions. In deterministic circuit, we use Boolean algebra but, in probabilistic circuit, we use probability theory, for example Markov Random Field (MRF). Compared to other methods, the MRF is so versatile that it can be used to describe any logic circuit. Moreover, it is a powerful tool for modeling uncertain and noisy computation, making it widely used in many areas, including physics and communications.

Probabilistic computing provides a new approach towards building more powerful fault-tolerant nanoarchitectures and systems. Since current designs employing probabilistic approach only focus on simpler circuits, so if extended to more complex circuits, this approach could lead to a paradigm shift in computing architecture.

## 1.2 Problem Statement

Circuits built with nanoscale devices will face design challenges that have not been much of an issue so far. Until now, the fabrication of nanoelectronic circuits has been limited to a few devices intended to demonstrate simple logic or memory operations. There are no actual data to measure the characteristics of large networks of devices. However, it is possible to predict two likely characteristics that will have to be confronted:

i. High and dynamic failure rate
   - It is expected that a significant fraction of the devices and their interconnections will fail.
   - These failures will occur both during fabrication and at steady rate after fabrication.

ii. Operation near the thermal limit
   - As device sizes shrink, the energy difference between the logic states will approach the thermal limit.
   - There will be errors due to temporary malfunction of the device while operating near the thermal limit and they are more significant in nanoscale devices due to very low noise margin, reduced supply voltages and low stored charges in nodes.

The first characteristic is a simple extrapolation of current device fabrication experience. The smaller the dimensions of the device, the more phenomena can interfere with correct operation. It seems likely that architectures will have to cope with device and connection failure rates of 10% or more. The second characteristic can be extrapolated from the evolution of current CMOS technology towards smaller device sizes. Current processor chips have about 100 million transistors and dissipate over 100 W. It can be assumed that this power level is already near the practical limit in terms of battery life and dangerous external temperatures.

The increased number of gates per chip and clock speed will decrease the logic transition energy limits to within a few orders of magnitude of $k_bT$. It can be shown that the energy cost of computation cannot be reduced below Landauer's principle, given as $(ln2)k_bT$ per bit. This basic result is derived from the necessary increase in randomness as information is lost during computation. Consequently, as nanocircuits operate near the thermal limit, there will be signal errors which directly account for thermal noise. This injected noise is random and dynamic in nature and it will start to affect the performance of the circuit. Since there will be structural faults due to process variations and defects and also signal faults due to random thermal noise, thus a probabilistic-based approach is more suitable (to handle these structural and signal errors) than the conventional deterministic circuit designs.

## 1.3 Objectives

In this project, there are two main objectives:

i. To improve the performance of CMOS circuits under noisy condition

ii. To design probabilistic circuits based on Markov Random Field (MRF) theory

## 1.4 Scope of Study

The scope of study of this project includes the following:

i. Probability and Statistics
   - Markov Random Field (MRF)
   - Belief Propagation (BP)

ii. Digital Systems Design

iii. Graph Theory

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Literature Review

The problems of structural and signal faults have become more significant as the size of the device approaches nanoscale level. Many fault-tolerant circuit techniques have been proposed. Folling [1] proposed architecture based on neural network which provides continuous adaptation to errors. They showed that arrays of simple neural processing elements show features such as association, fault tolerance and self-organization. Nevertheless, how their behavior generalizes to new computational examples is not clear and also their performance is difficult to optimize. MRF is a powerful tool for modeling uncertain and noisy computation and is widely used in many areas, such as physics, computer vision and communications.

Inspired by Von Neumann's pioneering work [2], MRF has proven a promising solution for the unreliability problem of nanoscale devices. Von Neumann asserted that device failure should not render faults in computing systems if it has been designed to be fault-tolerant from the beginning. Von Neumann proposed a majority voting scheme to overcome logic errors so that the errors will not be dominant. Using similar principles, [3-6] proposed a probabilistic approach using MRF. Bahar [3] showed that using MRF, structural and signal faults can be tolerated. The approach adapts to errors as a natural consequence of probability maximization, thus removing the need to detect faults which further reduces the complexity.

Correct state configuration is achieved by propagating state values through the network and updating each node assignment with a node state having the maximum probability. Current designs employing probabilistic approach only focus on elementary logic circuits. The probabilistic approach can be implemented to more complex circuits by combining these elementary logic units using belief propagation. Rao [7] used belief propagation algorithm to compute error probability at each node and thus overall circuit fault-tolerance performance can be evaluated. The future of nanocomputer is uncertain but if this probabilistic approach is successfully extended to more complex circuits, it would lead to a paradigm shift in the computer architecture.

## 2.2 Markov Random Field (MRF)

In this project, the probabilistic approach will be based on Markov Random Field (MRF). The main reason for selecting the Markov random network as the basis for the design is that its operation does not depend on perfect devices or perfect input signals. The basic idea of MRF design is that under the probabilistic framework, we cannot expect logic values in a circuit at a particular time to be correct. We can only expect the probability distribution of the values to have the highest or maximum likelihood in a correct logic state. The MRF is a powerful tool for modeling uncertain and noisy computation. It is a completely general computational framework and in principle any type of computation could be mapped onto the model.

The MRF approach can express any arbitrary logic circuits and logic operation is achieved by maximizing the probability of the state configurations in the logic network. Maximizing state probability is equivalent to minimizing a form of energy that depends on neighboring nodes in the network. Once elementary logic components have been developed, they can be linked together using belief propagation to build desired architectures. In the probabilistic-based circuit design, the logic states are considered to be random variables. Therefore, one can no longer expect a correct logic signal at all nodes at all times, but only that the joint probability distribution of signal values has the highest likelihood for valid logic states. This joint distribution can be considered to be a distribution on random vectors, with a vector element for each logic variable in the circuit.

Thus, with this probabilistic approach, circuit design is guided by the formulation of a multivariate distribution on vectors, aiming for a distribution that attains maximum probability for the valid states of the circuit. In a circuit with hundreds of logic variables, it is impractical to directly consider a joint probability distribution. The number of constraints required to enforce maximum probability for the valid states grows exponentially with the dimension of the random vector space and so the computation becomes intractable. However, the representation for these high dimensional joint distributions can be factored into low dimensional distributions using MRF.

The Markov random field defines a set of random variables, $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_k\}$. Each variable $\lambda_i$, can take on various values, e.g. state labels. Associated with each variable is a neighborhood, $N_i$, which is a set of variables from $\{\Lambda - \lambda_i\}$. Simply put, the probability of a given variable depends only on a (typically small) neighborhood of other variables. An appropriate model for the MRF is a graph structure, where the nodes of the graph represent logic variables and the edges represent statistical dependency between the variables. The definition of the MRF is as follows:

i. Positivity: $P(\lambda) > 0, \forall \lambda \in \Lambda$ (1)

ii. Markovianity: $P(\lambda_i | \{\Lambda - \lambda_i\}) = P(\lambda_i | N_i)$ (2)

In other words, a set of random variables form a MRF if all sites have a finite positive probability and the probability of a particular site in the neighborhood depends only on its immediate neighbors to which it is connected by an edge. The edges in the neighborhood represent the conditional dependence between the connected variables in the neighborhood. The conditional probability of a given site in terms of its neighborhood can be formulated in terms of the associated clique of the graph structure. Figure 2.1 shows an example of a neighborhood with one 1st order clique and one 2nd order clique.

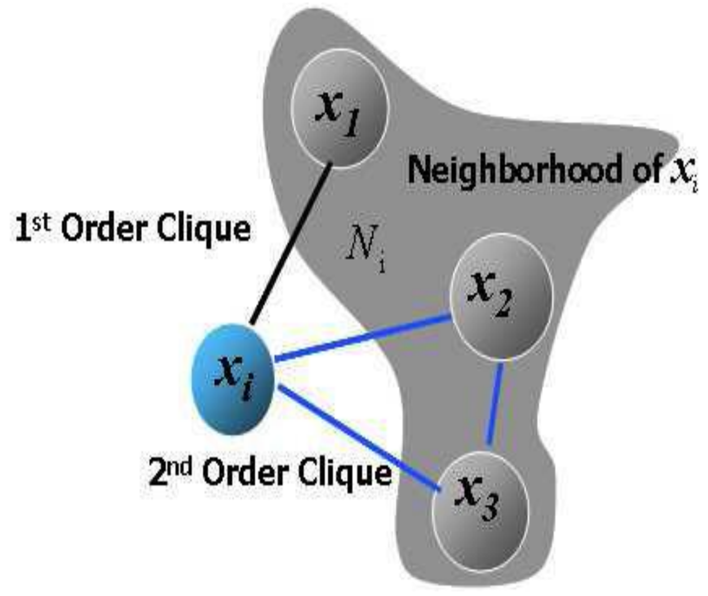

Figure 2.1: The MRF neighborhood system

The conditional probability of a node state in terms of its neighborhood can be formulated in terms of cliques. It can be shown due to the Hammersley-Clifford theorem:

$$P(x_i \mid \Lambda - \lambda_i) = \frac{1}{Z} e^{-\frac{1}{k_b T}\sum_{c\in C} U_c(x)} \tag{3}$$

This form for the probability is called the Gibbs distribution. The normalizing constant *Z* is called the partition function and insures that *P* is in the range [0, 1]. The set *C* is the set of cliques for a given node, *i*. The function $U_C$ is called the clique energy function and depends only on the nodes in the clique. It also can be seen that the probability of states depends on the ratio of clique energy of the MRF to the thermal energy, $k_bT$. For instance, the probability is uniform at high values of $k_bT$ and becomes sharply peaked at low values of $k_bT$.

