Characterizing Polynomial Arithmetic with Stochastic Circuits*

Patrick Holec

Department of Biological Engineering,
Massachusetts Institute of Technology, Cambridge,
MA, USA
E-mail: pholec@mit.com

Weikang Qian
University of Michigan-Shanghai Jiao Tong
University Joint Institute, Shanghai Jiao Tong
University, Shanghai, China
E-mail: qianwk@sjtu.edu.cn

Marc Riedel
Department of Electrical and Computer
Engineering, University of Minnesota, Twin Cities,
Minneapolis, MN, USA
E-mail: mriedel@umn.edu

Ivo Rosenberg
Mathematics and Statistics, University of
Montreal, Montreal, Quebec, Canada
E-mail: rosenb@DMS.UMontreal.CA

Abstract In this work, we outline a novel algorithm for the computation of real coefficient polynomials using stochastic circuits. Also, we prove that there exists restricted regions in polynomial space that no stochastic circuit can compute. Through the use of the fundamental theorem of algebra, we can obtain a finite set of roots $r_1, \ldots, r_i \in \mathbb{C}$ for a target polynomial $P_i(t)$ of the i^{th} degree. If a polynomial contains roots exclusive to the accessible regions of the complex plane, a stochastic circuit is guaranteed to exist which represents the target polynomial as $A \cdot P(t)$ where $0 < A < 1$ or, in certain cases, $A = 1$. When these solutions exist, the number of logic gates and the overall delay of the circuit is significantly less than those described by previous solutions to this problem.

Keywords stochastic computing · probabilistic circuits · polynomials

* This work is submitted posthumously with Ivo Rosenberg as an author. It is based on discussions with Ivo that transformed our thinking of this line of research, putting it on a firm mathematical footing.

1 Introduction

Before explaining the context and motivation for the mathematics in this paper, we first give a brief review of digital circuits. Then we introduce the reader to an alternative paradigm for arithmetic with digital circuits called *stochastic logic*, which is predicated on probability theory. Circuits designed with stochastic logic, called *stochastic circuits*, compute polynomial functions. The main contribution of this paper is to characterize regions of polynomial space that are computable with stochastic circuits.

1.1 Digital Circuits

Conceptually, a digital circuit consists of **gates** connected by **wires**. Each gate has one or more inputs and a single output. The symbols for common gates are shown in Figure 1. A bubble is used to indicate that an input or output is negated, as illustrated in Figure 2.

A gate implements a Boolean function, i.e., a mapping from Boolean inputs to a Boolean output,

$$g : \{0,1\}^k \to \{0,1\}.$$

For instance, an XOR gate performs the mapping

$$\mathrm{XOR}(y_1, \ldots, y_k) = \begin{cases} 1 & \text{if the number of 1's among } y_1, \ldots, y_k \text{ is odd} \\ 0 & \text{otherwise.} \end{cases}$$

The set of inputs to a gate are called its **fan-in** set. When we say a "fan-in k" gate, we mean a gate with fan-in set of cardinality k. The set of gates that are attached to a gate output are called its **fan-out** set. The truth tables for fan-in two AND, OR, NAND, NOR, and XOR gates, as well as a fan-in one NOT gate, are shown Table 1. The gates NAND and NOR are AND and OR gates with the output inverted, respectively.

Table 1: Truth table for common gates.

x	y	AND(x, y)	OR(x, y)	NAND(x, y)	NOR(x, y)	XOR(x, y)
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	1	0	0	0

x	NOT(x)
0	1
1	0

A digital **circuit** is built by attaching gates to the outputs of other gates, so *composing* the gate functions. An example of a circuit is shown in Figure 3. In general:

– A circuit accepts values $x_1, \ldots, x_m$, ranging over $\{0, 1\}$, called the **primary inputs**. Each primary input is fed into one or more gate inputs.
– The gates in the circuit produce **internal values** $z_1, \ldots, z_n$ ranging over $\{0, 1\}$.
– A subset of the set of internal values $\{f_1, \ldots, f_k\} \subseteq \{z_1, \ldots, z_n\}$ are designated as the **primary outputs**.

A circuit computes Boolean **functions** by mapping primary inputs to primary outputs, through the composition of the Boolean functions that the gates perform:

$$\forall_i \;\; f_i : \{0,1\}^m \rightarrow \{0,1\}.$$

(Computer engineers being cavalier about such things, we often abuse the notation, using the same symbols f_i to denote Boolean values at the primary outputs as well as the functions that compute these values.)

The **delay** of circuit is the longest path from a primary input to a primary output, measured in terms of the number of gates along that path. We generally exclude negations (so bubbles in our diagrams) from this count, considering them part of the corresponding gate operations. For instance, the delay of the circuit in Figure 3 is 2.

Throughout this paper, the discussion pertains to **combinational** circuits. Such circuits have no loops and no reconvergence:

– By **loop**, we mean a path along wires and through gates from a gate output back to itself.
– By **reconvergence**, we mean a path along wires and through gates that *splits* and then rejoins itself (without necessarily forming a loop).

Stated more simply, here we limit ourselves to circuits that are **trees** (where wires represent edges and gates nodes in a graph). Each primary output is the *root* of a tree, and each primary input a *leaf*.[1]

1.2 Stochastic Circuits

Humans are accustomed to counting in a positional number system – decimal radix. Nearly all computer systems operate on another positional number system – binary radix. From the standpoint of *representation*, such positional systems are compact: given a radix b, one can represent b^n distinct numbers with n digits. Each choice of the digits $d_i \in \{0, \ldots, b-1\}$, $i = 0, \ldots, n-1$, results in a different number N in $[0, \ldots, b^n - 1]$:

$$N = \sum_{i=0}^{n-1} b^i d_i.$$

However, from the standpoint of *computation*, positional systems impose a burden: for each operation such as addition or multiplication, the value must be "decoded,"

[1] We note that computer engineers generally define **combinational** circuits as those that have no memory, meaning the primary outputs depend only on the primary inputs, not on prior values stored on wires in the circuit. They generally allow for combinational circuits to have reconvergent paths but not to have loops. Confusingly, we have advocated for the design of combinational circuits *with* loops [8]. For simplicity, here we will use the term combinational to mean having a tree topology, so having *no* reconvergence and *no* loops.

with each digit weighted according to its position. The result must be "re-encoded" back in positional form.

Consider instead digital computation that is based on a *stochastic* representation of data. A number x ($0 \leq x \leq 1$) corresponds to a sequence of random bits. Each bit has *probability* x of being one and probability $(1 - x)$ of being zero. In the *serial* model, the bits stream on a single wire. In the *parallel* model, random values are assigned to wires in a bundle. When streaming, the computation is probabilistic in *time*, shown in in Figure 4(a); when in a bundle, the computation is probabilistic in *space*, shown in 4(b). All of our examples consist of serial computation. However, all the concepts apply equally to parallel computation.

