Three-valued digital systems

D. I. Porat, M.Sc., Ph.D.

Abstract

A survey is presented of the developments in algebras and techniques for the realisation of 3-valued switching functions. Digital arithmetic, ternary codes, composition algebras, minimisation, circuits and sequential circuit design are discussed. The feasibility of 3-valued digital systems is demonstrated.

1 Introduction

Digital equipment design is based on the binary-number system because of the availability, low cost and reliability of binary switching and storage elements. Higher radix systems are implemented by use of binary coding; however, at least one ternary computer[1] has been in operation which incorporates 3-valued elements.

A system which is based on a radix higher than 2 and built from multivalued elements offers the advantages of: (*a*) higher speed of serial and some serial-parallel arithmetical operations because of the smaller number of digits required for a given accuracy (it is assumed that ternary-logic elements can operate at a speed approaching that of the corresponding binary-logic elements) (*b*) better utilisation of transmission channels because of the higher information content carried by each line[2] (*c*) more efficient error detection and correction codes[3] (*d*) potentially higher density of information storage.

A set of n nodes of a k valued gate network can assume k^n states. Fewer terminals are thus required in a multivalued system to represent a given number of states, as compared with binary systems. Circuits for operators and compositions of k-valued algebras are more complex than those required to realise 2-valued switching functions. Integrated circuits are capable of performing complex functions economically, while the number of their terminals is rather limited. It appears reasonable to investigate properties of 3-valued systems as a first step towards exploration of problems involved in the design of multivalued systems.

This paper surveys developments in the ternary field and also includes material not published elsewhere before.[4]

2 Ternary digital arithmetic

Algorithms were established[4] for arithmetic operations in three representations:

(*a*) In signed magnitude

$$N = 3^n y_n + \sum_{i=-m}^{n-1} 3^i y_i \quad \ldots\ldots\ldots\ldots \quad (1)$$

where $y_n = 0$ for positive, and 2 for negative numbers. The ternary number N has $(m + n)$ digits, since i ranges from $-m$ to -1, and from 0 to $(n - 1)$. Also $0 \leqslant y_i \leqslant 2$.

(*b*) In signed complement of -3

$$N = 3^n 0 + \sum_{i=-m}^{n-1} 3^i y_i \qquad N \text{ positive} \quad \ldots \quad (2a)$$

$$N = 3^n 2 + \sum_{i=-m}^{n-1} 3^i \overline{(y_i)}_2 + 3^{-m} \qquad N \text{ negative} \quad \ldots \quad (2b)$$

where $\overline{(y_i)}_2$ symbolises the 2's complement of the ternary digit y_i.

(*c*) In signed complement -2

$$N = 3^n 0 + \sum_{i=-m}^{n-1} 3^i y_i \qquad N \text{ positive} \quad \ldots \quad (3a)$$

$$N = 3^n 2 + \sum_{i=-m}^{n-1} 3^i \overline{(y_i)}_2 \qquad N \text{ negative} \quad \ldots \quad (3b)$$

The complement of 2 of a ternary number,

$$\bar{N}_2 = 3^n - N - 3^{-m} \quad \ldots\ldots\ldots \quad (4)$$

Similarly, the complement of 2,

$$\bar{N}_3 = 3^n - N = \bar{N}_2 + 3^{-m} \quad \ldots\ldots\ldots \quad (5)$$

complementation of 2 is easily implemented through ternary inversion: $0 \leftrightarrow 2$, $1 \leftrightarrow 1$, while complementation of 3 is derived from the interrelation of $\bar{N}_3$ and $\bar{N}_2$, eqn. 5.

The rules for shifting (on multiplication or division) are analogous to the corresponding rules for binary or b.c.d. arithmetic.

The above conventions facilitated developing algorithms for arithmetic operations. Rules were established governing addition, subtraction, multiplication and division in the three representations of ternary numbers.[4] The availability of three choices at each step in the algorithm was sometimes found to simplify such algorithms.

3 Ternary codes

3.1 Ternary-coded decimal t.c.d.

A minimum of three ternary digits is required to code ten decimal digits. The 3^3 possible states of three variables in a 3-valued system result in $27!/(27 - 10)! \simeq 3 \cdot 1 \times 10^{12}$ t.c.d. codes. Not all of these codes are fundamentally different. Some exhibit useful properties for coded arithmetic, encoding techniques, error detection, minimum power consumption, etc.

The consistently weighted 9–3–1 code is shown in Table 1. A decimal digit X is given by the sum of the weights, each multiplied by its corresponding ternary coefficient x_i.

$$X = 3^2 x_3 + 3^1 x_2 + 3^0 x_1$$

The additive property of this code is useful for t.c.d. arithmetic, since the sum of the representation of two digits equals the representation of their sum, except when 'carries' are present.

Table 1

9–3–1-CODE

	9 x_3	3 x_2	1 x_1
0	0	0	0
1	0	0	1
2	0	0	2
3	0	1	0
4	0	1	1
5	0	1	2
6	0	2	0
7	0	2	1
8	0	2	2
9	1	0	0

A number of conditions have to be fulfilled for constructing t.c.d. codes, and designating the least-significant weight by w_1, the next w_2, and the most significant w_3:

(*a*) $\Sigma w_i \geqslant 9/2$

Paper 5838 E, first received 17th September 1968 and in revised form 2nd January 1969

Dr. Porat was formerly with the Department of Electrical Engineering & Electronics, University of Manchester Institute of Science & Technology, Manchester 1, England, and is now with Stanford University, PO Box 4349, Stanford, Calif., USA

(*b*) one weight must be 1, irrespective whether the weights are all positive or whether negative weights are included
(*c*) $(w_1 + w_2)2 \geqslant (w_3 - 1)$, to ensure that all successive numbers can be represented by the code
(*d*) to enable 10's complementation, $\Sigma w_i = 10/2$
(*e*) for 9 complementation, the sum must equal 9/2.

Table 2 shows all the possible positive integral weights for the ternary coding of decimal digits.

Table 2

T.C.D. CODES WITH POSITIVE INTEGRAL WEIGHTS

w_3	w_2	w_1		w_3	w_2	w_1
3	1	1	10s complementing	7	2	1
4	1	1		3	3	1
5	1	1		4	3	1
2	2	1	10s complementing	5	3	1
3	2	1		6	3	1
4	2	1		7	3	1
5	2	1		8	3	1
6	2	1	'natural'	9	3	1

To obtain codes which are capable of complementation in the diminished-radix representation, one has to introduce half-integer weights [see condition (*e*), above]. Codes with weights 2–2–$\frac{1}{2}$ and 2$\frac{1}{2}$–1–1 are 9 complementing.

Another way of looking at such codes is to consider the weight as being integral, but for a binary, rather than a ternary variable. For example, the weight of the l.s.d. in the 2–2–$\frac{1}{2}$ code is made '1', and the truth values in the respective column are thus 0 or 1. A coded decimal realisation employing mixed elements (binary and ternary) could be more economical than either b.c.d. or t.c.d. Consider the weights 3_t–2_b–1_b, where the subscripts indicate the radix used; *t* for ternary and *b* for binary. This and the 9_b–3_t–1_t code offer the advantage of decimal coding with three digits only and information storing, by use of ternary with the less expensive binary-memory elements.

