Some Applications of Ternary Algebras*

J. A. Brzozowski
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
brzozo@uwaterloo.ca http://maveric.uwaterloo.ca/~brzozo

May 1997
To appear in *Publicationes Mathematicae (Debrecen)*

Abstract—A De Morgan algebra is a distributive lattice with 0 and 1 that has a quasi-complementation operation satisfying De Morgan's laws. A ternary algebra is a De Morgan algebra that has a third constant Φ that satisfies $(\mathbf{a}+\overline{\mathbf{a}})+\Phi = \mathbf{a}+\overline{\mathbf{a}}$, $(\mathbf{a}*\overline{\mathbf{a}})*\Phi = \mathbf{a}*\overline{\mathbf{a}}$, and $\Phi = \overline{\Phi}$. In this paper we first briefly describe three older applications of ternary algebras, namely, the detection of static hazards in combinational circuits, the computation of outcomes of transitions in sequential circuits, and modelling of CMOS circuits. Next, we discuss a new application of special ternary algebras, called process spaces, to the specification of interacting systems. We survey some old and new results concerning ternary algebras, and we describe a recent set-theoretic characterization of finite ternary algebras.

Index Terms— CMOS circuit, combinational circuit, interacting system, lattice, process space, sequential circuit, static hazard, Stone's theorem, subset-pair algebra, ternary algebra.

1 Introduction

The pioneering work of Shannon in 1938 [38] introduced Boolean algebra as an important tool for the analysis and design of digital circuits. Even though the technology has changed drastically several times, from relays to VLSI, Boolean algebra continues to play a major role today. In contrast to this, the use of ternary algebra has been much more limited. Nevertheless, there are several practical applications of ternary algebra to digital systems, some dating back to the late 1940s, some arising very recently. Along with

*This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. OGP0000871.

these applications there are also theoretical developments. In this paper, we describe these applications and theoretical advances.

The paper is structured as follows. In Section 2 we define ternary algebras and state their basic properties. In Section 3 we describe the application of ternary algebras to the detection of hazards in combinational circuits. Next, in Section 4 we outline Eichelberger's ternary simulation method [13], which is used in the analysis of asynchronous sequential circuits. In Section 5 we indicate how ternary algebras apply to CMOS transistor circuits. Process spaces, recently introduced by Negulescu [31] for modelling interacting systems, are described in Section 6. A generalization of process spaces, used to characterize arbitrary ternary algebras in set theoretic terms [6], is described in Section 7. Section 8 concludes the paper.

2 Ternary Algebras

The material in this section is based on [10, 25, 27, 30, 39]. A *lattice* is an algebraic system $L = \langle \mathcal{A}, +, * \rangle$, where $\mathcal{A}$ is a set and $+$ and $*$ are binary operations on $\mathcal{A}$ that satisfy the laws of Table 1, where $\mathbf{a}, \mathbf{b}$, and $\mathbf{c}$ are any three elements of $\mathcal{A}$. A lattice can also be viewed as a partially ordered set $L = \langle \mathcal{A}, \leq \rangle$, where $\mathbf{a} \leq \mathbf{b}$ if and only if $\mathbf{a}+\mathbf{b} = \mathbf{b}$. Equivalently, $\mathbf{a} \leq \mathbf{b}$ if and only if $\mathbf{a}*\mathbf{b} = \mathbf{a}$.

Table 1: Laws of lattices.

T1	$\mathbf{a}+\mathbf{a} = \mathbf{a}$	T1′	$\mathbf{a}*\mathbf{a} = \mathbf{a}$
T2	$\mathbf{a}+\mathbf{b} = \mathbf{b}+\mathbf{a}$	T2′	$\mathbf{a}*\mathbf{b} = \mathbf{b}*\mathbf{a}$
T3	$\mathbf{a}+(\mathbf{b}+\mathbf{c}) = (\mathbf{a}+\mathbf{b})+\mathbf{c}$	T3′	$\mathbf{a}*(\mathbf{b}*\mathbf{c}) = (\mathbf{a}*\mathbf{b})*\mathbf{c}$
T4	$\mathbf{a}+(\mathbf{a}*\mathbf{b}) = \mathbf{a}$	T4′	$\mathbf{a}*(\mathbf{a}+\mathbf{b}) = \mathbf{a}$

A lattice $L = \langle \mathcal{A}, \leq \rangle$ is *complete* if every subset of $\mathcal{A}$ has a least upper bound *LUB* and a greatest lower bound *GLB*. We will be concerned with finite lattices only, and all such lattices are complete. The universal lower and upper bounds in a complete lattice will be denoted by 0 and 1, respectively. A complete lattice satisfies the laws

T5	$\mathbf{a}+0 = \mathbf{a}$	T5′	$\mathbf{a}*1 = \mathbf{a}$
T6	$\mathbf{a}+1 = 1$	T6′	$\mathbf{a}*0 = 0$

A lattice is *distributive* if it satisfies the laws

$$\text{T7} \quad \mathbf{a}+(\mathbf{b}*\mathbf{c}) = (\mathbf{a}+\mathbf{b})*(\mathbf{a}+\mathbf{c}) \qquad \text{T7}' \quad \mathbf{a}*(\mathbf{b}+\mathbf{c}) = (\mathbf{a}*\mathbf{b})+(\mathbf{a}*\mathbf{c})$$