There has been widespread interest in design digital circuits that compute based on this representation [2,3,4,7], called **stochastic circuits**. We point to [1], a survey of work in the area. The appeal of this approach stems from the fact that complex functions can be computed with very simple circuits. Consider the operation of multiplication: its implementation consists of a single AND gate. As illustrated in Figure 5, the inputs are two independent input stochastic bit streams a and b. The number represented by the output stochastic bit stream c is

$$\begin{aligned} c &= P(C = 1) = P(A = 1 \text{ and } B = 1) \\ &= P(A = 1)P(B = 1) \\ &= a \cdot b. \end{aligned} \tag{1}$$

The probability of getting a one at the output, $P(C = 1)$, is equal to the probability of simultaneously getting ones at the inputs, namely, $P(A = 1)$ times $P(B = 1)$. So the AND gate *multiplies* the two values represented by the stochastic bit streams, provided they are random and independent. (Multiplication here means computing a fraction of a fraction.)

Multiplication is the cannonical example, but research by computer engineers has shown that many functions of interest can be computed with stochastic circuits having remarkably few gates. For instance, Taylor series expansions of polynomial approximations to trigonometric functions can be computed with approximately a dozen gates [3].

1.3 Computing Polynomial with Stochastic Circuits

Consider basic logic gates. Table 2 describes the functions that they implement given stochastic inputs. These are all straight-forward to derive algebraically. For instance, given a stochastic input x representing the probability of seeing a 1 in a random stream of 1's and 0's, a NOT gate implements the function

$$\text{NOT}(x) = 1 - x. \tag{2}$$

Above, we say that given inputs x and y, an AND gate implements the function:

$$\text{AND}(x, y) = xy. \tag{3}$$

An OR gate implements the function:

$$\text{OR}(x, y) = x + y - xy. \tag{4}$$

An XOR gate implements the function:

$$\mathrm{XOR}(x, y) = x + y - 2xy. \tag{5}$$

Table 2: Stochastic function implemented by basic logic gates.

gate	inputs	function
NOT	x	$1 - x$
AND	x, y	xy
OR	x, y	$x + y - xy$
NAND	x, y	$1 - xy$
NOR	x, y	$1 - x - y + xy$
XOR	x, y	$x + y - 2xy$
XNOR	x, y	$1 - x - y + 2xy$

It is well known that any Boolean function can be expressed in terms of AND and NOT operations (or entirely in terms of NAND operations). Accordingly, the function of any circuit can be expressed as a nested sequence of multiplications and $(1 - x)$ type operations. It can easily be shown that this nested sequence results in a polynomial function.

We will make the argument based upon truth tables. Here we will consider only univariate functions, that is to say stochastic logic that receives multiple independent copies of a single variable t. (Technically, t is the Bernoulli coefficient of a random variable X_i, where $t = [\Pr(X_i = 1)]$.) Please see [5] for a generalization to multivariate polynomials.

Consider a combinational circuit computing a function $f(X_1, X_2, X_3)$ with the truth table shown Table 3. Now suppose that each variable has independent probability t of being 1:

$$[\Pr(X_1) = 1] = t, \tag{6}$$
$$[\Pr(X_2) = 1] = t, \tag{7}$$
$$[\Pr(X_3) = 1] = t. \tag{8}$$

The probability that the function evaluates to 1 is equal to the sum of the probabilities of occurrence of each row that evaluates to 1. The probability of each row, in turn, is obtained from the assignments to the variables, as shown in Table 4. Summing up the rows that evaluate to 1 in this example, we obtain

$$(1-t)^2 t + (1-t)t^2 + t(1-t)t + t^2(1-t) + t^3 \tag{9}$$
$$= (1-t)^2 t + 3(1-t)t^2 + t^3 \tag{10}$$
$$= t + t^2 - t^3. \tag{11}$$

Generalizing from this example, suppose that we are given any circuit with n inputs that each evaluates to 1 with independent probability t. We conclude that the probability that the output of the circuit evaluates to 1 is equal to the sum of terms of the form $t^i(1-t)^j$, where $0 \le i \le n$, $0 \le j \le n$, $i + j = n$, corresponding to rows of the truth table of the circuit that evaluate to 1. Expanding out this expression, we always obtain a polynomial in t. Since the stochastic inputs and

Table 3: Truth table for a combinational circuit.

X_1	X_2	X_3	$f(X_1, X_2, X_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 4: Probabilities of each row in Table 3, assuming each variable has independent probability t.

X_1	X_2	X_3	Probability of Row	$f(X_1, X_2, X_3)$
0	0	0	$(1-t)^3$	0
0	0	1	$(1-t)^2 t$	1
0	1	0	$(1-t)t(1-t)$	0
0	1	1	$(1-t)t^2$	1
1	0	0	$t(1-t)^2$	0
1	0	1	$t(1-t)t$	1
1	1	0	$t^2(1-t)$	1
1	1	1	t^3	1

outputs are probabilities, this polynomial function maps inputs from the unit interval $[0, 1]$ to outputs in the unit interval $[0, 1]$.

So, in a stochastic context, a circuit computes a polynomial function. The converse question is much more challenging: given a target polynomial function, can we synthesize stochastic logic to compute it? The answer is a qualified yes: we have proved that there exists a circuit that computes *any* polynomial function that maps the unit interval to the unit interval [6]. So the characterization of stochastic logic is complete. Our proof method is constructive: we describe a synthesis methodology for polynomial functions that is general and efficient in terms of the number of gates. Through polynomial approximations, it can also be used to synthesize non-polynomial functions. For further details, the reader is referred to [3,4].

The answer above is a qualified "yes" because the prior method permits circuits with reconvergence. The goal of this paper is to consider, from a theoretical and mathematical point of view, the space of functions that can be computed using only primitive operations (namely AND and NOT operations) with no reconvergence. Note that our definition limits the space of circuits to a specific subset. However, as shown below, this space is sufficient to represent most polynomial functions.

For what follows, we consider combinational circuits with random Boolean variables as inputs. When we say a "probability", we mean the probability of a random Boolean variable being one. A "circuit branch" refers to any section of a larger circuit that produces an output function that contributes to the formation of the final output function.

The rest of the paper is organized as follows. Section 2 will describe some basic concepts and facts. Section 3 will describe the main problems considered in this work and the main results. Sections 4 and 5 give proofs of the main results described in Section 3. To demonstrate applicability in stochastic circuit design, in Section 6, we provide an implementation procedure and example for our method.

Lastly, in Section 7, we compare our method to previous techniques and discuss the limitations of this framework.

2 Basic Concepts and Facts

This section first gives several definitions that will be used in the remainder of the paper and then presents the fundamental theorem of algebra.

Definition 1 *A **gate** is one of the two operations described below:*

$$\textit{AND gate: } \mathrm{AND}\left(f(t), g(t)\right) = f(t) \cdot g(t),$$
$$\textit{NOT gate: } \mathrm{NOT}\left(f(t)\right) = 1 - f(t).$$

Inputs to these functions may be any finite combination of aforementioned gates, input variable t, or constants $z_i \in [0, 1]$.

