Spectral techniques : algorithms, analyses and applications

Yan, Shixing

2007

Yan, S. (2007). Spectral techniques : algorithms, analyses and applications. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/3489>

<https://doi.org/10.32657/10356/3489>

Nanyang Technological University

Downloaded on 26 Apr 2023 18:03:03 SGT

Spectral Techniques: Algorithms, Analyses and Applications

Yan Shixing

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2007

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research done by me and has not been submitted for a higher degree to any other University or Institute.

.....

Date

.....

Yan Shixing

To *Hui Ying*

Acknowledgements

First and foremost, I would like to express my earnest appreciation to my advisor, Dr. Bogdan J. Falkowski, for his professional guidance, extensive support and continuous encouragement throughout the past three years. I appreciate what I have learned from him, not only the broad knowledge, sharp scientific and profound insight on research issues, but also the optimistic and confident personalities on all aspects of this voyage.

Special thanks to my fellow labmates in Information System Research Lab for making it a fun and friendly working environment. Appreciation is also extended to all my friends for their friendship during my graduate studies at Nanyang Technological University.

Last but not least, I would like to thank my wife Hui Ying and our parents whose love and support have always been the greatest inspiration for me. I am forever grateful to my grandmother who passed away during my second year of study. She was close to me in heart even in her last hours of life while I was thousands miles away. I know her love is always with me.

This research work received both the financial and technical support from School of Electrical & Electronic Engineering, Nanyang Technological University, which are greatly appreciated.

Summary

Spectral techniques which are a mathematical discipline in abstract harmonic analysis have been widely used in many applications including logic design and optimization and digital signal processing. This dissertation is devoted to the developments of the new algorithms, analyses and applications of spectral techniques.

The analysis of the mutual relationships between Hadamard-Haar and arithmetic transforms and their corresponding spectra in the form of matrix decomposition as layered vertical and horizontal Kronecker matrices are presented. The fast algorithms and butterfly diagrams for these relations have been developed to make the transformations more efficient. The relationships in arithmetic, Reed-Muller and Haar spectral domains of the Boolean operations in functional domain have been revealed. Based on the convolution theorems proposed for the arithmetic, Reed-Muller and Haar transforms, the new algorithms for the efficient calculations of the spectra of composite Boolean functions are developed.

A fixed sign Walsh hardware structure is introduced which can be easily implemented based on the fixed butterfly diagrams. The modified sign Haar transform with in-place structure is also developed as well as its prop-

erties for logic functions. The sign Hadamard-Haar transform is designed based on the combination of Hadamard and Haar butterfly diagrams. Furthermore, the sign Hadamard- γ -Haar- χ transform is developed based on the rationalized Hadamard-Haar transform while the sign Walsh, sign Haar and sign Hadamard-Haar transforms belong to our novel sign Hadamard- γ -Haar- χ transform as each of them is just one possible case of the new transform.

The multi-polarity arithmetic-Walsh transform and multi-polarity arithmetic-Haar transform are introduced with their corresponding fast algorithms and spectral transform decision diagrams. The hybrid multi-polarity arithmetic-Walsh transform combines the useful properties of both multi-polarity Walsh and multi-polarity arithmetic transforms. Both new transforms are in multi-polarity and provide the optimal expression for a given logic function.

A new ternary coding scheme named T coding is designed for the representation of Boolean functions. T coding which utilizes the GF(3) algebra is the first coding scheme that can provide a unified algebra for the Boolean logic with the consideration of don't care (DC) conditions. The ternary Walsh transform performed over GF(3) is developed which requires simpler arithmetic and less storage space than standard Walsh transform. The ternary Walsh transform is suited for processing both completely and incompletely specified Boolean functions in T coding while the other spectral transforms related to standard Walsh transform cannot handle the incompletely specified Boolean functions efficiently. Based on the properties for Boolean functions of the ternary Walsh transforms, the decomposition and symmetry detection of completely and incompletely specified Boolean functions using ternary Walsh transform are investigated.

Contents

Acknowledgements	i
Summary	ii
List of Figures	xiv
List of Tables	xvi
1 Introduction	1
1.1 Literature Review of Spectral Techniques	1
1.1.1 Orthogonal Spectral Transforms	2
1.1.2 Arithmetic and Reed-Muller Transforms	6
1.1.3 Sign Transforms	8
1.1.4 Spectral Transform Decision Diagrams	9
1.2 Outline and Contribution	10
2 Boolean Functions and Classical Spectral Transforms	15
2.1 Boolean Functions and Their Representations	16
2.1.1 Functions	16

2.1.2	Boolean Algebra	17
2.1.3	Boolean Function	18
2.1.4	Binary Decision Diagram	20
2.2	Classical Spectral Transforms	21
2.2.1	Kronecker Product of Matrices	22
2.2.2	Walsh Transform	23
2.2.3	Haar Transform	25
2.2.4	Hadamard-Haar Transform	26
2.2.5	Reed-Muller Transform	27
2.2.6	Arithmetic Transform	28
3	Mutual Relationships between Hadamard-Haar and Arithmetic Transforms	30
3.1	Introduction	30
3.2	Transformations between Hadamard-Haar and Arithmetic Spectra	31
3.3	Fast Algorithms	45
3.4	Analysis of Running Time of Algorithms	53
3.5	Summary	58
4	Relationships of Boolean Operations in Spectral Domains	59
4.1	Introduction	59
4.2	Relationships of Boolean Operations in Arithmetic and Reed-Muller Domains	60
4.2.1	Preliminaries	60

4.2.2	Calculation of Composite Spectra of Arithmetic and Reed-Muller Transforms	63
4.2.3	Analysis on the Complexities of Algorithms	77
4.3	Calculation of Composite Spectra of Haar Transform	82
4.4	Summary	92
5	Developments for Sign Transforms	93
5.1	Introduction	93
5.2	Review of Sign Walsh and Sign Haar Transforms	94
5.3	Fixed Sign Walsh Transform and Its Iterative Hardware Architecture	97
5.3.1	General Definitions of Quantized Transforms	97
5.3.2	Fixed Fast Walsh Transform Structure	98
5.3.3	Definition of Fixed Sign Walsh Transform	99
5.3.4	Iterative Architecture for Fixed Sign Walsh Transform	100
5.4	Modified Sign Haar Transform	104
5.4.1	Definitions of Modified Sign Haar Transform	104
5.4.2	Analysis of Butterfly Structure	105
5.4.3	Computational Costs	107
5.4.4	Application of Modified Sign Haar Transform in Logic Functions	108
5.5	Sign Hadamard-Haar Transform	111
5.5.1	Definitions of Sign Hadamard-Haar Transform	112

5.5.2	Properties of Sign Hadamard-Haar Spectra of Logic Functions and Variables	117
5.5.3	Calculation of Sign Hadamard-Haar Transform by Matrix Approach	121
5.5.4	Computational Costs	123
5.6	Sign Hadamard- γ -Haar- χ Transform	124
5.6.1	Definitions of Sign Hadamard- γ -Haar- χ Transform	124
5.6.2	Computational Costs	126
5.7	Summary	128
6	Hybrid Multi-Polarity Arithmetic-Walsh Transform	130
6.1	Introduction	130
6.2	Definitions of Multi-Polarity Arithmetic-Walsh Transform	131
6.3	Fast Algorithms	136
6.4	Analysis of Computational Costs	141
6.5	Arithmetic-Walsh Spectral Transform Decision Diagrams	143
6.6	Experimental Results	149
6.7	Summary	150
7	Multi-Polarity Arithmetic-Haar Transform	152
7.1	Introduction	152
7.2	Definitions of Multi-Polarity Arithmetic-Haar Transform	153
7.3	Fast Algorithms and Computational Costs	158
7.4	Calculation through Decision Diagrams	169

7.5	Arithmetic-Haar Spectral Transform Decision Diagrams	170
7.6	Experimental Results	176
7.7	Summary	177
8	Ternary Walsh Transform	182
8.1	Introduction	182
8.2	Definitions of Ternary Walsh Transform	183
8.2.1	Operations over $GF(3)$	183
8.2.2	Definitions and Properties of Ternary Walsh Transform .	184
8.3	Fast Algorithms	197
8.4	Properties of Ternary Walsh Spectrum for Boolean Functions . .	201
8.5	The Decomposition of Boolean Functions	212
8.6	Symmetry Detection of Boolean Functions	216
8.6.1	Symmetry Detection of Completely Specified Boolean Func- tions	217
8.6.2	Symmetry Detection of Incompletely Specified Boolean Functions	219
8.7	Ternary Walsh Spectral Transform Decision Diagrams	224
8.8	Experimental Results	228
8.9	Summary	230
9	Conclusion and Future Work	235
9.1	Conclusion	235
9.2	Future Work	240

Contents

ix

Author's Publications	242
Bibliography	255

List of Figures

2.1	Binary decision tree and BDD for the function $f = ab \vee c$	21
3.1	The flow diagrams for the transformation from non-normalized Hadamard-Haar spectrum to arithmetic spectrum using proposed method where $r = 1$ and $n = 3$	51
3.2	The flow diagrams for the transformation from non-normalized Hadamard-Haar spectrum to arithmetic spectrum using conventional method where $r = 1$ and $n = 3$	52
3.3	The reduction in running time when using proposed method for the transformation from non-normalized Hadamard-Haar spectrum to arithmetic spectrum for some cases.	55
3.4	The reduction in running time when using proposed method for the transformation from non-normalized Hadamard-Haar spectrum to arithmetic spectrum.	56
3.5	The reduction in running time when using proposed method for the transformation from arithmetic spectrum to non-normalized Hadamard-Haar spectrum.	57

4.1	The block diagrams of the procedures for both conventional method and the proposed method to calculate arithmetic or Reed-Muller spectrum.	78
4.2	Number of cases in all 4-variable Boolean functions for different values of Z_f (for the calculation of Reed-Muller spectrum). . . .	79
4.3	Percentage of the cases in all 4-variable Boolean functions (for the calculation of Reed-Muller spectrum using proposed method). . .	80
4.4	Percentage of the cases in all 4-variable Boolean functions (for the calculation of arithmetic spectrum using proposed method). . .	81
4.5	The block diagram of the calculation of arithmetic or Reed-Muller spectrum using conventional method.	82
4.6	The block diagram of the calculation of arithmetic or Reed-Muller spectrum using proposed method.	82
4.7	The block diagrams of the procedure for both conventional method and the proposed method to calculate Haar spectrum.	88
5.1	Butterfly diagrams for the forward and inverse sign Walsh transforms where $n = 3$	96
5.2	Butterfly diagrams for the forward and inverse sign Haar transforms where $n = 3$	97
5.3	Butterfly diagrams for the forward and inverse fixed sign Walsh transforms where $n = 3$	101
5.4	Iterative architecture for the forward fixed sign Walsh transform where $n = 3$	102
5.5	Iterative architecture for the inverse fixed sign Walsh transform where $n = 3$	103

5.6	Butterfly diagrams for the forward and inverse modified sign Haar transforms where $n = 3$	105
5.7	Butterfly diagrams for the forward and inverse modified sign Haar transforms where $n = 4$	106
5.8	Karnaugh and sign domain maps for S-coded functions f_1 , f_2 and f_3	118
5.9	Butterfly diagrams for the forward and inverse sign Hadamard-Haar transforms where $n = 3$	123
5.10	Fast butterfly diagrams for the forward and inverse sign Hadamard-2-Haar-2 transforms.	127
6.1	Fast butterfly diagrams for the forward and inverse multi-polarity arithmetic-Walsh transforms where $r = 2$, $n = 3$, and $\vec{\omega}_3 = \{\omega_1, \omega_2, \omega_3\} = \{0, 1, 1\}$	141
6.2	Arithmetic-Walsh spectral transform decision tree node at level i : (a) general form, (b) $1 \leq i \leq r$, (c) $r < i \leq n$	144
6.3	Arithmetic-Walsh spectral transform decision tree of function f_1 where $r = 2$, $n = 3$, and $\omega_3 = \{\omega_1, \omega_2, \omega_3\} = \{0, 1, 1\}$	146
6.4	Elimination rule of arithmetic-Walsh STDDs.	146
6.5	Merging rule of arithmetic-Walsh STDDs.	147
6.6	Divisor rule of arithmetic-Walsh STDDs.	147
6.7	Reduced arithmetic-Walsh STDD of f_1	148
7.1	Fast butterfly diagrams for the forward and inverse arithmetic-Haar transforms, $\omega_3 = \{0, 1, 1\}$, $r = 2$ and $n = 3$	165

7.2	Another fast butterfly diagrams for the forward and inverse arithmetic-Haar transforms, $\omega_3 = \{0, 1, 1\}$, $r = 2$ and $n = 3$	167
7.3	Fast butterfly diagrams for the forward and inverse Walsh transforms where $n = 3$	168
7.4	Reduction in computational costs of arithmetic-Haar transform comparing with Walsh and arithmetic transforms.	170
7.5	Calculation of multi-polarity arithmetic-Haar coefficients through DT.	171
7.6	Calculation of multi-polarity arithmetic-Haar coefficients for f_1 through DD.	171
7.7	Arithmetic-Haar spectral transform decision tree node at level i : (a) general form, (b) $i = 1$, (c) $2 \leq i \leq n$ and TR in TR_L , (d) $2 \leq i \leq n$ and TR in TR_R	173
7.8	Arithmetic-Haar spectral transform decision tree of function f_1 where $r = 2$, $n = 3$, and $\omega_3 = \{\omega_1, \omega_2, \omega_3\} = \{0, 1, 1\}$	174
7.9	Reduced arithmetic-Haar STDD of f_1	175
8.1	Computations of the ternary Walsh transform where $n = 3$	186
8.2	Diagram of the ternary Walsh transform where $n = 3$	197
8.3	Butterfly diagrams of the forward and inverse fixed fast ternary Walsh transforms where $n = 3$	199
8.4	Iterative structure of the fixed ternary Walsh transform where $n = 3$	200
8.5	TWSTDT node at level i	225
8.6	TWSTDT for the 3-variable function.	226

8.7	Elimination rule for TWSTDD.	227
8.8	Merging rule for TWSTDD.	227
8.9	TWSTDD for the 3-variable function in Example 8.7.1.	228
8.10	RTWSTDD for the 3-variable function in Example 8.7.1.	229

List of Tables

3.1	Comparison of Computational Costs using Conventional and Proposed Methods	54
5.1	Relationship between 16-Point Modified and Standard Sign Haar Spectra	107
5.2	Comparison of Computational Costs for Modified Sign Haar Transform and Sign Walsh Transform	108
5.3	Comparison of Computational Costs for Sign Hadamard- γ -Haar- χ Transform and Sign Walsh Transform	128
6.1	Comparison of Computational Costs of Multi-Polarity Arithmetic-Walsh and Multi-Polarity Walsh Transforms	142
6.2	Comparison of Number of Non-Zero Coefficients for Arithmetic-Walsh transform	151
7.1	Comparison of Computational Costs for Arithmetic-Haar Transform	169
7.2	Comparison of Number Non-Zero Coefficients for Arithmetic-Haar Transform.	179

8.1	Forward Ternary Operations $\oplus_3$ and $\ominus_3$	184
8.2	Inverse Ternary Operations $\boxplus_3$ and $\boxminus_3$	185
8.3	Multiplication Rules over $\text{GF}(3)$	185
8.4	Modulo-2 Inner Products $\langle i, j \rangle$ when $n = 3$	193
8.5	Rules of Ternary Coding Converter	198
8.6	Logic Operations in T Coding	202
8.7	Corresponding Relations between Ternary Walsh Coefficients and Variables for $n = 3$	204
8.8	Ternary Walsh Spectra of Logic Gates	208
8.9	Rules of MEXOR Operation in T Coding	220
8.10	Comparison of Number of Non-Zero Coefficients for Ternary Walsh Transform.	232

Chapter 1

Introduction

1.1 Literature Review of Spectral Techniques

In many applications of computer science and engineering, where logic functions need to be analyzed or synthesized, it is useful to transform such functions to the corresponding spectral domains that provide various new insights into solution to some important problems. The mathematics foundations of the spectral techniques go back to the study of the Fourier analysis which is based on the Fourier transforms [Boi81, Fol92]. The discrete Fourier transform (DFT) and its fast algorithms known as fast Fourier transform (FFT) were developed later and has been widely used in many different fields of science and technology where the digital signal is involved [Bri88, ER82]. The study of orthogonal functions, such as discrete Walsh functions and Haar functions arised in the research on the orthogonal systems [Haa10, Wal23, RNR75a]. The orthogonal

discrete transforms including discrete Haar, Walsh, Hadamard-Haar and other related transforms have been used in data transmission, digital signal and image processing, statistical analysis, solving differential equations, and the analysis, synthesis and classification and testing of logical circuits in their early applications [Aga80, AR75, Bea75, Bea84, Bes78, Col61, Der65, Edw75, EH78, ER82, FTY87, Har69, HMM85, HS84, Hur77, Hur78, Hur81, Kar76, Kar85, Lec71, LZ83, Mil87, Mor78, Muz80, Ohn85, RJJ77, RJJ80, RNR75a, RNR75b, Swi69, Yar85, Zal89]. There has been increasing great interest in spectral techniques for digital logic design in recent years, since modern computer-aided design (CAD) tools for digital circuits frequently use different graphical representations of Boolean functions with standard Boolean operators where the spectral transforms including Walsh, Haar, arithmetic and Reed-Muller transforms are suited for the applications [AAE95, Bla98, BSS99, BGG98, Cas96, CMZ⁺93, CMZ⁺97, Dam92, Fal99b, FK99, FK00, FSP92, GES91, Hei91, Hur92, KSA00, KSA03, OLRPP93, RF98, RF99, RMB92, RZ01, RZ04, Sas93, Sas99, Sch98, SF96, SF03, SS01, SSJ98, SSS96, Sta98, TDM01, TMS96, TMS97, TN95, YM91].

1.1.1 Orthogonal Spectral Transforms

Owing to the fact that Walsh and non-normalized Haar orthogonal functions assume values $+1$, 0 , and -1 only, the computation of the transforms requires arithmetic addition and subtraction. Hence, they require the most economical computational costs and have been used efficiently in applications involving two or more valued signals, situations that happen in digital signal processing of logical signals and in the design of multirate digital signal processing systems [AAE95, Bea84, GES91, Har69, HMM85, Kar76, Yar85, Zal89]. Walsh

functions are also very easy to implement on high-speed computers as they are binary related and they operate only on two numbers $(+1, -1)$ and can be used with very little storage space. Fast algorithms for Walsh and Haar transforms based on sparse matrix factorization have been developed, which reduce the computational costs for the corresponding calculations.

Discrete Haar transform is the simplest example of wavelet expansion and attracts much attention in engineering practice for its peculiar properties [BGG98, Cas96, FC97a, FC98, HS96, HS97, Hur81, Kar76, SF96, Tho97, VK95, Zal89]. Due to its low computing requirements, the Haar transform has been mainly used for pattern recognition and image processing [Cas96, Moh92, Yar85, Zal89]. Hence, two dimensional signal and image processing is an area of efficient applications of Haar transforms due to their wavelet-like structure. In this area, it is usually reported that the simplest possible orthogonal wavelet system is generated from the Haar scaling function and wavelet. Moreover, wavelets are considered as a generalization of the Haar functions and transforms [Bea84, BGG98, VK95]. Such a transform is also well suited in communication technology for data coding, multiplexing and digital filtering [Kul84, Sho73, Zal89]. For example, application of non-normalized Haar transform in a sequency division multiplex system is described in [Zha89]. For real time applications, hardware-based fast Haar chips have been developed [BMS88, Zal89]. In [Kul84], different generalizations of Haar functions and transform are used in digital speech processing with applications in voice controlled computing devices and robotics. The control system based on Haar spectrum for military airplane is also discussed in [Boi81]. The applications of Haar transform in control and communications are presented in [Kul84, Zal89, Zha89, ZZ93]. In [Ohn85], different forms of Haar functions are used in approximate calculations of analytic func-

tions. A brief discussion of various other applications, where the use of Haar and Walsh functions offers some advantages compared to the Fourier transform, is given in [Zal89]. The advantages of computational and memory requirements of the Haar transform make it of a considerable interest to VLSI designers as well. For example, the authors of [RMB88], presented a set of CAD tools to perform a switch-level fault detection and diagnosis of physical faults for practical MOS digital circuits using a reduced Haar spectrum analysis. In their system, the non-normalized Haar spectrum was used as a mean not only for diagnosing digital MOS ICs as a tool external to the circuit, but also as a possibility for a self-test strategy. The use of this set of CAD tools allowed the derivation of strategies for testing MOS circuits when memory states were encountered as a consequence of some type of faults.

The advantage of using Haar functions instead of Walsh functions in CAD system based on spectra methods for some classes of Boolean functions was shown in [Kar76, Zal89]. For example, the analysis in [Kar76] shows that the spectral complexity of conjunction and disjunction increases with the number of variables, exponentially for the Walsh functions and only linearly for the Haar functions. The circuit of the spectral multiplication logic module based on Haar functions was also developed [Kar76, Kul84]. It consists of a generator of basis functions, an adder, a multiplier, and the memory to store spectral coefficients. The module can be reprogrammed by changing dynamically its memory content. Such a behaviour of the module is useful in real-time adaptive control systems [Kul84, Zal89]. Karpovsky [Kar76] noticed that the size of the memory block can be optimized only when the Haar basis is used. It is due to the fact that the number of non-vanishing Haar coefficients is reduced with input permutation of variables - the situation that does not exist for the Walsh

basis. It should be noted that the realization of a permutation requires no special hardware [Kar76]. Another advantage of the Haar spectrum in this and similar applications is the smallest number of required arithmetic operations. In [Tho99], a method for probabilistically determining the equivalence of two switching functions through Haar spectral coefficients has been developed. The method is reported as an alternative for equivalence checking of function that is difficult to represent completely and is based on binary decision diagrams and Haar spectral diagrams [HS97]. As noted in [Lab84], different generalization of spectral methods, including Haar functions, are intended to provide a unified theory for uniform consideration of different tasks in digital signal processing and related areas. Such theories are useful researches in both signal processing and applied mathematics. They bring new methods and tools for solving practical tasks to engineers, and trace and determine versatile and actual directions of research for mathematicians. The multi-polarity Walsh and Haar transforms which are used for the Boolean functions in complemented form have been also developed in [Fal94, Fal02].

Walsh transforms in Gray code ordering was introduced in [Fal95] and different and efficient ways of generating such Walsh transforms have been presented. A number of important properties that characterize Walsh transforms in Gray code ordering were given. The derivations presented can be useful for finding new applications of Walsh transform in Gray code ordering in areas other than logic design.

Another orthogonal transform related to both Walsh and Haar functions is Hadamard-Haar transform including the normalized and non-normalized (rationalized) Hadamard-Haar transforms [RJY77, RJY80, RNR75a, RNR75b]. The

Hadamard-Haar transform is a hybrid version of the Haar and Walsh transforms which can be seen as the two special cases of Hadamard-Haar transform. The Hadamard-Haar transform which has the advantages of Walsh and Haar transforms has been applied in signal processing [DMC97, Dry92].

1.1.2 Arithmetic and Reed-Muller Transforms

The Boolean functions normally are represented with standard Boolean operators. Based on the arithmetic transform [AAE95, Fal99b, JABF92, Mal82, SA03, SF96, SS01, TDM01] which is also known as probabilistic transform [KB81, PM75] or algebraic transform [Sta95], the Boolean functions can be also expressed as arithmetic expansions with symbols ‘+’, ‘-’ and ‘ $\cdot$ ’ for addition, subtraction and multiplication, respectively. Such arithmetic expansions are used in probabilistic verification of a pair of functions and to describe stochastic behaviour of a logical circuit under random values of input variables. Arithmetic expansion was also used for testing of digital circuits [Hei91] and design verification [RZ01, RZ04]. Many important problems in CAD such as synthesis, verification and optimization can be solved very efficiently not in the original Boolean domain but rather through evaluating the corresponding arithmetic expression [SF96]. The sets of basis functions for the arithmetic transform obtained from its inverse transform matrix and known under the name of conjunctive transform [AAE95], or adding transform [Fal99b, FP90a], is the same as the sets of basis functions for the better known Reed-Muller transform [AAE95, DDT78]. As the conjunctive matrix can be easily constructed via Pascal triangle, which was shown for the first time in [Cal61], this transform is sometimes called a Pascal transform [AAE95].

The representation of Boolean functions by using AND, exclusive-OR (EXOR) and constant 1 is another type of expressions for logic circuits design. Reed-Muller transform is an invertible linear transform which utilizes algebra of Galois Field (2) ($\text{GF}(2)$) and any switching function may be completely realized by the modulo-2 sum-of-products expression which is known as the complement-free ring-sum [DDT78]. For many important functions that are nonunate (e.g., parity checkers, adders and multipliers), Reed-Muller realizations are advantageous when area, speed, and testability are of main concern [Sas93]. For such linear functions, they allow surprisingly complex designs to be implemented using very few product terms. Although the Reed-Muller transform has exactly the same matrix form with the adding transform, the 0 and 1 of the entries in those two transforms have different meaning. Since the calculations of the adding transform are performed over the integer number Z while the calculations of Reed-Muller transform are over $\text{GF}(2)$, the elements in the transform matrices of adding transform and Reed-Muller transform are in the integer number Z and $\text{GF}(2)$, respectively. When each variable throughout Reed-Muller expansion assumes either true or complemented form, such an expression is known as fixed polarity Reed-Muller expansion or generalized Reed-Muller expansion. When all variables in the arithmetic or Reed-Muller expansion are nonnegated, it is a zero polarity expansion. When some of the variables are uniformly negated, the corresponding expansion has a polarity not equal to zero. A family of 2^n arithmetic expansions in different polarities is known under the name of generalized arithmetic expansion [FC97b, FP90a]. An optimal arithmetic or Reed-Muller expansion with the smallest number of nonzero coefficients may occur for some polarity not equal to zero. Therefore when looking for efficient representations of Boolean functions in the arithmetic or Reed-Muller expressions, the gener-

alized arithmetic or Reed-Muller expansions should be considered rather than the arithmetic or Reed-Muller expansions in zero polarity. There exists a need for efficient ways of calculating the expansion that has the following properties: ability to evaluate only some chosen coefficients, operations on completely specified Boolean functions, efficient calculation from reduced representation of Boolean functions and efficient data representation for storage of calculated coefficients.

1.1.3 Sign Transforms

Another family of invertible nonlinear transforms, which utilize the sign function as the quantizer, are sign transforms. The main idea of the sign transforms is that the sign function is used as the quantized function and both the functional and spectral domains are operating in $\{+1, 0, -1\}$. The multiple-valued current-mode MOS integrated circuit has been proposed which includes the three-valued ($\{+1, 0, -1\}$) logic [HK95], so the sign transforms can be implemented in hardware by using butterfly diagrams. The first transform under the name of sign transform was based on Walsh functions and is known as Sign Walsh transform [BT96, DiT91, Sas93, SSS96]. An efficient method to calculate sign Walsh spectra from disjoint cube representation of logic functions was shown in [FK02]. The sign Walsh transform has been used in the applications of logic synthesis and testing [BT96, DiT91, Sas93, SSS96]. Based on the Haar functions, the sign Haar transform has been developed in [FR94, FR03]. The important feature of the sign transforms is the considerable reductions in computing time and storage requirements with keeping the original spectral transform's most useful properties.

1.1.4 Spectral Transform Decision Diagrams

Binary decision diagrams (BDDs) as a graphical representations of Boolean functions were introduced and popularized by [Ake78, Mor82]. The efficient methods for the manipulation of BDDs were developed as the concept of ordered binary decision diagrams (OBDDs) [Bry86]. One main advantage of OBDD representation is that it is canonical and remains well bound in size for many important Boolean functions. Many large digital systems have applied OBDDs to verification problems at the logic optimization stages [Bry91, Bry92, SF96]. In addition, the complement and tautology operations required in many heuristic minimization techniques can be performed in OBDDs efficiently [Bry86, SF96]. Many basic gates such as AND, OR, EXOR, etc. as well as more complex logic functions like majority voter exploit the powerful properties of OBDDs to a large extent in terms of both the space and time complexity. The disadvantages of OBDD representation are its sensitivity to the ordering of input variables and its relatively low space efficiency compared with other directed acyclic graph based data structures. The multi-terminal BDDs (MTBDDs) were developed for the representations of functions with integer values [CFM⁺93, CFZ95, CMZ⁺93, CMZ⁺97, FMY97, SF96]. Another canonical and compact structure to represent and manipulate pseudo Boolean functions is edge valued BDDs (EVBDDs) which use both arithmetic and Boolean operations [LPV94, LPV96, SF96]. Spectral transform DDs (STDDs) were developed to represent the spectra of the given logic functions [SA03, SF96, SSJ98]. STDDs is more efficient to calculate the spectral coefficients especially for large functions through STDDs than butterfly diagrams.

1.2 Outline and Contribution

The research work discussed in this dissertation is devoted to the algorithms, analyses and applications of spectral techniques. The relationships for different spectral transforms are analyzed, and new spectral transforms which provide the useful properties in applications are also developed.

Before presenting the contributions of this thesis, **Chapter 2** first provides some basic concepts of Boolean functions and classical spectral transforms.

The mutual relations between various spectral transforms are interesting topics since some transforms are better suited than others depending on the specific applications. The mutual relationships between Walsh, Haar, arithmetic, Reed-Muller transforms have been investigated [Fal99c, Fal05, Fin72, HS97, RZ04, Sta82, TDM01, TMD01]. In **Chapter 3**, the mutual relationships between Hadamard-Haar and arithmetic transforms and their corresponding spectra in the form of matrix decomposition as layered vertical and horizontal Kronecker matrices are discussed together with their proofs, fast algorithms and computational costs. The new relations apply to an arbitrary dimension of the transform matrices and allow performing direct conversions between arithmetic and Hadamard-Haar functions and their corresponding spectra. In addition, analysis of butterfly diagrams for these new relations is also introduced and it is shown that they are more efficient than the matrix decomposition method.

The corresponding relations in spectral domains for the Boolean operations in functional domain are not only important issues in theory, but also practically useful in the evaluation of the composite spectra. Based on the convolution theorem of Walsh and Chrestenson transforms [Kar76], the evaluation of the

Walsh spectra of the composite Boolean functions without reintroducing the Walsh transform has been studied in [Eri78, Muz78, Muz80]. The new methods for the efficient evaluation of the spectra of the logic product, logic sum, EXOR sum and negation of Boolean functions in Haar, arithmetic and Reed-Muller domains are developed in **Chapter 4**. The different properties and equations are presented for the new approaches based on the convolution theorems for various spectral transforms. The new algorithms presented in Chapter 4 have also addressed an important issue of efficient storage of calculated coefficients in the form of the vector. By representing the Boolean function and their corresponding spectral coefficients in the form of small matrices, the coefficients can be computed more rapidly with smaller number of required memory, while the ability to calculate only partial spectra is still preserved. Another advantage of the introduced methods is the extensive use of different properties and special cases of Boolean functions in spectral domains that make the computational time and memory requirements of the algorithms negligible in many instances.

In **Chapter 5**, the definitions of sign Walsh and sign Haar transforms are reviewed first. The rest of Chapter 5 describes the improvements on the sign Walsh and Haar transforms and the developments of the new sign Haar transforms. The fixed sign Walsh hardware structure is introduced which is based on fixed butterfly diagrams that can be easily implemented in hardware using look-up table cascades. The modified sign Haar transform with in-place structure is developed and its relation with sign Haar transform and butterfly structure are analyzed. The properties of logic functions and variables in the spectral domain of the modified sign Haar transform are also presented. A new sign transform named sign Hadamard-Haar transform is developed based on the combination of Hadamard and Haar butterfly diagrams. The sign Hadamard- γ -Haar- χ trans-

form is further developed where the rationalized Hadamard-Haar transform is quantized by using the sign function. In the new recursive sign transform, γ and χ show the number of butterflies in Hadamard and Haar parts of sign Hadamard- γ -Haar- χ transform. The sign Hadamard- γ -Haar- χ transform provides better security properties by its design and is suitable for cryptographic systems since the coefficients γ and χ can be adjusted.

Chapter 6 introduces the new hybrid multi-polarity arithmetic-Walsh transform that combines properties of both multi-polarity Walsh and multi-polarity arithmetic transforms. An optimal arithmetic-Walsh expansion with the smallest number of nonzero coefficients may occur for some polarity not equal to zero. Therefore when looking for efficient representations of discrete functions in the standard arithmetic-Walsh domain, the hybrid arithmetic-Walsh expansions should be considered rather than the arithmetic-Walsh expansion in zero polarity. In some applications of arithmetic-Walsh expansion, the values of only a few coefficients are needed, for example in testing and decomposition. The new multi-polarity arithmetic-Walsh transform and its corresponding arithmetic-Walsh STDD are expected to be used in the areas in which Walsh and arithmetic transforms as well as their decision diagrams have been used, especially for CAD tools in logic design. Its advantages over multi-polarity Walsh and multi-polarity arithmetic transforms are also shown in the experimental results for some MCNC benchmark functions [Yan91].

In [SMA01], an idea of a combined arithmetic-Haar transform was proposed. Such a transform was defined for the first eight functions in zero-polarity and experimental results shown in [SMA01] proved that arithmetic-Haar transform is more efficient than other transforms in logic design such as Walsh, Haar and

arithmetic transforms for some benchmark functions. Therefore it is interesting not only theoretically but also practically, to extend this arithmetic-Haar transform for higher matrix dimensions and from zero-polarity to multi-polarity. Based on the advantages of multi-polarity arithmetic and multi-polarity Haar transforms over their zero-polarity counterparts, the generalization of multi-polarity arithmetic-Haar transform for higher dimensions are introduced in **Chapter 7**. The new equations, fast transforms, and computational costs of this transform as well as experimental results are also presented. The new kind of STDD corresponding to the multi-polarity arithmetic-Haar transform is also proposed to represent the arithmetic-Haar spectra of discrete functions.

Chapter 8 is motivated by the interest of reducing the computational costs, computation arithmetic and required storage for the standard Walsh transform. We developed the novel ternary Walsh transform which utilizes the $GF(3)$ arithmetic to provide a simpler arithmetic and require less storage space for the calculations than standard Walsh transform. The ternary Walsh transform which can be seen as the $GF(3)$ -version of Walsh transform has many useful properties which are suitable for processing both the completely and incompletely specified Boolean functions. The same hardware implementation based on the fixed fast ternary Walsh transform can be used for both forward and inverse ternary Walsh transforms based on its fast algorithms and properties. A new coding scheme named T coding is designed for the representation of both the completely and incompletely specified Boolean functions. The T coding is the first coding scheme which can represent the Boolean algebra including the consideration of don't care (DC) conditions by using the calculations over $GF(3)$. The applications of the ternary Walsh transform on the decomposition and symmetry detections of the Boolean functions are shown where the T coding

is used to encode the functions, while in the previous studies, classical spectral transforms can handle only the completely specified Boolean functions. The corresponding STDD based on ternary Walsh transform is also developed for the representation of the ternary Walsh spectrum of the Boolean function.

Chapter 9 rounds off the dissertation with a conclusion of the work and possible future research directions.

Chapter 2

Boolean Functions and Classical Spectral Transforms

In this chapter, the basic concepts of the Boolean functions and classical spectral transforms are reviewed. Some basic mathematic concepts used throughout the remainder of this dissertation will be also introduced. More details about the concepts and discussions in this chapter can be also found in [AAE95, Aga80, AR75, Bea75, Bea84, BGG98, Bla74, Boo54, Bro90, Bry86, Bry92, DDT78, DM94, Dre00, ER82, Fal94, Fal99b, Fal02, FP90a, GES91, Gra81, Haa10, Har69, HR68, HMM85, HS96, HSB02, Hur92, Kar76, Kar85, KB81, Mil87, Moh92, Mor82, RJJ77, RNR75b, SA03, Sas93, Sas99, Sch89, SF96, Sha38, SS01, SSJ98, SSS96, Sta98, Swi69, TDM01, Wol98, Yar85].

2.1 Boolean Functions and Their Representations

2.1.1 Functions

A **set** is a collection of objects which are called **elements**. Let A and B be two sets, the **cartesian product** of A and B , denoted by $A \times B$, is defined to be the set of all ordered pairs whose first element is in A and whose second element is in B . In formula:

$$A \times B = \{(a, b) | a \in A \text{ and } b \in B\}. \quad (2.1)$$

A **binary relation** R between A and B is a subset of $A \times B$, denoted by:

$$aRb \Leftrightarrow (a, b) \in R \subseteq A \times B. \quad (2.2)$$

f is called a **function** from A to B if f is a relation between A and B and for every $a \in A$ there exists a unique element b in B such that afb . The function f is called as a **mapping** from A to B , denoted by $f : A \rightarrow B$. If an element a in A corresponds to an element b in B , we denote it by $f(a) = b$. A is called the **domain** of the function. The **range** of f is defined to be the set of all elements $b \in B$ such that there exists an $a \in A$ for which afb . A function f is called to be from A **onto** B when the range of f is B . If for every a_1fb and a_2fb implies $a_1 = a_2$, the function f is **one to one**. If the function is one to one, then it has an **inverse** f^{-1} which is a function from B to A .

2.1.2 Boolean Algebra

An **algebraic system** is defined to be a system

$$\langle D, o_1, \dots, o_k; R_1, \dots, R_m; c_1, \dots, c_n \rangle \quad (2.3)$$

where D is non-empty set, o_i is a function $D^{l_i} \rightarrow D$, l_i is an integer, R_i is a relation on D , and c_i is an element of D .

Boolean algebra is an algebraic system $\langle B, \vee, \wedge, 0, 1 \rangle$, also denoted as $\langle B, +, \cdot, 0, 1 \rangle$. The elements of B should satisfy the following axioms:

1. Idempotent laws: $a \vee a = a$, $a \wedge a = a$;
2. Commutative laws: $a \vee b = b \vee a$, $a \wedge b = b \wedge a$;
3. Associative laws: $a \vee (b \vee c) = (a \vee b) \vee c$, $a \wedge (b \wedge c) = (a \wedge b) \wedge c$;
4. Absorption laws: $a \vee (a \wedge b) = a$, $a \wedge (a \vee b) = a$;
5. Distributive laws: $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$, $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$;
6. Existence of complement.

The operation $\vee$ or $+$ is called **OR** or **Boolean sum**, $\wedge$ or $\cdot$ is called **AND** or **Boolean product**. The complement of an element in Boolean algebra is denoted by $\bar{a}$, which is also known as **NOT**. In Boolean algebra, the complement is unique, and $\bar{\bar{a}} = a$.

The **De Morgan's laws** are very useful in Boolean algebra, which holds:

$$\overline{(x + y)} = \bar{x} \cdot \bar{y} , \quad (2.4)$$

$$\overline{(x \cdot y)} = \bar{x} + \bar{y} . \quad (2.5)$$

2.1.3 Boolean Function

A **Boolean function** is a mapping from $B^n \rightarrow B$, which is also known as **switching function**. Since there are 2^n elements in B^n and 2 elements in B , there are total 2^{2^n} Boolean functions of n variables.

Let $\mathbf{x}_n = (x_n, x_{n-1}, \dots, x_1)$ be n -tuple over $\{0, 1\}$, and $f(\mathbf{x}_n)$ be an n -variable Boolean function. $f(\mathbf{x}_n)$ can be expanded as follows:

$$f(\mathbf{x}_n) = \bar{x}_n f(0, x_{n-1}, x_{n-2}, \dots, x_2, x_1) + x_n f(1, x_{n-1}, x_{n-2}, \dots, x_2, x_1). \quad (2.6)$$

This result is known as **Shannon expansion**. By applying the Shannon expansion to a Boolean function recursively, we can have:

$$\begin{aligned} f(\mathbf{x}_n) = & f(0, 0, \dots, 0) \bar{x}_1 \bar{x}_2 \dots \bar{x}_n \\ & + f(0, 0, \dots, 1) x_1 \bar{x}_2 \dots \bar{x}_n \\ & \vdots \\ & + f(1, 1, \dots, 1) x_1 x_2 \dots x_n. \end{aligned} \quad (2.7)$$

Note that the OR $+$ in (2.6) and (2.7) can be replaced by the exclusive-OR (EXOR) $\oplus$. The **positive and negative Davio expansions** are another two types of EXOR-based expansions of Boolean functions. The n -variable Boolean function $f(\mathbf{x}_n)$ can be composed by two $(n-1)$ -variable Boolean functions in positive and negative Davio expansions as follows:

$$f(\mathbf{x}_n) = f(0, x_{n-1}, \dots, x_1) \oplus x_n f_D(\mathbf{x}_{n-1}), \quad (2.8)$$

$$\text{and } f(\mathbf{x}_n) = f(1, x_{n-1}, \dots, x_1) \oplus \bar{x}_n f_D(\mathbf{x}_{n-1}), \quad (2.9)$$

where $f_D(\mathbf{x}_{n-1}) = f(0, x_{n-1}, \dots, x_1) \oplus f(1, x_{n-1}, \dots, x_1)$.

The value of the n -variable Boolean function $f(\mathbf{x}_n)$ can be represented by the truth vector $\mathbf{F}$ and $\mathbf{F} = [f(0) \ f(1) \ \dots \ f(2^n - 1)]^T$ where the superscript T is the matrix transpose operator. The logic operations between the Boolean functions can be represented by their truth vectors. Let subscripts B and Z to represent the operations over Boolean domain and integer number domain, respectively.

Let $f^{a \wedge b}$ be the Boolean product of logic functions f_a and f_b , i.e., $f^{a \wedge b} = f_a \wedge f_b$. Then,

$$f^{a \wedge b}(i) = (f_a(i) \wedge f_b(i))_B = (f_a(i) \cdot f_b(i))_Z. \quad (2.10)$$

Let $f^{a \vee b}$ be the Boolean sum of logic functions f_a and f_b , i.e., $f^{a \vee b} = f_a \vee f_b$. Then,

$$f^{a \vee b}(i) = (f_a(i) \vee f_b(i))_B = (f_a(i) + f_b(i) - f_a(i) \cdot f_b(i))_Z. \quad (2.11)$$

Let $f^{a \oplus b}$ be the EXOR sum of logic functions f_a and f_b , i.e., $f^{a \oplus b} = f_a \oplus f_b$. Then,

$$f^{a \oplus b}(i) = (f_a(i) \oplus f_b(i))_B = (f_a(i) + f_b(i) - 2f_a(i) \cdot f_b(i))_Z \quad (2.12)$$

Let $\bar{f}$ be the complement of logic function f , then:

$$\bar{f}(i) = (\overline{f(i)})_B = (1 - f(i))_Z \quad (2.13)$$

In (2.10)-(2.13). $0 \leq i < 2^n$.

An **incompletely specified Boolean function** is a mapping from $D \rightarrow B$ where $D \subseteq B^n$, while a mapping from $B^n \rightarrow B$ is also called **completely specified Boolean function**.

There are several types of coding that have been developed for the representation of incompletely specified Boolean functions, including **S coding** and **R coding**. In S coding, the **true** minterms are denoted by -1 , **false** minterms by 1 , and **don't care (DC)** minterms by 0 . In R coding, the true minterms are denoted by 1 , false minterms by 0 , and DC minterms by 0.5 .

2.1.4 Binary Decision Diagram

Binary decision diagram (BDD) is a graphical representation of a Boolean function as a data structure which has a more compact representation than other methods.

A **binary decision tree** is a tree to represent an n -variable Boolean function. Its internal nodes are labeled by a variable x_i and have exactly two outgoing edges, called **0-edge** and **1-edge**. And its terminal nodes are labeled by one of two constants 0 or 1 . Each label of the terminal nodes represents the value of the function on one input.

A BDD which has only two terminal nodes, called **0-terminal** and **1-terminal** node, is derived from a binary decision tree by eliminating the redundant nodes and isomorphic subgraphs.

As an example of the binary decision tree and BDD for a 3-variable Boolean function, the function $f = ab \vee c$ is considered. Its corresponding binary decision tree and BDD are shown in Fig. 2.1(a) and Fig. 2.1(b), respectively. The binary

Figure 2.1: Binary decision tree and BDD for the function $f = ab \vee c$.

decision tree and BDD rely on the representation of the Boolean function by the Shannon expansion.

2.2 Classical Spectral Transforms

A Boolean function can be transformed from the Boolean domain to different spectral domains. Let the Boolean function f to be represented by its truth vector $\mathbf{F}$ which is a $2^n \times 1$ matrix. The transformation from the Boolean domain to a spectral domain is called the **forward** transform and shown as follows:

$$\mathbf{S} = \mathbf{T}\mathbf{F} \quad (2.14)$$

where $\mathbf{T}$ is a $2^n \times 2^n$ matrix and corresponds to the specified spectral transform. $\mathbf{S}$ is called the **spectrum** of the function f for the corresponding spectral transform. Note that $\mathbf{S}$ is a $2^n \times 1$ matrix and its elements are called as the **spectral coefficients**.

The transform from the spectral domain to Boolean domain is called **inverse** transform which holds the inverse matrix of the forward transform. In formulae:

$$\mathbf{F} = \mathbf{T}^{-1}\mathbf{S}. \quad (2.15)$$

The matrices of the forward and inverse spectral transforms hold:

$$\mathbf{T}\mathbf{T}^{-1} = \mathbf{I}_{2^n} \quad (2.16)$$

where $\mathbf{I}_{2^n}$ is a $2^n \times 2^n$ identical matrix.

The **computational costs** stand for the number of the arithmetic operations (additions, subtractions and multiplications) required by the calculation of the spectral transform. The **fast transform**, also known as the **fast algorithm**, is the method to reduce the computational costs by calculating the products of the factored matrices for the spectral transform.

2.2.1 Kronecker Product of Matrices

The **Kronecker product** of matrices, denoted by $\otimes$, is an important matrix operation in the constructions of the spectral transforms. Let $\mathbf{A}$ and $\mathbf{B}$ be two matrices, the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$ is:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{00}\mathbf{B} & a_{01}\mathbf{B} & \cdots & a_{0(n-1)}\mathbf{B} \\ a_{10}\mathbf{B} & a_{11}\mathbf{B} & \cdots & a_{1(n-1)}\mathbf{B} \\ \vdots & \vdots & & \vdots \\ a_{(m-1)0}\mathbf{B} & a_{(n-1)1}\mathbf{B} & \cdots & a_{(m-1)(n-1)}\mathbf{B} \end{bmatrix} \quad (2.17)$$

If the order of matrix $\mathbf{A}$ is $m \times n$ and the order of matrix $\mathbf{B}$ is $p \times q$, then the order of $\mathbf{A} \otimes \mathbf{B}$ is $mp \times nq$.

The Kronecker product is not commutative, but it is associative and has many useful properties as shown below:

$$\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C} = (\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}), \quad (2.18)$$

$$(\mathbf{A} + \mathbf{B}) \otimes (\mathbf{C} + \mathbf{D}) = \mathbf{A} \otimes \mathbf{C} + \mathbf{A} \otimes \mathbf{D} + \mathbf{B} \otimes \mathbf{C} + \mathbf{B} \otimes \mathbf{D}, \quad (2.19)$$

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}), \quad (2.20)$$

$$(\mathbf{A} \otimes \mathbf{B})^T = \mathbf{A}^T \otimes \mathbf{B}^T, \quad (2.21)$$

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}. \quad (2.22)$$

2.2.2 Walsh Transform

The **Walsh function** is a set of orthogonal functions and classified to three groups according to the different types of orderings. The **Walsh transform** discussed in this dissertation is based on Walsh functions in **Hadamard ordering**.

