Advances in the Boolean Domain

Advances in the Boolean Domain

Edited by

Bernd Steinbach

Cambridge
Scholars
Publishing

Advances in the Boolean Domain

Edited by Bernd Steinbach

This book first published 2022

Cambridge Scholars Publishing

Lady Stephenson Library, Newcastle upon Tyne, NE6 2PA, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN (10): 1-5275-8872-6
ISBN (13): 978-1-5275-8872-1

Contents

IV Future Technologies: Quantum Computing 141

List of Figures

List of Tables

Foreword

Boolean variables are the basis of all research, progress, and applications belonging to the Boolean domain. Boolean variables are the simplest variables of all because these variables can store only two different values. In propositional logic the values true and false are usually used as values of Boolean variables that are also denoted by logic variables. Descriptions of circuits sometimes use the values H (high voltage) and L (low voltage) as values of a Boolean variable. Most often the values 1 and 0 determine the two different values of a Boolean variable.

Is it beneficial to read a book about advances in the Boolean domain? The answer to this question is YES. Looking back in the history, scholars, scientists, and engineers tried over a very long time to build a computer and failed. The use of switching elements with only two states (OFF and ON) facilitated Konrad Zuse's successful construction of the first computer. The performance of computers has been strongly increased since that point in time, but the basis remains switching elements and circuits that can be described by Boolean variables, Boolean expressions, Boolean functions, and several representations of such functions.

An old proverb says "Everything has two sides". This proverb is also true in the Boolean domain. The simplicity of Boolean variables on the one side usually requires on the other side many Boolean variables to describe a certain problem and a Boolean function of n Boolean variables determines 2^n Boolean function values. Hence, we are faced in the Boolean domain with an exponential complexity. Successful approaches that contribute to increased large numbers of Boolean variables are explained in this book.

Another important view to the Boolean domain relates to the word "bit" (short for binary digit). A bit is the smallest unit of information

and represents a logic state with only two values; hence, a bit can be expressed by a Boolean variable. The new aspect is the transmission of information using a sequence of bits, i.e., a sequence of values 0 and 1. It is possible to transmit more than 10^{11} bits per second using a single optical fiber.

The combination of computers, special digital devices, and transmission channels like wired Ethernet, wireless local area network (WLAN), digital enhanced cordless telecommunications (DECT), Bluetooth, and many others has generated a gigantic number of high quality applications used in our daily lives. We can notice day-to-day the progress in digitization. Results of the research in the Boolean domain are often the basis for new digital application that contribute to a better live.

Both the number of topics of research in the Boolean domain and the number of recently developed digital applications is so large that strong selection was required to realize a book of an appropriate size. I thank all authors of the chapters of this book for their great contributions and kind collaboration. Their successful research and preparing the found results in an understandable manner as chapters of this book are the key that this book could be created as a new highlight in the Boolean domain.

My thanks go also to Michael Miller for his careful proofreading of the complete book and all members of the staff of Cambridge Scholars Publishing for their support and fruitful collaboration in preparing this scientific book. I hope that the readers enjoy reading this book and get helpful suggestions for their own future work to reach further advances in the Boolean domain as well as associated applications.

Bernd Steinbach

Department of Computer Science
Freiberg University of Mining and Technology
Freiberg, Saxony, Germany
April 2022

Preface

The progress in the Boolean domain supports a wide field of applications in digitization. Many new theoretical insights, concepts, methods, and tools contribute to this progress. Some of these new results are presented in this book.

Computers are the main technical basis of digitization due to their ability to compute and store data. A second important resource of such applications is the net between computers so that an interchange of information becomes possible. Data must be protected against attacks while computed, stored, and transmitted. Encryption of data is an effective approach to protect data against several types of attacks. Logic functions are used to realize both encryption and decryption procedures. The achieved protection of the encrypted data against attacks strongly depends on the properties of the logic functions used. The three chapters of Part I contribute new results of research to increase cyber security.

Boolean functions, also denoted as logic functions or switching functions, assign a Boolean value 0 or 1 to each of the 2^n different vectors of n Boolean values. Such functions play a central role in almost all procedures that solve a certain Boolean problem. There are many possibilities to represent a Boolean function; several of them will be introduced in Chapter 1 so that this chapter serves also as a basis for all other chapters of this book. The effort to solve a given problem usually depends on the chosen representation. The description of a Boolean function as a disjunction (OR) of conjunctions (AND) of optionally negated Boolean variables is widely used; this form of a Boolean expression is often denoted as the Sum Of Minterms (SOM). However, important properties needed for cryptographic applications are not well observable in this form. The representation of a Boolean function in the Algebraic Normal Form (ANF) reveals cryptographic information much better. Chapter 1 presents several approaches to

transform a Boolean function into such an ANF explained by comprehensible examples. The ANF is an exclusive-OR of conjunctions consisting only of non-negated variables; hence, this form is also denoted as the positive polarity Reed-Muller form (PPRM).