Definition 2 *A **circuit** is a collection of gates connected in the form of a tree. It has multiple inputs and a single output. Each input is assigned with an independent random Boolean variable with either a variable probability t or a constant probability $z_i \in [0, 1]$. The circuit produces an output random Boolean variable, whose probability is a real-coefficient polynomial $P(t)$ with respect to the variable t. We say that **the circuit implements the polynomial** $P(t)$.*

Theorem 1 ***Fundamental Theorem of Algebra**. Given a real-coefficient polynomial $P_n(t)$ of degree n, there exist a multi-set of roots $\{r_1, r_2, \ldots, r_n\} \in \mathbb{C}$ such that $P_n(r_i) = 0$, where the multi-set may contain multiple roots of the same value.*

The following corollary follows directly from the Fundamental Theorem of Algebra.

Corollary 1 *A given real-coefficient polynomial $P_n(t)$ of degree n can be equivalently expressed as:*

$$P_n(t) = A \cdot \prod_{i=1}^{n} (t - r_i),$$

where A is a scalar and $\{r_1, r_2, \ldots, r_n\}$ forms the multi-set of roots of $P_n(t)$.

In what follows, without loss of generality, we assume that the multi-set of roots of a real-coefficient polynomial $P_n(t)$ of degree n can be partitioned into a list of real roots $\{r_1, r_2, \ldots, r_m\}$ and a list of complex conjugate root pairs $\{(r_{m+1}, r_{m+2}), \ldots, (r_{n-1}, r_n)\}$. By Corollary 1, we can also represent $P_n(t)$ as follows:

$$P_n(t) = A^* \cdot \prod_{i=1}^{m} P_1^{\{i\}}(t) \cdot \prod_{i=(m+1)/2}^{(n-1)/2} P_2^{\{2i,2i+1\}}(t), \tag{12}$$

where $A^* \neq 0$ is a real scalar, $P_1^{\{i\}}(t) = A_i(t - r_i)$ $(i = 1, 2, \ldots, m)$ is a polynomial of degree 1 with the root as r_i, and $P_2^{\{2i,2i+1\}}(t) = A_{2i}(t - r_{2i})(t - r_{2i+1})$ $(i = (m+1)/2, (m+3)/2, \ldots, (n-1)/2)$ is a polynomial of degree 2 with the roots as the complex conjugate root pair r_{2i} and r_{2i+1}.

3 Problems and Main Results

The main research goal of this work is to find a method to synthesize a circuit to implement a given polynomial. Of course, not all polynomials can be implemented by a circuit. However, the following theorem shows a subset of polynomials that can be implemented by a circuit.

Theorem 2 *Given a polynomial $P_n(t)$ represented in the form shown in Eq.* (12)*, if*

1. *for each $i = 1, 2, \ldots, m$, there exists a circuit implementing the polynomial $P_1^{\{i\}}(t)$,*
2. *for each $i = (m+1)/2, (m+3)/2, \ldots, (n-1)/2$, there exists a circuit implementing the polynomial $P_2^{\{i,i+1\}}(t)$, and*
3. *$A^* \in [0, 1]$,*

then $P_n(t)$ can be implemented by a circuit.

Proof: Suppose that the above conditions are satisfied. For each $1 \leq i \leq m$, let the circuit implementing the polynomial $P_1^{\{i\}}(t)$ be C_i. For each $(m+1)/2 \leq i \leq (n-1)/2$, let the circuit implementing the polynomial $P_2^{\{i,i+1\}}(t)$ be D_i. Then, we can build a circuit C that consists of the circuits $C_1, \ldots, C_m$, $D_{(m+1)/2}, \ldots, D_{(n-1)/2}$, a primary input I^* of probability A^*, and a tree of AND gates that merges the outputs of the circuits $C_1, \ldots, C_m$, $D_{(m+1)/2}, \ldots, D_{(n-1)/2}$ and the primary input I^* into a single output. By the function of an AND gate, clearly, the probability of the output is

$$A^* \cdot \prod_{i=1}^{m} P_1^{\{i\}}(t) \cdot \prod_{i=(m+1)/2}^{(n-1)/2} P_2^{\{2i,2i+1\}}(t),$$

which equals $P_n(t)$ according to Eq. (12). Thus, by construction, a circuit implementing the polynomial $P_n(t)$ exists. □

The above theorem shows a subset of polynomials that can be implemented by a circuit, and its proof also shows a method to construct circuits for such polynomials. In this work, we try to identify the subset of polynomials that satisfies the conditions of Theorem 2. We note that for the polynomials $P_1^{\{i\}}(t) = A_i(t - r_i)$ and $P_2^{\{2i,2i+1\}}(t) = A_{2i}(t - r_{2i})(t - r_{2i+1})$ in the decomposition form of $P_n(t)$ shown in Eq. (12), we have freedom to choose their constant factors A_i and A_{2i}, and once A_i's and A_{2i}'s are chosen, we can correspondingly adjust the constant factor A^* in Eq. (12). Thus, the main problems become 1) identifying the set of real roots so that for each root r in the set, there exists a circuit implementing a polynomial of degree 1 with the root as r, and 2) identifying the set of complex conjugate root pairs so that for each pair of roots r_1 and r_2 in the set, there exists a circuit implementing a polynomial of degree 2 with the roots as r_1 and r_2. Alternatively, we can state the problems as follows.

Problem 1 *Given a real root r, decide if there exist a circuit C and a polynomial $P_1(t)$ of degree 1 with the root as r, such that the circuit C implements $P_1(t)$. Moreover, if they exist, give the circuit and the polynomial.*

Problem 2 *Given a complex conjugate root pair r_1 and r_2, decide if there exist a circuit C and a polynomial $P_2(t)$ of degree 2 with the roots as r_1 and r_2, such that the circuit C implements $P_2(t)$. Moreover, if they exist, give the circuit and the polynomial.*

The main results for the above two problems are presented as two theorems shown below.

Theorem 3 *For a real root r, there exists a circuit implementing a polynomial of degree 1 with the root as r if and only if $r \leq 0$ or $r \geq 1$.*

Theorem 4 *Let a region S of $\mathbb{R}^2$ be*

$$S = \left\{(a,b)|a > 0, a^2 + b^2 < 1\right\} \cup \left\{(a,b)|a < 1, (a-1)^2 + b^2 < 1\right\}. \quad (13)$$

For a pair of complex conjugate roots $r_1 = a + bi$ and $r_2 = a - bi$, there exists a circuit implementing a polynomial of degree 2 with the roots as r_1 and r_2 if and only if $(a,b) \in \mathbb{R}^2 \backslash S$.

In what follows, we will first give constructive proofs of the "if" parts of Theorems 3 and 4 in Section 4. Then, we will prove the "only if" parts of these theorems in Section 5.

