Information Theoretic Approach to Logic Functions Minimization

by

Denis V. Popel

ISBN: 1-58112-095-8

DISSERTATION.COM

USA • 2000

Information Theoretic Approach to Logic Functions Minimization

By

Denis V. Popel

SUBMITTED IN CONFORMITY WITH THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
TECHNICAL UNIVERSITY OF SZCZECIN
SZCZECIN, POLAND
FEBRUARY 2000

To my son, Nikita

CONTENTS

ABSTRACT

Information theory methods are of wide use in contemporary logic design, but their proper application to Computer Aided Design (CAD) is rather impossible without strong theoretical and practical justification. Our research is focused on logic function minimization which is an essential component of any system for digital circuit design. The well-known information theory methods to minimize logic functions should be improved and developed towards new problems appeared while increasing the number of CAD applications.

We report new results on logic functions minimization by information theory standpoint. We have developed an information theoretic model of recursive decomposition of logic functions. Based on this model, a novel technique for efficient Decision Tree design of various types (AND/OR, Reed-Muller, Kronecker Decision Trees, etc.) is presented. We develop the algorithms that employ information theoretic measures to minimization of the logic functions represented by Decision Trees. We investigate the effect of information theory appliance to minimization of multiple-valued, incompletely specified functions, and to symmetry detection as well. We give the extended experimental study to validate the approach.

The presented results can be treated as a significant step towards better understanding of the behavior of digital circuits from the information theory point of view. These achievement creates the prospect of solving the wide circle of logic design problems, including low-power and high-testable circuits synthesis.

TERMINOLOGY AND ABBREVIATION

Logic function and its variables

$f = f(x_1, \cdots, x_n)$ - a *logic function* of n variables;

$f = (f_1, \cdots, f_m)$ - a *system of m functions* (or *multi-output function*);

$X = \{x_1, \cdots, x_n\}$ - the set of function variables;

Decomposition (expansion) of logic function

$f_{|x=a}$ - a *cofactor* (*sub-function*) of function f with respect to value a of variable x;

$f_{|x=0} \Leftrightarrow f_{\overline{x}}$ - the cofactor of function f with respect to $x = 0$;

$f_{|x=1} \Leftrightarrow f_x$ - the cofactor of function f with respect to $x = 1$;

$f = \overline{x} \cdot f_{\overline{x}} + x \cdot f_x$ - the *Shannon (S) expansion* of switching function f with respect to variable x in AND/OR form;

$f = \overline{x} \cdot f_{\overline{x}} \oplus x \cdot f_x$ - the *Shannon (S) expansion* of switching function f with respect to variable x in AND/EXOR form;

$f = f_{\overline{x}} \oplus x \cdot (f_{\overline{x}} \oplus f_x)$ - the *positive Davio (pD) expansion* of switching function f with respect to variable x;

$f = f_x \oplus \overline{x} \cdot (f_{\overline{x}} \oplus f_x)$ - the *negative Davio (nD) expansion* of switching function f with respect to variable x;

$\omega \in \{S, pD, nD\}$ - an expansion of switching function f.

Numbers of patterns

k - the number of distinct combinations (patterns) of variables values;

$k_{|f=b}$ - the number of combinations (patterns) where $f = b$;

$k_{|x=a}$ - the number of combinations (patterns) where $x = a$;

$k_{|f=b,x=a}$ - the number of combination (patterns) where $f = b$ and $x = a$.

Information theoretic measures

$H(f)$ - *entropy of function* f;

$H(f,x)$ - *joint entropy of function* f *and variable* x;

$H(f|x)$ - *conditional entropy* of function f given variable x;

$I(f;x)$ - *mutual information* between the function f and variable x (in other words, information in variable x about function f);

$H^{\omega}(f|x)$ - *information theoretic measure* of expansion ω given variable x;

$f^{level.node}$ - sub-function f that corresponds to the incoming edge of the *node* attaching at decision tree *level*;

$p^{level.node}$ - probability of the event: a *node* is attached at decision tree *level*;

$I^{level.node}$ - information carried by a *node* at decision tree *level*.