A complete distributive lattice is a *De Morgan algebra* [1] if it has a unary operation $\bar{\ }$ on $\mathcal{A}$ that satisfies the laws

$$\begin{array}{ll} \text{T8} & \overline{(\overline{\mathbf{a}})} = \mathbf{a} \\ \text{T9} & \overline{(\mathbf{a}+\mathbf{b})} = \overline{\mathbf{a}}*\overline{\mathbf{b}} \quad \text{T9}' \quad \overline{(\mathbf{a}*\mathbf{b})} = \overline{\mathbf{a}}+\overline{\mathbf{b}} \end{array}$$

The main topic of this paper are *ternary algebras*, which are De Morgan algebras with a third constant $\mathbf{\Phi}$, sometimes called *center* [25, 27], that satisfies

$$\begin{array}{ll} \text{T10} & (\mathbf{a}+\overline{\mathbf{a}})+\mathbf{\Phi} = \mathbf{a}+\overline{\mathbf{a}} \quad \text{T10}' \quad (\mathbf{a}*\overline{\mathbf{a}})*\mathbf{\Phi} = \mathbf{a}*\overline{\mathbf{a}} \\ \text{T11} & \mathbf{\Phi} = \overline{\mathbf{\Phi}} \end{array}$$

The laws above are not independent, but chosen for convenience, like similar laws of Boolean algebra. Also, there are equivalent sets of laws. For example, it can be shown that axioms T10 and T10$'$ can be replaced by

$$\text{T}'10 \quad (\mathbf{a}+\overline{\mathbf{a}})+(\mathbf{b}*\overline{\mathbf{b}}) = \mathbf{a}+\overline{\mathbf{a}} \qquad \text{T}'10' \quad (\mathbf{a}*\overline{\mathbf{a}})*(\mathbf{b}+\overline{\mathbf{b}}) = \mathbf{a}*\overline{\mathbf{a}}$$

The smallest ternary algebra is $T_0 = \langle \{0, \Phi, 1\}, +, *, \bar{\ }, 0, \Phi, 1\rangle$, where the set has only three elements, and $+$, $*$, and $\bar{\ }$ are the ternary OR, AND, and NOT operations defined in Table 2. Every finite ternary algebra has an odd number of elements, since the only element that is its own complement is $\mathbf{\Phi}$. Also, for every odd integer $n \geq 3$ there is at least one ternary algebra with n elements.

Table 2: The operations $+$, $*$, and $\bar{\ }$ in $\mathbf{T}_0$.

$\mathbf{a}_1+\mathbf{a}_2$		$\mathbf{a}_2 = 0$	Φ	1
	0	0	Φ	1
$\mathbf{a}_1$	Φ	Φ	Φ	1
	1	1	1	1

$\mathbf{a}_1*\mathbf{a}_2$		$\mathbf{a}_2 = 0$	Φ	1
	0	0	0	0
$\mathbf{a}_1$	Φ	0	Φ	Φ
	1	0	Φ	1

$\mathbf{a}$	$\overline{\mathbf{a}}$
0	1
Φ	Φ
1	0

A *ternary function* of $n \geq 0$ variables is any function $\mathbf{f}$ from $\{0, \Phi, 1\}^n$ to $\{0, \Phi, 1\}$.

We define addition of ternary functions as follows. For each n-tuple $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \{0, \Phi, 1\}^n$,

$$(\mathbf{f}+\mathbf{g})(\mathbf{a}) = \mathbf{f}(\mathbf{a})+\mathbf{g}(\mathbf{a}),$$

where, on the right-hand side, the addition is that of $\mathbf{T}_0$. Multiplication $(\mathbf{f}{*}\mathbf{g})$ and complementation $(\overline{\mathbf{f}})$ of ternary functions are defined similarly. The *constant* functions $|0|$, $|\Phi|$ and $|1|$ are defined as follows: For all $\mathbf{a} \in \{0, \Phi, 1\}^n$, $|0|(\mathbf{a}) = 0$, $|\Phi|(\mathbf{a}) = \Phi$, and $|1|(\mathbf{a}) = 1$. The following result is straightforward to verify.

Theorem 1 *Let $\mathbf{T}_n$ be the set of all ternary functions of n variables. Then $\mathbf{T}_n$ is a ternary algebra under the operations defined above, with $|0|$, $|\Phi|$, and $|1|$ acting as 0, Φ, and 1, respectively.*

The set $\mathcal{T}_n$ of ternary expressions over n variables $\mathbf{x}_1, \dots, \mathbf{x}_n$ is defined inductively as follows:

1. 0, Φ, 1, $\mathbf{x}_1, \dots, \mathbf{x}_n$ are ternary expressions.
2. If E and F are ternary expressions, then so are $(E{+}F)$, $(E{*}F)$, and $\overline{E}$.
3. Every ternary expression can be obtained by a finite number of applications of Rules 1 and 2.

The mapping $|\ |\colon \mathcal{T}_n \to \mathbf{T}_n$ associates a ternary function with each ternary expression. The expressions 0, Φ, 1, $\mathbf{x}_1, \dots, \mathbf{x}_n$ are mapped to the functions $|0|$, $|\Phi|$, $|1|$, $|\mathbf{x}_1|, \dots, |\mathbf{x}_n|$, respectively, where $|\mathbf{x}_i|$ is defined by $|\mathbf{x}_i|(\mathbf{a}_1, \dots, \mathbf{a}_n) = \mathbf{a}_i$ for all $(\mathbf{a}_1, \dots, \mathbf{a}_n) \in \{0, \Phi, 1\}^n$. Furthermore, we define $|(E{+}F)| = (|E|{+}|F|)$, $|(E{*}F)| = (|E|{*}|F|)$, and $|\overline{E}| = \overline{|E|}$. We shall show that not all ternary functions have corresponding ternary expressions.