4 Constructive Proofs of the "if" Parts of Theorems 3 and 4

This section presents the constructive proofs of the "if" parts of Theorems 3 and 4. In all subsequent figures, we use a dot to represent an inverter.

4.1 Constructive Proof of the "if" Part of Theorem 3

For the "if" part of Theorem 3, we have the following 2 cases corresponding to $r \geq 1$ and $r \leq 0$, respectively.
Case 1. For $r = a \geq 1$, the circuit shown in Fig 6A with constant $z = \frac{1}{a} \in [0,1]$ implements a polynomial of degree 1 with the root as r. The polynomial is:

$$P(t) = 1 - zt = 1 - \frac{1}{a}t = -\frac{1}{a}(t-a). \quad (14)$$

Case 2. For $r = a \leq 0$, the circuit shown in Fig 6B with constant $z = \frac{1}{1-a} \in [0,1]$ implements a polynomial of degree 1 with the root as r. The polynomial is:

$$P(t) = 1 - z\,(1-t) = 1 - \left(\frac{1}{1-a}\right)(1-t) = \frac{1}{1-a}(t-a). \quad (15)$$

4.2 Constructive Proof of the "if" Part of Theorem 4

For the "if" part of Theorem 4, we have the following 4 cases.
Case 3. For a pair of complex conjugate roots $r_{1,2} = a \pm bi$ such that

$$a \geq \frac{1}{2} \text{ and } (a-1)^2 + b^2 \geq 1,$$

the circuit shown in Fig. 7A with the constants $z_1 = \frac{2a}{a^2+b^2} \in [0,1]$ and $z_2 = \frac{1}{2a} \in [0,1]$ implements a polynomial of degree 2 with the roots as r_1 and r_2. The polynomial is:

$$\begin{aligned} P(t) &= 1 - (z_1 t\,(1 - z_2 t)) \\ &= 1 - \left(\frac{2a}{a^2+b^2} t \left(1 - \frac{1}{2a} t\right)\right) \\ &= \frac{1}{a^2+b^2}(t - (a+bi))(t - (a-bi)). \end{aligned} \tag{16}$$

Case 4. For a pair of complex conjugate roots $r_{1,2} = a \pm bi$ such that

$$a \leq \frac{1}{2} \text{ and } a^2 + b^2 \geq 1,$$

the circuit shown in Fig. 7B with the constants $z_1 = \frac{2(1-a)}{(1-a)^2+b^2} \in [0,1]$ and $z_2 = \frac{1}{2(1-a)} \in [0,1]$ implements a polynomial of degree 2 with the roots as r_1 and r_2. The polynomial is:

$$\begin{aligned} P(t) &= 1 - (z_1(1-t)\,(1 - z_2(1-t))) \\ &= 1 - \left(\frac{2(1-a)(1-t)}{(1-a)^2+b^2} \left(1 - \frac{1-t}{2(1-a)}\right)\right) \\ &= \frac{1}{(1-a)^2+b^2}(t-(a+bi)\,)(t-(a-bi)\,). \end{aligned} \tag{17}$$

Case 5. For a pair of complex conjugate roots $r_{1,2} = a \pm bi$ such that

$$a \geq 1,$$

the circuit shown in Fig. 8A with the constants $z_1 = \frac{2a-1}{a^2+b^2} \in [0,1]$ and $z_2 = \frac{1}{2a-1} \in [0,1]$ implements a polynomial of degree 2 with the roots as r_1 and r_2. The polynomial is:

$$\begin{aligned} P(t) &= 1 - z_1\,(1 - (1-t)\,(1 - z_2 t)) \\ &= 1 - \frac{2a-1}{a^2+b^2} \left(1 - (1-t)\left(1 - \frac{1}{2a-1} t\right)\right) \\ &= \frac{1}{a^2+b^2}(t-(a+bi)\,)(t-(a-bi)\,). \end{aligned} \tag{18}$$

Case 6. For a pair of complex conjugate $r_{1,2} = a \pm bi$ such that

$$a \leq 0,$$

the circuit shown in Fig. 8B with the constants $z_1 = \frac{1-2a}{(1-a)^2+b^2} \in [0,1]$ and $z_2 = \frac{1}{1-2a} \in [0,1]$ implements a polynomial of degree 2 with the roots as r_1 and r_2. The polynomial is:

$$\begin{aligned} P(t) &= 1 - z_1\,(1 - t\,(1 - z_2\,(1-t))) \\ &= 1 - \frac{1-2a}{(1-a)^2+b^2} \left(1 - t\left(1 - \frac{1}{1-2a}(1-t)\right)\right) \\ &= \frac{1}{(1-a)^2+b^2}(t-(a+bi)\,)(t-(a-bi)\,). \end{aligned} \tag{19}$$

Combining the above 4 cases, we conclude that for a pair of complex conjugate roots $r_{1,2} = a \pm bi$, if $(a, b) \in \mathbb{R}^2 \backslash S$, where S is given by Eq. (13) and illustrated in Fig. 9, then there exists a circuit implementing a polynomial of degree 2 with the roots as r_1 and r_2.

Note that Case 5 and Case 6 share complex plane coverage with Case 3 and Case 4, respectively. If a root in these regions is required, Case 3 and Case 4 are considered better options, as these circuits have a shorter circuit delay, or number of logic gates separating the earliest inputs with the output function.

5 Proofs of the "only if" Parts of Theorems 3 and 4

This section presents the proofs of the "only if" parts of Theorems 3 and 4.

5.1 Proof of the "only if" Part of Theorem 3

To prove the "only if" part of Theorem 3, we only need to prove the following claim: for a real root $r \in (0, 1)$, there does not exist a polynomial of degree 1 with the root as r such that it can be implemented by a circuit. In fact, for a real root $r \in (0, 1)$, a polynomial of degree 1 with the root as r is of the form $P_1(t) = A(t - r)$, where $A \neq 0$. Clearly, if $A > 0$, then we have $P_1(0) < 0$; if $A < 0$, then we have $P_1(1) < 0$. Thus, for any polynomial of degree 1 with the root as r, it cannot map the unit interval $[0, 1]$ into unit interval $[0, 1]$ and hence, cannot be implemented by a circuit. Thus, we prove the above claim and hence, the "only if" part of Theorem 3.

5.2 Proof of the "only if" Part of Theorem 4

To prove the "only if" part of Theorem 4, we first give the following lemma.

Lemma 1 *Given a circuit that implements a polynomial $P_2(t)$ with a pair of complex conjugate roots $r_{1,2} = a \pm bi$ with $b > 0$, then a circuit branch must exist in the form:*

$$\text{NOT}\left(\text{AND}\left(P_1'(t), P_1''(t)\right)\right) = 1 - P_1'(t) \cdot P_1''(t),$$

where $P_1'(t)$ and $P_1''(t)$ are two degree-1 polynomials that can be implemented by a circuit.