Decision Trees and Logic Expressions

DT - Decision Tree

DD - Decision Diagram

RM - Reed-Muller

PPRM - positive polarity Reed-Muller

FPRM - fixed polarity Reed-Muller

KRO - Kronecker

PSDRM - pseudo Reed-Muller

PSDKRO - pseudo Kronecker

GF - Galois field

ACKNOWLEDGEMENTS

I would like to thank Assoc. Prof. S. Yanushkevish, my supervisor, and Prof. V. Shmerko, my promoter, for constant support during this research. I am also thankful to Dr. V. Cheushev for some useful contributions.

Discussion with Prof. A. Zakrevskij (Academy of Science, Belarus), helped me to understand issues of exact and heuristic algorithms. Prof. R. Stanković (University of Nis, Yugoslavia) provided many useful comments during this research.

I wish to thank Prof. T. Sasao (Kyushu Institute of Technology, Japan) for reading and useful remarks.

I am also grateful to the head of the Department of Computer Science, Prof. R. Sadykhov, and the teaching stuff of State University of Informatics and Radioelectronics, Belarus.

During the last years the research has been supported by the Ministry of Education of Belarus (personal Grants), Fund of Fundamental Researches of Academy of Science (Belarus) and State Committee for Scientific Research of Poland (Poland-Belarus joint research/visit program).

On personal level, my family, provided a lot of love and constant support.

Szczecin, Poland **Denis Popel**

February 2000

1

INTRODUCTION

The recent achievements and opportunities created by modern microelectronic technology motivate the enhanced demand of logic synthesis tools. Thus, the complexity of integrated circuits has increased exponentially during last decades. To handle the complexity of today circuits, the design engineers are totally dependent on Computer Aided Design (CAD) tools. The capabilities and limitations of CAD tools have crucial impact on the performance and cost of produced circuits as well as on the resources required to develop a circuit.

Most of problems arising a CAD of integrated circuits are NP-hard optimization problems. One of them is the minimization of logic functions. Through the years, the enormous number of minimization techniques have been proposed to solve the problem, most of them are based on traditional paradigms and methods. In recent years, the methods of *Information Theory* have found the widespread application in optimization problems of CAD, offering a realistic alternative to traditional ones.

1.1 Motivation and Overview

In 1938, Shannon introduced the method for decomposition of switching functions, well-known as *Shannon expansion* [92]. In 1948, he suggested a measure to represent the information in numerical value, so called *Shannon entropy* [93][1]. Here we proposed an idea to merge both conceptions and direct it toward one of the circuit design problems, namely, *minimization of logic functions* in different design styles.

The minimization problem for logic functions is far from to be solved, and it seems that the algorithms (both the exact and heuristic) have been still permanently

[1] The effect of the paper on communications [93] in both theory and practice are still being felt, and Shannon entropy function has been applied very successfully to several wide-range areas of computer science.

improved [82, 101, 102]. Usually, they optimize realization cost, e.g. the number of product terms, or/and total literal cost, i.e. the number of literals in all terms. Such criteria are suitable for design of two-level circuits, i.e. *Programmable Logic Arrays* (PLAs) [87, 5], otherwise, there are no restriction for *Programmable Logic Devices* (PLDs) [116, 1, 47] or *Field Programmable Gates Arrays* (FPGAs) [116, 88, 83, 52, 29].

To solve the problem, we concentrate on state-of-the-art Decision Tree (DT) design[2] via information theoretic approach. We exploit the fact that a logic function can be represented by a DT. The structure of DT is extremely sensitive to variables order and selection of expansion types. We present an information theoretic model of DTs design and several algorithms for minimization of logic functions based on this model.

