PROCEEDINGS OF THE 2004 IASTED INTERNATIONAL CONFERENCE ON CIRCUITS, SIGNALS, AND SYSTEMS (CSS2004), PP. 90-95, CLEARWATER BEACH, FLORIDA, NOVEMBER, 2004

ON THE ANALYSIS OF DIGITAL CIRCUITS WITH UNCERTAIN INPUTS

Houssain Kettani
Department of Computer Science
Jackson State University
Jackson, MS 39217
houssain.kettani@jsums.edu

Abstract

Unlike the classical deterministic digital circuit analysis, we consider the analysis of uncertain digital circuits defined as follows. Given a binary function of n uncertain input binary variables, we express the probability of this binary function in terms of the probabilities of the corresponding input binary variables. This in turn, allows us to estimate appropriate probabilistic measure of the output of a digital circuit with uncertain input parameters and answer typical questions that arise in stochastic optimization.

Keywords: Digital Circuits, Stochastic Optimization, Number Base System, Conversion.

1 Introduction

The theory of deterministic digital circuits has been studied extensively — See [5] for example. Typically, a binary variable x_j is allowed to take only two values; 0 and 1. A binary function of n binary variables $f(x_n, x_{n-1}, \ldots, x_1)$, is also allowed to take only the values 0 and 1. These values may stand for false and true, or low and high. The latter is implemented through voltage levels; low for low voltage level, and high for high voltage level. We typically have a threshold level that separates the two levels.

However, due to noise, temperature variation, signal delay, and other factors, voltage level can be a random variable. Thus, it makes sense to consider the case when the binary variable x_j is a random variable too. Furthermore, if we consider a binary function of n binary variables $f(x_n, x_{n-1}, \ldots, x_1)$, then this in turn would be a random variable.

The introduction of uncertainty in digital circuits has been used in different areas to model complex systems. For example, see [7] for the introduction of probabilistic Boolean networks to model gene regulatory networks. In such a model, the x_j's represent the state of gene j, where $x_j = 1$ denotes the fact that gene j is expressed and $x_j = 0$ means it is not expressed. The binary function $f_j(x_n, x_{n-1}, \ldots, x_1)$, on the other hand, is referred to as a predictor, and is used to determine the value of x_j in terms of some other gene states.
In this paper, we reflect on multi-input mono-output digital networks as depicted in Figure 1. We consider the case when the binary variables x_j, $j = n, n-1, \ldots, 1$, are random. Thus, if we consider a binary function of the n random binary variables, $f(x_n, x_{n-1}, \ldots, x_1)$, then this in turn would be a random binary variable. In this context, all the variables under consideration are Bernoulli random variables since they take only the values 0 and 1.

Hence, throughout this paper, we consider the case when the x_j's are independent random variables with probabilities

$$P(x_j = 1) \doteq p_j = E[x_j].$$

Next, we consider the probability or expectation

$$\begin{aligned} \mathcal{P} &\doteq P(f(x_n, x_{n-1}, \ldots, x_1) = 1) \\ &= E[f(x_n, x_{n-1}, \ldots, x_1)]. \end{aligned}$$

We then pose and answer the following questions: Given a binary logic function, $f(x_n, x_{n-1}, \ldots, x_1)$, with known probabilities p_j's, what can we say about the probability $\mathcal{P}$? How can we address the problem of maximizing or minimizing $\mathcal{P}$? The latter may refer to best case or worst case scenarios.