Circuit networks can be expressed in terms of such neighborhoods and the interaction of the logic states and variables can be represented as a dependence graph. Figure 2.2 shows a simple multi-level circuit and its corresponding dependence graph. In this case, the graph is equivalent to a Markov random field, where the nodes are random logic variables that can hold values ranging from 0V to $V_{DD}$ and the edges are the conditional dependencies between the variables. Importantly, there is no notion of directed logic flow and causality, just statistical dependence. For instance, if the output of the first NAND gate is at logic 0, then both the inputs are constrained to be at logic 1 which means that there is a (backward) statistical dependency between the output state and the input state.

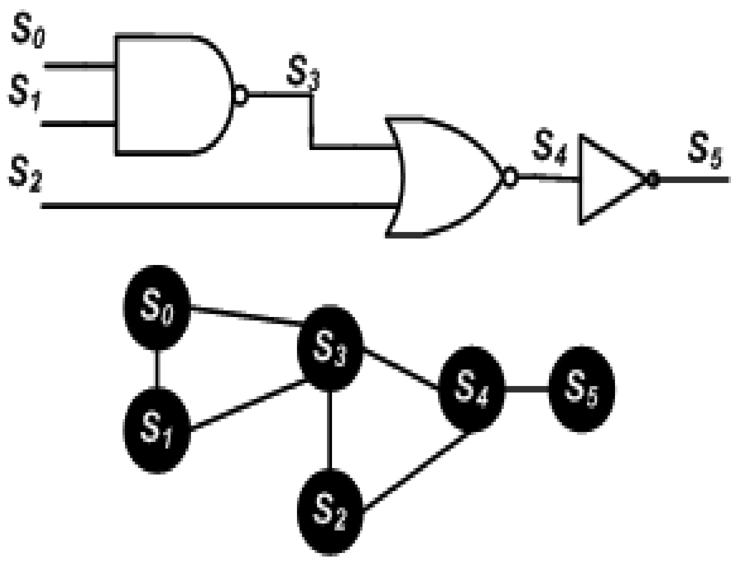

Figure 2.2: A logic circuit and its dependence graph

The general algorithm for finding individual site labels that maximize the probability of the overall network is called belief propagation and provides an efficient means of solving inference problems by propagating marginal probabilities through the network. There are three essential probability functions:

i. Joint probability:

$$p(x_0, x_1, ..., x_{n-1}) \tag{4}$$

ii. Marginal probability:

$$p(x_i) = \sum_{x_0} \sum_{x_1} \cdots \sum_{x_j} \cdots \sum_{x_{n-1}} p(x_0, x_1, ..., x_{n-1}), j \neq i \tag{5}$$

iii. Conditional probability:

$$p(x_0, x_1, ..., x_{i-1}, x_{i+1}, ... | x_i) = \frac{p(x_0, x_1, ..., x_{n-1})}{p(x_i)} \tag{6}$$

The basic idea of belief propagation is that the probability of state labels at a given node in the network can be determined by marginalizing (summing) over the joint probabilities for the node state given just the probabilities for site labels in the Markov neighborhood, $N_i$. The nodes in the network can be classified into:

i. Observable nodes
   - Those that have defined label probabilities
   - Correspond to a computational input whose value is constrained by the problem setup

ii. Hidden nodes
   - Those whose values must be determined by the propagation algorithm

In a logic circuit, the input/output can be thought of as observable nodes and the others as hidden nodes. The marginal probabilities that are computed approximately are referred as *beliefs*, and the belief at node *i* is denoted *by b(x$_i$).* In MRF, the *observable nodes*, $y_i$ are considered to be fixed, and we can write $\phi_i(x_i)$ as a short-hand for $\phi_i(x_i, y_i)$, where $x_i$ is the *hidden nodes*. We further assume that there is some statistical dependence between $x_i$ and $y_i$ at each position *i*, which we write as a joint probability $\phi_i(x_i, y_i)$.

The function $\phi_i(x_i, y_i)$ is often called the *evidence* for $x_i$. For us to be able to infer anything about the nanoscale computer architecture, there has to be some structure to the hidden nodes, $x_i$. We encode the assumed structure by saying that the variable $x_i$ should, insofar as possible, be "compatible" with nearby variable, $x_j$, as represented by compatibility function $\psi_{ij}(x_i, x_j)$, where $\psi_{ij}$ only connects nearby positions. We then take the joint probability distribution for the unknown variables $x_i$ as:

$$P(x) = \frac{1}{Z}\prod_{(ij)}\psi_{ij}(x_i, x_j)\prod_i \phi_i(x_i) \tag{7}$$

This marginalization establishes the label probabilities for the next propagation step. It can be shown that this propagation algorithm will converge to the maximum probability site label assignment for the entire network, provided there are no loops. However, it can be shown that the belief propagation algorithm usually converges to the maximum probability state even in the presence of loops. As an example of how the belief propagation is calculated, consider Figure 2.2, where the probability, *p(s$_0$, s$_1$, s$_2$, s$_3$, s$_4$, s$_5$)* can be decomposed into:

$$p(s_0, s_1, s_2, s_3, s_4, s_5) = U(s_0, s_1, s_3)U(s_2, s_3, s_4)U(s_4, s_5) \tag{8}$$

For belief propagation, we start from the primary inputs of the circuit network. As a first step, $s_0$ and $s_1$ are eliminated by summing $U(s_0, s_1, s_3)$ over all states of $s_0$ and $s_1$ to obtain $U(s_3)$, that is $s_0$ and $s_1$ are marginalized out. Then $s_2$ and $s_3$ can be eliminated by summing $U(s_3)U(s_2, s_3, s_4)$ over all states of $s_2$ and $s_3$, giving $U(s_4)$. Finally, $s_4$ can be eliminated similarly to obtain $U(s_5)$. This example shows that achieving the correct state configuration in the network corresponds to propagating state values through the network and updating each node assignment with a node state having the maximum probability. Moreover, the advantage of the Markov network model is that this probability is maximum when the total clique energy is a minimum.

# CHAPTER 3

# METHODOLOGY

## 3.1 Procedure Identification

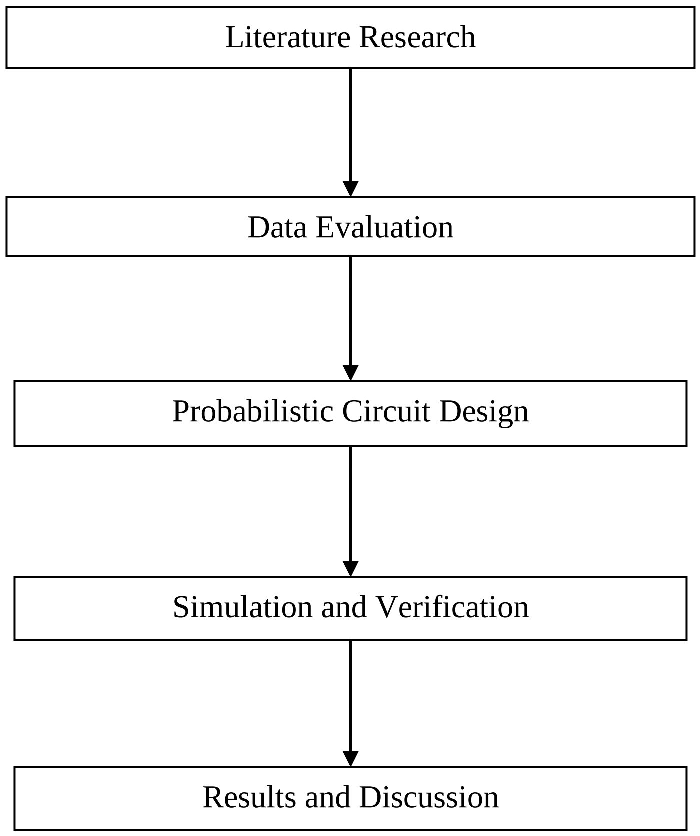

Figure 3.1: Flow chart of Final Year Project (FYP)

1. Literature Research

   - Probability and Statistics: Did research on Markov Random Field (MRF), Belief Propagation (BP) and few other probabilistic methods
   - Digital System Design: Did research on how to design basic logic circuits using probability theory
   - Graph Theory: Did learn how to use graph theory in Markov Random Field

2. Data Evaluation

   - Chose which data or reading materials were relevant to the scope and objectives of the project
   - Selected Markov Random Field and Belief Propagation to design probabilistic circuit

3. Probabilistic Circuit Design

   - Implemented probability theories, Markov Random Field and Belief Propagation in designing basic logic circuits
   - Did design on paper

4. Simulation and Verification

   - Based on paper design, did the probabilistic circuit in MATLAB, using Simulink

5. Results and Discussion

   - Analyzed the performance of the MRF CMOS circuits

## 3.2 Tool

The primary tool used in this project was MATLAB 7.1.

