Computing Polynomials with Positive Coefficients using Stochastic Logic by Double-NAND Expansion *

Sayed Ahmad Salehi, Yin Liu, Marc D. Riedel, Keshab K. Parhi
University of Minnesota
200 Union Street SE, Minneapolis, MN, 55455
{saleh022, liux1394, mriedel, parhi}@umn.edu

ABSTRACT

This paper proposes a novel method, referred to as *double-NAND expansion*, to implement polynomials with all positive coefficients using unipolar stochastic logic. The proposed double-NAND expansion leads to implementations of polynomials using no more than $2n$ NAND gates where n represents the degree of the polynomial. The proposed implementations are compared with those based on multiplexers, Bernstein polynomial method, finite state machine method and factorization. The paper also considers implementations of several functions expressed as polynomials using truncated Mclaurin series based on the proposed approach. The experimental results show that the proposed method outperforms the prior methods in terms of accuracy, hardware complexity, and critical path.

Keywords

Stochastic logic; double-NAND expansion

1. INTRODUCTION

This paper presents a novel method for implementing polynomials with all positive coefficients such that the sum of all coefficients is less than or equal to 1 by using stochastic logic circuits that require only two NAND gates for each term of the polynomial.

Stochastic logic circuits can be realized by simple combinational logic gates [2]. In this paper we only use AND and NAND gates for polynomial computations. Using unipolar representation, the AND gate computes the multiplication of its two inputs, $z = xy$ and the NAND gate computes $z = 1 - xy$. Although the proposed circuits are primarily based on NAND gates, for computation of some polynomials, AND gates also may be required only at the beginning and end of the circuits.

*This research is supported by the National Science Foundation Grants CCF-1423407 and CCF-1319107.

GLSVLSI '17, May 10-12, 2017, Banff, AB, Canada.

DOI: http://dx.doi.org/10.1145/3060403.3060410

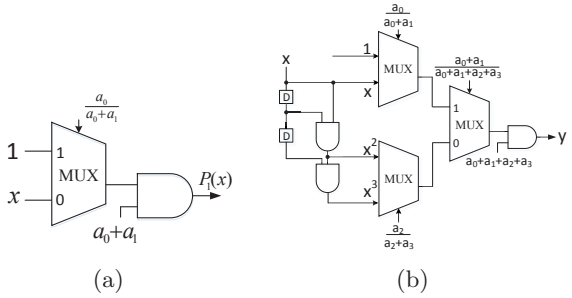


Figure 1: Multiplexer-based Stochastic logic circuits for computing (a) first-order polynomial and (b) third-order polynomial.

This paper considers a polynomial of degree n, i.e., $P_n(x)$, represented in power form as:

$$P_n(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_nx^n. \quad (1)$$

All a_i's are assumed to be positive or zero, and sum of a_i's is assumed to be less than or equal to 1.

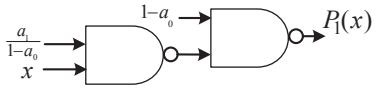
A multiplexer-based stochastic logic circuit for implementation of FIR digital filters was proposed in [1]. We can adapt this circuit for the purpose of computing polynomials with all positive coefficients. Figure 1(a) shows the multiplexer-based circuit for computation of $P_1(x) = a_0 + a_1x$. Similarly, $P_2(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ can be implemented by the circuit shown in Figure 1(b). The delay elements are used for decorrelation [5]. Factorization method, presented in [4], is another approach to synthesize stochastic logic circuits for computing polynomials.

This paper proposes a new approach that can map polynomials having all positive coefficients to a hardware-efficient circuit composed of NAND gates. For the proposed method, each term of the polynomial is implemented by two two-input NAND gates and the proposed expansion of polynomials is referred to as *double-NAND* expansion.

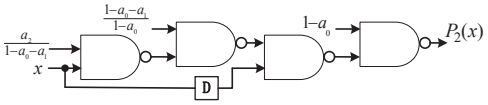
This paper is organized as follows. Section 2 presents the proposed method for computing a class of polynomials and certain functions. Section 3 presents the experimental results for the proposed method and comparisons with prior work.

2. PROPOSED DOUBLE-NAND METHOD

Using stochastic logic circuits composed of simple two-input NAND gates, this section presents the proposed method for computing polynomials with all positive coefficients, i.e.,



(a)



(b)

Figure 2: The proposed Stochastic logic circuits for computing (a) first-order polynomial and (b) second-order polynomial.