The *uncertainty partial order* [10, 11, 21] $\sqsubseteq$ on the set $\{0, \Phi, 1\}$ is defined as follows:

$$0 \sqsubseteq 0,\ 1 \sqsubseteq 1,\ \Phi \sqsubseteq \Phi,\ 0 \sqsubseteq \Phi, \text{ and } 1 \sqsubseteq \Phi,$$

and no other pairs are related by $\sqsubseteq$. The value Φ is considered *uncertain*, whereas 0 and 1 are *certain*. In the partially ordered set $\langle \{0, \Phi, 1\}, \sqsubseteq \rangle$, we define the concept of least upper bound as usual; we denote this least upper bound by *lub* (not to be confused with *LUB*). We have $lub\{0\} = 0$, $lub\{1\} = 1$, and *lub* of every other nonempty subset of $\{0, \Phi, 1\}$ is Φ. We write $\mathbf{a} \sqsubset \mathbf{b}$ if $\mathbf{a} \sqsubseteq \mathbf{b}$ and $\mathbf{a} \neq \mathbf{b}$.

The partial order $\sqsubseteq$ is extended to $\{0, \Phi, 1\}^n$ in the natural way:

$$\mathbf{a} \sqsubseteq \mathbf{b} \text{ if and only if } \mathbf{a}_i \sqsubseteq \mathbf{b}_i \text{ for all } i,\ 1 \leq i \leq n,$$

where $\mathbf{a} = (\mathbf{a}_1, \dots, \mathbf{a}_n)$ and $\mathbf{b} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ are in $\{0, \Phi, 1\}^n$. We simplify the notation for n-tuples by omitting parentheses and commas. For example,

$0\Phi10 \sqsubseteq 0\Phi1\Phi$, but $0\Phi1$ and $1\Phi1$ are not related by $\sqsubseteq$. The definition of *lub* is also extended to $\{0, \Phi, 1\}^n$, by using the component-by-component least upper bound. For example, $lub\{\Phi0101, 11101, 01001\} = \Phi\Phi\Phi01$.

A ternary function $\mathbf{f}$ is *monotonic* if $\mathbf{a} \sqsubseteq \mathbf{b}$ implies $\mathbf{f}(\mathbf{a}) \sqsubseteq \mathbf{f}(\mathbf{b})$, for all $\mathbf{a}, \mathbf{b} \in \{0, \Phi, 1\}^n$. This is interpreted as follows: If argument $\mathbf{b}$ is at least as uncertain as argument $\mathbf{a}$, then the function value $\mathbf{f}(\mathbf{b})$ is at least as uncertain as $\mathbf{f}(\mathbf{a})$. The next theorem [25, 27] characterizes monotonic functions.

Theorem 2 *A ternary function* $\mathbf{f}$ *is monotonic if and only if there exists a ternary expression* F *such that* $|F| = \mathbf{f}$.

For any Boolean function $f\colon \{0, 1\}^n \to \{0, 1\}$, there are many ternary functions $\mathbf{f}$ that agree with f when the argument is binary. We call any such ternary function $\mathbf{f}$ a *ternary generalization* of f. Of particular interest are ternary generalizations called "extensions" [10, 11, 21]. For $f\colon \{0, 1\}^n \to \{0, 1\}$, the *ternary extension* $\mathbf{f}\colon \{0, \Phi, 1\}^n \to \{0, \Phi, 1\}$ is defined by

$$\mathbf{f}(\mathbf{a}) = lub\{f(a) \mid a \in \{0, 1\}^n \text{ and } a \sqsubseteq \mathbf{a}\},$$

for all $\mathbf{a} \in \{0, \Phi, 1\}^n$. One verifies that every ternary extension of a Boolean function is monotonic.

A *B-ternary expression* [21, 39] is defined like a ternary expression, except that the basis does not include Φ. A ternary function f is a *B-ternary function* if there exists a B-ternary expression F such that $f = |F|$.

3 Hazards in Combinational Circuits

The earliest application of three-valued logic to switching circuits is in the work of Goto [15, 16]. He used the values 0, 1, and 1/2, giving the interpretation "indefinite" to 1/2. He defined the following operations on $\{0, 1/2, 1\}$: $\mathbf{a}{+}\mathbf{b} = max\,(\mathbf{a}, \mathbf{b})$, $\mathbf{a}{*}\mathbf{b} = min\,(\mathbf{a}, \mathbf{b})$, and $\overline{\mathbf{a}} = 1 - \mathbf{a}$. He applied this algebra to the analysis and synthesis of relay networks, using 1/2 to represent a transient state between 0 and 1. By solving three-valued logic equations, he derived conditions required to prevent hazardous behavior in sequential relay contact networks. Other work using ternary algebra for switching circuits includes the papers by Moisil [19, 20], Roginskii [32], Muller [30], Yoeli and Rinon [39], and Mukaidono [22].

In this section we briefly illustrate the application of ternary algebra to the problem of detecting static hazards in combinational gate circuits.

