MULTILEVEL-CIRCUIT DESIGN USING LOGIC TREES

by

HARRY LEE PUETT

B. S., Kansas State University, 1969

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1970

Approved by:

Major Professor

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

A number of different logic design aids have been devised and have been used in the past few years. The constant need for better and more economical logic circuits causes a never ending search for better design methods. This paper reports on the work of Green and Foulk (2) which deals with the uses of logic trees in multilevel-circuit design. These logic trees employ only 2-input 'AND' and 'OR' logic gates and are capable of mechanizing any desired Boolean function. The logic tree provides a standard adaptive-logic array for Boolean functions. Computer aided design procedures use the logic tree as a standard circuit, and multilevel representations of Boolean functions are derived using the logic tree (2). An original computer program which minimizes Boolean functions using the logic tree procedure is included in this report.

THE DERIVATION OF LOGIC TREES

This section presents a general derivation of the basic equations of the logic tree and the implementation of the basic equations using logic gates.

Let F_n represent a general Boolean function. F_n has n inde pendent variables X_i, where i = 1, 2, 3, . . ., n. F_n can be expressed in canonical form as a sum of products. Since F_n can be represented in this way, the general structure of F_n can be expressed as shown in Table 1. The X_i' means 'not X_i', '+' is the OR function, and '.' is the AND function. The K_i terms in Table 1 are either zero or one. For an example, consider the term $X_2 \cdot X_1$ in line (c) of Table 1. If the term $X_2 \cdot X_1$ is not a term in the canonical sum-of-products, the value of K_3 is zero. Thus the K_i terms are coefficients in the minterm expansion of the Boolean function being represented by F_n. If the proper choice of the values of the K_i variables is made, F_n can represent any Boolean function which contains n of the X_i variables (2).

The key to the logic tree is shown in equation (1). $F_j{}^*$ is the same general function as F_j, but $F_j{}^*$ and F_j have a different set of K_i coefficients.

$$F_n = X_n' \cdot F_{n-1} + X_n \cdot F_{n-1}^{*} \qquad (1)$$

The proof of equation (1) follows. Let line (b) in Table 1 be F_{n-1}. Then

$$F_{n-1} = F_1 = X_1' \cdot K_0 + X_1 \cdot K_1 \qquad (2)$$

and

$$F_{n-1}^{*} = F_1^{*} = X_1' \cdot K_2 + X_1 \cdot K_3 \; . \qquad (3)$$

Therefore

$$F_2 = X_2' \cdot (X_1' \cdot K_0 + X_1 \cdot K_1) + X_2 \cdot (X_1' \cdot K_2 + X_1 \cdot K_3) . \qquad (4)$$

If equation (4) is expanded,

$$F_2 = X_2' \cdot X_1' \cdot K_0 + X_2' \cdot X_1 \cdot K_1 + X_2 \cdot X_1' \cdot K_2 + X_2 \cdot X_1 \cdot K_3 \qquad (5)$$

which agrees with line (c) in Table 1. Thus equation (1) is true. Figure 1 shows the implementation of equation (1) using

Table 1

The Canonical Sum-of-Products Expressions for Each F_n

F_n	Line
$F_0 = K_0$	(a)
$F_1 = X_1' \cdot K_0 + X_1 \cdot K_1$	(b)
$F_2 = X_2' \cdot X_1' \cdot K_0 + X_2' \cdot X_1 \cdot K_1 + X_2 \cdot X_1' \cdot K_2$ $+ X_2 \cdot X_1 \cdot K_3$	(c)

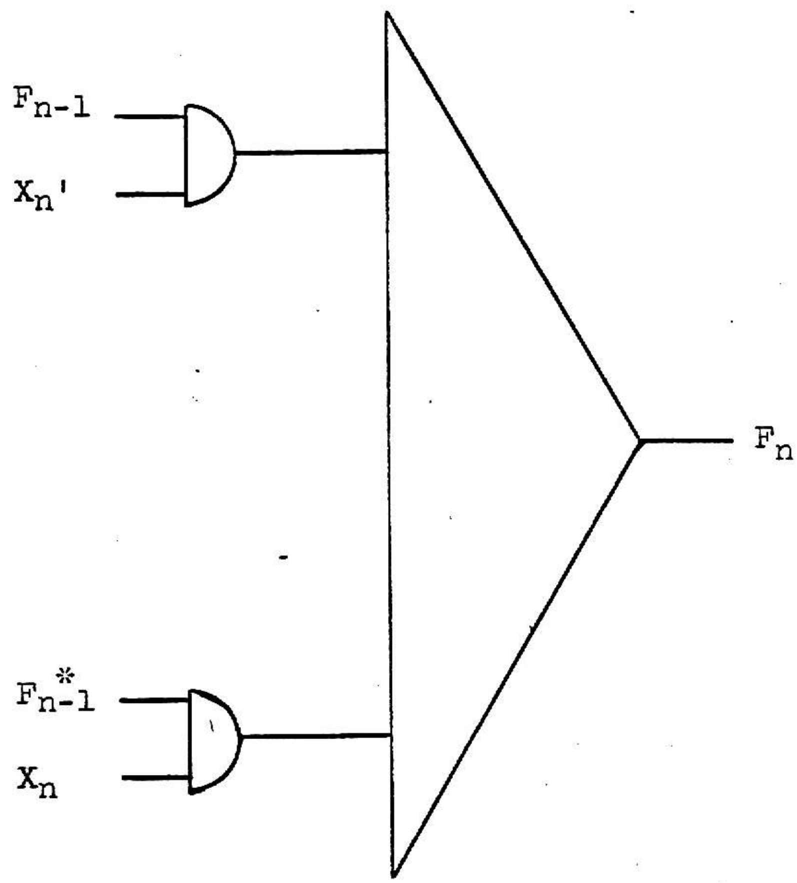

Fig. 1. The implementation of equation (1).
(Ref. 2.)