# CHAPTER 4

# RESULTS AND DISCUSSION

## 4.1 Results

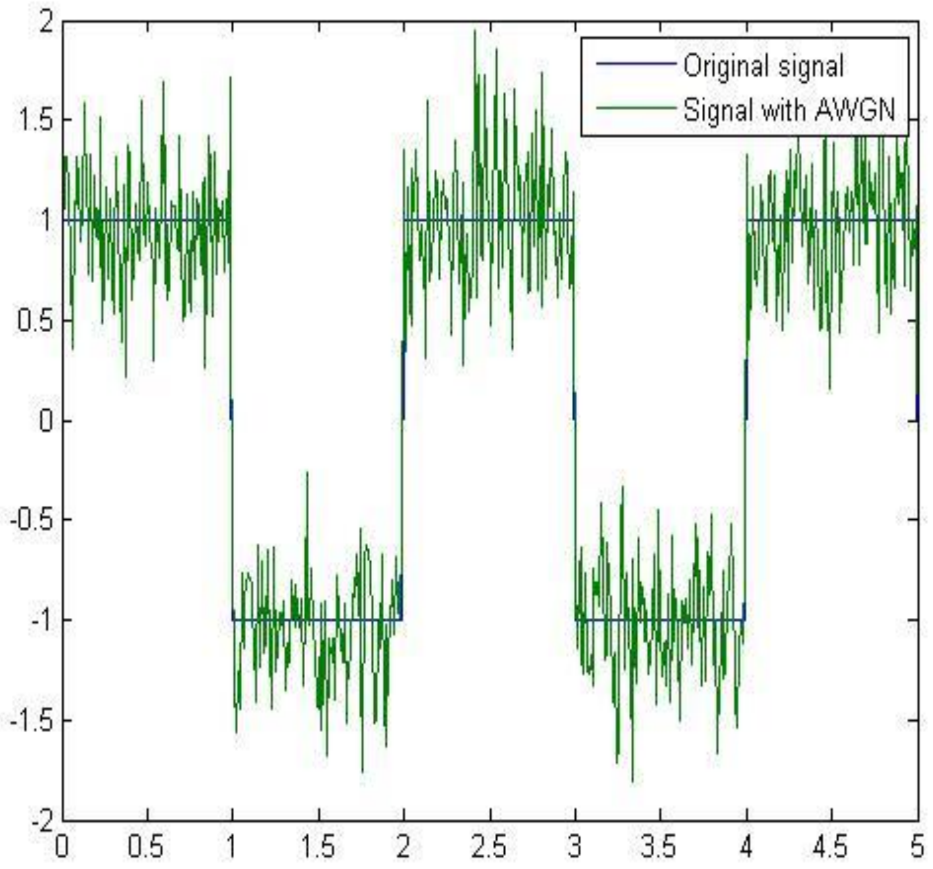

Figure 4.1: Digital signal corrupted by AWGN with SNR 10 dB

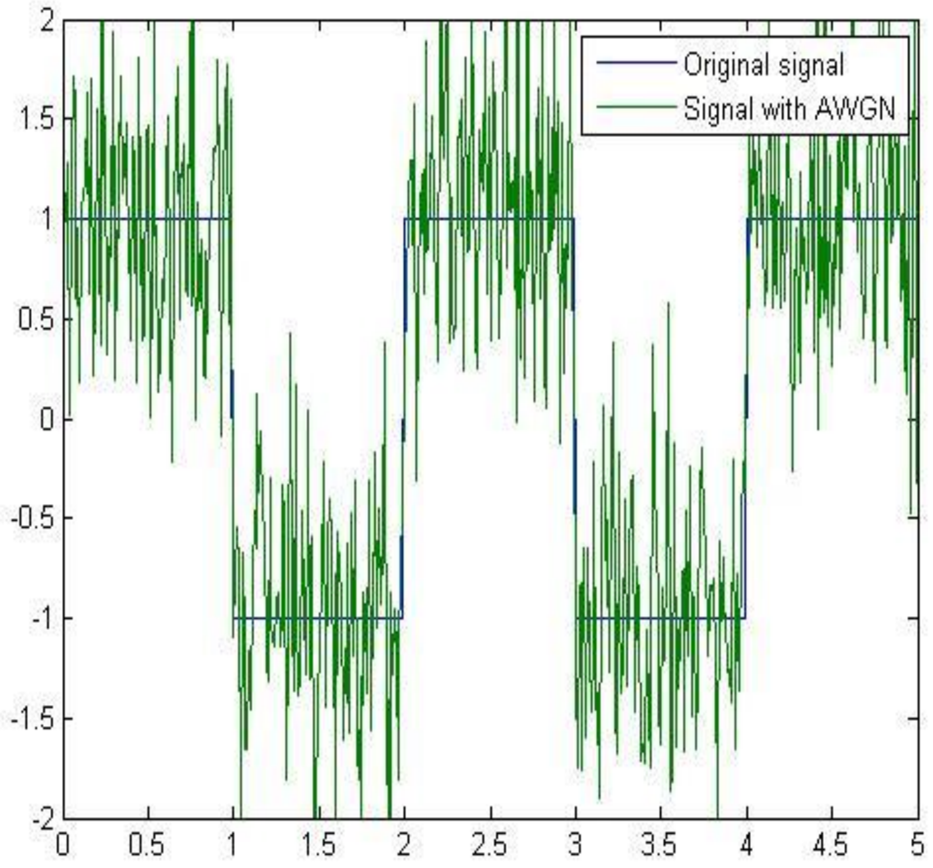

Figure 4.2: Digital signal corrupted by AWGN with SNR 5 dB

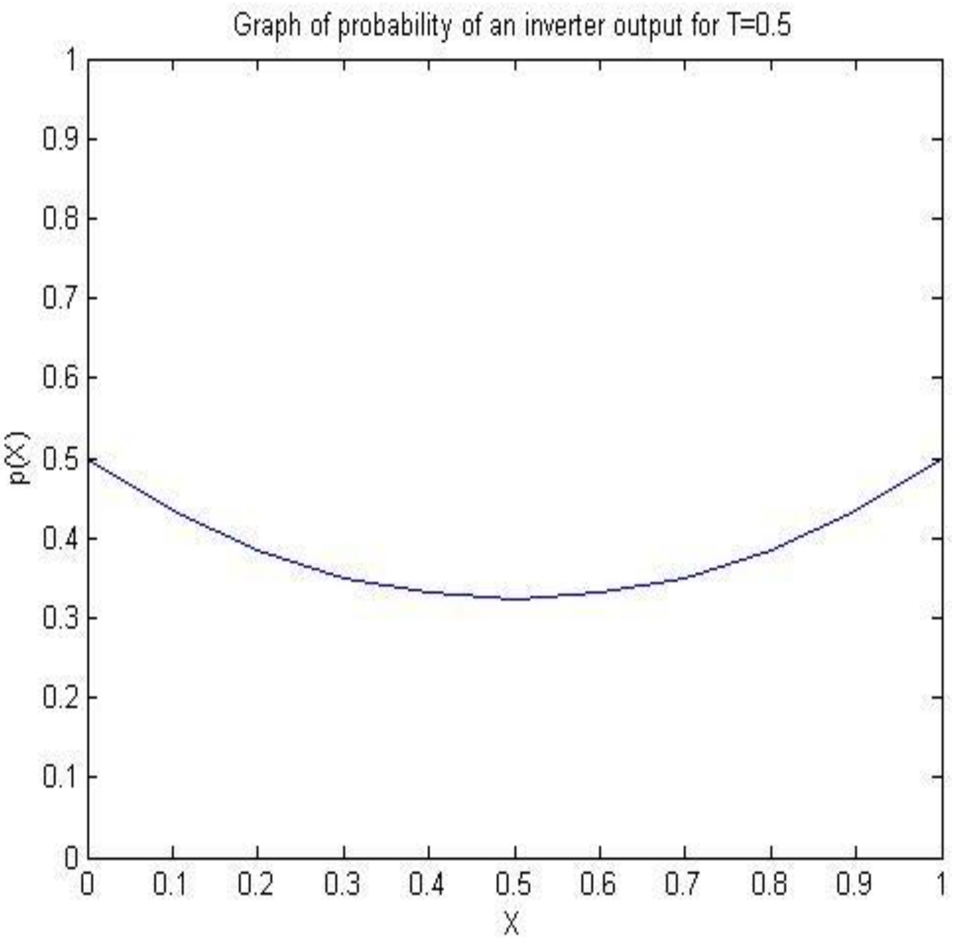

Figure 4.3: Graph of probability of an inverter output for T = 0.5

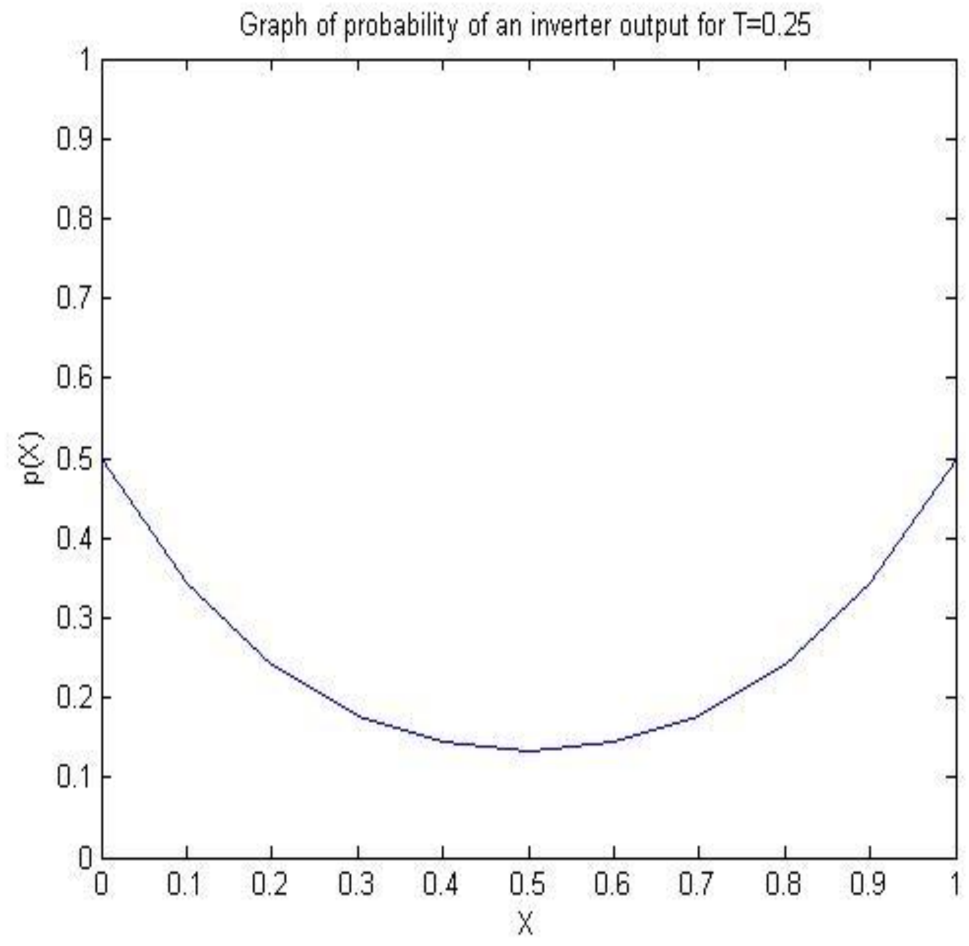

Figure 4.4: Graph of probability of an inverter output for T = 0.25

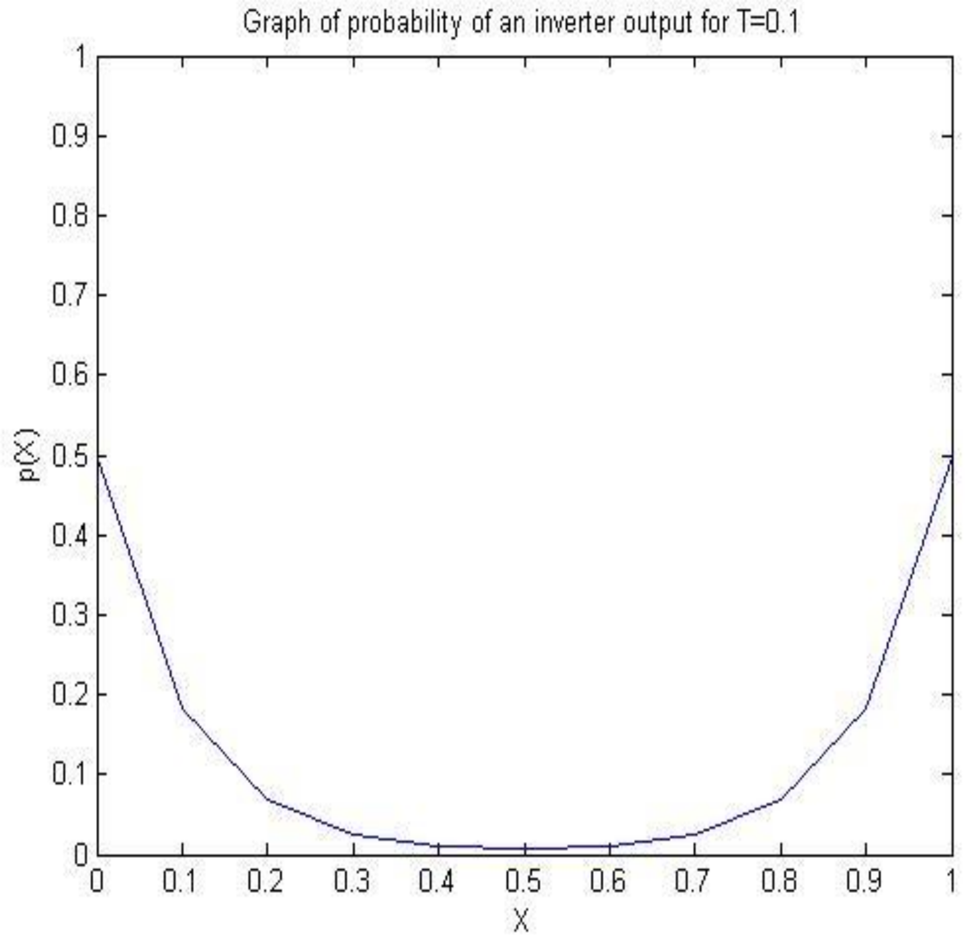

Figure 4.5: Graph of probability of an inverter output for T = 0.1

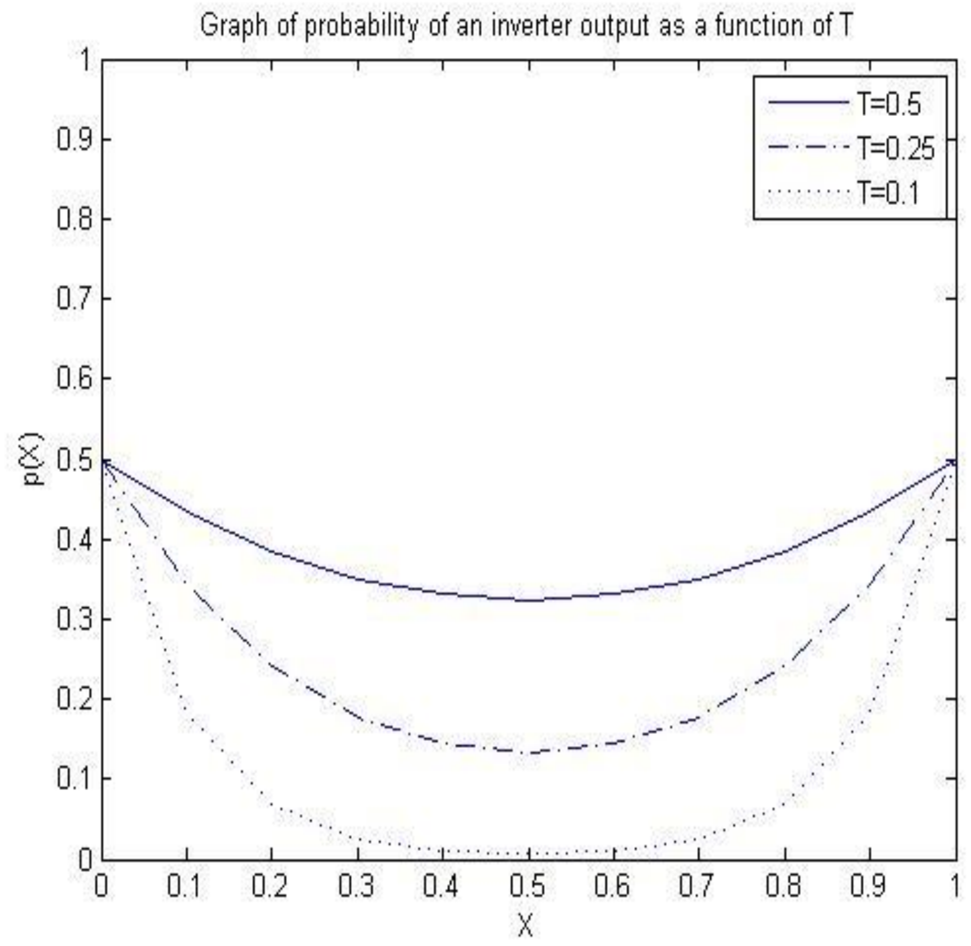

Figure 4.6: Graph of probability of an inverter output as a function of T

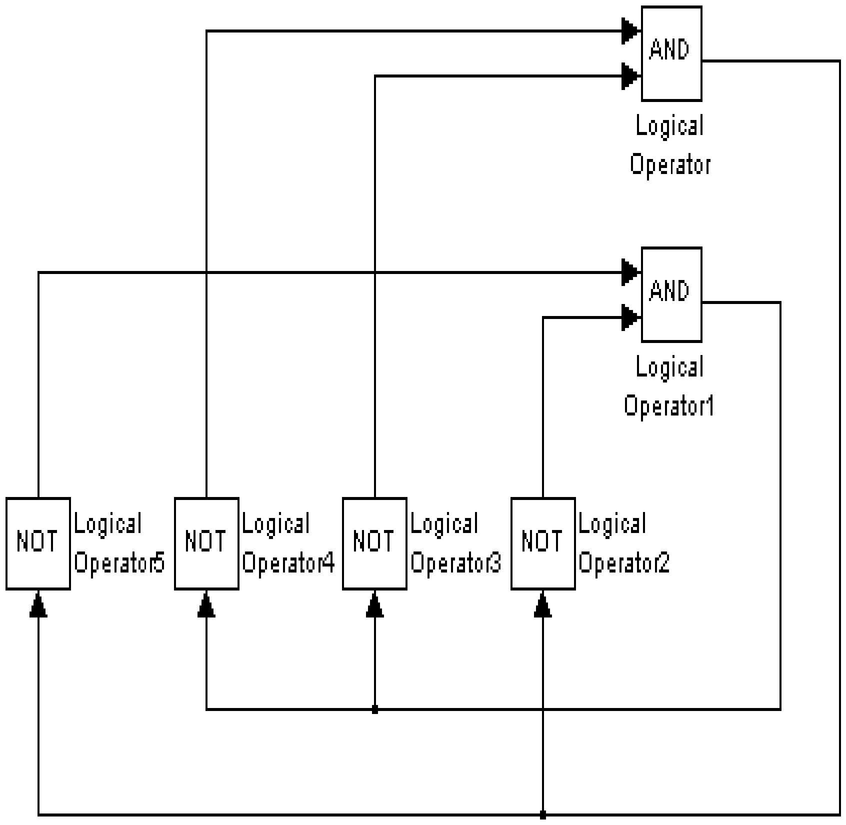

Figure 4.7: MRF NOT gate

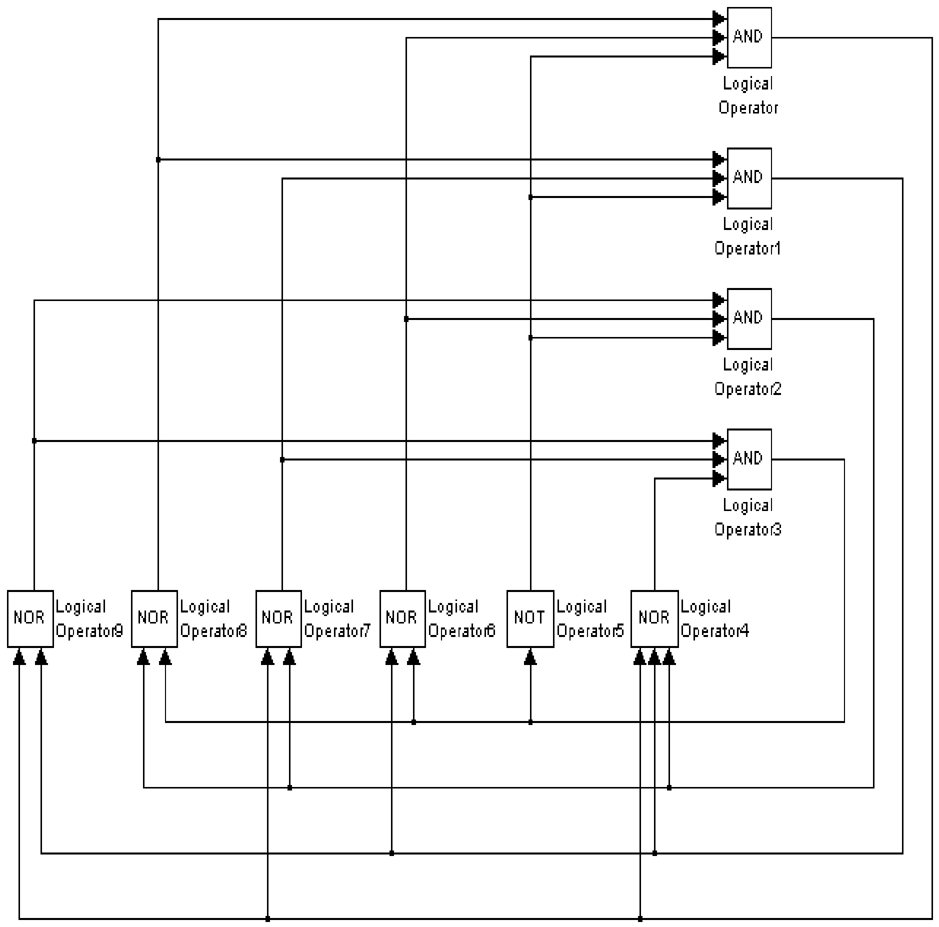

Figure 4.8: 2-input MRF NAND gate

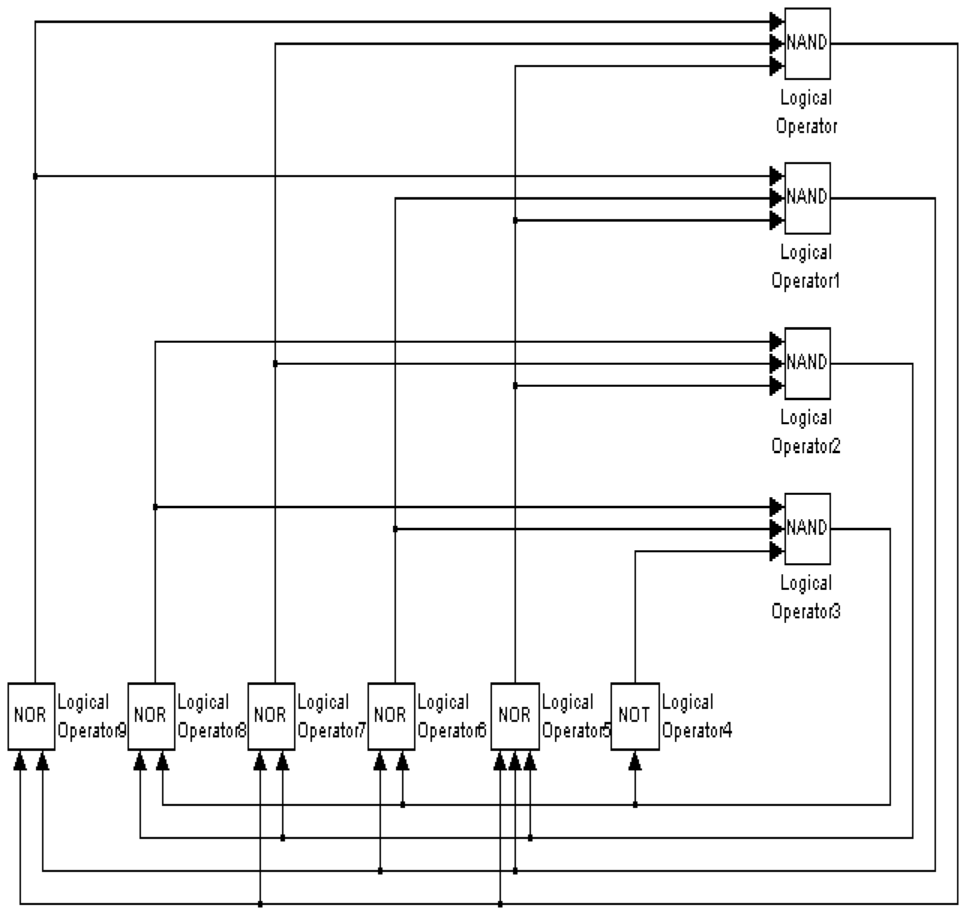

Figure 4.9: 2-input MRF AND gate

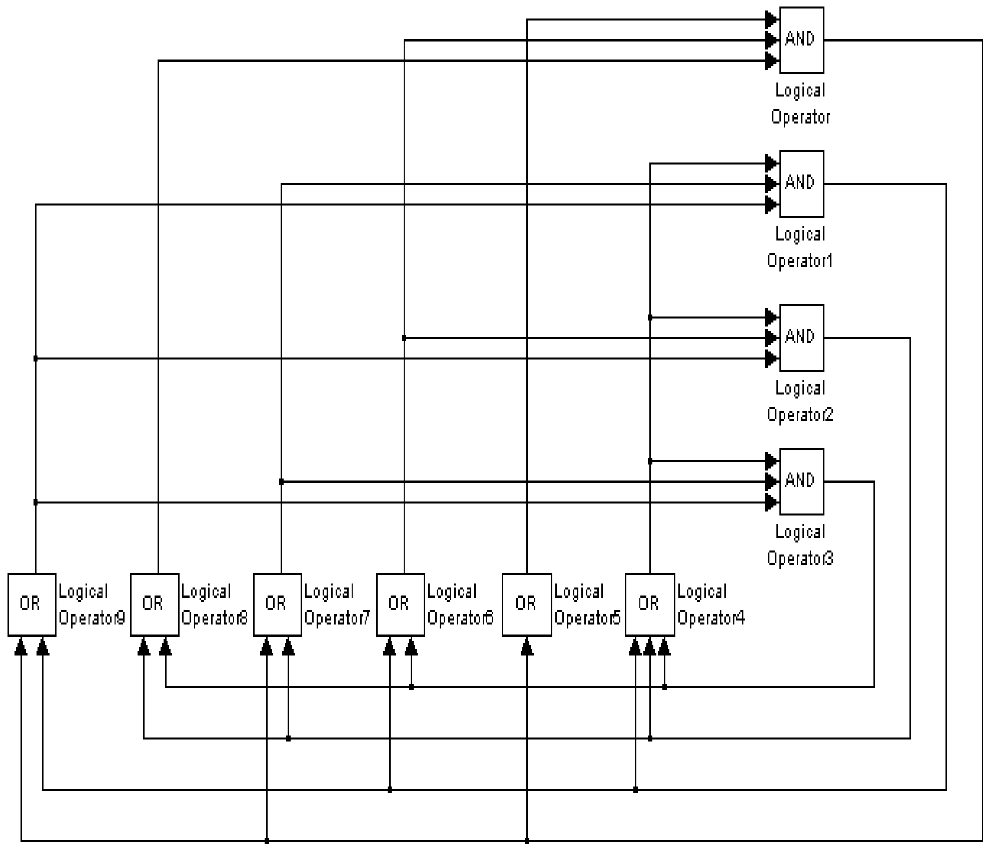

Figure 4.10: 2-input MRF NOR gate

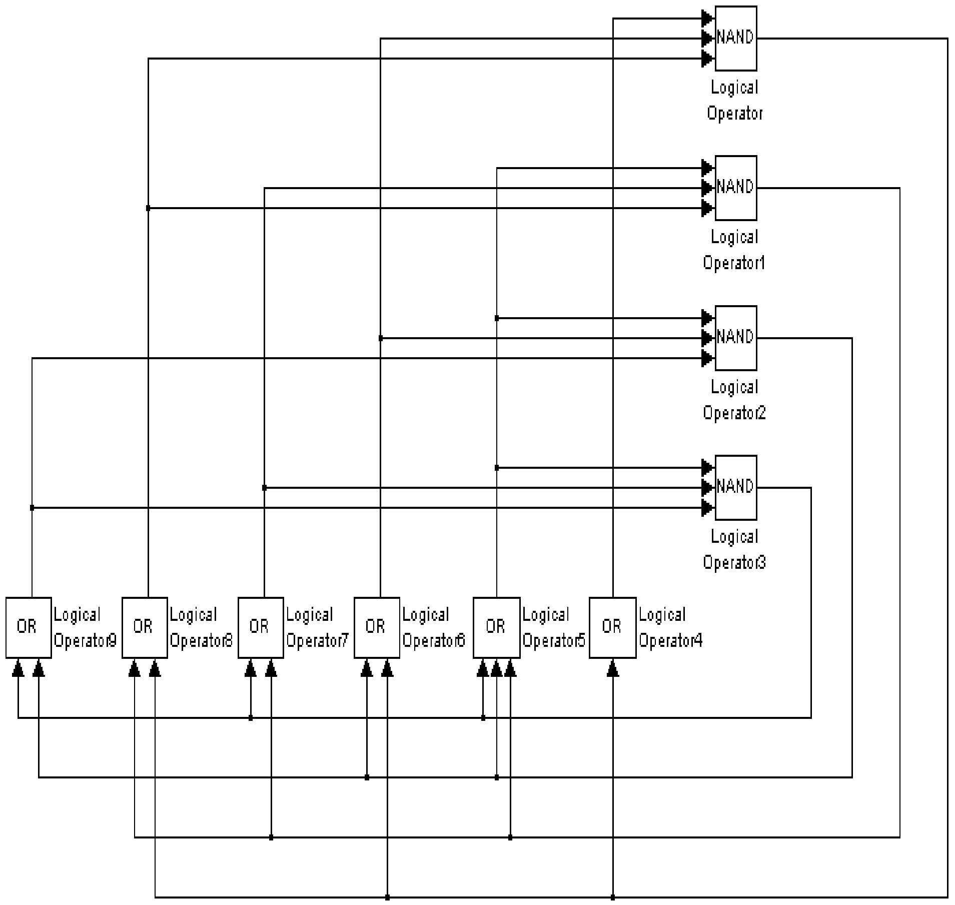

Figure 4.11: 2-input MRF OR gate

## 4.2 Discussion

The MRF is a completely general computational framework and in principle any type of computation could be mapped onto the model. In order to concretely illustrate the operation of the model, we will use combinatorial logic as an example. The programming of the MRF is straightforward in this case and will permit some analysis of