The forward Walsh transform matrix of order 2^n is defined as:

$$\mathbf{W}(n) = \bigotimes_{i=1}^n \mathbf{W}(1) \quad (2.23)$$

$$\text{or } \mathbf{W}(n) = \begin{bmatrix} \mathbf{W}(n-1) & \mathbf{W}(n-1) \\ \mathbf{W}(n-1) & -\mathbf{W}(n-1) \end{bmatrix} \quad (2.24)$$

$$\text{where } \mathbf{W}(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The inverse Walsh transform matrix $\mathbf{W}^{-1}(n)$ of order 2^n is defined as:

$$\mathbf{W}^{-1}(n) = \frac{1}{2^n} \mathbf{W}(n) \quad (2.25)$$

$$\text{or } \mathbf{W}(n) = \bigotimes_{i=1}^n \mathbf{W}^{-1}(1) \quad (2.26)$$

$$\text{where } \mathbf{W}^{-1}(1) = \frac{1}{2} \mathbf{W}(1) = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The **Walsh spectrum**, denoted by $\mathbf{S}_{\mathbf{W}}$, of an n -variable Boolean function f can be calculated as follows:

$$\mathbf{S}_{\mathbf{W}} = \mathbf{W}(n) \mathbf{F}. \quad (2.27)$$

The **Walsh expansion** of the function f can be derived from the corresponding Walsh spectrum $\mathbf{S}_{\mathbf{W}}$ as follows:

$$f = \left(\bigotimes_{i=1}^n [1 \quad 1 - 2x_{n+1-i}] \right) \mathbf{S}_{\mathbf{W}}. \quad (2.28)$$

The **multi-polarity Walsh transform** matrix of order 2^n is defined as:

$$\mathbf{MW}^{\omega_n}(n) = \bigotimes_{i=1}^n \mathbf{MW}^{\omega_i}(1) \quad (2.29)$$

$$\text{where } \omega_n = \{\omega_n, \dots, \omega_i, \dots, \omega_1\}, \text{ and } \mathbf{MW}^{\omega_i}(1) = \begin{cases} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & \text{if } \omega_i = 0, \\ \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} & \text{if } \omega_i = 1. \end{cases}$$

2.2.3 Haar Transform

The **Haar functions** form a complete system of 2^n continuous orthonormal functions over the interval $[0, 1]$. The Haar functions are defined by:

$$\begin{aligned} \text{haar}(0, 0, \theta) &= 1, \\ \text{haar}(i, j, \theta) &= \begin{cases} \sqrt{2^i} & \frac{j-1}{2^i} \leq \theta < \frac{j-1/2}{2^i}; \\ -\sqrt{2^i} & \frac{j-1/2}{2^i} \leq \theta < \frac{j}{2^i}; \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (2.30)$$

where $0 \leq \theta \leq 1$, $i = 0, 1, \dots, n-1$ and $j = 1, 2, \dots, 2^i$.

Discrete Haar functions can be defined as functions determined by sampling the Haar functions at 2^n points. The normalized and non-normalized Haar transforms are generated from the normalized and non-normalized Haar functions, respectively.

The **normalized Haar transform** matrix of order 2^n is defined as:

$$\mathbf{NH}(n) = \begin{bmatrix} \mathbf{NH}(n-1) \otimes [1 & 1] \\ 2^{\frac{n-1}{2}} \mathbf{I}_{2^{n-1}} \otimes [1 & -1] \end{bmatrix}, \quad \text{and } \mathbf{NH}(0) = [1]. \quad (2.31)$$

The **non-normalized Haar transform** is obtained from replacing the nonzero entries of normalized Haar matrix with their arithmetic signs. The non-normalized Haar transform preserves all the properties of the normalized Haar transform. Its matrix of order 2^n is defined as:

$$\mathbf{H}(n) = \begin{bmatrix} \mathbf{H}(n-1) \otimes [1 & 1] \\ \mathbf{I}_{2^{n-1}} \otimes [1 & -1] \end{bmatrix}, \quad \text{and } \mathbf{H}(0) = [1]. \quad (2.32)$$

The **normalized and non-normalized Haar spectra** are represented by $\mathbf{S}_{\mathbf{NH}}$ and $\mathbf{S}_{\mathbf{H}}$, respectively. $\mathbf{S}_{\mathbf{H}}$ and $\mathbf{S}_{\mathbf{NH}}$ can be calculated as follows:

$$\begin{aligned}\mathbf{S}_{\mathbf{H}} &= \mathbf{H}(n)\mathbf{F}, \\ \mathbf{S}_{\mathbf{NH}} &= \mathbf{NH}(n)\mathbf{F}.\end{aligned}\tag{2.33}$$

2.2.4 Hadamard-Haar Transform

The Hadamard-Haar transform is the hybrid orthogonal transform of Walsh and Haar transforms. The r th-order **normalized Hadamard-Haar transform** $\mathbf{NHH}_r(n)$ is defined as:

$$\mathbf{NHH}_r(n) = \mathbf{W}(r) \otimes \mathbf{NH}(n-r)\tag{2.34}$$

where r is an integer and $r \in [0, n]$.

Similar to Haar transform, the **non-normalized Hadamard-Haar transform** can be obtained from replacing the nonzero entries of normalized Hadamard-Haar matrix with their arithmetic signs. Its matrix is defined as:

$$\mathbf{HH}_r(n) = \mathbf{W}(r) \otimes \mathbf{H}(n-r).\tag{2.35}$$

The **normalized and non-normalized Hadamard-Haar spectra** are represented by $\mathbf{S}_{\mathbf{NHH}}$ and $\mathbf{S}_{\mathbf{HH}}$, respectively. $\mathbf{S}_{\mathbf{HH}}$ and $\mathbf{S}_{\mathbf{NHH}}$ can be calculated

as follows:

$$\begin{aligned}\mathbf{S}_{\mathbf{H}\mathbf{H}} &= \mathbf{H}\mathbf{H}(n)\mathbf{F}, \\ \mathbf{S}_{\mathbf{N}\mathbf{H}\mathbf{H}} &= \mathbf{N}\mathbf{H}\mathbf{H}(n)\mathbf{F}.\end{aligned}\tag{2.36}$$

2.2.5 Reed-Muller Transform

The **Reed-Muller transform** that is performing over Galois Field (2) (GF(2)) can realize any Boolean function by the modulo-2 sum-of-products expression. The matrix of order 2^n for the Reed-Muller transform is defined as:

$$\mathbf{R}(n) = \begin{bmatrix} \mathbf{R}(n-1) & 0 \\ \mathbf{R}(n-1) & \mathbf{R}(n-1) \end{bmatrix}, \text{ and } \mathbf{R}(0) = [1].\tag{2.37}$$

Alternatively,

$$\mathbf{R}(n) = \bigotimes_{i=1}^n \mathbf{R}(1)\tag{2.38}$$

$$\text{where } \mathbf{R}(1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

Since the Reed-Muller transform is performing over GF(2), it is self-inverse, i.e., $\mathbf{R}(n)\mathbf{R}(n) = \mathbf{I}_{2^n}$.

The **Reed-Muller spectrum**, denoted by $\mathbf{S}_{\mathbf{R}}$, of an n -variable Boolean function f can be calculated over GF(2) as follows:

$$\mathbf{S}_{\mathbf{R}} = \mathbf{R}(n)\mathbf{F}.\tag{2.39}$$

The **Reed-Muller expansion** of an n -variable function $f(\mathbf{x}_n)$ can be derived from the corresponding Reed-Muller coefficients as follows:

$$\begin{aligned} f(\mathbf{x}_n) = & s_R(0) \oplus s_R(1)x_1 \oplus s_R(2)x_2 \oplus s_R(3)x_1x_2 \oplus s_R(4)x_3 \oplus \dots \\ & \oplus s_R(2^n - 2)x_1x_2 \dots x_{n-1} \oplus s_R(2^n - 1)x_1x_2 \dots x_n. \end{aligned} \quad (2.40)$$

The **fixed polarity Reed-Muller transform** can provide the modulo-2 sum-of-products expression in which some of the variables are uniformly negated for the Boolean function. The fixed polarity Reed-Muller transform matrix of order 2^n is defined as:

$$\mathbf{FR}^{\omega_n}(n) = \bigotimes_{i=1}^n \mathbf{FR}^{\omega_i}(1) \quad (2.41)$$

$$\text{where } \mathbf{FR}^{\omega_i}(1) = \begin{cases} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} & \text{if } \omega_i = 0, \\ \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} & \text{if } \omega_i = 1. \end{cases}$$

2.2.6 Arithmetic Transform

The matrix of order 2^n for the **arithmetic transform** is defined as:

$$\mathbf{A}(n) = \begin{bmatrix} \mathbf{A}(n-1) & 0 \\ -\mathbf{A}(n-1) & \mathbf{A}(n-1) \end{bmatrix}, \quad \text{and } \mathbf{A}(0) = [1]. \quad (2.42)$$

Alternatively,

$$\mathbf{A}(n) = \bigotimes_{i=1}^n \mathbf{A}(1) \quad (2.43)$$

where $\mathbf{A}(1) = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$.

The inverse arithmetic transform $\mathbf{A}^{-1}(n)$, also known as **adding transform**, has exactly the same matrix structure with the Reed-Muller transform, while the adding transform is performed over real number field.

The **arithmetic spectrum**, denoted by $\mathbf{S}_{\mathbf{A}}$, of an n -variable Boolean function f can be calculated as follows:

$$\mathbf{S}_{\mathbf{A}} = \mathbf{A}(n)\mathbf{F}. \quad (2.44)$$

The **arithmetic expansion** of the function f can be derived from the corresponding arithmetic spectrum $\mathbf{S}_{\mathbf{A}}$ as follows:

$$f = \left(\bigotimes_{i=1}^n [1 \quad x_{n+1-i}] \right) \mathbf{S}_{\mathbf{A}}. \quad (2.45)$$

The **multi-polarity arithmetic transform** matrix of order 2^n is defined as:

$$\mathbf{MA}^{\omega_n}(n) = \bigotimes_{i=1}^n \mathbf{MA}^{\omega_i}(1) \quad (2.46)$$

$$\text{where } \mathbf{MA}^{\omega_i}(1) = \begin{cases} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} & \text{if } \omega_i = 0, \\ \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} & \text{if } \omega_i = 1. \end{cases}$$

Chapter 3

Mutual Relationships between Hadamard-Haar and Arithmetic Transforms

3.1 Introduction

The mutual relations between Walsh, Haar, arithmetic, Reed-Muller transforms have been investigated [Fal99c, Fal05, Fin72, HS97, RZ04, Sta82, TDM01, TMD01] since some transforms are better suited than others depending on the specific applications. The Hadamard-Haar transform is a hybrid version of the Haar and Walsh transforms which can be seen as the two special cases of Hadamard-Haar transform. The Hadamard-Haar transform has the advantages of Walsh and Haar transforms. Arithmetic transform has been used in logic design, testing of digital circuits and design verification [Hei91, RZ01, RZ04]. Therefore, the study of the mutual relations between Hadamard-Haar and arith-

metic transforms is crucial for the direct conversion between Hadamard-Haar and arithmetic spectra.

3.2 Transformations between Hadamard-Haar and Arithmetic Spectra

Since Hadamard-Haar transform is the hybrid of Walsh-Hadamard and Haar transform, the relations between Hadamard-Haar and arithmetic transforms can be derived from their definitions.

Property 3.2.1. Let $\mathbf{S}_{\mathbf{A}}(n)$ and $\mathbf{S}_{\mathbf{NHH}}(n)$ represent the arithmetic and normalized Hadamard-Haar spectra. The mutual relations between r th-order normalized Hadamard-Haar and arithmetic transforms for the general case of arbitrary n are as follows:

$$\mathbf{S}_{\mathbf{A}}(n) = \frac{1}{2^n} \mathbf{U}_r(n) \times \mathbf{S}_{\mathbf{NHH}}(n) \quad (3.1)$$

$$\text{and } \mathbf{S}_{\mathbf{NHH}}(\mathbf{n}) = \mathbf{V}_r(n) \times \mathbf{S}_{\mathbf{A}}(n) \quad (3.2)$$

where

$$\mathbf{U}_r(n) = \left(\bigotimes_{i=1}^r \begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix} \right) \otimes \left[\begin{array}{c} \left(\begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \right) \vdots \\ \sqrt{2} \left(\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-2} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \right) \vdots \\ 2 \left(\bigotimes_{i=1}^2 \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-3} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \vdots \\ 2\sqrt{2} \left(\bigotimes_{i=1}^3 \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-4} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \vdots \\ \vdots \\ 2^{\frac{j-1}{2}} \left(\bigotimes_{i=1}^{j-1} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-j} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \vdots \\ \vdots \\ 2^{\frac{n-r-1}{2}} \left(\bigotimes_{i=1}^{n-r-1} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \end{array} \right], \quad (3.3)$$

$$\mathbf{V}_r(n) = \left(\bigotimes_{i=1}^r \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix} \right) \otimes \left[\begin{array}{c} \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-1} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \sqrt{2} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-2} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline 2 \left(\bigotimes_{i=1}^2 \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-3} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline 2\sqrt{2} \left(\bigotimes_{i=1}^3 \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-4} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \dots \\ \hline 2^{\frac{j-1}{2}} \left(\bigotimes_{i=1}^{j-1} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-j} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \dots \\ \hline 2^{\frac{n-r-1}{2}} \left(\bigotimes_{i=1}^{n-r-1} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \end{array} \right], \quad (3.4)$$

$n = 1, 2, 3, \dots$, $0 \leq r \leq n$ and $0 \leq j \leq n - r$.

In the above equations, $\mathbf{U}_r(n)$ and $\mathbf{V}_r(n)$ are $2^n \times 2^n$ matrices, the vertical dotted lines denote the layered vertical Kronecker matrices, and the horizontal

dashed lines denote the layered horizontal Kronecker matrices, respectively. A layered horizontal Kronecker matrix is defined as the horizontal sum of Kronecker matrices [Yar85]. In a similar manner, the notion of a layered Kronecker vertical matrix is introduced that can be defined as the vertical sum of Kronecker matrices. When the Kronecker direct product of m matrices is carried out for the above equations for $m \leq 0$ then the term $\bigotimes_{i=1}^m$ disappears from the above equations. The meaning of the symbols and the restriction on the term $\bigotimes_{i=1}^m$ is the same as above also for the following equations.

Proof. Let the vector in function domain be represented by $\mathbf{F}(n)$, we have: $\mathbf{S}_A(n) = \mathbf{A}(n) \times \mathbf{F}(n)$ and $\mathbf{S}_{\mathbf{NHH}}(n) = \mathbf{NHH}_r(n) \times \mathbf{F}(n)$. Hence, $\mathbf{S}_A(n) = \mathbf{A}(n) \times (\mathbf{NHH}_r^{-1}(n) \times \mathbf{S}_{\mathbf{NHH}}(n))$.

From (2.35), we can get the arithmetic spectrum from normalized Hadamard-Haar spectrum as follows:

$$\begin{aligned} \mathbf{S}_A(n) &= \mathbf{A}(n) \times [(\mathbf{W}(r) \otimes \mathbf{NH}(n-r))^{-1} \times \mathbf{S}_{\mathbf{NHH}}(n)] \\ &= \mathbf{A}(n) \times (\mathbf{W}^{-1}(r) \otimes \mathbf{NH}^{-1}(n-r)) \times \mathbf{S}_{\mathbf{NHH}}(n). \end{aligned}$$

The inverse Walsh-Hadamard transform is known as: $\mathbf{W}^{-1}(n) = \frac{1}{2^n} \mathbf{W}(n)$, and the inverse normalized Haar transform is known as: $\mathbf{NH}^{-1}(n) = \frac{1}{2^n} \mathbf{NH}^T(n)$ where the superscript T is the matrix transpose operator. Using (2.23), (2.42), and the inverse transforms, we have:

$$\begin{aligned} \mathbf{S}_A(n) &= \mathbf{A}(n) \times \left[\left(\frac{1}{2^r} \mathbf{W}(r) \right) \otimes \left(\frac{1}{2^{n-r}} \mathbf{NH}^T(n-r) \right) \right] \times \mathbf{S}_{\mathbf{NHH}}(n) \\ &= \frac{1}{2^n} \left(\bigotimes_{i=1}^n \mathbf{A}(1) \right) \times \left[\left(\bigotimes_{i=1}^r \mathbf{W}(1) \right) \otimes \mathbf{NH}^T(n-r) \right] \times \mathbf{S}_{\mathbf{NHH}}(n). \end{aligned}$$

According to the properties of Kronecker product, we have:

$$\begin{aligned}
 \mathbf{S}_{\mathbf{A}}(n) &= \frac{1}{2^n} \left\{ \left[\left(\bigotimes_{i=1}^r \mathbf{A}(1) \right) \times \left(\bigotimes_{i=1}^r \mathbf{W}(1) \right) \right] \otimes \left[\left(\bigotimes_{i=1}^{n-r} \mathbf{A}(1) \right) \times \mathbf{NH}^T(n-r) \right] \right\} \times \mathbf{S}_{\mathbf{NH}}(n) \\
 &= \frac{1}{2^n} \left\{ \left[\bigotimes_{i=1}^r (\mathbf{A}(1) \times \mathbf{W}(1)) \right] \otimes \left[\left(\bigotimes_{i=1}^{n-r} \mathbf{A}(1) \right) \times \mathbf{NH}^T(n-r) \right] \right\} \times \mathbf{S}_{\mathbf{NH}}(n) \\
 &= \frac{1}{2^n} [\mathbf{C}(r) \otimes (\mathbf{A}(n-r) \times \mathbf{NH}^T(n-r))] \times \mathbf{S}_{\mathbf{NH}}(n) \tag{3.5}
 \end{aligned}$$

$$\text{where } \mathbf{C}(r) = \bigotimes_{i=1}^r \mathbf{C}(1) \text{ and } \mathbf{C}(1) = \mathbf{A}(1) \times \mathbf{W}(1) = \begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix}.$$

Comparing (3.1), (3.3), and (3.5), we need to prove $\mathbf{A}(m) \times \mathbf{NH}^T(m) = \mathbf{D}(m)$

for the proof of (3.1) where $m = n - r$ and

$$\mathbf{D}(m) = \begin{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{m-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) & \vdots \\ \vdots & \vdots \\ 2^{\frac{j-1}{2}} \left(\bigotimes_{i=1}^{j-1} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{m-j} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) & \vdots \\ \vdots & \vdots \\ 2^{\frac{m-1}{2}} \left(\bigotimes_{i=1}^{m-1} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} & \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{C}(1) \otimes \left(\bigotimes_{i=1}^{m-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) & \vdots \\ \vdots & \vdots \\ 2^{\frac{j-1}{2}} \mathbf{A}(j-1) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{m-j} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) & \vdots \\ \vdots & \vdots \\ 2^{\frac{m-1}{2}} \mathbf{A}(m-1) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} & \vdots \end{bmatrix}.$$

The argument $\mathbf{A}(m) \times \mathbf{NH}^T(m) = \mathbf{D}(m)$ will be proved by induction.

$$\text{Basis: From (2.31), we have: } \mathbf{NH}^T(m) = \begin{bmatrix} \mathbf{NH}^T(m-1) \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & \vdots & 2^{\frac{m-1}{2}} \mathbf{I}_{2^{m-1}} \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix}.$$

Hence, $\mathbf{NH}^T(1) = \mathbf{W}(1)$, then $\mathbf{A}(1) \times \mathbf{NH}^T(1) = \mathbf{C}(1) = \mathbf{D}(1)$.

Induction Step: Assume $\mathbf{A}(k) \times \mathbf{NH}^T(k) = \mathbf{D}(k)$. From (2.43), we have:

$$\begin{aligned}
 \mathbf{A}(k+1) \times \mathbf{NH}^T(k+1) &= (\mathbf{A}(k) \otimes \mathbf{A}(1)) \times \left[\mathbf{NH}^T(k) \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \vdots \quad 2^{\frac{k}{2}} \mathbf{I}_{2^k} \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right] \\
 &= \left[(\mathbf{A}(k) \otimes \mathbf{A}(1)) \times \left(\mathbf{NH}^T(k) \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) \quad \vdots \quad (\mathbf{A}(k) \otimes \mathbf{A}(1)) \times \left(2^{\frac{k}{2}} \mathbf{I}_{2^k} \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) \right] \\
 &= \left[(\mathbf{A}(k) \times \mathbf{NH}^T(k)) \otimes \left(\mathbf{A}(1) \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) \quad \vdots \quad 2^{\frac{k}{2}} (\mathbf{A}(k) \times \mathbf{I}_{2^k}) \otimes \left(\mathbf{A}(1) \times \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) \right] \\
 &= \left[\mathbf{D}(k) \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \vdots \quad 2^{\frac{k}{2}} \mathbf{A}(k) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right] \\
 &= \left[\begin{array}{c} \mathbf{C}(1) \otimes \left(\bigotimes_{i=1}^{k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \vdots \\ \dots \quad \vdots \\ 2^{\frac{j-1}{2}} \mathbf{A}(j-1) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{k-j} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \vdots \\ \dots \quad \vdots \\ 2^{\frac{k-1}{2}} \mathbf{A}(k-1) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \vdots \\ 2^{\frac{k}{2}} \mathbf{A}(k) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \end{array} \right]
 \end{aligned}$$

$$\begin{aligned}
&= \begin{bmatrix} \mathbf{C}(1) \otimes \left(\bigotimes_{i=1}^{(k+1)-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) & \vdots \\ \vdots & \vdots \\ 2^{\frac{j-1}{2}} \mathbf{A}(j-1) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{(k+1)-j} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) & \vdots \\ \vdots & \vdots \\ 2^{\frac{k-1}{2}} \mathbf{A}(k-1) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \vdots \\ 2^{\frac{(k+1)-1}{2}} \mathbf{A}((k+1)-1) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} & \vdots \end{bmatrix} \\
&= \mathbf{D}(k+1).
\end{aligned}$$

As $\mathbf{A}(m) \times \mathbf{NH}^T(m) = \mathbf{D}(m)$ is proved by induction, the proof of (3.1) is completed.

Similarly, we can prove (3.2). From the inverse arithmetic transform known as: $\mathbf{A}(n) = \bigotimes_{i=1}^n \mathbf{A}^{-1}(1)$ where $\mathbf{A}^{-1}(1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, we can get the normalized Hadamard-Haar spectrum from arithmetic spectrum as follows:

$$\begin{aligned}
\mathbf{S}_{\mathbf{NH}\mathbf{H}}(n) &= \mathbf{NH}\mathbf{H}(n) \times \mathbf{A}^{-1}(n) \times \mathbf{S}_{\mathbf{A}}(n) \\
&= (\mathbf{W}(r) \otimes \mathbf{NH}(n-r)) \times \mathbf{A}^{-1}(n) \times \mathbf{S}_{\mathbf{A}}(n) \\
&= [\mathbf{E}(r) \otimes (\mathbf{NH}(n-r) \times \mathbf{A}^{-1}(n-r))] \times \mathbf{S}_{\mathbf{A}}(n) \quad (3.6)
\end{aligned}$$

where $\mathbf{E}(r) = \bigotimes_{i=1}^r \mathbf{E}(1)$ and $\mathbf{E}(1) = \mathbf{W}(1) \times \mathbf{A}^{-1}(1) = \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix}$. Comparing (3.2), (3.4), and (3.6), we need to prove $\mathbf{NH}(m) \times \mathbf{A}^{-1}(m) = \mathbf{G}(m)$ for the

proof of (3.2) where $m = n - r$ and

$$\mathbf{G}(m) = \begin{bmatrix} \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{m-1} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \dots \\ \hline 2^{\frac{j-1}{2}} \left(\bigotimes_{i=1}^{j-1} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{m-j} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \dots \\ \hline 2^{\frac{m-1}{2}} \left(\bigotimes_{i=1}^{m-1} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \\ \hline \mathbf{E}(1) \otimes \left(\bigotimes_{i=1}^{m-1} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \dots \\ \hline 2^{\frac{j-1}{2}} \mathbf{A}^{-1}(j-1) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{m-j} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \dots \\ \hline 2^{\frac{m-1}{2}} \mathbf{A}^{-1}(m-1) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{E}(1) \otimes \left(\bigotimes_{i=1}^{m-1} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \dots \\ \hline 2^{\frac{j-1}{2}} \mathbf{A}^{-1}(j-1) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{m-j} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \dots \\ \hline 2^{\frac{m-1}{2}} \mathbf{A}^{-1}(m-1) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \end{bmatrix}.$$

The argument $\mathbf{NH}(m) \times \mathbf{A}^{-1}(m) = \mathbf{G}(m)$ will be proved by induction.

Basis: From (2.32), we have: $\mathbf{NH}(1) = \mathbf{W}(1)$, thus, $\mathbf{NH}(1) \times \mathbf{A}^{-1}(1) =$

$$\mathbf{E}(1) = \mathbf{G}(1).$$

Induction Step: Assume $\mathbf{NH}(k) \times \mathbf{A}^{-1}(k) = \mathbf{G}(k)$. We have:

$$\begin{aligned} \mathbf{NH}(k+1) \times \mathbf{A}^{-1}(k+1) &= \begin{bmatrix} \mathbf{NH}(k) \otimes [1 & 1] \\ \text{-----} \\ 2^{\frac{k}{2}} \mathbf{I}_{2^k} \otimes [1 & -1] \end{bmatrix} \times (\mathbf{A}^{-1}(k) \otimes \mathbf{A}^{-1}(1)) \\ &= \begin{bmatrix} (\mathbf{NH}(k) \otimes [1 & 1]) \times (\mathbf{A}^{-1}(k) \otimes \mathbf{A}^{-1}(1)) \\ \text{-----} \\ 2^{\frac{k}{2}} (\mathbf{I}_{2^k} \otimes [1 & -1]) \times (\mathbf{A}^{-1}(k) \otimes \mathbf{A}^{-1}(1)) \end{bmatrix} \\ &= \begin{bmatrix} (\mathbf{NH}(k) \times \mathbf{A}^{-1}(k)) \otimes ([1 & 1] \times \mathbf{A}^{-1}(1)) \\ \text{-----} \\ 2^{\frac{k}{2}} (\mathbf{I}_{2^k} \times \mathbf{A}^{-1}(k)) \otimes ([1 & -1] \times \mathbf{A}^{-1}(1)) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{G}(k) \otimes [2 & 1] \\ \text{-----} \\ 2^{\frac{k}{2}} \mathbf{A}^{-1}(k) \otimes [0 & -1] \end{bmatrix} = \mathbf{G}(k+1). \end{aligned}$$

This completes the proof of (3.2). $\square$

Example 3.2.1. For $n = 3$ and $r = 1$, the relationships between the normalized Hadamard-Haar and arithmetic transforms can be described by using Property 3.2.1. The corresponding $\mathbf{U}_1(3)$ and $\mathbf{V}_1(3)$ are:

$$\mathbf{U}_1(3) = \left(\begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix} \right) \otimes \left[\sqrt{2} \begin{bmatrix} 1 & 1 \\ 0 & -2 \\ 1 & 0 \\ -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \\ 1 \\ -2 \end{bmatrix} \right]$$

$$\begin{aligned}
&= \begin{bmatrix} 1 & 1 & \sqrt{2} & 0 & 1 & 1 & \sqrt{2} & 0 \\ 0 & 0 & -2\sqrt{2} & 0 & 0 & 0 & -2\sqrt{2} & 0 \\ 0 & -2 & -\sqrt{2} & \sqrt{2} & 0 & -2 & -\sqrt{2} & \sqrt{2} \\ 0 & 0 & 2\sqrt{2} & -2\sqrt{2} & 0 & 0 & 2\sqrt{2} & -2\sqrt{2} \\ 0 & 0 & 0 & 0 & -2 & -2 & -2\sqrt{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4\sqrt{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 2\sqrt{2} & -2\sqrt{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & -4\sqrt{2} & 4\sqrt{2} \end{bmatrix}, \\
&\text{and } \mathbf{V}_1(3) = \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix} \otimes \left[\begin{array}{c} \left[\begin{array}{cc} 2 & 1 \\ 0 & -1 \end{array} \right] \otimes \left[\begin{array}{cc} 2 & 1 \end{array} \right] \\ \hline \sqrt{2} \left[\begin{array}{cc} 1 & 0 \\ 1 & 1 \end{array} \right] \otimes \left[\begin{array}{cc} 2 & 1 \end{array} \right] \end{array} \right] \\
&= \begin{bmatrix} 8 & 4 & 4 & 2 & 4 & 2 & 2 & 1 \\ 0 & 0 & -4 & -2 & 0 & 0 & -2 & -1 \\ 0 & -2\sqrt{2} & 0 & 0 & 0 & -\sqrt{2} & 0 & 0 \\ 0 & -2\sqrt{2} & 0 & -2\sqrt{2} & 0 & -\sqrt{2} & 0 & -\sqrt{2} \\ 0 & 0 & 0 & 0 & -4 & -2 & -2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & \sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sqrt{2} & 0 & \sqrt{2} \end{bmatrix}.
\end{aligned}$$

Hence, the transformations between the normalized Hadamard-Haar spectrum $\mathbf{S}_{\text{NHH}}(3)$ where $r = 1$ and arithmetic spectrum $\mathbf{S}_{\mathbf{A}}(3)$ can be described by using

$\mathbf{U}_1(3)$ and $\mathbf{V}_1(3)$ as follows:

$$\begin{aligned}\mathbf{S}_{\mathbf{A}}(3) &= \frac{1}{8}\mathbf{U}_1(3) \times \mathbf{S}_{\mathbf{NHH}}(3), \\ \text{and } \mathbf{S}_{\mathbf{NHH}}(3) &= \mathbf{V}_1(3) \times \mathbf{S}_{\mathbf{A}}(3).\end{aligned}$$

The relationships between non-normalized Hadamard-Haar and arithmetic transforms are shown in Property 3.2.2. The relationships between non-normalized Hadamard-Haar and arithmetic transforms can also be derived similarly as the non-normalized Hadamard-Haar transform is obtained from the normalized version by using signs only.

Property 3.2.2. Let $\mathbf{S}_{\mathbf{HH}}(n)$ represent the non-normalized Hadamard-Haar spectra. The mutual relations between r th-order non-normalized Hadamard-Haar and arithmetic transforms for the general case of arbitrary n are as follows:

$$\mathbf{S}_{\mathbf{A}}(n) = \frac{1}{2^n}\mathbf{U}'_r(n) \times \mathbf{S}_{\mathbf{HH}}(n) \tag{3.7}$$

$$\text{and } \mathbf{S}_{\mathbf{HH}}(n) = \mathbf{V}'_r(n) \times \mathbf{S}_{\mathbf{A}}(n) \tag{3.8}$$

where

$$\mathbf{U}'_r(n) = \left(\bigotimes_{i=1}^r \begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix} \right) \otimes \left[\begin{array}{c} \begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \\ \vdots \\ 2 \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-2} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \\ \vdots \\ 4 \left(\bigotimes_{i=1}^2 \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-3} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \\ \vdots \\ 8 \left(\bigotimes_{i=1}^3 \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-4} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \\ \vdots \\ \dots \\ 2^{j-1} \left(\bigotimes_{i=1}^{j-1} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-j} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \\ \vdots \\ \dots \\ 2^{n-r-1} \left(\bigotimes_{i=1}^{n-r-1} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \end{array} \right], \quad (3.9)$$

$$\mathbf{V}'_r(n) = \left(\bigotimes_{i=1}^r \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix} \right) \otimes \left[\begin{array}{c} \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-1} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-2} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \left(\bigotimes_{i=1}^2 \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-3} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \dots \\ \hline \left(\bigotimes_{i=1}^{j-1} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r-j} \begin{bmatrix} 2 & 1 \end{bmatrix} \right) \\ \hline \dots \\ \hline \left(\bigotimes_{i=1}^{n-r-1} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 0 & -1 \end{bmatrix} \end{array} \right], \quad (3.10)$$

$$n = 1, 2, 3, \dots, 0 \leq r \leq n \text{ and } 0 \leq j \leq n - r.$$

Proof. Similar to the proof of Property 3.2.1. □

Since Walsh-Hadamard and Haar transforms can be seen as the special cases (when $r = n$ and $r = 0$) of Hadamard-Haar transform, the Walsh and Haar domains are subdomains of Hadamard-Haar domain. Hence, the relation-

ships shown in Property 3.2.1 and Property 3.2.2 can be applied for Walsh-Hadamard, Haar and arithmetic transforms as well. For example, the relationships between Walsh-Hadamard and arithmetic transforms can be derived from (3.1) as follows: $\mathbf{S}_A(n) = \frac{1}{2^n} \mathbf{J}_n(n) \times \mathbf{S}_W(n) = \frac{1}{2^n} \mathbf{C}(n) \times \mathbf{S}_W(n)$ and $\mathbf{S}_W(n) = \mathbf{V}_n(n) \times \mathbf{S}_A(n) = \mathbf{E}(n) \times \mathbf{S}_A(n)$ where $\mathbf{S}_W(n)$ is the corresponding Walsh-Hadamard spectrum. The relationships between normalized Haar and arithmetic transforms can be also derived from (3.1) as follows: $\mathbf{S}_A(n) = \frac{1}{2^n} \mathbf{J}_0(n) \times \mathbf{S}_{NH}(n) = \frac{1}{2^n} \mathbf{D}(n) \times \mathbf{S}_{NH}(n)$ and $\mathbf{S}_{NH}(n) = \mathbf{V}_n(n) \times \mathbf{S}_A(n) = \mathbf{G}(n) \times \mathbf{S}_A(n)$.

Example 3.2.2. For $n = 3$ and $r = 1$, the relationships between the non-normalized Hadamard-Haar and arithmetic transforms can be described by using Property 3.2.2 as follows:

$$\begin{aligned} \mathbf{S}_A(3) &= \frac{1}{8} \mathbf{U}'_1(3) \times \mathbf{S}_{HH}(3), \\ \text{and } \mathbf{S}_{HH}(3) &= \mathbf{V}'_1(3) \times \mathbf{S}_A(3). \end{aligned}$$

where the corresponding $\mathbf{U}'_1(3)$ and $\mathbf{V}'_1(3)$ are:

$$\mathbf{U}'_1(3) = \left(\begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix} \right) \otimes \left[\begin{array}{c} \begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 2 \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \end{array} \right]$$

$$\begin{aligned}
&= \begin{bmatrix} 1 & 1 & 2 & 0 & 1 & 1 & 2 & 0 \\ 0 & 0 & -4 & 0 & 0 & 0 & -4 & 0 \\ 0 & -2 & -2 & 2 & 0 & -2 & -2 & 2 \\ 0 & 0 & 4 & -4 & 0 & 0 & 4 & -4 \\ 0 & 0 & 0 & 0 & -2 & -2 & -4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 4 & -4 \\ 0 & 0 & 0 & 0 & 0 & 0 & -8 & 8 \end{bmatrix}, \\
&\text{and } \mathbf{V}'_1(3) = \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix} \otimes \left[\begin{array}{c} \left[\begin{array}{cc} 2 & 1 \\ 0 & -1 \end{array} \right] \otimes \left[\begin{array}{cc} 2 & 1 \end{array} \right] \\ \hline \left[\begin{array}{cc} 1 & 0 \\ 1 & 1 \end{array} \right] \otimes \left[\begin{array}{cc} 2 & 1 \end{array} \right] \end{array} \right] \\
&= \begin{bmatrix} 8 & 4 & 4 & 2 & 4 & 2 & 2 & 1 \\ 0 & 0 & -4 & -2 & 0 & 0 & -2 & -1 \\ 0 & -2 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -2 & 0 & -2 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -4 & -2 & -2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}.
\end{aligned}$$

3.3 Fast Algorithms

The fast algorithms for different discrete transforms have been developed to reduce the computational costs, such as fast Fourier transform, fast Walsh

transform, fast Haar transform, fast arithmetic transform, fast Hadamard-Haar transform [Aga80, AAE95, Bea84, Hur92, Kar76, Mal97, SSJ98, SA03, Yar85, ER82, Har69, HMM85, RNR75b, RJK77, SWSP90, WSW72, Zal89, GES91, SF03, BGG98, Cas96, SF96, Tho97, VK95, Moh92, Kul84, KSZ90, KSY91, TMD01, TDM01]. The matrix transformation methods between Hadamard-Haar and arithmetic domains have been shown in Section 3.2. The fast algorithms for the transformations between Hadamard-Haar and arithmetic domains will be shown in this section.

Property 3.3.1. For the relations between r th-order normalized Hadamard-Haar and arithmetic transforms, the $\mathbf{U}_r(n)$ and $\mathbf{V}_r(n)$ in Property 3.2.1 can be expressed by the products of factorized matrices as follows:

$$\mathbf{U}_r(n) = \prod_{i=1}^n \mathbf{P}_i(n) \quad (3.11)$$

$$\text{and } \mathbf{V}_r(n) = \prod_{i=1}^n \mathbf{Q}_i(n) \quad (3.12)$$

where

$$\mathbf{P}_i(n) = \begin{cases} \mathbf{I}_{2^{i-1}} \otimes \mathbf{C}(1) \otimes \mathbf{I}_{2^{n-i}} & \text{for } 1 \leq i \leq r, \\ \mathbf{I}_{2^r} \otimes \text{diag} \left(\left[\begin{array}{c} \mathbf{I}_{2^{n-i}} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 2^{\frac{n-i}{2}} \mathbf{I}_{2^{n-i}} \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \end{array} \right], \overbrace{\mathbf{J}_{n-i+1}(n-i+1), \dots, \mathbf{J}_{n-i+1}(n-r-1)}^{i-r-1} \right) & \text{for } r < i \leq n, \end{cases}$$

$$\mathbf{Q}_i(n) = \begin{cases} \mathbf{I}_{2^{i-1}} \otimes \mathbf{E}(1) \otimes \mathbf{I}_{2^{n-i}} & \text{for } 1 \leq i \leq r, \\ \mathbf{I}_{2^r} \otimes \text{diag} \left(\left[\begin{array}{c} \mathbf{I}_{2^{i-r-1}} \otimes [2 \quad 1] \\ \text{-----} \\ 2^{\frac{i-r-1}{2}} \mathbf{I}_{2^{i-r-1}} \otimes [0 \quad -1] \end{array} \right], \overbrace{\mathbf{K}_{i-r}(i-r), \dots, \mathbf{K}_{i-r}(n-r-1)}^{n-i} \right) & \text{for } r < i \leq n, \end{cases}$$

$\mathbf{J}_j(k) = \mathbf{I}_{2^{j-1}} \otimes \mathbf{A}(1) \otimes \mathbf{I}_{2^{k-j}}$, $\mathbf{K}_j(k) = \mathbf{I}_{2^{j-1}} \otimes \mathbf{A}^{-1}(1) \otimes \mathbf{I}_{2^{k-j}}$, $\mathbf{C}(1)$ and $\mathbf{E}(1)$ are the same as in the proof of Property 3.2.1.

Proof. For $\mathbf{U}_r(n)$, we have:

$$\begin{aligned}\mathbf{U}_r(n) &= \mathbf{C}(r) \otimes \mathbf{D}(n-r) = (\mathbf{C}(r) \times \mathbf{I}_{2^r}) \otimes (\mathbf{I}_{2^{n-r}} \times \mathbf{D}(n-r)) \\ &= (\mathbf{C}(r) \otimes \mathbf{I}_{2^{n-r}}) \times (\mathbf{I}_{2^r} \otimes \mathbf{D}(n-r)) = \prod_{i=1}^r \mathbf{P}_i(n) \times (\mathbf{I}_{2^r} \otimes \mathbf{D}(n-r)).\end{aligned}$$

Since $\prod_{i=r+1}^n \mathbf{P}_i(n) = \mathbf{I}_{2^r} \otimes \left(\prod_{i=r+1}^n \mathbf{L}_{i-r}(n-r) \right)$ where

$$\mathbf{L}_i(n) = \text{diag} \left(\left[\mathbf{I}_{2^{n-i}} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right] : 2^{\frac{n-i}{2}} \mathbf{I}_{2^{n-i}} \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right], \overbrace{\mathbf{J}_{n-i+1}(n-i+1), \dots, \mathbf{J}_{n-i+1}(n-1)}^{i-1} \right),$$

we need to prove $\prod_{i=1}^m \mathbf{L}_i(m) = \mathbf{D}(m)$ by inductive method for the proof of (3.11)

where $m = n - r$.

Basis: $\prod_{i=1}^1 \mathbf{L}_i(1) = \mathbf{L}_1(1) = \mathbf{C}(1) = \mathbf{D}(1)$.

Induction Step: Assume $\prod_{i=1}^k \mathbf{L}_i(k) = \mathbf{D}(k)$. From the structure of $\mathbf{L}_i(n)$, it can be noticed that: $\mathbf{L}_i(n) = \text{diag}(\mathbf{L}_{i-1}(n-1), \mathbf{J}_{n-i+1}(n-1))$. Therefore,

$$\begin{aligned}\prod_{i=1}^{k+1} \mathbf{L}_i(k+1) &= \mathbf{L}_1(k+1) \times \prod_{i=2}^{k+1} [\text{diag}(\mathbf{L}_{i-1}(k), \mathbf{J}_{k+2-i}(k))] \\ &= \mathbf{L}_1(k+1) \times \text{diag} \left(\prod_{i=1}^k \mathbf{L}_i(k), \prod_{i=1}^k \mathbf{J}_i(k) \right) \\ &= \left[\mathbf{I}_{2^k} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} : 2^{\frac{k}{2}} \mathbf{I}_{2^k} \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right] \times \text{diag}(\mathbf{D}(k), \mathbf{A}(k)) \\ &= \left[\mathbf{D}(k) \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} : 2^{\frac{k}{2}} \mathbf{A}(k) \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right] \\ &= \mathbf{D}(k+1).\end{aligned}$$

This completes the proof of (3.11).

Similarly, for $\mathbf{V}_r(n)$, we have: $\mathbf{V}_r(n) = \mathbf{E}(r) \otimes \mathbf{G}(n-r) = \prod_{i=1}^r \mathbf{Q}_i(n) \times (\mathbf{I}_{2^r} \otimes \mathbf{G}(n-r))$. Then we need to prove $\prod_{i=1}^m \mathbf{M}_i(m) = \mathbf{G}(m)$ by induction for the proof of (3.12) where

$$\mathbf{M}_i(n) = \text{diag} \left(\begin{bmatrix} \mathbf{I}_{2^{i-1}} \otimes [2 & 1] \\ \text{-----} \\ 2^{\frac{i-1}{2}} \mathbf{I}_{2^{i-1}} \otimes [0 & -1] \end{bmatrix}, \overbrace{\mathbf{K}_i(i), \dots, \mathbf{K}_i(n-1)}^{n-i} \right).$$

Basis: $\prod_{i=1}^1 \mathbf{M}_i(1) = \mathbf{M}_1(1) = \mathbf{E}(1) = \mathbf{G}(1)$.

Induction Step: Assume $\prod_{i=1}^k \mathbf{M}_i(k) = \mathbf{G}(k)$. Since $\mathbf{M}_i(n) = \text{diag}(\mathbf{L}_i(n-1), \mathbf{K}_i(n-1))$, we have:

$$\begin{aligned} \prod_{i=1}^{k+1} \mathbf{M}_i(k+1) &= \prod_{i=1}^k [\text{diag}(\mathbf{M}_i(k), \mathbf{K}_i(k))] \times \mathbf{M}_{k+1}(k+1) \\ &= \text{diag} \left(\prod_{i=1}^k \mathbf{M}_i(k), \prod_{i=1}^k \mathbf{K}_i(k) \right) \times \mathbf{M}_{k+1}(k+1) \\ &= \text{diag}(\mathbf{G}(k), \mathbf{A}^{-1}(k)) \times \begin{bmatrix} \mathbf{I}_{2^k} \otimes [2 & 1] \\ \text{-----} \\ 2^{\frac{k}{2}} \mathbf{I}_{2^k} \otimes [0 & -1] \end{bmatrix} = \begin{bmatrix} \mathbf{G}(k) \otimes [2 & 1] \\ \text{-----} \\ 2^{\frac{k}{2}} \mathbf{A}^{-1}(k) \otimes [0 & -1] \end{bmatrix} \\ &= \mathbf{G}(k+1). \end{aligned}$$

This completes the proof of (3.12). $\square$

Property 3.3.2. For the relations between r th-order non-normalized Hadamard-Haar and arithmetic transforms, the $\mathbf{U}'_r(n)$ and $\mathbf{V}'_r(n)$ in Property 3.2.2 can be

expressed by the products of factorized matrices as follows:

$$\mathbf{U}'_r(n) = \prod_{i=1}^n \mathbf{P}'_i(n) \quad (3.13)$$

$$\text{and } \mathbf{V}'_r(n) = \prod_{i=1}^n \mathbf{Q}'_i(n) \quad (3.14)$$

where

$$\mathbf{P}'_i(n) = \begin{cases} \mathbf{I}_{2^{i-1}} \otimes C(1) \otimes \mathbf{I}_{2^{n-i}} & \text{for } 1 \leq i \leq r, \\ \mathbf{I}_{2^r} \otimes \text{diag} \left(\begin{bmatrix} \mathbf{I}_{2^{n-i}} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 2^{n-i} \mathbf{I}_{2^{n-i}} \otimes \begin{bmatrix} 1 \\ -2 \end{bmatrix} \end{bmatrix}, \overbrace{\mathbf{J}_{n-i+1}(n-i+1), \dots, \mathbf{J}_{n-i+1}(n-r-1)}^{i-r-1} \right) & \text{for } r < i \leq n, \end{cases}$$

$$\text{and } \mathbf{Q}'_i(n) = \begin{cases} \mathbf{I}_{2^{i-1}} \otimes \mathbf{E}(1) \otimes \mathbf{I}_{2^{n-i}} & \text{for } 1 \leq i \leq r, \\ \mathbf{I}_{2^r} \otimes \text{diag} \left(\begin{bmatrix} \mathbf{I}_{2^{i-r-1}} \otimes [2 \ 1] \\ - \dots - \\ \mathbf{I}_{2^{i-r-1}} \otimes [0 \ -1] \end{bmatrix}, \overbrace{\mathbf{K}_{i-r}(i-r), \dots, \mathbf{K}_{i-r}(n-r-1)}^{n-i} \right) & \text{for } r < i \leq n. \end{cases}$$

Proof. Similar to the proof of Property 3.3.1. □

The fast algorithms shown in Property 3.3.1 and Property 3.3.2 can also be used for Walsh-Hadamard or Haar transform when r is set to n for the first transform or 0 for the second transform.

Example 3.3.1. We can use the fast algorithms to factorize $\mathbf{U}'_1(3)$ in Example 3.2.2 as follows:

$$\mathbf{P}'_1(3) = \begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned}
&= \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{bmatrix}, \\
\mathbf{P}'_2(3) &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 0 & -4 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & -4 \end{bmatrix}, \\
&= \begin{bmatrix} 1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 \end{bmatrix}, \\
\text{and } \mathbf{P}'_3(3) &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix},
\end{aligned}$$

Figure 3.1: The flow diagrams for the transformation from non-normalized Hadamard-Haar spectrum to arithmetic spectrum using proposed method where $r = 1$ and $n = 3$.

$$= \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix},$$

Since $\mathbf{S}_A(3) = \frac{1}{8}\mathbf{U}'_1(3) \times \mathbf{S}_{HH}(3) = \frac{1}{8} \prod_{i=1}^3 \mathbf{P}'_i(3) \times \mathbf{S}_{HH}(3)$, the butterfly diagram for the transformation between the non-normalized Hadamard-Haar and arithmetic spectra can be constructed based on the factorized matrices as shown in Fig. 3.1, where $\mathbf{P}'_1(3)$ corresponds to the rightmost stage, $\mathbf{P}'_2(3)$ corresponds to

Figure 3.2: The flow diagrams for the transformation from non-normalized Hadamard-Haar spectrum to arithmetic spectrum using conventional method where $r = 1$ and $n = 3$.

the middle stage, and $\mathbf{P}'_3(3)$ corresponds to the leftmost stage of the butterfly diagram.

The same transformation using conventional method is also given here for the comparison. Conventionally, the transformation from the non-normalized Hadamard-Haar spectrum to arithmetic spectrum is calculated by using fast inverse Hadamard-Haar transform and fast arithmetic transform. We need to transform the non-normalized Hadamard-Haar spectrum to the functional domain using fast inverse Hadamard-Haar transform, then the corresponding arithmetic spectrum can be calculated from the functional domain using fast arithmetic transform. The corresponding fast butterfly diagram using conventional methods is shown in Fig. 3.2, where the left three stages in the butterfly diagram correspond to the fast inverse Hadamard-Haar transform which transforms the non-normalized Hadamard-Haar spectrum $\mathbf{S}_{\text{HH}}(3)$ to the functional domain, and the right three stages correspond to the fast arithmetic spectrum

which transforms the data in functional domain to the corresponding arithmetic spectrum $\mathbf{S}_A(3)$.

As shown in the butterfly diagrams in Fig. 3.1 and Fig. 3.2, the transformation is processed from the left side to the right side when the calculations in the same stage are parallel processed. Since the calculations in every stage require the output results of its left stage as the inputs, it can be noted the number of the stages in the butterfly diagram determines the processing time of the butterfly diagram. We can notice that there are three stages of calculations in the butterfly diagrams shown in Fig. 3.1, while there are six stages of calculations in Fig. 3.2. Therefore, the transformation using proposed method is much efficient than the one that is using conventional method. The more detailed analysis for the comparison of the proposed and conventional methods is shown in Section 3.4.