As in all b.c.d. or t.c.d. codes, corrections have to be applied in arithmetic operations whenever a 'carry' or a 'borrow' occurs, or when the weights do not represent a geometric progression. When the digits representing the sum (difference) of a 9–3–1 coded t.c.d. are in the same decade as the digits of the augend and addend (minuend and subtrahend), no correction is required, since a 'carry' ('borrow') from one ternary digit to the next finds the conditions $3^i/3^{i-1} = 3$. For $X + Y = Z \geqslant 10_{10}$, $(122)_3 = (17)_{10}$ has to be added to the uncorrected sum in order to obtain the correct *Z*.

The sum of two ternary-coded decimals in the 9–3–1 code does not generate a 'carry', showing an advantage over all BCD codes.

The mixed radix 4_t–2_b–1_b code of Table 3 requires a simple correction algorithm. For $Z \leqslant (201)_3$, no correction is required. When $Z > (201)_3 = 9_{10}$, add 002 to the intermediate sum to obtain the correct result. This rule is simpler than many corresponding rules for b.c.d. codes.

Table 3

4_t–2_b–1_b CODED DECIMAL

	2_t^2 x_3	2_b^1 x_2	2_b^0 x_1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	2	0	0
9	2	0	1

A weighted 9–3–1 code[5] with positve and negative levels, as in Table 4, exhibits several advantages in that: (*a*) negative numbers have the same code as their positive counterparts, with signs reversed. Thus the usual operation of complementation is replaced by the simpler operation of sign change. Subtraction involves a ternary inversion of all digits followed by addition (*b*) results from arithmetic operations are obtained with proper sign and magnitude (*c*) the sign of a number is easily determined by sensing the sign of each digit, starting with the most significant, since $3^j > \sum_0^{j-1} 3^i$ (*d*) the ternary-coded decimal includes numbers −9–+9, utilising 19 out of 27 (70%) possible states. The corresponding percentage utilisations for the ternary 9–3–1 code employing positive coefficients only and the binary 8–4–2–1 code are 37 and 62·5%, respectively (*e*) the correction algorithm for decimal addition and subtraction is simple.

Truth tables for sum, carry and product modulo 3 are shown in Table 5.

Table 4

9–3–1 CODE WITH POSITIVE AND NEGATIVE LEVELS

	9	3	1
−9	−1	0	0
−8	−1	0	1
−7	−1	1	−1
−6	−1	1	0
−5	−1	1	1
−4	0	−1	−1
−3	0	−1	0
−2	0	−1	1
−1	0	0	−1
0	0	0	0
+1	0	0	1
+2	0	1	−1
+3	0	1	0
+4	0	1	1
+5	1	−1	−1
+6	1	−1	0
+7	1	−1	1
+8	1	0	−1
+9	1	0	0

Table 5

TRUTH TABLES

		x_1		
Sum mod 3		−1	0	+1
x_2	−1	+1	−1	0
	0	−1	0	+1
	+1	0	+1	−1

Carry mod 3	−1	0	+1
−1	−1	0	0
0	0	0	0
+1	0	0	+1

Product mod 3	−1	0	+1
−1	+1	0	−1
0	0	0	0
+1	−1	0	+1

3.2 Mixed-radix 'excess' codes

Codes analogous to the b.c.d. excess-3 code can be generated by use of mixed radixes. They exhibit some desirable properties, such as simple correction algorithms for arithmetic, inexpensive hardware realisation, and 9 complementation (ternary elements undergo 2 complementation and binary elements 1 complementation). One example is the excess-4 code derived from the 9_b–3_t–1_t t.c.d. code.

For $S < 10_{10}$ (no overflow) subtract $(011)_3$; for $S \geqslant 10_{10}$ (overflow) add $(011)_3$.

The excess-1 code is derived from the 4_t–2_b–1_b t.c.d. code and the correction requires subtraction or addition of 001.

3.3 Error checking

Parity checking by means of linear congruences[6] applies to the ternary case. For an 'even'-digit-parity check of a ternary number having *n* digits,

$$p = (x_0 + x_1 + \ldots x_{n-1}) \bmod 3$$

The 'odd'-digit-parity check can be defined as

$$p = (x_0 + x_1 + \ldots x_{n-1} + 2) \bmod 3$$

The number-parity check is defined as

$$N = p \bmod b \quad b \neq 3$$

Parity checks for arithmetic operations are based on the arithmetical invariance of these checks. In the 'diminished-base' numerical check of ternary numbers, $b = (r - 1) = 2$, where *r* is the radix employed in the arithmetic. Conventional binary circuits can thus be used for their implementation. The 'augmented-base' check $b = (r + 1)$ offers no advantages over the diminished-base check for ternary systems. However, it provides an easy check for arithmetic operations in binary representation, the diminished-base check being meaningless for binary systems since $N \bmod 1 = 0$. Ternary circuits are thus required to implement the augmented-base number check of binary systems.

Considering forbidden combination checks, it has been shown[4] that t.c.d. codes can be constructed having higher probabilities of single-error detection than the best b.c.d. codes available.

Variable-weight codes, unit-distance codes, ternary coding and decoding techniques are treated in Reference 4.

4 Algebras

4.1 Introduction

A logic system admitting k ($\geqslant 2$) truth values will be called a k-valued system. A function of m variables will be referred to as an m-place function or m-place composition. 1-place functions will also be referred to as unary operators or unary functions. 'Functional completeness' is the property of a set of compositions which enables one to synthetise any arbitrary switching function within a particular class. Berlin[8] has shown that if a set of compositions realises all 2-place, k-valued functions, then it also realises all m-place, k-valued functions. The criteria for the functional completeness of 2-place functions can be found in References 7 and 8.

The 'closure' property allows interconnection of operators which realise the operations of a particular algebra. That is, under closure, every output level can be used as a valid input level to another operator.

The great number of ternary compositions does not lead to an easy choice of a basic set for synthesis. Criteria can be established leading to a relatively efficient selection. The physical realisation of the composition should be simple and inexpensive, and the algebraic expressions should be amenable to minimisation. Since there are $3^{3^2} = 19683$ 2-place ternary compositions, it is useful to employ isomorphism for their classification.[9] The importance of isomorphism is in the preservation of the algebraic properties within an isomorphic class. A composition which is isomorphic to another is also isomorphic to all compositions within the class.

4.2 Composition algebras

Any set of functionally complete compositions can serve as a basis for a composition algebra. An expansion theorem is required, based on such a set, which relates a function of $(n + 1)$ variables to a function of n variables. This makes possible the synthesis of any arbitrary switching function in the particular class, using the available compositions.

There are several algebras available for the design of ternary switching functions. Two of these, the Post[10] and the 'modular'[11, 12] algebra have the advantages of similarity with ordinary algebra. One algebra[13] is based on a single composition; the ternary analogue of the Sheffer stroke function. There exist 3744 3-valued Sheffer functions, each functionally complete. The use of a single function for synthesis of networks appears attractive at first sight. However, the corresponding canonical forms are complex, and the expressions are difficult to manipulate. Nor does there exist a simple circuit realising any of these functions.