the fault tolerance of the architecture. Combinatorial logic can be implemented using a simple, yet powerful, form for the clique energy, called the *auto-model*. For cliques up to order three, the energy function is given by:

$$U_C(\lambda) = \kappa + \sum\nolimits_{i \in C_0} \alpha_i \lambda_i + \sum\nolimits_{i,j \in C_1} \beta_{ij} \lambda_i \lambda_j + \sum\nolimits_{i,j,k \in C_2} \gamma_{ijk} \lambda_i \lambda_j \lambda_k \qquad (9)$$

The constants, $\alpha_i$, $\beta_{ij}$ and $\gamma_{ijk}$ are called *interaction coefficients*. The constant $\kappa$ acts as an energy offset. This form for $U_C(\lambda)$ has been used in many MRF applications including image segmentation, texture classification and object recognition.

There are two aspects of fault tolerance that must be considered. The first one is structural fault and the second one is signal fault. However, until now, only tolerance to discrete errors in signal is simulated. The structural fault and continuous errors in signal will be considered in future work. Based on the results obtained:

i. For Figure 4.1, the generated signal is a square wave signal that resembles the digital waveform. Then, this signal is added with Additive White Gaussian Noise (AWGN) to reflect the nature of random and dynamic noise in nanoscale devices. The signal-to-noise ratio (SNR) is arbitrarily defined as 10 dB. This is done so to observe the effect of noise to the signal. It can be seen that +1 V and -1 V oscillate randomly.