The netlist of a switching function can be a more compact representation in comparison to the truth table of the associated logic function. Hence, the computation of the coefficients of the ANF of a given netlist of a circuit can be more efficient than the Möbius transform explained in the first chapter. Chapter 2 provides an algorithm that computes all spectral coefficients of the ANF for all logic functions realized by a circuit using a single traversal of all N gates in a time complexity of $O(N)$. The basis for this approach is the representation of the transfer functions of the gates in common Hilbert space. It is noteworthy that this approach facilitates computing the spectral coefficients in an arbitrary order. An algorithm is presented that generates the netlist based on a black box version of a cryptographic switching function, extracts efficiently the associated ANF coefficients and computes finally the degree of the cryptographic primitives.

Boolean functions used as primitives of cryptographic applications must satisfy several properties to be resistant again possible attacks. Unfortunately some of these properties cannot be satisfied by a single function. Bent functions have the wanted property of a maximum non-linearity; however, these functions are not balanced which is another wanted property of Boolean functions for cryptographic applications. Plateaued functions are a class of Boolean functions providing a good trade-off between these properties which makes them useful in cryptographic applications. Chapter 3 proposes a method to construct additional plateaued functions based on a single given plateaued function and a class of permutation matrices derived from factor matrices describing steps in the Cooley-Tukey Fast Fourier Transform (FFT).

It is well known that the number of different assignments to n Boolean variables is equal to 2^n; hence, we are faced with algorithms of an exponential time complexity in the Boolean domain. Therefore, it is an important aim for scientists to find more efficient algorithms for Boolean problems of great practical interest. Part II shows one selected example of high practical importance, where the exponential time complexity has been reduced to a polynomial one.

The verification of a synthesized circuit is an important step within the design procedure. 2^n output patterns must be verified for a combinational circuit with n inputs. One already published result of research was that the correctness of certain types of designed adders can be verified in polynomial time using an approach based on Binary Decision Diagrams (BDDs). However, not only the worst case time complexity but also the coefficients of each term within the polynomial determines the time to solve the real problem. Chapter 4 demonstrates the computation of the coefficients of the polynomials that determine the time needed to verify two different types of adders of an arbitrary number of bits. Experimental results confirm the known worst case theoretical results. It is an impressive result that the verification of carry look-ahead adders to add two integers of 256 bits needs less than half a second and the verification of ripple carry adders of the same size is even faster.

The progress in the Boolean domain contributes to the solution of practical problems of almost all areas. Out of the huge number of applications we selected in Part III an application belonging to the main challenges that human welfare is facing at this time.

Renewable energy sources must replace traditional fossil burnings and large power stations. The energy generated by a large number of relatively small wind farms and photovoltaic installations must be distributed to the locations of the consumers of energy in a manner that satisfies the equilibrium in the distribution net. Chapter 5 explains how distributed agents efficiently solve this combinatorial problem of demand and supply using a lightweight power exchange protocol, ternary vector lists, and operations of XBOOLE. It should be mentioned that this solution, based on approaches of the Boolean domain, is already used in several locations to control electrical sub-nets.

The last part of this book is related to future technologies. The technological effort of reducing the size of gates causes the change from classical logic circuits to quantum circuits. Quantum computing is based on completely different paradigms. The two chapters of Part IV explore these new paradigms stepwise. The approaches to analyze reversible circuits of Chapter 6 are reused in Chapter 7 to analyze quantum circuits. This partition facilitates the understanding of this topic.

One property of quantum circuits is that these circuits realize reversible functions. For that reason the realization of systems of logic functions by reversible logic circuits has been intensively explored. In addition to approaches for the synthesis of reversible circuits, their analysis and verification have also become an important topic of research. Chapter 6 presents three approaches to compute the behavior of a given reversible logic circuit and to compare the computed behavior with the expected behavior. It is shown that problem programs executable by the XBOOLE-monitor XBM 2 solve this task very efficiently. Background information about reversible functions, reversible gates, properties and the XBOOLE functions used as well as hints for the download and use of the XBOOLE-monitor XBM 2 increase the understandability of this and the subsequent chapter.