The 3-valued disjunction V, given by

$$f(x_1,x_2) = max\,(x_1,x_2) = x_1 V x_2,$$

and the cycling operator x', given by

$$f(x) = x' = (x + 1) \bmod 3$$

form a functionally complete set in the Post algebra. V is easily realised with diode circuits. A 3-transistor circuit[4] implements the operation x'.

A modular algebra has been shown to be functionally complete if the modulus is a prime number.[1] Its operators are

(*a*) sum modulo 3:

$$f(x_1,x_2) = (x_1 + x_2) \bmod 3$$

and (*b*) product modulo 3:

$$f(x_1,x_2) = (x_1 x_2) \bmod 3$$

The expansion theorem, using sum and product modulo 3, is[12]

$$f(x_1,x_2 \ldots x_n) = x_1 \odot [x_1 \odot \{2f(0,x_2 \ldots x_n) \oplus 2f(1,x_2 \ldots x_n) \oplus 2f(2,x_2 \ldots x_n)\} \oplus \{2f(1,x_2 \ldots x_n) \oplus f(2,x_2 \ldots x_n)\}] \oplus f(0,x_2 \ldots x_n)$$

Modular algebra should be efficient for digital arithmetic, provided that an inexpensive realisation of the necessary functions is found.

Lee and Chen[15] utilise the cycling operator, together with a Tgate, for the synthesis of 3-valued networks. A Tgate, performing the function of the 'conditioned disjunction', is a function of four variables in a 3-valued system:

$$\mathrm{T}(p,q,r;x) = \begin{cases} p & \text{if } x = 2 \\ q & x = 1 \\ r & x = 0 \end{cases}$$

Their expansion theorem is

$$f(p,q \ldots s) = \mathrm{T}\{f(2,q \ldots s), f(1,q \ldots s), f(0,q \ldots s);\ p\}$$

A 3-input ternary adder[15] requires ten Tgates, together with five cycling operators.

The Tgate and the set of sum and product modulo 3 are functionally complete only if ternary constants are assumed to be available. This poses no limitation in a practical circuit.

Shannon's theory of symmetric binary switching functions has been extended to ternary functions.[17] The detection of symmetric, ternary switching functions and their realisation with threshold logic is discussed by Merrill.[18]

Ternary threshold switching functions have been defined and studied by Hanson[19] and Merrill.[20] Hanson's synthesis procedure uses two basic ternary operations: (*a*) the n-variable threshold-like operation

$$x_1^t x_2^t \ldots x_n = \begin{cases} +1 & \text{if } \sum_{i=1}^{n} x_i \geqslant t \\ -1 & \text{if } \sum_{i=1}^{n} x_i \leqslant -t \\ 0 & \text{otherwise} \end{cases}$$

and (*b*) the ternary inversion $1 \leftrightarrow -1$ and $0 \leftrightarrow 0$. He proves that these two operations constitute a functionally complete set. For more efficient synthesis, he introduces a set of building-block operations, constructed from the two basic operations and ternary constants.

A design for a signed ternary adder based on his method requires two ternary parametrons, while a ternary comparator is realised with one parametron. The author does not show how to select building blocks which would yield minimum expressions or would simplify the synthesis procedure.

Merrill[20] defines ternary, threshold Sheffer functions, establishes a minimal decomposition theorem and presents synthesis procedures which yield efficient threshold-network realisations for modulo 3 addition of 3-valued variables, parallel addition and subtraction in the weighted ternary code, and modulo m addition in the weighted ternary code. A general solution is presented with a tight upper bound on the number of ternary threshold devices which is required for functions of each class.

4.3 Device-oriented algebras

The discussion has so far assumed the formulation of an algebra as a first step, followed next by its hardware implementation. The large number of ternary compositions suggests that these steps could be reversed. First, one could look for physical devices which realise 3-valued compositions, and a set of such compositions could then be tested for 'functional completeness'. Finally, an algebra could be developed around the available devices. Several such attempts are found in the literature.

Kooi and Weaver[21] describe a ternary-logic device based on the nonlinear magneto-conductivity effect in bismuth. A suitable expansion theorem is found in Reference 16.

Lowenschuss[12] shows how a pair of 2-collector transistors can be connected to yield a functionally complete set. Frequency-memory devices[22] and parametrons[23, 24] are also shown to implement multivalued logic. These and other devices listed in the literature[16] do not offer as many advantages as conventional semiconductor switching circuits.

A different approach is taken by Pugh,[25] who decomposes 3-valued functions into two Boolean functions. He then uses Boolean algebra, modified by necessary constraints, to synthetise 3-valued switching functions from two 2-valued

functions; one utilising positive and the other negative levels to represent the binary 'true'.

4.4 Synthesis with operations &, V, j_k and h_k

The 2-place compositions which are easiest to implement are the 3-valued conjunction &:

$$f(x_1, x_2) = \min(x_1, x_2) = x_1 \& x_2$$

and the 3-valued disjunction V:

$$f(x_1, x_2) = \max(x_1, x_2) = x_1 V x_2$$

For $k = 2$, these reduce to the Boolean AND and OR functions, respectively. The 3-valued composition & is commutative, associative, idempotent and has a 'zero' at truth value 0 and a 'unit' at truth value 2. Similar properties are exhibited by V, except for reversal of the location of 'zero' and 'unit'. Their properties make them amenable to manipulation and minimisation, and both are easily realised with simple semiconductor switching circuits.

It is not surprising, therefore, that most progress in physical realisation of 3-valued functions was achieved by use of these compositions.[4] & and V above are not functionally complete. Adding the set of j_k, or h_k, operators[14] ensures completeness. On the assumption that monotonic functions are easier to implement than nonmonotonic, one may select a set of unary operators containing some j_k and some h_k operators.[26]

Overall efficiency can be improved by use of redundant operators.[4] Nonminimal sets are, of course, used liberally in the design of Boolean functions.

The unary operators are defined[14] by

$$j_k(x)\begin{cases} = 0, & x \neq k \\ = 2, & x = k \end{cases}$$

$$h_k(x)\begin{cases} = 2, & x \neq k \\ = 0, & x = k \end{cases}$$

For brevity we shall use the notation

$$j_0(x) = x^0, \quad j_1(x) = x^1, \quad j_2(x) = x^2$$

$$h_0(x) = x^{12}, \quad h_1(x) = x^{02}, \quad h_2(x) = x^{01}$$

Because of their practical importance, we shall devote more attention to synthesis procedures using operations &, V, j_k and h_k. The rules given below result from the definitions of the operators and can best be proved by truth tables.