We address to design DTs of different classes (e.g. Reed-Muller, Kronecker DTs) based on information theoretic measures. Our investigation is motivated by growing interest to AND/EXOR expressions in CAD [90, 101, 21]. Implementation of AND/EXOR expressions often results in a more economical realization of the circuit (in terms of gates and interconnections, less area in VLSI implementation) and is often more easily tested. This is particularly efficient for error control, arithmetic circuits, encrypting and coding schemes [24, 102, 86]. However, the known optimization strategies on DTs cannot be directly used to optimize of AND/EXOR expressions.

We overview below the information theoretic methods whenever developed for various CAD applications.

1.2 Applications of Information Theory in VLSI CAD

For three decades history of application of information theory methods to the CAD of integrated circuits, the number of promising results have been obtained:

Minimization of switching functions [43, 16] and multiple-valued logic functions [53] that related to earlier works on conversion of decision tables and truth tables to AND/OR DTs [30, 37, 32];

Testing of combinational circuits [2, 110, 14];

[2]Decision Trees and Decision Diagrams (DDs) design has proven their usefulness in many applications of logic synthesis as efficient data structure for representing and manipulating switching functions [8, 7, 21, 86].

Estimation of Power Dissipation [56, 40, 57];

Application to Other CAD Related Problems: DDs design [79] based on machine learning methods ($ID3$ algorithm and its modifications [39]), information theoretic measures of logical gates [17] and logical expressions [123], estimation of logical work in digital networks [38], theory of information engine and information networks [113, 114].

There are well-known methods to exploit the probabilities, for example, to decomposition [81, 42] and minimization of logic functions [61], that are closely related to information theoretic approach.

In our study on application of information theory methods to minimization of logic functions, we are based on previously obtained results in AND/OR DTs design [32, 43, 53, 16]. Note, that these results are related to earlier works [30, 37] on conversion of decision tables into DTs. Because the truth table of a logic function is a special case of a decision table, a natural step was to use the proposed algorithms for minimization of switching functions. In the thesis, we show only one application of information theory concept, namely, to logic functions minimization.

The recent results have been devoted to application of information theoretic methods to minimize AND/OR expressions via design of AND/OR DTs. However, there were no reports on application of information theory concept to minimization of AND/EXOR expressions, that is a key unsolved problem in CAD [90, 63]. Such investigation is of primary interest of our research. We will show that the known information theoretic methods for logic function minimization cannot be directly applied to AND/EXOR expressions.

1.3 Thesis Contribution

Based on the previous remarks, this research addresses the following issues.

Efficient Design of AND/EXOR DTs. We investigate the prospects to exploit information theoretic measures for design of AND/EXOR DTs. We direct toward top-to-down DTs design and present an information theoretic model of recursive decomposition of logic functions, where different decomposition types are used (Shannon, positive and negative Davio). From information theory viewpoint, DT design is interpreted as step-by-step reduction of entropy of the minimized function. The main distinctive feature of the proposed model

is *unified* measures to choose the "best" variable and the "best" expansion type during decomposition. The experimental results demonstrate the potential ability of information theoretic measuring in AND/EXOR DT-based algorithms to minimize logic functions.

We propose information theoretic criteria that exploits the fact that the behaviour of cost function (the number of terms and literals) is coherent to the behaviour of entropy function (see Figures 2.5, 4.5 and 4.11). This issue will be discussed in detail in Section 4.1.3.

Recognition of Symmetries. The properties of symmetries can make DTs design more simple, thus the symmetric variables are assigned to neighboring nodes of the DTs. We propose a method of symmetries recognition that it is an integral part of the DT design where the identical information theoretic measures are used. We successfully recognize symmetries in 70% of cases and obtain an improvement of run-time in about two times against the well-known method that exploits DDs design for symmetric functions. It is discussed and supported with examples in Sections 5.3 and 5.4.

Self-sufficient Contributions.

Information Prognosis. The pruning technique is widely used in exact algorithms of DT design to reduce the search space of possible solutions [52, 35]. We propose a novel technique, called *information prognosis*, that enables us to decide about rudiment solutions during DT design (Section 4.2). Our technique greatly accelerates exact algorithms and enlarges the scale of the functions to which the exact algorithm can be applied.