ii. For Figure 4.2, the generated signal is again a square wave signal that imitates the digital waveform. This signal is then injected with AWGN to reflect the nature of random and dynamic noise in nanoscale devices. The difference between Figure 4.2 and Figure 4.1 is only in the SNR. In Figure 4.2, the SNR is decreased to 5 dB. This means that if the signal power remains constant, the noise power increases. This soundly reflects the effect of scaling CMOS down to nanoscale level, where noise becomes more significant. It also can be seen that the +1 V can be mistakenly interpreted as -1 V since the noise power is now more significant. The same applies to -1 V. Thus, the logic operation can be deemed as no longer correct since 1 can be 0 and vice versa.

iii. Successful operation of a gate is designated by the *compatibility function*, $f(x_0, x_1)$ as shown below:

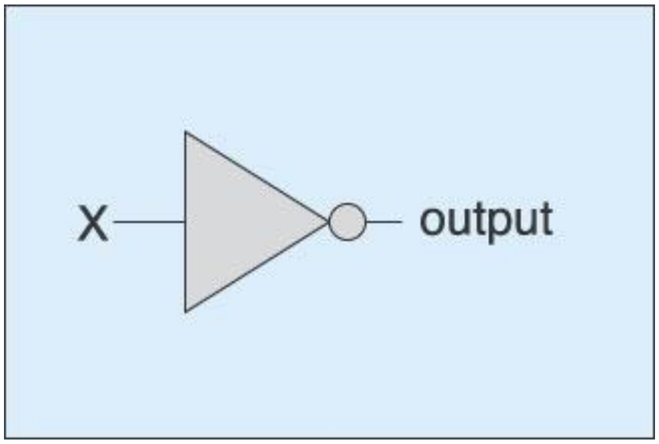

Figure 4.12: Conventional NOT gate

Table 4.1: NOT gate logic compatibility function with all possible states

| States | Input | Output | Validity |
|---|---|---|---|
| $i$ | $x_0$ | $x_1$ | $f$ |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 2 | 1 | 0 | 1 |
| 3 | 1 | 1 | 0 |