$0 \leq a_i < 1$ for $i = 0, ..., n$. Each term of the polynomial has the form of $a_i x^i$ and is monotonically increasing. Therefore, the polynomials considered in this paper are monotonically increasing in the interval [0,1). It is important to note that not all monotonically increasing polynomials satisfy the all positive coefficient property as considered in this paper.

First Consider the simple example of the first-order polynomial given by:

$$P_1(x) = a_0 + a_1 x = 1 - (1 - a_0)(1 - \frac{a_1}{(1 - a_0)} x) \quad (2)$$

For a_0,$a_1 \in [0, 1)$, Equation (2) can be mapped to the stochastic logic circuit shown in Figure 2(a). Similarly, the second-order polynomial, $P_2(x)$, with all coefficients in the unit interval can be expressed as (3), and implemented as shown in Figure 2(b).

$$\begin{aligned} P_2(x) &= a_0 + a_1 x + a_2 x^2 \\ &= 1 - (1 - a_0)[1 - x(1 - \tfrac{1-a_0-a_1}{1-a_0}(1 - \tfrac{a_2 x}{1-a_0-a_1}))] \end{aligned} \quad (3)$$

This approach can be extended for the general case of a polynomial of degree n, $P_n(x)$, with all positive coefficients. $P_n(x)$ can be represented as a sum of its first term, a_0, and the other terms, $xP_{n-1}(x)$, as given by:

$$P_n(x) = a_0 + \underbrace{a_1 x + a_2 x^2 + a_3 x^3 + ... + a_n x^n}_{x P_{n-1}(x)} = a_0 + x P_{n-1}(x) \quad (4)$$

Equation (4) can be rewritten as:

$$\begin{aligned} P_n(x) &= a_0 + x P_{n-1}(x) = \\ 1 - (1 - a_0)(1 - x(&\underbrace{\frac{a_1 + a_2 x + a_3 x^2 + ... + a_n x^{n-1}}{1 - a_0}}_{P_{n-1}(x)})) \end{aligned} \quad (5)$$

Equation (5) shows that $P_{n-1}(x)$ has a form similar to the original $P_n(x)$; however, its degree is reduced by one and the coefficients are divided by $(1 - a_0)$. Thus, the above mentioned step can be repeated for $P_{n-1}(x)$ and the procedure can be continued as required. Using this approach $P_n(x)$ can be reformatted as

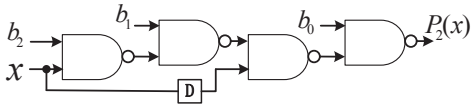


Figure 4: Stochastic logic circuit for Equation (7).

$$\begin{aligned} P_n(x) = 1 - (1 - a_0)(1 - x(1 - \frac{1 - (a_0 + a_1)}{1 - a_0}(1 - x(\\ ...(1 - \frac{1 - \sum_{i=0}^{k-1} a_i}{1 - \sum_{i=0}^{k-2} a_i}(1 - x(... \\ (1 - \frac{1 - \sum_{i=0}^{n-1} a_i}{1 - \sum_{i=0}^{n-2} a_i}(1 - \frac{a_n x}{1 - \sum_{i=0}^{n-1} a_i}))))))))) \end{aligned} \quad (6)$$

THEOREM 1. *All the coefficients for the representation of $P_n(x)$ in Equation (6) are greater than 0 and less than 1.*

PROOF. We know that the sum of all a_i's for $i = 0, ..., n$ is less than or equal to one, i.e., $\sum_{i=0}^{n} a_i \leq 1$. All of the coefficients in Equation (6), except the one in the innermost parentheses, have the form of $\frac{1 - \sum_{i=0}^{k} a_i}{1 - \sum_{i=0}^{k-1}}$ for $k = 0, ..., n - 1$. It is clear that for these coefficients the numerator is less than the denominator. Thus, the values of the coefficients are less than 1. Since $\sum_{i=0}^{n} a_i \leq 1$ we have $a_n + \sum_{i=0}^{n-1} a_i \leq 1$ or equivalently $a_n \leq 1 - \sum_{i=0}^{n-1} a_i$. Thus, the coefficient inside the innermost parentheses is also less than one, i.e., $\frac{a_n}{1 - \sum_{i=0}^{n-1} a_i} \leq 1$. Moreover, the the numerator and denominator of all the coefficients in Equation (6) are greater than 0. Therefore, these coefficients are greater than 0. □

Equation (6) can be mapped to a stochastic logic circuit containing cascaded NAND gates as shown in Figure 3. Note that the delay elements are used for decorrelation [5].

In the proposed circuits, for the first-order polynomial, $P_1(x)$, two NAND gates are required, and for each additional term another two NAND gates are needed. Thus, $2(n_t - 1)$ NAND gates are required to implement a polynomial that has n_t terms. For example, $P_n(x)$ represented in Equation (1) requires $2n$ two-input NAND gates.