Proof: First, the formation of a 2^{nd}-degree polynomial may only be achieved through the usage of the gate operation $\text{AND}(P_1'(t), P_1''(t))$ since the NOT gate operation $\text{NOT}(P_n(t))$ does not modify the degree of the polynomial, and $\text{AND}(P_2'(t), P_0''(t))$ requires a pre-formed 2^{nd} degree polynomial $P_2'(t)$.

Additionally, since $\text{AND}(P_1'(t), P_1''(t))$ will retain the roots of both $P_1'(t)$ and $P_1''(t)$, therefore, both roots of the polynomial $\text{AND}(P_1'(t), P_1''(t))$ are real. Subsequent AND gate operations will raise the degree of the polynomial. As a result, a NOT gate operation must be used to move the roots of $\text{AND}(P_1'(t), P_1''(t))$ into the complex plane. Therefore, a circuit branch of $P_2(t)$ must appear as described. □

Given Lemma 1, if we want to realize a 2^{nd} degree polynomial with complex roots, we are limited to the following form:

$$P_2(t) = 1 - P_1'(t) \cdot P_1''(t), \tag{20}$$

where $P_1'(t)$ and $P_1''(t)$ represent two degree-1 polynomials that can be implemented by a circuit.

To prove the "only if" part of Theorem 4, we only need to prove the following claim: a polynomial $P_2(t)$ of the form shown in Eq. (20) cannot have a pair of complex conjugate roots $r_{1,2} = a \pm bi$ such that $(a, b) \in S$.

We prove the above claim by contraposition. Suppose that a polynomial $P_2(t)$ of the form shown in Eq. (20) has a pair of complex conjugate roots $a \pm bi$ such that $(a, b) \in S$. Then, we further have the real component $a \in (0, 1)$. Suppose the polynomial is

$$P_2(t) = c_2 t^2 + c_1 t + c_0.$$

The ratio c_1/c_2 must be equal to $-2a$ (by completion of squares). Therefore, we have $-2 < c_1/c_2 < 0$. Since a degree-1 polynomial that can be implemented by a circuit is either the output polynomial of Case 1 multiplying a coefficient or the output polynomial of Case 2 multiplying a coefficient, we are limited to three cases for $P_1'(t)$ and $P_1''(t)$: (A) both in a form as the output polynomial of Case 1 multiplying a coefficient, (B) both in a form as the output polynomial of Case 2 multiplying a coefficient, or (C) one in a form as the output polynomial of Case 1 multiplying a coefficient and the other in a form as the output polynomial of Case 2 multiplying a coefficient. Next, we discuss these cases in turn.

A) The general equation for this case can be described as:

$$\begin{aligned} P_2(t) &= 1 - (A_1 (1 - z_1 t)) (A_2 (1 - z_2 t)) \\ &= 1 - A (1 - z_1 t) (1 - z_2 t) \\ &= -A z_1 z_2 \left(t^2 - \left(\frac{z_1 + z_2}{z_1 z_2} \right) t + \frac{1}{z_1 z_2} - \frac{1}{A z_1 z_2} \right). \end{aligned}$$

In this case, $c_1/c_2 \leq -2$ for any $z_1, z_2 \in [0, 1]$, which fails the requirement on c_1/c_2. Thus, for Case (A), we cannot find a polynomial $P_2(t)$ with a pair of complex conjugate roots $a \pm bi$ such that $(a, b) \in S$.

B) The general equation for this case can be described as:

$$\begin{aligned} &P_2(t) = 1 - (A_1 (1 - z_1 (1 - t))) (A_2 (1 - z_2 (1 - t))) \\ &= 1 - A (1 - z_1 (1 - t)) (1 - z_2 (1 - t)) \\ &= -A z_1 z_2 \left(t^2 + \left(\frac{1 - z_1}{z_1} + \frac{1 - z_2}{z_2} \right) t + \frac{1 - z_1}{z_1} \frac{1 - z_2}{z_2} - \frac{1}{A z_1 z_2} \right). \end{aligned}$$

In this case, $c_1/c_2 \geq 0$ for any $z_1, z_2 \in [0, 1]$, which fails the requirement on c_1/c_2. Thus, for Case (B), we cannot find a polynomial $P_2(t)$ with a pair of complex conjugate roots $a \pm bi$ such that $(a, b) \in S$.

C) The general equation for this case can be described as:

$$\begin{aligned} P_2(t) &= 1 - (A_1 (1 - z_1 t)) (A_2 (1 - z_2 (1 - t))) \\ &= 1 - A (1 - z_1 t) (1 - z_2 (1 - t)) \\ &= A z_1 z_2 \left(t^2 + \left(\frac{z_1 - z_2}{z_1 z_2} - 1 \right) t + \frac{1}{A z_1 z_2} - \frac{1 - z_2}{z_1 z_2} \right). \end{aligned} \tag{21}$$

In this case, $c_1/c_2 \in (-\infty, \infty)$ for $z_1, z_2 \in [0,1]$, providing an opportunity to reach roots with real components $a \in (0,1)$.

Now that we have isolated Case (C) as a candidate for $P_2(t)$ polynomials, we further analyze the implications for the complex plane coverage. First, completion of squares will translate (21) to:

$$= Az_1z_2\left(\left(t + \left(\frac{z_1 - z_2}{2z_1z_2} - \frac{1}{2}\right)\right)^2 + \frac{1}{Az_1z_2} - \frac{1 - z_2}{z_1z_2} - \left(\frac{z_1 - z_2}{2z_1z_2} - \frac{1}{2}\right)^2\right).$$

Recall the definition of the constants z_1 and z_2 by Case 1 and Case 2. These can be expressed as their respective real roots $r_1 = \frac{1}{z_1}$ and $r_2 = \frac{z_2 - 1}{z_2}$. The existing polynomial can be expressed through these relations as:

$$= \frac{A}{r_1(1 - r_2)}\left(\left(t - \frac{1}{2}(r_1 + r_2)\right)^2 + \frac{r_1(1 - r_2)}{A} + r_1r_2 - \left(\frac{1}{2}(r_1 + r_2)\right)^2\right).$$

In this form, the pair of complex conjugate roots we are aiming to represent, $a \pm bi$, should satisfy that:

$$a = \frac{1}{2}(r_1 + r_2) \quad \text{and} \quad b^2 = \frac{r_1(1 - r_2)}{A} + r_1r_2 - \left(\frac{1}{2}(r_1 + r_2)\right)^2.$$

Since $A \in (0,1]$ and $r_1(1 - r_2) = \frac{1}{z_1} \cdot \frac{1}{z_2} > 0$, we can always select a smaller value of A that increases the value of b^2. Therefore, we are interested in finding the lower limit of b^2, and can therefore set A to its maximum value of 1.
Given that $a = \frac{1}{2}(r_1 + r_2)$ and $A = 1$, we have

$$b^2 = r_1(1 - r_2) + r_1r_2 - a^2 = r_1 - a^2.$$

Therefore, we have

$$a^2 + b^2 = r_1.$$

The lower limit of the complex plane coverage will occur when r_1 achieves its minimal value. Given that $z_1, z_2 \in [0,1]$, we require $r_1 = \frac{1}{z_1} \geq 1$ and $r_2 = \frac{z_2 - 1}{z_2} \leq 0$. Furthermore, as $a \in (0,1)$, we require that $0 < r_1 + r_2 = 2a < 2$. To derive the minimal value for r_1, we consider the two symmetric sides of region S:

Left side: **For a root $r = a \pm bi$ in the region with $0 < a \leq \frac{1}{2}$.**

In this case, the minimal achievable value of r_1 is 1. When $r_1 = 1$, $r_2 = 2a - r_1 = 2a - 1 = 0$, which satisfies the requirement on r_2. This arc on the complex plane is described by:

$$a^2 + b^2 = 1.$$

The region below the lower bound described here can be defined as:

$$S_1 = \left\{(x, y) \;\middle|\; 0 < x < \frac{1}{2},\;\; x^2 + y^2 < 1\right\}.$$

Right side: **For a root $r = a \pm bi$ in the region with $\frac{1}{2} < a < 1$.**

In this case, $r_2 = 0$ will produce the minimum values for r_1. Under this condition, the minimal achievable value of $r_1 = 2a - r_2 = 2a > 1$, which satisfies the requirement on r_1. This arc on the complex plane is described by:

$$a^2 + b^2 = 2a,$$

or

$$(a-1)^2 + b^2 = 1.$$

The region below the lower bound described here can be defined as:

$$S_2 = \left\{ (x,y) \;\middle|\; \frac{1}{2} < x < 1, \;\; (x-1)^2 + y^2 < 1 \right\}.$$

Combining these two cases, we arrive at the region of inaccessibility, S_3, in the complex plane:

$$\begin{aligned} S_3 &= S_1 \cup S_2 \\ &= \left\{ (x,y) \;\middle|\; x > 0, \;\; x^2 + y^2 < 1 \right\} \cup \left\{ (x,y) \;\middle|\; x < 1, \;\; (x-1)^2 + y^2 < 1 \right\}. \end{aligned}$$

Clearly, S_3 equals S shown in Eq. (13), which indicates S is also a region of inaccessibility. Thus, for Case (C), we cannot find a polynomial $P_2(t)$ with a pair of complex conjugate roots $a \pm bi$ such that $(a, b) \in S$.

6 Synthesis Procedure

In this section, we will demonstrate an algorithm to synthesize a circuit that realizes any polynomial with a known factorization and without roots inside the inaccessible region of the complex plane, S. We illustrate our method using the following polynomial:

$$P_5(t) = \frac{1}{3}\left(1 - t + t^2 - t^3 + t^4 - t^5\right),$$

which has the roots: $r_i = \{1, \frac{1}{2} \pm \frac{i\sqrt{3}}{2}, -\frac{1}{2} \pm \frac{i\sqrt{3}}{2}\}$.

Step 1: *Classify cases and constants.* For each real root or pair of complex conjugate roots, identify the case that matches its location on the complex plane, and find corresponding constants. For roots where multiple cases match the same root location (overlap between Case 3 & 5 and Case 4 & 6), choose the lower method number, as the circuit delay is lowered by two logic gates per root pair in these cases:

$$r_1 = 1: \qquad \text{Case } 1 \rightarrow z = 1;$$

$$r_{2,3} = \frac{1}{2} \pm \frac{i\sqrt{3}}{2}: \qquad \text{Case } 3 \rightarrow z_1 = 1, z_2 = 1;$$

$$r_{4,5} = -\frac{1}{2} \pm \frac{i\sqrt{3}}{2}: \qquad \text{Case } 4 \rightarrow z_1 = 1, z_2 = \frac{1}{3}.$$

Step 2: *Implement circuit.* Take each circuit branch found in the previous step and connect them using successive AND gates. For this polynomial, we obtain the circuit shown in **Fig. 10**.

Step 3: *Remove redundancies and add coefficient.* For any constants $z_i = 1$, remove the connected AND gate. As discussed earlier, the resulting polynomial $P^*(t)$ will contain all the roots of the target polynomial $P(t)$, but the target polynomial $P(t)$ may be a scaled version of $P^*(t)$. Suppose that

$$P(t) = A \cdot P^*(t).$$

If $A = 1$, we simply return the circuit for $P^*(t)$. If $A < 1$, the final circuit can be constructed by adding an additional AND gate to the output of the circuit for $P^*(t)$. The other input of the AND is set as the constant A. If $A > 1$, the original polynomial cannot be realized by our method. We will also return the circuit for $P^*(t)$, which calculates a scaled version of the desired polynomial. The final form of our circuit is shown in **Fig. 11**.

7 Conclusions

This paper has characterized a subset of polynomials that stochastic circuits can compute. The proofs are constructive, in the sense that they directly provide a method to synthesize circuits that compute polynomials. We note that polynomials are particularly useful functions in practice, arising in computer systems performing signal, image, and video processing. In prior work, we proposed a method for synthesizing stochastic circuits that compute general polynomials by decomposing them into Bernstein polynomials [6].[2] The method that we propose here is generally superior, in the sense that it produces circuits with fewer gates and with shorter delay.

To demonstrate this, we have computed logic gate costs and circuit delays for a set of basic functions. We compare the results of the methods in this paper, which we will call the **root method**, to previously published results, which used what we will call the **Bernstein method** [4], in Table 5. Circuits generated for each function using the root method are included in the Supplementary Information. Cases in which the Bernstein method outperforms the root method we have discussed are nearly nonexistent, yet there are several caveats:

- Firstly, the root method produces a scaled version of the desired polynomial, with a leading coefficient, A. If $A > 1$, it can be scaled down via an AND gate with the constant, $1/A$, to achieve an identical match for the function. If $A < 1$ however, there is no direct method to scale a function up in the same manner. Empirically defining a filter to identify these cases would circumvent the need for polynomial deconstruction once implemented. Notwithstanding, generating stochastic circuits with the root method is always preferable when $A \geq 1$.
- Secondly, there are certain functions which cannot be represented by the root method due to a root landing in the exclusion region, S. In such a case, the

[2] We acknowledge here that Ivo Rosenberg was a co-author on this prior work. His mathematical insights were indispensable in that work as well.

polynomial would have to be approximated to a new form which moves the root to an accessible region.

We note that the root method uses exclusively AND and NOT gates and produces circuits with no reconvergence. In some cases, this may reduce the flexibility compared to the Bernstein method, which uses a "scaled addition" operator [4] which involves reconvergent paths. For instance, different functions can be implemented with the Bernstein method simply by supplying different inputs to the scaled addition operator. So the same circuit can be reprogrammed after being built [3, 4].