Where $x_0$ is the input and $x_1$ is the output. Here, all possible states are listed as shown in Table 4.1 (valid states with $f = 1$ and invalid states with $f = 0$) because the probabilistic approach adapts to errors and makes no assumption about the occurrence of errors. In order to relate the logic compatibility function to Gibbs energy form, it is necessary to use the axioms of the *Boolean ring*. The Boolean ring expresses the rules of symbolic Boolean logic in terms of algebraic manipulations as follows:

$$X' \rightarrow (1 - X) \tag{10}$$

$$X_1 \wedge X_2 \rightarrow X_1 X_2 \tag{11}$$

$$X_1 \vee X_2 \rightarrow X_1 + X_2 + X_1 X_2 \tag{12}$$

The logic variables are treated as real valued algebraic quantities and logic operations are transformed into arithmetic operations. Additionally, it is desired that valid input/output states should have lower clique energies than invalid states. Thus, the clique energy expression is obtained by a negative sum over minterms from the valid states ($f = 1$):

$$U(x_0, x_1) = -\sum_i f_i(x_0, x_1) \tag{13}$$

Where $f_i = 1$, and the minterms are transformed using the Boolean ring rules. For the NOT gate example, by summing over the valid states, the clique energy can be computed as follows:

The Boolean ring conversion for the minterm $(x_0, x_1) = 01$ is,

$$\begin{aligned} x_0' \wedge x_1 &= (1 - x_0)x_1 \\ &= x_1 - x_0x_1 \end{aligned} \tag{14}$$

The Boolean ring conversion for the minterm $(x_0, x_1) = 10$ is,

$$\begin{aligned} x_0 \wedge x_1' &= x_0(1 - x_1) \\ &= x_0 - x_0x_1 \end{aligned} \tag{15}$$

