Correspondence

Probabilistic Treatment of General Combinational Networks

KENNETH P. PARKER AND EDWARD J. McCLUSKEY

Abstract—**In this correspondence two methods are given for calculating the probability that the output of a general combinational network is 1 given the probabilities for each input being 1. We define the notions of the *probability of a signal* and *signal independence*. Then several proofs are given to show the relationship between Boolean operations and algebraic operations upon probabilities. As a result of these, two simple algorithms are presented for calculating output probabilities. An example of the usefulness of these results is given with respect to the generation of tests for the purpose of fault detection.**

Index Terms—**General combinational networks, input probability, output probability, probability, test generation.**

I. INTRODUCTION

In this correspondence we present two algorithms for general combinational logic circuits which allow the formulation of an *output probability* given a set of *input probabilities*. An output probability is defined as the probability that the output attains a specified value, and similarly, an input probability is the probability that a given input has the specified value. The circuit function (which is independent of circuit structure) effectively provides a transform from input to output probabilities. Here we emphasize that we are treating the signals in a probabilistic sense and not the circuit components or function. Indeed, it will be seen that when all input probabilities are set at either extreme (1 or 0), the probability function of the circuit produces identically the same result as the Boolean function of the circuit.

Previous work in the area is generally based on the classic paper by Von Neuman [6], which treated concepts of reliability, stochastic processing, processing in the presence of noise (error), etc. The derivation of the first algorithm (which we have formalized) has been informally presented in part by other authors [1], [5]. Our second algorithm offers significant improvement over the first, since it allows us to calculate output probabilities directly from the circuit description without need of any Boolean manipulations. This is done for general networks despite the fact that dependencies may exist between gate inputs. Finally, we have been careful to remove all physical interpretation from our concepts of probabilities. This allows us full freedom of application in either time or space.

II. ASSIGNING A PROBABILITY TO A LOGIC SIGNAL

First, we arbitrarily choose a "reference value" about which to define probabilities. We choose "1" and then define the *probability of a logic signal* as follows.

Manuscript received December 10, 1973; revised September 7, 1974. This work was supported in part by the National Science Foundation under Grant GJ-40286.

The authors are with the Digital Systems Laboratory, Departments of Electrical Engineering and Computer Science, Stanford University, Stanford, Calif.

Definition 1: The probability of a (logic) signal, expressed as

$$a = P(A = 1)$$

for signal A, is a real number on the interval [0, 1] which expresses the probability that signal A equals 1. [We use the convention here that upper case letters correspond to signal names (Boolean variables) and lower case letters represent the corresponding probabilities.] Since Boolean algebra is based on two-valued variables, we give the following definition.

Definition 2: The probability that signal $A = 0$ is given as

$$P(A = 0) = 1 - P(A = 1) = 1 - a.$$

III. OPERATIONS ON SIGNAL PROBABILITIES

The following statements relate Boolean operations to corresponding operations on probabilities.

Lemma 1: Boolean negation (NOT) such as in the expression $B = \bar{A}$ corresponds to the probability expression

$$b = 1 - a.$$

Proof: $b = P(B = 1) = P(\bar{A} = 1) = P(A = 0) = 1 - P(A = 1) = 1 - a$.

Before continuing, we must clarify the notion of *independence*. The concept of independence is that given a set of events, these events are individually determined by properties that are in no way interrelated. That is, event A (or subset of events A) in no way affects the outcome of event B (or subset of events B). The mathematical equivalent of the statement "A is independent of B" is

$$P(AB) = P(A)P(B).^1$$

Independence is a family property, i.e., if events $X_1, X_2, \cdots, X_n$ are independent, $[P(X_1X_2\cdots X_n) = P(X_1)P(X_2)\cdots P(X_n)]$ then all possible subsets of events are independent. We continue.

Lemma 2: Boolean conjunction (AND) of two independent signals A,B in the expression $C = AB$ corresponds to the probability expression $c = ab$.