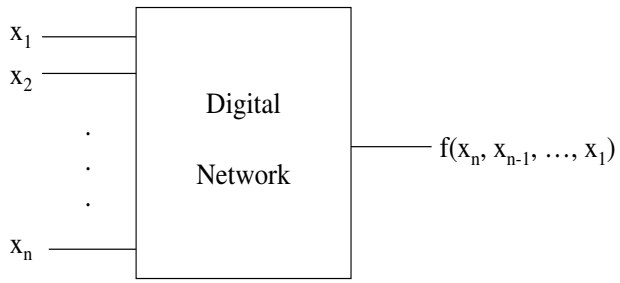

Figure 1. Network Configuration

The flow of this paper is as follows. In Section 2, we

consider the problem of converting a decimal number to a base b number. We present a conversion function that relates each digit in the base b system to the decimal integer value that is equal to the base b number in question. Thus, each base b digit of the related base b number can be obtained directly from the corresponding decimal number without the requirement of knowing any other base b digit. The result of Section 2 is of particular interest in the proof of the result in Section 3.

In Section 3, we present and prove a result that expresses the probability $\mathcal{P}$ in terms of the probabilities p_j's, with $j = n, n-1, \ldots, 1$. Section 4 answers the question of maximizing and minimizing $\mathcal{P}$. We present a numerical example in Section 5 to illustrate the ideas of this paper. Finally, a summary and a suggestion of further research directions is presented in Section 6.

2 Conversion between Decimal and Base b Numbers

We represent an unsigned b-ary, radix b, or base b number as the following string of digits:

$$(d_n d_{n-1} \ldots d_0 . d_{-1} d_{-2} \ldots d_{-m+1} d_{-m})_b,$$

where the integer $b \geq 2$, $d_j \in \{0, 1, \ldots, b-1\}$, $j = n, n-1, \ldots, -m+1, -m$, and "." is the radix point.

The term to the left of the radix point is referred to as the integer part, while that to the right of the radix point is referred to as the fractional part.

The everyday number system that we use is the decimal (base 10). Other number systems that are used in computer work are binary (base 2), octal (base 8), and hexadecimal (base 16).

The classical way to convert a base b_1 number to another number in base b_2, so that both have the same decimal value, is to first convert base b_1 to decimal then convert from the latter to base b_2 — for example, see [5].

A more convenient conversion method is adopted when $b_1 = b_2^p$ where p is a positive decimal integer. Then, to convert from base b_2 to base b_1, we gather p base b_1 digits (from right to left for the integer part, and from left to right for the fraction part and pad with zeros if necessary) into one base b_2 digit so that both numbers have the same decimal value.

On the other hand, to convert from base b_1 to base b_2, we expand one base b_2 digit into p base b_1 digits so that both have the same decimal value. In other words, the relation between base b and base b^p is given by

$$(\ldots d_1 d_0 . d_{-1} \ldots)_b = (\ldots d'_1 d'_0 . d'_{-1} \ldots)_{b^p},$$

where

$$\begin{aligned} d'_j &= (d_{pj+p-1} \ldots d_{pj+1} d_{pj})_b \\ &= \sum_{i=0}^{p-1} d_{pj+i} b^i. \end{aligned}$$

To illustrate this point, consider conversion between binary and hexadecimal. In this case, we have $b_2 = 2$ and $p = 4$. As an illustrative numerical example, we have $(00100101)_2 = (25)_{16}$.

Suppose now we would like to write

$$(d)_{10} = (d_n d_{n-1} \ldots d_0 . d_{-1} d_{-2} \ldots d_{-m+1} d_{-m})_b,$$

where the left hand side is a decimal number and the right hand side is a base b number. Then, by definition, to convert from base b to decimal, we write

$$d = \sum_{j=-m}^{n} d_j b^j. \tag{1}$$

Alternatively, the classical way to convert from a decimal number d to a base b number, is to repeatedly divide the integer part of d by b and record the remainders. Then, starting with the first recorded one, we write down these remainders from right to left starting to the left of the radix point. Thus, we obtain the integer part of the corresponding base b number.

On the other hand, the fraction part is obtained as follows. We repeatedly multiply the fraction part of d by b and record the resultant integer. Then, starting with the first recorded one, we write down these integers from left to right starting to the right of the radix point. Thus, we obtain the fraction part of the corresponding base b number — see [5] for details and illustrative examples.