So, the clique energy or auto-model of an inverter is,

$$\begin{aligned} U &= -(x_1 - x_0x_1 + x_0 - x_0x_1) \\ &= -(-2x_0x_1 + x_0 + x_1) \\ &= 2x_0x_1 - x_0 - x_1 \end{aligned} \tag{16}$$

Then, the Gibbs distribution becomes,

$$p(x_0, x_1) = \frac{1}{Z} e^{-\frac{1}{k_bT}(2x_0x_1 - x_0 - x_1)} \tag{17}$$

Suppose the input, $x_0$ takes on values from {0,1}. The dependence on the input $x_0$ can be marginalized away by summing over its possible values:

$$p(x_1) = \frac{1}{Z} \sum_{x_0 \in \{0,1\}} e^{-\frac{1}{k_bT}(2x_0x_1 - x_0 - x_1)} = \frac{e^{\frac{x_1}{k_bT}} + e^{\frac{(1-x_1)}{k_bT}}}{2\left(1 + e^{\frac{1}{k_bT}}\right)} \tag{18}$$

In the marginalization, it is assumed that the input to the inverter is equally likely to be 0 or 1 and that the inverter has exact clique energy weights. These assumptions are somewhat idealized since in practice the inverter will have variable clique coefficients and the input will range over a continuous set of values near 0 or 1 according to the distribution of signal noise and device error.

iv. Figure 4.6 is the plot of marginalized inverter output distribution function for various values of thermal energy, $k_bT$. It can be seen that both outputs 0 and 1 are equally likely. However, note that the most likely outputs are 0 and 1 and the likelihood of any intermediate values become significantly small as $k_bT$ approaches 0. This behavior is the characteristic of Markov random network processing. As long as the energy balance is favorable to correct logic state, decreasing $k_bT$ will lock in the valid configurations.

As described earlier, the key of the MRF circuit behavior is that the logic states of network nodes need to depend, in a probabilistic fashion, on the logic states of some finite number of neighboring nodes. For the purposes of probabilistic computation, a physical embodiment of interacting logic levels and the clique energy function must be determined. To obtain correct logic operation, clique energy must be minimized.

This energy minimization can be achieved by a device or device configuration that produces a bi-stable energy function. A binary flip-flop circuit possesses this desired energy behavior where the required asymmetry of state energy is created by the summing mechanism just described. Nepal [4] found two requirements in mapping the MRF model into CMOS circuitry:

i. Each logic state, $s_i$, should be represented as a bistable storage element, taking on logical values of ”0” an ”1” with equal probability. The probability for any other signal value should be low.
ii. The constraints of each logic graph clique should be enforced by feedback to the appropriate storage elements, implementing the logic compatibility functions to maximize the joint probability of the correct logical values.

The first requirement ensures that the MRF logic states are maintained so that the conditional probabilities among the neighboring elements can propagate. The feedback paths, required by the second design principle, are based on conditional probabilities and ensure that the correct logic states are the most probable states. Whereas the bistable element allows us to maintain a particular logic state at a given node, the feedback mechanism allows us to model the belief propagation and the dependence of a node on the state of its neighborhood.

For 2-input NAND gate, from Table 4.2 below, it can be seen that there are a total of four minterms. Each minterm is a valid input-output pair whose probability must be be maximized using a bistable storage element. The feedback to $x'_2$ comes from the first three minterms containing $x_2$, while the feedback to $x_2$ comes only from the final minterm containing $x'_2$. Since more than one minterm can determine the state of a logic variable, a complex feedback network consisting of NOR logic gates are needed as shown in Figure 4.8. The circuit shows that a bistable element is required for each minterm.

Table 4.2: 2-input NAND gate logic compatibility function with all possible states

| States | Inputs | | Output | Validity |
|---|---|---|---|---|
| $i$ | $x_0$ | $x_1$ | $x_2$ | $f$ |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | 1 |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 0 |

In designing MRF CMOS circuits, the feedback network models the belief propagation and it shows that achieving the correct state configuration corresponds to propagating the state values and updating each node assignment with a node state having the maximum probability. The same approach was used in designing other basic logic gates, AND, NOR and OR. The logic compatibility function for each of the remaining basic logic gates (AND, NOR and OR):

Table 4.3: 2-input AND gate logic compatibility function with all possible states

| States | Inputs | | Output | Validity |
|---|---|---|---|---|
| *i* | $x_0$ | $x_1$ | $x_2$ | *f* |
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | 0 |
| 6 | 1 | 1 | 0 | 0 |
| 7 | 1 | 1 | 1 | 1 |

Table 4.4: 2-input NOR gate logic compatibility function with all possible states

| States | Inputs | | Output | Validity |
|---|---|---|---|---|
| *i* | $x_0$ | $x_1$ | $x_2$ | *f* |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | 0 |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 0 |

Table 4.5: 2-input OR gate logic compatibility function with all possible states

| States | Inputs | | Output | Validity |
|---|---|---|---|---|
| $i$ | $x_0$ | $x_1$ | $x_2$ | $f$ |
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | 1 |
| 6 | 1 | 1 | 0 | 0 |
| 7 | 1 | 1 | 1 | 1 |

As an example of how the MRF logic circuit is designed, consider 2-input AND gate:

i. All possible states with valid states correspond to f = 1 and invalid states, f = 0 was listed in logic compatibility function table as shown in Table 4.3.

ii. For each valid state, its minterm was written down as shown in Table 4.6.

Table 4.6: 2-input AND gate logic compatibility function with only valid states

| States | Inputs | | Output | Validity |
|---|---|---|---|---|
| $i$ | $x_0$ | $x_1$ | $x_2$ | $f$ |
| 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 | 1 |
| 4 | 1 | 0 | 0 | 1 |
| 7 | 1 | 1 | 1 | 1 |

For state:

- $i = 0$,
  $x_0 = 0$, $x_1 = 0$, $x_2 = 0$,
  minterm for 000 = $x_0' x_1' x_2'$
- $i = 2$,
  $x_0 = 0$, $x_1 = 1$, $x_2 = 0$,
  minterm for 010 = $x_0' x_1 x_2'$
- $i = 4$,
  $x_0 = 1$, $x_1 = 0$, $x_2 = 0$,
  minterm for 100 = $x_0 x_1' x_2'$
- $i = 7$,
  $x_0 = 1$, $x_1 = 1$, $x_2 = 1$,
  minterm for 111 = $x_0 x_1 x_2$

iii. Then, the bistable element network, consisting of NAND or AND gates was designed according to the minterms as shown below:

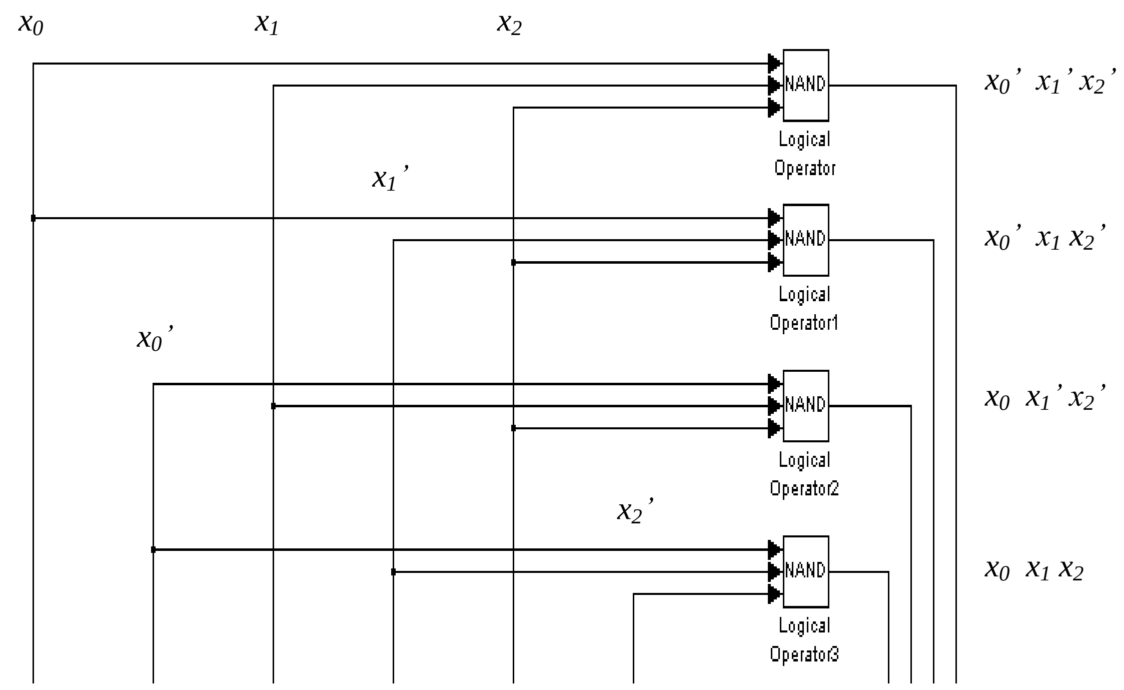

Figure 4.13: Bistable element network of 2-input MRF AND gate

iv. Next, the complex feedback network, consisting of NOR or OR gates was designed based on the minterms as shown below:

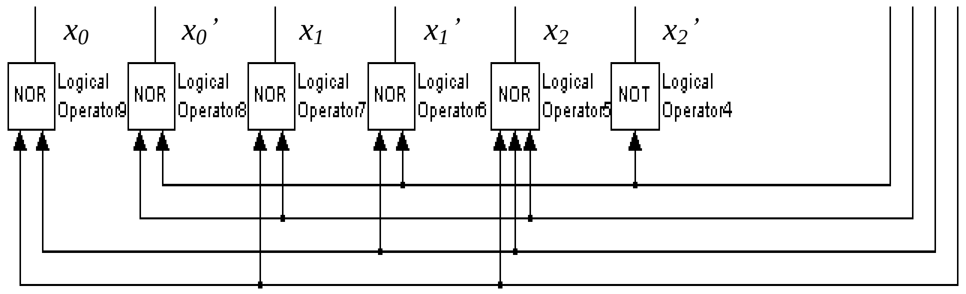

Figure 4.14: Feedback network of 2-input MRF AND gate

v. Lastly, the MRF-implemented circuit was checked for validity using all the possible states.