We now show that for any stochastic logic circuit implemented by cascaded double-NAND gates, the coefficients add to less than 1. To illustrate this, consider the circuit shown in Figure 4. The second-order polynomial implemented by this circuit is given by:

$$\begin{aligned} P_2(x) &= 1 - b_0(1 - x(1 - b_1(1 - b_2 x))) \\ &= (1 - b_0) + b_0(1 - b_1)x + b_0 b_1 b_2 x^2. \end{aligned} \quad (7)$$

Therefore, $a_0 = 1 - b_0$, $a_1 = b_0(1 - b_1)$, and $a_2 = b_0 b_1 b_2$. For this example, $\sum_{i=0}^{2} a_i = 1 - b_0 b_1 (1 - b_2) < 1$.

For a polynomial of degree n, stochastic logic circuit in Figure 5 implements $P_n(x)$ shown in Equation (8).

$$P_n(x) = 1 - b_0(1 - x(1 - b_1(1 - x(...(1 - b_{n-1}(1 - b_n x)))))) \quad (8)$$

The polynomial represented in Equation (8) can be expressed in power form in Equation (9).

$$\begin{aligned} P_n(x) = (1 - b_0) + b_0(1 - b_1)x + b_0 b_1 (1 - b_2)x^2 + ... \\ + b_0 b_1 .. b_{n-2}(1 - b_{n-1})x^{(n-1)} + b_0 b_1 .. b_{n-1} b_n x^n \end{aligned} \quad (9)$$

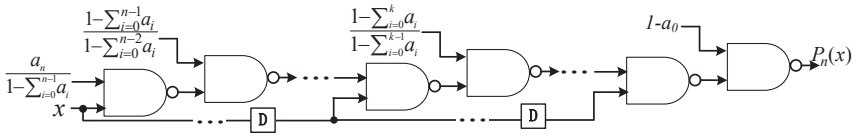


Figure 3: General double-NAND structure for polynomial of degree n. Based on the proposed approach we change the form of Equation (4) to the form of Equation (6) and map it to a cascade of NAND gates.

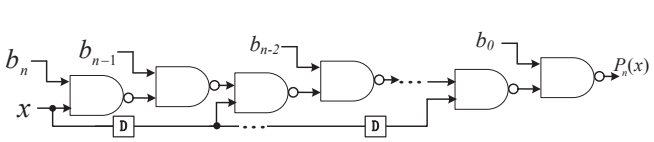


Figure 5: Stochastic logic circuit for implementation of polynomial in Equation (8).

The sum of the coefficients in Equation (9) is equal to $1 - b_0b_1...b_{n-1}(1-b_n)$. Considering $0 \leq b_i \leq 1$ for $i = 0, ..., n-1$ and $0 < b_n \leq 1$, it is easy to show that this sum is less than or equal to 1 and greater than 0.

Example 1: Consider the polynomial $P(x) = \frac{4}{5} + \frac{x}{15} + \frac{x^2}{10}$. Using Equation (6), it can be rewritten as:

$$P(x) = 1 - \frac{1}{5}(1 - x(1 - \frac{2}{3}(1 - \frac{3}{4}x))) \tag{10}$$

This equation can be implemented by the circuit in Figure 4 with $b_0 = \frac{1}{5}$, $b_1 = \frac{2}{3}$ and $b_2 = \frac{3}{4}$.

The proposed approach can be used to implement certain arithmetic functions using stochastic logic. For this purpose, first the target functions are approximated by truncating their Maclaurin series expansions [5].

We consider three functions whose Maclaurin series expansions are listed below.

$$e^{(x-1)} = \frac{1}{e}\sum_{n=0}^{\infty}\frac{(x)^n}{n!} \approx \frac{1}{e} + \frac{1}{e}x + \frac{x^2}{2e} + \frac{x^3}{6e} = 1 - \frac{e-1}{e}(1- x(1 - \frac{e-2}{e-1}(1 - x(1 - \frac{2e-5}{2(e-2)}(1 - \frac{x}{6e-15})))))$$

$$\sec x - 1 = \sum_{n=1}^{\infty}\frac{(-1)^n E_{2n}}{(2n)!}x^{2n} \approx \frac{x^2}{2} + \frac{5}{24}x^4 + \frac{61}{720}x^6 = x^2(1 - \frac{1}{2}(1 - x^2(1 - \frac{7}{12}(1 - \frac{61}{210}x^2))))$$

$$\cosh x - 1 = \sum_{n=0}^{\infty}\frac{x^{2n}}{(2n)!} \approx \frac{x^2}{2} + \frac{x^4}{24} + \frac{x^6}{720} = x^2(1 - \frac{1}{2}(1 - x^2(1 - \frac{11}{12}(1 - \frac{x^2}{330}))))$$

where E_n is an Euler number.

