

2

REPRESENTATIONS OF LOGIC FUNCTIONS USING EXOR OPERATORS
Tsutomu SASAO

Department of Computer Science and Electronics
Kyushu Institute of Technology
Iizuka, Japan

Abstract—Logic functions are usually represented by logical expressions or decision diagrams using AND and OR operators. However, some functions have more compact representations with EXOR operators. This chapter surveys representations of logic functions using EXOR operators.

2.1 INTRODUCTION

Various methods exist to represent logic functions. A truth table is a straightforward representation, but its size increases exponentially when \( n \), the number of the input variables, increases.

Logical expressions and decision diagrams are useful to represent functions with many inputs, since they are often more compact than truth tables. The logical expressions and decision diagrams usually use AND and OR operators. However, some functions have more compact representation with EXOR operators. For example, to represent a parity function of \( n \) variables: \( f = x_1 \oplus x_2 \oplus \cdots \oplus x_n \), a sum-of-products expression (SOP) requires \( 2^{n-1} \) products while an AND-EXOR expression requires only \( n \) products. Thus, for this function, the EXOR-based representation is much more compact than the SOP-based ones.

This chapter surveys various methods to represent logic functions by using EXOR operators.

The rest of this chapter is organized as follows:
Section 2.2 introduces various trees using EXOR operators.

Section 2.3 defines various AND-EXOR expressions, compares the relations among them, surveys optimization methods, and compares their complexities.

Section 2.4 shows various decision diagrams using EXORs.

Section 2.5 introduces EXOR ternary decision diagrams (ETDDs).

2.2 TREES USING EXOR OPERATORS

In this section, we survey tree representation of logic functions.

2.2.1 Three Types of Expansions

Consider the following three types of expansions using EXOR operators:

\begin{align}
  f &= x f_0 \oplus x f_1, \\
  f &= f_0 \oplus x f_2, \\
  f &= f_1 \oplus \bar{x} f_2,
\end{align}

where \( f_0 \) (\( f_1 \)) is \( f \) with \( x \) replaced by 0 (1), and \( f_2 = f_0 \oplus f_1 \).

(2.2.1) is the Shannon expansion, where the EXOR operator is used instead of the inclusive OR operator. The Shannon expansion is denoted by S. (2.2.2) is the positive Davio expansion. This expansion uses only the positive literal, and is denoted by pD. (2.2.3) is the negative Davio expansion. This expansion uses only the negative literal, and is denoted by nD. Fig. 2.2.1 shows circuits corresponding to the three types of expansions.

These three expansions are the basis of the EXOR-based representation of logic functions.

2.2.2 Shannon Tree

By applying the Shannon expansions recursively to a logic function, we can represent a logic function by an expansion tree. Fig. 2.2.2 shows an example of