The state probability for each logic gate can be found using Markov Random Field. The state probabilities for NOT and NAND gates are shown below:

i. NOT gate

From Table 4.1, the minterms are:

01 $= x_0' \wedge x_1$

$= (1 - x_0)\, x_1$

$= x_1 - x_0\, x_1$

10 $= x_0 \wedge x_1'$

$= x_0\, (1 - x_1)$

$= x_0 - x_0\, x_1$

So the clique energy, U becomes,

U $= -[x_1 - x_0\, x_1 + (x_0 - x_0\, x_1)]$

$= -[x_1 - x_0\, x_1 + x_0 - x_0\, x_1]$

$= 2\, x_0\, x_1 - x_0 - x_1$

Partition function, Z,

$$Z = \sum_w e^{-U(w)} \qquad (19)$$

w = ($x_0$, $x_1$) or all possible states

Therefore the partition function, Z becomes,

$$Z = \sum_w e^{-\frac{1}{k_b T}(2x_0x_1 - x_0 - x_1)}$$

$$= e^{-\frac{1}{k_bT}(2(0)(0)-0-0)} + e^{-\frac{1}{k_bT}(2(0)(1)-0-1)} +$$

$$e^{-\frac{1}{k_bT}(2(1)(0)-1-0)} + e^{-\frac{1}{k_bT}(2(1)(1)-1-1)}$$

$$= 2 + 2e^{\frac{1}{k_bT}}$$

$$= 2\left(1 + e^{\frac{1}{k_bT}}\right) \qquad (20)$$

The output probability,

$$\mathrm{p}(x_1) = \frac{1}{Z}\sum_{x_0=\{0,1\}} e^{-\frac{1}{k_bT}(2x_0x_1-x_0-x_1)}$$

$$= \frac{1}{Z}\left[e^{-\frac{1}{k_bT}(2(0)x_1-0-x_1)} + e^{-\frac{1}{k_bT}(2(1)x_1-1-x_1)}\right]$$

$$= \frac{1}{Z}\left[e^{\frac{x_1}{k_bT}} + e^{-\frac{1}{k_bT}(x_1-1)}\right]$$

$$= \frac{1}{Z}\left[e^{\frac{x_1}{k_bT}} + e^{\frac{(1-x_1)}{k_bT}}\right]$$

$$= \frac{e^{\frac{x_1}{k_bT}}+e^{\frac{(1-x_1)}{k_bT}}}{2\left(1+e^{\frac{1}{k_bT}}\right)} \qquad (21)$$

ii. NAND gate

The clique energy, U,

$$U = -[(1 - x_0)(1 - x_1)x_2 + (1 - x_0)x_1x_2 + x_0(1 - x_1)x_2 + x_0\, x_1(1 - x_2)]$$

$$= -[x_2 + x_0\, x_1 - 2\, x_0\, x_1\, x_2]$$

$$= -x_2 - x_0\, x_1 + 2\, x_0\, x_1\, x_2 \qquad (22)$$

Partition function, Z,

$Z = \sum_w e^{-U(w)}$

w = ($x_0$, $x_1$, $x_2$) or all possible states

Therefore, the partition function, Z becomes,

$$\begin{aligned} Z &= \textstyle\sum_{w} e^{-\frac{1}{k_bT}(-x_2 - x_0x_1 + 2x_0x_1x_2)} \\ &= 4e^{\frac{1}{k_bT}} + 4 \\ &= 4\left(1 + e^{\frac{1}{k_bT}}\right) \end{aligned} \tag{23}$$

The output probability,

$$\begin{aligned} p(x_2) &= \frac{1}{Z}\textstyle\sum_{x_0=\{0,1\}}\sum_{x_1=\{0,1\}} e^{-\frac{1}{k_bT}(-x_2 - x_0x_1 + 2x_0x_1x_2)} \\ &= \frac{1}{Z}\textstyle\sum_{x_0=\{0,1\}}\left[e^{-\frac{1}{k_bT}(-x_2-0+0)} + e^{-\frac{1}{k_bT}(-x_2-x_0+2x_0x_2)}\right] \\ &= \frac{1}{Z}\textstyle\sum_{x_0=\{0,1\}}\left[e^{\frac{x_2}{k_bT}} + e^{-\frac{1}{k_bT}(2x_0x_2-x_0-x_2)}\right] \\ &= \frac{1}{Z}\left\{\left[e^{\frac{x_2}{k_bT}} + e^{-\frac{1}{k_bT}(-x_2)}\right] + \left[e^{\frac{x_2}{k_bT}} + e^{-\frac{1}{k_bT}(2x_2-1-x_2)}\right]\right\} \\ &= \frac{1}{Z}\left[3e^{\frac{x_2}{k_bT}} + e^{\frac{(1-x_2)}{k_bT}}\right] \\ &= \frac{3e^{\frac{x_2}{k_bT}} + e^{\frac{(1-x_2)}{k_bT}}}{4\left(1+e^{\frac{1}{k_bT}}\right)} \end{aligned} \tag{24}$$

The expression of the state probability for the remaining basic logic gates, AND, NOR and OR were derived in the same way. The beauty of MRF implementation is that this output probability which corresponds to correct logic operation is maximized through two requirements:

i. Bistable storage element
ii. Feedback network

# CHAPTER 5

# CONCLUSION AND RECOMMENDATION

## 5.1 Conclusion

In MRF design of logic gates, the logical computation is embedded directly in a network with immunity to noise. The key advantage of the MRF approach is that its operation does not depend on perfect devices or connections. Achieving the correct state configuration in the network corresponds to propagating state values through the network and updating each node assignment with a node state having the maximum probability. Successful operation only requires that the joint energy of correct states be lower than the energy of errors (or the probability is maximum when the total clique energy is a minimum), thus fault tolerance is built in.

## 5.2 Recommendation

Probabilistic circuit with MRF implementation proves to be a promising solution to the problem of random and dynamic failure due to thermal noise in nanoscale devices. However, there is still a lot of work need to be done. As a recommendation for further research, MRF-implemented logic circuit need to demonstrate its signal fault tolerance, using the architecture presented in this project.

# REFERENCES

[1] S. Folling, O. Turel, and K. Likharv, *Single-electron Latching Switches as Nanoscale Synapses*.

[2] J. Von Neumann, *Probabilistic Logic and the Synthesis of Reliable Organisms from Unreliable Components*, Princeton University Press, 1956.

[3] R.I. Bahar, et al., *A Probabilistic Approach to Nano-computing*.

[4] K. Nepal, et al., *Designing Logic Circuits for Probabilistic Computation in the Presence of Noise*.

[5] I.C. Wey, et al., *A 0.18 um Probabilistic-Based Noise-Tolerate Circuit Design and Implementation with 28.7 dB Noise-Immunity Improvement*.

[6] X. Lu, X. Song, and J. Li, *The Probabilistic Logic for Nanosize Circuits*.

[7] H. Rao, et al., *An Efficient Methodology to Evaluate Nanoscale Circuit Fault-tolerance Performance Based on Belief Propagation*.

# APPENDICES

## APPENDIX A

## MATLAB Code of Square Wave Signal Corrupted by AWGN (SNR 10 dB)

```
>> fs = 100;
>> t = 0:1/fs:5;
>> x = square(pi*t);
>> y = awgn(x,10);
>> plot(t,x,t,y), axis([0 5 -2 2])
>> legend ('Original signal','Signal with AWGN')
```

# APPENDIX B

## MATLAB Code of Marginalized Inverter Output Distribution Function Plot for Various Values of $k_bT$

```
>> x=0:0.1:1;
>> T1=0.5;
>> T2=0.25;
>> T3=0.1;
>> p1=(exp(x/T1)+exp((1-x)/T1))/(2*(1+exp(1/T1)));
>> p2=(exp(x/T2)+exp((1-x)/T2))/(2*(1+exp(1/T2)));
>> p3=(exp(x/T3)+exp((1-x)/T3))/(2*(1+exp(1/T3)));
>> plot(x,p1,'-'), axis([0 1 0 1])
>> hold on
>> plot(x,p2,'-.'), axis([0 1 0 1])
>> hold on
>> plot(x,p3,':'), axis([0 1 0 1])
>> legend('T=0.5','T=0.25','T=0.1')
>> xlabel('X')
>> ylabel('p(X)')
>> title('Graph of probability of an inverter output as a function of T')
```