The behaviors of both a quantum circuit and the gates used are basically described by unitary matrices. Such a matrix of a quantum circuit of n qubits consists of 2^n row and 2^n columns of complex numbers. At first glance we are leaving the Boolean domain when we are dealing with quantum circuits. A more detailed analysis shows that the occurring complex numbers specify, due to the gates used, only a finite number of positions in the complex plane. Due to this finite number an encoding of all required complex numbers by Boolean values can be defined. Chapter 7 determines such an encoding and demonstrates the analysis of the behavior of a quantum circuit in the Boolean domain. Logic values, superposed quantum states, rotations, and possible entanglements are easily computed and shown for the behaviors between the inputs and the cut positions after each gate. Using these insights is was possible to remove two gates from the explored quantum circuit without changing the global behavior.

Bernd Steinbach

Department of Computer Science
Freiberg University of Mining and Technology
Freiberg, Saxony, Germany
April 2022

Acronyms

ABL	Arithmetic Bit-Level
ALU	Arithmetic Logic Unit
ANF	Algebraic Normal Form
BDC	Boolean differential calculus
BDD	Binary Decision Diagram
BLIF	Berkeley Logic Interchange Format
BMD	Binary Moment Diagram
***BMD**	Multiplicative Binary Moment Diagram
CNF	Conjunctive Normal Form
CNI	Critical National Infrastructure
CNOT	Controlled NOT gate
CCNOT	Controlled NOT gate with two control lines
C^{n}NOT	Controlled NOT gate with n control lines
COHDA	Combinatorial Optimization Heuristic for Distributed Agents
CUDA	Compute Unified Device Architecture
D	Disjunctive form
DD	Decision Diagram
DER	Distributed Energy Resources
DFT	Discrete Fourier Transform
EDA	Electronic Design Automation
ESOP	Exclusive-OR Sum-Of-Products
EXOR	Exclusive-OR
FDD	Functional Decision Diagram
FFT	Fast Fourier Transform
FWT	Fast Walsh Transform
GCD	Greatest Common Divisor
GPU	Graphical Processing Unit
HDL	Hardware Description Language
ICT	Information and Communication Technology
IEEE	Institute of Electrical and Electronics Engineers
ISAAC	Intelligent Self-orgAnizing Aggregator and Controller

ISCAS	International Conference on Circuits And Systems
ITE	If-Then-Else
IWLS	International Workshop on Logic Synthesis
K*BMD	Kronecker Multiplicative Binary Moment Diagram
LPEP	Lightweight Power Exchange Protocol
LUT	Look Up Table
MAS	Multi-Agent System
MIS	Multilevel (Logic) Interactive Synthesis (System)
NP	Nondeterministic Polynomial
OBDD	Ordered Binary Decision Diagram
ODA	Orthogonal Disjunctive or Antivalence form
PHL	PHase List
PLA	Programmable Logic Array
PMU	Phase Measurement Unit
POM	Product-Of-Maxterms
PPRM	Positive-Polarity Reed-Muller
PRNG	Pseudo-Random Number Generator
PRP	PRoblem Program
PV	Photovoltaic
RM	Reed-Muller
ROBDD	Reduced Ordered Binary Decision Diagram
RTL	Register Transfer Level
RV	Random Variable
SAT	Satisfiability
SCA	Symbolic Computer Algebra
SOM	Sum-Of-Minterms
TEPCO	Tokyo Electric Power Company
TV	Ternary Vector
TVL	Ternary Vector List
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VT	Variable Tuple
XBM	XBOOLE Monitor

Cyber Security

1. Computing the Reed-Muller Spectrum / Algebraic Normal Form: Functional Methods

Mitchell A. Thornton David K. Houngninou
D. Michael Miller

1.1. Introduction

Researchers and Electronic Design Automation (EDA) tool developers use Reed-Muller (RM) spectra for various applications including synthesis and verification. Separately, the cryptographic community has interest in the Algebraic Normal Form (ANF) of switching functions that comprise primitive building blocks present in various different cryptographic applications. These two forms are connected. In particular, the ANF coefficients of a switching function denoted $a_i \in \mathbb{B}$ where $\mathbb{B} = \{0, 1\}$ are known as the Positive-Polarity Reed-Muller (PPRM) spectral coefficients among switching theorists.

Cryptanalysts are interested in the algebraic degree of switching functions because the degree is a measure of linearity. For example, bent functions have a high degree, whereas linear functions have a degree of unity. The ANF/PPRM spectrum provides one with the degree of a switching function of interest.