Efficient Design of Ternary DTs. We propose the information content of Ternary DTs. In this connection, we introduce special class of SOP TDTs, we call it Morreale Ternary DTs, and extend information theoretic approach to SOP minimization. We improve the results of minimization of AND/EXOR expression using EXOR ternary DTs as well.

Efficient Design of Word-level DTs. We investigate the applications of information theoretic approach to minimize arithmetic expression using word-level DTs. As the main contribution, we present free pseudo Kronecker Binary Moment Trees and extend our information theoretic model. Experiments show that we obtain the arithmetic expressions of

about 70% fewer terms and about 60% fewer literals in comparison with the expressions that obtained using Binary Moment Trees.

Minimization of Multiple-valued Functions. We develop information theoretic approach to minimize polynomial expressions over GF(4). Also, we introduce free pseudo Kronecker GF(4) DTs. Our results show that, in addition to providing an elegant theoretic framework for nodes selection criteria, this approach results in up to 30% improvement in the number of terms and literals against the methods that utilize analogues of AND/OR DTs in GF(4).

Minimization of Incompletely Specified Functions. In many practical applications related to digital system design, it is a basic technique to use functions containing don't cares [13, 124, 50]. In our information theoretic technique to handle don't cares, we are based on principle "*do not care about don't cares*". Experimental results show that this technique is very fast, yet produced better results in comparison with the well-known methods to explore don't cares. In particular, for weakly specified switching functions we archive the improvement about 1.5 times in the number of terms and literals in comparison with *Staircase* strategy [121].

Efficient Design of Free DTs. In all proposed algorithms, we adopt *free* DTs. The term 'free' means that different variables and types of node (types of expansion) can occur at every DT level. In practice, we obtain the run-time reduction in about 30% (see Section 4.1) in comparison with the results on ordered DTs where the occurrence of variables at every DT level is fixed.

The principal part of these results has appeared in [97, 94, 72, 74] and extended in [98, 120, 75, 77, 78, 76, 73].

1.4 Thesis Outline

Chapter 2 collects the definitions and notations necessary to understand the remaining Chapters. It also introduces the useful information theory notations - we rely heavily on information measures employed in DT design.

Chapter 3 yields an information theoretic model of DT design and provides theoretic issues of the thesis. The information theoretic model of DT design is the most important idea of the research.

Chapter 4 contains the algorithms to minimize logic expressions of switching functions based on the proposed information theoretic approach. We offer the heuristic and exact algorithms to solve minimization problem. We also develop the algorithms for incompletely specified functions.

Chapter 5 presents a novel method to recognize symmetries that is based on information theoretic measures. The properties of symmetry are of common use in digital system design, so the proposed method has to find many applications.

Chapter 6 is devoted to information theoretic approach to AND/OR and AND/EXOR minimization via efficient design of Ternary DTs, and improves the results presented in Chapter 4.

Chapter 7 develops the information theoretic approach towards minimization of multiple-valued polynomial expressions over GF(4) via design of Reed-Muller GF(4) and Kronecker GF(4) DTs.

Chapter 8 provides another application of information theoretic approach towards minimization of arithmetic expressions and introduce new types of word-level DTs.

Chapter 9 summarizes the results and gives the directions of future research.

Appendix contains an example of applications of information theoretic approach to decision making, and also specification of the $Info$ package, a collection of the developed programs for minimization, that has been created as a helpful tool for research.

Figure 1.1 illustrates the structure of the thesis and obtained results.