We introduce the problem by means of an example. Consider the Boolean expression $F = x * y + \overline{x} * z$; it represents the gate circuit of Fig. 1 (a). When $x = 1$, $y = 1$, and $z = 1$, or $x = 0$, $y = 1$, and $z = 1$, the Boolean function $f(x, y, z)$ denoted by F is 1. Assume that initially the inputs are $xyz = 111$ and the output is $f = 1$; this is a stable state, meaning that the value of the function computed by the circuit is the same as the output value. The circuit will remain in that state unless one or more of the inputs change. Suppose now that input x changes to 0; since $f(011) = 1$, there should be no change in the output. If the inverter is slow, however, just after x changes from 1 to 0 we may have a situation in which $x = 0$ and $a = 0$; output b may then become 0, causing f to also become 0. This possibility of a gate circuit having a temporarily incorrect output is called a *static hazard*.

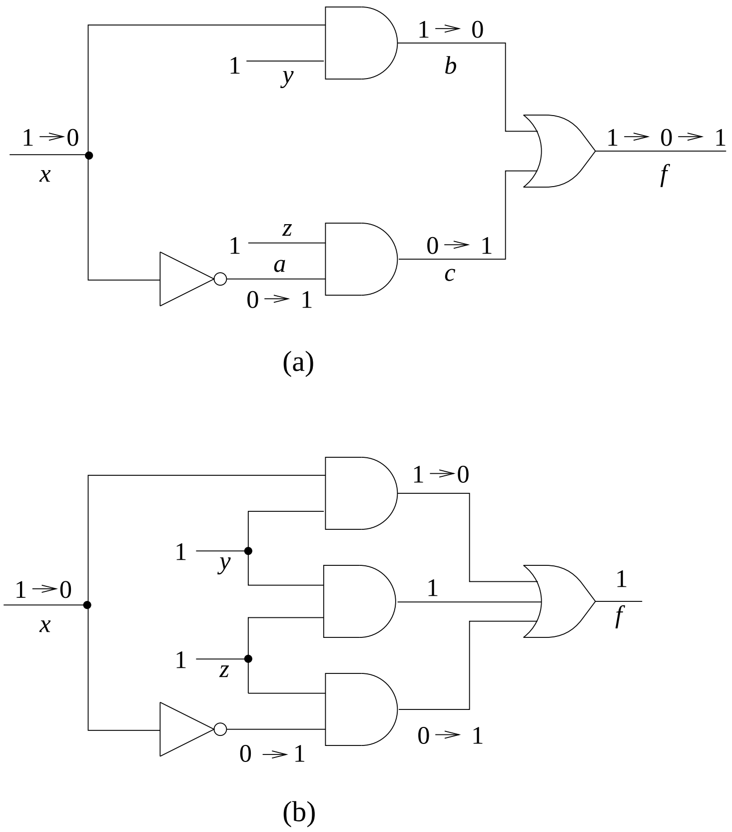


Figure 1. Hazard: (a) circuit with hazard; (b) circuit without hazard.

The hazard described above is not present in a circuit of Fig. 1 (b) corresponding to the expression $E = x * y + \overline{x} * z + y * z$. This circuit has an additional AND gate computing the product $y * z$. When the input x changes from 1 to 0, the product $y * z$ is 1, keeping the output at 1 as required.

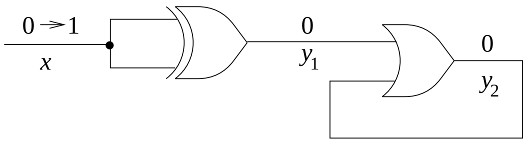
From the example it is clear that Boolean expressions are not powerful enough to describe static hazards. The Boolean function associated with the expression $x * y + \overline{x} * z$ is exactly the same as that associated with the expression $x * y + \overline{x} * z + y * z$. Suppose, however, that we interpret these expressions as B-ternary expressions, and consider their corresponding ternary functions. Then the hazard can be detected if we set $y = z = 1$ and $x = \Phi$, indicating that input x is changing, and therefore uncertain. The ternary function associated with $x * y + \overline{x} * z$ has the value Φ, detecting the hazard, while that associated with $x * y + \overline{x} * z + y * z$ has the value 1, showing that there is no hazard.

4 Ternary Simulation

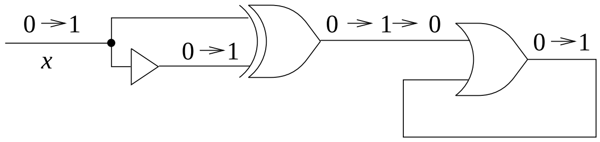
Figure 2 (a) shows a sequential gate circuit. The gate with output y_1 is an exclusive OR (XOR) gate, and its operation is denoted by $\oplus$. Suppose that we take into account only the gate delays in the circuit, and ignore wire delays. Then the state of the circuit is described by the variables y_1 and y_2 corresponding to the gate outputs. In general let y_i represents the present state of a gate variable, and Y_i, its "excitation," i.e., the value to which that variable is being driven. Because we assume that each gate has a delay, these two values are not necessarily the same.

In the binary model, the circuit behavior is governed by the Boolean functions of the two gates: $Y_1 = x \oplus x = 0$ and $Y_2 = y_1 + y_2$. Assume that the circuit starts in total state $x = y_1 = y_2 = 0$. Here, the excitation of each gate is equal to the gate output; hence the state is stable. If the input changes from 0 to 1, there is no change in the excitation of the XOR gate. Hence the binary model predicts that the state of the circuit will remain $y_1 = y_2 = 0$ after the change.