Conjunctions and disjunctions of variables with constants:

$x \& 0 = 0$	$xV0 = x$
$x \& 1 = x^{12} \& 1$	$xV1 = x^{12}V1$
$x \& 2 = x$ ('unit' at truth value 2)	$xV2 = 2$ ('zero' at truth value 2)

Variable x:

$x \& x = x$ (idempotent)	$xVx = x$ (idempotent)
$x \& x^0 = 0$	$xVx^0 = x^{02}V1$
$x \& x^1 = x^1 \& 1$	$xVx^1 = x^{12}$
$x \& x^2 = x^2$	$xVx^2 = x$

$$x = x^2 \& 2Vx^1 \& 1Vx^0 \& 0 = x^2Vx^1 \& 1$$

Relations between h_k and j_k operators:

$$j_r(x) \equiv x^r \qquad h_r(x) \equiv \overline{j_r(x)} \equiv \overline{x^r} \equiv x^{st}$$

where $r \neq s \neq t$, $\quad 0 \leqslant r, s, t \leqslant 2$

$$x^r \& x^s = x^{rs}\begin{cases} = x^r & r = s \\ = 0 & r \neq s \end{cases}$$

$$x^rVx^s = x^{rVs}\begin{cases} = x^r & r = s \\ = x^{rs} & r \neq s \end{cases}$$

Unary operation on operators:

$$j_0(x^0) = j_1(x)Vj_2(x) = h_0(x) = x^{12}$$
$$j_0(x^1) = j_0(x)Vj_2(x) = h_1(x) = x^{02}$$
$$j_0(x^2) = j_0(x)Vj_1(x) = h_2(x) = x^{01}$$
$$j_1(x^i) = 0, \; 0 \leqslant i \leqslant 2$$
$$j_2(x^i) = x^i, \; 0 \leqslant i \leqslant 2$$

A 'minterm' is a conjunction involving all the variables of a ternary switching function (t.s.f.). Similarly, a 'maxterm' is a disjunction involving all the variables of a t.s.f. Any term of an $n$variable function, expressed by $(n - 1i)$ variables, represents 3^i minterms or maxterms, where $0 \leqslant i \leqslant (n - 1)$.

A lower case m will be used to identify a minterm, and a capital M to indicate maxterm. A subscript is attached to each m as follows: write down each term in the same order of variables. The superscripts, read from left to right, form a ternary number. Use the decimal equivalent of the ternary number thus obtained as the identifying subscript of m. Subscripts for maxterms are obtained from the decimal equivalent of the ternary number, formed by the subscripts of the h_k operators used in experessing the relevant terms. This convention facilitates the inversion of functions, the change of canonical forms and other operations. Table 6 shows m_i, and M_i, the minterms and maxterms of a ternary function of two variables.

Note that subscripts for m_i and M_i are identical in every row. Thus, for inversion, one has the useful relation $m_i = \overline{M}_i$. Expansions in the disjunctive- and conjunctive-normal forms are

$$f(x_1 \ldots x_n) = [\Sigma\, m_i]\, V\, [\Sigma\, (1 \,\&\, m_j)] \quad \ldots \quad (6)$$
$$f(a_i) = 2 \qquad f(a_i) = 1$$

$$f(x_1 \ldots x_n) = [\Pi\, M_i] \,\&\, [\Pi\, (1 V M_j)] \quad \ldots \quad (7)$$
$$f(a_i) = 0 \qquad f(a_j) = 1$$

Table 6

MINTERMS m_i AND MAXTERMS M_i OF A 2-PLACE FUNCTION

x y	Minterms	i of m_i	Maxterms	i of M_i
0 0	x^0y^0	0	$x^{12}Vy^{12} = h_0(x)Vh_0(y)$	0
0 1	x^0y^1	1	$x^{12}Vy^{02} = h_0(x)Vh_1(y)$	1
0 2	x^0y^2	2	$x^{12}Vy^{01} = h_0(x)Vh_2(y)$	2
1 0	x^1y^0	3	$x^{02}Vy^{12} = h_1(x)Vh_0(y)$	3
1 1	x^1y^1	4	$x^{02}Vy^{02} = h_1(x)Vh_1(y)$	4
1 2	x^1y^2	5	$x^{02}Vy^{01} = h_1(x)Vh_2(y)$	5
2 0	x^2y^0	6	$x^{01}Vy^{12} = h_2(x)Vh_0(y)$	6
2 1	x^2y^1	7	$x^{01}Vh^{02} = h_2(x)Vh_1(y)$	7
2 2	x^2y^2	8	$x^{01}Vy^{01} = h_2(x)Vh_2(y)$	8

De Morgan's theorem can be easily extended to 3-valued functions. Thus the inverse of a function (as in eqn. 6) is obtained by changing all Σ to Π and all m_i to M_i. Alternatively, eqn. 6 can be written as a disjunction of terms for which the function is equal to 0 or 1. Corresponding rules apply for the inversion of an expression given in the other canonical form. Changing from one canonical form to the other is simple, recognising that eqns. 6 and 7 represent identical functions.

Design examples are found in Reference 4 for (*a*) a serial full adder using half adder and carry circuits (*b*) a 3-input serial adder (*c*) a ternary comparator, and a number of other circuits including code translation and mixed radixes (binary with ternary). These circuits are comparable in cost with binary circuits which perform similar functions.

5 Minimisation

The minimisation of a ternary switching function can be effected in three ways.

(*a*) the manipulation of algebraic expressions; as in Boolean algebra, simplification by this method requires experience
(*b*) the tabular method
(*c*) the map method.

5.1 Tabular method[26, 27]

The function is expressed as a disjunction of g and 1 & h terms,

$$f(x_1, x_2 \ldots x_n) = g(x_1, x_2 \ldots x_n) V 1 \,\&\, h(x_1; x_2 \ldots x_n)$$

where the g terms include minterms only for which $f(x_1, \ldots x_n) = 2$, while the h terms include minterms for which the function has the value 1. A systematic search is then conducted

to find 'essential' and other 'prime implicants' yielding the most economical configuration. In the binary method, one organises the systematic search for prime implicants by repeated use of a relation of the form

$$x \,\&\, (yV\bar{y}) = x$$

The ternary equivalent to this is

$$x^i \,\&\, (y^2Vy^1Vy^0) = x^i,\ 0 \leqslant i \leqslant 2$$

Also, the expression $(2 \,\&\, x^2V1 \,\&\, x^1V0 \,\&\, x^0)$ is identically equal to x.

All $h$type implicants can be considered 'don't care' conditions for $g$type implicants, since $x^iV1 \,\&\, x^i = x^i$.

The method is suitable for computer implementation. The set of unary operators chosen for functional completeness determines details of the procedure. Use of redundant compositions can result in considerable economies;[4] for example,

$$x_3^0x_2^0x_1^2Vx_3^1x_2^0x_1^2Vx_3^0x_2^1x_1^2Vx_3^1x_2^1x_1^2 = x_3^{01}x_2^{01}x_1^2$$

A tabular minimisation procedure for ternary switching functions, including threshold gating functions, is due to Merrill.[28]

5.2 Ternary maps[4, 26]

The map method is quick and easy but, unlike the exhaustive tabular method, it does not indicate a definite point of termination of the procedure. As a consequence, a map may not ensure that a minimum expression has been extracted. With a little practice, however, no difficulty should be experienced in obtaining minimised expressions for 4-variable functions. Simplified functions can be obtained by the map method in both canonical forms.

Table 7 shows ternary maps for two and three variables. In each square is entered the number of the minterm which represents it.