Cryptographic primitives serve as the building blocks of larger cryptographic systems [15, 26]. Such primitives include Pseudo-Random Number Generators (PRNGs), cryptographic hash functions, substitution boxes (S-boxes), permutation boxes (P-boxes), and others. De-

pending upon the application of the primitive within a larger cryptographic system, various properties are of interest. For example, [26], a cryptographic hash should approximate the properties of a one-way function to prevent pre-image, attacks and it should be very difficult to find two different strings that produce the same hash value (i.e., cause a collision) to prevent attacks such as birthday attacks [15, 26].

A class of attacks known as algebraic cube attacks [5] is based upon the exploitation of switching functions with a low algebraic degree. More generally, property testing such as cube testing can be used to detect structure (non-randomness) in switching functions and may be used to devise tests for balance, linearity, the presence of linear or neutral variables, and others [1, 16].

For the case of cryptographic applications, binary-valued Boolean, or switching function, properties such as 0-1 balance, nonlinearity, and others are commonly considered and measured for use as a qualifying metric [4, 22]. Secure hash functions should have these properties as well as consistency with the strict avalanche criterion, collision resistance, and resistance to correlation attacks [4]. Switching functions with these properties are often characterized as being *cryptographically strong.*

While the theory is rich and provides many important concepts, it is typically the case that a closed form representation of a switching function is required for these types of analyses to be applied. This motivates one to devise and use sample sets and statistical methods for analysis when the underlying switching functions are unknown or too large for direct analysis to be practical [10].

The ANF allows for direct observation of the algebraic degree of a switching function. Switching functions that characterize a cryptographic primitive whose primary purpose is to generate confusion should have a high algebraic degree since low-degree functions are susceptible to attack. For example, in the case of block ciphers, low-degree switching functions are susceptible to higher order differential attacks [17, 18].

Likewise, low-degree switching functions representing a stream cipher allow the actual functions f_{true} to be approximated as linear functions

denoted here as f_{lin}. Under a linear function approximation, f_{lin} and the corresponding affine function, f_{aff}, can be computed or vice versa. The affine function is $f_{aff} = 1 \oplus f_{lin}$ and likewise, $f_{lin} = 1 \oplus f_{aff}$. Given the approximations of f_{lin} and f_{aff} as an estimate of a low-degree function of interest, f_{true}; a so-called *fast correlation attack* can be successfully applied if the degree of the function f_{true} is sufficiently small since the approximations can be viewed as errors in transmission and an error-correcting code is applied to recover f_{true} [21]. Another important class of attacks known as algebraic cube attacks exploits low-degree switching functions [5].

Hash functions may be modeled as collections of switching functions where the collection is parameterized by the size of the input message. In each of these examples, knowledge of the ANF of the modeled switching functions is of high interest, and thus there is a motivation for enabling the efficient computation of the ANF or PPRM.

While high-degree functions are advantageous in several cryptographic applications, there are tradeoffs in maximizing the degree. One example is a cryptographic switching function that models or implements a stream cipher. It may be desirable for the observed output of such a cipher to resemble a pseudorandom sequence to render it indistinguishable from a random bit sequence. A pseudorandom sequence is very difficult to distinguish from a binary random walk wherein each bit appears to be the result of an independent Random Variable (RV) that is Bernoulli distributed.

A consequence of pseudo-randomness is that the expected number of observations of $f = 1$, denoted as N_1, is approximately equal to the expected number of observations of $f = 0$, likewise denoted as N_0, when the total number of observations, $N_{tot} = N_0 + N_1$, grows without bound. When $N_0 = N_1$, the switching function is said to be balanced. Generally, linear or low-degree functions are balanced since balanced higher degree functions are rare and difficult to find. This is a contradictory constraint to that of maximizing the degree.

Another application of the ANF is in the general area of property testing [1, 16]. Property testing, such as cube testing, can be used to detect structure (non-randomness) in switching functions and may be used to test balance, linearity, the presence of linear or neutral

variables, and others. For example, a χ^2 goodness-of-fit test can be used to test for the distribution of a collection of ANF coefficients since an ensemble of randomly selected switching functions should have, collectively, binomially-distributed ANF coefficients [10].

For at least these reasons, it is desirable to extract or formulate the ANF for a switching function of interest. This chapter describes methods that allow for ANF/PPRM computation to be accomplished with decreased complexity. The techniques apply to a variety of function representations. ANF/PPRM computation directly from a circuit netlist is considered in Chapter 2.

1.2. Background

Switching functions are of the form $f : \mathbb{B}^n \rightarrow \mathbb{B}$, where $\mathbb{B} = \{0, 1\}$ and n is a positive integer representing the number of dependent variables of f. A switching function can be expressed in a variety of normal forms. Normal forms are canonical because any unique, fully specified switching function has one and only one representation for a particular normal form. Due to this canonicity property, normal forms are convenient for use in equivalence proofs and other common design and analysis tasks.