However, is there any simple function that we can use to convert from decimal to base b? Moreover, what if we are only interested in the value of d_j for some $j \geq 0$? Then based on the repeated division algorithm, we need to compute all the values d_k for $k = 0, 1, \ldots, j$. In addition, what if we are only interested in the value of d_j for some $j < 0$? Then based on the repeated multiplication algorithm, we need to compute all the values d_k for $k = -1, -2, \ldots, j$.

The following theorem answers the questions just posed and presents a very simple and useful result. By this theorem, the d_j's are directly accessible without the requirement of computing any other d_j except the one of interest. The result of this conversion theorem is of particular interest in the proof of Theorem 3.1. This theorem and its proof was presented in [2]. We include the theorem and its proof in this paper for the sake of

completeness. In this theorem, we use $\lfloor \cdot \rfloor$ to indicate the floor function. The floor function is defined by setting $\lfloor x \rfloor$ to the closest integer less than or equal to x.

2.1 Theorem

Let $d \geq 0$ be a decimal number and $b \geq 2$ be a decimal integer. Let also $d_j \in \{0, 1, \ldots, b-1\}$, $j = n, n-1, \ldots, -m+1, -m$. Suppose that the decimal number d is expressible as

$$(d)_{10} = (d_n d_{n-1} \ldots d_0 . d_{-1} \ldots d_{-m+1} d_{-m})_b. \quad (2)$$

Then, (2) is equivalent to

$$d_j = \lfloor db^{-j} \rfloor - b \lfloor db^{-j-1} \rfloor \quad (3)$$

2.2 Proof of Theorem

In this section, we prove Theorem 2.1. To prove the if part, we have

$$\begin{aligned} \sum_{j=-m}^{n} d_j b^j &= \sum_{j=-m}^{n} (\lfloor db^{-j} \rfloor - b \lfloor db^{-j-1} \rfloor) b^j \\ &= \sum_{j=-m}^{n} \lfloor db^{-j} \rfloor b^j - \sum_{j=-m}^{n} \lfloor db^{-j-1} \rfloor b^{j+1} \\ &= \lfloor db^m \rfloor b^{-m} - \lfloor db^{-n-1} \rfloor b^{n+1}. \end{aligned}$$

Now, note that the second term is equal to zero since $0 \leq db^{-n-1} \leq 1$. This is the case since otherwise we need an extra digit to the left to represent d. Note also that the term db^m is an integer. This is also the case since otherwise we need an extra digit to the right to represent d. Thus,

$$\sum_{j=-m}^{n} d_j b^j = d,$$

and this concludes the proof of the first part of the theorem.

Before proceeding to prove the second part of the theorem, we adopt the following generalization of the divisibility concept.

Generalized Divisibility: *Let a_1 and a_2 be two real numbers. We say a_1 is divisible by a_2 if the ratio a_1/a_2 is an integer.*

Now, to prove the only if part, we first let $y = db^{-j-1}$, and consider two cases:

Case 1: $y \in \mathbb{N}$. This means that d is divisible by b^{j+1}. This in turn, forces $d_k = 0$ for $k \leq j$. Accordingly, $\lfloor by \rfloor - b \lfloor y \rfloor = 0$.

Case 2: $y \notin \mathbb{N}$. Then we can write $y = x + \epsilon$, where $x \in \mathbb{N}$ and $\epsilon \in (0, 1)$. Thus,

$$\lfloor by \rfloor - b \lfloor y \rfloor = \lfloor b\epsilon \rfloor.$$

On the other hand, we can write

$$d = xb^{j+1} + \epsilon b^{j+1}.$$

Now, the term xb^{j+1} would modify d_k for some $k > j$. To see the effect on d_j, we need to observe the term ϵb^{j+1}. To do this, we partition the interval $(0, 1)$ into b subintervals of equal length. Thus, we consider the following cases $\epsilon \in [\frac{b-i}{b}, \frac{b-i+1}{b})$, $i = b, b-1, \ldots, 1$. Therefore, we have $(b-i)b^j \leq \epsilon b^{j+1} < (b-i+1)b^j$, which implies that $d_j = b - i$. Accordingly, $\lfloor b\epsilon \rfloor = b - i$. This concludes the proof.