3.4 Analysis of Running Time of Algorithms

The running time of the algorithms depends on the computational costs required by the calculations which includes additions, subtractions, and multiplications. The running time of the conventional methods and proposed methods for the transformations between Hadamard-Haar and arithmetic transforms as well as some simulation results in MATLAB when using fast algorithms will be compared in this section.

It has been known that there are $n2^n$ arithmetic operations (additions and subtractions) required for the computation of 2^n -point Walsh transform [Bea84, HMM85, Kar76, SSJ98]. The computation of 2^n -point normalized Haar trans-

Table 3.1: Comparison of Computational Costs using Conventional and Proposed Methods

Transformation	Conventional Methods		Proposed Methods	
	Arithmetic Operations	Multiplications	Arithmetic	Multiplications
$\mathbf{S}_{\mathbf{NHH}}(n) \rightarrow \mathbf{S}_{\mathbf{A}}(n)$	$(n + 2r + 4)2^{n-1} - 2^{r+1}$	2^n	$n2^{n-1}$	$(r + 5)2^{n-1} - 3 \cdot 2^r$
$\mathbf{S}_{\mathbf{A}}(n) \rightarrow \mathbf{S}_{\mathbf{NHH}}(n)$	$(n + 2r + 4)2^{n-1} - 2^{r+1}$	$2^n - 2^{r+1}$	$n2^{n-1}$	$(r + 4)2^{n-1} - 3 \cdot 2^r$
$\mathbf{S}_{\mathbf{HH}}(n) \rightarrow \mathbf{S}_{\mathbf{A}}(n)$	$(n + 2r + 4)2^{n-1} - 2^{r+1}$	2^n	$n2^{n-1}$	$(r + 5)2^{n-1} - 3 \cdot 2^r$
$\mathbf{S}_{\mathbf{A}}(n) \rightarrow \mathbf{S}_{\mathbf{HH}}(n)$	$(n + 2r + 4)2^{n-1} - 2^{r+1}$	0	$n2^{n-1}$	$(r + 2)2^{n-1} - 2^r$

form requires $2(2^n - 1)$ arithmetic operations and $2^n - 2$ multiplications, and the computation of 2^n -point non-normalized Haar transform does not require multiplications [HMM85, SSJ98]. Since Hadamard-Haar transform includes Walsh and Haar parts, we can derive the computational costs for the computations of two versions of Hadamard-Haar transforms.

The number of arithmetic operations required by the calculation of forward or inverse Hadamard-Haar transform can be calculated as follows: $r2^n + 2^{r+1}(2^{n-r} - 1) = (r + 2)2^n - 2^{r+1}$. The numbers of multiplications required by calculating forward and inverse normalized Hadamard-Haar transform are: $2^r(2^{n-r} - 2) = 2^n - 2^{r+1}$ and $2^n + 2^n - 2^{r+1} = 2^{n+1} - 2^{r+1}$, respectively. There are 2^n multiplications required by the calculation of inverse non-normalized Hadamard-Haar transform.

It is also known that there are $n2^{n-1}$ arithmetic operations required by the calculation of forward or inverse arithmetic transform. So we can derive the computational costs for the transformations between Hadamard-Haar and arithmetic spectra using conventional methods based on the computational costs of forward (inverse) Hadamard-Haar and inverse (forward) arithmetic transforms. The computational costs of new proposed methods can be obtained from (3.11)-

Figure 3.3: The reduction in running time when using proposed method for the transformation from non-normalized Hadamard-Haar spectrum to arithmetic spectrum for some cases.

(3.15) and are shown in Table 3.1 for different cases. The numbers of arithmetic operations when using proposed methods are always less than the numbers when using conventional methods. It should be noticed that there are additional multiplications while using proposed methods than conventional methods in two cases for the normalized version, but the numerical value of the multipliers in the additional multiplications are all powers-of-two, i.e., the multiplier is $\pm 2^q$. The multiplications in the computations when using conventional and proposed methods for the non-normalized version are also $\pm 2^q$. Such multiplications can be easily implemented in hardware by bit-shift operations (plus inversion of sign digit), so this kind of multiplications are even more efficient than the arithmetic

Figure 3.4: The reduction in running time when using proposed method for the transformation from non-normalized Hadamard-Haar spectrum to arithmetic spectrum.

operations. For the convenience of comparing the computational costs when using conventional and proposed methods, we consider only the cases for the non-normalized versions and combine the arithmetic operations and multiplications, i.e., the sum of the arithmetic operations and multiplications. The proposed methods show computational advantages over the traditional method due to less computational costs. For example, the computational costs for the transformation from non-normalized Hadamard-Haar spectrum to arithmetic spectrum using conventional and proposed methods are: $(n + 2r + 6)2^{n-1} - 2^{r+1}$ and $(n + r + 5)2^{n-1} - 3 \cdot 2^r$, respectively. Therefore, the reduction in computational costs when using proposed method is: $\frac{(r + 1)2^{n-1} + 2^r}{(n + 2r + 6)2^{n-1} - 2^{r+1}} \times 100\%$. From the reductions for several cases ($r = n - 5, \dots, n - 1$) shown in Fig. 3.3,

Figure 3.5: The reduction in running time when using proposed method for the transformation from arithmetic spectrum to non-normalized Hadamard-Haar spectrum.

we can notice that most reductions in Fig. 3.3 are larger than 20%. The reductions in computational costs, which can also be seen as the reductions in running time, when using proposed method for the transformations between non-normalized Hadamard-Haar spectrum and arithmetic spectrum for general cases (all r) where $n \leq 30$ are shown in Fig. 3.4 and Fig. 3.5, respectively.

We will now show the significant savings in the time complexity when using proposed methods. Generally, there are 2^{n+1} (or $2^{n+1} + 1$) stages required in the fast diagrams for the computations when using conventional methods, while there are only 2^n (or $2^n + 1$) stages required when using proposed methods. Hence, the proposed methods carry about 50% saving in time complexity compared with the conventional methods.

The efficiency of proposed methods in the running time of the algorithms has been shown, which also means the simplicity of the design and the reduction of hardware costs, especially when all the operations are performed in sign complement arithmetic.

The presented derivations based on layered matrices and corresponding butterfly diagrams can be efficiently implemented in the form of operations on spectral decision diagrams using software or as hardware operations using LUT cascades in a manner similar to Walsh transform [FS04].

3.5 Summary

The mutual relationships between Hadamard-Haar and arithmetic transforms and their corresponding spectra in the form of matrix decomposition as layered vertical and horizontal Kronecker matrices are discussed in this chapter. The corresponding proofs, fast algorithms and computational costs are also provided. The new relations allow performing direct conversions between Arithmetic and Hadamard-Haar functions and their corresponding spectra. In addition, analysis of butterfly diagrams for these new relations is also introduced and it is shown that they are more efficient than the matrix decomposition method.

Chapter 4

Relationships of Boolean Operations in Spectral Domains

4.1 Introduction

The arithmetic, Reed-Muller and Haar spectra of the logic product, logic sum, EXOR sum and negation of Boolean functions are frequently computed in the applications of the corresponding spectral transforms in logic design [Hei91, RZ01, RZ04, AAE95, DDT78, Sas93]. The evaluation of the Walsh spectra of the composite Boolean functions without reintroducing the Walsh transform has been studied in [Eri78, Muz78, Muz80]. For arithmetic, Reed-Muller, or Haar spectra, the transformations between spectral domain and Boolean domain for the composite spectra from the known spectra of the Boolean functions are needed in current methods. In this chapter, the new methods for the calculations of the composite arithmetic, Reed-Muller and Haar spectra directly from the spectra of the individual Boolean functions are proposed. In the new meth-

ods, there are only the calculations in spectral domain which improves both computation and time complexity over the former approach where the transformations between spectral and Boolean domains are necessary which takes a lot of time to calculate the composite spectra. The study of the relationships of the Boolean operations in arithmetic, Reed-Muller and Haar domains is also theoretically interesting for the manipulations of Boolean functions in spectral domains.

4.2 Relationships of Boolean Operations in Arithmetic and Reed-Muller Domains

Although the arithmetic and Reed-Muller transforms are performed in different algebra, they are quite related and will be considered together in this section.

4.2.1 Preliminaries

We will show some known properties of arithmetic and Reed-Muller transforms and propose some new properties needed by the following sections.

The arithmetic transform is also known as integer-valued Reed-Muller transform due to the relation between the arithmetic expression and Reed-Muller expression [Fal99b]. It can be noted that the matrix of Reed-Muller transform can be derived by using modulo-2 algebra for the matrix of arithmetic transform, and the difference between Reed-Muller and arithmetic expressions is the EXOR (modulo-2 addition) and arithmetic addition. Hence, Reed-Muller transform can also be seen as the modulo-2 arithmetic transform in $\text{GF}(2)$.

Property 4.2.1. Let the j -th element of i -th row in the matrix of arithmetic transform be represented by $A_{i,j}$ where $0 \leq i < N$ and $0 \leq j < N$. $A_{i,j}$ can be derived by another method instead of Kronecker product as shown in [FC97b]:

$$A_{i,j} = (-1)^{\sum_{k=1}^n i_k \oplus j_k} \prod_{k=1}^n (i_k \vee \bar{j}_k). \quad (4.1)$$

Property 4.2.2. Let the j -th element of i -th row in the matrix of Reed-Muller transform be represented by $R_{i,j}$. Due to the relationship between arithmetic transform with Reed-Muller transform, $R_{i,j}$ can be generated in a manner similar to $A_{i,j}$ as:

$$R_{i,j} = \prod_{k=1}^n (i_k \vee \bar{j}_k). \quad (4.2)$$

The arithmetic transform and Reed-Muller transform can be considered as the mappings from functional domain to arithmetic and Reed-Muller spectral domains, respectively. The converse relations of these two mappings are also mappings since the arithmetic transform and Reed-Muller transform are bijections. The transform matrix of arithmetic transform is nonsingular and the inverse arithmetic transform is known under the name of 'adding transform' [FP90a, Fal99b]. The transform matrix of adding transform can be formed by replacing all the non-zero elements having entries equal to 1 from the transformation matrix of arithmetic transform. Although the adding transform has exactly the same matrix form with Reed-Muller transform, the 0 and 1 of the entries in those two transforms have different meaning. Since the calculations of the adding transform are performed over the integer number Z while the calculations of Reed-Muller transform are over $\text{GF}(2)$, the elements in the transform matrices of adding transform and Reed-Muller transform are in the integer number Z and $\text{GF}(2)$, respectively. For the convenience of the representation in this

section, we also use $\mathbf{R}(n)$ and $R_{i,j}$ to represent the transform matrix and its elements of adding transform, i.e., $\mathbf{A}(n) \cdot \mathbf{R}(n) = \mathbf{I}_{2^n}$. For the Reed-Muller transform, its transform is self-inverse since the addition and subtraction coincide in $\text{GF}(2)$, i.e., $(\mathbf{R}(n) \cdot \mathbf{R}(n))_{\text{GF}(2)} = \mathbf{I}_{2^n}$.

Property 4.2.3. For the elements in the transform matrices of arithmetic and adding transforms, we have:

$$\sum_{l=0}^{N-1} A_{i,l} R_{l,j} = \prod_{k=1}^n (i_k j_k + \bar{i}_k \bar{j}_k) \quad (4.3)$$

Proof. Since the arithmetic and adding transforms are the inverse transforms of each other, i.e., $\mathbf{A}(n) \cdot \mathbf{R}(n) = \mathbf{I}_{2^n}$, we can have:

$$\sum_{l=0}^{N-1} A_{i,l} R_{l,j} = I_{i,j}. \quad (4.4)$$

Since the element of the identity matrix is $I_{i,j} = \begin{cases} 1 & \text{if } i=j, \\ 0 & \text{if } i \neq j, \end{cases}$ and $i = (i_1 i_2 \dots i_n)_2$,

$j = (j_1 j_2 \dots j_n)_2$, we have:

$$I_{i,j} = \begin{cases} 1 & \text{if } i_k = j_k, \text{ for } k = 1, 2, \dots, n \\ 0 & \text{otherwise,} \end{cases} = \prod_{k=1}^n (i_k j_k + \bar{i}_k \bar{j}_k).$$

Hence, we can rewrite (4.4) as $\sum_{l=0}^{N-1} A_{i,l} R_{l,j} = \prod_{k=1}^n (i_k j_k + \bar{i}_k \bar{j}_k)$. $\square$

Property 4.2.4. For the elements in the transform matrix of adding transform, we have:

$$R_{i,j} R_{i,l} = R_{i,j \underline{\vee} l} \quad (4.5)$$

where the symbol $\underline{\vee}$ is the bit-wise OR.

Proof. From (4.2), we have:

$$\begin{aligned}
R_{i,j}R_{i,l} &= \prod_{k=1}^n (i_k \vee \bar{j}_k) \prod_{k=1}^n (i_k \vee \bar{l}_k) = \prod_{k=1}^n [(i_k \vee \bar{j}_k) (i_k \vee \bar{l}_k)] \\
&= \prod_{k=1}^n (i_k \vee i_k \bar{l}_k \vee i_k \bar{j}_k \vee \bar{j}_k \bar{l}_k) = \prod_{k=1}^n (i_k \vee \bar{j}_k \bar{l}_k) \\
&= \prod_{k=1}^n (i_k \vee (\bar{j}_k \vee \bar{l}_k)) = R_{i,j \vee l} .
\end{aligned}$$

This completes the proof of (4.5). $\square$

4.2.2 Calculation of Composite Spectra of Arithmetic and Reed-Muller Transforms

In this section, the relationships for basic Boolean formulae in arithmetic and Reed-Muller spectral domains are shown. Based on the relationships, the composite spectra can be calculated in the arithmetic or Reed-Muller spectral domain without the transformation to functional domain.

Let $\mathbf{S}_{\mathbf{A}_a}$ and $\mathbf{S}_{\mathbf{R}_a}$, $\mathbf{S}_{\mathbf{A}_b}$ and $\mathbf{S}_{\mathbf{R}_b}$ be the corresponding arithmetic and Reed-Muller spectra of two n -variable Boolean functions f_a and f_b .

Property 4.2.5. Let $\mathbf{S}_{\mathbf{A}}^{a \wedge b}$ be the arithmetic spectrum of $f^{a \wedge b}$. The arithmetic spectral coefficients $s_{\mathbf{A}}^{a \wedge b}$ can be generated from the arithmetic spectra of f_a and f_b as follows:

$$s_{\mathbf{A}}^{a \wedge b}(i) = \sum_{j=0}^{N-1} R_{i,j} s_{\mathbf{A}_a}(j) \sum_{l=0}^{N-1} R_{j,l} s_{\mathbf{A}_b}(l \oplus i) \quad (4.6)$$

where the symbol $\oplus$ represent the bit-wise EXOR.

Proof. From (2.44) and Property 4.1, the arithmetic spectrum of $f^{a \wedge b}$ can be

calculated as follows:

$$\begin{aligned}
s_A^{a \wedge b}(i) &= \sum_{j=0}^{N-1} A_{i,j} f^{a \wedge b}(j) = \sum_{j=0}^{N-1} A_{i,j} f_a(j) f_b(j) \\
&= \sum_{j=0}^{N-1} A_{i,j} \left(\sum_{p=0}^{N-1} R_{j,p} s_{A_a}(p) \right) \left(\sum_{q=0}^{N-1} R_{j,q} s_{A_b}(q) \right) \\
&= \sum_{q=0}^{N-1} \sum_{p=0}^{N-1} \sum_{j=0}^{N-1} A_{i,j} R_{j,p} R_{j,q} s_{A_a}(p) s_{A_b}(q). \tag{4.7}
\end{aligned}$$

Using Property 4.2.3 and Property 4.2.4, we can rewrite (4.7) as follows:

$$\begin{aligned}
s_A^{a \wedge b}(i) &= \sum_{q=0}^{N-1} s_{A_b}(q) \sum_{p=0}^{N-1} s_{A_a}(p) \sum_{j=0}^{N-1} A_{i,j} R_{j,p \vee q} \\
&= \sum_{q=0}^{N-1} s_{A_b}(q) \sum_{p=0}^{N-1} s_{A_a}(p) \prod_{k=1}^n [i_k(p_k \vee q_k) + \bar{i}_k(\overline{p_k \vee q_k})] \\
&= \sum_{q=0}^{N-1} s_{A_b}(q) \sum_{p=0}^{N-1} s_{A_a}(p) \prod_{k=1}^n (i_k p_k \vee i_k q_k \vee \bar{i}_k \bar{p}_k \bar{q}_k). \tag{4.8}
\end{aligned}$$

Then we consider the right part of (4.6). Using (4.2), the right part of (4.6) can be calculated as follows:

$$\begin{aligned}
&\sum_{j=0}^{N-1} R_{i,j} s_{A_b}(j) \sum_{l=0}^{N-1} R_{j,l} s_{A_a}(l \oplus i) \\
&= \sum_{j=0}^{N-1} R_{i,j} s_{A_b}(j) \sum_{p=0}^{N-1} R_{j,p \oplus i} s_{A_a}(p) \\
&= \sum_{j=0}^{N-1} s_{A_b}(j) \sum_{p=0}^{N-1} R_{i,j} R_{j,p \oplus i} s_{A_a}(p) \\
&= \sum_{j=0}^{N-1} s_{A_b}(j) \sum_{p=0}^{N-1} s_{A_a}(p) \prod_{k=1}^n (i_k \vee \bar{j}_k) \prod_{k=1}^n (i_k \vee \overline{p_k \oplus i_k}) \\
&= \sum_{j=0}^{N-1} s_{A_b}(j) \sum_{p=0}^{N-1} s_{A_a}(p) \prod_{k=1}^n [(i_k \vee \bar{j}_k) (i_k \vee i_k p_k \vee \bar{i}_k \bar{p}_k)]
\end{aligned}$$

$$\begin{aligned}
&= \sum_{j=0}^{N-1} s_{A_b}(j) \sum_{p=0}^{N-1} s_{A_a}(p) \prod_{k=1}^n (i_k j_k \vee i_k p_k \vee i_k \bar{j}_k p_k \vee i_k \bar{j}_k \bar{p}_k) \\
&= \sum_{j=0}^{N-1} s_{A_b}(j) \sum_{p=0}^{N-1} s_{A_a}(p) \prod_{k=1}^n (i_k j_k \vee i_k p_k \vee \bar{i}_k \bar{j}_k \bar{p}_k). \tag{4.9}
\end{aligned}$$

Comparing (4.9) and (4.8), the left part and right part of (4.6) are equal. Property 4.2.5 is proved. $\square$

Since the AND operation in Boolean algebra corresponds to the multiplication operation, Property 4.2.5 can be seen as the convolution theorem for arithmetic transform.

Example 4.2.1. Let $\mathbf{S}_{A_a}$ and $\mathbf{S}_{A_b}$ be the corresponding arithmetic spectra of f_a and f_b which are 3-variable Boolean functions. When those two arithmetic spectra are known, then the arithmetic spectrum $\mathbf{S}_{A^{a \wedge b}}(3)$ can be calculated using traditional method shown in (4.7) and proposed method shown in Property 4.2.5.

The algorithm to calculate $\mathbf{S}_{A^{a \wedge b}}(3)$ using traditional method is shown below:

```

for  $i = 0$  to 7 do
     $temp = 0$  ;
    for  $j = 0$  to 7 do
        if  $A_{i,j} \neq 0$  then
             $tempA = 0$  ;
            for  $p = 0$  to 7 do
                 $tempA = tempA + R_{j,p} s_{A_a}(p)$  ;
            end;
             $f_a(j) = tempA$  ;
             $tempB = 0$  ;

```

```

    for  $q = 0$  to 7 do
         $tempB = tempB + R_{j,p} s_{A_b}(p)$  ;
    end;
     $f_b(j) = tempB$  ;
     $f^{a \wedge b}(j) = f_a(j) f_b(j)$  ;
     $temp = temp + A_{i,j} f^{a \wedge b}(j)$  ;
end;
end;
 $a^{a \wedge b}(i) = temp$  ;
end;
```

The algorithm to calculate $\mathbf{S}_{\mathbf{A}}^{a \wedge b}(3)$ using proposed method is shown below:

```

for  $i = 0$  to 7 do
     $temp = 0$  ;
    for  $j = 0$  to 7 do
         $tempA = R_{i,j} s_{A_b}(j)$  ;
        if  $tempA \neq 0$  then
             $tempB = 0$  ;
            for  $l = 0$  to 7 do
                 $tempB = tempB + R_{j,l} s_{A_a}(l \oplus i)$  ;
            end;
             $temp = temp + tempA tempB$  ;
        end;
    end;
     $a^{a \wedge b}(i) = temp$  ;
end;
```

The proposed method shows the advantages in time complexity than conventional method due to smaller number of loops in the algorithm.

Since the arithmetic transform can be seen as the integer-valued Reed-Muller transform, the relationship in Reed-Muller spectral domain for the logic product of Boolean functions can be derived in GF(2) similarly.

Property 4.2.6. Let $\mathbf{S}_{\mathbf{R}}^{a \wedge b}$ be the Reed-Muller spectrum of $f^{a \wedge b}$. The Reed-Muller spectral coefficients $s_R^{a \wedge b}$ can be generated from the Reed-Muller spectra of f_a and f_b as follows:

$$s_R^{a \wedge b}(i) = \bigoplus_{j=0}^{N-1} R_{i,j} s_{R_b}(j) \bigoplus_{l=0}^{N-1} R_{j,l} s_{R_a}(l \oplus i) . \quad (4.10)$$

All the computations in (4.10) are over GF(2).

Similar to Property 4.2.5, Property 4.2.6 can be seen as the convolution theorem for Reed-Muller transform.

The relationships shown in Property 4.2.5 and Property 4.2.6 can be rewritten in matrix form in order to make more convenient for interpretation. We can construct a $2^n \times 2^n$ matrix $\mathbf{U}^f(n)$ by the arithmetic spectrum of n -variable Boolean function f . The j -th element of i -th column in matrix $\mathbf{U}^f(n)$ is defined as follows:

$$U_{i,j}^f = R_{i,j} \sum_{l=0}^{N-1} R_{j,l} s_A(l \oplus i) . \quad (4.11)$$

Hence, Property 4.2.5 can be interpreted as:

$$\mathbf{S}_{\mathbf{A}}^{a \wedge b} = \mathbf{U}^{f_a}(n) \cdot \mathbf{S}_{\mathbf{A}b} . \quad (4.12)$$

According to the commutativity in the logic product, we have also $\mathbf{S}_A^{a \wedge b} = \mathbf{U}^{f_b}(n) \cdot \mathbf{S}_{A_a}$.

Similarly, we can also construct a $2^n \times 2^n$ matrix $\mathbf{V}^f(n)$ by the Reed-Muller spectrum of n -variable Boolean function f . The j -th element of i -th column in matrix $\mathbf{V}^f(n)$ is defined as follows:

$$V_{i,j}^f = R_{i,j} \bigoplus_{l=0}^{N-1} R_{j,l} s_R(l \oplus i), \quad (4.13)$$

and Property 4.2.6 can be interpreted as:

$$\mathbf{S}_R^{a \wedge b} = \mathbf{V}^{f_a}(n) \cdot \mathbf{S}_{R_b} = \mathbf{V}^{f_b}(n) \cdot \mathbf{S}_{R_a}. \quad (4.14)$$

Example 4.2.2. For the 2-variable Boolean function, the structure of the corresponding matrix $\mathbf{U}^f(2)$ is:

$$\mathbf{U}^f(2) = \begin{bmatrix} s_A(0) & 0 & 0 & 0 \\ s_A(1) & s_A(0) + s_A(1) & 0 & 0 \\ s_A(2) & 0 & s_A(0) + s_A(2) & 0 \\ s_A(3) & s_A(2) + s_A(3) & s_A(1) + s_A(3) & s_A(0) + s_A(1) + s_A(2) + s_A(3) \end{bmatrix}.$$

Property 4.2.7. Let the matrix $\mathbf{U}^f(n)$ be denoted by four sub-matrices of order 2^{n-1} as:

$$\mathbf{U}^f(n) = \begin{bmatrix} \mathbf{U}_a^f(n) & \mathbf{U}_b^f(n) \\ \mathbf{U}_c^f(n) & \mathbf{U}_d^f(n) \end{bmatrix}.$$

We note that the top left sub-matrix $\mathbf{U}_a^f(n)$ has the same structure with the matrix $\mathbf{U}^f(n-1)$, and the top right sub-matrix $\mathbf{U}_b^f(n)$ is a matrix of zeros, i.e., $\mathbf{U}_a^f(n) = \mathbf{U}^f(n-1)$, $\mathbf{U}_b^f(n) = \mathbf{0}$. Also the $\mathbf{U}_c^f(n)$ can be considered as the

isomorphism of $\mathbf{U}_a^f(n)$ since the matrix $\mathbf{U}_c^f(n)$ can be generated by substituting $s_A(p)$ in $\mathbf{U}_a^f(n)$ for $s_A(p + \frac{N}{2})$. Furthermore, we can also note that $\mathbf{U}_d^f(n)$ is the sum matrix of matrices $\mathbf{U}_a^f(n)$ and $\mathbf{U}_c^f(n)$.

This property can be deduced from the structure of the matrix $\mathbf{U}^f(n)$, and the matrix $\mathbf{U}^f(n)$ has exactly the same property for its structure.

Example 4.2.3. Let the matrix $\mathbf{U}^f(3) = \begin{bmatrix} \mathbf{U}_a^f(3) & \mathbf{U}_b^f(3) \\ \mathbf{U}_c^f(3) & \mathbf{U}_d^f(3) \end{bmatrix}$. From (4.11), we have:

$$\mathbf{U}_a^f(3) = \begin{bmatrix} s_A(0) & 0 & 0 & 0 \\ s_A(1) & s_A(0) + s_A(1) & 0 & 0 \\ s_A(2) & 0 & s_A(0) + s_A(2) & 0 \\ s_A(3) & s_A(2) + s_A(3) & s_A(1) + s_A(3) & s_A(0) + s_A(1) + s_A(2) + s_A(3) \end{bmatrix},$$

$$\mathbf{U}_b^f(3) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{U}_c^f(3) = \begin{bmatrix} s_A(4) & 0 & 0 & 0 \\ s_A(5) & s_A(4) + s_A(5) & 0 & 0 \\ s_A(6) & 0 & s_A(4) + s_A(6) & 0 \\ s_A(7) & s_A(6) + s_A(7) & s_A(5) + s_A(7) & s_A(4) + s_A(5) + s_A(6) + s_A(7) \end{bmatrix},$$

and $\mathbf{U}_d^f(3) =$

$$\begin{bmatrix} s_A(0)+s_A(4) & 0 & 0 & 0 \\ s_A(1)+s_A(5) & s_A(0)+s_A(1)+s_A(4)+s_A(5) & 0 & 0 \\ s_A(2)+s_A(6) & 0 & s_A(0)+s_A(2)+s_A(4)+s_A(6) & 0 \\ s_A(3)+s_A(7) & s_A(2)+s_A(3)+s_A(6)+s_A(7) & s_A(1)+s_A(3)+s_A(5)+s_A(7) & \sum_{l=1}^7 s_A(l) \end{bmatrix}.$$

It is shown that $\mathbf{U}_a^f(3)$ has the same structure with $\mathbf{U}^f(2)$, $\mathbf{U}_b^f(3)$ is a matrix of zeros, and $\mathbf{U}_c^f(3)$ can be generated by substituting $s_A(p)$ in $\mathbf{U}_a^f(3)$ for $s_A(p+4)$. $\mathbf{U}_d^f(3)$ can be seen as the sum matrix of matrices $\mathbf{U}_a^f(3)$ and $\mathbf{U}_c^f(3)$.

Based on relationships in arithmetic and Reed-Muller spectral domains for the Boolean logic product in functional domains, several relationships for different Boolean formulae can be derived.

Property 4.2.8. Let $\mathbf{S}_A^{a \vee b}$ and $\mathbf{S}_R^{a \vee b}$ be the arithmetic and Reed-Muller spectra of $f^{a \vee b}$. $\mathbf{S}_A^{a \vee b}$ and $\mathbf{S}_R^{a \vee b}$ can be generated from the corresponding spectra of f_a and f_b as follows:

$$\begin{aligned} \mathbf{S}_A^{a \vee b} &= \mathbf{S}_{A_a} + \mathbf{S}_{A_b} - \mathbf{U}^{f_a}(n) \mathbf{S}_{A_b} = \mathbf{S}_{A_a} + (\mathbf{I}_{2^n} - \mathbf{U}^{f_a}(n)) \mathbf{S}_{A_b} \\ &= \mathbf{S}_{A_a} + \mathbf{S}_{A_b} - \mathbf{U}^{f_b}(n) \mathbf{S}_{A_a} = (\mathbf{I}_{2^n} - \mathbf{U}^{f_b}(n)) \mathbf{S}_{A_a} + \mathbf{S}_{A_b}, \end{aligned} \quad (4.15)$$

and

$$\begin{aligned} \mathbf{S}_R^{a \vee b} &= \mathbf{S}_{R_a} \oplus \mathbf{S}_{R_b} \oplus \mathbf{U}^{f_a}(n) \mathbf{S}_{R_b} = \mathbf{S}_{R_a} \oplus (\mathbf{I}_{2^n} \oplus \mathbf{U}^{f_a}(n)) \mathbf{S}_{R_b} \\ &= \mathbf{S}_{R_a} \oplus \mathbf{S}_{R_b} \oplus \mathbf{U}^{f_b}(n) \mathbf{S}_{R_a} = (\mathbf{I}_{2^n} \oplus \mathbf{U}^{f_b}(n)) \mathbf{S}_{R_a} \oplus \mathbf{S}_{R_b}. \end{aligned} \quad (4.16)$$

Proof. According to (2.11), $\mathbf{S}_A^{a \vee b} = \mathbf{A}(n) \mathbf{F}^{a \vee b} = \mathbf{A}(n) (\mathbf{F}_a + \mathbf{F}_b - \mathbf{F}^{a \wedge b}) = \mathbf{A}(n) \mathbf{F}_a + \mathbf{A}(n) \mathbf{F}_b - \mathbf{A}(n) \mathbf{F}^{a \wedge b} = \mathbf{S}_{A_a} + \mathbf{S}_{A_b} - \mathbf{U}^{f_a}(n) \mathbf{S}_{A_a} = \mathbf{S}_{A_b} + (\mathbf{I}_{2^n} -$

$\mathbf{U}^{f_a}(n)\mathbf{S}_{\mathbf{A}_b}$. This completes the proof of (4.15). We can prove (4.16) similarly where all the operations are over $\text{GF}(2)$. $\square$

Property 4.2.9. Let $\mathbf{S}_{\mathbf{A}}^{a\oplus b}$ and $\mathbf{S}_{\mathbf{R}}^{a\oplus b}$ be the arithmetic and Reed-Muller spectra of $f^{a\oplus b}$. $\mathbf{S}_{\mathbf{A}}^{a\oplus b}$ and $\mathbf{S}_{\mathbf{R}}^{a\oplus b}$ can be generated from the corresponding spectra of f_a and f_b as follows:

$$\begin{aligned}\mathbf{S}_{\mathbf{A}}^{a\oplus b} &= \mathbf{S}_{\mathbf{A}_a} + \mathbf{S}_{\mathbf{A}_b} - 2\mathbf{U}^{f_a}(n)\mathbf{S}_{\mathbf{A}_b} = \mathbf{S}_{\mathbf{A}_a} + (\mathbf{I}_{2^n} - 2\mathbf{U}^{f_a}(n))\mathbf{S}_{\mathbf{A}_b} \\ &= \mathbf{S}_{\mathbf{A}_a} + \mathbf{S}_{\mathbf{A}_b} - 2\mathbf{U}^{f_b}(n)\mathbf{S}_{\mathbf{A}_a} = (\mathbf{I}_{2^n} - 2\mathbf{U}^{f_b}(n))\mathbf{S}_{\mathbf{A}_a} + \mathbf{S}_{\mathbf{A}_b},\end{aligned}\quad (4.17)$$

and

$$\mathbf{S}_{\mathbf{R}}^{a\oplus b} = \mathbf{S}_{\mathbf{R}_a} \oplus \mathbf{S}_{\mathbf{R}_b} \quad (4.18)$$

Proof. Similar to the proof of Property 4.2.8, this property can be proved using (2.12). $\square$

Property 4.2.10. Let $\overline{\mathbf{A}}$ and $\overline{\mathbf{R}}$ be the arithmetic and Reed-Muller spectra of $\overline{f}$. $\overline{\mathbf{A}}$ and $\overline{\mathbf{R}}$ can be generated from the corresponding spectra of f as follows:

$$\overline{\mathbf{A}} = \mathbf{K} - \mathbf{S}_{\mathbf{A}} \quad (4.19)$$

and

$$\overline{\mathbf{R}} = \mathbf{K} \oplus \mathbf{S}_{\mathbf{A}} \quad (4.20)$$

where $\mathbf{K}$ is an $2^n \times 1$ matrix and $\mathbf{K} = [1 \ 0 \ 0 \ \dots \ 0 \ 0]^T$.

Proof. From (2.13), we have: $\overline{\mathbf{A}} = \mathbf{A}(n)\overline{\mathbf{F}} = \mathbf{A}(n)([1 \ 1 \ \dots \ 1 \ 1]^T - \mathbf{F})$. Accord-

ing to the structure of the transform matrix $\mathbf{A}(n)$, we have [KB81]: $\mathbf{A}(n) \times [1 \ 1 \dots 1 \ 1]^T = \mathbf{K}$. Hence, $\overline{\mathbf{A}} = \mathbf{K} - \mathbf{A}(n)\mathbf{F} = \mathbf{K} - \mathbf{S}_{\mathbf{A}}$. This completes the proof of (4.19). Similarly, we can prove (4.20) while all the calculations are over $\text{GF}(2)$. $\square$

The relationships in arithmetic and Reed-Muller spectral domains for basic Boolean formulae have been shown, and the arithmetic or Reed-Muller spectra of more complex Boolean expressions consisting of more Boolean functions can be calculated by using the proposed relationships repetitively without need to transform back to the functional domain. The correlation methods in arithmetic and Reed-Muller transforms can also be used widely as the ones in discrete Fourier transform which plays important roles in many fields including signal processing.

These relationships shown in previous properties also allow calculating only some chosen arithmetic or Reed-Muller coefficients without generation of the whole spectrum which is useful in some applications of arithmetic and Reed-Muller spectra, for example, in testing and verification [Hei91, RZ04].

Example 4.2.4. Let $\mathbf{S}_{\mathbf{A}_1}$ and $\mathbf{S}_{\mathbf{A}_2}$ be the corresponding arithmetic spectra of f_1 and f_2 which are 3-variable Boolean functions. Those two arithmetic spectra are known as: $\mathbf{S}_{\mathbf{A}_1} = [0 \ 1 \ 1 \ -2 \ 1 \ -2 \ -2 \ 3]^T$, and $\mathbf{S}_{\mathbf{A}_2} = [1 \ -1 \ 0 \ 0 \ -1 \ 2 \ 0 \ -1]^T$. We will use both traditional method and proposed method to get the arithmetic coefficients $A^{1\vee 2}(3)$ for the comparison of the calculations in these two methods.

Traditional Method:

Step 1: The truth vectors of f_1 and f_2 can be calculated using adding trans-

form that is the same as Reed-Muller transform as:

$$\mathbf{F}_1 = \mathbf{R}(3)\mathbf{S}_{\mathbf{A}_1} = [0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0]^T,$$

and

$$\mathbf{F}_2 = \mathbf{R}(3)\mathbf{S}_{\mathbf{A}_2} = [1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0]^T.$$

Step 2: From the truth vectors of f_1 and f_2 , the truth vector of function $f^{1\vee 2}$ can be obtained as:

$$\mathbf{F}^{1\vee 2} = [1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0]^T.$$

Step 3: We can calculate the arithmetic spectrum of $f^{1\vee 2}$ as:

$$\mathbf{S}_{\mathbf{A}}^{1\vee 2} = \mathbf{A}(3)\mathbf{F}^{1\vee 2} = [1 \ 0 \ 0 \ -1 \ 0 \ 0 \ -1 \ 1]^T$$

We have: $A^{1\vee 2}(3) = -1$.

Proposed Method:

Step 1: From the structure of $\mathbf{U}^f(3)$ shown in Example 4.2.3, we can have the fourth row of matrix $\mathbf{U}^{f_1}(3)$ as:

$$\begin{aligned} \mathbf{U}^{f_1}(3)_{4th-row} &= [s_{A_1}(3) \ s_{A_1}(2)+s_{A_1}(3) \ s_{A_1}(1)+s_{A_1}(3) \\ &\quad s_{A_1}(0)+s_{A_1}(1)+s_{A_1}(2)+s_{A_1}(3) \ 0 \ 0 \ 0 \ 0] \\ &= [-2 \ -1 \ -1 \ 0 \ 0 \ 0 \ 0 \ 0]. \end{aligned}$$

Step 2: The arithmetic coefficients $A^{1\vee 2}(3)$ can be calculated as:

$$\mathbf{S}_{\mathbf{A}}^{1\vee 2} = s_{A_1}(3) + s_{A_2}(3) - \mathbf{U}^{f_1}(3)_{4th-row} \mathbf{S}_{\mathbf{A}_2} = -2 + 0 - (-1) = -1.$$

In this example, it can be seen that the proposed method shows advantage in time complexity over conventional method, since the calculation using conventional method requires the transformations of spectral and functional domains which are not necessary when using proposed method. As only selected row of matrix $\mathbf{U}^{f_1}(3)$ is needed for the calculation when using proposed method, our new method also shows advantage in computational complexity especially for the calculation of the chosen spectral coefficient.

Example 4.2.5. Let $\mathbf{S}_{\mathbf{A}_3}$ and $\mathbf{S}_{\mathbf{A}_4}$ be the corresponding arithmetic spectra of f_3 and f_4 which are 3-variable Boolean functions. Those two arithmetic spectra are known as: $\mathbf{S}_{\mathbf{A}_3} = [1 \ -1 \ 0 \ 1 \ -1 \ 2 \ 1 \ -2]^T$, and $\mathbf{S}_{\mathbf{A}_4} = [0 \ 1 \ 1 \ -1 \ 1 \ -2 \ -1 \ 1]^T$. For the function $f_3 \wedge \overline{f_4}$, we can calculate its arithmetic spectrum $\mathbf{S}_{\mathbf{A}}^{3\wedge 4}$ by using the more than one proposed relationships.

Step 1: According to Property 4.2.10, we can have the arithmetic spectra $\overline{f_4}$ by using (4.19) as follows:

$$\overline{\mathbf{A}_4} = [1 \ 0 \ 0 \dots 0 \ 0]^T - \mathbf{S}_{\mathbf{A}_4} = [1 \ -1 \ -1 \ 1 \ -1 \ 2 \ 1 \ -1]^T.$$

Step 2: From (4.14):

$$\mathbf{S}_{\mathbf{A}}^{3\wedge 4} = \mathbf{U}^{f_3}(3) \cdot \overline{\mathbf{A}_4} = [1 \ -1 \ -1 \ 1 \ -1 \ 2 \ 1 \ -1].$$

As shown above, we do not need to make the conversions between spectral

and functional domains. The spectrum $\mathbf{S}_{\mathbf{A}}^{3\wedge\bar{4}}$ can be also derived by using conventional method which requires more calculations due to the transformations between spectral and functional domains. The procedure to calculate $\mathbf{S}_{\mathbf{A}}^{3\wedge\bar{4}}$ by using the conventional method is shown as follows:

Step 1: The truth vectors of f_3 and f_4 can be calculated using adding transform that is the same as Reed-Muller transform as:

$$\mathbf{F}_3 = \mathbf{R}(3)\mathbf{S}_{\mathbf{A}_3} = [1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1]^T,$$

and

$$\mathbf{F}_4 = \mathbf{R}(3)\mathbf{S}_{\mathbf{A}_4} = [0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0]^T.$$

From $\mathbf{F}_4$, the truth vector $\mathbf{F}_{\bar{4}}$ of $\bar{f}_4$ can be calculated as:

$$\mathbf{F}_{\bar{4}} = [1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1]^T.$$

Step 2: From the truth vectors of f_3 and $\bar{f}_4$, the truth vector of function $f_3 \wedge \bar{f}_4$ can be obtained as:

$$\mathbf{F}^{3\wedge\bar{4}} = [1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1]^T.$$

Step 3: We can calculate the arithmetic spectrum of $f^{1\vee 2}$ as:

$$\mathbf{S}_{\mathbf{A}}^{1\vee 2} = \mathbf{A}(3)\mathbf{F}^{1\vee 2} = [1 \ -1 \ -1 \ 1 \ -1 \ 2 \ 1 \ -1]^T.$$

The generalized Reed-Muller transform of an n -variable Boolean function

is known to have 2^n different polarities [FP90b, SF96, SS01, TMS96, TMS97] where the polarities correspond to the affirmation or negation of the variables in Boolean polynomials. The generalized arithmetic and adding transforms are developed in [FP90a] and their properties are also presented in [FC97b]. The properties shown in this section reveals the relationships of the basic Boolean Formulae for arithmetic and Reed-Muller spectra. Those relationships can be also used for the calculation of the generalized arithmetic or Reed-Muller spectra when the elementary spectra of the Boolean functions are in the same polarity, but the properties shown in this section can not be used directly for the calculation of the generalized arithmetic or Reed-Muller spectra if the elementary spectra are in different polarities. The algorithm to convert the generalized arithmetic spectrum from one polarity to another polarity has been shown in [FC97b], and similar algorithm can be also used for the generalized Reed-Muller transform due to the links between generalized arithmetic and Reed-Muller transforms. Therefore, for the cases that elementary spectra are in different polarities, we need to use the algorithm shown in [FC97b] to convert the spectra to desired polarity, then we can use the properties shown in this section to calculate the spectrum of the Boolean Formulae.

Example 4.2.6. Let $\mathbf{S}_{\mathbf{A}_a}^{\omega_a}$ and $\mathbf{S}_{\mathbf{A}_b}^{\omega_b}$ be the generalized arithmetic spectra in polarities ω_a and ω_b of f_a and f_b . To get the generalized spectrum $\mathbf{S}_{\mathbf{A}}^{a \wedge b}$ in polarity ω_a when $\mathbf{S}_{\mathbf{A}_a}^{\omega_a}$ and $\mathbf{S}_{\mathbf{A}_b}^{\omega_b}$ are given, the procedure of the calculation is shown below.

Step 1: Convert the generalized arithmetic spectrum $\mathbf{S}_{\mathbf{A}_b}^{\omega_b}$ to the spectrum $\mathbf{S}_{\mathbf{A}_b}^{\omega_a}$ in polarity ω_a by using algorithms shown in [FC97b].

Step 2: Construct the matrix $\mathbf{U}^{f_a}(n)$ from the generalized arithmetic spec-

trum $\mathbf{S}_{\mathbf{A}_a}^{\omega_a}$.

Step 3: The generalized arithmetic spectrum $\mathbf{S}_{\mathbf{A}}^{a \wedge b}$ in polarity ω_a can be calculated as: $\mathbf{S}_{\mathbf{A}}^{a \wedge b} = \mathbf{U}^{f_a} \mathbf{S}_{\mathbf{A}_b}^{\omega_b}$ by using Property 4.2.5.

4.2.3 Analysis on the Complexities of Algorithms

In this section, the complexities of conventional method and proposed method will be compared. Since the proposed methods for generalized arithmetic and Reed-Muller spectra are based on the properties for the calculation of zero-polarity arithmetic and Reed-Muller spectra, we will concentrate on the analysis of the complexities of the algorithms shown in Section 4.2.2 for the arithmetic and Reed-Muller transforms.

The procedures to calculate the arithmetic or Reed-Muller spectrum of Boolean formula using conventional method and proposed method has been described in Section 4.2.2. The block diagrams of the procedure for both traditional method and the proposed method are shown in Fig. 4.1. It can be seen from the block diagrams that the procedure of traditional method includes the transformations between spectral domain and Boolean domain, while the transformations are not required when using proposed method.

According to the layered Kronecker structure of the arithmetic and Reed-Muller transforms, the number of non-zero entries of arithmetic and Reed-Muller transform can be known as 3^n in the transform matrices. Due to the transformations between functional and spectral domains are required when using the conventional method to calculate the arithmetic or Reed-Muller spectrum of the Boolean Formulae, the time complexity is: $2^{n+1} \times 3^n = 2 \times 6^n$ for the

Figure 4.1: The block diagrams of the procedures for both conventional method and the proposed method to calculate arithmetic or Reed-Muller spectrum.

conventional method.

Let Z_f represent the number of non-zero entries of $R_{i,j}a(j)$ for i and j are integers from 0 to $n - 1$. Since the number of non-zero entries of $R(n)$ is 3^n , we can have $0 \leq Z_f \leq 3^n$. The time complexity is $2^n Z_f$ for using the proposed method to calculate the arithmetic or Reed-Muller spectrum of the Boolean Formulae. When there is no zero-entries in the arithmetic spectrum $\mathbf{S}_A$, Z_f has the maximum value 3^n . Hence, the time complexity for the worst case using the proposed method is $2^n \times 3^n = 6^n$. We can see that the time complexity using proposed method is just half of the one using conventional method even for the worst case.

Example 4.2.7. The algorithms using conventional and proposed methods for the arithmetic spectrum of logic product when $n = 3$ are given in Example 4.2.1. From the analysis of the algorithm for conventional method, we can know that the number of loops needed in the algorithm is 432. Comparatively, the number of loops needed in the algorithm for proposed method is $8Z_{f_b}$ where Z_{f_b} is the number of non-zero entries of $R_{i,j}s_{A_b}(j)$ for $0 \leq i \leq 2$ and $0 \leq j \leq 2$ and $0 \leq Z_{f_b} \leq 27$. Hence, the number of maximum loops needed in the algorithm

Figure 4.2: Number of cases in all 4-variable Boolean functions for different values of Z_f (for the calculation of Reed-Muller spectrum).

for proposed method is 216.

From comparing the conventional and proposed methods to calculate the arithmetic or Reed-Muller spectrum of Boolean Formula, the reduction in running time when using proposed method is: $\left(1 - \frac{Z_f}{2 \times 3^n}\right) \times 100\%$ where $0 \leq Z_f \leq 3^n$. It can be noted that the percentage of the reduction in running time when using proposed method is dependent on the value of Z_f , so the proposed method always saves at least 50% in running time than the conventional method.

Let all the 2^{2^4} 4-variable Boolean functions be considered. From previous analysis, we have: $0 \leq Z_f \leq 81$. For all 65536 cases of 4-variable Boolean

Figure 4.3: Percentage of the cases in all 4-variable Boolean functions (for the calculation of Reed-Muller spectrum using proposed method).

functions, the result of Z_f for Reed-Muller spectrum using proposed method are simulated as shown in Fig. 4.2. According to the percentage of the reduction in running time when using proposed method to calculate Reed-Muller spectrum, the percentage of the cases in all 65536 4-variable Boolean functions can be simulated as shown in Fig. 4.3. Similarly, the analysis for the calculation of arithmetic spectrum using proposed method can be done. The percentage of the cases in all 4-variable Boolean functions is shown in Fig. 4.4, which also shows the reduction in running time when using proposed method is much significant than using conventional method.

The previous analysis on the time complexity is based on the algorithmic

Figure 4.4: Percentage of the cases in all 4-variable Boolean functions (for the calculation of arithmetic spectrum using proposed method).

aspect. Since Walsh transform has been implemented by using lookup table (LUT) cascades [FS04], the arithmetic and Reed-Muller transforms can be also implemented similarly. Such hardware makes the arithmetic and Reed-Muller transforms to be easily implemented by parallel computing for the spectral coefficients. The block diagram of the calculation of arithmetic or Reed-Muller spectrum using conventional method is shown in Fig. 4.5. The matrix $\mathbf{U}^f$ and $\mathbf{V}^f$ used in proposed method can be also implemented by LUT cascades, and the block diagram of the calculation of arithmetic or Reed-Muller spectrum using proposed method is shown in Fig. 4.6. Comparing Fig. 4.5 and Fig. 4.6, the advantages in reducing the time complexity and hardware implementation

Figure 4.5: The block diagram of the calculation of arithmetic or Reed-Muller spectrum using conventional method.

Figure 4.6: The block diagram of the calculation of arithmetic or Reed-Muller spectrum using proposed method.

of the proposed method can be seen due to smaller number of steps.

4.3 Calculation of Composite Spectra of Haar Transform

Since the applications of Haar transform in logic design mainly concerns about the non-normalized Haar transform, we study the relations of Boolean functions for non-normalized Haar spectra only and the Haar transform and Haar spec-

trum mentioned in this section are referred to the normalized Haar transform and spectrum. Let function $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$ be the n -variable Boolean functions, and $\mathbf{S}_{\mathbf{H}_a}$ and $\mathbf{S}_{\mathbf{H}_b}$ be the corresponding Haar spectra of these Boolean functions.