In Fig. 2 (b) we have added a delay in the bottom input wire of the XOR gate. Now, when x changes from 0 to 1, the top input to the XOR gate becomes 1, but the bottom input may remain 0 for some time, because of the wire delay. The XOR gate output may temporarily become 1, and this may cause the OR gate output also to become 1. Because of the feedback in the OR gate, y_2 may become permanently 1. Altogether, binary analysis predicts that two results are possible for the transition caused by changing x from 0 to 1: The final state y_1y_2 may be 00 if there are no appreciable wire delays, or 01 if there is a wire delay as shown.



(a)



(b)

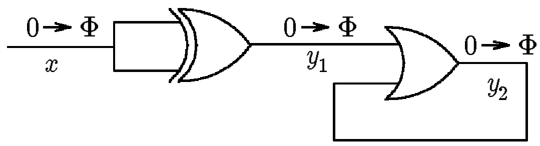
Figure 2. Binary analysis: (a) without wire delay; (b) with delay.

This example shows that a gate circuit may end up in an incorrect state because of wire delays. Since delays cannot be avoided in physical circuits, it is important to have efficient ways of detecting the possibility of incorrect outcomes. Binary analysis algorithms [10] are, unfortunately, exponential in the number of state variables. However, there is a ternary method called "ternary simulation" that is linear in the number of state variables. This method does not compute the set of all possible outcomes ($\{00, 01\}$ in our example), but it gives the *lub* of these outcomes. In our example, ternary simulation would give the result 0Φ, showing that the first variable is guaranteed to be 0, but the second is uncertain. For many applications, this information is sufficient.

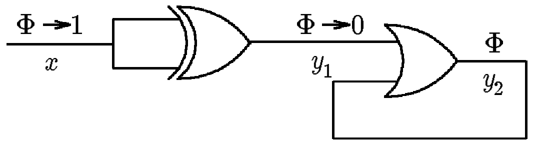
Ternary simulation [10, 13] consists of two algorithms, which will be called A and B. Each gate in the circuit is represented by the ternary extension of its Boolean function. The circuit is assumed to be started in a stable state. In ternary simulation, the transition of x from 0 to 1 is represented by two transitions: from 0 to Φ and from Φ to 1. In algorithm A, x is set to Φ, indicating that it is changing, and so its value is uncertain.

See Fig. 3 (a). The "uncertainty" may then spread to other parts of the circuit; in our example both gates become Φ. In Algorithm B, we start with the state reached in Algorithm A, and change the input to its final value, i.e., 1. See Fig. 3 (b). Now "certainty" may spread through the circuit; in our example y_1 becomes 0, but y_2 retains the value Φ, showing that its value may be either 0 or 1.

Ternary simulation was introduced by Eichelberger [13] in 1965 rather informally. The conjecture that ternary simulation gives the *lub* of the results of the so-called General-Multiple-Winner binary analysis was formally stated in 1979 [11]. This conjecture was finally proved in 1986 [7, 8, 34]. In 1994 this result was generalized in [37] so as to permit the analysis of a circuit that is started in an *unstable* state. The proofs of both the initial and the generalized theorem are challenging. The monotonicity of ternary extensions of Boolean functions plays an important role. The ternary simulation results have been used to prove that certain behaviors are not realizable with gates in a delay-insensitive manner, meaning that there is always some delay distribution that leads to incorrect behavior [5, 37]. For a comprehensive discussion of these issues see [10]. Related work appears in [17] and [36].



(a)



(b)

Figure 3. Ternary simulation: Algorithm A; Algorithm B.

5 CMOS Circuits

Many modern VLSI circuits use CMOS transistors as basic components. To a first approximation, a transistor can be viewed as a simple switch. The first formal switch-level model was developed by Bryant in 1984 [4]. Ternary algebra has been used for the analysis of CMOS circuits in [3, 9, 10, 12, 17, 35].

A CMOS inverter circuit is shown in Fig. 4. It consists of two transistors: a P-transistor (top) and an N-transistor (bottom). The terminal marked 1 is the power supply terminal connected to some fixed high voltage, say 5 volts. The terminal marked 0 is the ground terminal connected to 0 volts. Each of the two input terminals a and b of the inverter is connected to either the high voltage or ground. When the control input a, called the *gate* of the P-transistor, is at a high voltage—which we represent by logic 1—the P-transistor has an open circuit between the terminals marked 1 and c. When a is low (connected to ground)—which is represented by logic 0—the P-transistor may be thought of as a closed switch, capable of conducting current. Dually, the N-transistor is an open switch when $b = 0$, and a closed switch when $b = 1$. Terminal c is the output of the inverter.

Consider now all possible input combinations. When $a = 0$ and $b = 0$, the P-transistor is a closed switch and the N-transistor is an open switch. The output c is connected to the high voltage (1), and not connected to the low voltage (0); thus the output value is $c = 1$. When $a = 1$ and $b = 1$, the output is connected to 0 but not to 1; hence it has the value $c = 0$. When $a = 1$ and $b = 0$, the output c is connected to neither 0 nor 1, i.e., it is electrically isolated. In a physical circuit there is capacitance between the output terminal and ground, and this capacitance tends to retain the charge previously stored when the output was connected to 1. But this charge may also leak out with time; hence, it may not be safe to assume that the isolated output retains its previous value. Consequently, its value should be considered uncertain. Finally, when $a = 0$ and $b = 1$, the output is connected to both the high and the low voltage. Since the transistors are not ideal switches but have some resistance, the output will be at an intermediate value between the high and the low voltages. Thus its value is again uncertain.

It is natural to assign a third value, say Φ, to represent the last two possibilities; thus a ternary model is appropriate for CMOS circuits.