2.3 Remarks

The minimum value of index n for which (2) holds can be easily expressed from (1) and is given by

$$n_{min} = \lfloor log_b d \rfloor.$$

The minimum value of index m, on the other hand, is not as elegantly expressible. In fact, m may diverge to infinity. For example, consider the decimal number 0.1, then we have

$$(0.1)_{10} = (0.0001100110011\ldots)_2.$$

In this case, although the decimal fractional part has finite number of digits, the corresponding binary fractional part does not have finite number of bits.

There are other equivalent expressions for expressing the d_j's in (3). In fact, it can easily be shown that the values of the d_j's in (3) are also equal to $\lfloor b(db^{-j-1} - \lfloor db^{-j-1} \rfloor) \rfloor$, and $\lfloor b^{-j} mod(d, b^{j+1}) \rfloor$. This follows from the properties of the modulus and floor functions.

Let us now compare the efficiency of the classical algorithm to that of the new algorithm introduced in Theorem 2.1. Let us first remind ourselves of what the problem is. Given a decimal number d, we would like to find the value of j^{th} digit in the corresponding base b number. To this end, A brute-force implementation of the classical algorithm will require $2j$ multiplications, $2j$ additions, and j applications of the floor function. The new algorithm on the other hand, which can be viewed as a divide-and-conquer algorithm for base conversion, this algorithm requires $3 + 2j$ multiplication, one addition, and two applications of the floor function.

Thus, even though both algorithms are linear in multiplication, the classical algorithm remains linear in addition and floor while the new algorithm is constant. This in turn shows that the new algorithm is more efficient and consequently faster than the classical algorithm.

3 Stochastic Measures

Suppose the binary variables x_j's, $j = n, n-1, \ldots, 1$, are independent random variables with given probabilities

$$P(x_j = 1) \doteq p_j = E[x_j].$$

Given a binary logic function of the n binary random variables, $f(x_n, x_{n-1}, \ldots, x_1)$, let us define

$$\begin{aligned} \mathcal{P} &\doteq P(f(x_n, x_{n-1}, \ldots, x_1) = 1) \\ &= E[f(x_n, x_{n-1}, \ldots, x_1)]. \end{aligned}$$

Then what can we say about the probability (or expectation) $\mathcal{P}$? How can we address the problem of maximizing or minimizing $\mathcal{P}$?

To answer these questions, let us consider the following theorem.

3.1 Theorem

Let $f(x_n, x_{n-1}, \ldots, x_1)$ be a binary function of n independent binary random variables with $P(x_j = 1) = p_j$. Let I be the set of minterm indices for which $f(x_n, x_{n-1}, \ldots, x_1)$ is 1. Then

$$\mathcal{P} = \sum_{i \in I} \prod_{j=1}^{n} P(x_j = \lfloor i 2^{-j+1} \rfloor - 2 \lfloor i 2^{-j} \rfloor). \qquad (4)$$

3.2 Proof of Theorem

We devote this section to proving Theorem 3.1. Let $f(x_n, x_{n-1}, \ldots, x_1)$ be a binary function of n variables. Let I be the set of minterm indices for which $f(x_n, x_{n-1}, \ldots, x_1)$ is 1. Clearly we have $0 \leq i < 2^n$ for $i \in I$. Now note that any binary function can be written as a sum of its minterms — see [5] for details. Thus, we write

$$f(x_n, x_{n-1}, \ldots, x_1) = \sum_{i \in I} m_i.$$

Let us now write $(i)_{10} = (i_n i_{n-1} \ldots i_1)_2$, where $i_j \in \{0, 1\}$. Hence, we have