Proof: $C = 1$ iff $A = 1$ AND $B = 1$. It follows from the independence property that;

$$c = P(C = 1) = P(A = 1 \text{ AND } B = 1)$$
$$= P[(A = 1)(B = 1)] = P(A = 1)P(B = 1) = ab.$$

Corollary: $Z = X_1X_2X_3\cdots X_n$ corresponds to the probability expression $z = x_1x_2x_3\cdots x_n$ provided all X_i are independent.

Lemma 3: Boolean disjunction (OR) of two independent signals A,B in the expression $C = A \vee B$ corresponds to the probability expression

$$c = a + b - ab.$$

Proof:

$$C = A \vee B = \overline{\bar{A}\bar{B}}$$
$$c = P(C = 1) = P(\overline{\bar{A}\bar{B}} = 1) = 1 - P(\bar{A}\bar{B} = 1)$$
$$= 1 - P(\bar{A} = 1)P(\bar{B} = 1)$$
$$= 1 - (1 - a)(1 - b) = a + b - ab.$$

[1] Interested readers are invited to pursue the issue of independence and the basic axioms of probability in a basic text such as Papoulis [3].

Corollary: $Z = X_1 \vee X_2 \vee \cdots \vee X_n$ corresponds to

$$z = 1 - \prod_{i=1}^{n} (1 - x_i)$$

provided all X_i are independent.

Lemma 4: Boolean disjunction (OR) in the expression $C = A \vee B$ with the restriction that the inputs A,B are not simultaneously 1 corresponds to the following expression:

$$c = a + b.$$

Proof: Since only 1 input A or B equals 1 at a time, their union is disjoint (i.e., $AB = 0$ and $P(AB = 1) = 0$). A basic axiom [3] of probability is that the probability of the disjoint union of events is the sum of the individual probabilities. Hence, the lemma follows directly from the axiom.

Corollary: $Z = \bigcup_{i=1}^{n} X_i$ where $X_iX_j = 0$ for all $i \neq j$ corresponds to

$$z = \sum_{i=1}^{n} x_i.$$

Notice that Lemma 3 can be shown from Lemma 4. Let $C = A \vee B$ with A,B independent. Then $C = AB \vee \bar{A}B \vee A\bar{B}$. But $AB\bar{A}B = 0$ and $ABA\bar{B} = 0$ and $\bar{A}BA\bar{B} = 0$ so Lemma 4 produces

$$\begin{aligned} c &= P(AB = 1) + P(\bar{A}B = 1) + P(A\bar{B} = 1) \\ &= ab + (1 - a)b + a(1 - b) \\ &= ab + a - ab + b - ab \\ &= ab + b - ab, \text{ which is the result of Lemma 3.} \end{aligned}$$

The following lemma defines the result of combining two signals that are totally dependent.

Lemma 5: Boolean conjunction (AND) of a signal A with itself in the expression $B = AA$ corresponds to the following probability expression:

$$b = a.$$

Proof: The proof makes use of conditional probability [3].

$$\begin{aligned} b &= P(B = 1) = P(AA = 1) \\ &= P(A = 1)P(A = 1 \mid A = 1) \\ &= P(A = 1)1 \\ &= a, \qquad \text{(from [3]).} \end{aligned}$$

Corollary: The following correspondences follow similarly:

$$P(A\bar{A} = 1) = 0$$

$$P(A \vee A = 1) = a$$

$$P(A \vee \bar{A} = 1) = 1.$$

The immediate consequence of Lemma 5 is that $P(AA = 1) = a$ and not a^2 as one gets from Lemma 2 by assuming independence. Similarly, by Lemma 2 we get $P(A\bar{A} = 1) = a - a^2$, by Lemma 3 we get $P(A \vee A = 1) = 2a - a^2$ and $P(A \vee \bar{A} = 1) = 1 - a + a^2$. Clearly, it is wrong to use Lemmas 2 and 3 in these cases since the operands are not independent. However, we note in each case that suppressing the exponent yields the correct result given by Lemma 5. This observation forms the basis for the second algorithm.

IV. TWO ALGORITHMS FOR PRODUCING PROBABILITY EXPRESSIONS FROM GENERAL NETWORKS

We base the first algorithm on the following facts:

1) Any general combinational network function can be expressed in a sum-of-products form.

2) A sum-of-products expression can be written as a canonical sum [2] of fundamental products Π_i.