Various models of CMOS circuits are discussed in [10]. Different properties of circuits, such as transistor strengths, node sizes, charge sharing, etc., can be represented. In each case, the excitation of a node in the circuit is represented by an expression of the form $\mathbf{Y} = \mathbf{u} + \overline{\mathbf{z}}\Phi$, where $\mathbf{u}$ describes the conditions under which the node is declared to be 1 in the particular model used, and $\mathbf{z}$ denotes the conditions under which the node is declared to be 0. Otherwise, $\mathbf{Y}$ is declared to be Φ. To simplify the analysis of CMOS circuits, one can use a program [18] written by Mayo for manipulating ternary expressions.

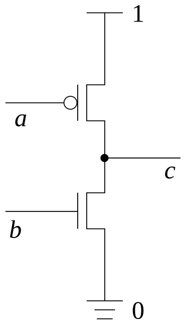


Figure 4. CMOS inverter.

6 Process Spaces

In 1995 Negulescu [31] introduced a very general model called "process space" for specifying the behavior of interacting systems. Surprisingly, process spaces turned out to be special ternary algebras. We now illustrate the process space model by a simple example.

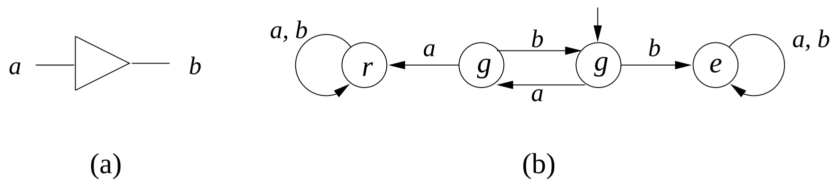


Figure 5. Buffer process.

Figure 5 (a) shows a symbol for a buffer, and Fig. 5 (b) shows a sequential machine describing its behavior. The buffer starts in the state marked with a short incoming arrow. If it receives a signal on its input a, it moves to a new state. (In a typical implementation, a transition from 0 to 1 or from 1 to 0 constitutes a signal a.) It is eventually expected to respond to this input signal by producing a signal on its output b and returning to the original state. Thus, one expects the normal operation of the buffer to consist of an alternating sequence of a's and b's. The two states involved in this normal operation are marked g, representing the fact that they are the *goal* states of the process.

It is possible that the buffer process malfunctions by producing an undesired output b when it is in the initial state, or by producing a sequence of two b's in response to an input a. This is considered an *error* of the process, and the sequential machine moves to the rightmost state labelled e. After this error, all other sequences of inputs and outputs are also considered to be erroneous. Similarly, it is possible that the *environment* of the buffer does not behave according to the specified goal states, and produces two consecutive a's when the process is in the initial state. (This is usually undesirable, because, in a circuit implementing the buffer, the two a's would represent a narrow pulse, which may be ignored if the circuit has inertia.) In this example, from the point of view of the buffer, this environment behavior can be rejected as illegal; hence the state diagram moves to a *reject* state marked r. Note that, if the behavior is not correct, the blame can be attached to the environment or to the process, but not to both. Consequently, there are only three types of states.

Words over the alphabet $\{a, b\}$ can be thought of as *executions* of the process in an environment. Executions in which the device operates correctly are called *accessible*. These are the words leading from the initial state to a goal or reject state. Words that lead to goal or error states are called *acceptable*.

An abstract process is formally defined as follows. Let E be an abstract set of executions (not necessarily finite words, as in our example). Let X and Y be two subsets of E, satisfying the condition $X \cup Y = E$. A *process* is an ordered pair $p = (X, Y)$, where X and Y are interpreted as the sets of accessible and acceptable executions of p, respectively. The set of all processes over E is called the *process space* $\mathcal{P}$ over E. If the cardinality of E is n, one verifies that the process space over E has 3^n elements.

Negulescu defined a partial order on $\mathcal{P}$ as follows. Let $p = (X, Y)$ and $p' = (X', Y')$ be two processes. Then p' *refines* p, written $p \leq p'$, iff $X \supseteq X'$ and $Y \subseteq Y'$. Thus p' refines p if it has more accessible executions and fewer acceptable executions. Consequently, p can be replaced by p' without any "bad effects." We illustrate this concept by the example of Fig. 6, which is the process of Fig. 5 (b) with the left goal state considered an error state. One can interpret the process of Fig. 6 as insisting that a b be produced after every a, whereas, in Fig. 5 (b), the process may remain indefinitely in the left goal state without producing an output. For the motivation for this definition see [31].

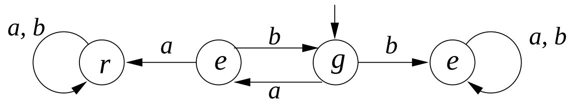


Figure 6. Illustrating refinement.

Given the refinement partial order on a process space $\mathcal{P}$, it is natural to define the following operations on $\mathcal{P}$:

$$(X, Y) + (X', Y') = (X \cap X', Y \cup Y')$$
$$(X, Y) * (X', Y') = (X \cup X', Y \cap Y')$$
$$\overline{(X, Y)} = (Y, X)$$

Also, introduce the following notation for three special processes: $1 = (\emptyset, E)$, $\Phi = (E, E)$, and $0 = (E, \emptyset)$. It is now straightforward to verify that the system $\langle \mathcal{P}, +, *, \bar{}, 0, \Phi, 1 \rangle$ is a ternary algebra.