The coefficients of the polynomials produced using the root method present a number of mathematical challenges that are unaddressed here. Although it is relatively straightforward to construct a stochastic circuit representing a polynomial and assess its leading coefficient on a case-by-case basis, it is nontrivial to generate a closed-form expression to calculate this value without implementing the full algorithm. This problem could be resolved through an empirical definition over which polynomials can be constructed with this method, without need for an operation to scale the leading coefficient up. Certain cases are relatively simple to identify, such as when all roots of a polynomials reside exclusively in regions corresponding to Cases 1 & 3, or exclusively in regions corresponding to Cases 2 & 4. Since each circuit branch produces a polynomial that has a maximum $P(t) = 1$ at either $t = 0$ or $t = 1$, products of these roots will continue to have the same maximum at the same value of t. If the polynomial produced has a leading coefficient less than one, that would indicate the target polynomial has a maximum $P(t) > 1$ and therefore would break the definition of a probabilistic function, that is, one that maps the unit interval to the unit interval. Therefore, these two cases demonstrate systems where we are guaranteed to be able to compute polynomials exactly matching the desired function. It is unknown for which other root combinations these stochastic circuits indeed exist. Nevertheless, continuing existing development of these methods may lead to the conditions required to compute unscaled polynomials.

Figures and Tables with Captions

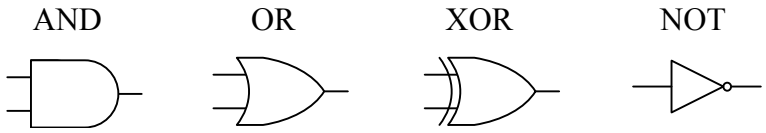


Fig. 1: Symbols for different types of gates.

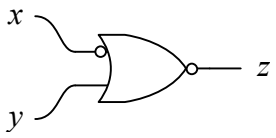
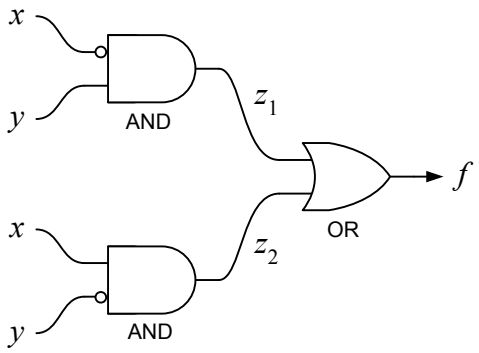


Fig. 2: Bubbles on the inputs or the output of a gate indicate negation. Here $z = \mathrm{NOT}(\mathrm{OR}(\mathrm{NOT}(x), y))$.



primary inputs: x, y
internal values: z_1, z_2
primary output: f

Fig. 3: An example of a digital circuit, consisting of gates and wires.

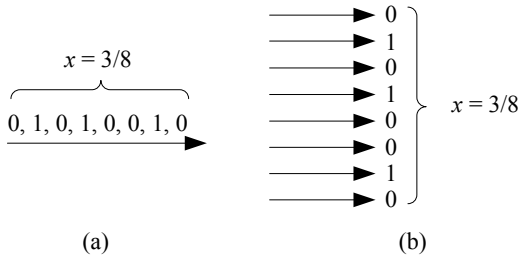


Fig. 4: Stochastic representation: (a) A random bit stream; (b) A bundle of wires with random values. A value x in $[0, 1]$ is represented as either a bit stream or a bundle. The probability that each bit in the stream or bundle is one is x; the probability that it is zero is $1 - x$.

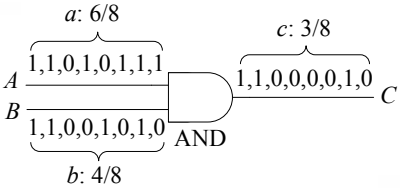


Fig. 5: Multiplication with a **stochastic representation**: an AND gate. The inputs are stochastic bit streams A and B and the output is a stochastic bit stream C. Here, the probability of seeing a 1 on A is $6/8$ and that of seeing a 1 on B is $4/8$. Accordingly, the probability of seeing a 1 on C is $6/8 \times 4/8 = 3/8$. This is illustrated with bit streams of length 8 here. In general, the computation is probabilistic; it only holds on *average* over a relatively long bit stream.

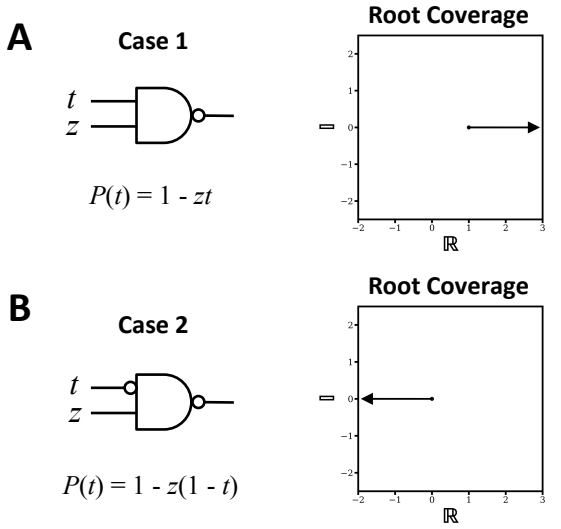


Fig. 6: *Circuit branch for cases 1 & 2.* (A) Circuit module for a polynomial representing a root $r = a \pm 0i$ where $a \geq 1$. (B) Circuit module for a polynomial representing a root $r = a \pm 0i$ where $a \leq 0$. Right of each circuit shows the roots covered by the given module, represented on complex plane in black.

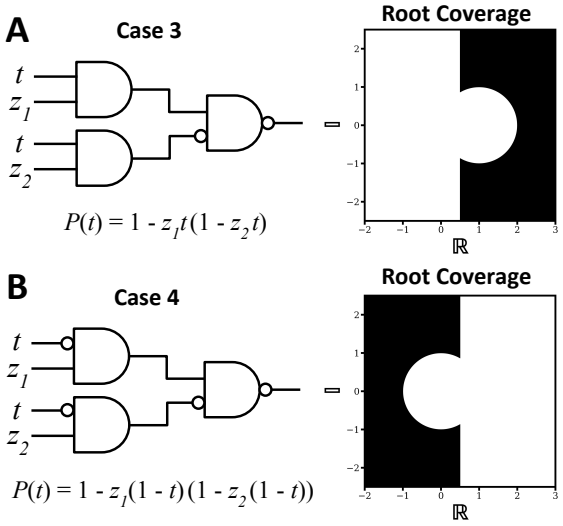


Fig. 7: *Circuit branch for cases 3 & 4.* (A) Circuit module for a polynomial representing a root $r = a \pm bi$ where $a \geq \frac{1}{2}, (a-1)^2 + b^2 \geq 1$. (B) Circuit module for a polynomial representing a root $r = a \pm bi$ where $a \leq \frac{1}{2}, a^2 + b^2 \geq 1$. Right of each circuit shows the roots covered by the given module, represented on complex plane in black.