3) In a canonical sum of Π_i, at most one $\Pi_i = 1$ for any input, i.e., $\Pi_i \Pi_j = 0$ for all $i \neq j$.

Algorithm One:

Step 1: From the circuit description obtain the canonical sum expression[2]

$$Z = \bigcup_{i=1}^{k} \Pi_i.$$

Step 2: For each Π_i in the canonical sum, produce the appropriate π_i product, where $\pi_i = P(\Pi_i = 1)$.

Step 3: The probability expression is given as

$$z = \sum_{i=1}^{k} \pi_i$$

which may be simplified algebraically.

Since each Boolean function has a unique canonical sum, there is a unique probability expression for each Boolean function.

Example 1: Given the circuit in Fig. 1, find the probability expression.

Step 1:

$$G = A(B \vee C) \vee BC = ABC \vee \bar{A}BC \vee A\bar{B}C \vee AB\bar{C}.$$

Step 2:

$$P(ABC = 1) = abc$$

$$P(\bar{A}BC = 1) = (1 - a)bc = bc - abc$$

$$P(A\bar{B}C = 1) = a(1 - b)c = ac - abc$$

$$P(AB\bar{C} = 1) = ab(1 - c) = ab - abc.$$

Step 3:

$$\begin{aligned} g &= abc + ab - abc + ac - abc + bc - abc \\ &= ab + ac + bc - 2abc. \end{aligned}$$

A quick analysis of algorithm one shows that though it is simple and straightforward, its complexity has an attainable upper bound of order 2^n where n is the number of circuit inputs. Clearly, the Boolean analysis in Step 1 is the cause of the complexity. We now present the second algorithm which overcomes this problem by eliminating all Boolean analysis and derives the output probability expressions directly from the circuit description.

We base the second algorithm on the following facts:

1) Dependencies occur between gate input signals only in circuits containing recombined fanout, only at the site of recombination.

2) The gate function (either AND or OR) at a "recombination site" will in either case require (from Lemmas 2 and 3) the multiplication of input expressions, and these input expressions will contain common variables.

3) Lemma 5 states that we should suppress exponents after multiplying expressions with common terms.

Algorithm Two:

Step 1: For each input signal and gate output in the circuit, assign a unique variable.

Step 2: Starting at the inputs and proceeding to the outputs, write the expression for the output of each gate as a function (using Lemmas 2 and 3) of its input expressions.

Step 3: Suppress all exponents in a given expression to obtain the correct probability expression for that signal.

We note that Step 3 is deliberately somewhat vague about when to suppress exponents. The reader should verify that as long as exponents are suppressed in the output expression the correct

[2] We can simplify the process by summing the product expressions for disjoint implicants.

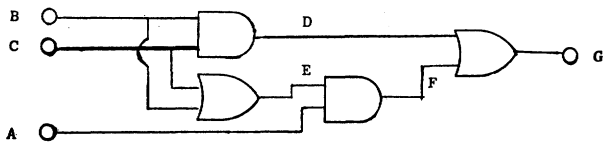

Fig. 1. Circuit realizing $G = A(B \vee C) \vee BC$.

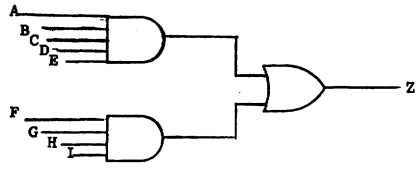

Fig. 2. Circuit realizing $Z = ABCDE \vee FGHI$.

result is obtained regardless of whether or not exponents are suppressed for internal signals. This has important implications towards the implementation of algorithm two, one of which being that existing symbolic mathematical processing programs can be used to derive output expressions for circuits with the simple requirement that the final result be "fixed up" by suppressing any exponents.

Example 2: Given the circuit in Fig. 1, find the probability expressions for the output and all internal signals.

Step 1: Assign variables A,B,C,D,E,F,G to the indicated signals.