7 A Characterization of Ternary Algebras

A process space over a set E consists of *all* pairs (X, Y) satisfying $X \cup Y = E$. It is natural to generalize this notion to the notion of *subset-pair algebra* [6], which is *any* set $\mathcal{S}$ of pairs (X, Y) with $X \cup Y = E$ that contains 0, Φ, and 1, and is closed under the process space operations $+$, $*$, and $\bar{}$. For example, the following set of seven pairs is a subset-pair algebra:

$$(\emptyset, \{1,2\}), (\{1\}, \{1,2\}), (\{2\}, \{1,2\}), (\{1,2\}, \{1,2\}), (\{1,2\}, \{1\}),$$
$$(\{1,2\}, \{2\}), (\{1,2\}, \{1,2\})$$

Clearly, every subset-pair algebra is a ternary algebra. Surprisingly, the converse result also holds for finite ternary algebras [6]:

Theorem 3 *Every finite ternary algebra is isomorphic to a subset-pair algebra.*

The proof is constructive. Given any finite ternary algebra T, we first view it as a distributive lattice with partial order $\leq$. We then represent it as a subdirect product [33] of two lattices, the lattice L_1 of all the elements of T that are of the form $x+\Phi$, where x is any element of T, and the lattice L_0 of all the elements of T that are of the form $x*\Phi$. Because T is a De Morgan algebra, it has a quasi-complementation operation $\overline{\ }$. The following is easy to verify. For any two elements a and b of T, $a \geq b$ iff $\overline{b} \geq \overline{a}$ and $a \geq \Phi$ iff $\Phi \geq \overline{a}$. It follows then that the lattices $(L_1, \geq)$ and $(L_0, \leq)$ are isomorphic. Let the length of the maximal chain from the largest element Φ of L_0 to its smallest element 0 be n. Then L_0 can be decomposed as a subdirect product of n two-element lattices. This permits us to associate a distinct subset of the set $\{1, 2, \ldots, n\}$ with each element of L_0. Next, we also label the elements of L_1 with subsets of $\{1, 2, \ldots, n\}$, by assigning to x in L_1 the same subset as is assigned to $\overline{x}$ in L_0. Finally, T is represented as a subdirect product of L_1 and L_0, from which one immediately obtains a subset-pair representation for T. For details see [6].

8 Concluding Remarks

The result of [6] has been recently generalized by Ésik to infinite ternary algebras [14]. The proof uses the well-known representation of distributive lattices by lattices of sets. It is also shown in [14] that each ternary algebra has a representation as an algebra of ternary functions over a set. This is a result analogous to the Caley representation of Boolean algebras [2].

We close by also mentioning some earlier related work by Mukaidono. Some results on B-ternary logic have been extended to fuzzy logic in [21, 26]. Properties of ternary logic functions, including functional completeness, were studied in [23, 28]. Ternary functions capable of detecting input failures were considered in [29]. Axiom systems for several algebraic systems related to ternary algebra were studied in [24].

References

[1] R. Balbes and P. Dwinger, *Distributive Lattices*, University of Missouri Press, 1974.

[2] S. L. Bloom, Z. Ésik, and E. G. Manes, "A Caley Theorem for Boolean Algebras," *Amer. Math. Monthly*, vol. 97, pp. 831–833, 1990.

[3] R. E. Bryant, "Race Detection in MOS Circuits by Ternary Simulation," *Proceedings of the IFIP TC10/WG10.5 International Conference on Very Large Scale Integration*, F. Anceau and E. J. Aas, eds., North-Holland Publishing Company, Amsterdam, The Netherlands, pp. 85–95, August 1983.

[4] R. E. Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems," *IEEE Trans. on Computers*, vol. C-33, no. 2, pp. 160–177, February 1984.

[5] J. A. Brzozowski and J. C. Ebergen, "On the Delay-Sensitivity of Gate Networks," *IEEE Trans. on Computers*, vol. 41, no. 11, pp. 1349–1360, November 1992.

[6] J. A. Brzozowski, J. J. Lou, and R. Negulescu, "A Characterization of Finite Ternary Algebras," *Int. J. Algebra and Computation*, to appear. (Also Research Report CS-96-26, Department of Computer Science, University of Waterloo, Waterloo, Ontario Canada, 9 pp., July 1996.)

[7] J. A. Brzozowski and C-J. Seger, "Correspondence between Ternary Simulation and Binary Race Analysis in Gate Networks," *Proceedings of the 13th International Colloquium on Automata, Languages and Programming*, L. Kott, ed., Springer-Verlag, Berlin, pp. 69–78, July 1986.

[8] J. A. Brzozowski and C-J. Seger, "A Characterization of Ternary Simulation of Gate Networks," *IEEE Trans. on Computers*, vol. C-36, no. 11, pp. 1318–1327, November 1987.

[9] J. A. Brzozowski and C-J. Seger, "A Unified Framework for Race Analysis of Asynchronous Networks," *J. ACM*, vol. 36, no. 1, pp. 20–45, January 1989.

[10] J. A. Brzozowski and C-J. Seger, *Asynchronous Circuits*, Springer-Verlag, New York, 1995.

[11] J. A. Brzozowski and M. Yoeli, "On a Ternary Model of Gate Networks," *IEEE Trans. on Computers*, vol. C-28, no. 3, pp. 178–183, March 1979.

[12] J. A. Brzozowski and M. Yoeli, "Combinational Static CMOS Networks," *Integration, The VLSI Journal*, vol. 5, pp. 103–122, 1987.