Table 7

TERNARY MAPS

$f(x, Y)$	y: 2	1	0
x: 2	8	7	6
1	5	4	3
0	2	1	0

a

$f(x,y,z)$	z: 2	1	0	z: 2	1	0	z: 2	1	0	
	26	25	24	17	16	15	8	7	6	2
	23	22	21	14	13	12	5	4	3	1
	20	19	18	11	10	9	2	1	0	0
	$x = 2$			$x = 1$			$x = 0$			

y

b

A function, not given in minterms, does not have to be expanded before plotting it on the map. The 'missing' variable or variables are entered under all three superscripts. The procedure of obtaining prime implicants from the map is based on ready recognition of adjacencies. Each set of operators used in the description of ternary switching functions has its own criteria for adjacencies. The outline below relates to a system using the five unary operators[4] j_0, j_1, j_2, h_1 and h_2. A minimum adjacency involves two minterms and occurs when two terms differ in one variable only. The superscripts of the latter must be 0 in one term and 1 or 2 in the other. This situation allows the use of an h_1 or $h_2$operator, e.g.

$$x^2y^1z^0Vx^2y^0z^0 = x^2y^{01}z^0$$

a saving of one minterm (see Table 7*b*). Adjacencies of three terms make the corresponding literal vanish. A 3-term adjacency in a 3 × 3 map involves the three squares in a column or in a row, e.g. $\Sigma_m(1, 4, 7)$ or $\Sigma_M(6, 7, 8)$. In a 3 × 9 map, the adjacencies are in rows, columns, or in squares occupying the same relative position as in a 3 × 3 submap, e.g. $\Sigma_m(3, 12, 21)$. To extend the argument to a 4-variable (9 × 9) map, the map is divided into three submaps, e.g. $f(2, x, y, z)$, $f(1, x, y, z)$ and $f(0, x, y, z)$. Adjacencies are then recognised in the same way as in a 3 × 9 map. In addition to that, each set of three squares occupying the same relative positions in the submaps forms a valid adjacency. For example, 13, 40 and 67 in a 4-variable map is a 3-term adjacency. 4-, 6-, 9- etc. adjacencies of $g$type terms can be defined in a similar manner.

All 2 in the map can be considered 'partial don't care' conditions for the purpose of mapping $h$type adjacencies. Thus, minterms Σ_m 13, 14, 16, 17 in Table 8 comprise an adjacency. We distinguish between a 'partial' and a 'total don't care' condition. The former occurs when a '2' is utilised as a 'don't care' condition for mapping $h$type adjacencies. The 'total don't care' condition, denoted by '—', can be utilised for the mapping of both types of adjacencies.

A special type of adjacency occurs when a row or a column has an entry 210, as shown in Table 8. An adjacency of this type makes the superscript of the variable vanish, i.e. the variable is used directly in the algebraic expression rather than using the result of a unary function operating on the variable. Three such terms in Table 8 are $x^0y^0zVx^0yz^0Vxy^1z^2$, where the first term represents (m_2, 1 & m_1), the second (m_6, 1 & m_3) and the last (m_{23}, 1 & m_{14}).

Table 8

ADJACENCIES

z: 2	1	0	z: 2	1	0	z: 2	1	0	
			1	2				2	2
2			1	1		0		1	1
						2	1	0	0
$x = 2$			$x = 1$			$x = 0$			

y

6 Devices and circuits

6.1 Combinational logic

A survey of physical (and chemical) phenomena which are potentially useful for ternary-logic design is given in Volume 1 of Reference 16, together with research results on selected physical phenomena. These include the nonlinear Hall effect,[21] and a threshold switching device utilising the saturable Hall effect and the constant-fluxoid principle.

A 2-aperture square-loop ferrite device is, according to the authors of Reference 29, capable of generating more than 100 ternary 2-place functions. A pair of 2-collector transistors can generate a set of functionally complete compositions[12] or can be utilised in the realisation of all three j_k operators.[23]

A critical evaluation of the devices mentioned above shows that they cannot compete with transistor switches in speed, simplicity, low power consumption, reliability or ease of operation. Semiconductors were used by Hallworth and Heath[30] in the design of the cycling operator, ternary invertor, $j_k$operators and a modulo3 half-sum circuit. Porat[4] showed the design of all the 27 unary operators of a 3-valued function using transistors in their switching mode only. His design showed high tolerance to component and supply-voltage variations. A simple four-transistor circuit realising five unary operators is shown in Fig. 1. [A single i.c. chip could easily contain six unary operators (j_0, j_1, j_2, h_0, h_1, and h_2) and could serve as a basic building block for the efficient realisation of 3-valued combinational networks.] The transfer function of the $h_1$operator is shown in Fig. 2. Circuits which implement other unary operators show similar sharp transfer characteristics. A ternary invertor and the cycling operator are shown in Figs. 3 and 4, respectively. The importance of these operators lies in the fact that they often serve as operations in the expansion of functions.[10,15,19] In addition, the ternary invertor is indispensable in forming 3 and 2 complements, which are necessary for the representation of negative ternary numbers. Ternary invertors have been realised by use of nonlinear loads.[4, 31]

Circuits which implement the Tgate[15] have also been demonstrated.[4, 32]

6.2 Memory elements

The development of reliable 3-state memory elements could considerably enhance storage density. Morris and Alexander[33] show a ternary-core switch which uses coincident currents, and a ternary-core store. A superconductive ternary

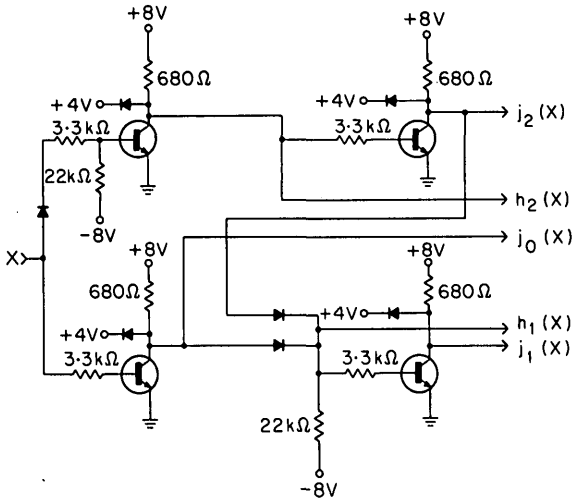


Fig. 1
Realisation of five unary operators

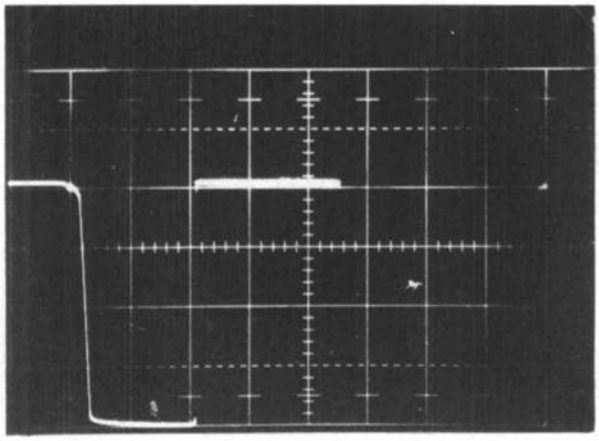


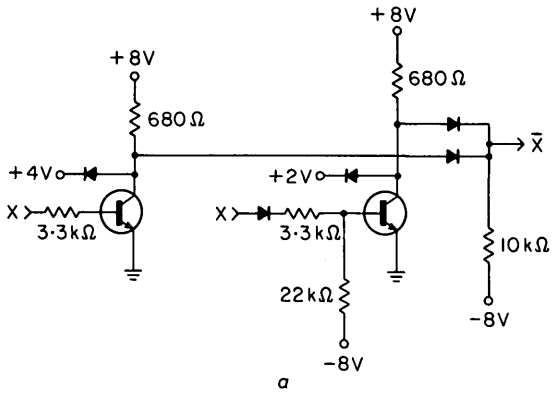
Fig. 2
h_1 operator
Horizontal scale: e_{in}, 1 major division = 1 V
Vertical scale: e_{out}, 1 major division = 1 V

information-storage device and random-access cryogenic-memory systems are discussed in Reference 16. Magnetic thin films have been proposed as ternary-memory cells:[4, 16] three stable states could be obtained by the superposition of two thin films differing in their coercive force.