Property 4.3.1. Let the logic product of $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$ be represented by $f^{a \wedge b}(\mathbf{x}_n)$. The Haar spectrum $\mathbf{S}_{\mathbf{H}}^{a \wedge b}$ of function $f^{a \wedge b}(\mathbf{x}_n)$ can be generated from the Haar spectra of $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$ as follows:

$$\mathbf{S}_{\mathbf{H}}^{a \wedge b} = \frac{1}{2^n} \mathbf{Q}^{f_a}(n) \cdot \mathbf{S}_{\mathbf{H}_b} \quad (4.21)$$

where $\mathbf{Q}^{f_a}(n)$ is a $2^n \times 2^n$ matrix, i.e.,

$$\mathbf{Q}^{f_a}(n) = \begin{bmatrix} Q_{0,0}^{f_a} & Q_{0,1}^{f_a} & Q_{0,2}^{f_a} & \cdots & Q_{0,2^{n-2}}^{f_a} & Q_{0,2^{n-1}}^{f_a} \\ Q_{1,0}^{f_a} & Q_{1,1}^{f_a} & Q_{1,2}^{f_a} & \cdots & Q_{1,2^{n-2}}^{f_a} & Q_{1,2^{n-1}}^{f_a} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ Q_{2^{n-1},0}^{f_a} & Q_{2^{n-1},1}^{f_a} & Q_{2^{n-1},2}^{f_a} & \cdots & Q_{2^{n-1},2^{n-2}}^{f_a} & Q_{2^{n-1},2^{n-1}}^{f_a} \end{bmatrix},$$

the element of the matrix $Q_{i,j}^{f_a}(n)$ is defined as follows:

$$Q_{i,j}^{f_a}(n) = \begin{cases} s_{H_a}(i \oplus j) & \text{if } i \leq 1, j \leq 1; \\ (-1)^{i \cdot j k} 2^k s_{H_a}(j) \text{ where } j \in [2^k, 2^{k+1}) & \text{if } i \leq 1, j > 1; \\ (-1)^{i k \cdot j} s_{H_a}(i) \text{ where } i \in [2^k, 2^{k+1}) & \text{if } i > 1, j \leq 1; \\ s_{H_a}(0) + \sum_{m=1}^k (-1)^{i m} 2^{m-1} h[\overbrace{(j_{m+1} \cdots j_2)}^m]_2 \text{ where } j \in [2^k, 2^{k+1}) & \text{if } i > 1, j = i; \\ (-1)^{i m} 2^{k+m} s_{H_a}(j) \text{ where } i \in [2^k, 2^{k+1}), m \in [1, n-k-1] & \text{if } i \in [2, 2^{n-1}) \text{ and} \\ & j \in [(\overbrace{j_{k+1} \cdots j_1}^{k+1} \overbrace{0 \cdots 0}^m)_2, (\overbrace{j_{k+1} \cdots j_1}^{k+1} \overbrace{1 \cdots 1}^m)_2]; \\ (-1)^{i k - m} 2^m s_{H_a}(i) \text{ where } i \in [2^k, 2^{k+1}), m \in [1, k-1] & \text{if } i \geq 4 \text{ and } j = \overbrace{(i_{k+1} i_k \cdots i_{k+1-m})}^{m+1}_2; \\ 0 & \text{otherwise.} \end{cases}$$

In (4.21), $i, j \in [0, 2^n - 1]$, i and j can be represented in binary format as $(i_n i_{n-1} \cdots i_1)_2$ and $(j_n j_{n-1} \cdots j_1)_2$.

From the commutative law in Boolean algebra, (4.21) can be also expressed as: $\mathbf{S}_H^{a \wedge b} = \mathbf{S}_H^{b \wedge a} = \frac{1}{2^n} \mathbf{Q}^{f_b}(n) \cdot \mathbf{S}_{H_a}$.

As Property 4.3.1 describes the relationships in spectral domain of the logic product, it can be seen as the convolution theorem for Haar transform.

Example 4.3.1. For the 2-variable Boolean function, the corresponding Haar spectra of the logic product can be generated using (4.22). The corresponding matrix $\mathbf{Q}^{f_a}(2)$ in the equation is:

$$\mathbf{Q}^{f_a}(2) = \begin{bmatrix} s_{H_a}(0) & s_{H_a}(1) & 2s_{H_a}(2) & 2s_{H_a}(3) \\ s_{H_a}(1) & s_{H_a}(0) & 2s_{H_a}(2) & -2s_{H_a}(3) \\ s_{H_a}(2) & s_{H_a}(2) & s_{H_a}(0) + s_{H_a}(1) & 0 \\ s_{H_a}(3) & -s_{H_a}(3) & 0 & s_{H_a}(0) - s_{H_a}(1) \end{bmatrix}.$$

The construction of the matrix $\mathbf{Q}^{f_a}(n)$ which is defined in (4.22) is the core element of the evaluation of the logic product, so the analysis of the general structure of $\mathbf{Q}^{f_a}(n)$ is an important issue. The following property states the relationships between the structures of $\mathbf{Q}^{f_a}(n)$ with different orders.

Property 4.3.2. The matrix $\mathbf{Q}^{f_a}(n)$ is constructed by operating the Haar spectrum of the Boolean function. Since $\mathbf{Q}^{f_a}(n)$ can be regarded as the matrix constructed by four sub-matrices, i.e., $\mathbf{Q}^{f_a}(n) = \begin{bmatrix} \mathbf{Q}_a^{f_a}(n) & \mathbf{Q}_b^{f_a}(n) \\ \mathbf{Q}_c^{f_a}(n) & \mathbf{Q}_d^{f_a}(n) \end{bmatrix}$, then the top left sub-matrix $\mathbf{Q}_a^{f_a}(n)$ has the same structure as the matrix $\mathbf{Q}^{f_a}(n-1)$.

Example 4.3.2. The matrix $\mathbf{Q}^{f_a}(3)$ can be evaluated using (4.22) as shown below:

$$\mathbf{Q}^{f_a}(3) = \begin{bmatrix} s_{H_a}(0) & s_{H_a}(1) & 2s_{H_a}(2) & 2s_{H_a}(3) & 4s_{H_a}(4) \\ s_{H_a}(1) & s_{H_a}(0) & 2s_{H_a}(2) & -2s_{H_a}(3) & 4s_{H_a}(4) \\ s_{H_a}(2) & s_{H_a}(2) & s_{H_a}(0) + s_{H_a}(1) & 0 & 4s_{H_a}(4) \\ s_{H_a}(3) & -s_{H_a}(3) & 0 & s_{H_a}(0) - s_{H_a}(1) & 0 \\ s_{H_a}(4) & s_{H_a}(4) & 2s_{H_a}(4) & 0 & s_{H_a}(0) + s_{H_a}(1) + 2s_{H_a}(2) \\ s_{H_a}(5) & s_{H_a}(5) & -2s_{H_a}(5) & 0 & 0 \\ s_{H_a}(6) & -s_{H_a}(6) & 0 & 2s_{H_a}(6) & 0 \\ s_{H_a}(7) & -s_{H_a}(7) & 0 & -2s_{H_a}(7) & 0 \\ 4s_{H_a}(5) & 4s_{H_a}(6) & 4s_{H_a}(7) \\ 4s_{H_a}(5) & -4s_{H_a}(6) & -4s_{H_a}(7) \\ -4s_{H_a}(5) & 0 & 0 \\ 0 & 4s_{H_a}(6) & -4s_{H_a}(7) \\ 0 & 0 & 0 \\ s_{H_a}(0) + s_{H_a}(1) - 2s_{H_a}(2) & 0 & 0 \\ 0 & s_{H_a}(0) - s_{H_a}(1) + 2s_{H_a}(3) & 0 \\ 0 & 0 & s_{H_a}(0) - s_{H_a}(1) - 2s_{H_a}(3) \end{bmatrix},$$

while the general structure of $\mathbf{Q}^{f_a}(2)$ which is used for the 2-variable Boolean functions is shown in Example 4.3.1.

Comparing the structure of matrices $\mathbf{Q}^{f_a}(2)$ and $\mathbf{Q}^{f_a}(3)$, we can find that the top left part $\mathbf{Q}_a^{f_a}(3)$ of the matrix $\mathbf{Q}^{f_a}(3)$ has the same structure as the matrix $\mathbf{Q}^{f_a}(2)$.

Property 4.3.3. Let the logic sum of $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$ be represented by $f^{a \vee b}(\mathbf{x}_n)$. The Haar spectrum $\mathbf{S}_H^{a \vee b}$ of function $f^{a \vee b}(\mathbf{x}_n)$ can be generated from

the Haar spectra of $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$ as follows:

$$\begin{aligned}
 \mathbf{S}_{\mathbf{H}}^{a \vee b} &= \mathbf{S}_{\mathbf{H}_a} + \mathbf{S}_{\mathbf{H}_b} - \mathbf{S}_{\mathbf{H}}^{a \wedge b} \\
 &= \mathbf{S}_{\mathbf{H}_a} + \mathbf{S}_{\mathbf{H}_b} - \frac{1}{2^n} \mathbf{Q}^{f_a}(n) \cdot \mathbf{S}_{\mathbf{H}_b} \\
 &= \mathbf{S}_{\mathbf{H}_a} + (\mathbf{I}_{2^n} - \frac{1}{2^n} \mathbf{Q}^{f_a}(n)) \cdot \mathbf{S}_{\mathbf{H}_b} .
 \end{aligned} \tag{4.22}$$

Property 4.3.4. Let the EXOR sum of $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$ be represented by $f^{a \oplus b}(\mathbf{x}_n)$. The Haar spectrum $\mathbf{S}_{\mathbf{H}}^{a \oplus b}$ of function $f^{a \oplus b}(\mathbf{x}_n)$ can be generated from the Haar spectra of $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$ as follows:

$$\begin{aligned}
 \mathbf{S}_{\mathbf{H}}^{a \oplus b} &= \mathbf{S}_{\mathbf{H}_a} + \mathbf{S}_{\mathbf{H}_b} - 2\mathbf{S}_{\mathbf{H}}^{a \wedge b} \\
 &= \mathbf{S}_{\mathbf{H}_a} + \mathbf{S}_{\mathbf{H}_b} - \frac{1}{2^{n-1}} \mathbf{Q}^{f_a}(n) \cdot \mathbf{S}_{\mathbf{H}_b} \\
 &= \mathbf{S}_{\mathbf{H}_a} + (\mathbf{I}_{2^n} - \frac{1}{2^{n-1}} \mathbf{Q}^{f_a}(n)) \cdot \mathbf{S}_{\mathbf{H}_b} .
 \end{aligned} \tag{4.23}$$

Property 4.3.5. Let the negation of $f(\mathbf{x}_n)$ be represented by $\bar{f}(\mathbf{x}_n)$. The Haar spectrum $\bar{\mathbf{S}}_{\mathbf{H}}$ of function $\bar{f}(\mathbf{x}_n)$ can be generated from the Haar spectra of $f(\mathbf{x}_n)$ as follows:

$$\bar{\mathbf{S}}_{\mathbf{H}} = 2^n \mathbf{K}(n) - \mathbf{S}_{\mathbf{H}} \tag{4.24}$$

where $\mathbf{K}(n)$ is an $2^n \times 1$ matrix and $\mathbf{K}(n) = [1, 0, 0, \dots, 0]^T$.

When the Haar spectra of the Boolean functions have already been calculated, it is more convenient to use the above properties to evaluate the Haar spectrum of the Boolean formula composed by Boolean functions. Using conventional method, firstly we calculate the truth vectors of the Boolean functions from the known Haar spectra through inverse Haar transform applied to each

Boolean function separately, then the truth vector of the Boolean formula can be obtained by the truth vectors of the Boolean functions. When the truth vector of the Boolean formula is evaluated, the corresponding Haar spectrum can be obtained by using (2.33). The properties shown in this section can be used to simplify the calculation of the composite Haar spectrum, since we do not need the transformations between Boolean domain and spectral domain which is necessary in conventional method. It can be noticed that the construction of the matrix $\mathbf{Q}^{f_a}(n)$ needs some multiplication operations. Generally, the multiplication operations have great effect in the computational efficiency, but the multiplication operations in $\mathbf{Q}^{f_a}(n)$ are always powers-of-two (2^q) related to bit-shift operations that can be easily implemented in hardware by simply shifting the original Haar spectra by q bits. Hence, the computation complexity of the method using those properties is still much lower than the one using the conventional method. The block diagrams of the procedure for both conventional method and the method using the properties are shown in Fig. 4.7. It can be seen from the block diagrams that the procedure of conventional method includes the transformations between spectral domain and Boolean domain, so the corresponding inverse and forward Haar transform matrices are used in these calculations. The fast algorithm of Haar transform has been widely used to reduce the computation complexity of the calculation for Haar transform, but it still requires extra time when compared with our new method.

The procedure of conventional method includes three steps which need longer processing time than the one required by proposed method. It can be seen from Fig. 4.1 that the time complexity of the proposed method is lower than the one using conventional method due to smaller number of steps. When the fast algorithm of Haar transform is used to reduce the computation complexity of

Figure 4.7: The block diagrams of the procedure for both conventional method and the proposed method to calculate Haar spectrum.

the calculation in conventional method, the time complexity increases. Hence, the proposed method shows more advantages in time complexity than the conventional method where the fast algorithm of Haar transform is used in the practical applications.

An example is shown below for the comparison between the calculations using conventional and proposed methods.

Example 4.3.3. Let f_1 and f_2 be 3-variable Boolean functions, and the corresponding truth vectors are $\mathbf{F}_1$ and $\mathbf{F}_2$, respectively. The corresponding Haar spectra of f_1 and f_2 are known as $\mathbf{S}_{\mathbf{H}_1} = [5, 1, -1, 0, 1, 0, 1, -1]^T$ and $\mathbf{S}_{\mathbf{H}_2} = [5, -1, -2, -1, 0, 0, -1, 0]^T$. The evaluations of the Haar spectrum of $f_1 \oplus f_2$ are shown below by conventional and proposed methods.

Conventional Method:

Step 1: The inverse Haar transform is needed to get the truth vectors of f_1 and f_2 from the corresponding Haar spectra. The inverse Haar transform

$\mathbf{H}^{-1}(3)$ can be obtained as:

$$\mathbf{H}^{-1}(3) = \frac{1}{8} \begin{bmatrix} 1 & 1 & 2 & 0 & 4 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & -4 & 0 & 0 & 0 \\ 1 & 1 & -2 & 0 & 0 & 4 & 0 & 0 \\ 1 & 1 & -2 & 0 & 0 & -4 & 0 & 0 \\ 1 & -1 & 0 & 2 & 0 & 0 & 4 & 0 \\ 1 & -1 & 0 & 2 & 0 & 0 & -4 & 0 \\ 1 & -1 & 0 & -2 & 0 & 0 & 0 & 4 \\ 1 & -1 & 0 & -2 & 0 & 0 & 0 & -4 \end{bmatrix}.$$

Thus, the truth vectors $\mathbf{F}_1$ and $\mathbf{F}_2$ are evaluated as follows:

$$\mathbf{F}_1 = \mathbf{HAR}^{-1}(3) \cdot \mathbf{S}_{\mathbf{H}_1} = [1, 0, 1, 1, 1, 0, 0, 1]^T,$$

$$\mathbf{F}_2 = \mathbf{HAR}^{-1}(3) \cdot \mathbf{S}_{\mathbf{H}_2} = [0, 0, 1, 1, 0, 1, 1, 1]^T.$$

Step 2: Let the truth vector of $f_1 \oplus f_2$ be represented by $\mathbf{F}^{1\oplus 2}$. The EXOR sum of Boolean functions is described in (2.12), so the truth vector $\mathbf{F}^{1\oplus 2}$ can be obtained: $\mathbf{F}^{1\oplus 2} = [1, 0, 0, 0, 1, 1, 1, 0]^T$.

Step 3: By using (2.32), the forward Haar transform $\mathbf{H}(3)$ can be obtained

as:

$$\mathbf{H}(3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}.$$

The Haar spectrum $\mathbf{H}^{1\oplus 2}$ can be evaluated by using (2.33) as:

$$\mathbf{S}_{\mathbf{H}}^{1\oplus 2} = \mathbf{H}(3) \cdot \mathbf{F}^{1\oplus 2} = [4, -2, 1, 1, 1, 0, 0, 1]^T.$$

Proposed Method:

The evaluation of the Haar spectrum of EXOR sum of two Boolean functions can be processed as shown in Property 4.3.4.

Step 1: The general structure of the matrix $\mathbf{Q}^{f_1}(3)$ is shown in Example 4.3.2 and the Haar spectra of f_1 and f_2 are known. As f_1 here corresponds to f_a in Example 4.3.2, so we can construct the corresponding matrix of size $n = 3$

by the Haar spectrum of f_1 as:

$$\mathbf{Q}^{f_1}(3) = \begin{bmatrix} 5 & 1 & -2 & 0 & 4 & 0 & 4 & -4 \\ 1 & 5 & -2 & 0 & 4 & 0 & -4 & 4 \\ -1 & -1 & 6 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 4 & 4 \\ 1 & 1 & 2 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 1 & -1 & 0 & 2 & 0 & 0 & 4 & 0 \\ -1 & 1 & 0 & 2 & 0 & 0 & 0 & 4 \end{bmatrix}.$$

Step 2: By using (4.23), the Haar spectrum $\mathbf{S}_{\mathbf{H}}^{1\oplus 2}$ can be obtained as:

$$\mathbf{S}_{\mathbf{H}}^{1\oplus 2} = \mathbf{S}_{\mathbf{H}_1} + \mathbf{S}_{\mathbf{H}_2} - \frac{1}{4} \mathbf{Q}^{f_1}(3) \cdot \mathbf{S}_{\mathbf{H}_2} = [4, -2, 1, 1, 1, 0, 0, 1]^T.$$

Alternatively, from the commutative law the Haar spectrum $\mathbf{H}^{3\oplus 4}$ can be also obtained by:

$$\mathbf{S}_{\mathbf{H}}^{1\oplus 2} = \mathbf{S}_{\mathbf{H}_1} + \mathbf{S}_{\mathbf{H}_2} - \frac{1}{4} \mathbf{Q}^{f_2}(3) \cdot \mathbf{S}_{\mathbf{H}_1}.$$

In this case, the computational costs required by applying $\mathbf{Q}^{f_2}(3)$ are lower than using $\mathbf{Q}^{f_1}(3)$ due to the fact that the matrix $\mathbf{Q}^{f_2}(3)$ has more zero elements than $\mathbf{Q}^{f_1}(3)$ since $\mathbf{S}_{\mathbf{H}_2}$ has more coefficients equal to zero than $\mathbf{S}_{\mathbf{H}_1}$. In practical applications, we can choose the Haar spectra which has bigger number of zero coefficients to construct the corresponding matrix with more zero elements which results in the final lower computational costs.

4.4 Summary

The methods to obtain the composite arithmetic, Reed-Muller, and Haar spectra including the Haar spectra of logic product, logic sum, EXOR sum and negation of Boolean functions are shown in this chapter. Based on the proposed properties, the calculations for the composite spectra are all performed in the spectral domain. Comparatively, the transformations between spectral and Boolean domains lead to the higher computation and time complexities by the conventional method.

Chapter 5

Developments for Sign Transforms

5.1 Introduction

Sign transform is an important family of invertible nonlinear transforms and provides considerable reductions in computing time and storage requirements with keeping the original spectral transform's most useful properties. An efficient method to calculate sign Walsh spectra from disjoint cube representation of logic functions was shown in [FK02]. The sign Walsh transform and sign Haar transform has been used in the applications of logic synthesis and testing [BT96, DiT91, FK02, Sas93, SSS96, FR94, FR03]. In this chapter, the definitions of sign Walsh and sign Haar transforms will be reviewed in Sections 5.2. The improvements on the sign Walsh and Haar transforms and the developments of the new sign Haar transforms will be given in the rest.

The following symbols are used in this chapter: Let $\mathbf{x}_n = \{x_n, x_{n-1}, \dots, x_p, \dots, x_2,$

$x_1\}$, $\omega_n = \{\omega_n, \omega_{n-1}, \dots, \omega_p, \dots, \omega_2, \omega_1\}$ be n -tuples over $\text{GF}(2)$. The symbol x_p stands for a data variable, ω_p represents a sign transform variable, p is an integer and $1 \leq p \leq n$. Let $\mathbf{F} = [F_0, F_1, \dots, F_j, \dots, F_{N-2}, F_{N-1}]$ be a ternary vector where $N = 2^n$. For example, it can be the S-coded truth vector of $f : (0, 1)^n \rightarrow (-1, 0, 1)$ where the value of F_j ($0 \leq j < N$) is given by $f(\mathbf{x}_n)$ when $\sum_{p=1}^n x_p 2^{p-1} = j$. Let $\mathbf{O}_i$ represent the vector of i zeros, $1 \leq i \leq n$. The sign

$$\text{function is calculated as: } \text{sign } z = \begin{cases} -1 & \text{if } z < 0, \\ 0 & \text{if } z = 0, \\ 1 & \text{if } z > 0. \end{cases}$$

Let the symbol $\oplus_c$ represent cyclic addition, the symbol $\oplus_d$ represent dyadic addition, and the symbol $\triangle$ represent bit-by-bit logical AND. When the above operations are applied to two vectors $\mathbf{A}_j$ and $\mathbf{B}_k$, $1 \leq j < k$, j and k are two different integer numbers, they result in the vector $\mathbf{C}_k$ of the length k . Only j elements of $\mathbf{B}_k$ and all elements of $\mathbf{A}_j$ are manipulated on, remaining $(k - j)$ elements of the resulting vector $\mathbf{C}_k$ are not affected by the applied operation and are simply the same as the elements of the vector $\mathbf{B}_k$ between positions k and $j + 1$.

5.2 Review of Sign Walsh and Sign Haar Transforms

Let $\mathbf{SW} = [sw_0, sw_1, \dots, sw_j, \dots, sw_{N-2}, sw_{N-1}]$ and $\mathbf{SH} = [sh_0, sh_1, \dots, sh_j, \dots, sh_{N-2}, sh_{N-1}]$ be the vector corresponding to sign Walsh spectrum and sign Haar spectrum of $\mathbf{F}$, accordingly. The value of sw_j ($0 \leq j < N$) is given by $\mathbf{SW}(\omega_n)$ when $\sum_{p=1}^n \omega_p 2^{p-1} = j$. The value of sh_j ($0 \leq j < N$) is given by

$\mathbf{SH}(\omega_n)$ when $\sum_{p=1}^n \omega_p 2^{p-1} = j$.

Definition 5.2.1. An invertible forward sign Walsh transform $\mathbf{SW}$ is [BT96, DiT91, Sas93, SSS96]:

$$sw(\omega_n) = \text{sign} \sum_{x_n=0}^1 [\text{sign} \sum_{x_{n-1}=0}^1 [\text{sign} \sum_{x_{n-2}=0}^1 [\dots \text{sign} \sum_{x_1=0}^1 [f(\mathbf{x}_n) (-1)^{\sum_{p=1}^n \omega_p x_p} \dots]]]. \quad (5.1)$$

The inverse sign Walsh transform is [BT96, DiT91, Sas93, SSS96]:

$$f(\mathbf{x}_n) = \text{sign} \sum_{\omega_1=0}^1 [\text{sign} \sum_{\omega_2=0}^1 [\text{sign} \sum_{\omega_3=0}^1 [\dots \text{sign} \sum_{\omega_n=0}^1 [sw(\omega_n) (-1)^{\sum_{p=1}^n \omega_p x_p} \dots]]]. \quad (5.2)$$

In (5.1) and (5.2), $1 \leq p \leq n$.

The fast diagrams for calculations of forward and inverse sign Walsh transforms for $n = 3$ are shown in Fig. 5.1(a) and Fig. 5.1(b), respectively. In all the figures for fast diagrams of this chapter, note that the solid lines and dotted lines represent additions and subtraction, respectively, and the symbols $\bullet$ and $\circ$ represent application of sign function and lack of any operation, respectively.

Definition 5.2.2. An invertible forward sign Haar transform $\mathbf{SH}$ is [FR03]:

$$sh(\mathbf{O}_n \oplus_d \omega_1) = \text{sign} \sum_{x_n=0}^1 [\text{sign} \sum_{x_{n-1}=0}^1 [\text{sign} \sum_{x_{n-2}=0}^1 [\dots \text{sign} \sum_{x_1=0}^1 [(-1)^{\omega_1 x_n} f(\mathbf{x}_n)] \dots]]], \quad (5.3)$$

and

$$sh(\mathbf{O}_n \oplus_d \omega_i \oplus_d 2^i) = \text{sign} \sum_{x_{n-i}=0}^1 [\text{sign} \sum_{x_{n-i-1}=0}^1 [\dots \text{sign} \sum_{x_1=0}^1 [(-1)^{x_{n-1}} \dots f\{[(\mathbf{O}_n \oplus_d \omega_i) \oplus_c (n-i)] \oplus_d \mathbf{x}_{n-i}\}]]] \quad (5.4)$$

Figure 5.1: Butterfly diagrams for the forward and inverse sign Walsh transforms where $n = 3$.

where $1 \leq i < n$.

The inverse sign Haar transform is [FR03]:

$$\begin{aligned}
 f(\mathbf{x}_n) = & \text{sign}\{(-1)^{x_1} h\{[(\mathbf{O}_1 \triangle \mathbf{x}_n) \oplus_c 1] \oplus_d 2^{n-1}\} \\
 & + \text{sign}\{(-1)^{x_2} h\{[(\mathbf{O}_2 \triangle \mathbf{x}_n) \oplus_c 2] \oplus_d 2^{n-2}\} + \dots \\
 & + \text{sign}\{(-1)^{x_i} h\{[(\mathbf{O}_i \triangle \mathbf{x}_n) \oplus_c i] \oplus_d 2^{n-i}\} + \dots \\
 & + \text{sign}\{(-1)^{x_{n-1}} h\{[(\mathbf{O}_{n-1} \triangle \mathbf{x}_n) \oplus_c (n-1)] \oplus_d 2\} \\
 & + \text{sign}\left[\sum_{\omega_1=0}^1 (-1)^{x_n \omega_1} h(\mathbf{O}_n \oplus_d \omega_1)\right] \dots \}. \tag{5.5}
 \end{aligned}$$

Fig. 5.2(a) and Fig. 5.2(b) show the butterfly diagrams for forward and inverse 8-point ($n = 3$) sign Haar transforms.

Figure 5.2: Butterfly diagrams for the forward and inverse sign Haar transforms where $n = 3$.

5.3 Fixed Sign Walsh Transform and Its Iterative Hardware Architecture

5.3.1 General Definitions of Quantized Transforms

In S-coding [HMM85, Hur92], the binary vectors formed of only $\{+1, -1\}$ represent logical values of completely specified Boolean functions, and formed of $\{+1, 0, -1\}$ the values of incompletely specified Boolean functions. In the continuation, the functional and spectral data will be represented by $\{+, -\}$ and $\{+, 0, -\}$ to shorten the notations, respectively. Let $R_1 = \{+, -\}$, $R_2 = \{+, 0, -\}$, and R_l^n means n -space Cartesian product of a set R_l where $(l = 1, 2)$.

Property 5.3.1. An n -variable S-coded completely specified Boolean function can be considered as the mapping: $R_1^n \rightarrow R_1$.

Property 5.3.2. An n -variable S-coded incompletely specified Boolean function can be considered as the mapping: $R_1^n \rightarrow R_2$.

Property 5.3.3. When the input is the truth vector of n -variable S-coded incompletely specified Boolean function, the forward and inverse sign Walsh transforms can be seen as the mappings: $(R_2^N)_F \rightarrow (R_2^N)_S$ and $(R_2^N)_S \rightarrow (R_2^N)_F$ where the subscripts F and S correspond to functional and spectral domains, respectively. When the input is the truth vector of n -variable S-coded incompletely specified Boolean function, the forward and inverse ternary Walsh transforms can be seen as the mappings: $(R_1^N)_F \rightarrow (R_2^N)_S$ and $(R_2^N)_S \rightarrow (R_1^N)_F$, respectively.

5.3.2 Fixed Fast Walsh Transform Structure

A fixed butterfly diagram structure consists of the same parts for every level, such fast Walsh transform [Bea75] is named as fixed fast Walsh transform. The general form of fixed fast Walsh transform is shown below.

Property 5.3.4. The forward Walsh transform can be factorized using fixed fast Walsh transform as:

$$\mathbf{W}(n) = \prod_{i=1}^n \mathbf{D}(n) \quad (5.6)$$

where $\mathbf{D}(n) = \begin{bmatrix} \mathbf{I}_{2^{n-1}} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ \mathbf{I}_{2^{n-1}} \otimes \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \end{bmatrix}$ and $\mathbf{I}_{2^{n-1}}$ is the identity matrix of order 2^{n-1} .

Property 5.3.5. The inverse Walsh transform can be factorized using fixed fast Walsh transform as:

$$\mathbf{W}^{-1}(n) = \frac{1}{2^n} \prod_{i=1}^n \mathbf{E}(n) \quad (5.7)$$

$$\text{where } \mathbf{E}(n) = \left[\mathbf{I}_{2^{n-1}} \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{I}_{2^{n-1}} \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right].$$

Property 5.3.6. The matrix $\mathbf{E}(n)$ in (5.7) is the transpose of the matrix $\mathbf{D}(n)$ in (5.6), and $\mathbf{E}(n)$ is the inverse of $\mathbf{D}(n)$ with scaling factor of $\frac{1}{2}$. The above relationships are shown as follows:

$$\mathbf{E}(n) = \mathbf{D}^T(n) \quad (5.8)$$

$$\text{and } \mathbf{E}(n) = \frac{1}{2} \mathbf{D}^{-1}(n) \quad (5.9)$$

where the superscript T is the matrix transpose operator.

It should be noticed that the construction of the factorized matrix $\mathbf{D}(n)$ is similar to the factorized matrix of fast Haar transform, so the basic butterfly part of the butterfly diagram for the forward fixed Walsh transform is the same as the leftmost part of the butterfly diagram for forward Haar transform. Similarly, the basic butterfly part for the inverse fixed Walsh transform is the same as the rightmost part of the butterfly diagram for the inverse Haar transform.

5.3.3 Definition of Fixed Sign Walsh Transform

Since the fixed fast Walsh transform structure makes the Walsh transform to be implemented easily, we can also use it for the hardware implementation of

the sign Walsh transform.

Definition 5.3.1. The fixed fast Walsh transform structure is used for the fixed sign Walsh transform together with the sign function as the quantizer at every point of the additions and subtractions in the butterfly diagram.

The spectrum calculated through fixed sign Walsh transform is exactly the same as the spectrum calculated through the standard sign Walsh transform, but the butterfly structure of the fixed sign Walsh transform provides more convenience than the one of standard sign Walsh transform in the hardware implementation. The fixed sign Walsh transform has the same equations as the equations shown in Definition 5.2.1.

Example 5.3.1. The forward and inverse fixed sign Walsh transforms of order 8 are shown in Fig. 5.3(a) and Fig. 5.3(b), respectively. From the comparison of Fig. 5.3(a) and Fig. 5.3(b), it can be noticed that the butterfly diagrams for the fixed sign Walsh transform can be easily implemented as they have the same three butterfly parts by using the iterative architecture.

5.3.4 Iterative Architecture for Fixed Sign Walsh Transform

The main idea of the sign transforms is that the sign function is used as the quantized function and both the functional and spectral domains are operating in $\{+1, 0, -1\}$. As shown in the previous section, the sign transforms can be implemented in hardware by using butterfly diagrams. As the interconnection pattern is the same for all computation stages of the fixed sign Walsh transform

Figure 5.3: Butterfly diagrams for the forward and inverse fixed sign Walsh transforms where $n = 3$.

of order 2^n , the hardware can be reduced to only one stage while the output is fed back to the input $n-1$ times.

The functional block using iterative architecture for fixed sign Walsh transform consists of several parts that include Input Register, Butterfly Structure, Sign Adders, Sign Subtractors, Output Register, and Control Unit. The Input Register is used to get the values fed back from the Output Register and send these values to the Butterfly Structure. The Butterfly Structure is corresponding to the basic butterfly part of the forward or inverse fixed sign Walsh transform of order 2^n , and the output values are computed by arithmetic elements (Sign Adders and Sign Subtractors) after the Butterfly Structure. In order to control the cycle of the feedback, the Control Unit switches the states of the modes of Input Register and Output Register. When the new data is ready to be supplied to the block, the New Data signal is sent to initialize the states of Mode 1 and Mode 2 to 0. When the state of Mode 1 is 0, the new data

Figure 5.4: Iterative architecture for the forward fixed sign Walsh transform where $n = 3$.

are supplied to the Butterfly Structure from Input Register. When the input of the block has already been supplied, the Mode 1 is set to 1 and then the Input Register get the data by the feedback from Output Register and supplies the data to Butterfly Structure. When the Mode 2 is 0, the Output Register feeds the data back to the Input Register. When the computations have been processed $n - 1$ times, Mode 2 is set to 1 and then the output values computed by the arithmetic elements are supplied as the output of the block, i.e., the final fixed Sign Walsh spectrum has been obtained. At the end of the conversion process, the control unit will send a Data Ready signal to report that the spectrum is available in the storage register. When the spectrum data are taken from the storage register, a Data Taken signal is sent to the Control Unit, which will then accept the new data for the next calculation sequence.

Example 5.3.2. The iterative architecture transform for forward fixed sign

Figure 5.5: Iterative architecture for the inverse fixed sign Walsh transform where $n = 3$.

Walsh transform is shown in Fig. 5.4 where the symbols $\boxplus$ and $\boxminus$ represent the Sign Adder and Sign Subtractor, respectively. It can be seen that the Butterfly Structure and the arithmetic elements in Fig. 5.4 follows exactly from the basic butterfly parts in Fig. 5.3(a). When a New Data signal is sent to the Control Unit, the input data of the block are supplied and Mode 1 and Mode 2 are set to 0. After the Input Register bypass the data, Mode 1 is set to 1. When the Output Register has fed back twice, Mode 2 is set to 1. The iterative architecture transform for inverse fixed sign Walsh transform is shown directly in Fig. 5.5. The Butterfly Structure in Fig. 5.5 corresponds to the basic butterfly part for the inverse fixed sign Walsh transform in Fig. 5.3(b).

5.4 Modified Sign Haar Transform

Let $\mathbf{MSH}=[msh_0, msh_1, \dots, msh_j, \dots, msh_{N-2}, msh_{N-1}]$ be the ternary vector corresponding to modified sign Haar spectrum of $\mathbf{F}$. The value of msh_j ($0 \leq j < N$) is given by $\mathbf{MSH}(\boldsymbol{\omega}_n)$ when $\sum_{p=1}^n \omega_p 2^{p-1} = j$.

5.4.1 Definitions of Modified Sign Haar Transform

Definition 5.4.1. An invertible forward modified sign Haar transform $\mathbf{MSH}$ is:

$$\begin{aligned}
 msh(\boldsymbol{\omega}_n) = & \text{sign}\left\{\sum_{x_n=0}^1 (-1)^{\omega_n x_n [1 - \text{sign}(\sum_{p=0}^{n-1} \omega_p)]} [1 - (x_n \oplus \omega_n) \right. \\
 & \text{sign}(\sum_{p=0}^{n-1} \omega_p)] \dots \text{sign}\left\{\sum_{x_i=0}^1 (-1)^{\omega_i x_i [1 - \text{sign}(\sum_{p=0}^{i-1} \omega_p)]} [1 - \right. \\
 & (x_i \oplus \omega_i) \text{sign}(\sum_{p=0}^{i-1} \omega_p)] \dots \text{sign}\left\{\sum_{x_2=0}^1 (-1)^{\omega_2 x_2 (1 - \omega_1)} [1 - \right. \\
 & \left. (x_2 \oplus \omega_2) \omega_1] \text{sign}\left[\sum_{x_1=0}^1 (-1)^{x_1 \omega_1} f(\mathbf{x}_n)\right]\right\} \dots \left. \right\}. \quad (5.10)
 \end{aligned}$$

The inverse modified sign Haar transform is:

$$\begin{aligned}
 f(\mathbf{x}_n) = & \text{sign}\left\{\sum_{\omega_1=0}^1 (-1)^{x_1 \omega_1} \text{sign}\left\{\sum_{\omega_2=0}^1 (-1)^{\omega_2 x_2 (1 - x_1)} [1 - \right. \right. \\
 & \left. \left. ((x_2 \oplus \omega_2) x_1) \dots \text{sign}\left\{\sum_{\omega_i=0}^1 (-1)^{\omega_i x_i [1 - \text{sign}(\sum_{p=0}^{i-1} x_p)]} [1 - \right. \right. \right. \\
 & \left. \left. \left. ((x_i \oplus \omega_i) \text{sign}(\sum_{p=0}^{i-1} x_p)] \dots \text{sign}\left\{\sum_{\omega_n=0}^1 (-1)^{\omega_n x_n [1 - \text{sign}(\sum_{p=0}^{n-1} x_p)]} \right. \right. \right. \\
 & \left. \left. \left. ([1 - (x_n \oplus \omega_n) \text{sign}(\sum_{p=0}^{n-1} x_p)] msh(\boldsymbol{\omega}_n)) \dots \right\} \dots \right\} \right\}. \quad (5.11)
 \end{aligned}$$

Figure 5.6: Butterfly diagrams for the forward and inverse modified sign Haar transforms where $n = 3$.

In Eqs. (5.10) and (5.11), $1 \leq p \leq n$, $1 \leq i \leq n$.

5.4.2 Analysis of Butterfly Structure

The modified sign Haar transform is related to sign Haar transform, but its butterfly structure is derived from sign Walsh transform. Fig. 5.6(a) and Fig. 5.6(b) show the butterfly diagrams for forward and inverse 8-point ($n = 3$) modified sign Haar transform. Comparing the butterfly diagrams for sign Walsh transform and modified sign Haar transform, it can be seen that the modified sign Haar transform has similar butterfly structure with sign Walsh transform. Therefore, the modified sign Haar transform has also the advantages of symmetric butterfly structure. For example, the 16-point ($n = 4$) modified Haar transform can be extended from 8-point ($n = 3$) modified Haar transform easily. Fig. 5.7(a) and Fig. 5.7(b) show the butterfly diagrams for forward and inverse 16-point modified sign Haar transform. In Fig. 5.7(a), the left three level of

Figure 5.7: Butterfly diagrams for the forward and inverse modified sign Haar transforms where $n = 4$.

the butterfly diagram can be considered as two blocks of forward 8-point modified sign Haar transform. Similar property also exists in inverse modified sign Haar transform. Using such structure, the modified sign Haar transform can be implemented by hardware and extended more conveniently than standard sign Haar transform.

The modified sign Haar transform can be considered as another version of standard sign Haar transform, since there are some relationships between the outputs of modified and standard sign Haar transforms. Comparing the butter-

Table 5.1: Relationship between 16-Point Modified and Standard Sign Haar Spectra

Standard sign Haar spectrum	Modified sign Haar spectrum
(0, 0, 0, 0)	(0, 0, 0, 0)
(0, 0, 0, 1)	(1, 0, 0, 0)
(0, 0, 1, 0)	(0, 1, 0, 0)
(0, 0, 1, 1)	(1, 1, 0, 0)
(0, 1, 0, 0)	(0, 0, 1, 0)
(0, 1, 0, 1)	(0, 1, 1, 0)
(0, 1, 1, 0)	(1, 0, 1, 0)
(0, 1, 1, 1)	(1, 1, 1, 0)
(1, 0, 0, 0)	(0, 0, 0, 1)
(1, 0, 0, 1)	(0, 0, 1, 1)
(1, 0, 1, 0)	(0, 1, 0, 1)
(1, 0, 1, 1)	(0, 1, 1, 1)
(1, 1, 0, 0)	(1, 0, 0, 1)
(1, 1, 0, 1)	(1, 0, 1, 1)
(1, 1, 1, 0)	(1, 1, 0, 1)
(1, 1, 1, 1)	(1, 1, 1, 1)

fly diagrams of modified and standard sign Haar transforms, the relationship between 16-point modified and standard sign Haar transforms can be found. The modified and standard sign Haar spectra can be derived from reordering each other for the same input function, for example, $sh(0, 0, 1, 1) = msh(1, 1, 0, 0)$. Table 5.1 shows the corresponding relationship between 16-point modified and standard sign Haar transforms.

5.4.3 Computational Costs

The computational costs which mean the number of additions and subtractions required for the generation of forward and inverse transforms represent the computational complexity of the calculations for the transforms. The modified

Table 5.2: Comparison of Computational Costs for Modified Sign Haar Transform and Sign Walsh Transform

n	modified sign Haar transform	sign Walsh transform
	$4(2^n - 1)$	$n2^{n+1}$
2	12	16
3	28	48
4	60	128
5	124	320
6	252	768
7	508	1792
8	1020	4096
9	2044	9216
10	4092	20480
11	8188	45056
12	16380	98304
13	32764	212992
14	65532	458752
15	131068	983040
16	262140	2097152

sign Haar transform has same computational costs $4(2^n - 1)$ with standard sign Haar transform for N -point ($N = 2^n$) matrix [FR03]. Comparing with known sign Walsh transform, Table 5.2 shows that the computational costs of sign Walsh transform increase considerably when compared with the modified sign Haar transform for higher n .

5.4.4 Application of Modified Sign Haar Transform in Logic Functions

The modified sign Haar spectra for logic functions are shown in this section which corresponds to the properties of standard sign Haar spectra [FR03]. In the following properties, let function f and modified sign Haar spectrum msh be defined as in the previous section. There are two types of coding which have

been widely used in logic design.

Property 5.4.1. Let function $f(\mathbf{x}_n)$ be a constant, such that its ternary vector $\mathbf{F}$ has all the coefficients equal and $F_j (0 \leq j < 2^n)$; $F_j \in \{-1, 0, 1\}$. Then,

$$f(\mathbf{x}_n) = 0 \iff msh(\boldsymbol{\omega}_n) = 0, \quad (5.12)$$

$$f(\mathbf{x}_n) = \pm 1 \iff msh(\boldsymbol{\omega}_n) = \pm 1 \times \left(\bigwedge_{j=1}^n \bar{\omega}_j \right) \quad (5.13)$$

where $\omega_k \in \{0, 1\}$ and the logic AND operations in brackets () will yield value 1 or 0.

Example 5.4.1. For a 3-variable function $f_1(\mathbf{x}_3) = -1 = (-, -, -, -, -, -, -, -)$, its modified sign Haar spectrum can be evaluated using (5.13) as follows:

$$msh(\boldsymbol{\omega}_3) = -1 \times \left(\bigwedge_{j=1}^3 \bar{\omega}_j \right) = -\bar{\omega}_3 \bar{\omega}_2 \bar{\omega}_1 = (-, 0, 0, 0, 0, 0, 0, 0).$$

Property 5.4.2. When S-coded n -variable function is functionally dependent on a single Boolean variable in affirmation, i.e.,

$$f(\mathbf{x}_n) = x_j, \quad j \in \{1, 2, \dots, n\}, x_j \in \{+1, -1\}.$$

Then its corresponding modified sign Haar spectrum is:

$$msh(\boldsymbol{\omega}_n) = +1 \times (\omega_j \wedge \left\{ \bigwedge_{k=1}^{j-1} \bar{\omega}_k \right\}) \quad (5.14)$$

where $\omega_k \in \{0, 1\}$ and the expression $\bigwedge_{k=1}^{j-1} \bar{\omega}_k = 1$ if $j = 1$. The restriction of the value of $\bigwedge_{k=1}^{j-1} \bar{\omega}_k$ when $j = 1$ is the same as above also for the following properties.

Example 5.4.2. For a 3-variable function $f_2(\mathbf{x}_3) = x_2 = (+, +, -, -, +, +, -, -)$,

its modified sign Haar spectrum can be evaluated using (5.14) as follows:

$$msh(\omega_3) = +1 \times (\omega_2 \wedge \{\bigwedge_{k=1}^{2-1} \bar{\omega}_k\}) = \omega_2 \bar{\omega}_1 = (0, 0, +, 0, 0, 0, +, 0).$$

Property 5.4.3. When S-coded n -variable function is functionally dependent on a single Boolean variable in negation, i.e.,

$$f(\mathbf{x}_n) = \bar{x}_j, \quad j \in \{1, 2, \dots, n\}, x_j \in \{+1, -1\}.$$

Then its corresponding modified sign Haar spectrum is:

$$msh(\omega_n) = -1 \times (\omega_j \wedge \{\bigwedge_{k=1}^{j-1} \bar{\omega}_k\}). \quad (5.15)$$

Property 5.4.4. For S-coded n -variable Boolean function $f(\mathbf{x}_n)$ whose modified sign Haar spectrum is $msh(\omega_n)$, the spectrum of the negated function is derived simply by inverting all the signs of the original spectra, i.e.,

$$f(\mathbf{x}_n) \iff msh(\omega_n), \quad \text{then } \overline{f(\mathbf{x}_n)} \iff -msh(\omega_n). \quad (5.16)$$

Property 5.4.5. When R-coded n -variable function is functionally dependent on a single Boolean variable in affirmation, i.e.,

$$f(\mathbf{x}_n) = x_j, \quad j \in \{1, 2, \dots, n\}, x_j \in \{0, 1\}.$$

Then its corresponding modified sign Haar spectrum is:

$$msh(\omega_n) = \bigwedge_{k=1}^n \bar{\omega}_k - (\omega_j \wedge \{\bigwedge_{k=1}^{j-1} \bar{\omega}_k\}). \quad (5.17)$$

Property 5.4.6. When R-coded n -variable function is functionally dependent on a single Boolean variable in negation, i.e.,

$$f(\mathbf{x}_n) = \bar{x}_j, \quad j \in \{1, 2, \dots, n\}, x_j \in \{0, 1\}.$$

Then its corresponding modified sign Haar spectrum is:

$$msh(\omega_n) = \bigwedge_{k=1}^n \bar{\omega}_k + (\omega_j \wedge \{\bigwedge_{k=1}^{j-1} \bar{\omega}_k\}). \quad (5.18)$$

Example 5.4.3. For a 3-variable function $f_3(\mathbf{x}_3) = \bar{x}_3 = (+, +, +, +, 0, 0, 0, 0)$, its modified sign Haar spectrum can be evaluated using (5.18) as follows:

$$\begin{aligned} msh(\omega_3) &= \bigwedge_{k=1}^3 \bar{\omega}_k + (\omega_3 \wedge \{\bigwedge_{k=1}^{3-1} \bar{\omega}_k\}) = \bar{\omega}_3 \bar{\omega}_2 \bar{\omega}_1 + \omega_3 \bar{\omega}_2 \bar{\omega}_1 \\ &= (+, 0, 0, 0, +, 0, 0, 0). \end{aligned}$$

5.5 Sign Hadamard-Haar Transform

Let $\mathbf{SHH} = [shh_0, shh_1, \dots, shh_j, \dots, shh_{N-2}, shh_{N-1}]$ be the ternary vector corresponding to sign Hadamard-Haar spectrum of $\mathbf{F}$. The value of shh_j ($0 \leq j < N$) is given by $\mathbf{SHH}(\omega_n)$ when $\sum_{p=1}^n \omega_p 2^{p-1} = j$.

5.5.1 Definitions of Sign Hadamard-Haar Transform

Definition 5.5.1. An invertible forward sign Hadamard-Haar transform **SHH** is defined as:

$$\begin{aligned} shh(\mathbf{O}_n \oplus_d \omega_1 \oplus_d \omega_n 2^{n-1}) = \text{sign} \sum_{x_n=0}^1 [(-1)^{x_n \omega_n} [\text{sign} \sum_{x_{n-1}=0}^1 [\text{sign} \sum_{x_{n-2}=0}^1 [\\ \dots \text{sign} \sum_{x_1=0}^1 [(-1)^{x_1 \omega_1} f(\mathbf{x}_n)] \dots]]], \quad (5.19) \end{aligned}$$

and

$$\begin{aligned} shh(\mathbf{O}_n \oplus_d \boldsymbol{\omega}_i \oplus_d 2^i \oplus_d \omega_n 2^{n-1}) = \text{sign} \sum_{x_n=0}^1 [(-1)^{x_n \omega_n} [\text{sign} \sum_{x_{n-i-1}=0}^1 [\\ \text{sign} \sum_{x_{n-i-2}=0}^1 [\dots \text{sign} \sum_{x_2=0}^1 [\text{sign} \sum_{x_1=0}^1 [(-1)^{x_{n-2}} \\ f\{[(\mathbf{O}_{n-1} \oplus_d \boldsymbol{\omega}_i) \oplus_c (n-i-1)] \oplus_d \mathbf{x}_{n-i-1}\}] \dots]]]] \quad (5.20) \end{aligned}$$

where $1 \leq i < n-1$ and $\boldsymbol{\omega}_i = \{\omega_i, \omega_{i-1}, \dots, \omega_2, \omega_1\}$.