$$m_i = x_n^{i_n} x_{n-1}^{i_{n-1}} \ldots x_1^{i_1} = \prod_{j=1}^{n} x_j^{i_j},$$

where we adopt the notion that $x_i^0 = \overline{x}_j$; which is the complement of x_j, and $x_j^1 = x_j$. Suppose now that $P(x_j = 1) = p_j$, and let

$$\mathcal{P} \doteq P(f(x_n, x_{n-1}, \ldots, x_1) = 1).$$

Now, since the events $\{m_i = 1\}$ are mutually exclusive, we have

$$P(f(x_n, x_{n-1}, \ldots, x_1) = 1) = \sum_{i \in I} P(m_i = 1).$$

Next, note that

$$P(m_i = 1) = \prod_{j=1}^{n} P(x_j = i_j),$$

since x_j's are independent. Now, applying Theorem 2.1 we have

$$i_j = \lfloor i 2^{-j+1} \rfloor - 2 \lfloor i 2^{-j} \rfloor.$$

Thus, (4) follows and this concludes the proof.

3.3 Remark

In Theorem 3.1, the set I consists of those minterms that are equal to 1. Thus, if a minterm is a "don't-care," it will not be included in the sum in (4). In other words, the index of such a minterm is not a member of I. This is the case since, by definition, a "don't care" is a condition that does not happen.

On another note, the variance of the binary function can easily be obtained and is expressed as

$$Var[f(x_n, x_{n-1}, \ldots, x_1)] = \mathcal{P} - \mathcal{P}^2.$$

4 Stochastic Optimization

Suppose that the probabilities p_j can be picked from intervals $\mathcal{I}_j = [p_j^-, p_j^+]$. Consequently, the tuple $(p_1, p_2, \ldots, p_n)$ can be picked from the hypercube

$$\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2 \times \ldots \times \mathcal{I}_n.$$

Then, what value should we set the probabilities p_j to in order to maximize or minimize $\mathcal{P}$? To answer this question, we first introduce the following definition.

4.1 Essential Variables

A binary variable x_k is said to be *essential* if the following condition holds. There does not exist admissible values of the $(n-1)$ remaining variables $x_j \in \mathcal{I}_j$, $j \neq k$ making the probability $\mathcal{P}$ independent of $x_k \in \mathcal{I}_k$.

If x_k is essential, it can readily be shown that the partial derivative $\partial \mathcal{P} / \partial p_k$ is non-zero over $\mathcal{I}$. Hence, by an intermediate value argument, if the variable x_k is essential, the partial derivative $\partial \mathcal{P} / \partial p_k$ has one sign over $\mathcal{I}$. In view of this, let

$$s_k \doteq \text{sign}\left(\frac{\partial \mathcal{P}}{\partial p_k}\right)$$

denote this invariant sign; i.e., s_k is constant over $\mathcal{I}$ having the value $s_k = -1$ or $s_k = 1$.

The following theorem answers the question posed in the introduction of this section.

4.2 Theorem

Let $\mathcal{P}$ be a function of some p_j's. For the case of maximizing $\mathcal{P}$, if the variable x_k is essential, then pick $p_k = p_k^-$ when $s_k = -1$, and pick $p_k = p_k^+$ when $s_k = 1$.

For the case of minimizing $\mathcal{P}$, if the variable x_k is essential, then pick $p_k = p_k^+$ when $s_k = -1$, and pick $p_k = p_k^-$ when $s_k = 1$.

If x_k is not essential, then for either case pick $p_k = p_k^-$ or $p_k = p_k^+$.

4.3 Proof of Theorem

To prove Theorem 4.2 we first note that Theorem 3.1 shows that $\mathcal{P}$ is multilinear in the p_i's. Thus, to maximize or minimize $\mathcal{P}$ we invoke the well-known result that a multilinear function on a hypercube is both maximized and minimized at its vertices — see [4] for example. Vertex points here are the tuples $(p_1^{\pm}, p_2^{\pm}, \ldots, p_n^{\pm})$, where $p_j^{\pm}$ stands for the extreme values p_j^- or p_j^+.