A ternary computer built at Moscow State University uses two ferrites to achieve three stable states.[1] A ferrite core with a winding of the 'Fluxlok' type requires large currents in the orthogonal coil.[34]

The properties sought in active memory elements (tristable flip–flops) merit a brief discussion, since they constitute the basic building blocks for sequential-network design. The availability of several algebras points to a greater variety of tristable than binary flip–flops. For greater efficiency, more than one algebra may be employed in the realisation of a ternary system.[30]

The input circuits to tristable flip–flops should exhibit economical gating requirements in any arbitrary sequential network application. Since the applications are not specified, it appears reasonable to incorporate as many 'don't care' conditions as possible in the truth table of the tristable. In other words, if the level to one input is specified, the other input(s) should preferably accept any of the three levels without causing malfunction of the flip–flop. Such 'don't care' conditions, when mapped in a ternary map, offer opportunities for gate simplification through the availability of adjacencies. The 'don't care' condition discussed above will be denoted by a dash '—' in the map, signifying that any of the three levels can be applied to the particular input. In addition to these 3-state 'don't care' conditions, there exist 2-state 'don't care' conditions which are entered in a map or a table as (0, 1) or (1, 2). These may also contribute to gate simplification, because of a greater flexibility in forming adjacencies in the



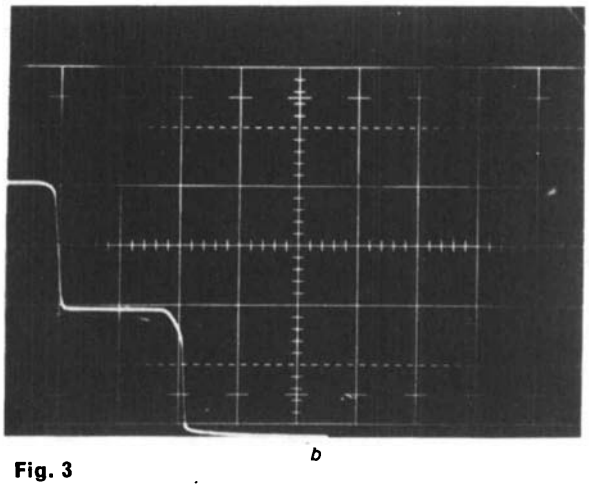
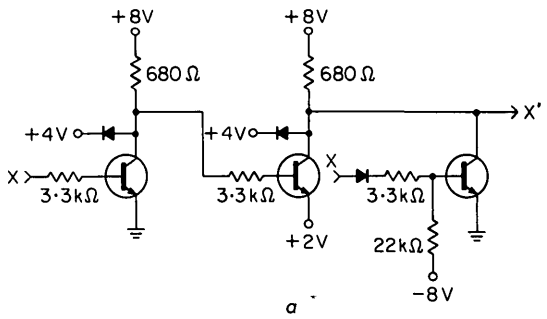


Fig. 3
Ternary invertor
a Circuit
b Transfer function
Horizontal scale: e_{in}, 1 major division = 1 V
Vertical scale: e_{out}, 1 major division = 1 V



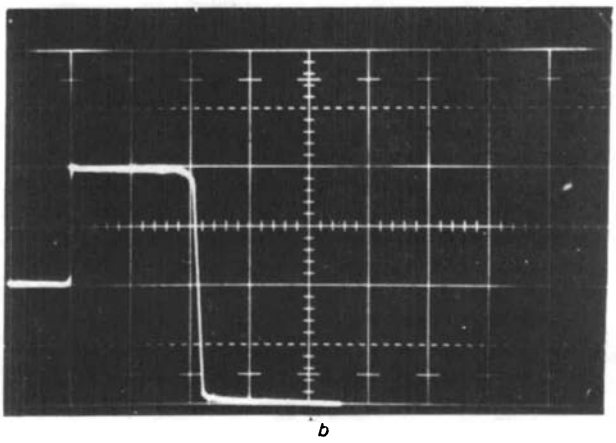


Fig. 4
Cycling operator
a Circuit
b Transfer function
Horizontal scale: e_{in}, 1 major division = 1 V
Vertical scale: e_{out}, 1 major division = 1 V

presence of such terms. One thus strives to design tristables exhibiting as many '—', (0, 1) and (1, 2) terms as possible.

Porat[4] shows several tristable flip-flops which he utilises in the design of ternary counters.

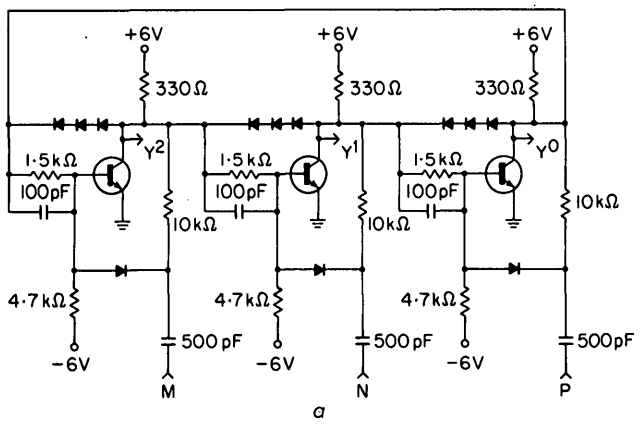
A tristable flip-flop can be obtained by cross-coupling two ternary invertors. The addition of a delay results in a D tristable, see Table 9. Q^n is the state of the D tristable at t^n, and Q^{n+1} is its state at t^{n+1} as a result of the input d at t^n.