Notice that, since all the terms of the Maclaurin series of these three functions have coefficients that are less than one, are positive, and have sums less or equal to 1, they can be mapped to stochastic logic circuits using the proposed method. Note that because approximations for $\sec(x) - 1$ and $\cosh(x) - 1$ are polynomials in x^2 with no constant term, they require two AND gates in addition to the NAND gates. The implementations are shown in Fig. 6.

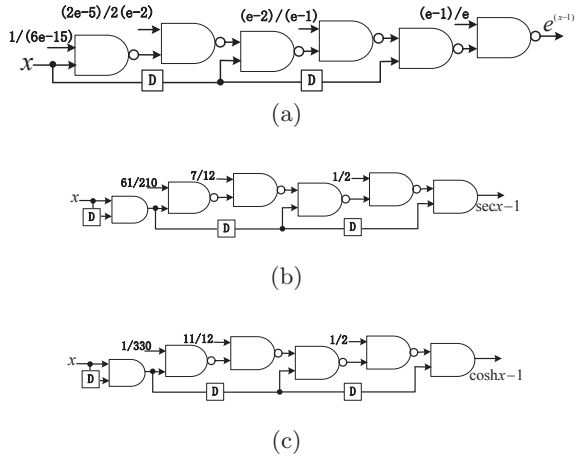


Figure 6: Stochastic implementations of (a) $e^{(x-1)}$, (b) $\sec x - 1$ and (c) $\cosh x - 1$ by mapping Maclaurin series to double-NAND circuits.

3. EXPERIMENTAL RESULTS

In our experiments, we compare accuracy and hardware complexity of the proposed circuits with those of prior work. With respect to the performance, we compare our approach with four other alternative approaches: multiplexer-based [1], Bernstein polynomial [6], finite-state-machine (FSM) using a 2-dimensional FSM with 8 states [3], and factorization [4]. For the factorization method, the scale factors of $e^{(x-1)}$, $\sec(x) - 1$, and $\cosh(x) - 1$ are, respectively, given by $\frac{1}{e}$, $\frac{1}{2}$, and $\frac{1}{2}$.

Table 1 presents the mean absolute error (MAE) of different approaches for computing the target functions. The errors are computed using Montecarlo simulation for each of the 101 inputs, between 0 and 1, i.e., $x = 0, 0.01, 0.02..., 1$. The length of the stochastic bit streams is 1024 bits. Note that stochastic logic circuits of approximate Maclaurin polynomials implemented by the proposed method outperform other methods.

The implementation of double-NAND expansion requires $2n$ NAND gates, $(n+1)$ coefficients and $(n-1)$ delay units needed for decorrelation. The structure presented in [1] requires $3n$ gates to implement n multiplexers, $(n-1)$AND gates and $(n-1)$ delays to calculate x^i for $i = 2$ to n, one AND gate to scale the computed output, and $(n+1)$ coefficients. We assume that each delay unit can be implemented by 10 gates. For the coefficients, we assume that one LFSR is used to generate all coefficients and each coefficient uses a unique comparator and delay unit. A 10-bit LFSR can be realized using 160 gates. Although the number of gates

Table 1: The mean absolute error (MAE) for computing polynomials in Example 1, $e^{(x-1)}$, $\sec(x)-1$ and $\cosh(x)-1$ functions using the proposed method and previous methods.

Function		Proposed			Multiplexer			Bernstein Polynomial			Factorization			FSM
Example 1		0.0040			0.0040			-	-	-	0.0040			-
$e^{(x-1)}$	degree	2	3	4	2	3	4	2	3	4	2	3	4	8-state
	error	0.0197	0.0042	0.0011	0.0199	0.0050	0.0015	0.0253	0.0164	0.0122	0.0199	0.0074	0.0032	0.0163
$\sec(x)-1$	degree	4	6	8	4	6	8	4	6	8	4	6	8	8-state
	error	0.0172	0.0053	0.0027	0.00175	0.0068	0.0034	0.0505	0.0330	0.0245	0.0175	0.0068	0.0061	0.1200
$\cosh(x)-1$	degree	4	6	8	4	6	8	4	6	8	4	6	8	8-state
	degree	0.0016	0.0008	0.0005	0.00017	0.0008	0.0005	0.0235	0.0154	0.0113	0.0017	0.0008	0.0006	0.0399

Table 2: The synthesis results for area in terms of two-input NAND gates and critical path (ns) for different stochastic logic polynomial and functions using 5 approaches.