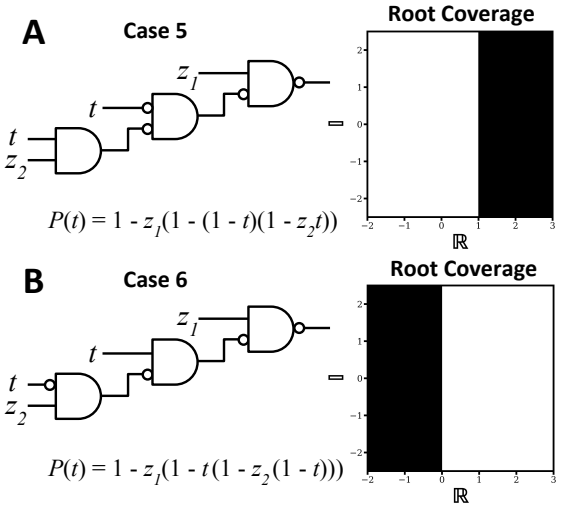


Fig. 8: *Circuit branch for cases 5 & 6.* (A) Circuit module for a polynomial representing a root $r = a \pm bi$ where $a \geq 1$. (B) Circuit module for a polynomial representing a root $r = a \pm bi$ where $a \leq 0$. Right of each circuit shows the roots covered by the given module, represented on complex plane in black.

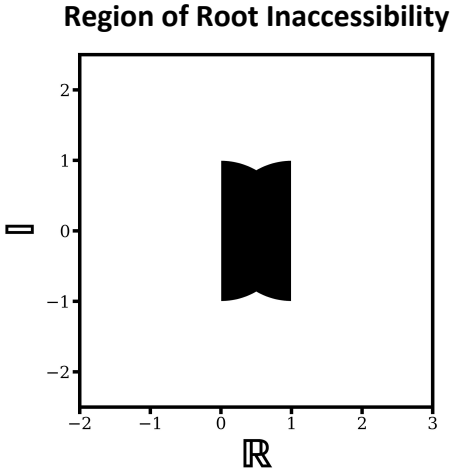


Fig. 9: *Inaccessible polynomial roots.* Region of complex plane that contains roots inaccessible to nonconvergent, stochastic circuits.

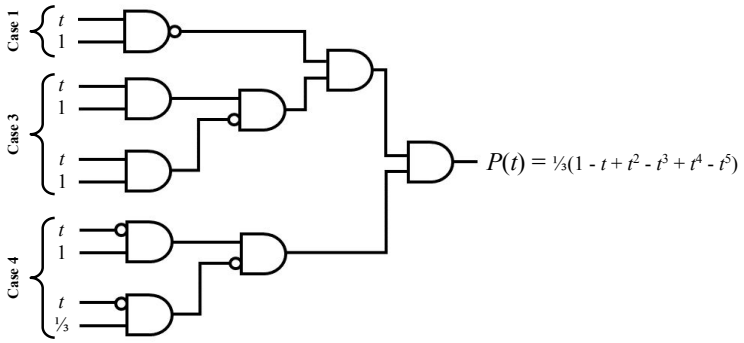


Fig. 10: *Polynomial implementation, before reduction.* Circuit implemented for polynomial $\frac{1}{3}(1-t+t^2-t^3+t^4-t^5)$ before redundancy removal.

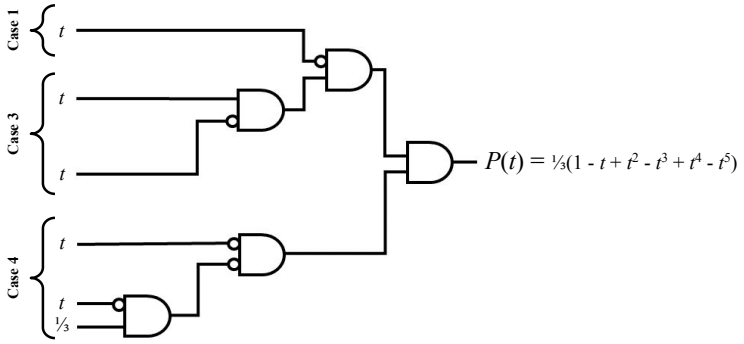


Fig. 11: *Final polynomial implementation.* Final circuit implemented for polynomial $\frac{1}{3}(1-t+t^2-t^3+t^4-t^5)$.

Table 5: Comparison of Bernstein method and root method for constructing MacLaurin polynomials for basic trigonometric and algebraic functions.

	Bernstein Method		**Root Method**		**Relative Change**	
$f(x)$	Gate Cost	Delay	Gate Cost	Delay	Gate Cost	Delay
$sin(x)$	49	20	14	6	-71%	-70%
$cos(x)$	58	20	20	6	-66%	-70%
$tan(x)$	49	20	14	6	-71%	-70%
$arcsin(x)$	49	20	14	6	-71%	-70%
$arctan(x)$	49	20	16	7	-67%	-65%
$sinh(x)$	49	20	14	6	-71%	-70%
$tanh(x)$	49	20	16	7	-67%	-65%
$cosh(x)$	58	20	21	7	-64%	-65%
$arcsinh(x)$	49	20	14	6	-71%	-70%
$exp(x)$	58	20	21	7	-64%	-65%
$ln(x+1)$	58	20	20	6	-66%	-70%
Average	**52.3**	**20.0**	**16.7**	**6.4**	**-68%**	**-68%**

Acknowledgements

We would like to thank Hidenori Shinohara for early assistance in developing the preliminary concepts backing the major theorems in this paper. Additionally, we would like to thank the anonymous reviewers for their insightful comments during the revision process.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Alaghi, A., Hayes, J.P.: Survey of stochastic computing. ACM Transactions on Embedded computing systems (TECS) **12**(2s), 92 (2013)
2. Gaines, B.R.: Stochastic computing systems. In: Advances in information systems science, pp. 37–172. Springer (1969)
3. Qian, W., Li, X., Riedel, M.D., Bazargan, K., Lilja, D.J.: An architecture for fault-tolerant computation with stochastic logic. IEEE Transactions on Computers **60**(1), 93–105 (2011)
4. Qian, W., Riedel, M.D.: The synthesis of robust polynomial arithmetic with stochastic logic. In: Proceedings of the 45th annual Design Automation Conference, pp. 648–653. ACM (2008)
5. Qian, W., Riedel, M.D.: The synthesis of stochastic logic to perform multivariate polynomial arithmetic. In: Proceedings of the International Workshop on Logic and Synthesis, pp. 79–86. IEEE (2008)
6. Qian, W., Riedel, M.D., Rosenberg, I.: Uniform approximation and bernstein polynomials with coefficients in the unit interval. European Journal of Combinatorics **32**(3), 448–463 (2011)
7. Qian, W., Riedel, M.D., Zhou, H., Bruck, J.: Transforming probabilities with combinational logic. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **30**(9), 1279–1292 (2011)
8. Riedel, M.D.: Cyclic combinational circuits. Ph.D. thesis, Caltech (2004)