Table 9

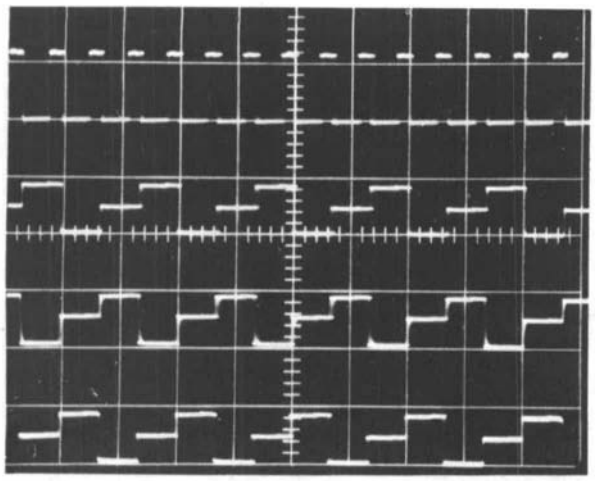
TRUTH TABLE OF THE 3-STATE D-TYPE FLIP–FLOP

Q^n	0			1			2		
Q^{n+1}	0	1	2	0	1	2	0	1	2
d	0	1	2	0	1	2	0	1	2

Fig. 5 shows a 3-transistor tristable flip-flop.[4] Its three inputs are (arbitrarily) denoted M–N–P, while the three outputs are Y^0, Y^1, Y^2, analogous to the binary $\bar{Y}$ and Y. In any



a



b

Fig. 5

M–N–P 3-state flip–flop

a Circuit diagram
b Waveforms for Fig. 5*a* connected as a mod 3 counter
(i) Input
(ii) Output at Y^2
(iii) Output at Y^1
(iv) Output at Y^0

state of the flip–flop, one transistor is n saturation while two transistors are in cut off. Assume that the collector at the Y^2 terminal is saturated ('0' sta c); then, Y^0 will be at a potential corresponding to logic '2', while Y^1 will be at logic '1', due to the potential drop across the diode string between Y^0 and Y^2 (Zener diodes could serve this function equally well). A pulse of an amplitude equivalent to logic '2', applied to terminal N will initiate regenerative action which will result in a new state, with T_{r1} in conduction and the collectors of T_{r2} and T_{r0} at logic '2' and '1', respectively. The tristable is now in state '2'. All the possible state transitions as a result of the inputs are shown in Table 10, where Q^n and Q^{n+1} refer to terminal Y^2.

Connecting the three input terminals together produces a modulo 3 counter which is analogous to the binary T flip–flop. Fig. 5*b* shows that, in any one state of the tristable, the outputs are 120° apart.

Table 10

TRUTH TABLE OF THE 3-STATE FLIP–FLOP OF FIG. 5*a*

Q^n (at Y^2)	0			1			2		
Q^{n+1}	0	1	2	0	1	2	0	1	2
M	–	0	0	2	(0,1)	(0,1)	(1,2)	0	0
N	(0,1)	(0,1)	2	0	0	(1,2)	0	0	–
P	0	(1,2)	0	0	–	0	(0,1)	2	(0,1)

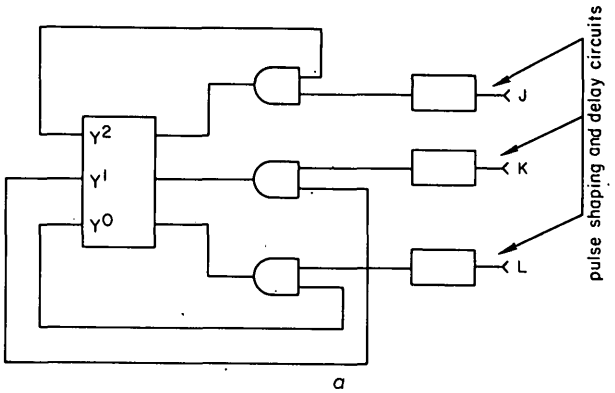
Fig. 6 shows the circuit of the J–K–L 3-state flip–flop. The similarity of its truth table (Table 11), with that of a binary J–K flip–flop should be noted. It differs from the D and the

Table 11

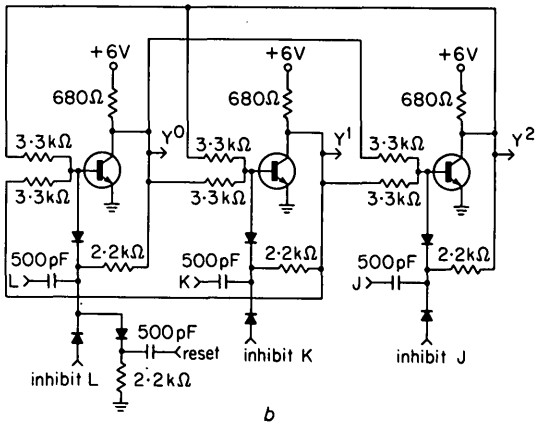
TRUTH TABLE OF THE 3-STATE FLIP–FLOP OF FIG. 6

Q^n	0			1			2		
Q^{n+1}	0	1	2	0	1	2	0	1	2
M	0	0	(1,2)	0	0	(1,2)	–	–	–
N	0	(1,2)	0	–	–	–	0	(1,2)	0
P	–	–	–	(1,2)	0	0	(1,2)	0	0

M–N–P types in that every output line can assume either logic '0' or logic '2' but not logic '1'. The three states of the flip–flop are defined according to which of the transistors is in the conducting state. In a sense, this flip–flop represents the



a



b

Fig. 6

J–K–L 3-state flip–flop

a Logic diagram
Circuit diagram

functions of a 3-state active memory cell in which each output is followed by a different j_k operator. The 'inhibit' terminals require a positive input to inhibit a change of state of the corresponding transistor. Modulo 3 counters are thus easily

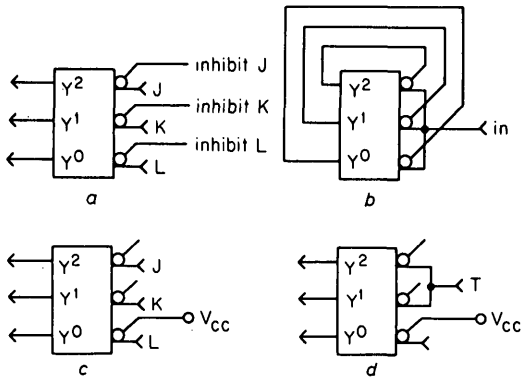


Fig. 7

J–K–L 3-state flip–flop

a Logic symbol including 'inhibit'
b Modulo 3 counter in ascending sequence (0–1–2)
c Connected as J–K (binary) flip–flop
d Connected as T (binary) flip–flop

realised, as shown in Fig. 7*b*. The J–K–L flip–flop transforms readily into a binary J–K or T flip–flop, Figs. 7*c* and 7*d*, respectively. A duodecimal counter was constructed,[4] utilising three J–K–L flip–flops without feedback connections or gates.

7 Sequential networks

Porat[4] constructed a variety of ternary counting circuits using the 3-state flip–flops of Section 6.2. Ternary-coded decimal 'up–down' counters were designed, and comparisons were drawn between the gating efficiencies of various ternary flip–flops. No difficulties were experienced in mixing ternary and binary flip–flops. The latter combination proved more efficient in realising coded-decimal counters than the corresponding networks built from elements of one radix only. In each case, one requires three ternary flip–flops, or a combination of binary with ternary flip–flops, to code one decade. Diode decoding of t.c.d.s requires fewer components than that decoding of b.c.d.s; forbidden combinations are efficiently represented, especially in mixed-radix counters which have fewer redundant states.

Design procedures are closely analogous to the well known binary techniques: (*a*) A table is drawn showing the states of the memory elements at times t^n and at t^{n+1}, following the application of a clock pulse. (*b*) The information contained in the tables is transferred to maps. (*c*) Excitation maps are drawn with help of the truth tables of the memory elements in use (see Section 6.2), to ensure that the desired states of $(Y_i)^{n+1}$ are uniquely achieved as a result of the inputs derived from the maps. (*d*) Gating expressions are obtained, after simplification of the maps obtained in step (*c*).