Now, if the variable x_j is essential, then $s_j = -1$ implies that $\mathcal{P}$ is decreasing with respect to p_j. Thus, to maximize $\mathcal{P}$, we decrease p_j, and to minimize $\mathcal{P}$, we increase p_j. Similarly, $s_j = 1$ implies that $\mathcal{P}$ is increasing with respect to p_j. Thus, to maximize $\mathcal{P}$, we increase p_j, and to minimize $\mathcal{P}$, we decrease p_j. Since $p_j \in [p_j^-, p_j^+]$, the result of the theorem follows.

5 Numerical Examples

To illustrate the use of the ideas introduced in this paper, we consider two logic functions:

$$f_1(x_3, x_2, x_1) = x_3 + \overline{x}_1,$$

and

$$f_2(x_3, x_2, x_1) = \overline{x}_2 x_1 + x_3 \overline{x}_1,$$

with $p_1 \in [0.4, 0.6]$, $p_2 \in [0.1, 0.5]$, and $p_3 \in [0.2, 0.8]$.

Let I_1 and I_2 be the sets of minterm indices for which $f_1(x_3, x_2, x_1) = 1$ and $f_2(x_3, x_2, x_1) = 1$, respectively. Then $I_1 = \{0, 2, 4, 5, 6, 7\}$ and $I_2 = \{1, 4, 5, 6\}$. Thus, from (4), we have

$$\mathcal{P}_1 = (1 - p_3)(1 - p_1) + p_3$$

and

$$\mathcal{P}_2 = (1 - p_2)p_1 + (1 - p_1)p_3.$$

We note here that $\mathcal{P}_2$ can be expressed directly from the expression of $f_2(x_3, x_2, x_1)$. This is because $f_2(x_3, x_2, x_1)$ is expressed as the sum of products that are mutually exclusive. Note that a similar argument cannot be made with regard to $f_1(x_3, x_2, x_1)$.

Now, suppose we would like to maximize $\mathcal{P}_1$. Then note that both x_1 and x_3 are essential with $s_1^{(1)} = -1$ and $s_3^{(1)} = 1$, respectively. Thus, the maximum $\mathcal{P}_1^+$ is obtained with $p_1 = 0.4$ and $p_3 = 0.8$. Consequently, $\mathcal{P}_1^+ = 0.92$.

Suppose now that we would like to minimize $\mathcal{P}_2$. Then note that both x_2 and x_3 are essential with $s_2^{(2)} = -1$ and $s_3^{(2)} = 1$, respectively. Thus, the minimum $\mathcal{P}_2^-$ is obtained with $p_2 = 0.5$ and $p_3 = 0.2$. However, the variable x_1 is not essential. Thus, we try both values 0.4 and 0.6 for p_1, which results in $\mathcal{P}_2 = 0.32$ and $\mathcal{P}_2 = 0.38$, respectively. Consequently, $\mathcal{P}_2^- = 0.32$.

6 Summary and Further Research

We started the paper discussing conversion between any two number systems. Unlike the classical conversion from decimal to base b system, we have presented a functional conversion that expresses each digit in base b in terms of the decimal number in question. This in turn, provides us with a direct access to each base b digit instead of the need of knowing any previous base b digits.

Next, we considered the case of digital circuits with uncertain input variables. We have presented a probabilistic measure of the output function in terms of the probabilities of the input. The result is a multilinear function, which facilitates the optimization problem of the probability of the output.

With respect to number base conversion, we suggest several research directions as follows. A general version of number systems suggests that b can be any nonzero number (real or imaginary) and that the d_j's can be chosen from any specific set of numbers so that any number is uniquely expressible in this base. In such case, very interesting results and properties can arise. For more information and historical prospective on such general case, see [3, pp. 195 – 213]. Thus, it would be of interest to generalize the result of this paper to the case when $b \in \mathbb{C}^*$ and d_j's are chosen from any appropriately defined set.