The inverse sign Hadamard- γ -Haar- χ transform is:

$$\begin{aligned} f(\mathbf{x}_n) = & \text{sign}\{(-1)^{x_1} \{\text{sign}\{shh\{[(\mathbf{O}_1 \triangle \mathbf{x}_{n-1}) \oplus_c 1] \oplus_d 2^{n-2}\} \\ & + (-1)^{x_n} shh\{[(\mathbf{O}_1 \triangle \mathbf{x}_{n-1}) \oplus_c 1] \oplus_d 2^{n-2} \oplus_d 2^{n-1}\}\}\} \\ & + \text{sign}\{(-1)^{x_2} \{\text{sign}\{shh\{[(\mathbf{O}_2 \triangle \mathbf{x}_{n-1}) \oplus_c 2] \oplus_d 2^{n-3}\} \\ & + (-1)^{x_n} shh\{[(\mathbf{O}_2 \triangle \mathbf{x}_{n-1}) \oplus_c 2] \oplus_d 2^{n-3} \oplus_d 2^{n-1}\}\}\} \end{aligned}$$

$$\begin{aligned}
& + \cdots + \text{sign}\{(-1)^{x_i}\{\text{sign}\{shh\{[(\mathbf{O}_i \triangle \mathbf{x}_{n-1}) \oplus_c i] \oplus_d 2^{n-i-1}\} \\
& + (-1)^{x_n} shh\{[(\mathbf{O}_i \triangle \mathbf{x}_{n-1}) \oplus_c i] \oplus_d 2^{n-i-1} \oplus_d 2^{n-1}\}\}\} + \cdots \\
& + \text{sign}\{(-1)^{x_{n-2}}\{\text{sign}\{shh\{[(\mathbf{O}_{n-2} \triangle \mathbf{x}_{n-1}) \oplus_c (n-2)] \oplus_d 2\} \\
& + (-1)^{x_n} shh\{[(\mathbf{O}_{n-2} \triangle \mathbf{x}_{n-1}) \oplus_c (n-2)] \oplus_d 2 \oplus_d 2^{n-1}\}\}\} \\
& + \text{sign}\left\{\sum_{\omega_1=0}^1 (-1)^{x_{n-1}\omega_1}\{\text{sign}[shh(\mathbf{O}_{n-1} \oplus_d \omega_1) \right. \\
& \left. + (-1)^{x_n} shh(\mathbf{O}_{n-1} \oplus_d \omega_1 \oplus_d 2^{n-1})]\}\}\cdots\}\right\} \quad (5.21)
\end{aligned}$$

where $1 \leq i < n-1$ and $\mathbf{x}_i = \{x_i, x_{i-1}, \dots, x_2, x_1\}$.

In (5.21),

$$\begin{aligned}
\mathbf{O}_i \wedge \mathbf{x}_{n-1} &= \mathbf{O}_i \wedge \{x_{n-1}, x_{n-2}, \dots, x_{i+1}, x_i, x_{i-1}, \dots, x_2, x_1\} \\
&= \{x_{n-1}, x_{n-2}, \dots, x_{i+1}, 0, 0, \dots, 0, 0\}.
\end{aligned}$$

Let us prove uniqueness of sign Hadamard-Haar transform for a single variable and the general case for n variables follows by induction.

Proof. The output function $f(x_1) \in \{-1, 0, +1\}$ where $x_1 \in \{0, 1\}$. From (5.19), the forward sign Hadamard-Haar transform is calculated as follows:

$$\begin{aligned}
shh(\boldsymbol{\omega}_1) &= shh(\omega_1) \\
&= \text{sign}\left[\sum_{x_1=0}^1 (-1)^{\omega_1 x_1} f(\mathbf{x}_1)\right] \\
&= \text{sign}\left[\sum_{x_1=0}^1 (-1)^{\omega_1 x_1} f(x_1)\right] \\
&= \text{sign}[f(0) + (-1)^{\omega_1} f(1)]. \quad (5.22)
\end{aligned}$$

From (5.21), the inverse sign Hadamard-Haar transform is calculated as follows:

$$\begin{aligned}
 f(\mathbf{x}_1) &= f(x_1) \\
 &= \text{sign} \left[\sum_{\omega_1=0}^1 (-1)^{x_1 \omega_1} shh(\mathbf{O}_1 \oplus_d \omega_1) \right] \\
 &= \text{sign} \left[\sum_{\omega_1=0}^1 (-1)^{x_1 \omega_1} shh(\omega_1) \right] \\
 &= \text{sign}[shh(0) + (-1)^{x_1} shh(1)]. \tag{5.23}
 \end{aligned}$$

For the case of $x_1 = 0$, the right-hand side of (5.23) yields:

$$\begin{aligned}
 \text{sign}[shh(0) + (-1)^{x_1} shh(1)] &= \text{sign}[\text{sign}[f(0) + f(1)] + \text{sign}[f(0) - f(1)]] \\
 &= f(0).
 \end{aligned}$$

The case of $x_1 = 1$ can be proved similarly. $\square$

Example 5.5.1. For $n = 3$, the definitions of forward sign Hadamard-Haar transform shh become:

$$\begin{aligned}
 shh(\mathbf{O}_3 \oplus_d \omega_1 \oplus_d \omega_3 2^2) &= shh(\omega_3, 0, \omega_1) \\
 &= \text{sign} \sum_{x_3=0}^1 [(-1)^{x_3 \omega_3} [\text{sign} \sum_{x_2=0}^1 [\text{sign} \sum_{x_1=0}^1 \{(-1)^{x_2 \omega_1} f(\mathbf{x}_3)\}]]]
 \end{aligned}$$

and for $1 \leq i < n - 1$,

$$\begin{aligned}
 shh(\mathbf{O}_3 \oplus_d \omega_1 \oplus_d 2^i \oplus_d \omega_3 2^2) &= shh(\omega_3, 1, \omega_1) \\
 &= \text{sign} \sum_{x_3=0}^1 [(-1)^{x_3 \omega_3} [\text{sign} \sum_{x_1=0}^1 (-1)^{x_1} f\{(\mathbf{O}_2 \oplus_d \omega_1) \oplus_c 1\} \oplus_d \mathbf{x}_1]] \\
 &= \text{sign} \sum_{x_3=0}^1 [(-1)^{x_3 \omega_3} [\text{sign} \sum_{x_1=0}^1 (-1)^{x_1} f(x_3, \omega_1, x_1)]]]. \tag{5.24}
 \end{aligned}$$

In a similar manner definitions for inverse transform from (5.20) when $n = 3$ can be derived.

Example 5.5.2. Consider the logic function given by $f_0(\mathbf{x}_3) = (1, 0, 1, 1, 0, 1, 1, 0)$. The elements of its corresponding Sign Hadamard-Haar spectrum $\mathbf{SHH}_0(\omega_3)$ could be calculated from $f_0(\mathbf{x}_3)$ as follows:

$$\begin{aligned}
 shh_0(0, 0, 0) &= \text{sign} \sum_{x_3=0}^1 [(-1)^{x_3\omega_3} [\text{sign} \sum_{x_2=0}^1 [\text{sign} \sum_{x_1=0}^1 \{(-1)^{x_2\omega_1} f_0(\mathbf{x}_3)\}]]] \\
 &= \text{sign} \sum_{x_3=0}^1 [\text{sign} \sum_{x_2=0}^1 [\text{sign} \sum_{x_1=0}^1 f_0(\mathbf{x}_3)]] \\
 &= \text{sign}[\text{sign}[\text{sign}[f_0(0, 0, 0) + f_0(0, 0, 1)] + \text{sign}[f_0(0, 1, 0) + f_0(0, 1, 1)]] \\
 &\quad + \text{sign}[\text{sign}[f_0(1, 0, 0) + f_0(1, 0, 1)] + \text{sign}[f_0(1, 1, 0) + f_0(1, 1, 1)]]] \\
 &= \text{sign}[\text{sign}[\text{sign}[1+0] + \text{sign}[1+1]] + \text{sign}[\text{sign}[0+1] + \text{sign}[1+0]]] \\
 &= 1, \\
 shh_0(1, 1, 1) &= \text{sign} \sum_{x_3=0}^1 [(-1)^{x_3\omega_3} [\text{sign} \sum_{x_1=0}^1 (-1)^{x_1} f_0(x_3, \omega_1, x_1)]] \\
 &= \text{sign} \sum_{x_3=0}^1 [(-1)^{x_3} [\text{sign} \sum_{x_1=0}^1 (-1)^{x_1} f_0(x_3, 1, x_1)]] \\
 &= \text{sign}[\text{sign}[f_0(0, 1, 0) - f_0(0, 1, 1)] - \text{sign}[f_0(1, 1, 0) - f_0(1, 1, 1)]] \\
 &= \text{sign}[\text{sign}[1-1] - \text{sign}[1-0]] \\
 &= -1.
 \end{aligned}$$

In the similar manner the other elements of $\mathbf{SHH}_0(\omega_3)$ can be calculated and the sign Hadamard-Haar spectrum is $\mathbf{SHH}_0(\omega_3) = (1, 0, 0, 1, 0, 0, 1, -1)$.

The elements of $f_0(\mathbf{x}_3)$ can be also calculated from the spectrum $\mathbf{SHH}_0(\omega_3)$. Using (5.19) and (5.20), when $n = 3$, the first and the last minterms in the truth

vector are obtained as follows:

$$\begin{aligned}
f_0(0, 0, 0) &= \text{sign}\{(-1)^{x_1} \{\text{sign}\{shh_0\{[(\mathbf{O}_1 \triangle \mathbf{x}_2) \oplus_c 1] \oplus_d 2\} \\
&\quad + (-1)^{x_3} shh_0\{[(\mathbf{O}_1 \triangle \mathbf{x}_2) \oplus_c 1] \oplus_d 2 \oplus_d 2^2\}\}\} \\
&\quad + \text{sign}\left\{\sum_{\omega_1=0}^1 (-1)^{x_2 \omega_1} \{\text{sign}[shh_0(\mathbf{O}_2 \oplus_d \omega_1) \right. \\
&\quad \left. + (-1)^{x_3} shh_0(\mathbf{O}_2 \oplus_d \omega_1 \oplus_d 2^2)]\}\right\} \\
&= \text{sign}\{\text{sign}[shh_0(0, 1, 0) + shh_0(1, 1, 0)] + \text{sign}\{\text{sign}[shh_0(0, 0, 0) \\
&\quad + shh_0(1, 0, 0)] + \text{sign}[shh_0(0, 0, 1) + shh_0(1, 0, 1)]\}\} \\
&= \text{sign}\{\text{sign}[0 + 0] + \text{sign}\{\text{sign}[1 + 0] + \text{sign}[0 + 1]\}\} \\
&= 1, \\
f_0(1, 1, 1) &= \text{sign}\{(-1)^{x_1} \{\text{sign}\{shh_0\{[(\mathbf{O}_1 \wedge \mathbf{x}_2) \oplus_c 1] \oplus_d 2\} \\
&\quad + (-1)^{x_3} shh_0\{[(\mathbf{O}_1 \wedge \mathbf{x}_2) \oplus_c 1] \oplus_d 2 \oplus_d 2^2\}\}\} \\
&\quad + \text{sign}\left\{\sum_{\omega_1=0}^1 (-1)^{x_2 \omega_1} \{\text{sign}[shh_0(\mathbf{O}_2 \oplus_d \omega_1) \right. \\
&\quad \left. + (-1)^{x_3} shh_0(\mathbf{O}_2 \oplus_d \omega_1 \oplus_d 2^2)]\}\right\} \\
&= \text{sign}\{-\text{sign}[shh_0(0, 1, 1) - shh_0(1, 1, 1)] + \text{sign}\{\text{sign}[shh_0(0, 0, 0) \\
&\quad - shh_0(1, 0, 0)] - \text{sign}[shh_0(0, 0, 1) - shh_0(1, 0, 1)]\}\} \\
&= \text{sign}\{-\text{sign}[1 - (-1)] + \text{sign}\{\text{sign}[1 - 0] + \text{sign}[0 - 1]\}\} \\
&= -1.
\end{aligned}$$

The other elements of $f_0(\mathbf{x}_3)$ can be calculated in a similar way.

5.5.2 Properties of Sign Hadamard-Haar Spectra of Logic Functions and Variables

There is no direct relationship between Sign Hadamard-Haar spectra calculated for S and R coded logic functions except some special cases for selected logic functions described in following properties, which differs from other transforms used in logic design (i.e., Walsh, Haar, arithmetic). Therefore, basic properties for logic operators have to be derived separately for both codings.

In the following presentation of the properties, let logic function f and its corresponding sign Hadamard-Haar transform **SHH** be defined as in the previous section.

Property 5.5.1. Let function $f(\mathbf{x}_n)$ be a constant, such that its ternary vector $\mathbf{F}$ has all the coefficients equal and F_j ($1 \leq j \leq 2^n$); $F_j \in \{-, 0, +\}$. Then, we have:

$$f(\mathbf{x}_n) = 0 \Leftrightarrow shh(\omega_n) = 0, \quad (5.25)$$

$$f(\mathbf{x}_n) = \pm 1 \Leftrightarrow shh(\omega_n) = \pm \prod_{j=1}^n (1 - \omega_j) \quad (5.26)$$

where $x_i, \omega_j \in \{0, 1\}$ and $1 \leq i \leq n$.

Example 5.5.3. For $n = 3$, $f_1(\mathbf{x}_3) = (+, +, +, +, +, +, +, +) \Leftrightarrow shh_1(\omega_3) = (+, 0, 0, 0, 0, 0, 0, 0)$. The corresponding Karnaugh map of the function $f_1(\mathbf{x}_3)$ and sign domain map of the spectrum **SHH**₁(ω_3) are shown in Fig. 5.8(a).

Property 5.5.2. When S -coded n -variable function is dependent on a single logic variable in affirmation, $f(\mathbf{x}_n) = x_j, j \in \{1, \dots, n\}, x_j \in \{+1, -1\}$, then its

Figure 5.8: Karnaugh and sign domain maps for S-coded functions f_1 , f_2 and f_3 .

sign Hadamard-Haar spectrum is:

$$shh(\omega_n) = \begin{cases} +1 \times (\omega_{n-j} \wedge \{\bigwedge_{k=n-j+1}^n \bar{\omega}_k\}) & \text{when } j < n, \\ +1 \times (\omega_n \wedge \{\bigwedge_{k=1}^{n-1} \bar{\omega}_k\}) & \text{when } j = n, \end{cases} \quad (5.27)$$

where $\omega_k \in \{0, 1\}$ and the logic AND operations in bracket () will yield value 1 or 0. The symbol $\bar{\omega}_k$ represents the logical inversion of the transform variable ω_k .

Example 5.5.4. For $n = 3$, when $f_2(\mathbf{x}_3) = x_3$, its sign Hadamard-Haar spec-

trum is:

$$shh_2(\omega_3) = +1 \times (\omega_3 \wedge \{\bigwedge_{k=1}^2 \bar{\omega}_k\}) = +1(\omega_3 \wedge \bar{\omega}_1 \wedge \bar{\omega}_2) = \omega_3 \bar{\omega}_2 \bar{\omega}_1$$

Hence, $f_2(\mathbf{x}_3) = (+, +, +, +, -, -, -, -) \Leftrightarrow shh_2(\omega_3) = (0, 0, 0, 0, +, 0, 0, 0)$.

The function $f_2(\mathbf{x}_3)$ and its corresponding sign Hadamard-Haar spectrum are shown in Fig 5.8(b).

Example 5.5.5. For $n = 3$, when $f_2(\mathbf{x}_3) = x_2$, its sign Hadamard-Haar spectrum is:

$$shh_3(\omega_3) = +1 \times (\omega_{3-2} \wedge \{\bigwedge_{k=3-2+1}^3 \bar{\omega}_k\}) = +1(\omega_1 \wedge \bar{\omega}_2 \wedge \bar{\omega}_3) = \bar{\omega}_3 \bar{\omega}_2 \omega_1$$

Hence, $f_3(\mathbf{x}_3) = (+, +, -, -, +, +, -, -) \Leftrightarrow shh_3(\omega_3) = (0, +, 0, 0, 0, 0, 0, 0)$.

The function $f_3(\mathbf{x}_3)$ and its corresponding sign Hadamard-Haar spectrum are shown in Fig 5.8(c).

Property 5.5.3. When S -coded n -variable function is dependent on a single logic variable in negation, $f(\mathbf{x}_n) = \bar{x}_j, j \in \{1, \dots, n\}, x_j \in \{+1, -1\}$, then its sign Hadamard-Haar spectrum is:

$$shh(\omega_n) = \begin{cases} -1 \times (\omega_{n-j} \wedge \{\bigwedge_{k=n-j+1}^n \bar{\omega}_k\}) & \text{when } j < n, \\ -1 \times (\omega_n \wedge \{\bigwedge_{k=1}^{n-1} \bar{\omega}_k\}) & \text{when } j = n. \end{cases} \quad (5.28)$$

Property 5.5.4. When R -coded n -variable function is dependent on a single logic variable in affirmation, $f(\mathbf{x}_n) = x_j, j \in \{1, \dots, n\}, x_j \in \{+1, -1\}$, then its

sign Hadamard-Haar spectrum is:

$$shh(\omega_n) = \begin{cases} \bigwedge_{k=1}^n \bar{\omega}_k - (\omega_{n-j} \wedge \{\bigwedge_{k=n-j+1}^n \bar{\omega}_k\}) & \text{when } j < n, \\ \bigwedge_{k=1}^n \bar{\omega}_k - (\omega_n \wedge \{\bigwedge_{k=1}^{n-1} \bar{\omega}_k\}) & \text{when } j = n. \end{cases} \quad (5.29)$$

Example 5.5.6. For $n = 4$, when $f_2(\mathbf{x}_4) = x_4$, its sign Hadamard-Haar spectrum can be calculated by using (5.29) as follows:

$$\begin{aligned} shh_4(\omega_4) &= \bigwedge_{k=1}^4 \bar{\omega}_k - (\omega_{4-2} \wedge \{\bigwedge_{k=4-2+1}^4 \bar{\omega}_k\}) \\ &= (\bar{\omega}_1 \wedge \bar{\omega}_2 \wedge \bar{\omega}_3 \wedge \bar{\omega}_4) - (\omega_2 \wedge \bar{\omega}_3 \wedge \bar{\omega}_4) \\ &= \bar{\omega}_4 \bar{\omega}_3 \bar{\omega}_2 \bar{\omega}_1 - \bar{\omega}_4 \bar{\omega}_3 \omega_2. \end{aligned}$$

Hence,

$$\begin{aligned} f_4(\mathbf{x}_4) &= (0, 0, +, +, 0, 0, +, +, 0, 0, +, +, 0, 0, +, +) \Leftrightarrow \\ shh_4(\omega_4) &= (+, 0, -, -, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0). \end{aligned}$$

Property 5.5.5. When R -coded n -variable function is dependent on a single logic variable in negation, $f(\mathbf{x}_n) = \bar{x}_j, j \in \{1, \dots, n\}, x_j \in \{+1, -1\}$, then its sign Hadamard-Haar spectrum is:

$$shh(\omega_n) = \begin{cases} \bigwedge_{k=1}^n \bar{\omega}_k + (\omega_{n-j} \wedge \{\bigwedge_{k=n-j+1}^n \bar{\omega}_k\}) & \text{when } j < n, \\ \bigwedge_{k=1}^n \bar{\omega}_k + (\omega_n \wedge \{\bigwedge_{k=1}^{n-1} \bar{\omega}_k\}) & \text{when } j = n. \end{cases} \quad (5.30)$$

5.5.3 Calculation of Sign Hadamard-Haar Transform by Matrix Approach

Sign Hadamard-Haar transform may be evaluated using a matrix approach. It is also possible to evaluate a single spectral coefficient of sign Hadamard-Haar spectrum without need to calculate the whole spectrum.

Definition 5.5.2. Let $\mathbf{T}_n$ and $\mathbf{T}_n^{-1}$ be $N \times N$ forward and inverse sign Hadamard-Haar transform matrices. Then, we have:

$$\mathbf{T}_n = [\mathbf{t}_0, \mathbf{t}_1, \dots, \mathbf{t}_i, \dots, \mathbf{t}_{N-1}]^T, \quad \mathbf{T}_n^{-1} = [\mathbf{t}_0^{-1}, \mathbf{t}_1^{-1}, \dots, \mathbf{t}_i^{-1}, \dots, \mathbf{t}_{N-1}^{-1}]^T, \quad (5.31)$$

where $0 \leq i \leq N - 1$.

Definition 5.5.3. Let $\mathbf{A} = [\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_i, \dots, \mathbf{a}_{N-1}]$ be a $1 \times N$ row vector and let $\mathbf{B} = [\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_i, \dots, \mathbf{b}_{N-1}]^T$ be a $N \times 1$ row vector with the elements derived from component-wise multiplication:

$$\mathbf{A} \cdot \mathbf{B} = [\mathbf{a}_0 \mathbf{b}_0, \mathbf{a}_1 \mathbf{b}_1, \dots, \mathbf{a}_i \mathbf{b}_i, \dots, \mathbf{a}_{N-1} \mathbf{b}_{N-1}]. \quad (5.32)$$

Definition 5.5.4. Let $\mathbf{A}$ be a $1 \times N$ row vector where entries are $[\mathbf{a}_i]$. The sign modulo function $[\mathbf{A}]$ is a scalar where the elements in $\mathbf{A}$ are summed in pairs in the form of a binary tree summation, and the tree is evaluated from the bottom up.

Property 5.5.6. Let $\mathbf{F} = [F_0, F_1, \dots, F_j, \dots, F_{N-2}, F_{N-1}]$, $\mathbf{SHH} = [shh_0, shh_1, \dots, shh_j, \dots, shh_{N-2}, shh_{N-1}]$ define the ternary vector and its sign Hadamard-Haar spectra coefficients. Then, $shh_i = [\mathbf{t}_i \cdot \mathbf{F}]$ and $F_i = [\mathbf{t}_i^{-1} \cdot \mathbf{SHH}]$.

Example 5.5.7. Let $n = 3$, and the ternary vector $\vec{F} = [-, -, +, -, +, -, -, +]$. From Definition 5.5.4, we can have:

$$\begin{aligned} \mathbf{T}_n = & [(+++++), (+- - - - + + - - -), \\ & (+ - - 00 + - - 00), (00 + - - 00 + - -), \\ & (++++ - - - - - - -), (+- - - - - - - + +), \\ & (+ - - 00 - - + 00), (00 + - - 00 - - +)]^T. \end{aligned}$$

By using Property 5.5.6 and Definition 5.5.4, we have:

$$\begin{aligned} shh_0 &= [(+++++) \cdot (- - + - + - -)^T] \\ &= [(- - + - + - -)] = [(-000)] = [(-0)] \\ &= -1, \\ shh_1 &= [(++ - - + + - -) \cdot (- - + - + - -)^T] \\ &= [(- - - + + - + -)] = [(-000)] = [(-0)] \\ &= -1, \\ shh_2 &= [(+ - 00 + - 00) \cdot (- - + - + - -)^T] \\ &= [(- + 00 + + 00)] = [(00 + 0)] = [(0+)] \\ &= 1, \\ shh_3 &= [(00 + - 00 + -) \cdot (- - + - + - -)^T] \\ &= [(00 + + 00 - -)] = [(0 + 0 -)] = [(+-)] \\ &= 0, \\ shh_4 &= [(++++ - - - -) \cdot (- - + - + - -)^T] \\ &= [(- - + - - + + -)] = [(-000)] = [(-0)] \\ &= -1. \end{aligned}$$

The other elements of **SHH** can be calculated in the similar manner and the

Figure 5.9: Butterfly diagrams for the forward and inverse sign Hadamard-Haar transforms where $n = 3$.

sign Hadamard-Haar spectrum is $\mathbf{SHH} = [-, -, +, 0, -, -, -, +]$.

5.5.4 Computational Costs

Fast flow diagrams for calculation of forward and inverse sign Hadamard-Haar transforms are shown for $N = 8$ in Fig. 5.9(a) and Fig. 5.9(b) accordingly. The total number of operations (additions and subtractions) required to perform forward fast sign Hadamard-Haar transform for a single element of set $\{+1, 0, -1\}^N$ and inverse sign Hadamard-Haar transform for a single element of set $\{+1, 0, -1\}_{shh}^N$ and for transform matrix of order $N = 2^n$ is equal to $3 \times 2^{n+1} - 8$.

5.6 Sign Hadamard- γ -Haar- χ Transform

Based on the rationalized Hadamard-Haar transform, the sign Hadamard- γ -Haar- χ transform is further developed. The sign Walsh, Haar and Hadamard-Haar transforms can be seen as the special cases of sign Hadamard- γ -Haar- χ transform.

Let $\mathbf{SHW} = [shw_0, shw_1, \dots, shw_j, \dots, shw_{N-2}, shw_{N-1}]$ be the ternary vector corresponding to sign Hadamard- γ -Haar- χ spectrum of $\mathbf{F}$. The value of shw_j ($0 \leq j < N$) is given by $\mathbf{SHW}(\omega_n)$ when $\sum_{p=1}^n \omega_p 2^{p-1} = j$. In the new recursive sign transform, γ and χ show the number of butterflies in Hadamard and Haar parts of sign Hadamard- γ -Haar- χ transform. $1 \leq p \leq n$, γ and χ are integers where $0 \leq \gamma < n$, $0 < \chi \leq n$ and $\gamma + \chi = n$.

5.6.1 Definitions of Sign Hadamard- γ -Haar- χ Transform

Definition 5.6.1. An invertible sign Hadamard- γ -Haar- χ transform shw and its inverse transform shw^{-1} are the mappings $hh : \{+1, 0, -1\}^N \rightarrow \{+1, 0, -1\}_{hh}^N$ and $hh^{-1} : \{+1, 0, -1\}_{hh}^N \rightarrow \{+1, 0, -1\}^N$, where $N=2^n$. In the above equations, $\{+1, 0, -1\}_{hh}^N$ represents a set with the elements from $\{+1, 0, -1\}^N$ permuted by the mapping shw of all the elements of the set $\{+1, 0, -1\}^N$. In order to obtain the sign Hadamard- γ -Haar- χ spectrum shw and its inverse, the results of each fast forward or inverse Hadamard- γ -Haar- χ butterfly block are quantized first. In the above equations, The cardinality of the original data set $\{+1, 0, -1\}^N$ and its transformed spectrum is equal to 3^N .

Definition 5.6.2. An invertible forward sign Hadamard- γ -Haar- χ transform

shw is defined as:

$$\begin{aligned}
& shw(\mathbf{O}_n \oplus_d \omega_1 \oplus_d \omega_{\chi+1} 2^\chi \oplus_d \omega_{\chi+2} 2^{\chi+1} \oplus_d \dots \oplus_d \omega_{n-1} 2^{n-2} \oplus_d \omega_n 2^{n-1}) \\
&= \text{sign} \sum_{x_n=0}^1 [(-1)^{x_n \omega_n} [\text{sign} \sum_{x_{n-1}=0}^1 [(-1)^{x_{n-1} \omega_{n-1}} [\dots \text{sign} \sum_{x_{\chi+2}=0}^1 [(-1)^{x_{\chi+2} \omega_{\chi+2}} [\\
&\quad \text{sign} \sum_{x_{\chi+1}=0}^1 [(-1)^{x_{\chi+1} \omega_{\chi+1}} [\text{sign} \sum_{x_\chi=0}^1 [\text{sign} \sum_{x_{\chi-1}=0}^1 [\dots \text{sign} \sum_{x_2=0}^1 [\\
&\quad \text{sign} \sum_{x_1=0}^1 [(-1)^{x_1 \omega_1} f(\mathbf{x}_n)]] \dots]]]]] \dots]]]] \quad (5.33)
\end{aligned}$$

and

$$\begin{aligned}
& shw(\mathbf{O}_n \oplus_d \omega_i \oplus_d 2^i \oplus_d \omega_{\chi+1} 2^\chi \oplus_d \omega_{\chi+2} 2^{\chi+1} \oplus_d \dots \oplus_d \omega_{n-1} 2^{n-2} \oplus_d \omega_n 2^{n-1}) \\
&= \text{sign} \sum_{x_n=0}^1 [(-1)^{x_n \omega_n} [\text{sign} \sum_{x_{n-1}=0}^1 [(-1)^{x_{n-1} \omega_{n-1}} [\dots \text{sign} \sum_{x_{\chi+1}=0}^1 [(-1)^{x_{\chi+1} \omega_{\chi+1}} [\\
&\quad \text{sign} \sum_{x_{\chi-i}=0}^1 [\text{sign} \sum_{x_{\chi-i-1}=0}^1 [\dots \text{sign} \sum_{x_2=0}^1 [\text{sign} \sum_{x_1=0}^1 [(-1)^{x_{\chi-1}} \\
&\quad f\{[(\mathbf{O}_\chi \oplus_d \omega_i) \oplus_c (\chi-i)] \oplus_d \mathbf{x}_{\chi-i}]\}]] \dots]]] \dots]]]] \quad (5.34)
\end{aligned}$$

where $1 \leq i < \chi$.

The inverse sign Hadamard- γ -Haar- χ transform is:

$$\begin{aligned}
f(\mathbf{x}_n) = & \text{sign} \left\{ (-1)^{x_1} [\text{sign} \sum_{\omega_{\chi+1}=0}^1 (-1)^{x_{\chi+1}\omega_{\chi+1}} [\text{sign} \sum_{\omega_{\chi+2}=0}^1 (-1)^{x_{\chi+2}\omega_{\chi+2}} [\dots \right. \\
& \text{sign} \sum_{\omega_{n-1}=0}^1 (-1)^{x_{n-1}\omega_{n-1}} [\text{sign} \sum_{\omega_n=0}^1 (-1)^{x_n\omega_n} [shw\{[(\mathbf{O}_1 \wedge \mathbf{x}_{\chi}) \oplus_c 1] \oplus_d \\
& 2^{\chi-1} \oplus_d \omega_{\chi+1} 2^{\chi} \oplus_d \omega_{\chi+2} 2^{\chi+1} \oplus_d \dots \oplus_d \omega_{n-1} 2^{n-2} \oplus_d \omega_n 2^{n-1} \}]] \dots] \\
& + \text{sign} \left\{ (-1)^{x_2} [\text{sign} \sum_{\omega_{\chi+1}=0}^1 (-1)^{x_{\chi+1}\omega_{\chi+1}} [\dots \text{sign} \sum_{\omega_n=0}^1 (-1)^{x_n\omega_n} [\right. \\
& shw\{[(\mathbf{O}_2 \wedge \mathbf{x}_{\chi}) \oplus_c 2] \oplus_d 2^{\chi-2} \oplus_d \omega_{\chi+1} 2^{\chi} \oplus_d \omega_{\chi+2} 2^{\chi+1} \oplus_d \dots \dots \dots
\end{aligned}$$

$$\begin{aligned}
& \oplus_d \omega_{n-1} 2^{n-2} \oplus_d \omega_n 2^{n-1} \} \dots] + \dots + \text{sign} \left\{ (-1)^{x_{\chi-1}} [\right. \\
& \text{sign} \sum_{\omega_{\chi+1}=0}^1 (-1)^{x_{\chi+1}\omega_{\chi+1}} [\dots \text{sign} \sum_{\omega_n=0}^1 (-1)^{x_n\omega_n} [shw\{[(\mathbf{O}_{\chi-1} \wedge \mathbf{x}_{\chi}) \oplus_c \\
& (\chi-1)] \oplus_d 2 \oplus_d \omega_{\chi+1} 2^{\chi} \oplus_d \omega_{\chi+2} 2^{\chi+1} \oplus_d \dots \oplus_d \omega_{n-1} 2^{n-2} \oplus_d \\
& \omega_n 2^{n-1} \}] + \text{sign} \left\{ (-1)^{x_{\chi}} [\text{sign} \sum_{\omega_{\chi+1}=0}^1 (-1)^{x_{\chi+1}\omega_{\chi+1}} [\dots \right. \\
& \text{sign} \sum_{\omega_n=0}^1 (-1)^{x_n\omega_n} [shw(\mathbf{O}_{\chi} \oplus_d \omega_1 \oplus_d \omega_{\chi+1} 2^{\chi} \oplus_d \omega_{\chi+2} 2^{\chi+1} \oplus_d \dots \\
& \left. \oplus_d \omega_{n-1} 2^{n-2} \oplus_d \omega_n 2^{n-1})] \dots] \right\} \dots] \left. \right\} \dots \left. \right\} \left. \right\}. \tag{5.35}
\end{aligned}$$

Property 5.6.1. It can be noticed that the sign Walsh transform, sign Haar transform, and sign Hadamard-Haar transform can be seen as special cases of sign Hadamard- γ -Haar- χ transform. When $\gamma = n - 1$ and $\chi = 1$, the sign Hadamard- γ -Haar- χ transform is the standard sign Walsh transform. When $\gamma = 0$ and $\chi = n$, the sign Hadamard- γ -Haar- χ transform is the standard sign Haar transform. When $\gamma = 1$ and $\chi = n - 1$, the sign Hadamard- γ -Haar- χ transform is the standard sign Hadamard-Haar transform.

5.6.2 Computational Costs

The recursive definitions of forward and inverse sign Hadamard- γ -Haar- χ transforms have been developed. The fast butterfly diagrams can be used for the calculation of the sign Hadamard- γ -Haar- χ transform. The fast butterfly diagrams for forward and inverse sign Hadamard- γ -Haar- χ transforms are shown in Fig. 5.10(a) and Fig. 5.10(b), respectively, as an example, where $\gamma = 2$ and $\chi = 2$.

Figure 5.10: Fast butterfly diagrams for the forward and inverse sign Hadamard-2-Haar-2 transforms.

The computational costs which mean the number of additions and subtractions required for the generation of forward and inverse transforms represent the computational complexity of the calculations for the transforms. According to the fast butterfly structures of the sign Hadamard- γ -Haar- χ transform, the computational costs of the sign Hadamard- γ -Haar- χ transform are $(\gamma + 2)2^{n+1} - 2^{\gamma+2}$. The computational costs of the sign Walsh transform are known as $n2^{n+1}$. Table 5.3 shows some results about computational costs of sign Hadamard- γ -Haar- χ transform and sign Walsh transform. From Table 5.3,

Table 5.3: Comparison of Computational Costs for Sign Hadamard- γ -Haar- χ Transform and Sign Walsh Transform

n	sign Hadamard- γ -Haar- χ transform		sign Walsh trans- form
	$(\gamma + 2) 2^{n+1} - 2^{\gamma+2}$		$n2^{n+1}$
1	4 ($\gamma=0$)		4
2	12 ($\gamma=0$)	16 ($\gamma=1$)	16
3	28 ($\gamma=0$)	40 ($\gamma=1$)	48
4	60 ($\gamma=0$)	112 ($\gamma=2$)	128
5	184 ($\gamma=1$)	288 ($\gamma=3$)	320
6	1880 ($\gamma=1$)	704 ($\gamma=4$)	768
7	1472 ($\gamma=4$)	1792 ($\gamma=6$)	1792
8	2032 ($\gamma=2$)	3456 ($\gamma=5$)	4096
9	5088 ($\gamma=3$)	8704 ($\gamma=7$)	9216
10	14208 ($\gamma=5$)	17920 ($\gamma=7$)	20480
11	16368 ($\gamma=2$)	32512 ($\gamma=6$)	45056
12	24568 ($\gamma=1$)	40928 ($\gamma=3$)	98304
13	98240 ($\gamma=4$)	114560 ($\gamma=5$)	212992
14	229248 ($\gamma=5$)	294400 ($\gamma=7$)	458752
15	262128 ($\gamma=2$)	524032 ($\gamma=6$)	983040

it is obvious that the system based on sign Hadamard- γ -Haar- χ transform is much more efficient than the one based on sign Walsh transform due to much smaller number of computations especially for higher n .

5.7 Summary

The fixed sign Walsh hardware structure and the modified sign Haar transform with in-place structures are developed which can be easily implemented in hardware using look-up table cascades. Sign Hadamard-Haar transform is a new sign transform based on the combination of Hadamard and Haar butterfly diagrams. The sign Hadamard- γ -Haar- χ transform is further developed which can provide better security properties by its design and is suitable for cryptographic systems

since the coefficients γ and χ can be adjusted.

Chapter 6

Hybrid Multi-Polarity

Arithmetic-Walsh Transform

6.1 Introduction

Many important problems in CAD such as synthesis, verification and optimization can be solved very efficiently not in the original Boolean domain but rather through evaluating the corresponding arithmetic expression [AAE95, Bla98, BSS99, BGG98, Cas96, CMZ⁺93, CMZ⁺97, Dam92, Fal99b, FK99, FK00, FSP92, GES91, Hei91, Hur92, KSA00, KSA03, OLRPP93, RF98, RF99, RMB92, RZ01, RZ04, Sas93, Sas99, Sch98, SF96, SF03, SS01, SSJ98, SSS96, Sta98, TDM01, TMS96, TMS97, TN95, YM91]. Both Walsh and arithmetic transforms are widely used while each of these transforms has some advantages and disadvantages based on the specific problem to be solved. Therefore, it is interesting to propose a new arithmetic-Walsh transform which combines the advantages of both Walsh and arithmetic transforms.

6.2 Definitions of Multi-Polarity Arithmetic-Walsh Transform

In this section, a new hybrid multi-polarity transform based on multi-polarity arithmetic [FP90a, SSJ98] and multi-polarity Walsh transforms [Fal94] will be proposed. This new hybrid multi-polarity transform is named hybrid arithmetic-Walsh transform.

Let $\omega_{\mathbf{n}} = \{\omega_1, \omega_2, \dots, \omega_i, \dots, \omega_{n-1}, \omega_n\}$ be n -tuples over $\text{GF}(2)$. This vector represents the polarity of new multi-polarity arithmetic-Walsh transform.

Definition 6.2.1. For an n -variable discrete binary function, the r th-order multi-polarity arithmetic-Walsh transform matrix $\mathbf{MAW}_r^{\omega_{\mathbf{n}}}(n)$ and its inverse $(\mathbf{MAW}_r^{\omega_{\mathbf{n}}})^{-1}(n)$ are defined as:

$$\mathbf{MAW}_r^{\omega_{\mathbf{n}}}(n) = \left(\bigotimes_{i=1}^r \mathbf{MW}^{\omega_i}(1) \right) \otimes \left(\bigotimes_{i=r+1}^n \mathbf{MA}^{\omega_i}(1) \right), \quad (6.1)$$

$$(\mathbf{MAW}_r^{\omega_{\mathbf{n}}})^{-1}(n) = \frac{1}{2^r} \left(\bigotimes_{i=1}^r (\mathbf{MW}^{\omega_i})^{-1}(1) \right) \otimes \left(\bigotimes_{i=r+1}^n (\mathbf{MA}^{\omega_i})^{-1}(1) \right) \quad (6.2)$$

where $0 \leq r \leq n$,

$$\mathbf{MW}^{\omega_i}(1) = \begin{cases} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & \text{if } \omega_i = 0, \\ \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} & \text{if } \omega_i = 1, \end{cases}$$

$$\begin{aligned}
 (\mathbf{MW}^{\omega_i})^{-1}(1) &= \begin{cases} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & \text{if } \omega_i = 0, \\ \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} & \text{if } \omega_i = 1, \end{cases} \\
 \mathbf{MA}^{\omega_i}(1) &= \begin{cases} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} & \text{if } \omega_i = 0, \\ \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} & \text{if } \omega_i = 1, \end{cases} \\
 \text{and } (\mathbf{MA}^{\omega_i})^{-1}(1) &= \begin{cases} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} & \text{if } \omega_i = 0, \\ \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} & \text{if } \omega_i = 1. \end{cases}
 \end{aligned}$$

When the Kronecker product of the matrices is carried out in (6.1) and (6.2) for $j = m$, then the term $'\bigotimes_{i=j}^m'$ disappears from them.

Property 6.2.1. It can be noticed that the special cases of r th-order multi-polarity arithmetic-Walsh transform are the multi-polarity arithmetic transform for $r = 0$ and the multi-polarity Walsh transform for $r = n$.

When $r = 0$ and $\omega_{\mathbf{n}} = \{0, \dots, 0\}$, the multi-polarity arithmetic-Walsh transform is standard arithmetic transform. When $r = n$ and $\omega_{\mathbf{n}} = \{0, \dots, 0\}$, the multi-polarity arithmetic-Walsh transform is standard Walsh transform.

Definition 6.2.2. For an n -variable function $f(x_1, x_2, \dots, x_i, \dots, x_{n-1}, x_n)$, the basic functions for the multi-polarity arithmetic-Walsh expansion can be

generated. For the r th-order multi-polarity arithmetic-Walsh transform whose polarity is $\vec{\omega}_n$, the basic functions for its corresponding expansion are:

$$\mathbf{X}_f = [1 \quad 1 - 2\dot{x}_1] \otimes [1 \quad 1 - 2\dot{x}_2] \otimes \cdots \otimes [1 \quad 1 - 2\dot{x}_r] \\ \otimes [1 \quad \dot{x}_{r+1}] \otimes \cdots \otimes [1 \quad \dot{x}_n]. \quad (6.3)$$

$$\text{where } \dot{x}_i = \begin{cases} x_i & \text{if } \omega_i = 0, \\ \bar{x}_i & \text{if } \omega_i = 1, \end{cases} \quad \text{and } 1 < i \leq n.$$

Property 6.2.2. Let $\mathbf{S}_f$ be the multi-polarity arithmetic-Walsh spectrum of the n -variable function f , the original function can be represented as follow:

$$f = \frac{1}{2^r} \mathbf{X}_f \mathbf{S}_f. \quad (6.4)$$

where $\mathbf{S}_f$ is defined in (6.3) and r is defined in (6.1).

Example 6.2.1. For $r = 2$, $n = 3$ and $\boldsymbol{\omega}_3 = \{\omega_1, \omega_2, \omega_3\} = \{0, 1, 1\}$, the multi-polarity arithmetic-Walsh transform matrix $\mathbf{MAW}_2^{\boldsymbol{\omega}_3}(3)$ and its inverse $(\mathbf{MAW}_2^{\boldsymbol{\omega}_3})^{-1}(3)$ are:

$$\mathbf{MAW}_2^{\boldsymbol{\omega}_3}(n) = \left(\bigotimes_{i=1}^2 \mathbf{MW}^{\omega_i}(1) \right) \otimes \mathbf{MA}^{\omega_3}(1) \\ = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 & -1 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix},$$

$$\begin{aligned} (\mathbf{MAW}_2^{\omega_3})^{-1}(3) &= \frac{1}{2^2} \left(\bigotimes_{i=1}^2 (\mathbf{MW}^{\omega_i})^{-1}(1) \right) \otimes (\mathbf{MA}^{\omega_3})^{-1}(1) \\ &= \frac{1}{4} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \\ &= \frac{1}{4} \begin{bmatrix} 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \end{bmatrix}. \end{aligned}$$

Using (6.3), the basic functions for this multi-polarity arithmetic-Walsh transform can be derived as follow:

$$\begin{aligned}
\mathbf{X}_f &= [1 \quad 1 - 2x_1] \otimes [1 \quad 1 - 2\bar{x}_2] \otimes [1 \quad \bar{x}_3] \\
&= [1 \quad \bar{x}_3 \quad 1 - 2\bar{x}_2 \quad (1 - 2\bar{x}_2)\bar{x}_3 \quad 1 - 2x_1 \quad (1 - 2x_1)\bar{x}_3 \\
&\quad (1 - 2x_1)(1 - 2\bar{x}_2) \quad (1 - 2x_1)(1 - 2\bar{x}_2)\bar{x}_3]. \quad (6.5)
\end{aligned}$$

For a 3-variable function $f_1(x_1, x_2, x_3) = \sum(0, 1, 4, 7)$, the corresponding multi-polarity arithmetic-Walsh spectrum can be calculated as follow:

$$\begin{aligned}
\mathbf{S}_{f_1} &= \mathbf{MAW}_2^{\omega_3}(3) \times \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 & -1 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} 2 \\ 0 \\ 0 \\ -2 \\ 0 \\ 0 \\ -2 \\ 2 \end{bmatrix}. \quad (6.6)
\end{aligned}$$

From (6.5) and (6.6), the corresponding multi-polarity arithmetic-Walsh expan-

sion can be calculated as:

$$\begin{aligned}
 f_1 &= \frac{1}{2^r} \mathbf{X}_f \mathbf{S}_{f_1} \\
 &= \frac{1}{4} [2 - 2(1 - 2\bar{x}_2)\bar{x}_3 - 2(1 - 2x_1)(1 - 2\bar{x}_2) + 2(1 - 2x_1)(1 - 2\bar{x}_2)\bar{x}_3] \\
 &= \frac{1}{2} [1 - (1 - 2\bar{x}_2)\bar{x}_3 - (1 - 2x_1)(1 - 2\bar{x}_2) + (1 - 2x_1)(1 - 2\bar{x}_2)\bar{x}_3]. \quad (6.7)
 \end{aligned}$$

6.3 Fast Algorithms

Both the arithmetic and Walsh transforms have the fast algorithms to calculate the corresponding spectra. Since the multi-polarity arithmetic-Walsh transform has the recursive relations given in the form of layered Kronecker matrices, the fast algorithms can be derived for the calculation of the multi-polarity arithmetic-Walsh spectrum. Similar to the fast Walsh transform and other known fast transforms, the multi-polarity arithmetic-Walsh transform matrix can be calculated by the products of the factored matrices.

Property 6.3.1. The forward multi-polarity arithmetic-Walsh transform matrix $\mathbf{MAW}_r^{\omega_n}(n)$ can be factorized as:

$$\mathbf{MAW}_r^{\omega_n}(n) = \prod_{i=1}^n \mathbf{D}^{\omega_i}(i). \quad (6.8)$$

$$\text{where } \mathbf{D}_r^{\omega_i}(i) = \left(\bigotimes_{j=1}^{i-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{P}_r^{\omega_i}(i) \otimes \left(\bigotimes_{j=1}^{n-i} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right),$$

$$\mathbf{P}_r^{\omega_i}(i) = \begin{cases} \mathbf{MW}^{\omega_i}(1) & \text{if } i \leq r, \\ \mathbf{MA}^{\omega_i}(1) & \text{if } i > r, \end{cases} \quad \text{and } 1 \leq i \leq n.$$

Property 6.3.2. The inverse multi-polarity arithmetic-Walsh transform matrix $(\mathbf{MAW}_r^{\omega_n})^{-1}(n)$ can be factorized as:

$$(\mathbf{MAW}_r^{\omega_n})^{-1}(n) = \prod_{i=1}^n \mathbf{E}_r^{\omega_i}(i). \quad (6.9)$$

$$\text{where } \mathbf{E}_r^{\omega_i}(i) = \left(\bigotimes_{j=1}^{i-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{Q}_r^{\omega_i}(i) \otimes \left(\bigotimes_{j=1}^{n-i} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right),$$

$$\mathbf{Q}_r^{\omega_i}(i) = \begin{cases} (\mathbf{MW}^{\omega_i})^{-1}(1) & \text{if } i \leq r, \\ (\mathbf{MA}^{\omega_i})^{-1}(1) & \text{if } i > r, \end{cases} \quad \text{and } 1 \leq i \leq n.$$

Example 6.3.1. Using Property 6.3.1 and Property 6.3.2, the multi-polarity arithmetic-Walsh transforms whose polarity is 0, 1, 1 in Example 6.2.1 can be factorized to the products of three matrices. The forward transform matrix $\mathbf{MAW}_2^{\omega_3}(3)$ can be factorized as $\mathbf{MAW}_2^{\omega_3}(n) = \prod_{i=1}^3 \mathbf{D}_r^{\omega_i}(i)$.