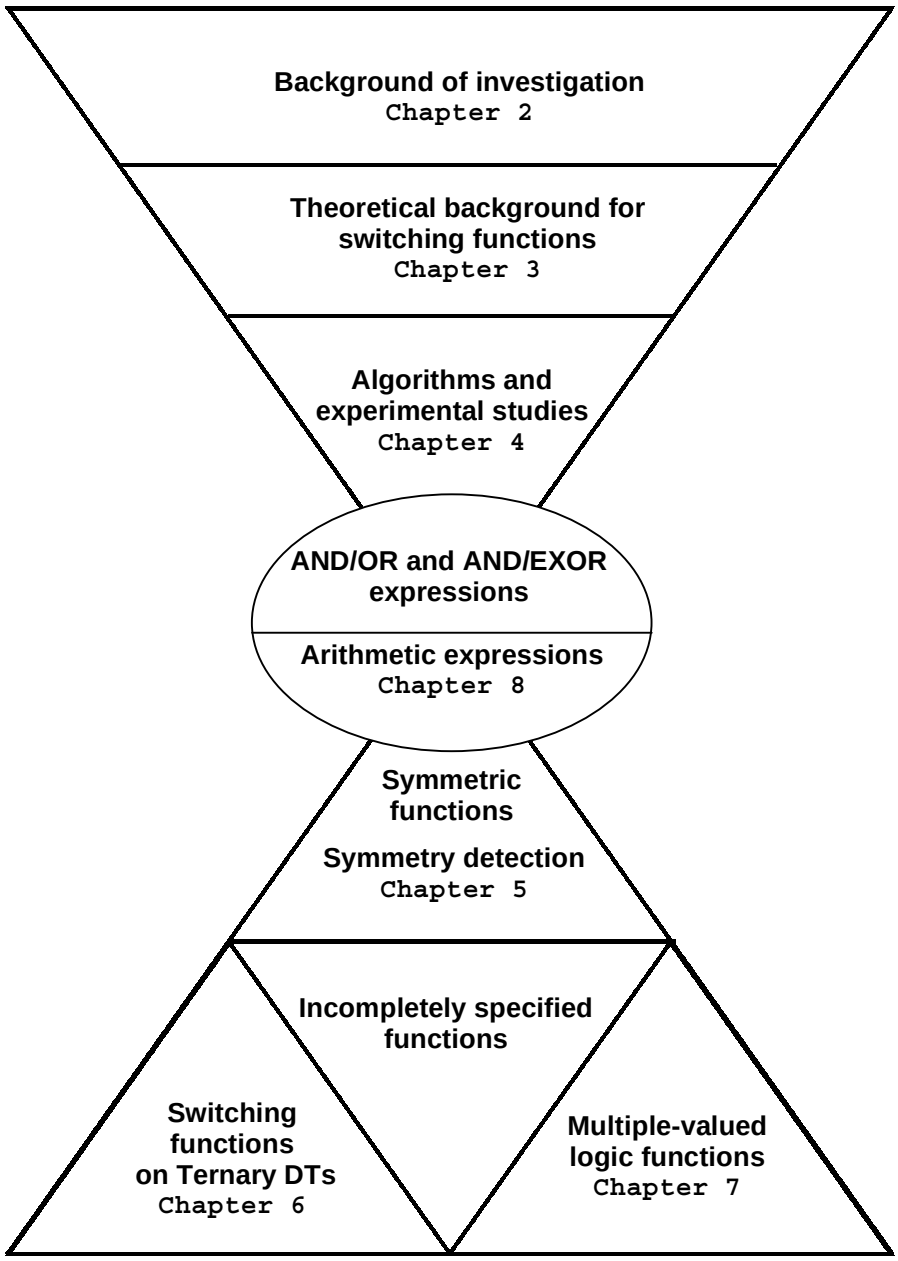

Figure 1.1: Thesis structure and obtained results

2

BACKGROUND

This Chapter provides a necessary background needed for further explanation. We give a number of basic definitions for various representation forms of logic functions (truth table, DTs, logic expressions) and analyze the relationship between DTs and logic expression. Next, we discuss information theoretic measures for logic functions as far as they are necessary for the following Chapters. Furthermore, we overview up-to-date preliminaries to logic function minimization and give optimization criteria. The problems of DTs design are considered as well.