On the other hand, we note that the conversion result of Section 2 may have a strong impact on many areas of research. For example, consider the problem of converting a very large decimal number to binary. Then, we can use the method suggested in this paper together with parallel computing to optimize with respect to time and space. In other words, we can compute each bit or set of bits independently of the others, then the final result is

constructed from the results of the subproblems. The latter is a nice example of a divide-and-conquer algorithm.

With respect to the analysis of logic circuits with uncertain input variables, we suggest four research directions. Firstly, it would be of interest to consider the case when the input variables are dependent. In this case, the challenge is that the joint distribution cannot be expressed as the product of the marginal distributions. Thus, we have

$$\mathcal{P} = \sum_{i \in I} P(x_n = i_n, x_{n-1} = i_{n-1}, \ldots, x_1 = i_1).$$

The second suggested research direction is in b-ary logic. Although binary logic is by far the most practical among any b-ary logic, there is some use of the trinary (or ternary) logic — see [1] and [6] for more information. A fundamental issue in this case is that minterm concept is not properly defined. As a consequence, the events of the term in which all the variables appear exactly once may not be mutually exclusive.

Thirdly, it would be of interest to consider multi-output networks as depicted in Figure 2. In such case, we have m output binary function, each is a function of the n input binary variables. We also have

$$\mathcal{P}_i \doteq P(f_i(x_n, x_{n-1}, \ldots, x_1) = 1),$$

for $i = m, m-1, \ldots, 1$. The challenge here is that since f_i's are typically dependent, the maximizers of the probability of one function may not the maximizers of the probability for another function. So how do we mitigate such effect.

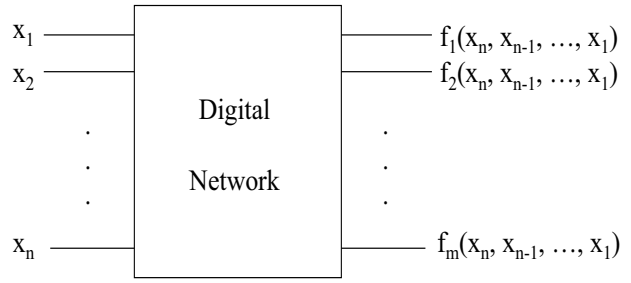

Figure 2. Network Configuration

Finally, it would be of interest to broaden the concept of uncertain digital networks to include uncertain logic gates and extend the results of this paper to such case. For example, an uncertain AND gate would have a corresponding probability $P(xy = 1) = p$. This in turn, would lead to a new probabilistic Boolean algebra.

References

[1] S. Grubb (2001), www.trinary.cc.

[2] H. Kettani (2004), "*On the Conversion between Number Systems*," proceedings of the 2004 International Conference on Algorithmic Mathematics and Computer Science (AMCS'04), Las Vegas, Nevada, June, 2004.

[3] D. E. Knuth (1998), "*The Art of Computer Programming, Volume 2: Seminumerical Algorithms*," third edition, Addison-Wesley.

[4] D. G. Luenberger (1973), "*Introduction to Linear and Nonlinear Programming.*" Addison-Wesley Publishing Company.

[5] M. M. Mano and C. R. Kime (2001), "*Logic and Computer Design Fundamentals*," second edition, Prentice Hall.

[6] T. Sasao (1999), "*Arithmetic Ternary Decision Diagrams Applications and complexity*," proceedings of the Fourth International Workshop on Applications of the Reed-Muller Expansion in Circuit Design, (Reed-Muller 99), Victoria, Canada, August, 1999.

[7] I. Shnulevich, E. R. Dougherty, S. Kim, and W. Zhang (2002), "*Probabilistic Boolean Networks: A Rule-Based Uncertainty Model for Gene Regulatory Networks*," Bioinformatics, Vol. 18, pp. 261–274.