For $i \leq 2$:

$$\begin{aligned}
 \mathbf{D}_2^0(1) &= \mathbf{M}\mathbf{W}^0(1) \otimes \left(\bigotimes_{j=1}^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\
 &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \left(\bigotimes_{j=1}^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\
 &= \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix},
 \end{aligned}$$

$$\begin{aligned}
 \mathbf{D}_2^1(2) &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes (\mathbf{M}\mathbf{W}^0)^{-1}(1) \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
 \end{aligned}$$

$$= \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}.$$

For $i > 2$:

$$\begin{aligned} \mathbf{D}_2^1(3) &= \left(\bigotimes_{j=1}^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes (\mathbf{MA}^0)^{-1}(1) \\ &= \left(\bigotimes_{j=1}^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}. \end{aligned}$$

Similarly, the inverse transform matrix $(\mathbf{MAW}_2^{\omega_3})^{-1}(3)$ can be factorized to the products of matrices by (6.9) as follow:

$$\begin{aligned}
(\mathbf{MAW}_2^{\omega_3})^{-1}(n) &= \frac{1}{4} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} \\
&\times \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \\
&\times \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.
\end{aligned}$$

Figure 6.1: Fast butterfly diagrams for the forward and inverse multi-polarity arithmetic-Walsh transforms where $r = 2$, $n = 3$, and $\vec{\omega}_3 = \{\omega_1, \omega_2, \omega_3\} = \{0, 1, 1\}$.

The corresponding fast flow diagrams for calculation of forward and inverse multi-polarity arithmetic-Walsh transform matrices for $n = 3$ are shown in Fig. 6.1(a) and Fig. 6.1(b), respectively. In both figures, the solid lines and dotted lines represent addition and subtraction, respectively.

6.4 Analysis of Computational Costs

The fast algorithms to calculate the new multi-polarity transform have been described in Section 6.3. Such fast algorithms greatly reduce the number of arithmetic operations as compared to the computation of multi-polarity arithmetic-Walsh transform by the whole matrix. The computational costs which mean the number of additions and subtractions required for the generation of forward and inverse multi-polarity arithmetic-Walsh transforms are $(n + r)2^n$, while the

Table 6.1: Comparison of Computational Costs of Multi-Polarity Arithmetic-Walsh and Multi-Polarity Walsh Transforms

n	arithmetic-Walsh		Walsh
	$(n + r)2^n$		$n2^{n+1}$
2	12	($r=1$)	16
3	32	($r=1$)	48
4	96	($r=2$)	128
5	224	($r=2$)	320
6	576	($r=3$)	768
7	1280	($r=3$)	1792
8	3072	($r=4$)	4096
9	6656	($r=4$)	9216
10	15360	($r=5$)	20480
11	32768	($r=5$)	45056
12	73728	($r=6$)	98304
13	155648	($r=6$)	212992
14	344064	($r=7$)	458752
15	720896	($r=7$)	983040
16	1572864	($r=8$)	2097152

the computational costs of multi-polarity Walsh and arithmetic transforms are known as $n2^{n+1}$ and $n2^n$, respectively [FP90a, SSJ98, FC97b]. Consequently, the calculation of the multi-polarity arithmetic-Walsh transform requires more arithmetic operations than the ones of multi-polarity arithmetic transform excluding the case when $r = 0$, and less arithmetic operations than the ones of multi-polarity Walsh transform excluding the case when $r = n$. The computational costs of the multi-polarity arithmetic-Walsh transforms are dependent on r for the same n , which means that the number of arithmetic parts or Walsh parts in the hybrid transform makes the difference to the computational complexity. More arithmetic parts in the transform will bring the hybrid transform lower computational costs, and vice versa.

The effects on the computational costs by the arithmetic and Walsh parts in the multi-polarity arithmetic-Walsh transform can be also seen from its but-

terfly structures. The left $n - r$ blocks of the diagram are arithmetic parts which are corresponding to the structure of fast multi-polarity arithmetic transform [FP90a, SSJ98], while the right r blocks of the diagram are Walsh parts which are related to the butterfly diagram of fast multi-polarity Walsh transform [Fal94]. The computational costs of the multi-polarity arithmetic-Walsh transform increase with r for the same n . Table 6.1 shows some results about computational costs of multi-polarity arithmetic-Walsh transform and multi-polarity Walsh transform.

6.5 Arithmetic-Walsh Spectral Transform Decision Diagrams

Decision Diagrams are data structures which are widely used in logic synthesis and optimization algorithms since they provide compact representation of discrete logic functions [Bry86, Bry91, Bry92, SF96, CFM⁺93, CFZ95, CMZ⁺93, CMZ⁺97, FMY97, LPV94, LPV96, SA03, SSJ98]. Since the standard Walsh and arithmetic transforms can be considered as the special cases of the zero-polarity ($\omega_{\mathbf{n}} = \{0, 0, \dots, 0\}$) for arithmetic-Walsh transform, the arithmetic-Walsh STDDs will also include the Walsh and arithmetic STDDs as two special cases.

Definition 6.5.1. For an n -variable function $f(x_1, x_2, \dots, x_i, \dots, x_{n-1}, x_n)$, the arithmetic-Walsh spectral transform decision diagram is a directed acyclic graph with exactly one root. It consists of two types of nodes:

- *Sink* is the terminal node which is located in the bottom of the graph. It has not outgoing edges, while it has integer value. The number of sinks in

Figure 6.2: Arithmetic-Walsh spectral transform decision tree node at level i : (a) general form, (b) $1 \leq i \leq r$, (c) $r < i \leq n$.

one arithmetic-Walsh spectral transform decision diagram is in the range of $[1, 2^n]$.

- *Non-sink node* is the non-terminal node v which has two outgoing edges, which are called low-edge and high-edge, respectively. Both the edges of the non-sink node have its values denoted by $value(edge)$ which are called edge-values.

The successor nodes determined by the low-edge and high-edge are denoted by $Child_l(v)$ and $Child_h(v)$, respectively.

From the relationships between arithmetic-Walsh spectral and logical interpretations of the discrete functions which are shown in Definition 6.2.2 and Property 6.2.2, the spectral interpretation of the multi-polarity arithmetic-Walsh transform through STDDs can be derived.

Definition 6.5.2. The multi-polarity arithmetic-Walsh spectral transform decision tree (STDT) in which each node has at most one incoming edge is the base for STDD in which each sink has unique value.

Let the root node of the STDT be at level 1, the successor nodes of the

node v which is at level i are at level $i + 1$, i.e., the level is in ascending order from the top to the bottom of the graph. For an n -variable function $f(x_1, x_2, \dots, x_i, \dots, x_{n-1}, x_n)$, the non-sink node at i -th level from the root is shown in Fig. 6.2(a). In Fig. 6.2, f denotes the discrete function at level i in the STDT, f_l and f_h represent the subfunctions of f which are corresponding to the subtrees determined by low-edge and high-edge, respectively. The relationships between f , f_l and f_h are:

$$f = \begin{bmatrix} \text{value}(\text{low-edge}) & \text{value}(\text{high-edge}) \end{bmatrix} \begin{bmatrix} f_l & f_h \end{bmatrix}^T \quad (6.10)$$

$$= \begin{cases} \frac{1}{2}[f_l + (1 - 2\dot{x}_i)f_h] & \text{for } 1 \leq i \leq r, \\ f_l + \dot{x}_i f_h & \text{for } r < i \leq n, \end{cases} \quad (6.11)$$

where $\dot{x}_i$ given in Definition 6.2.2 is dependent on the polarity ω_i of the arithmetic-Walsh transform. The values of the sinks in arithmetic-Walsh Decision Tree are corresponding to the multi-polarity arithmetic-Walsh spectrum of the discrete function in sequential order. According to the properties of multi-polarity arithmetic-Walsh transform, all the sinks values can be divided by 2^r for the simplicity of the edge-values in the first case in (6.11) that means removing the coefficient $1/2$ from the equation. This makes the structure of the Decision Tree more compact.

The two cases of the node described by (6.11) are shown in Fig. 6.2(b) and Fig. 6.2(c), respectively. When $1 \leq i \leq r$, the corresponding parts can be considered as multi-polarity Walsh parts. When $r < i \leq n$, the corresponding parts can be considered as multi-polarity arithmetic parts.

Example 6.5.1. Considering the 3-variable function f_1 and arithmetic-Walsh

Figure 6.3: Arithmetic-Walsh spectral transform decision tree of function f_1 where $r = 2$, $n = 3$, and $\omega_3 = \{\omega_1, \omega_2, \omega_3\} = \{0, 1, 1\}$.

Figure 6.4: Elimination rule of arithmetic-Walsh STDDs.

transform $\mathbf{MAW}_2^{\omega_3}(3)$ in Example 6.2.1, the corresponding arithmetic-Walsh spectral transform decision tree is shown in Fig. 6.3. $\mathbf{S}_{f_1}$ in the figure is the arithmetic-Walsh spectrum of function f_1 which has been calculated using (6.6). Hence, the $\mathbf{S}_{f_1}$ which represent the values of the sinks in Fig. 6.3 can be replaced by detail numbers. The arithmetic-Walsh STDDs can be obtained from merging the sinks with the same values.

The reduction of the size is one of the most important problem in DDs techniques. The aim of the STDDs is not only to try to provide a simple representation of discrete functions, but also to find a number of useful properties

Figure 6.5: Merging rule of arithmetic-Walsh STDDs.

Figure 6.6: Divisor rule of arithmetic-Walsh STDDs.

by transforming the information to spectral domain. Hence, the simplification of the STDDs is helpful for the graphical representations of spectral transforms for discrete functions and finding more properties. There are different reduction rules for the different kinds of DDs, but some general rules for the reduction also exist.

The following reduction rules can be used for arithmetic-Walsh STDDs:

Elimination rule:

- If two outgoing edges of a node v point to the same node, then eliminate node v and redirect all the incoming edges of v to u , while the edge-values of the new incoming edges of v are the products of the old edge-values of the incoming edges of v and the sum of the old edge-values of the outgoing edges of v .
- If the node v are in the form of Fig. 6.2 (c) and the $Child_h(v)$ points to a sink whose value is 0, then eliminate node v and redirect all the incoming edges of v with their edge-values to node u which is determined

Figure 6.7: Reduced arithmetic-Walsh STDD of f_1 .

by $Child_l(v)$.

The two cases of Elimination rule are illustrated in Fig. 6.4(a) and Fig. 6.4(b), respectively.

Merging rule: If the two subtrees in the graph are isomorphic, then one of them can be deleted and the incoming edges of it with their edge-values can be redirected to the remaining one. This rule is illustrated in Fig. 6.5.

Divisor rule: If the common divisors of all the sink values can be detected, then use the greatest common divisor to divide all the sink values to get the updated sink values. The divisor is sent to the common coefficients of the sinks. This rule is illustrated in Fig. 6.6.

Using the above reduction rules, the reduced arithmetic-Walsh STDD of f_1 in Example 6.5.1 is shown in Fig. 6.7.

By using the above reduction rules, the reduced arithmetic-Walsh STDD can be obtained from the arithmetic-Walsh STDT.

Example 6.5.2. The reduced arithmetic-Walsh STDD for function f_1 in Exam-

ple 6.5.1 which is shown in Fig. 6.7 can be obtained from the arithmetic-Walsh STDT shown in Fig. 6.3 by using reduction rules. From the reduced arithmetic-Walsh STDD, the corresponding arithmetic-Walsh expansion can be obtained which is the same with the expansion shown in (6.7).

6.6 Experimental Results

There are several known transforms which have been applied to many areas including logic design. The Walsh and arithmetic transforms are most widely used [Kar76, SSJ98, SF96, SSS96, SA03]. Our proposed transforms have been introduced in previous sections, and this novel multi-polarity transform can also be applied to the logic design as well. By using different transforms different spectra for benchmark functions can be obtained, but there must be some transforms that can produce simpler spectra than others.

Table 6.2 shows the comparison of the number of non-zero coefficients in some MCNC benchmark functions [Yan91] used in logic design for the multi-polarity arithmetic-Walsh, Walsh and arithmetic transforms. The benchmark functions shown in Table 6.2 are multi-output functions. Since each output of the multi-output benchmark functions can be considered as a single function, the NAME- i represents the i -th output of the benchmark function 'NAME'. From this table, we can compare the numbers of non-zero coefficients for those three transforms in every benchmark function. The minimum numbers of non-zero coefficients for each benchmark function are indicated by boldface in Table 6.2. For benchmark function 5xp1-6, squar5-7 and z5xp1- i ($i = 4$ to 7), the multi-polarity arithmetic-Walsh transform provides the smallest number of non-zero

coefficients. Otherwise, it provides the same number of non-zero coefficients as multi-polarity arithmetic or Walsh transform for the other functions in Table 6.2.

6.7 Summary

The new hybrid multi-polarity arithmetic-Walsh transform that combines properties of both multi-polarity Walsh and multi-polarity arithmetic transforms is introduced. The new multi-polarity arithmetic-Walsh transform and its corresponding arithmetic-Walsh STDD are expected to be used in the areas in which Walsh and arithmetic transforms as well as their decision diagrams have been used, especially for CAD tools in logic design. The experimental results for some MCNC benchmark functions also show the advantages of the new transform over multi-polarity Walsh and multi-polarity arithmetic transforms.

Table 6.2: Comparison of Number of Non-Zero Coefficients for Arithmetic-Walsh transform

Function	arithmetic-Walsh	Walsh	arithmetic
5xp1-1	13	80	13
5xp1-2	33	128	33
5xp1-3	41	65	41
5xp1-4	27	33	27
5xp1-5	17	17	17
5xp1-6	6	9	9
5xp1-7	3	5	3
5xp1-8	2	2	3
5xp1-9	1	2	1
5xp1-10	3	128	3
squar5-1	3	32	3
squar5-2	4	32	4
squar5-3	6	32	6
squar5-4	8	23	8
squar5-5	9	16	9
squar5-6	5	10	5
squar5-7	2	4	3
squar5-8	1	4	1
z5xp1-1	3	128	3
z5xp1-2	13	80	13
z5xp1-3	33	128	33
z5xp1-4	22	65	41
z5xp1-5	15	33	27
z5xp1-6	10	17	17
z5xp1-7	6	9	9
z5xp1-8	3	5	3
z5xp1-9	2	2	3
z5xp1-10	1	2	1

Chapter 7

Multi-Polarity Arithmetic-Haar Transform

7.1 Introduction

Both the Haar wavelet transform (non-normalized version of the transform where only signs are entered into the transform matrix) and arithmetic transform have been used in many applications of logic design [Fal99a, Fal99b, Kar76, KSA00, Sas93, Sas99, SF96, SA03]. In [SMA01], an idea of a combined arithmetic-Haar transform was proposed. Such a transform was defined for the first eight functions in zero-polarity and experimental results shown in [SMA01] proved that arithmetic-Haar transform is more efficient than other used transforms in logic design such as Walsh, Haar and arithmetic transforms for some benchmark functions. Therefore it is interesting not only theoretically but also practically, to extend this arithmetic-Haar transform for higher matrix dimensions and from zero-polarity to multi-polarity.

7.2 Definitions of Multi-Polarity Arithmetic-Haar Transform

Let $\omega_{\mathbf{n}} = \{\omega_1, \omega_2, \dots, \omega_i, \dots, \omega_{n-1}, \omega_n\}$ be n -tuple over $\text{GF}(2)$. This vector represents the polarity of multi-polarity arithmetic-Haar transform, n, i are integers and $1 \leq i \leq n$.

Definition 7.2.1. For an n -variable function $f(x_1, x_2, \dots, x_r, \dots, x_{n-1}, x_n)$, the basic functions for the multi-polarity arithmetic-Haar expansions can be combined from two sets of basic functions. The first 2^{n-1} basic functions are generated from the positive Davio expansion [DDT78] for variables $\dot{x}_1$ to $\dot{x}_n$, excluding $\dot{x}_r$ where $1 \leq r \leq n$:

$$[1 \quad \dot{x}_1] \otimes [1 \quad \dot{x}_2] \otimes \dots \otimes [1 \quad \dot{x}_{r-1}] \otimes [1 \quad \dot{x}_{r+1}] \otimes \dots \otimes [1 \quad \dot{x}_{n-1}] \otimes [1 \quad \dot{x}_n] \quad (7.1)$$

For the other 2^{n-1} basic functions, the functions can be generated from multiplying the Shannon expansion for variables x_1 to x_n , excluding x_r :

$$[\bar{x}_1 \quad x_1] \otimes [\bar{x}_2 \quad x_2] \otimes \dots \otimes [\bar{x}_{r-1} \quad x_{r-1}] \otimes [\bar{x}_{r+1} \quad x_{r+1}] \otimes \dots \otimes [\bar{x}_{n-1} \quad x_{n-1}] \otimes [\bar{x}_n \quad x_n] \quad \text{by } (1 - 2\dot{x}_r). \quad (7.2)$$

$$\text{In (7.1) and (7.2), } \dot{x}_i = \begin{cases} x_i, & \text{if } \omega_i = 0, \\ \bar{x}_i, & \text{if } \omega_i = 1, \end{cases} \quad \text{where } 1 \leq i \leq n.$$

Such an approach allows us to generalize the multi-polarity arithmetic-Haar matrices and corresponding expansions in many ways by selecting the polarity $\omega_{\mathbf{n}}$ and the order r , so this is a general method that can provide compact

spectral representation with many zeros for any n -variable logic function.

Definition 7.2.2. From the multi-polarity arithmetic-Haar expansions, the r th-order multi-polarity arithmetic-Haar transform matrix $\mathbf{AH}_r(n)$ and its inverse $\mathbf{AH}_r^{-1}(n)$ can be defined as:

$$\mathbf{AH}_r(n) = \begin{bmatrix} \left(\bigotimes_{i=1}^{r-1} \mathbf{A}_{\omega_i} \right) \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r} \mathbf{A}_{\omega_{i+r}} \right) : \\ \left(\bigotimes_{i=1}^{r-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{B}_{\omega_r} \otimes \left(\bigotimes_{i=1}^{n-r} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \end{bmatrix} \quad (7.3)$$

$$\mathbf{AH}_r^{-1}(n) = \frac{1}{2} \begin{bmatrix} \left(\bigotimes_{i=1}^{r-1} \mathbf{A}_{\omega_i}^{-1} \right) \otimes \begin{bmatrix} 1 & 1 \end{bmatrix} \otimes \left(\bigotimes_{i=1}^{n-r} \mathbf{A}_{\omega_{i+r}}^{-1} \right) \\ \left(\bigotimes_{i=1}^{r-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{B}_{\omega_r}^T \otimes \left(\bigotimes_{i=1}^{n-r} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \end{bmatrix} \quad (7.4)$$

$$\mathbf{A}_{\omega_k} = \begin{cases} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} & \text{if } \omega_k = 0, \\ \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} & \text{if } \omega_k = 1, \end{cases} \quad \mathbf{A}_{\omega_k}^{-1} = \begin{cases} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} & \text{if } \omega_k = 0, \\ \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} & \text{if } \omega_k = 1, \end{cases}$$

$$\mathbf{B}_{\omega_k} = \begin{cases} \begin{bmatrix} 1 \\ -1 \end{bmatrix} & \text{if } \omega_k = 0, \\ \begin{bmatrix} -1 \\ 1 \end{bmatrix} & \text{if } \omega_k = 1, \end{cases} \quad \text{and } \mathbf{B}_{\omega_k}^T = \begin{cases} \begin{bmatrix} 1 & -1 \end{bmatrix} & \text{if } \omega_k = 0, \\ \begin{bmatrix} -1 & 1 \end{bmatrix} & \text{if } \omega_k = 1, \end{cases}$$

where $n = 2, 3, 4 \dots$, $1 \leq r \leq n$ and $1 \leq k \leq n$.

In the above equations, the symbol $\bigotimes_{i=1}^j$ represents the Kronecker product of j matrices. When the Kronecker product of j matrices is carried out for the above equations for $j = 0$, then the term $\bigotimes_{i=1}^j$ disappears from the above equations. The vertical dotted lines denote the layered vertical Kronecker matrices, and the horizontal dashed lines denote the layered horizontal Kronecker matrices, respectively. The layered horizontal Kronecker matrix is defined as the horizontal sum of Kronecker matrices [Yar85], and the layered vertical Kronecker matrix is defined as the vertical sum of Kronecker matrices [Fal99c].

Example 7.2.1. For $n = 3$, $r = 2$ and $\omega_3 = \{\omega_1, \omega_2, \omega_3\} = \{0, 1, 1\}$, the arithmetic-Haar forward transform matrix $\mathbf{AH}_2(3)$ is:

$$\mathbf{AH}_2(3) = \left[\begin{array}{c} \left[\begin{array}{cc} 1 & 0 \\ 1 & 1 \end{array} \right] \otimes \left[\begin{array}{c} 1 \\ 1 \end{array} \right] \otimes \left[\begin{array}{cc} 1 & 1 \\ 1 & 0 \end{array} \right] \\ \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] \otimes \left[\begin{array}{c} -1 \\ 1 \end{array} \right] \otimes \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] \\ \vdots \end{array} \right] = \begin{bmatrix} 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

and its inverse $\mathbf{AH}_2^{-1}(3)$ is:

$$\begin{aligned} \mathbf{AH}_2^{-1}(3) &= \frac{1}{2} \left[\begin{array}{c} \left[\begin{array}{cc} 1 & 0 \\ -1 & 1 \end{array} \right] \otimes \left[\begin{array}{cc} 1 & 1 \end{array} \right] \otimes \left[\begin{array}{cc} 0 & 1 \\ 1 & -1 \end{array} \right] \\ \hline \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] \otimes \left[\begin{array}{cc} -1 & 1 \end{array} \right] \otimes \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] \end{array} \right] \\ &= \frac{1}{2} \left[\begin{array}{cccccccc} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 & 0 & 1 \\ -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{array} \right]. \end{aligned}$$

Definition 7.2.3. For an n -variable function $f(x_1, x_2, \dots, x_r, \dots, x_{n-1}, x_n)$ whose truth vector is $\mathbf{F}$, its arithmetic-Haar spectrum $\mathbf{S}_f$ is defined as follow:

$$\mathbf{S}_f = \mathbf{AH}_r^{-1}(n)\mathbf{F} \quad (7.5)$$

The values in the spectrum are called the arithmetic-Haar coefficients of the function.

Comparatively, the most known transform which has been applied to many areas including logic synthesis and optimization is Walsh transform. Similarly, the proposed multi-polarity arithmetic-Haar transform can also be used for

the logic design. The following example shows the applications of Walsh and arithmetic-Haar transforms for the same function.

Example 7.2.2. Consider a 3-variable function $f_1(x_1, x_2, x_3) = \sum(0, 2, 4, 5, 7)$. Using the multi-polarity arithmetic-Haar transform in Example 7.2.1 and corresponding 8-point Walsh transform for f_1 , the corresponding arithmetic-Haar coefficients and Walsh coefficients can be calculated respectively as follows:

$$\mathbf{S}_f = \mathbf{A}\mathbf{H}_2^{-1}(3) \times \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 \\ 2 \\ 2 \\ -3 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix}, \quad \left(\bigotimes_{i=1}^3 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \right)^{-1} \times \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 5 \\ 1 \\ 1 \\ 1 \\ -1 \\ 3 \\ -1 \\ -1 \end{bmatrix}.$$

Using (7.1) and (7.2) the corresponding arithmetic-Haar expansion for the function f_1 is:

$$f_1 = \frac{1}{2} [2\bar{x}_3 + 2x_1 - 3x_1\bar{x}_3 - x_1\bar{x}_3(1 - 2\bar{x}_2)]. \quad (7.6)$$

From the Walsh coefficients calculated above, the corresponding Walsh polynomial expansion for the function f_1 is:

$$f_1 = \frac{1}{8} [5 + (1 - 2x_3) + (1 - 2x_2) + (1 - 2x_2)(1 - 2x_3) - (1 - 2x_1) + 3(1 - 2x_1)(1 - 2x_3) - (1 - 2x_1)(1 - 2x_2) - (1 - 2x_1)(1 - 2x_2)(1 - 2x_3)]. \quad (7.7)$$

In Example 7.2.2, it is obvious that the arithmetic-Haar coefficients have

less non-zero coefficients than the Walsh coefficients, so the arithmetic-Haar transform and the corresponding polynomial expansion are more efficient than the Walsh transform and its expansion.

7.3 Fast Algorithms and Computational Costs

Since the multi-polarity arithmetic-Haar transform has the recursive relations given in the form of layered Kronecker matrices, it is possible to derive fast algorithm for the calculation of the multi-polarity arithmetic-Haar transform matrices. Similar to the fast Walsh transform and other known fast transforms, the multi-polarity arithmetic-Haar transform matrix can be calculated by the products of the factored matrices. The fast algorithms for the forward and inverse multi-polarity arithmetic-Haar transforms are presented in the following properties.

The fast algorithms give us a way to calculate the transform more efficiently. Furthermore, the butterfly diagrams can be developed based on the factorized matrices, which can be used for the hardware implementation of the transform.

Property 7.3.1. The forward multi-polarity arithmetic-Haar transform $\mathbf{AH}_r(n)$ can be factorized as:

$$\mathbf{AH}_r(n) = \prod_{i=1}^n \mathbf{D}_r^i(n). \quad (7.8)$$

where

$$\mathbf{D}_r^i(n) = \begin{cases} \left(\bigotimes_{j=1}^{r-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{W}_{\omega_r} \otimes \left(\bigotimes_{j=1}^{n-r} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) & \text{for } i = 1, \\ \left(\bigotimes_{j=1}^{r-i} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{C}_{\omega_{i-1}} \otimes \left(\bigotimes_{j=1}^{n+i-r-2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) & \text{for } 1 < i \leq r, \\ \left[\begin{bmatrix} 1 & 0 \end{bmatrix} \otimes \left(\bigotimes_{j=1}^{i-2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{A}_{\omega_i} \otimes \left(\bigotimes_{j=1}^{n-i} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \right] & \\ \vdots & \\ \left[\begin{bmatrix} 0 & 1 \end{bmatrix} \otimes \left(\bigotimes_{j=1}^{n-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \right] & \text{for } r < i \leq n. \end{cases} \quad (7.9)$$

In the above equation,

$$\mathbf{A}_{\omega_k} = \begin{cases} \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} & \text{if } \omega_k = 0, \\ \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} & \text{if } \omega_k = 1, \end{cases} \quad \mathbf{W}_{\omega_k} = \begin{cases} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} & \text{if } \omega_k = 0, \\ \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} & \text{if } \omega_k = 1, \end{cases}$$

$$\mathbf{C}_{\omega_k} = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{if } \omega_k = 0, \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{if } \omega_k = 1, \end{cases} \quad \text{and } 1 \leq k \leq n.$$

Property 7.3.2. The inverse multi-polarity arithmetic-Haar transform $\mathbf{A}\mathbf{H}_r^{-1}(n)$ can be factorized as:

$$\mathbf{A}\mathbf{H}_r^{-1}(n) = \frac{1}{2} \prod_{i=1}^n \mathbf{E}_r^i(n) \quad (7.10)$$

where

$$\mathbf{E}_r^i(n) = \begin{cases} \left[\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \left(\bigotimes_{j=1}^{n-i-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{Q}_{\omega_{n+1-i}}^{-1} \otimes \left(\bigotimes_{j=1}^{i-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \right] & \text{for } 1 \leq i < n-r+1, \\ \left[\begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \left(\bigotimes_{j=1}^{n-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \right] & \text{for } i = n-r+1, \\ \left(\bigotimes_{j=1}^{r+i-n-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{C}_{\omega_{n-i}}^{-1} \otimes \left(\bigotimes_{j=1}^{2n-i-r-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) & \text{for } n-r+1 \leq i < n, \\ \left(\bigotimes_{j=1}^{r-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{W}_{\omega_r} \otimes \left(\bigotimes_{j=1}^{n-r} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) & \text{for } i = n. \end{cases} \quad (7.11)$$

In the above equation,

$$\mathbf{Q}_{\omega_k}^{-1} = \begin{cases} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} & \text{if } \omega_k = 0, \\ \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} & \text{if } \omega_k = 1, \end{cases} \quad \mathbf{C}_{\omega_k}^{-1} = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{if } \omega_k = 0, \\ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{if } \omega_k = 1, \end{cases}$$

$\mathbf{W}_{\omega_k}$ is defined in Property 7.3.1, and $1 \leq k \leq n$.

Such fast algorithms will greatly reduce the number of arithmetic operations as compared to the computation of multi-polarity arithmetic-Haar transform by the whole matrix. The computational costs mean the number of additions and subtractions required for the generation of forward and inverse transforms. Using the fast algorithms, the computational costs for all the cases of the multi-polarity arithmetic-Haar transform are $(n + 3)2^{n-1}$, which are not dependent on r and its polarity ω_n , but only dependent on n .

There is another method to generate the fast algorithm for the multi-polarity arithmetic-Haar transform when $r = n - 1$.

Property 7.3.3. The forward multi-polarity arithmetic-Haar transform matrix $\mathbf{AH}_{n-1}(n)$ can be factorized as:

$$\mathbf{AH}_{n-1}(n) = \prod_{i=1}^n \mathbf{P}_{n-1}^i(n). \quad (7.12)$$

where

$$\mathbf{P}_{n-1}^i(n) = \begin{cases} \left(\bigotimes_{j=1}^{n-2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{W}_{\omega_{n-1}} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \text{for } i = 1 \\ \left(\bigotimes_{j=1}^{n-2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{G}_{\omega_n} & \text{for } i = 2 \\ \left(\bigotimes_{j=1}^{n-i} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{C}_{\omega_{i-2}} \otimes \left(\bigotimes_{j=1}^{i-2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) & \text{for } 2 < i \leq n \end{cases} \quad (7.13)$$

$$\text{In the above equation, } \mathbf{G}_{\omega_k} = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{if } \omega_k = 0, \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{if } \omega_k = 1, \end{cases}$$

$\mathbf{W}_{\omega_k}$ and $\mathbf{C}_{\omega_k}^{-1}$ are same as those have been defined in Property 7.3.1 and Property 7.3.2, and $1 \leq k \leq n$.

Property 7.3.4. The inverse multi-polarity arithmetic-Haar transform $\mathbf{AH}_{n-1}^{-1}(n)$ can be factorized as:

$$\mathbf{AH}_{n-1}^{-1}(n) = \frac{1}{2} \prod_{i=1}^n \mathbf{Q}_{n-1}^i(n). \quad (7.14)$$

where

$$\mathbf{Q}_{n-1}^i(n) = \begin{cases} \left(\left(\bigotimes_{j=1}^{i-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{C}_{\omega_{n-i-1}}^{-1} \otimes \left(\bigotimes_{j=1}^{n-i-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \right) & \text{for } 1 \leq i < n-1 \\ \left(\bigotimes_{j=1}^{n-2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{G}_{\omega_n}^{-1} & \text{for } i = n-1 \\ \left(\bigotimes_{j=1}^{n-2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes \mathbf{W}_{\omega_{n-1}} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \text{for } i = n \end{cases} \quad (7.15)$$

$$\text{In the above equation, } \mathbf{G}_{\omega_k}^{-1} = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{if } \omega_k = 0, \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{if } \omega_k = 1, \end{cases}$$

$\mathbf{W}_{\omega_k}$ and $\mathbf{C}_{\omega_k}$ are the same as those defined in Property 7.3.1 and Property 7.3.2, and $1 \leq k \leq n$.

The fast algorithms shown in Property 7.3.3 and Property 7.3.4 for multi-polarity arithmetic-Haar transform where $r = n - 1$ require the same computational costs as the fast algorithms shown in Property 7.3.1 and Property 7.3.2. From these properties, it can be seen that there are some relationships between these two methods of fast algorithms for multi-polarity arithmetic-Haar

transform when $r = n - 1$.

Property 7.3.5. From Property 7.3.1 and Property 7.3.3, the relationships between the two methods of fast algorithms for the forward multi-polarity arithmetic-Haar transform when $r = n - 1$ can be found. The relationships between the factorized matrices of the multi-polarity arithmetic-Haar transform in (7.8) and (7.12) are shown as follows:

$$\mathbf{D}_{n-1}^1(n) = \mathbf{P}_{n-1}^1(n), \quad (7.16)$$

$$\mathbf{D}_{n-1}^i(n) = \mathbf{P}_{n-1}^{i+1}(n) \quad \text{where } 1 < i < n. \quad (7.17)$$

Property 7.3.6. From Property 7.3.2 and Property 7.3.4, the relationships between the two methods of fast algorithms for the inverse multi-polarity arithmetic-Haar transform when $r = n - 1$ can be found. The relationships between the factorized matrices of the multi-polarity arithmetic-Haar transform in (7.10) and (7.14) are shown as follows:

$$\mathbf{E}_{n-1}^i(n) = \mathbf{Q}_{n-1}^{i-1}(n) \quad \text{where } 1 < i < n, \quad (7.18)$$

$$\mathbf{E}_{n-1}^n(n) = \mathbf{Q}_{n-1}^n(n). \quad (7.19)$$

The fast algorithms for the multi-polarity arithmetic-Haar transform have been proposed. The generalized fast algorithms which can be used for all the multi-polarity arithmetic-Haar transforms are shown in Property 7.3.1 and Property 7.3.2, while another method of fast algorithms for the multi-polarity arithmetic-Haar transform where $r = n - 1$ is shown in Property 7.3.3 and Property 7.3.4. Since the arithmetic-Haar transform proposed in [SMA01] can be considered as one case of the multi-polarity arithmetic-Haar transform where $r = 2$ and $n = 3$, the fast algorithm used in [SMA01] follows the fast algorithms

Figure 7.1: Fast butterfly diagrams for the forward and inverse arithmetic-Haar transforms, $\omega_3 = \{0, 1, 1\}$, $r = 2$ and $n = 3$.

shown in Property 7.3.3 and Property 7.3.4.

The fast algorithms for the multi-polarity arithmetic-Haar transform can be represented in the form of butterfly flow diagrams. Those structures allow to make the hardware implementation of the multi-polarity arithmetic-Haar transform to be realized in a manner similar to other known transforms.

Example 7.3.1. The multi-polarity arithmetic-Haar transform $\mathbf{AH}_2(3)$ in Example 7.2.1 can be factorized to three matrices by fast algorithms shown in Property 7.3.1:

$$\mathbf{AH}_2(3) = \mathbf{D}_2^1(3)\mathbf{D}_2^2(3)\mathbf{D}_2^3(3),$$

$$\begin{aligned}
\text{where } \mathbf{D}_2^1(3) &= \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}, \\
\mathbf{D}_2^2(3) &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } \mathbf{D}_2^3(3) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned}$$

The inverse multi-polarity arithmetic-Haar transform $\mathbf{AH}_2^{-1}(3)$ can be also factorized in a similar way using fast algorithm shown in Property 7.3.2. From the representations of $\mathbf{AH}_2(3)$ and $\mathbf{AH}_2^{-1}(3)$ by products of the factored matrices, the corresponding fast flow diagrams for the calculation of forward and inverse multi-polarity arithmetic-Haar transforms are shown in Fig. 7.1(a) and Fig. 7.1(b), accordingly. In all the figures in this chapter, the solid lines and dotted lines represent addition and subtraction, respectively. Since this multi-polarity arithmetic-Haar transform ($r = 2$, $n = 3$) satisfies the condition $r = n - 1$, $\mathbf{AH}_2(3)$ and $\mathbf{AH}_2^{-1}(3)$ can be factorized by the fast algorithms shown in Property 7.3.3 and Property 7.3.4 as well. The corresponding fast

Figure 7.2: Another fast butterfly diagrams for the forward and inverse arithmetic-Haar transforms, $\omega_3 = \{0, 1, 1\}$, $r = 2$ and $n = 3$.

flow diagrams using these other fast algorithms are shown in Fig. 7.2. The relationships between these two methods of fast algorithms shown in Property 7.3.5 and Property 7.3.6 can be obtained by comparing the butterfly diagrams in Fig. 7.1 and Fig. 7.2.

The structure of the butterfly diagrams shown in Fig. 7.2 are a little similar to the butterfly diagrams of the Walsh transform which is one of the discrete transforms most often used in logic design. In comparison with the multi-polarity arithmetic-Haar transform, the fast diagrams for calculation of forward and inverse Walsh transforms for 3-variable functions are shown in Fig. 7.3(a) and Fig. 7.3(b), accordingly. Comparing Fig. 7.1 and Fig. 7.3, or Fig. 7.2 and Fig. 7.3, the computational advantages of the multi-polarity arithmetic-Haar transform over known Walsh transform are clearly seen as the total number of operations required to perform the new transform are less than the ones for Walsh transform. Using the corresponding fast algorithms, the computational

Figure 7.3: Fast butterfly diagrams for the forward and inverse Walsh transforms where $n = 3$.

costs of multi-polarity Walsh and arithmetic transforms are known as $n2^{n+1}$ and $n2^n$, respectively. From (7.9), (7.11), (7.13), and (7.15), the computational costs for the calculation of forward and inverse multi-polarity arithmetic-Haar transform can be calculated as: $2 \times [2^n + (r-1) \times 2^{n-2} + (n-r) \times 2^{n-2}] = (n+3)2^{n-1}$. Hence, the computational costs of multi-polarity Walsh and arithmetic transforms increase considerably when compared with the multi-polarity arithmetic-Haar transform for higher n as shown in Table 7.1. Based on the computational costs required by different transforms, the reduction in computational costs of arithmetic-Haar transform comparing with Walsh and arithmetic transforms can also be calculated as shown in Figure 7.4. It can be noted that the reductions in computational costs of arithmetic-Haar transform when $n > 5$ are above 20% when comparing with arithmetic transform and above 60% when comparing with Walsh transform.

Table 7.1: Comparison of Computational Costs for Arithmetic-Haar Transform

n	arithmetic-Haar $(n + 3)2^{n-1}$	Walsh $n2^{n+1}$	arithmetic $n2^n$
2	10	16	8
3	24	48	24
4	56	128	64
5	128	320	160
6	288	768	384
7	640	1792	896
8	1408	4096	2048
9	3072	9216	4608
10	6656	20480	10240
11	14336	45056	22528
12	30720	98304	49152
13	65536	212992	106496
14	139264	458752	229376
15	294912	983040	491520
16	622592	2097152	1048576

7.4 Calculation through Decision Diagrams

Decision diagrams (DDs) are data structures which are widely used in logic synthesis and optimization algorithms since they provide compact representation of discrete logic functions. Based on the fast algorithms described in Section 7.3, the method for calculation of the multi-polarity arithmetic-Haar spectrum through DDs can be proposed. The method using DDs is more suitable for the calculation of the spectrum of large functions than the fast algorithms. The representation of a Boolean function by DDs are derived by the application of the reduction to decision trees (DTs).

The arithmetic-Haar coefficients for Boolean functions can be calculated through DT or DD.

Example 7.4.1. By using the arithmetic-Haar transform in Example 7.2.1,

Figure 7.4: Reduction in computational costs of arithmetic-Haar transform comparing with Walsh and arithmetic transforms.

the calculation of multi-polarity arithmetic-Haar spectrum $\mathbf{S}_f$ through BDT is shown in Fig. 7.5. For the 3-variable function $f_1(x_1, x_2, x_3)$ in Example 7.2.2, its corresponding arithmetic-Haar spectrum $\mathbf{S}_f$ can be calculated through BDD as shown in Fig. 7.6.

7.5 Arithmetic-Haar Spectral Transform Decision Diagrams

Spectral transform decision diagrams (STDDs) are used to represent the spectra of the given discrete functions. It is more efficient to calculate especially for large discrete functions through STDDs than using butterfly diagrams. Several kinds

Figure 7.5: Calculation of multi-polarity arithmetic-Haar coefficients through DT.

Figure 7.6: Calculation of multi-polarity arithmetic-Haar coefficients for f_1 through DD.

of STDDs have been developed, and they are the graphical representations for the spectra of the discrete functions. A new type of STDDs named arithmetic-Haar spectral transform decision diagram is proposed in this section, which is STDD for the multi-polarity arithmetic-Haar transform.

Definition 7.5.1. For an n -variable function $f(x_1, x_2, \dots, x_i, \dots, x_{n-1}, x_n)$, the arithmetic-Haar spectral transform decision diagram is a directed acyclic graph with exactly one root. It consists of two types of nodes:

- *Sink* is the terminal node which is located in the bottom of the graph. It has not outgoing edges, while it has integer value. The number of sinks in

one arithmetic-Haar spectral transform decision diagram is in the range of $[1, 2^n]$.

- *Non-sink* node is the non-terminal node which has two outgoing edges, which are called low-edge and high-edge, respectively. Both the edges of the Non-sink node v have its values denoted by $value(edge)$ which are called edge-values.

The successor nodes determined by the low-edge and high-edge are denoted by $Child_l(v)$ and $Child_h(v)$, respectively.

From the relationships between arithmetic-Haar spectral and logical interpretations of the discrete functions, the spectral interpretation of the multi-polarity arithmetic-Haar transform through STDDs can be derived.

Definition 7.5.2. The multi-polarity arithmetic-Haar spectral transform decision tree (STDT) in which each node has at most one incoming edge is the base for STDD in which each Sink has unique value.

Let the root node of the STDT be at level 1, the successor nodes of the node v which is at level i are at level $i + 1$, i.e., the level is in ascending order from the top to the bottom of the graph. For an n -variable function $f(x_1, x_2, \dots, x_i, \dots, x_{n-1}, x_n)$, the Non-sink node at i -th level from the root is shown in Fig. 7.7(a). In Fig. 7.7, f denotes the discrete function at level i in the STDT, f_l and f_h represent the subfunctions of f which are corresponding to the subtrees determined by low-edge and high-edge, respectively. Let the TR represent the tree corresponding to the function f , TR_L and TR_R represent the subtrees determined by low-edge and high-edge of the root, respectively. The

Figure 7.7: Arithmetic-Haar spectral transform decision tree node at level i : (a) general form, (b) $i = 1$, (c) $2 \leq i \leq n$ and TR in TR_L , (d) $2 \leq i \leq n$ and TR in TR_R .

relationships between f , f_l and f_h are:

$$f = \begin{bmatrix} \text{value}(\text{low-edge}) & \text{value}(\text{high-edge}) \end{bmatrix} \begin{bmatrix} f_l & f_h \end{bmatrix}^T \quad (7.20)$$

$$= \begin{cases} \frac{1}{2}[f_l + (1 - 2\dot{x}_r)f_h] & \text{for } i = 1, \\ f_l + \dot{x}_i f_h & \text{for } 2 \leq i \leq n \text{ and } TR \text{ in } TR_L \\ \bar{x}_i f_l + x_i f_h & \text{for } 2 \leq i \leq n \text{ and } TR \text{ in } TR_R \end{cases} \quad (7.21)$$

where $\dot{x}_i$ given in Definition 7.2.1 is dependent on the polarity ω_i of the arithmetic-Haar transform. The values of the sinks in arithmetic-Haar Decision Tree are corresponding to the multi-polarity arithmetic-Haar spectrum of the discrete function in sequential order. According to the properties of multi-polarity arithmetic-Haar transform, all the values of the sinks can be divided by 2 for the simplicity of the edge-values in the first case in (7.21) that means removing the coefficient $1/2$ from the equation. This makes the structure of the Decision Tree more compact.

The cases of the node described by (7.21) are shown in Fig. 7.7(b), Fig. 7.7(c) and Fig. 7.7(d), respectively.

Figure 7.8: Arithmetic-Haar spectral transform decision tree of function f_1 where $r = 2$, $n = 3$, and $\omega_3 = \{\omega_1, \omega_2, \omega_3\} = \{0, 1, 1\}$.

Example 7.5.1. Considering the 3-variable function f_1 and arithmetic-Haar transform $\mathbf{AH}_2(3)$ in Example 7.2.1, the corresponding arithmetic-Haar spectral transform decision tree is shown in Fig. 7.8. $\mathbf{S}_{f_1}$ in the figure is the arithmetic-Haar spectrum of function f_1 which has been calculated in Example 7.2.2. Hence, the $\mathbf{S}_{f_1}$ which represents the values of the sinks in Fig. 7.8 can be replaced by detailed numbers. The arithmetic-Haar STDD can be obtained from merging the sinks with the same values.

The reduction of the size is one of the most important problem in DDs techniques. The aim of the STDDs is not only to try to provide a simple representation of discrete functions, but also to find a number of useful properties by transforming the information to the spectral domain. Hence, the simplification of the STDDs is helpful for the graphical representations of spectral transforms for discrete functions and for finding more of their properties. There are different kinds of DDs that have been developed, such as edge-valued binary decision diagram (EVBDD) [LPV94, SF96], multi-terminal binary decision dia-

Figure 7.9: Reduced arithmetic-Haar STDD of f_1 .

gram (MTBDD) [CFZ95, SF96] and other types of STDDs [SF96, SA03, SSJ98]. There are some general or similar reduction rules for different DDs.

The reductions which can be used for arithmetic-Haar STDDs include Elimination rule, Merging rule and Divisor rule. These three reduction rules have been described in Section 6.5 while Elimination rule is illustrated in Fig. 6.4, Merging rule is illustrated in Fig. 6.5, and Divisor rule is illustrated in Fig. 6.6. By using the reduction rules, the reduced arithmetic-Haar STDD can be obtained from the arithmetic-Haar STDT.

Example 7.5.2. The reduced arithmetic-Haar STDD for function f_1 in Example 7.2.2 which is shown in Fig. 7.9 can be obtained from the arithmetic-Haar STDT shown in Fig. 7.8 by using reduction rules. From the reduced arithmetic-Haar STDD, the corresponding arithmetic-Haar expansion can be obtained which is the same as the expansion shown in (7.6).

7.6 Experimental Results

Table 7.2 shows the comparison of the number of non-zero coefficients in some MCNC benchmark functions [Yan91] used in logic design for the best multi-polarity arithmetic-Haar, multi-polarity Walsh and multi-polarity arithmetic transforms. The benchmark functions shown in Table 7.2 are all multi-output functions without 9sym. Since each output of the multi-output benchmark functions can be considered as a single function, the NAME- i represents the i -th output of the benchmark function 'NAME'. The 'Zero-Polarity' means the results in the corresponding columns are obtained by using the zero-polarity ($\omega_n = \{0, 0, \dots, 0\}$) transforms, and 'Multi-Polarity' means the results in the corresponding columns are obtained by selecting the optimal results with smallest number of spectral coefficients using the multi-polarity transforms. The smallest numbers of non-zero coefficients in 'Zero-Polarity' or 'Multi-Polarity' for each benchmark function are indicated by boldface in Table 7.2.

From Table 7.2, we can compare the numbers of non-zero coefficients for those three transforms in every benchmark function. The results by using the zero-polarity arithmetic-Haar, Walsh, and arithmetic transforms are discussed here. For bw-4, clip-4 and rd84-2, the Walsh coefficients have less non-zero coefficients than the zero-polarity arithmetic-Haar and arithmetic coefficients. For bw-2 and rd84-3, the arithmetic transform provides better spectra than the other two transforms. However, there are some cases that the arithmetic-Haar and arithmetic transforms produce the same minimum number of non-zero coefficients for the benchmark functions, such as for inc- i ($i = 1, 2, 9$). For the rest of the benchmark functions, the zero-polarity arithmetic-Haar transform is more efficient than the zero-polarity Walsh and arithmetic transforms due to

the smallest numbers of non-zero coefficients.

From the results by selecting the best result by using the multi-polarity arithmetic-Haar, Walsh, and arithmetic transforms shown in Table 7.2, we can also find the multi-polarity arithmetic-Haar transform is much more efficient than the multi-polarity Walsh and arithmetic transforms for most benchmark functions. For $\text{bw-}i$ ($i = 2, 10, 13, 17, 18, 26, 27, 28$), $\text{inc-}i$ ($i = 4, 5, 6, 8$), rd84-3 and sao2-3 , the multi-polarity arithmetic transform provides better spectra with smaller number of non-zero coefficients than the other two transforms. However, there are some cases that the multi-polarity arithmetic-Haar and multi-polarity arithmetic transforms produce the same minimum number of non-zero coefficients for the benchmark functions, such as for $\text{bw-}i$ ($i = 3, 4, 6, 7, 9, 16, 20, 24$), con1-2 , and $\text{inc-}i$ ($i = 1, 2, 3, 7, 9$).

7.7 Summary

The generalizations of multi-polarity arithmetic-Haar transform for any size are presented in this chapter. By selecting the polarity and r of the multi-polarity arithmetic-Haar transform, we can find the optimal expression for a given logic function. The fast algorithms for the new transform are also developed which can calculate the transform more efficiently. The new multi-polarity transform shows both the computational advantage and better experimental results over multi-polarity Walsh and multi-polarity arithmetic transforms. The calculation of this transform through DDs can also be developed, which is more suitable for large discrete functions. The new kind of STDD corresponding to the multi-polarity arithmetic-Haar transform is also proposed to represent the arithmetic-

Haar spectra of discrete functions. It provide the data structure to represent the arithmetic-Haar spectra of the given discrete functions.

Table 7.2: Comparison of Number Non-Zero Coefficients for Arithmetic-Haar Transform.