Two very well-known canonical forms are the Sum-Of-Minterms (SOM) and Product-Of-Maxterms (POM). The SOM form is ubiquitous in the digital design community and can be expressed in a variety of ways, including as a list of minterms, a *truth table*, a symbolic expression, or a cube list.

For example, a switching function with $n = 3$ can be written as shown in (1.1) where $m_i \in \mathbb{B}$. This can be generalized to any n as shown in (1.2) where $\bigvee$ denotes the inclusive-OR of the terms that are conjunctions (AND) in which each variable appears either negated or non-negated.

$$\begin{aligned} f = {} & m_0\overline{x}_1\overline{x}_2\overline{x}_3 \vee m_1\overline{x}_1\overline{x}_2 x_3 \vee m_2\overline{x}_1 x_2\overline{x}_3 \vee m_3\overline{x}_1 x_2 x_3 \\ & \vee m_4 x_1\overline{x}_2\overline{x}_3 \vee m_5 x_1\overline{x}_2 x_3 \vee m_6 x_1 x_2\overline{x}_3 \vee m_7 x_1 x_2 x_3 \end{aligned} \tag{1.1}$$

$$f = \bigvee_{i=0}^{2^n-1} m_i \dot{x}_1 \dot{x}_2 \ldots \dot{x}_n \tag{1.2}$$

Another normal form is the Algebraic Normal Form (ANF) which, for n=3, takes the form shown in (1.3) where $a_i \in \mathbb{B}$ and $\oplus$ denotes the exclusive-OR.

$$\begin{aligned} f = {} & a_0(1) \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_3 \oplus a_{12} x_1 x_2 \\ & \oplus a_{13} x_1 x_3 \oplus a_{23} x_2 x_3 \oplus a_{123} x_1 x_2 x_3 \end{aligned} \tag{1.3}$$

The general symbolic form for the ANF/PPRM representation of a switching function is given in (1.4) where p_i represents a conjunctive term composed of $0 \leq k \leq n$ positive-polarity literals. Because the degree k varies for each conjunctive term or monomial, p_i is characterized by its degree. Examination of the form of (1.3) illustrates that each product term p_i in (1.4) contains a number of distinct literals equal to the degree k where k is binomially distributed.

$$f = \bigoplus_{i=0}^{2^n-1} a_i p_i \tag{1.4}$$

The ANF illustrated in (1.3) and (1.4) is most commonly known to the switching theory community as the positive polarity Reed-Muller expansion [25] and the set of a_i coefficients is referred to as the Reed-Muller spectrum of f.

One can transform from SOM to ANF as follows:

$$\mathbf{a}_n = \mathbf{R}_n \mathbf{m}_n \ . \tag{1.5}$$

where $\mathbf{m}_n = [m_0, m_1, ..., m_{2^n-1}]^t$, $\mathbf{a}_n = [a_0, a_1, ..., a_{2^n-1}]^t$ and $\mathbf{R}_n$ is a transformation matrix. For this computation, modulo-2 addition is used and multiplication is performed by the conjunctive logical-AND operation. In other words, all operations are over the Galois field GF_2, also denoted as $\mathbb{F}_2$.

It can be shown [31] that

$$\mathbf{R}_n = \bigotimes_{i=1}^{n} \mathbf{R}_1 \ , \ \mathbf{R}_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{1.6}$$

where $\bigotimes$ is the Kronecker product.

For example, for $n = 3$:

$$\mathbf{R}_3 = \mathbf{R}_1 \otimes \mathbf{R}_1 \otimes \mathbf{R}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \quad (1.7)$$

1.3. Switching Function Representations

Methods for the computation of ANF/PPRM depend upon the form in which the switching function is represented. For example, the straightforward transformation shown in (1.5) has several computational issues that cause it to be impractical for all but the smallest of switching functions. For this reason, several different methods have been formulated that allow for the calculation of the ANF/PPRM coefficients as an entire set or as single values or subsets.

In this section, we provide a brief survey of representations of switching functions. The following section then describes methods for computing the entire or the partial ANF/PPRM for a switching function for these various representations.

1.3.1. Classical Representations

Switching functions can be represented in various forms, including sets, tables, graphs, symbolic, and textual objects. Furthermore, there are multiple variations of each of these representations.

The SOM and ANF were introduced in Section 1.2. In terms of tabular forms, popular representations include the well-known truth table and