The design of a ternary-error check-digit generator[4] showed that state diagrams, equivalences and reduction of state tables can be handled in the same manner as the binary case.

8 Conclusions

The feasibility of 3-valued digital systems has been demonstrated. More theoretical work in multivalued logic, oriented towards synthesis of switching functions, is required.

Practical results have been obtained based on an algebra using operations &, V, j_k and h_k, utilising semiconductors in their switching mode, and retaining the reliability and safety margins typical of binary circuits. Efficiencies of combinational and sequential networks (in terms of number of active components) approach those of binary networks.

It appears that the problem of the most efficient algebra (in terms of gating requirements) should be solved first. Any composition(s) which are required for the implementation of such an algebra could then be obtained in integrated circuit form, by the iteration of operations that may be more amenable to semiconductor realisation than the original composition(s).

A multivalued gate can realise more complicated logic functions than a 2-valued gate with the same number of inputs. It also requires fewer output lines because of the higher information content available at each gate output. Thus, the properties of integrated circuits and multivalued gates seem well matched. A better understanding of multivalued algebras, together with the potentialities of integrated circuits, in implementing complex functions at low cost should stimulate more work in multivalued logic.

9 Acknowledgments

The author is indebted to Prof. F. G. Heath for his interest and encouragement and for illuminating discussions.

10 References

1 CARR, J. W., PERLIS, A. J., ROBERTSON, J. E., and SCOTT, N. R.: 'Proceedings of the seminar on the status of digital computer and data processing developments in the Soviet Union'. ONR Symposium report ACR-37, Washington, USA
2 BELL, D. A.: 'Information theory and its engineering applications' (Pitman, 1962, 3rd edn.)
3 SCHERER, E. H.: 'An error correcting code for quaternary data transmission'. Tech. Report AG-20, Westinghouse Electric Corporation, Baltimore, Md., USA, 1958
4 PORAT, D. I.: 'Three-valued combinational and sequential logic networks'. Ph.D. thesis, University of Manchester, England, 1967
5 'Synthesis of electronic computing and control circuit' (Harvard University Press, 1951)
6 GARNER, H. L.: 'Generalised parity checking', *IRE Trans.*, 1958, **EC-7**, pp. 207–213
7 SLUPECKI, J.: 'A criterion of fullness of many valued systems of propositional logics', *C. R. Soc. Sci. Varsovie*, 1939, **32**, Pt. III, pp. 102–109
8 BERLIN, R. D.: 'Synthesis of *n*-valued switching circuits', *IRE Trans.*, 1958, **EC-7**, pp. 52–56
9 KEIR, Y. A.: 'Algebraic properties of 3-valued compositions', *IEEE Trans.*, 1964, **EC-13**, pp. 635–639
10 POST, E. L.: 'Introduction to a general theory of elementary propositions', *Amer. J. Math.*, 1921, **43**, pp. 163–185
11 BERNSTEIN, B. A.: 'Modular representation of finite algebras'. Proceedings of the 7th international congress of mathematicians, Toronto, Canada, **1**, pp. 207–216, (University of Toronto Press, 1928)
12 LOWENSCHUSS, O.: 'Non-binary switching theory', *IRE Conv. Rec.*, 1958, **6**, pp. 305–318
13 MARTIN, N. M.: 'The Sheffer functions of 3-valued logic', *J. Symbolic Logic*, 1954, **19**, pp. 45–51
14 ROSSER, J. B., and TURQUETTE, A. R.: 'Many-valued logics' (North-Holland, 1952)
15 LEE, C. Y., and CHEN, W. H.: 'Several-valued combinational switching circuits', *Trans. Amer. Inst. Elect. Engrs.*, 1956, **75**, Pt. I, pp. 278–283
16 TANAKA, R. I. (Ed.): 'Research on automatic computer electronics'. Technical document report RTD-TDR-63-4173 (3 vols.), Wright-Patternson Air Force Base, Ohio, USA, 1964
17 MUKHOPADHYAY, A.: 'Symmetric ternary switching functions', *IEEE Trans.*, 1966, **EC-15**, pp. 731–739
18 MERRILL, R. D., JR.: 'Symmetric ternary switching functions: Their detection and realisation with threshold logic'. Lockheed Missile and Space Co., Sunnyvale, Calif. LMSC-6-75-65-29, Star, **3**, 1965
19 HANSON, W. H.: 'Ternary threshold logic', *IEEE Trans.*, 1963, **EC-12**, pp. 191–197
20 MERRILL, R. D.: 'Some properties of ternary threshold logic', *ibid.*, 1964, **EC-13**, pp. 632–635 (see also Reference 16, Vol. 2, pp. B187–B230)
21 KOOI, C. F., and WEAVER, J. L.: 'Nonlinear Hall effect ternary logic element', *Solid-State Electron.*, 1964, **7**, pp. 311–321
22 EDSON, W. A.: 'Frequency memory in multi-mode oscillators'. Technical report 16, Stanford Electronics Laboratory (Stanford University Press, 1954)
23 MUEHLDORF, E.: 'Multivalued switching algebras and their application to digital systems'. Proceedings of the National Electronics Conference, 1959, **15**, pp. 467–480
24 SCHAUER, R. F., STEWART, JR., R. M., POHM, A. V., and READ, A. A.: 'Some applications of magnetic film parametrons as logical devices', *IRE Trans.*, 1960, **EC-9**, pp. 315–320
25 PUGH, A.: 'Application of binary devices and Boolean algebra to the realisation of 3-valued logic circuits', *Proc. IEE*, 1967, **114**, (3), pp. 335–338
26 YOELI, M., and ROSENFELD, G.: 'Logical design of ternary switching circuits', *IEEE Trans.*, 1965, **EC-14**, pp. 19–29
27 ALLEN, C. M., and GIVONE, D. D.: 'A minimization technique for multiple-valued logic systems', *ibid.*, 1968, **EC-17**, pp. 182–184
28 MERRILL, R. D.: 'A tabular minimization procedure for ternary switching functions', *ibid.*, 1966, **EC-15**, pp. 578–585
29 ANDERSON, D. J., and DIETMEYER, D. L.: 'A magnetic ternary device', *ibid.*, 1963, **EC-12**, pp. 911–914
30 HALLWORTH, R. P., and HEATH, F. G.: 'Semiconductor circuits for ternary logic', *Proc. IEE*, 1962, **109C**, pp. 219–225
31 SANTOS, J., and ARANGO, H.: 'On the analysis and synthesis of three-valued digital systems'. Proceedings of the Joint Computer Conference, Spring 1964, pp. 463–475
32 THELLIEZ, S.: 'Note on the synthesis of ternary combinational networks using T gate operators', *Electron. Lett.*, 1967, **3**, pp. 204–207
33 MORRIS, D. J., and ALEXANDER, W.: 'An introduction to the ternary code number system', *Electronic Engng.*, 1960, **32**, pp. 554–557
34 SANTOS, J., ARANGO, H., and PASCUAL, M.: 'A ternary storage element using a conventional ferrite core', *IEEE Trans.*, 1965, **EC-14**, p. 248