Function		Proposed			Multiplexer			Bernstein Polynomial			Factorization			FSM
Example 1	area	488.8			494.5			-	-	-	494.5			-
	delay	2.42			2.71			-	-	-	2.71			-
$e^{(x-1)}$	degree	2	3	4	2	3	4	2	3	4	2	3	4	8-state
	area	488.8	559.5	634.4	494.5	569.4	648.9	496.1	569.4	664.0	494.5	563.9	641.2	1144.0
	delay	2.42	2.68	3.02	2.71	2.94	3.16	2.59	2.96	2.90	2.71	2.78	3.11	2.76
$\sec(x)-1$	degree	4	6	8	4	6	8	4	6	8	4	6	8	8-state
	area	463.3	530.9	612	466.4	541.3	626.6	664.0	872.0	1153.2	466.4	541.3	623.5	1144.0
	delay	2.43	2.84	3.03	2.47	3.04	3.47	2.90	4.01	3.98	2.47	3.04	3.44	2.76
$\cosh(x)-1$	degree	4	6	8	4	6	8	4	6	8	4	6	8	8-state
	area	463.3	530.9	612	466.4	541.3	626.6	664.0	872.0	1153.2	466.4	541.3	623.5	1144.0
	delay	2.43	2.84	3.03	2.47	3.04	3.47	2.90	4.01	3.98	2.47	3.04	3.44	2.76

required for the comparators varies for each coefficient, we estimate that an average of 30 gates is required for each comparator. These numbers are taken from our synthesis results. Hence, for $(n+1)$ coefficients $160+30(n+1)+10(n+1)$ gates are used. Collecting these values together, computation of $P_n(x)$ based on the stochastic logic in [1] requires $54n+190$ gates while the double-NAND method requires $52n+190$ gates, leading to saving 4% of the number of gates for a large n.

With respect to the critical path, for the double-NAND circuits the critical path is in the order of n. Since the structure in [1] has $\log_2 n$ levels of multiplexers and the critical path for each multiplexer is about two gates, the critical path for the multiplexer levels is $\mathcal{O}(\lceil \log_2 n \rceil)$. When we consider $(n-1)$ AND gates at the input and 1 AND gate at the output the total critical path for multiplexer-based stochastic logic circuit is $\mathcal{O}(\lceil \log_2 n \rceil + n)$. One should note that, while for small values of n the critical paths for the two methods are close, the difference is remarkable for higher values of n. Table 2 compares area and critical path for all 4 polynomials using discussed methods. The architectures are implemented using 65nm libraries and synthesized using Synopsys Design Compiler. All required SNGs including 10-bit LFSRs as random number generators are considered in our synthesis. The operating conditions for each implementation are specified by a supply voltage of 1.05 V and a temperature of 25 degree Celsius. It can be observed from Table 2 that the proposed stochastic circuits have less hardware complexity compared to prior approaches.

4. CONCLUSIONS

A novel double-NAND expansion approach has been presented to expand any polynomial with positive coefficients where the sum of coefficients is less than or equal to one. Note that the coefficients need not be either increasing or decreasing, but can be in any arbitrary order. The proposed expansion leads to stochastic logic implementations of these polynomials using cascaded two-input NAND gates. The reader is referred to [5] for stochastic logic implementation of polynomials with alternating signs and decreasing magnitude.

5. REFERENCES

[1] Y.-N. Chang and K. K. Parhi. Architectures for digital filters using stochastic computing. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2697–2701. IEEE, May 2013.

[2] B. R. Gaines. Stochastic computing. *in Proceedings of AFIPS spring joint computer conference, ACM*, pages 149–156, 1967.

[3] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. Riedel. The synthesis of complex arithmetic computation on stochastic bit streams using sequential logic. *in Proceedings of the International Conference on Computer-Aided Design*, pages 480–487, 2012.

[4] Y. Liu and K. K. Parhi. Computing complex functions using factorization in unipolar stochastic logic. In *Proceedings of the 26th edition on Great Lakes Symposium on VLSI*, pages 109–112. ACM, 2016.

[5] K. K. Parhi and Y. Liu. Computing arithmetic functions using stochastic logic by series expansion. *IEEE Transactions on Emerging Technologies in Computing (TETC)*, DOI: 10.1109/TETC.2016.2618750.

[6] W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja. An architecture for fault-tolerant computation with stochastic logic. *IEEE Transactions on Computers*, 60(1):93–105, 2011.