[13] E. B. Eichelberger, "Hazard Detection in Combinational and Sequential Switching Circuits," *IBM J. Research and Development*, vol. 9, pp. 90–99, March 1965.

[14] Z. Ésik, "A Caley Theorem for Ternary Algebras," *Int. J. Algebra and Computation,*, to appear.

[15] M. Goto, "Application of Three-Valued Logic to Construct the Theory of Relay Networks" (in Japanese), *Proceedings of the Joint Meeting of IEE, IECE, and I.of Illumination E. of Japan*, 1948.

[16] M. Goto, "Application of Logical Mathematics to the Theory of Relay Networks" (in Japanese), *Journal of the Institute of Electrical Engineers of Japan*, vol. 69, No. 729, 1949.

[17] T. Lengauer and S. Näher, "An Analysis of Ternary Simulation as a Tool for Race Detection in Digital MOS Circuits," *Integration, The VLSI Journal*, vol. 4, pp. 309–330, 1986.

[18] P. Mayo, "Decision Diagrams for Ternary Expressions," MMath Essay, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 47 pp., October 1993.

[19] Gr. C. Moisil, "Sur l'application des logiques à trois valeurs à l'étude des schémas à contacts et relais," *Actes proc. congr. itern. automatique*, p. 48, 1956.

[20] Gr. C. Moisil, "Aplicatiile logicii trivalente in studiul functionarii reale a schemelor cu contacte si relee," *Bul. Mat. al Soc. de St. Mat.-Fiz. din Rep. Pop. Romine I*, vol. 49, no. 2, pp. 147–191, 1957.

[21] M. Mukaidono, "On the B-Ternary Logical Function—A Ternary Logic Considering Ambiguity," *Trans. IECE, Japan,*, vol. 55–D, no. 6, pp. 355–362, June 1972. Available in English in *Systems, Computers, Controls*, vol. 3, no. 3, pp. 27–36, 1972.

[22] M. Mukaidono, "The B-Ternary Logic and Its Applications to the Detection of Hazards in Combinational Switching Circuits," *Proc. 8th International Symposium on Multiple-Valued Logic*, IEEE Computer Society Press, pp. 81–87, 1978.

[23] M. Mukaidono, “Some Kinds of Functional Completeness of Ternary Logic Functions,” *Proc. 10th International Symposium on Multiple-Valued Logic,* IEEE Computer Society Press, pp. 269–275, 1978.

[24] M. Mukaidono, “A Set of Independent and Complete Axioms for Fuzzy Algebra (Kleene Algebra),” *Proc. 11th International Symposium on Multiple-Valued Logic,* IEEE Computer Society Press, pp. 27–34, 1981.

[25] M. Mukaidono, “Regular Ternary Logic Functions—Ternary Logic Functions Suitable for Treating Ambiguity,” *Proc. 13th International Symposium on Multiple-Valued Logic,* IEEE Computer Society Press, pp. 286–291, 1983.

[26] M. Mukaidono, “Advanced Results on Application of Fuzzy Switching Functions to Hazard Detection,” *Advances in Fuzzy Sets, Possibility Theory and Applications,* P. P. Wong, ed., Plenum Publishing Corporation, pp. 335–349, 1983.

[27] M. Mukaidono, “Regular Ternary Logic Functions—Ternary Logic Functions Suitable for Treating Ambiguity,” *IEEE Trans. on Computers,* vol. C-35, no. 2, pp. 179–183, February 1986.

[28] M. Mukaidono, “Meaningful Special Classes of Ternary Logic Functions—Regular Ternary Logic Functions and Ternary Majority Functions,” *IEEE Trans. on Computers,* vol. 37, no. 7, pp. 799–806, July 1988.

[29] M. Mukaidono, “P-Functions—Ternary Logic Functions Capable of Correcting Input Failures and Suitable for Treating Ambiguities,” *IEEE Trans. on Computers,* vol. 41, no. 1, pp. 28–35, January 1992.

[30] D. E. Muller, “Treatment of Transition Signals in Electronic Switching Circuits by Algebraic Methods,” *IRE Trans. on Electronic Computers,* vol. EC-8, no. 3, p. 401, September, 1959.

[31] R. Negulescu, *Process Spaces,* Research Report CS–95–48, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 30 pp., December 1995.

[32] V. N. Roginskii, “The Operation of Relay Networks in Transitional Periods,” *Avtomatika i Telemekhanika,* vol. 20, no. 10, pp. 1408–1416, October 1959.

[33] D. E. Rutherford, *Introduction to Lattice Theory,* Hafner Publishing Company, 1965.

[34] C-J. H. Seger, *Ternary Simulation of Asynchronous Gate Networks,* MMath Thesis, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 95 pp., May 1986.

[35] C-J. H. Seger, *Models and Algorithms for Race Analysis in Asynchronous Circuits,* PhD Thesis, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 186 pp., May 1988.

[36] C-J. Seger and J. A. Brzozowski, "An Optimistic Ternary Simulation of Gate Races," *Theoretical Computer Science* vol. 61, pp. 49–66, 1988.

[37] C-J. Seger and J. A. Brzozowski, "Generalized Ternary Simulation of Sequential Circuits," *Theoretical Informatics and Applications* vol. 28, no. 3/4, pp. 159–186, 1994.

[38] C. E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits," *Trans. AIEE,* vol. 57, pp. 713–723, 1938.

[39] M. Yoeli and S. Rinon, "Application of Ternary Algebra to the Study of Static Hazards," *J. ACM,* vol. 11, no. 1, pp. 84–97, January 1964.