2.1 Decomposition of a Switching Function with Respect to Variable

In the following, if not mentioned opposite, we consider the *switching functions* f: $\mathbf{B}^n \rightarrow \mathbf{B}^m$ over the variable set $X = \{x_1, \cdots, x_n\}$, where $\mathbf{B}$=$\{0,1\}$. Here, n is the number of variables, and m is the number of functions (outputs).

We now investigate different types of decomposition (expansion) of a switching function f with respect to variable x, to uniquely determined sub-functions so that it is possible to reconstruct f if the sub-functions and decomposition type are known.

Definition 2.1. A *Decomposition (Expansion) of type ω of a switching function f* is defined as

$$f = Expansion(x, \omega, f_0, \ldots, f_q), \tag{2.1}$$

such that for $\forall x \in X$, there exist $q+1$ uniquely determined sub-functions $f_0, \ldots, f_q$.

Notation 2.1. For a switching function f,

$$f_{|x_i=c} = f(x_1, \ldots, x_{i-1}, c, x_{i+1}, \ldots, x_n)$$

is called a *cofactor* of f, when x is fixed to $c \in \{0,1\}$. We also write $f_{\overline{x}}$ and f_x instead of $f_{|x=0}$ and $f_{|x=1}$, respectively.

Thus, Shannon decomposition is given by

$$f = \overline{x} \cdot f_{|x=0} \vee x \cdot f_{|x=1}. \tag{2.2}$$

Generally, the decomposition of a switching function f with respect to arbitrary variable x, or, in other words, the expansion of f given x, can be represented by the formula:

$$f = (c_0 \cdot \overline{x} \cdot f_0) \circ (c_1 \cdot x \cdot f_1) \circ (c_d \cdot f_d), \tag{2.3}$$

where c_0, c_1, c_d are the coefficients of the expansion, and symbol $< \circ >$ denotes a logical operation, e.g. inclusive OR and exclusive OR (EXOR).

There are different expansions derived from possible combinations of the operation $< \circ >$ and coefficients $C = [c_0, c_1, c_d]$ in Equation (2.3). Some well-known of them are:

Shannon (S) **expansion,** $C = [110]$, $< \circ >$ is OR or EXOR operation[1]

$$f = \overline{x} \cdot f_{|x=0} \oplus x \cdot f_{|x=1}, \tag{2.4}$$

positive Davio (pD) **expansion,** $C = [011]$, $< \circ >$ is EXOR operation

$$f = f_{|x=0} \oplus x \cdot (f_{|x=0} \oplus f_{|x=1}), \tag{2.5}$$

negative Davio (nD) **expansion,** $C = [101]$, $< \circ >$ is EXOR operation

$$f = f_{|x=1} \oplus \overline{x} \cdot (f_{|x=0} \oplus f_{|x=1}), \tag{2.6}$$

ternary (T) **expansion,** $C = [111]$, $< \circ >$ is OR or EXOR operation

$$f = \overline{x} \cdot f_0 \circ x \cdot f_1 \circ f_d. \tag{2.7}$$

Here $f|_{x=0} \oplus f|_{x=1} = {}^{\partial f}/_{\partial x}$ is a *Boolean derivative* of f with respect to x. Based on the Shannon and Davio expansions, any switching function f can be recursively decomposed to simple sub-functions until the constant functions are obtained.

Remark 2.1. It has been proven in [19] that there exist only three relevant different unique decompositions, S, pD and nD, of a switching function to two sub-functions.

[1]Note, that a switching function is unchanged if exclusive OR ($\oplus$) is replaced with inclusive OR for Shannon expansion.