Function	Zero-Polarity			Multi-Polarity		
	Arithmetic-Haar	Walsh	Arithmetic	Arithmetic-Haar	Walsh	Arithmetic
9sym	233	256	465	233	256	352
apex4-1	0	0	0	0	0	0
apex4-2	152	512	201	139	512	179
apex4-3	275	486	325	261	486	310
apex4-4	236	460	324	236	460	297
apex4-5	272	471	332	264	471	316
apex4-6	260	473	332	253	473	315
apex4-7	262	472	326	258	472	321
apex4-8	256	470	347	249	470	313
apex4-9	278	471	339	270	471	322
apex4-10	274	471	344	262	471	304
apex4-11	266	480	330	252	480	310
apex4-12	239	461	316	236	461	316
apex4-13	217	512	279	206	512	278
apex4-14	209	454	337	205	454	279
apex4-15	251	512	330	234	512	299
apex4-16	86	376	132	48	376	60
apex4-17	75	512	128	36	512	40
apex4-18	94	400	130	51	400	60
apex4-19	140	416	234	58	416	76
bw-1	13	28	24	12	28	15
bw-2	5	32	4	5	32	4
bw-3	18	32	29	12	32	12
bw-4	15	14	15	8	14	8
bw-5	21	32	30	13	32	14
bw-6	11	32	18	7	32	7
bw-7	20	32	32	13	32	13
bw-8	17	27	22	14	27	15
bw-9	20	32	31	10	32	10
bw-10	7	32	8	6	32	5

continued on next page

<i>continued from previous page</i>						
Function	Zero-Polarity			Multi-Polarity		
	Arithmetic-Haar	Walsh	Arithmetic	Arithmetic-Haar	Walsh	Arithmetic
bw-11	16	18	26	9	18	10
bw-12	15	32	21	11	32	15
bw-13	7	32	8	6	32	5
bw-14	19	32	29	11	32	12
bw-15	13	32	18	11	32	14
bw-16	19	32	32	10	32	10
bw-17	7	32	8	6	32	5
bw-18	20	32	30	12	32	10
bw-19	11	32	31	11	32	13
bw-20	17	32	27	9	32	9
bw-21	19	32	26	12	32	14
bw-22	15	24	28	9	24	10
bw-23	17	32	22	10	32	11
bw-24	21	32	32	14	32	14
bw-25	8	32	10	8	32	8
bw-26	21	32	30	15	32	11
bw-27	17	32	21	9	32	8
bw-28	17	32	32	2	32	1
clip-1	65	89	122	65	89	122
clip-2	106	169	119	106	169	119
clip-3	154	161	163	153	161	157
clip-4	119	89	195	115	89	166
clip-5	76	89	172	62	89	112
con1-1	11	40	12	9	40	10
con1-2	10	20	10	9	20	9
ex1010-1	664	1024	861	629	1024	787
ex1010-2	657	975	935	614	975	792
ex1010-3	660	1024	937	625	1024	758
ex1010-4	688	979	908	650	979	799
ex1010-5	645	1024	853	626	1024	804
ex1010-6	647	972	921	635	972	788
ex1010-7	679	977	879	643	977	790
ex1010-8	658	965	849	614	965	759
ex1010-9	639	1024	855	628	1024	800
<i>continued on next page</i>						

<i>continued from previous page</i>						
Function	Zero-Polarity			Multi-Polarity		
	Arithmetic-Haar	Walsh	Arithmetic	Arithmetic-Haar	Walsh	Arithmetic
ex1010-10	638	1024	854	595	1024	798
inc-1	18	45	18	12	45	12
inc-2	25	64	25	18	64	18
inc-3	42	80	45	28	80	28
inc-4	50	95	61	36	95	34
inc-5	39	128	51	23	128	22
inc-6	36	80	50	16	80	15
inc-7	37	88	45	18	88	18
inc-8	50	128	59	32	128	30
inc-9	4	16	4	2	16	2
rd84-1	163	256	191	156	256	183
rd84-2	129	2	255	129	2	255
rd84-3	2	256	1	2	256	1
rd84-4	128	256	163	125	256	158
sao2-1	208	1024	380	41	1024	43
sao2-2	324	640	768	62	640	80
sao2-3	326	856	578	55	856	51
sao2-4	509	1024	936	53	1024	62
z9sym	233	256	465	233	256	352

Chapter 8

Ternary Walsh Transform

8.1 Introduction

The standard Walsh transform is also well known as the integer-valued discrete Fourier transform and provides simpler arithmetic with keeping many useful properties in many applications [Fal99a, Fal99b, Kar76, KSA00, Sas93, Sas99, SF96, SA03]. Motivated by the interest of reducing the computational costs, computation arithmetic and required storage for the standard Walsh transform, the $GF(3)$ arithmetic is utilized to provide a simpler arithmetic and require less storage space for the calculations. By doing fundamental properties rather than merely offer design procedures we have been guided here by a mathematical spirit. The final goal of this work is the synthesis of Boolean functions (possibly incompletely specified) by means of given new elementary ternary operations. A number of important issues extending currently known standard Walsh transform and its expansions are proposed and solved here for the ternary Walsh expansion.

8.2 Definitions of Ternary Walsh Transform

The operations in the new ternary Walsh transform are all over $\text{GF}(3)$, which provides the unique and invertible properties of the transform.

8.2.1 Operations over $\text{GF}(3)$

There are two ternary operations used in the forward ternary Walsh transform and two ternary operations used in the inverse ternary Walsh transform.

Let a and b be ternary variables.

Definition 8.2.1. The following ternary operations $\oplus_3$ and $\ominus_3$ are used in forward ternary Walsh transform. The ternary operation $\oplus_3$ replaces the additions in fast Walsh transform structure, and it is defined as the modular 3 addition: $a \oplus_3 b = (a + b) \bmod 3$. The ternary operation $\ominus_3$ replaces the subtractions in fast Walsh transform structure, and it is defined as the modular 3 subtraction: $a \ominus_3 b = (3 + a - b) \bmod 3$.

Table 8.1 shows the ternary operations $\oplus_3$ and $\ominus_3$ over $\text{GF}(3)$.

Definition 8.2.2. The ternary operations $\boxplus_3$ and $\boxminus_3$ are used in inverse ternary Walsh transform. The ternary operation $\boxplus_3$ replaces the additions in fast Walsh transform structure, and it is defined as: $a \boxplus_3 b = (2a + 2b) \bmod 3$. The ternary operation $\boxminus_3$ replaces the subtractions in fast Walsh transform structure, and it is defined as: $a \boxminus_3 b = (6 + 2a - 2b) \bmod 3$.

Table 8.2 shows the ternary operations $\boxplus_3$ and $\boxminus_3$ over $\text{GF}(3)$.

Table 8.1: Forward Ternary Operations $\oplus_3$ and $\ominus_3$

$a \oplus_3 b$		b		
		0	1	2
a	0	0	1	2
	1	1	2	0
	2	2	0	1

$a \ominus_3 b$		b		
		0	1	2
a	0	0	2	1
	1	1	0	2
	2	2	1	0

8.2.2 Definitions and Properties of Ternary Walsh Transform

The ternary Walsh transform is based on the fast Walsh transform structure while the four types of ternary operations are used in the butterfly structure instead of the additions and subtractions. Let the vector $\mathbf{F}$ be the truth vector of function f , and $f(i)$ be the i -th element of the vector.

Definition 8.2.3. The forward ternary Walsh transform can be computed by replacing the additions and subtractions by ternary operations $\oplus_3$ and $\ominus_3$ in the butterfly structure of fast Walsh transform, respectively. The vector $\mathbf{S}$ computed by using ternary Walsh transform is called as ternary Walsh spectrum which is n -tuples over $\text{GF}(3)$, and $s(i)$ is the i -th element of the vector which is also known as the ternary Walsh coefficient.

Definition 8.2.4. The inverse ternary Walsh transform can be computed by replacing the additions and subtractions by ternary operations $\boxplus_3$ and $\boxminus_3$ in the butterfly structure of fast Walsh transform, respectively.

Property 8.2.1. Let $R_1 = \{0, 1\}$, $R_2 = \{0, 1, 2\}$, and R_p^q means q -space Cartesian product of a set R_p where $p = 1, 2$. If the input vectors are in domain

Table 8.2: Inverse Ternary Operations $\boxplus_3$ and $\boxminus_3$

$a \boxplus_3 b$		b		
		0	1	2
a	0	0	2	1
	1	2	1	0
	2	1	0	2

$a \boxminus_3 b$		b		
		0	1	2
a	0	0	1	2
	1	2	0	1
	2	1	2	0

Table 8.3: Multiplication Rules over GF(3)

$\times_3$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

$R_2 = \{0, 1, 2\}$, the forward and inverse ternary Walsh transforms can be seen as the mappings: $(R_2^N)_F \rightarrow (R_2^N)_S$ and $(R_2^N)_S \rightarrow (R_2^N)_F$ where the subscripts F and S correspond to functional and spectral domains, respectively. When the input vectors are the truth vectors of binary logic functions, the forward and inverse ternary Walsh transforms can be seen as the mappings: $(R_1^N)_F \rightarrow (R_2^N)_S$ and $(R_2^N)_S \rightarrow (R_1^N)_F$, respectively.

The ternary Walsh transform is a linear transform over GF(3), and it is also unique and invertible.

Example 8.2.1. For $n = 3$, let the ternary vector $\mathbf{F} = [0 \ 2 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1]^T$ where the superscript T is the matrix transpose operator. We can use ternary Walsh transform to compute its ternary Walsh spectrum $\mathbf{S}$ as shown in Fig. 8.1(a) where the $\oplus$ and $\ominus$ represents the ternary addition $\oplus_3$ and ternary subtraction $\ominus_3$, respectively, in all the figures of this chapter. Due to the invertibility of

Figure 8.1: Computations of the ternary Walsh transform where $n = 3$.

ternary Walsh transform, the vector $\mathbf{F}$ can be also recovered from $\mathbf{S}$ by using inverse Walsh transform as shown in Fig. 8.1(b) where the $\boxplus$ and $\boxminus$ represents the ternary addition $\boxplus_3$ and ternary subtraction $\boxminus_3$, respectively, in all the figures of this chapter.

Although the definitions of ternary Walsh transform are based on replacing the additions and subtractions in fast Walsh transform structure by the ternary operations, we can also use the matrix representation to express the ternary Walsh transform where both the elements of the matrix and the arithmetic operations in the matrix are also over $\text{GF}(3)$. The alternative definitions of ternary Walsh transform by modifying standard Walsh matrix are obtained as shown below.

Definition 8.2.5. The basic matrix of ternary Walsh transform is modified by using modular arithmetic over $\text{GF}(3)$, so that the forward ternary Walsh

transform matrix $\mathbf{TW}(n)$ of order 2^n can be defined as follows:

$$\mathbf{TW}(n) = \left(\bigotimes_{i=1}^n \mathbf{TW}(1) \right) \mod 3 \quad (8.1)$$

where $\mathbf{TW}(1) = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$.

Definition 8.2.6. From the definitions of inverse ternary operations $\boxplus_3$ and $\boxminus_3$, the relationships between forward and inverse ternary Walsh transforms can be obtained. The inverse ternary Walsh transform matrix $\mathbf{TW}^{-1}(n)$ of order 2^n can be defined as follows:

$$\mathbf{TW}^{-1}(n) = \left(\bigotimes_{i=1}^n \mathbf{TW}^{-1}(1) \right) \mod 3 \quad (8.2)$$

where $\mathbf{TW}^{-1}(1) = \begin{bmatrix} 2 & 2 \\ 2 & 1 \end{bmatrix}$.

Alternatively,

$$\mathbf{TW}^{-1}(n) = 2^{u_n} \times_3 \mathbf{TW}(n). \quad (8.3)$$

where $u_n = \begin{cases} 0 & \text{when } n \text{ is even,} \\ 1 & \text{when } n \text{ is odd,} \end{cases}$ while the symbol $\times_3$ represent the multipli-

cations over $\text{GF}(3)$ and its rules are shown in Table 8.3.

We can also derive the forward and inverse ternary Walsh transform matrices by apply the operation $\times_3$ in the construction of the Kronecker product.

For the forward and inverse ternary Walsh transforms, we can have: $\mathbf{TW}(n) \times_3 \mathbf{TW}^{-1}(n) = \mathbf{I}_{2^n}$ where $\mathbf{I}_{2^n}$ is the identity matrix of order 2^n .

Property 8.2.2. The matrices of forward and inverse ternary Walsh transforms are defined in Definitions 8.2.5 and 8.2.6, respectively. From (8.3) and Table 8.3, it can be observed that the inverse ternary Walsh transform matrix is the same as the forward ternary Walsh transform when n is even, and can be constructed by simply replacing the elements 1 and 2 by 2 and 1, respectively, in the corresponding forward ternary Walsh transform matrix when n is odd. The forward ternary Walsh transform matrix can be also generated from the inverse ternary Walsh transform matrix in the same way.

Property 8.2.3. The relations between the basic matrices of standard and ternary Walsh transforms can be represented as follows:

$$\mathbf{TW}(n) = (3\mathbf{O}(n) + \mathbf{W}(n)) \bmod 3, \quad (8.4)$$

$$\text{and } \mathbf{TW}^{-1}(n) = (3\mathbf{O}(n) - \mathbf{W}(n)) \bmod 3, \quad (8.5)$$

where $\mathbf{O}(n)$ is a $2^n \times 2^n$ matrix of ones. Note that all the elements in the ternary Walsh transform matrix are over $\text{GF}(3)$.

Example 8.2.2. The forward ternary Walsh matrix for $n = 2$ can be constructed using (8.1) as:

$$\mathbf{TW}(2) = \left(\bigotimes_{i=1}^2 \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \right) \bmod 3 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{bmatrix}.$$

The corresponding inverse ternary Walsh transform matrix can be calculated

using (8.2) as:

$$\begin{aligned}
 \mathbf{TW}^{-1}(2) &= \left(\bigotimes_{i=1}^3 \begin{bmatrix} 2 & 2 \\ 2 & 1 \end{bmatrix} \right) \bmod 3 \\
 &= \begin{bmatrix} 4 & 4 & 4 & 4 \\ 4 & 2 & 4 & 2 \\ 4 & 4 & 2 & 2 \\ 4 & 2 & 2 & 1 \end{bmatrix} \bmod 3 \\
 &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{bmatrix}.
 \end{aligned}$$

Note that $\mathbf{TW}^{-1}(2)$ is the the same with $\mathbf{TW}(2)$ as described in Property 8.2.2.

The standard Walsh transform matrix for $n = 2$ is:

$$\mathbf{W}(2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

Hence, according to (8.4) we have

$$\begin{aligned}
 \mathbf{TW}(2) &= (3\mathbf{O}(n) + \mathbf{W}(2)) \bmod 3 \\
 &= \begin{bmatrix} 4 & 4 & 4 & 4 \\ 4 & 2 & 4 & 2 \\ 4 & 4 & 4 & 4 \\ 4 & 2 & 4 & 4 \end{bmatrix} \bmod 3 \\
 &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{bmatrix}.
 \end{aligned}$$

Similarly, we have: $\mathbf{TW}^{-1}(2) = (3\mathbf{O}(2) - \mathbf{W}(2)) \bmod 3$. These results hold Property 8.2.3.

Example 8.2.3. The forward ternary Walsh matrix for $n = 3$ can be constructed using (8.1) as:

$$\mathbf{TW}(3) = \bigotimes_{i=1}^3 \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 & 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 & 1 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 \\ 1 & 2 & 1 & 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 1 & 2 & 2 & 1 & 2 & 1 & 1 & 2 \end{bmatrix}.$$

The corresponding inverse ternary Walsh transform matrix can be calculated

using (8.2) as:

$$\mathbf{TW}^{-1}(3) = \bigotimes_{i=1}^3 \begin{bmatrix} 2 & 2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 \\ 2 & 2 & 1 & 1 & 2 & 2 & 1 & 1 \\ 2 & 1 & 1 & 2 & 2 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 \\ 2 & 1 & 2 & 1 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 1 & 2 & 2 & 1 \end{bmatrix}.$$

By comparing $\mathbf{TW}(3)$ with $\mathbf{TW}^{-1}(3)$, it can be seen that $\mathbf{TW}^{-1}(3)$ can be generated from $\mathbf{TW}(3)$ by replacing the elements 1 and 2 by 1 and 2, respectively, and vice versa. This relationship has been described in Property 8.2.2.

The inverse ternary Walsh transform $\mathbf{TW}^{-1}(3)$ can also be generated from

the standard Walsh transform $\mathbf{W}(3)$ by using (8.5) as follows:

$$\begin{aligned}
 (3\mathbf{O}(3) - \mathbf{W}(3)) \bmod 3 &= \left(3\mathbf{O}(3) - \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \right) \bmod 3 \\
 &= \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 4 & 2 & 4 & 2 & 4 & 2 & 4 \\ 2 & 2 & 4 & 4 & 2 & 2 & 4 & 4 \\ 2 & 4 & 4 & 2 & 2 & 4 & 4 & 2 \\ 2 & 2 & 2 & 2 & 4 & 4 & 4 & 4 \\ 2 & 4 & 2 & 4 & 4 & 2 & 4 & 2 \\ 2 & 2 & 4 & 4 & 4 & 4 & 2 & 2 \\ 2 & 4 & 4 & 2 & 4 & 2 & 2 & 4 \end{bmatrix} \bmod 3 \\
 &= \mathbf{TW}^{-1}(3).
 \end{aligned}$$

Property 8.2.4. Let the j -th element of i -th row in the matrix of ternary Walsh transform $\mathbf{TW}(n)$ be represented by $TW_{i,j}$ where $0 \leq i < 2^n$ and $0 \leq j < 2^n$. The elements of ternary Walsh transform matrix can be generated as follows:

$$TW_{i,j} = 2^{\langle i,j \rangle} \quad (8.6)$$

where $\langle i, j \rangle$ is the modulo-2 inner product of the two binary representations of

Table 8.4: Modulo-2 Inner Products $\langle i, j \rangle$ when $n = 3$

i	j							
	000	001	010	011	100	101	110	111
000	0	0	0	0	0	0	0	0
001	0	1	0	1	0	1	0	1
010	0	0	1	1	0	0	1	1
011	0	1	1	0	0	1	1	0
100	0	0	0	0	1	1	1	1
101	0	1	0	1	1	0	1	0
110	0	0	1	1	1	1	0	0
111	0	1	1	0	1	0	0	1

i and j .

According to the relationships between forward and inverse ternary Walsh transform shown in Property 8.2.2, the elements of inverse ternary Walsh transform matrix can be derived as follows:

$$TW_{i,j}^{-1} = 2^{u_n \oplus \langle i, j \rangle} \quad (8.7)$$

where $\oplus$ represents the exclusive-OR (EXOR).

Example 8.2.4. As the forward ternary Walsh matrix for $n = 3$ can be constructed by using (8.6), the inner products which are required for the generation of elements of the matrix are shown in Table 8.4. The matrix $\mathbf{TW}(3)$ which can be seen as corresponding to Table 8.4 has been shown in Example 8.2.3. Similarly, the inverse ternary Walsh matrix for $n = 3$ can also be constructed by using (8.7).

Property 8.2.5. From Property 8.2.4, the matrix of ternary Walsh transform $\mathbf{TW}(n)$ is symmetric, i.e., $\mathbf{TW}(n) = \mathbf{TW}^T(n)$ or $TW_{i,j} = TW_{j,i}$, where the superscript T is the transpose symbol.

Property 8.2.6. From Property 8.2.4 and 8.2.5, for the elements of the ternary Walsh transform matrix $\mathbf{TW}(n)$, we have:

$$TW_{i,j} \times_3 TW_{i,k} = TW_{i,j \oplus k} \quad (8.8)$$

$$\text{and } TW_{i,j} \times_3 TW_{k,j} = TW_{i \oplus k, j} \quad (8.9)$$

where $\oplus$ is the bit-by-bit EXOR by the binary representations, and the multiplication $\times_3$ is performed over $\text{GF}(3)$.

Definition 8.2.7. The ternary Walsh spectrum $\mathbf{S}$ of vector $\mathbf{F}$ can be calculated by using forward ternary Walsh transform matrix as follows:

$$\mathbf{S} = \mathbf{TW}(n) \times_3 \mathbf{F} \quad (8.10)$$

Similarly, the input vector $\mathbf{F}$ can be also obtained as follows:

$$\mathbf{F} = \mathbf{TW}^{-1}(n) \times_3 \mathbf{S} \quad (8.11)$$

The additions and multiplications in (8.10) and (8.11) are over $\text{GF}(3)$ that correspond to the additions and multiplications over $\text{GF}(3)$ shown in Table 8.1 and 8.3.

Property 8.2.7. The calculation of input vector $\mathbf{F}$ from its ternary Walsh spectrum by using (8.11) is shown in Definition 8.2.7. Let us rewrite (8.11) by using (8.3) as:

$$\begin{aligned} \mathbf{F} &= \mathbf{TW}^{-1}(n) \times_3 \mathbf{S} = 2^{u_n} \times_3 \mathbf{TW}(n) \times_3 \mathbf{S} \\ &= \mathbf{TW}(n) \times_3 (2^{u_n} \times_3 \mathbf{S}). \end{aligned} \quad (8.12)$$

From (8.12), the vector $\mathbf{F}$ can be obtained by using forward instead of inverse ternary Walsh transform. Hence, we can use forward ternary Walsh transform to complete the functions of both forward and inverse ternary Walsh transforms and the inverse ternary Walsh transform can be ignored in the computations. According to the value of u_n , i.e., n is odd or even, we can modify $\mathbf{S}$ by keeping or replacing its elements 1 and 2 by 2 and 1, respectively, to use only the forward ternary Walsh transform for the calculation.

The Walsh transform is known as the most popular orthogonal transform. Due to the fact that the ternary Walsh transform is related to Walsh transform, ternary Walsh transform also exhibits the orthogonality.

Property 8.2.8. Let i -th row in the matrix of ternary Walsh transform $\mathbf{TW}(n)$ be represented by row vector $\mathbf{TW}_{i*}(n)$, and j -th column in the matrix $\mathbf{TW}(n)$ be represented by column vector $\mathbf{TW}_{*j}(n)$ where $0 \leq i < 2^n$ and $0 \leq j < 2^n$. The orthogonality of $\mathbf{TW}(n)$ are shown by the modulo-3 inner product of any two matrix rows $\mathbf{TW}_{i_1*}(n)$ and $\mathbf{TW}_{i_2*}(n)$ as follows:

$$\bigoplus_{k=0}^{2^n-1} {}_3 TW_{i_1,k} TW_{i_2,k} = \begin{cases} 0 & \text{if } i_1 \neq i_2, \\ 2^{u_n} & \text{if } i_1 = i_2. \end{cases} \quad (8.13)$$

Similarly, for the modulo-3 inner product of any two matrix columns $\mathbf{TW}_{*j_1}(n)$ and $\mathbf{TW}_{*j_2}(n)$, we have:

$$\bigoplus_{k=0}^{2^n-1} {}_3 TW_{k,j_1} TW_{k,j_2} = \begin{cases} 0 & \text{if } j_1 \neq j_2, \\ 2^{u_n} & \text{if } j_1 = j_2. \end{cases} \quad (8.14)$$

According to the relationships between forward and inverse ternary Walsh

transform shown in Property 8.2.2, the inverse ternary Walsh transform also has the same orthogonality for its matrix.

Property 8.2.9. From Property 8.2.7, the involution property of ternary Walsh transform can be proposed which means that the ternary Walsh spectrum of any ternary Walsh spectrum is the original ternary function with a normalizing factor 2^{u_n} , i.e.,

$$\mathbf{S}' = \mathbf{TW}(n) \times_3 \mathbf{S} = 2^{u_n} \times_3 \mathbf{F} \quad (8.15)$$

where $\mathbf{S}'$ is the ternary Walsh spectrum of $\mathbf{S}$ and $\mathbf{S}$ is the ternary Walsh spectrum of $\mathbf{F}$.

Property 8.2.10. Let the ternary function f^{12} be the ternary products of f_1 and f_2 , and their vectors be represented by $\mathbf{F}^{12}$, $\mathbf{F}_1$, and $\mathbf{F}_2$. Let $\mathbf{S}_1$, $\mathbf{S}_2$, and $\mathbf{S}^{12}$ represent the ternary Walsh spectrum of f_1 , f_2 , and f^{12} . The convolution of the ternary Walsh spectra can be expressed as follows:

$$s^{12}(i) = 2^{u_n} \times_3 \bigoplus_3^{2^n-1} (s_1(i \oplus k) \times_3 s_2(k)) = 2^{u_n} \times_3 \bigoplus_3^{2^n-1} s_1(k) \times_3 s_2(i \oplus k). \quad (8.16)$$

Let the symbol $\otimes$ represent the convolution of the ternary Walsh spectra. Hence, we have: $\mathbf{S}^{12} = \mathbf{S}_1 \otimes \mathbf{S}_2 = \mathbf{S}_2 \otimes \mathbf{S}_1$.

According to the relationships between forward and inverse ternary Walsh transform shown in Property 8.2.2, the convolution of the ternary Walsh transform shown in Property 8.2.10 can also be used for inverse ternary Walsh transform.

Figure 8.2: Diagram of the ternary Walsh transform where $n = 3$.

8.3 Fast Algorithms

Similar to the matrix factorization of fast Walsh transform, the ternary Walsh transform can also be factorized while the Kronecker product of the matrices and the operations are over $\text{GF}(3)$.

Property 8.3.1. The forward or inverse ternary Walsh transform matrix of order 2^n can be factorized to the products of n matrices as follows:

$$\mathbf{TW}(n) = \prod_{i=1}^n \mathbf{TC}_i(n), \quad (8.17)$$

$$\text{and } \mathbf{TW}^{-1}(n) = \prod_{i=1}^n \mathbf{TD}_i(n) \quad (8.18)$$

Table 8.5: Rules of Ternary Coding Converter

Input	Output of Ternary Coding Converter	
	Converting Enable=0	Converting Enable=1
0	0	0
1	1	2
2	2	1

$$\text{where } \mathbf{TC}_i(n) = \bigotimes_{j=1}^n \mathbf{TC}_i^j(1) \text{ and } \mathbf{TC}_i^j(1) = \begin{cases} \mathbf{I}_2, & \text{for } j \neq i, \\ \mathbf{TW}(1), & \text{for } j = i, \end{cases}$$

$$\mathbf{TD}_i(n) = \bigotimes_{j=1}^n \mathbf{TD}_i^j(1) \text{ and } \mathbf{TD}_i^j(1) = \begin{cases} \mathbf{I}_2, & \text{for } j \neq i, \\ \mathbf{TW}^{-1}(1), & \text{for } j = i. \end{cases}$$

The computational costs when using the fast algorithms will be reduced from 2^{2n} to $n2^n$ compared with when using the whole matrix. Fig. 8.1(a) and Fig. 8.1(b) have shown the corresponding fast diagrams for Example 8.2.1.

From Property 8.2.7, the fast Walsh transform structure with only ternary operations $\oplus_3$ and $\ominus_3$ can be used in the design of butterfly structure, while the other ternary operations from Table 8.2 can be ignored by adding a ternary coding converter between the input and the butterfly which can change the input values 1 and 2 to 2 and 1 or bypass all the input values. The diagram of this structure for $n = 3$ is shown in Fig. 8.2 as an example, where the value of Converting Enable decide whether to apply the ternary coding conversion or not for the input. Table 8.5 shows the rules for the ternary coding converter where the Converting Enable is set to 1 only when using it for the inverse transform and when n is odd.

While the fast diagrams for the ternary Walsh transform are based on the

(a) Forward fixed fast ternary Walsh transform (b) Inverse fixed fast ternary Walsh transform

Figure 8.3: Butterfly diagrams of the forward and inverse fixed fast ternary Walsh transforms where $n = 3$.

well known fast in-place Walsh transform, we will introduce a fixed fast ternary Walsh transform which can be implemented by iterative hardware structure. The matrix factorization of fixed fast ternary Walsh transform are shown in the following properties. It is noted that all the elements of the matrix and operations are over $\text{GF}(3)$.

Property 8.3.2. The forward ternary Walsh transform can be factorized using fast fixed ternary Walsh transform as:

$$\mathbf{TW}(n) = \prod_{i=1}^n \mathbf{TP}(n) \quad (8.19)$$

$$\text{where } \mathbf{TP}(n) = \begin{bmatrix} \mathbf{I}_{2^{n-1}} \otimes \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \end{bmatrix}.$$

Property 8.3.3. The inverse ternary Walsh transform can be factorized using

Figure 8.4: Iterative structure of the fixed ternary Walsh transform where $n = 3$.

fixed fast ternary Walsh transform as:

$$\mathbf{TW}^{-1}(n) = \prod_{i=1}^n \mathbf{TQ}(n) \quad (8.20)$$

$$\text{where } \mathbf{TQ}(n) = \left[\begin{array}{cc} \mathbf{I}_{2^{n-1}} \otimes \begin{bmatrix} 2 \\ 2 \end{bmatrix} & \mathbf{I}_{2^{n-1}} \otimes \begin{bmatrix} 2 \\ 1 \end{bmatrix} \end{array} \right].$$

Since the factorized matrices of fixed fast ternary Walsh transform in the same order are the same, the corresponding fast butterfly diagrams have the same structure in every stage. The fast butterfly diagrams of forward and inverse fixed fast ternary Walsh transform are shown in Fig. 8.3(a) and Fig. 8.3(b), respectively.

As the interconnection pattern is the same for all computational stages of the fixed fast ternary Walsh transform of order 2^n , the hardware can be reduced

to only one stage with the output is fed back to the input $n - 1$ times. The block diagram of such an iterative structure is shown in Fig. 8.4 which can be used for both forward and inverse ternary Walsh transform of order 8. The functional block using iterative architecture for fixed fast ternary Walsh transform consists of several parts that include Ternary Coding Converter, Butterfly Structure, Ternary Adders, Ternary Subtractors, Output Register. The Converting Enable in the block diagram controls the Ternary Coding Converter to be used as a simple input register or converting the input data as shown in Table 8.5. The input data are calculated and fed back to Ternary Coding Converter until the calculations have been processed $n - 1$ times.

8.4 Properties of Ternary Walsh Spectrum for Boolean Functions

The most popular spectral transform used for Boolean functions is Reed-Muller transform over $\text{GF}(2)$. While the Reed-Muller transform provides useful properties for the completely specified Boolean functions in many applications, it will lose information when handling the incompletely specified Boolean functions due to the limitations of operating over $\text{GF}(2)$. The novel ternary Walsh transform can save all the information for both completely and incompletely specified Boolean functions by using the proper coding as it operates over $\text{GF}(3)$. In this section, the basic properties of ternary Walsh spectrum for Boolean functions are proposed which are useful for providing various insights into the functions.

The following symbols are used: Let $\mathbf{x}_n = \{x_n, x_n, \dots, x_p, \dots, x_2, x_1\}$ be n -tuples over $\text{GF}(2)$ where $1 \leq p \leq n$. The symbol x_p stands for a data variable

Table 8.6: Logic Operations in T Coding

Operations		Result		
		0	1	2
$\neg$ (Negation)		1	0	2
$\wedge$ (AND)	0	0	0	0
	1	0	1	2
	2	0	2	2
$\vee$ (OR)	0	0	1	2
	1	1	1	1
	2	2	1	2
$\oplus$ (EXOR)	0	0	1	2
	1	1	0	2
	2	2	2	2

and $f(\mathbf{x}_n)$ stands for a Boolean function. Let the index i of the ternary Walsh coefficient $s(i)$ be represented in binary format as $(i_n i_{n-1} \cdots i_p \cdots i_2 i_1)_2$.

Several types of coding have been developed for the incompletely specified Boolean functions, including S coding and R coding. In S coding, the true minterms are denoted by -1 , false minterms by 1 , and don't care (DC) minterms by 0 . In R coding, the true minterms are denoted by 1 , false minterms by 0 , and DC minterms by 0.5 . However, the DCs are just given a specified value in present coding schemes. All the present coding schemes cannot process the DCs as true or false minterms by the logical expressions.

Definition 8.4.1. For the representation of the incompletely specified Boolean functions, a new type of coding named as T coding for the truth vector are proposed where the true minterms are denoted by 1 , the false minterms are denoted by 0 , and the DC minterms are denoted by 2 . The definitions of various logic operations for logic functions in T coding are shown in Table 8.6.

Property 8.4.1. The expressions of logic operations, i.e., Negation, AND, OR, and EXOR, have been given in standard binary format for the completely spec-

ified Boolean functions. The expressions of the logic operations in T coding for the completely specified Boolean functions can be also derived by converting the operations of known expressions in binary coding to the operations over GF(3) as follows:

Binary Coding T Coding

$$\bar{a} = 1 - a, \Rightarrow \bar{a} = 1 \ominus_3 a = 1 \oplus_3 2 \times_3 a;$$

$$a \wedge b = ab, \Rightarrow a \wedge b = a \times_3 b;$$

$$a \vee b = a + b - ab, \Rightarrow a \vee b = a \oplus_3 b \ominus_3 a \times_3 b = a \oplus_3 b \oplus_3 2 \times_3 a \times_3 b;$$

$$a \oplus b = a + b - 2ab, \Rightarrow a \oplus b = a \oplus_3 b \ominus_3 2 \times_3 a \times_3 b = a \oplus_3 b \oplus_3 a \times_3 b.$$

While the above expressions are applicable for the completely specified Boolean functions, the DCs cannot be processed in the standard binary coding. The proposed T coding is designed to solve this problem.

Property 8.4.2. The expressions of the logic operations in T coding for the incompletely specified Boolean functions are shown as follows:

$$\bar{a} = 1 \ominus_3 a = 1 \oplus_3 2 \times_3 a; \tag{8.21}$$

$$\begin{aligned} a \wedge b &= a \times_3 b \times_3 (1 \oplus_3 \bar{a} \times_3 \bar{b}) \\ &= a \times_3 b \times_3 (2 \oplus_3 2 \times_3 a \oplus_3 2 \times_3 b \oplus_3 a \times_3 b); \end{aligned} \tag{8.22}$$

$$\begin{aligned} a \vee b &= a \oplus_3 b \oplus_3 2 \times_3 (a \wedge b) \\ &= a \oplus_3 b \oplus_3 2 \times_3 a \times_3 b \times_3 (1 \oplus_3 \bar{a} \times_3 \bar{b}); \end{aligned} \tag{8.23}$$

$$a \oplus b = a \oplus_3 b \oplus_3 a \times_3 b. \tag{8.24}$$

The above expressions can be used for the logic operations in T coding for both

Table 8.7: Corresponding Relations between Ternary Walsh Coefficients and Variables for $n = 3$

Coefficient	i_3	i_2	i_1	Corresponding Variables
s_0	0	0	0	Constant
s_1	0	0	1	x_1
s_2	0	1	0	x_2
s_3	0	1	1	x_1, x_2
s_4	1	0	0	x_3
s_5	1	0	1	x_1, x_3
s_6	1	1	0	x_2, x_3
s_7	1	1	1	x_1, x_2, x_3

the completely and incompletely specified Boolean functions, and all the results in Table 8.6 can also be derived from (8.21)-(8.24). It should be noted that the expressions of negation and EXOR in T coding shown in Property 8.4.1 and Property 8.4.2 are the same.

The ternary Walsh transform is developed according to the Hadamard order. Therefore, the ternary Walsh spectrum $\mathbf{S}$ of the function $f(\mathbf{x}_n)$ can be interpreted as follows:

Every ternary Walsh coefficient $s(i)$ is corresponding to a number of input variables which satisfy the condition: The variable x_p is included if the $i_p = 1$ in the binary formatted index of $s(i)$, and $s(0)$ corresponds to the constant.

The corresponding relations between the ternary Walsh coefficients and the variables for $n = 3$ are given in Table 8.7 as an example.

More generally, the ternary Walsh transform matrix can be interpreted by the expressions over GF(3) for every row of the matrix in the following rule. The i -th row of the ternary Walsh transform matrix can be constructed by the product $\prod_{p \in P_i} (1 + x_p)$ over GF(3) where p is in set P_i if $i_p = 1$.

As an example, the interpretation of the ternary Walsh transform matrix of $n = 3$ is shown below:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 & 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 & 1 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 \\ 1 & 2 & 1 & 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 1 & 2 & 2 & 1 & 2 & 1 & 1 & 2 \end{bmatrix} \begin{matrix} 1 \\ 1 \oplus_3 x_1 \\ 1 \oplus_3 x_2 \\ (1 \oplus_3 x_1) \times_3 (1 \oplus_3 x_2) \\ 1 \oplus_3 x_3 \\ (1 \oplus_3 x_1) \times_3 (1 \oplus_3 x_3) \\ (1 \oplus_3 x_2) \times_3 (1 \oplus_3 x_3) \\ (1 \oplus_3 x_1) \times_3 (1 \oplus_3 x_2) \times_3 (1 \oplus_3 x_3) \end{matrix}$$

Property 8.4.3. According to the representations of the function and the fast algorithms of ternary Walsh transform, the derivation of the ternary Walsh coefficient can be written as:

$$s(i) = \bigoplus_{|\mathbf{x}_n|=0}^{2^n-1} (2^{<i, |\mathbf{x}_n|>} \times_3 f(\mathbf{x}_n)) \quad (8.25)$$

$$= \bigoplus_{x_1=0}^1 (2^{i_1 x_1} \times_3 \bigoplus_{x_2=0}^1 (2^{i_2 x_2} \times_3 \dots \bigoplus_{x_{n-1}=0}^1 (2^{i_{n-1} x_{n-1}} \times_3 \bigoplus_{x_n=0}^1 (2^{i_n x_n} \times_3 f(\mathbf{x}_n))) \dots)). \quad (8.26)$$

Property 8.4.4. The interpretation of the ternary Walsh transform matrix has been shown which can also be used for the inverse ternary Walsh transform. Hence, the logical interpretation of the ternary Walsh spectrum can be shown as follows:

$$f(\mathbf{x}_n) = 2^{u_n} \times_3 \left(s(0) \oplus_3 \left(\bigoplus_{i=1}^{2^n-1} \left(s(i) \times_3 \prod_{p \in P_i} (1 \oplus_3 x_p) \right) \right) \right). \quad (8.27)$$

The logical interpretation of the ternary Walsh spectrum is named as the ternary Walsh expansion. The incompletely specified Boolean functions cannot be expressed by standard logic equations in binary coding, while the ternary Walsh expansion which is the logic expressions over $\text{GF}(3)$ can be used to represent both the completely and incompletely specified Boolean function by using T coding.

Example 8.4.1. The truth vector of a 2-variable Boolean function is represented in T coding. The truth vector and its ternary Walsh spectrum are shown as follows:

x_2	x_1	Vector	Ternary Walsh Spectrum
0	0	2	0
0	1	0	1
1	0	0	1
1	1	1	0

According to (8.27), the corresponding ternary Walsh expansion is:

$$f(\mathbf{x}_2) = 2^0 \times_3 ((1 \oplus_3 x_1) \times_3 1 \oplus_3 (1 \oplus_3 x_2) \times_3 1) = 2 \oplus_3 x_1 \oplus_3 x_2.$$

The ternary Walsh spectrum of several special cases of logic functions will be considered in the following properties.

Property 8.4.5. When the function is the constant, its corresponding ternary

Walsh spectrum has the following formations:

$$\text{If } f(\mathbf{x}_n) = 0, \quad s(i) = 0 \text{ for } 0 \leq i < 2^n. \quad (8.28)$$

$$\text{If } f(\mathbf{x}_n) = 1, \quad s(i) = \begin{cases} 2^{u_n} & \text{if } i = 0, \\ 0 & \text{if } i \neq 0. \end{cases} \quad (8.29)$$

$$\text{If } f(\mathbf{x}_n) = 2, \quad s(i) = \begin{cases} 2^{u_n \oplus 1} & \text{if } i = 0, \\ 0 & \text{if } i \neq 0, \end{cases} \quad (8.30)$$

where $\oplus$ represents the EXOR. The equations (8.29) and (8.30) can also be written as:

$$\text{If } f(\mathbf{x}_n) = 2^q, \quad s(i) = \begin{cases} 2^{q \oplus u_n} & \text{if } i = 0, \\ 0 & \text{if } i \neq 0. \end{cases} \quad (8.31)$$

where $q = 0$ or 1 .

Property 8.4.6. When the function depends only on a single variable, its corresponding ternary Walsh spectrum has the following formations:

$$\text{If } f(\mathbf{x}_n) = 2^q \dot{x}_p, \quad s(i) = \begin{cases} 2^{u_n \oplus q \oplus 1} & \text{if } i = 0, \\ 2^{u_n \oplus v \oplus q} & \text{if } i = 2^p, \\ 0 & \text{otherwise,} \end{cases} \quad (8.32)$$

$$\text{where } \dot{x}_p = \begin{cases} x_p & \text{if } v = 0, \\ \bar{x}_p & \text{if } v = 1, \end{cases} \quad \text{and } q, v \in \{0, 1\}.$$

Property 8.4.7. When the function is the exponent function of a single vari-

Table 8.8: Ternary Walsh Spectra of Logic Gates

Variable		AND ($x_1 \wedge x_2$)		OR ($x_1 \vee x_2$)		EXOR ($x_1 \oplus x_2$)	
x_2	x_1	Function	Spectrum	Function	Spectrum	Function	Spectrum
0	0	0	1	0	0	0	2
0	1	0	2	1	2	1	0
1	0	0	2	1	2	1	0
1	1	1	1	1	2	0	1

able, its corresponding ternary Walsh spectrum has the following formations:

$$\text{If } f(\mathbf{x}_n) = 2^{\dot{x}_p}, \quad s(i) = \begin{cases} 2^{u_n \oplus v \oplus 1} & \text{if } i = 2^p, \\ 0 & \text{otherwise.} \end{cases} \quad (8.33)$$

Property 8.4.8. The two-variable logic gates that include AND, OR, and EXOR gates are most useful logic parts. The ternary Walsh spectra of these gates are shown in Table 8.8.

The study of the permutations and negations of the input variables in Boolean function are important issues in both classification and symmetry detections of Boolean functions. Hence, the relationships between the ternary Walsh spectrum and the mentioned operations are also interesting.

Property 8.4.9. Let the ternary Walsh spectrum of a logic function $f(\mathbf{x}_n)$ be $\mathbf{S}$ and the ternary Walsh spectrum of the logic function with interchange of two variables x_p and x_m from $f(\mathbf{x}_n)$ be $\mathbf{S}'$. The permutation of the ternary Walsh spectra $\mathbf{S}$ and $\mathbf{S}'$ can be performed by interchanging the 2^{n-2} pairs of ternary Walsh coefficients as follows:

$$s(i) \Leftrightarrow s'(j)$$

$$\text{where } j = (i_n i_{n-1} \cdots i_{p+1} 0 \ i_{p-1} \cdots i_{m+1} 1 \ i_{m-1} \cdots i_2 i_1)_2 \text{ if } i_p = 1 \text{ and } i_m = 0.$$

Property 8.4.10. Let the ternary Walsh spectrum of a logic function $f(\mathbf{x}_n)$ be $\mathbf{S}$ and the ternary Walsh spectrum of the logic function with the negation of variable x_p from $f(\mathbf{x}_n)$ be $\mathbf{S}''$. The permutation of the ternary Walsh spectra $\mathbf{S}$ and $\mathbf{S}''$ can be performed by interchanging the 2^{n-1} pairs of ternary Walsh coefficients as follows:

$$s(i) = 2 \times_3 s''(i), \text{ also } s''(i) = 2 \times_3 s(i), \quad \text{if } i_p = 1.$$

The relationships between the Walsh spectra for the composite Boolean functions were shown in [Eri78]. The relationships in the spectral domain for the composite Boolean functions are theoretically interesting and practically useful, such as the convolution theorems of standard Walsh transform and discrete Fourier transform [Kar76]. However, the previous study of the relationships in the spectral domain for the logic operations in the functional domain mainly concentrates on the completely specified Boolean functions. The evaluations of the ternary Walsh spectra of the composite Boolean functions from the ternary Walsh spectra of the elementary Boolean functions will be shown.

Property 8.4.11. Let $\bar{\mathbf{F}}$ and $\bar{\mathbf{S}}$ be the truth vector and ternary Walsh spectrum of function $\overline{f(\mathbf{x}_n)}$ which is the negation of Boolean function $f(\mathbf{x}_n)$. According to the expressions of negation in T coding shown in Property 8.4.1 and 8.4.2, $\bar{\mathbf{S}}$ can be generated by:

$$\bar{s}(i) = \begin{cases} 2^{u_n} \oplus_3 2 \times_3 s(i) & \text{if } i = 0, \\ 2 \times_3 s(i) & \text{otherwise,} \end{cases} \quad (8.34)$$

$$\text{i.e., } \bar{\mathbf{S}} = 2^{u_n} \times_3 \mathbf{K}_{2^n} \oplus_3 2 \times_3 \mathbf{S}, \quad (8.35)$$

where $\mathbf{K}_{2^n}$ is an $2^n \times 1$ matrix and $\mathbf{K}_{2^n} = [1 \ 0 \ 0 \ \cdots \ 0 \ 0]^T$.

Property 8.4.12. Let $\mathbf{S}^{a \oplus b}$ be the ternary Walsh spectrum of function $f^{a \oplus b}(\mathbf{x}_n)$ which is the EXOR sum of two Boolean functions $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$. According to the expressions of EXOR in T coding shown in Property 8.4.1 and 8.4.2, $\mathbf{S}^{a \oplus b}$ can be generated by:

$$\mathbf{S}^{a \oplus b} = \mathbf{S}_a \oplus_3 \mathbf{S}_b \oplus_3 (\mathbf{S}_a \otimes \mathbf{S}_b). \quad (8.36)$$

Property 8.4.13. Let $\mathbf{S}^{a \wedge b}$ be the ternary Walsh spectrum of function $f^{a \wedge b}(\mathbf{x}_n)$ which is the Boolean product of two logic functions $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$. If $f^{a \wedge b}(\mathbf{x}_n)$ is completely specified Boolean function, both $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$ are required to be completely specified Boolean functions. According to the expressions of AND in T coding shown in Property 8.4.1, $\mathbf{S}^{a \wedge b}$ of completely specified Boolean function $f^{a \wedge b}(\mathbf{x}_n)$ can be generated as follows:

$$\mathbf{S}^{a \wedge b} = \mathbf{S}_a \otimes \mathbf{S}_b. \quad (8.37)$$

When at least one of the functions $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$ is incompletely specified Boolean function, $f^{a \wedge b}(\mathbf{x}_n)$ is incompletely specified Boolean function. According to the expressions of AND in T coding shown in (8.22), $\mathbf{S}^{a \wedge b}$ can be generated by:

$$\mathbf{S}^{a \wedge b} = \mathbf{S}_a \otimes \mathbf{S}_b \otimes (2^{u_n} \times_3 \mathbf{K}_{2^n} \oplus_3 (\bar{\mathbf{S}}_a \otimes \bar{\mathbf{S}}_b)) \quad (8.38)$$

$$= \mathbf{S}_a \otimes \mathbf{S}_b \otimes (2^{u_n \oplus 1} \times_3 \mathbf{K}_{2^n} \oplus_3 2 \times_3 \mathbf{S}_a \oplus_3 2 \times_3 \mathbf{S}_b \oplus_3 (\mathbf{S}_a \otimes \mathbf{S}_b)) \quad (8.39)$$

The computations for completely specified Boolean functions can also use (8.38) and (8.39).

Property 8.4.14. Let $\mathbf{S}^{a\vee b}$ be the ternary Walsh spectrum of function $f^{a\vee b}(\mathbf{x}_n)$ which is the Boolean sum of two logic functions $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$. If $f^{a\vee b}(\mathbf{x}_n)$ is completely specified Boolean function, both $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$ are required to be completely specified Boolean functions. According to the expressions of OR in T coding shown in Property 8.4.1, $\mathbf{S}^{a\vee b}$ of completely specified Boolean function $f^{a\vee b}(\mathbf{x}_n)$ can be generated as follows:

$$\mathbf{S}^{a\vee b} = \mathbf{S}_a \oplus_3 \mathbf{S}_b \oplus_3 2 \times_3 (\mathbf{S}_a \otimes \mathbf{S}_b). \quad (8.40)$$

When at least one of the functions $f_a(\mathbf{x}_n)$ and $f_b(\mathbf{x}_n)$ is incompletely specified Boolean function, $f^{a\vee b}(\mathbf{x}_n)$ is incompletely specified Boolean function. According to the expressions of OR in T coding shown in (8.23), $\mathbf{S}^{a\vee b}$ can be generated by:

$$\mathbf{S}^{a\vee b} = \mathbf{S}_a \oplus_3 \mathbf{S}_b \oplus_3 2 \times_3 \mathbf{S}^{a\wedge b} \quad (8.41)$$

$$= \mathbf{S}_a \oplus_3 \mathbf{S}_b \oplus_3 2 \times_3 (\mathbf{S}_a \otimes \mathbf{S}_b \otimes (2^{u_n} \times_3 \mathbf{K}_{2^n} \oplus_3 (\bar{\mathbf{S}}_a \otimes \bar{\mathbf{S}}_b))). \quad (8.42)$$

The computations for completely specified Boolean functions can also use (8.41) and (8.42).