Step 2: Then

$$d = bc,$$

$$e = b + c - bc,$$

$$f = ae = ab + ac - abc$$

$$g = d + f - df = bc + ab + ac - abc - ab^2c - abc^2 + ab^2c^2.$$

Step 3: Only expression for g needs to be operated on

$$g = ab + bc + ac - abc - abc - abc + abc$$

$$= ab + bc + ac - 2abc.$$

Analysis of algorithm two shows that it is less complex than algorithm one. It can be implemented using existing software with little or no modification. We obtain all internal expressions and we can be completely cavalier about when exponent suppression is done as long as it is done to the desired output expression. We will not discard algorithm one since for some circuits (e.g., circuits that have some regular structure to their prime implicants such as threshold functions and parity generators) it is quite suitable.

V. MOTIVATION

Motivation for this work comes from the observation that test generation for the detection of faults in digital circuitry can be treated probabilistically [4]. Consider the circuit in Fig. 2. We see that in order to detect the failure Z stuck-at-0 we need to provide a $Z = 1$ output. Thus, the probability of detecting Z stuck-at-0 is simply $P(Z = 1)$ when we apply a probabilistic distribution of ones and zeroes to the inputs.

Now

$$z = abcde + fghi - abcdefghi.$$

Let the input test vectors be chosen such that $P(a = 1) = P(B = 1) = \cdots = P(I = 1) = x$, which is the case when tests are created from a large set of random numbers, a popular method.

Then

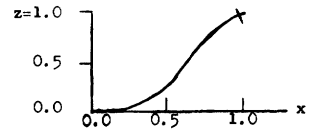

Fig. 3. Graph of $z = x^4 + x^5 - x^9$.

$$z = x^4 + x^5 - x^9$$

which is graphed in Fig. 3. Clearly, the probability of detecting the fault is highly dependent upon the value of x. It can be shown that many methods of producing tests from a list of raondom numbers inherently fixes x at the value of 0.5 and for the previous example, when $x = 0.5$, $z = 0.09$, and we can see in this instance why purely random test generation can be inefficient. As reported in [4], fair increases in testing efficiency can be obtained by controlling the value of x.

VI. CONCLUSION

We have demonstrated two methods for producing probability expressions for general combinational networks which relate the output probability to the set of input probabilities. These formulations are consistent with the requirements of probability theory, (can be easily implemented) and are immediately applicable to the practical problem of fault detection in logic circuits.

REFERENCES

[1] G. Klir, *Introduction to the Methodology of Switching Circuits.* New York: Van Nostrand, 1972.
[2] E. J. McCluskey, *Introduction to the Theory of Switching Circuits.* New York: McGraw-Hill, 1965.
[3] A. Papoulis, *Probability, Random Variables, and Stochastic Processes.* New York: McGraw-Hill, 1965.
[4] K. P. Parker, "Probabilistic test generation," Digital Syst. Lab., Stanford Univ., Stanford, Calif., Tech. Note 18, Aug. 1973.
[5] S. T. Ribeiro, "Random-pulse machines," *IEEE Trans. Electron. Comput.*, vol. EC-16, pp. 261–276, June 1967.
[6] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," C. Shannon and J. McCarthy, Ed. Princeton, N. J.: Princeton Univ. Press, 1956.

An Improved Bound for Checking Experiments that Use Simple Input–Output and Characterizing Sequences

THEODORE T. TYLASKA AND JAMES D. BARGAINER

***Abstract*—The least upper bound of $n(n - 1)/2$ input symbols is derived for the total length of characterizing sequences for a reduced n-state machine. When a machine can be characterized by a set of r characterizing sequences, a reduced upper bound on their total length will be $r(2n - r - 1)/2$ input symbols. To shorten transfer sequences, better use is made of the strong-connectedness assumptions of the machines to be tested. The reduced bounds on characterizing and transfer sequences are utilized to improve the upper bounds on checking experiments employing simple input/output (I/O) sequences.**

***Index Terms*—Characterizing sequences, checking experiments, sequential machine testing, simple I/O sequence.**

Manuscript received November 28, 1972; revised September 30, 1974.
T. T. Tylaska is with the Naval Underwater Systems Center, New London Laboratory, New London, Conn. 06320.
J. D. Bargainer is with the Department of Electrical Engineering, University of Houston, Houston, Tex. 77004.