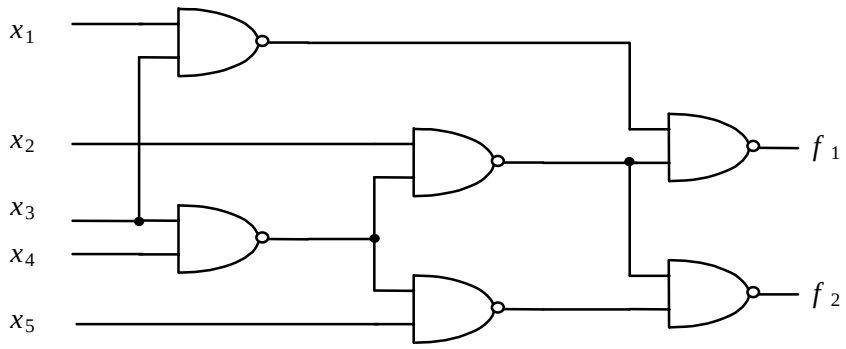

Figure 2.1: An example of digital network - circuit c17

2.2 Representation of Logic Functions

Below, the different representations of logic functions are given (i.e. table-like notations, DTs or DDs, logic expressions). We utilize these notations in next Chapters.

Example 2.1. *To illustrate the different representation forms of logic functions in the following, a small example is shown. We have chosen the benchmark c17* [2] *for that purpose (Figure 2.1).*

Any switching function f can be uniquely determined by a *truth table* on n combinations of variables values. As alternative, a *truth column vector* can be used.

Example 2.2. *The truth table for the circuit from Example 2.1 is given in Table 2.1. The truth column vector of the multi-output function f is represented by integer values:* [01010100333333000101232233333322].

2.2.1 Decision Trees and Graph-Based Notations

Decision Trees (DTs) and Decision Diagrams (DDs) are graph-based structures which have become the advanced structures in VLSI CAD for representation and manipulations of logic functions [90, 86, 21]. The core of the data structures is a

[2] *http://www.cbl.ncsu.edu/pub/Benchmark_dirs/LGSynth91*

x_1	x_2	x_3	x_4	x_5	f_1	f_2	f	x_1	x_2	x_3	x_4	x_5	f_1	f_2	f
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	1	1	1	0	0	0	1	0	1	1
0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0
0	0	0	1	1	0	1	1	1	0	0	1	1	0	1	1
0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	2
0	0	1	0	1	0	1	1	1	0	1	0	1	1	1	3
0	0	1	1	0	0	0	0	1	0	1	1	0	1	0	2
0	0	1	1	1	0	0	0	1	0	1	1	1	1	0	2
0	1	0	0	0	1	1	3	1	1	0	0	0	1	1	3
0	1	0	0	1	1	1	3	1	1	0	0	1	1	1	3
0	1	0	1	0	1	1	3	1	1	0	1	0	1	1	3
0	1	0	1	1	1	1	3	1	1	0	1	1	1	1	3
0	1	1	0	0	1	1	3	1	1	1	0	0	1	1	3
0	1	1	0	1	1	1	3	1	1	1	0	1	1	1	3
0	1	1	1	0	0	0	0	1	1	1	1	0	1	0	2
0	1	1	1	1	0	0	0	1	1	1	1	1	1	0	2

Table 2.1: Truth table of switching function f from Example 2.1

directed acyclic graph which forms a canonical representation of a given function[3]. DTs are described here in detail, since they will be considered in Chapters below.

Definition 2.2. *Decision Tree* is a connected, directed acyclic graph $Tree = \{V, E\}$ with the vertex (node) set V and the edge set E, where:

(i) Each non-terminal vertex is labeled by an expansion ω with respect to a variable x assigned, as decision variable, to this node.

(ii) Each non-terminal vertex corresponds to a decomposition step of the function f (incoming *edge*) into sub-functions (outgoing edges: $edge_l$ and $edge_r$) with respect to the couple (x, ω).

(iii) A terminal vertex is labeled with the leaf value and has no successors.

(iv) A non-terminal vertex has exactly two successors for binary DT and three successors for ternary DT.

Remark 2.2. The *size* of graph-based representation (DT or DD) is derived by the number of its nodes.

Definition 2.3. A DT is called *pseudo* if an arbitrary expansion type is assigned to each variable.

[3]The same or similar representations have been successfully applied in studies of information theory, where the term *branching programs* has been used [30, 115]. These representations describe a target function by a series of decisions that depend on input values; the overall decision takes one of possible function values.