Properties 8.4.11-8.4.14 reveal the corresponding relationships in ternary Walsh domain for the logic operations in the functional domain. Both the completely and incompletely specified Boolean functions can be handled by ternary Walsh transform. Therefore, the ternary Walsh transform shows the advantage over other known spectral transforms such as standard Walsh transform and Reed-Muller transform.

8.5 The Decomposition of Boolean Functions

Most methods used for the decomposition of Boolean functions are only for the completely specified Boolean functions only. Therefore all possible cases must be counted separately for the incompletely specified Boolean functions in the decomposition of Boolean functions when using these methods. The ternary Walsh transform has been shown in Section 8.4 that it can handle the incompletely specified Boolean functions as well as the completely specified Boolean functions where the T coding is used to encode the logic functions.

Definition 8.5.1. The Shannon expansion is widely used as canonical disjoint form for the Boolean functions. The n -variable Boolean function $f(\mathbf{x}_n)$ can be decomposed to two $(n-1)$ -variable Boolean functions as shown in (2.6).

Every variable of $f(\mathbf{x}_n)$ can be used as the variable x_n in (2.6) by interchanging the variables. The function $f(\mathbf{x}_n)$ can also be decomposed to 2^m $(n-m)$ -variable functions by applying (2.6) m times as shown in (2.7).

The Shannon expansion can be represented by applying T coding in (2.6) as:

$$\begin{aligned} f(\mathbf{x}_n) &= \bar{x}_n \times_3 f(0, x_{n-1}, x_{n-2}, \dots, x_2, x_1) \oplus_3 x_n \times_3 \\ &\quad f(1, x_{n-1}, x_{n-2}, \dots, x_2, x_1). \end{aligned} \quad (8.43)$$

Similarly, we can also rewrite (2.7) by applying T coding in it as:

$$\begin{aligned} f(\mathbf{x}_n) &= \prod_{i=n-m+1}^n \bar{x}_i \times_3 f(0, 0, \dots, 0, x_{n-m}, \dots, x_1) \oplus_3 \dots \\ &\quad \oplus_3 \prod_{i=n-m+1}^n x_i \times_3 f(1, 1, \dots, 1, x_{n-m}, \dots, x_1). \end{aligned} \quad (8.44)$$

Property 8.5.1. Let $\mathbf{F}_0^1$ and $\mathbf{F}_1^1$ represent the vectors of $f(0, x_{n-1}, \dots, x_1)$ and $f(1, x_{n-1}, \dots, x_1)$ and their ternary Walsh spectra are $\mathbf{S}_0^1$ and $\mathbf{S}_1^1$, respectively. From (8.26) and (8.43), the ternary Walsh coefficients of $\mathbf{F}$ can be derived from the ternary Walsh coefficients of $\mathbf{F}_0^1$ and $\mathbf{F}_1^1$ as follows:

$$\begin{aligned}
 s(i) &= \bigoplus_{x_1=0}^1 (2^{i_1 x_1} \times_3 \dots \bigoplus_{x_n=0}^1 (2^{i_n x_n} \times_3 (\bar{x}_n \times_3 f(0, x_{n-1}, \dots, x_1) \oplus_3 \\
 &\quad x_n \times_3 f(1, x_{n-1}, \dots, x_1))) \dots) \\
 &= \bigoplus_{x_1=0}^1 (2^{i_1 x_1} \times_3 \dots \bigoplus_{x_{n-1}=0}^1 (2^{i_{n-1} x_{n-1}} \times_3 (f(0, x_{n-1}, \dots, x_1) \oplus_3 \\
 &\quad 2^{i_n} \times_3 f(1, x_{n-1}, \dots, x_1))) \dots) \\
 &= s_0^1(i - i_n 2^{n-1}) \oplus_3 2^{i_n} \times_3 s_1^1(i - i_n 2^{n-1}).
 \end{aligned}$$

Hence, we have:

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_0^1 \oplus_3 \mathbf{S}_1^1 \\ \mathbf{S}_0^1 \oplus_3 2 \times_3 \mathbf{S}_1^1 \end{bmatrix} = \mathbf{TW}(1) \times_3 \begin{bmatrix} \mathbf{S}_0^1 \\ \mathbf{S}_1^1 \end{bmatrix}. \quad (8.45)$$

Let $\mathbf{F}_j^m$ represent the vector of function $f(j_m, j_{m-1}, \dots, j_2, j_1, x_{n-m}, \dots, x_1)$ and its ternary Walsh spectrum is $\mathbf{S}_j^m$ where $j = (j_m j_{m-1} \dots j_2 j_1)_2$. Let the ternary Walsh spectrum $\mathbf{S}$ be divided to 2^m parts and the j -th part of $\mathbf{S}$ be represented as $\hat{\mathbf{S}}_j^m$ where $0 \leq j < 2^m$. The representation of the Shannon expansion of n -variable function $f(\mathbf{x}_n)$ in ternary Walsh domain is:

$$\mathbf{S} = \begin{bmatrix} \hat{\mathbf{S}}_0^m \\ \vdots \\ \hat{\mathbf{S}}_j^m \\ \vdots \\ \hat{\mathbf{S}}_{2^m-1}^m \end{bmatrix} = \mathbf{TW}(m) \times_3 \begin{bmatrix} \mathbf{S}_0^m \\ \vdots \\ \mathbf{S}_j^m \\ \vdots \\ \mathbf{S}_{2^m-1}^m \end{bmatrix}. \quad (8.46)$$

Therefore, the ternary Walsh spectra of the decomposed functions can be calculated from $\mathbf{S}$ as follows:

$$\begin{bmatrix} \mathbf{S}_0^m \\ \vdots \\ \mathbf{S}_j^m \\ \vdots \\ \mathbf{S}_{2^m-1}^m \end{bmatrix} = 2^{u_m} \times_3 \mathbf{TW}(m) \times_3 \begin{bmatrix} \hat{\mathbf{S}}_0^m \\ \vdots \\ \hat{\mathbf{S}}_j^m \\ \vdots \\ \hat{\mathbf{S}}_{2^m-1}^m \end{bmatrix}. \quad (8.47)$$

From (8.47), the decomposed functions by using Shannon expansion can be derived from the manipulation of ternary Walsh coefficients of the original function where the method is suitable for both completely and incompletely specified Boolean functions in T coding.

Definition 8.5.2. The positive and negative Davio expansions are also well known decomposition types of the Boolean functions. The n -variable Boolean function $f(\mathbf{x}_n)$ can be composed by two $(n-1)$ -variable Boolean functions in positive and negative Davio expansions as follows:

$$f(\mathbf{x}_n) = f(0, x_{n-1}, \dots, x_1) \oplus x_n f_D(\mathbf{x}_{n-1}), \quad (8.48)$$

$$\text{and } f(\mathbf{x}_n) = f(1, x_{n-1}, \dots, x_1) \oplus \bar{x}_n f_D(\mathbf{x}_{n-1}), \quad (8.49)$$

where $f_D(\mathbf{x}_{n-1}) = f(0, x_{n-1}, \dots, x_1) \oplus f(1, x_{n-1}, \dots, x_1)$.

The positive and negative Davio expansions are represented in T coding as follows:

$$f(\mathbf{x}_n) = f(0, x_{n-1}, \dots, x_1) \oplus_3 x_n \times_3 f_D(\mathbf{x}_{n-1}) \times_3 (1 \oplus_3 f(0, x_{n-1}, \dots, x_1)), \quad (8.50)$$

$$\text{and } f(\mathbf{x}_n) = f(1, x_{n-1}, \dots, x_1) \oplus_3 \bar{x}_n \times_3 f_D(\mathbf{x}_{n-1}) \times_3 (1 \oplus_3 f(1, x_{n-1}, \dots, x_1)). \quad (8.51)$$

Every variable of $f(\mathbf{x}_n)$ can be used as the variable x_n in the positive and negative Davio expansions by interchanging the variables.

Property 8.5.2. Let $\mathbf{F}_D$ represent the vector of $f_D(\mathbf{x}_{n-1})$ and its ternary Walsh spectrum is $\mathbf{S}_D$. According to the definition of $\mathbf{F}_D$, we have: $\mathbf{F}_D = \mathbf{F}_0^1 \oplus \mathbf{F}_1^1$, i.e., $\mathbf{F}_0^1 = \mathbf{F}_1^1 \oplus \mathbf{F}_D$, or $\mathbf{F}_1^1 = \mathbf{F}_0^1 \oplus \mathbf{F}_D$. The ternary Walsh spectrum of the original function can be obtained from the ternary Walsh spectra of its two expanded functions by positive Davio expansion as follows:

$$\mathbf{S} = \mathbf{TW}(1) \times_3 \begin{bmatrix} \mathbf{S}_0^1 \\ \mathbf{S}_0^1 \oplus_3 \mathbf{S}_D \oplus_3 (\mathbf{S}_0^1 \otimes \mathbf{S}_D) \end{bmatrix}. \quad (8.52)$$

Similarly, the ternary Walsh spectrum $\mathbf{S}$ can also be obtained from $\mathbf{S}_1^1$ and $\mathbf{S}_D$ which are the corresponding ternary Walsh spectra of the expanded functions by negative Davio expansion as follows:

$$\mathbf{S} = \mathbf{TW}(1) \times_3 \begin{bmatrix} \mathbf{S}_1^1 \oplus_3 \mathbf{S}_D \oplus_3 (\mathbf{S}_1^1 \otimes \mathbf{S}_D) \\ \mathbf{S}_1^1 \end{bmatrix}. \quad (8.53)$$

The relationships of the function and its positive and negative Davio expansions are revealed in ternary Walsh domain and are capable of handling both completely and incompletely specified Boolean functions.

8.6 Symmetry Detection of Boolean Functions

The symmetry detections of the Boolean functions play important roles in the logic synthesis and testing as its knowledge leads to a more proficient realization of the function. The spectral methods for the symmetry detections have been developed in [EH78, FK99, FK00, Hur77, RF98, TMS96]. But there is no efficient spectral methods which can detect the symmetries in both the completely and incompletely specified Boolean functions. In this section, the new spectral technique based on ternary Walsh spectra is applied to detect symmetries in both the completely and incompletely specified Boolean functions.

There are different types of Boolean symmetries between two variables including nonequivalent (NE) symmetry, equivalent (E) symmetry, skew-nonequivalent (SNE) symmetry, and skew-equivalent (SE) symmetry.

Definition 8.6.1. The n -variable Boolean function $f(\mathbf{x}_n)$, i.e., $f(x_n, \dots, x_i, \dots, x_j, \dots, x_1)$, is nonequivalent symmetric in x_i and x_j if the function is the same when the variables x_i and x_j are interchanged in $f(\mathbf{x}_n)$. The nonequivalent symmetric in x_i and x_j is denoted by $NE(x_i, x_j)$.

Definition 8.6.2. The n -variable Boolean function $f(\mathbf{x}_n)$ is equivalent symmetric in x_i and x_j if the function remains unchanged when the variables x_i is replaced by $\bar{x}_j$ and x_j is replaced by $\bar{x}_i$. The equivalent symmetric in x_i and x_j is denoted by $E(x_i, x_j)$.

Definition 8.6.3. The n -variable Boolean function $f(\mathbf{x}_n)$ is skew-nonequivalent symmetric in x_i and x_j if the function changes to its complement when the variables x_i and x_j of the opposite values are interchanged in $f(\mathbf{x}_n)$. The skew-nonequivalent symmetric in x_i and x_j is denoted by $SNE(x_i, x_j)$.

Definition 8.6.4. The n -variable Boolean function $f(\mathbf{x}_n)$ is skew-equivalent symmetric in x_i and x_j if the function changes to its complement when the variables x_i and x_j of the same values are interchanged in $f(\mathbf{x}_n)$. The skew-equivalent symmetric in x_i and x_j is denoted by $SE(x_i, x_j)$.

8.6.1 Symmetry Detection of Completely Specified Boolean Functions

Let the variables x_n and x_{n-1} be first considered for the symmetry detection while they are equivalent to any pair of variables in $f(\mathbf{x}_n)$. According to the Shannon expansion, we have:

$$\begin{aligned} f(\mathbf{x}_n) &= \bar{x}_n \times_3 \bar{x}_{n-1} \times_3 f(0, 0, x_{n-2}, \dots, x_1) \oplus_3 \bar{x}_n \times_3 x_{n-1} \times_3 f(0, 1, x_{n-2}, \dots, x_1) \\ &\quad \oplus_3 x_n \times_3 \bar{x}_{n-1} \times_3 f(1, 0, x_{n-2}, \dots, x_1) \\ &\quad \oplus_3 x_n \times_3 x_{n-1} \times_3 f(1, 1, x_{n-2}, \dots, x_1). \end{aligned} \tag{8.54}$$

Property 8.6.1. Based on the Definitions 8.6.1-8.6.4 and (8.54), the different types of Boolean symmetries for the completely specified Boolean function can

be represented as follows:

$$NE(x_n, x_{n-1}) \Leftrightarrow \mathbf{F}_1^2 = \mathbf{F}_2^2 ; \quad (8.55)$$

$$E(x_n, x_{n-1}) \Leftrightarrow \mathbf{F}_0^2 = \mathbf{F}_3^2 ; \quad (8.56)$$

$$SNE(x_n, x_{n-1}) \Leftrightarrow \mathbf{F}_1^2 = \overline{\mathbf{F}_2^2} ; \quad (8.57)$$

$$SE(x_n, x_{n-1}) \Leftrightarrow \mathbf{F}_0^2 = \overline{\mathbf{F}_3^2} . \quad (8.58)$$

By using (8.47), we have:

$$\begin{bmatrix} \mathbf{S}_0^2 \\ \mathbf{S}_1^2 \\ \mathbf{S}_2^2 \\ \mathbf{S}_3^2 \end{bmatrix} = \mathbf{TW}(2) \times_3 \begin{bmatrix} \hat{\mathbf{S}}_0^2 \\ \hat{\mathbf{S}}_1^2 \\ \hat{\mathbf{S}}_2^2 \\ \hat{\mathbf{S}}_3^2 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{S}}_0^2 \oplus_3 \hat{\mathbf{S}}_1^2 \oplus_3 \hat{\mathbf{S}}_2^2 \oplus_3 \hat{\mathbf{S}}_3^2 \\ \hat{\mathbf{S}}_0^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_1^2 \oplus_3 \hat{\mathbf{S}}_2^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_3^2 \\ \hat{\mathbf{S}}_0^2 \oplus_3 \hat{\mathbf{S}}_1^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_2^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_3^2 \\ \hat{\mathbf{S}}_0^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_1^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_2^2 \oplus_3 \hat{\mathbf{S}}_3^2 \end{bmatrix} . \quad (8.59)$$

Property 8.6.2. From (8.55)-(8.59), the relationships between ternary Walsh spectra and different types of Boolean symmetries for the completely specified Boolean function can be obtained as follows:

$$NE(x_n, x_{n-1}) \Leftrightarrow \hat{\mathbf{S}}_1^2 = \hat{\mathbf{S}}_2^2 \quad \text{or} \quad \hat{\mathbf{S}}_1^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_2^2 = \mathbf{0} ; \quad (8.60)$$

$$E(x_n, x_{n-1}) \Leftrightarrow \hat{\mathbf{S}}_1^2 \oplus_3 \hat{\mathbf{S}}_2^2 = \mathbf{0} ; \quad (8.61)$$

$$SNE(x_n, x_{n-1}) \Leftrightarrow \hat{\mathbf{S}}_0^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_3^2 \oplus_3 2^{u_{n-2}} \mathbf{K}_{2^{n-2}} = \mathbf{0} ; \quad (8.62)$$

$$SE(x_n, x_{n-1}) \Leftrightarrow \hat{\mathbf{S}}_0^2 \oplus_3 \hat{\mathbf{S}}_3^2 \oplus_3 2^{u_{n-2}} \mathbf{K}_{2^{n-2}} = \mathbf{0} \quad (8.63)$$

where $\mathbf{0}$ is an $2^{n-2} \times 1$ matrix of all zeros.

Although the relationships given in Property 8.6.2 are based on the symmetry detections of variables x_n and x_{n-1} in $f(\mathbf{x}_n)$ and represented by means of

ternary Walsh spectra, we can derive the relationships of ternary Walsh coefficients and the Boolean symmetries in any pair of variables x_i and x_j in $f(\mathbf{x}_n)$ as well.

Property 8.6.3. For the different types of Boolean symmetries in the completely specified Boolean functions, the corresponding relationships of the ternary Walsh coefficients are shown as follows:

$$NE(x_i, x_j) \Leftrightarrow s(p + 2^{i-1}) = s(p + 2^{j-1}) \text{ or } s(p + 2^{i-1}) \oplus_3 2 \times_3 s(p + 2^{j-1}) = 0 ; \quad (8.64)$$

$$E(x_i, x_j) \Leftrightarrow s(p + 2^{i-1}) \oplus_3 s(p + 2^{j-1}) = 0 ; \quad (8.65)$$

$$SNE(x_i, x_j) \Leftrightarrow s(p) \oplus_3 2 \times_3 s(p + 2^{i-1} + 2^{j-1}) \oplus_3 2^{u_{n-m}} \times_3 K(q) = 0 ; \quad (8.66)$$

$$SE(x_i, x_j) \Leftrightarrow s(p) \oplus_3 s(p + 2^{i-1} + 2^{j-1}) \oplus_3 2^{u_{n-m}} \times_3 K(q) = 0 . \quad (8.67)$$

In (8.64)-(8.67), p is the inner product of two $(n-2) \times 1$ matrices $\mathbf{Q}$ and $\mathbf{V}$, i.e., $p = \mathbf{Q}\mathbf{V}$, where $\mathbf{Q} = [q_{n-2} \ q_{n-1} \cdots q_2 \ q_1]$, $\mathbf{V} = [2^{n-1} \ 2^{n-2} \cdots 2^i \ 2^{i-2} \cdots 2^j \ 2^{j-2} \cdots 2 \ 1]^T$, q is an integer in $[0, 2^{n-2})$ and $q = (q_{n-2} \ q_{n-1} \cdots q_2 \ q_1)_2$. Hence, there are 2^{n-2} cases for the value of p . It is noted that there are 2^{n-2} pairs of ternary Walsh coefficients required to check for each type of Boolean symmetries.

8.6.2 Symmetry Detection of Incompletely Specified Boolean Functions

As noted in Property 8.6.1, the Boolean symmetries for the completely specified Boolean functions are shown based on the equivalences of the corresponding pairs of function values. However, the same manner cannot be used for the incompletely specified Boolean functions as the symmetry exists if one of the value

Table 8.9: Rules of MEXOR Operation in T Coding

$\oplus_M$	0	1	2
0	0	1	0
1	1	0	0
2	0	0	0

in the pair is DC. Therefore, we need to define a new form for the representation of the symmetry for incompletely specified Boolean functions.

In T coding, we need a unified operation which can represent whether the symmetry exists in two values and this operation should be applicable for both completely and incompletely specified Boolean functions. It is known that the EXOR can be used as the operation to detect the symmetry depending on the EXOR sum of the two values in the incompletely specified Boolean function. The expression of EXOR operation in standard binary coding is known as:

$$a \oplus b = a\bar{b} + \bar{a}b \text{ and } a \oplus b = \begin{cases} 1 & \text{if } a \neq b, \\ 0 & \text{if } a = b. \end{cases}$$

The expected new operation in T coding should have similar property with EXOR operation: the result is 0 when the two values are equivalent or at least one of them is DC, otherwise the result is 1. Consequently, we can construct the rules of the new operation as shown in Table 8.9 and the new operation is named as Modified Exclusive-OR (MEXOR) and denoted by $\oplus_M$. The MEXOR operation is exactly the same with EXOR operation when there is no DC to be considered. The corresponding expression of EXOR operation in T coding can be written as:

$$a \oplus b = a \times_3 \bar{b} \oplus_3 \bar{a} \times_3 b. \quad (8.68)$$

Now we can present a comprehensive definition of the new operation MEXOR in T coding.

Definition 8.6.5. The MEXOR operation $\oplus_M$ is developed for detecting the symmetry of two values while its operation rules are shown in Table 8.9. The MEXOR operation can be expressed by the operations over GF(3) where the minterms are encoded in T coding as follows:

$$a \oplus_M b = (a \oplus_3 b) \times_3 (1 \oplus_3 a \oplus_3 b) \times_3 (a \times_3 b \oplus_3 2). \quad (8.69)$$

The Boolean symmetries in both completely and incompletely Boolean functions can be discussed based on the ternary Walsh spectra when MEXOR operations are used. Similar to the discussion for completely Boolean functions, let us consider the Boolean symmetries in variables x_n and x_{n-1} of $f(\mathbf{x}_n)$ first.

Property 8.6.4. According to the definitions of different Boolean symmetries shown in Definitions 8.6.1-8.6.4, the conditions of different symmetries can be derived by using the MEXOR operations in the corresponding pairs of function values encoded in T coding. Hence, the relationships between ternary Walsh spectra and different types of Boolean symmetries which are suitable for both the completely and incompletely specified Boolean functions can be obtained as follows:

$$\begin{aligned} NE(x_n, x_{n-1}) \Leftrightarrow \\ (2 \times_3 \hat{\mathbf{S}}_0^2 \oplus_3 \hat{\mathbf{S}}_3^2) \otimes (2^{u_{n-2}} \times_3 \mathbf{K}_{2^{n-2}} \oplus_3 2 \times_3 \hat{\mathbf{S}}_0^2 \oplus_3 \hat{\mathbf{S}}_3^2) \otimes ((\hat{\mathbf{S}}_0^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_1^2 \oplus_3 \hat{\mathbf{S}}_2^2 \oplus_3 \\ 2 \times_3 \hat{\mathbf{S}}_3^2) \otimes (\hat{\mathbf{S}}_0^2 \oplus_3 \hat{\mathbf{S}}_1^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_2^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_3^2) \oplus_3 2^{u_{n-2} \oplus 1} \times_3 \mathbf{K}_{2^{n-2}}) = \mathbf{0} ; \end{aligned} \quad (8.70)$$

$$\begin{aligned}
 E(x_n, x_{n-1}) &\Leftrightarrow \\
 (\hat{\mathbf{S}}_0^2 \oplus_3 \hat{\mathbf{S}}_3^2) \otimes (2^{u_{n-2} \oplus 1} \times_3 \mathbf{K}_{2^{n-2}} \oplus_3 \hat{\mathbf{S}}_0^2 \oplus_3 \hat{\mathbf{S}}_3^2) \otimes ((\hat{\mathbf{S}}_0^2 \oplus_3 \hat{\mathbf{S}}_1^2 \oplus_3 \hat{\mathbf{S}}_2^2 \oplus_3 \hat{\mathbf{S}}_3^2) \otimes \\
 (\hat{\mathbf{S}}_0^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_1^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_2^2 \oplus_3 \hat{\mathbf{S}}_3^2) \oplus_3 2^{u_{n-2} \oplus 1} \times_3 \mathbf{K}_{2^{n-2}}) &= \mathbf{0}; \quad (8.71)
 \end{aligned}$$

$$\begin{aligned}
 SNE(x_n, x_{n-1}) &\Leftrightarrow \\
 ((\hat{\mathbf{S}}_1^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_2^2) \otimes (\hat{\mathbf{S}}_1^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_2^2)) \oplus_3 2^{u_{n-2} \oplus 1} \times_3 \mathbf{K}_{2^{n-2}} \otimes ((\hat{\mathbf{S}}_0^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_1^2 \oplus_3 \\
 \hat{\mathbf{S}}_2^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_3^2) \otimes (2 \times_3 \hat{\mathbf{S}}_0^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_1^2 \oplus_3 \hat{\mathbf{S}}_2^2 \oplus_3 \hat{\mathbf{S}}_3^2 \oplus_3 2^{u_{n-2}} \times_3 \mathbf{K}_{2^{n-2}}) \\
 \oplus_3 2^{u_{n-2} \oplus 1} \times_3 \mathbf{K}_{2^{n-2}}) &= \mathbf{0}; \quad (8.72)
 \end{aligned}$$

$$\begin{aligned}
 SE(x_n, x_{n-1}) &\Leftrightarrow \\
 (\hat{\mathbf{S}}_1^2 \oplus_3 \hat{\mathbf{S}}_2^2) \otimes (\hat{\mathbf{S}}_1^2 \oplus_3 \hat{\mathbf{S}}_2^2) \oplus_3 2^{u_{n-2} \oplus 1} \times_3 \mathbf{K}_{2^{n-2}} \otimes ((\hat{\mathbf{S}}_0^2 \oplus_3 \hat{\mathbf{S}}_1^2 \oplus_3 \hat{\mathbf{S}}_2^2 \oplus_3 \hat{\mathbf{S}}_3^2) \otimes \\
 (2 \times_3 \hat{\mathbf{S}}_0^2 \oplus_3 \hat{\mathbf{S}}_1^2 \oplus_3 \hat{\mathbf{S}}_2^2 \oplus_3 2 \times_3 \hat{\mathbf{S}}_3^2 \oplus_3 2^{u_{n-2}} \times_3 \mathbf{K}_{2^{n-2}}) \oplus_3 2^{u_{n-2} \oplus 1} \times_3 \mathbf{K}_{2^{n-2}}) &= \mathbf{0}. \quad (8.73)
 \end{aligned}$$

Property 8.6.5. The corresponding relationships of ternary Walsh coefficients and the Boolean symmetries in any pair of variables x_i and x_j in $f(\mathbf{x}_n)$ are shown as follows:

$$\begin{aligned}
 NE(x_i, x_j) &\Leftrightarrow \\
 (2 \times_3 s(p) \oplus_3 s(p + 2^{i-1} + 2^{j-1})) \times_3 \bigoplus_{k=0}^{2^{n-2}-1} (2^{u_{n-2}} \times_3 K(q') \oplus_3 2 \times_3 s(p') \oplus_3 \\
 s(p' + 2^{i-1} + 2^{j-1})) \times_3 \bigoplus_{k'=0}^{2^{n-2}-1} ((s(r) \oplus_3 2 \times_3 s(r + 2^{j-1}) \oplus_3 s(r + 2^{i-1}) \oplus_3 \\
 2 \times_3 s(r + 2^{i-1} + 2^{j-1})) \times_3 \bigoplus_{h=0}^{2^{n-2}-1} (s(r') \oplus_3 s'(r + 2^{j-1}) \oplus_3 2 \times_3 s(r' + 2^{i-1}) \oplus_3 \\
 2 \times_3 s(r' + 2^{i-1} + 2^{j-1})) \oplus_3 2^{u_{n-2} \oplus 1} \times_3 K(t)) &= 0; \quad (8.74)
 \end{aligned}$$

$$\begin{aligned}
E(x_i, x_j) \Leftrightarrow & \\
& (s(p) \oplus_3 s(p+2^{i-1}+2^{j-1})) \times_3 \bigoplus_{k=0}^{2^{n-2}-1} (2^{u_{n-2} \oplus 1} \times_3 K(q') \oplus_3 s(p') \oplus_3 s(p'+2^{i-1}+2^{j-1})) \\
& \times_3 \bigoplus_{k'=0}^{2^{n-2}-1} ((s(r) \oplus_3 s(r+2^{j-1}) \oplus_3 s(r+2^{i-1}) \oplus_3 s(r+2^{i-1}+2^{j-1})) \\
& \times_3 \bigoplus_{h=0}^{2^{n-2}-1} (s(r') \oplus_3 2 \times_3 s'(r+2^{j-1}) \oplus_3 2 \times_3 s(r'+2^{i-1}) \oplus_3 s(r'+2^{i-1}+2^{j-1})) \oplus_3 \\
& 2^{u_{n-2} \oplus 1} \times_3 K(t)) = 0; \tag{8.75}
\end{aligned}$$

$$\begin{aligned}
SNE(x_i, x_j) \Leftrightarrow & \\
& ((2 \times_3 s(p+2^{i-1}) \oplus_3 s(p+2^{j-1})) \times_3 \bigoplus_{k=0}^{2^{n-2}-1} (2 \times_3 s(p'+2^{i-1}) \oplus_3 s(p'+2^{j-1})) \oplus_3 \\
& 2^{u_{n-2} \oplus 1} K(q)) \times_3 \bigoplus_{k'=0}^{2^{n-2}-1} ((s(r) \oplus_3 2 \times_3 s(r+2^{j-1}) \oplus_3 s(r+2^{i-1}) \oplus_3 \\
& 2 \times_3 s(r+2^{i-1}+2^{j-1})) \times_3 \bigoplus_{h=0}^{2^{n-2}-1} (2 \times_3 s(r') \oplus_3 2 \times_3 s'(r+2^{j-1}) \oplus_3 s(r'+2^{i-1}) \oplus_3 \\
& s(r'+2^{i-1}+2^{j-1}) \oplus_3 2^{u_{n-2}} \times_3 K(t')) \oplus_3 2^{u_{n-2} \oplus 1} \times_3 K(t)) = 0; \tag{8.76}
\end{aligned}$$

$$\begin{aligned}
SE(x_i, x_j) \Leftrightarrow & \\
& (s(p+2^{i-1}) \oplus_3 s(p+2^{j-1})) \times_3 \bigoplus_{k=0}^{2^{n-2}-1} (s(p'+2^{i-1}) \oplus_3 s(p'+2^{j-1})) \oplus_3 2^{u_{n-2} \oplus 1} K(q)) \\
& \times_3 \bigoplus_{k'=0}^{2^{n-2}-1} ((s(r) \oplus_3 s(r+2^{j-1}) \oplus_3 s(r+2^{i-1}) \oplus_3 s(r+2^{i-1}+2^{j-1})) \times_3 \\
& \bigoplus_{h=0}^{2^{n-2}-1} (2 \times_3 s(r') \oplus_3 s'(r+2^{j-1}) \oplus_3 s(r'+2^{i-1}) \oplus_3 2 \times_3 s(r'+2^{i-1}+2^{j-1}) \\
& \oplus_3 2^{u_{n-2}} \times_3 K(t')) \oplus_3 2^{u_{n-2} \oplus 1} \times_3 K(t)) = 0; \tag{8.77}
\end{aligned}$$

In (8.74)-(8.77), p' , r and r' are defined as: $p' = \mathbf{Q}'\mathbf{V}$, $r = \mathbf{T}\mathbf{V}$, and $r' =$

$\mathbf{T}'\mathbf{V}$, where $\mathbf{Q}' = [q'_{n-2} \ q'_{n-1} \cdots q'_2 \ q'_1]$, $\mathbf{R} = [r_{n-2} \ r_{n-1} \cdots r_2 \ r_1]$, $\mathbf{R}' = [r'_{n-2} \ r'_{n-1} \cdots r'_2 \ r'_1]$, q', r and r' are integers in $[0, 2^{n-2})$ and $q' = q \oplus k$, $t = q' \oplus k'$, $t' = t \oplus h$, p and q are the same as the ones defined in Property 8.6.3.

The relationships in ternary Walsh domain which are corresponding to the different types of Boolean symmetries in functional domain have been analyzed. The study of the properties of ternary Walsh spectrum for the symmetries are discussed in Section 8.6.1 for completely specified Boolean functions, while an more comprehensive discussion for both completely and incompletely specified Boolean functions are given in Section 8.6.2. The ternary Walsh transform shows its capability of handling DCs of Boolean functions in the symmetry detections based on T coding.

8.7 Ternary Walsh Spectral Transform Decision Diagrams

Spectral transform decision diagrams (STDDs) are used to represent the spectra of the given logic functions. The STDDs for Reed-Muller transform, Walsh transform, arithmetic transform, and Haar transforms have been developed, and they are the graphical representations for the spectra of the logic functions. A new type of STDDs named ternary Walsh spectral transform decision diagram (TWSTDD) is proposed in this section, which is the graphical representations of ternary Walsh spectra for the logic functions.

From the relationships between ternary Walsh spectral and logical interpretations of the discrete functions, the spectral interpretation through decision trees can be derived.

Figure 8.5: TWSTDT node at level i .

Definition 8.7.1. The ternary Walsh spectral transform decision tree (TW-STDT) is a directed acyclic graph with exactly one root. It consists of two types of nodes:

- Sink is the terminal node which is located in the bottom of the graph. It has not outgoing edges and its value is over $\text{GF}(3)$, i.e., the sink values is in $\{0, 1, 2\}$.
- Non-sink node is the non-terminal node which has two outgoing edges, which are called low-edge and high-edge, respectively.

Let the root node of the TWSTDT be at level 1, the successor nodes of the node v which is at level i are at level $i + 1$, i.e., the level is in ascending order from the top to the bottom of the graph. For an n -variable function $f(\mathbf{x}_n)$, the non-sink node at i -th level from the root is shown in Fig. 8.5 where f_{node} denotes the logic function at level i in the TWSTDT, f_l and f_h represent the subfunctions of f which are corresponding to the subtrees determined by low-edge and high-edge, respectively. According to (8.27), the relationship between f_{node} and its subfunctions f_l and f_h can be represented over $\text{GF}(3)$ as follows:

$$f_{node} = 2^{u_n} \times_3 [f_l \oplus_3 (1 \oplus_3 x_{n+1-i}) \times_3 f_h]. \quad (8.78)$$

Figure 8.6: TWSTDT for the 3-variable function.

The sinks in the TWSTDT for the n -variable function are at level n . There are 2^n sinks in the TWSTDT and the values of the sinks are corresponding to the ternary Walsh coefficients of the logic function in sequential order. The TWSTDT for the 3-variable function is shown in Fig. 8.6 as an example.

Definition 8.7.2. The ternary Walsh spectral transform decision diagram (TWSTDD) is derived from TWSTDT by combining the sinks that have the same values. Hence, each sink has unique value in TWSTDD, and there are at most three sinks, i.e., the possible values of the sinks are in $\{0, 1, 2\}$.

The reduction of the size is one of the most important problem in decision diagram techniques. The aim of the STDDs is not only to try to provide a simple representation of discrete functions, but also to find a number of useful properties by transforming the information to the spectral domain. Hence, the simplification of the STDDs is helpful for the graphical representations of spectral transforms for discrete functions and for finding more of their properties.

Definition 8.7.3. The reduced TWSTDD (RTWSTDD) is the decision diagram obtained after the reductions from TWSTDD by the following rules.

Figure 8.7: Elimination rule for TWSTDD.

Figure 8.8: Merging rule for TWSTDD.

- Elimination rule: If two outgoing edges of a node v point to the same node u , then eliminate node v and redirect all the incoming edges of v to u , while the edge-values of the new incoming edges of v are the ternary products of the old edge-values of the incoming edges of v and the sum of the old edge-values of the outgoing edges of v . The elimination rule is illustrated in Fig. 8.7.
- Merging rule: If the two subtrees in the graph are isomorphic, then one of them can be deleted and the incoming edges of it with their edge-values can be redirected to the remaining one. This rule is illustrated in Fig. 8.8.

Example 8.7.1. Let the ternary Walsh spectrum of a 3-variable function is: $\mathbf{S}=[0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 2 \ 2]^T$. The corresponding TWSTDD can be constructed as shown in Fig. 8.9. Its RTWSTDD is derived from the TWSTDD as shown in Fig. 8.10.

Figure 8.9: TWSTDD for the 3-variable function in Example 8.7.1.

8.8 Experimental Results

Since the ternary Walsh transform is over $\text{GF}(3)$, it has simpler arithmetic than the standard Walsh transform which is performing over the real number field. In addition, the ternary Walsh transform keeps all the information of the logic functions as well as the standard Walsh transform and has the capability of handling the incompletely specified Boolean functions which cannot be processed efficiently by only the one standard Walsh transform [FSP92]. For an n -variable logic function, the value of its corresponding standard Walsh coefficient is in the range of $[-2^n, 2^n]$ which is increasing exponentially when n is increasing, while the value of the ternary Walsh coefficient is over $\{0, 1, 2\}$ which is not related to the variable numbers of the logic function. Hence, the ternary Walsh transform has advantages in terms of the storage space of the spectral coefficients over standard Walsh transform. Although the Reed-Muller transform that is also performing over $\text{GF}(2)$ requires less storage space than standard Walsh transform, Reed-Muller transform cannot process the incompletely specified Boolean

Figure 8.10: RTWSTDD for the 3-variable function in Example 8.7.1.

functions. The proposed ternary Walsh transform provides the lossless information of both the completely and incompletely specified Boolean functions as well as useful properties for processing them.

The number of non-zero spectral coefficients is another important factor in the estimation of the storage space required by using the spectral transform. Table 8.10 shows the comparison of the number of non-zero coefficients in some MCNC benchmark functions [Yan91] used in logic design by using the standard Walsh transform in S coding and R coding, and ternary Walsh transform in T coding, respectively. The benchmark functions shown in Table 8.10 are all multi-output functions. Since each output of the multi-output benchmark functions can be considered as a single function, the NAME- i represents the i -th output of the benchmark function 'NAME'. In Table 8.10, the columns under 'Input' and 'Output' are the numbers of inputs and outputs of the benchmark functions, the columns under 'Standard Walsh' are the numbers of non-zero standard Walsh coefficients for the benchmark functions in S coding and R coding, the columns under 'Ternary Walsh' are the numbers of non-zero ternary

Walsh coefficients for the benchmark functions in T coding, and the columns under 'Reductions%' show the reduction percentages of the numbers of non-zero coefficients for ternary Walsh transform over the standard Walsh transform using S coding and R coding. From Table 8.10, it can be noted that the ternary Walsh transform provides less non-zero coefficients to record the information of the benchmark functions than the standard Walsh transform, especially for the large functions, such as apex4 and ex1010. The average reduction percentages for all the benchmark functions shown in Table 8.10 are about 31.77% for S coding and 31.87% for R coding, respectively. Hence, the ternary Walsh transform also shows the advantages over the standard Walsh transform in terms of the number of non-zero coefficients.

8.9 Summary

In this chapter, the ternary Walsh transform and its different ternary operations are presented. The new proposed T coding for the representation of both the completely and incompletely specified Boolean functions is the first coding scheme which can represent the Boolean algebra including the consideration of don't care (DC) conditions by using the calculations over $GF(3)$. The applications of the ternary Walsh transform on the decomposition and symmetry detections of the Boolean functions are shown where the T coding is used to encode the functions. The corresponding STDD based on ternary Walsh transform is also developed for the representation of the ternary Walsh spectrum of the Boolean function. Compared to Walsh STDD, the TWSTDD requires less storage space since its values of sinks are in $GF(3)$ while the Walsh STDD's values of sinks are in $[-2^n, 2^n]$. By using presented ternary operations and their

properties and relations in the corresponding spectral data, we can design the function generator with larger dimensions that are especially important in the recent applications of the Walsh functions in VLSI testing, pattern recognition and digital signal and image processing [ABMR01, AAE95, Kar76, Kar85, SF96, SSJ98, Hur92, MB04, Par99, SA03, Yar03].

Table 8.10: Comparison of Number of Non-Zero Coefficients for Ternary Walsh Transform.

Function	Input	Output	Standard Walsh		TernaryWalsh	Reductions %	
			S coding	R coding	T coding	S coding	R coding
apex4-1	9	19	1	0	0	—	—
apex4-2	9	19	512	512	353	45.04	45.04
apex4-3	9	19	486	486	351	38.46	38.46
apex4-4	9	19	460	460	330	39.39	39.39
apex4-5	9	19	471	471	343	37.32	37.32
apex4-6	9	19	473	473	340	39.12	39.12
apex4-7	9	19	472	472	321	47.04	47.04
apex4-8	9	19	470	470	344	36.63	36.63
apex4-9	9	19	471	471	324	45.37	45.37
apex4-10	9	19	471	471	345	36.52	36.52
apex4-11	9	19	480	480	342	40.35	40.35
apex4-12	9	19	461	461	343	34.4	34.4
apex4-13	9	19	512	512	340	50.59	50.59
apex4-14	9	19	454	454	336	35.12	35.12
apex4-15	9	19	512	512	355	44.23	44.23
apex4-16	9	19	376	376	312	20.51	20.51
apex4-17	9	19	512	512	340	50.59	50.59
apex4-18	9	19	400	400	358	11.73	11.73
apex4-19	9	19	416	416	384	8.33	8.33
bw-1	5	28	28	28	23	21.74	21.74
bw-2	5	28	32	32	28	14.29	14.29
bw-3	5	28	32	32	23	39.13	39.13
bw-4	5	28	13	14	13	0	7.69
bw-5	5	28	32	32	24	33.33	33.33
bw-6	5	28	32	32	24	33.33	33.33
bw-7	5	28	32	32	19	68.42	68.42
bw-8	5	28	27	27	23	17.39	17.39
bw-9	5	28	32	32	21	52.38	52.38
bw-10	5	28	32	32	27	18.52	18.52
bw-11	5	28	18	18	17	5.88	5.88

continued on next page

continued from previous page							
Function	Input	Output	Standard Walsh		TernaryWalsh	Reductions %	
			S coding	R coding	T coding	S coding	R coding
bw-12	5	28	32	32	22	45.45	45.45
bw-13	5	28	32	32	27	18.52	18.52
bw-14	5	28	32	32	20	60	60
bw-15	5	28	32	32	23	39.13	39.13
bw-16	5	28	32	32	20	60	60
bw-17	5	28	32	32	27	18.52	18.52
bw-18	5	28	32	32	20	60	60
bw-19	5	28	32	32	25	28	28
bw-20	5	28	32	32	24	33.33	33.33
bw-21	5	28	32	32	23	39.13	39.13
bw-22	5	28	24	24	24	0	0
bw-23	5	28	32	32	25	28	28
bw-24	5	28	32	32	20	60	60
bw-25	5	28	32	32	24	33.33	33.33
bw-26	5	28	32	32	20	60	60
bw-27	5	28	32	32	17	88.24	88.24
bw-28	5	28	32	32	32	0	0
con1-1	7	2	40	40	34	17.65	17.65
con1-2	7	2	20	20	17	17.65	17.65
ex1010-1	10	10	1024	1024	702	45.87	45.87
ex1010-2	10	10	975	975	684	42.54	42.54
ex1010-3	10	10	1024	1024	691	48.19	48.19
ex1010-4	10	10	979	979	668	46.56	46.56
ex1010-5	10	10	1024	1024	703	45.66	45.66
ex1010-6	10	10	972	972	672	44.64	44.64
ex1010-7	10	10	977	977	666	46.7	46.7
ex1010-8	10	10	965	965	692	39.45	39.45
ex1010-9	10	10	1024	1024	690	48.41	48.41
ex1010-10	10	10	1024	1024	696	47.13	47.13
inc-1	7	9	45	45	39	15.38	15.38
inc-2	7	9	64	64	43	48.84	48.84
inc-3	7	9	80	80	64	25	25
inc-4	7	9	95	95	79	20.25	20.25
inc-5	7	9	128	128	109	17.43	17.43
continued on next page							

<i>continued from previous page</i>							
Function	Input	Output	Standard Walsh		TernaryWalsh	Reductions %	
			S coding	R coding	T coding	S coding	R coding
inc-6	7	9	80	80	60	33.33	33.33
inc-7	7	9	88	88	76	15.79	15.79
inc-8	7	9	128	128	93	37.63	37.63
inc-9	7	9	16	16	12	33.33	33.33
sao2-1	10	4	1024	1024	828	23.67	23.67
sao2-2	10	4	640	640	440	45.45	45.45
sao2-3	10	4	856	856	854	0.23	0.23
sao2-4	10	4	1024	1024	728	40.66	40.66
squar5-1	5	8	32	32	30	6.67	6.67
squar5-2	5	8	32	32	28	14.29	14.29
squar5-3	5	8	32	32	28	14.29	14.29
squar5-4	5	8	23	23	22	4.55	4.55
squar5-5	5	8	16	16	15	6.67	6.67
squar5-6	5	8	10	10	9	11.11	11.11
squar5-7	5	8	4	4	4	0	0
squar5-8	5	8	4	4	4	0	0

Chapter 9

Conclusion and Future Work

9.1 Conclusion

Spectral techniques are important ideas from the area of abstract harmonic analysis and provide alternative methods to solve the complex problems in terms of storage space and executing time. During the last decade there has been increasing interest in applications of different transforms used in digital signal processing for spectral analysis, synthesis and testing of discrete functions [AAE95, BGG98, Bla98, BSS99, Cas96, CMZ⁺97, Fal99b, FK99, FK00, KSA00, KSA03, RF98, RF99, RZ04, Sas99, Sch98, SF96, SF03, SS01, SSJ98, SSS96, Sta98, TDM01, TMS96, TMS97, TN95]. The different spectral transforms which convert the functions or signals from original domain to the corresponding spectral domain provide various new insights in the applications.

This dissertation is dedicated to the algorithms, analysis and applications of spectral techniques. The new algorithms for the efficient calculations of the spectral transforms have been developed and the analysis of Boolean algebra

in spectral domains has been demonstrated to be very useful for the effective calculation of the composite spectra. The developments of the new spectral transforms have been proposed which provide new useful properties in applications. The main contribution of this dissertation is summarized as follows.

- The mutual conversions between Hadamard-Haar and arithmetic transforms for an arbitrary transform size using matrix decomposition methods and butterfly diagrams as well as their computational costs have been developed. The relations between both transforms are introduced through recursive equations in the form of layered vertical and horizontal Kronecker products. The presented relations allow transferring known results of spectral logic design in arithmetic domain to Hadamard-Haar domain and vice versa and compare efficiency of both approaches in different applications for large discrete functions. It should be also noticed that presented derivations based on layered matrices and corresponding butterfly diagrams can be efficiently implemented in the form of operations on spectral decision diagrams, using software or as hardware operations using LUT cascades in a manner similar to Walsh transform [FS04]. Hence, all the presented results are not only very interesting theoretically but also are very important for the practical applications of Haar, Walsh, Hadamard-Haar and arithmetic transforms in many research areas.
- The relationships in arithmetic, Reed-Muller and Haar spectral domains corresponding to the Boolean operations in functional domain have been analyzed. The possibility of calculating only some expansion coefficients made possible by this research is very important since there are many methods in digital logic design for which the values of only selected arith-

metric, Reed-Muller or Haar coefficients are needed. The fundamental equalities of these relations are very useful since either representation may be more convenient for local optimization, optimal covering and technology mapping during different stages of VLSI CAD process. The stated equations are a working tool to translate knowledge about Boolean functions in both arithmetic, Reed-Muller and Haar transforms. The research summarized here will have not only impact on the more efficient applications of both representations in digital circuit design, verification and testing but also should open the new path in applications of such representations in other areas, such as machine learning, knowledge discovery and applied mathematics. The similar relationships for arithmetic product of the functions in different spectral domains were studied which is known as the convolution theorem. The convolution theorem plays important roles in linear function analysis, such as circular convolution for discrete Fourier transform and dyadic convolution for Walsh transform. Hence, the development for the similar relationships in arithmetic, Reed-Muller and Haar spectral domains for the Boolean logic product in functional domain is not only theoretically interesting, but also practically useful for the applications of arithmetic, Reed-Muller and Haar spectra in many areas of modern CAD tools for digital circuits.

- A unified method to generate fixed sign Walsh transform using an iterative hardware architecture has been shown. The fast fixed butterfly diagrams have been introduced to facilitate the calculation of the transform in a very efficient way. Another new non-linear transform developed is modified sign Haar transform which is related to sign Haar transform. The relationships between logic functions and modified sign Haar spectral representations

of Boolean functions have also been stated. Those properties of logic functions for different codings of incompletely specified functions will be useful in the application of modified Haar transform in logic synthesis. The sign Hadamard-Haar transform utilizes the butterfly structure of both sign Walsh and sign Haar transforms. Formulae to represent the sign Hadamard-Haar spectra for basic logic functions are shown. The sign Hadamard- γ -Haar- χ transform has been developed while the sign Walsh, sign Haar and sign Hadamard-Haar transforms can be seen as the special cases of sign Hadamard- γ -Haar- χ transform. It has been shown that the transform exhibits a non-linear property. Essentially, it transforms ternary data into ternary spectrum. Though non-linear, the transform is unique, and hence invertible. When a fast flow diagram is directly implemented in software, there is no need to keep the original data, and each consecutive butterfly requires a geometrically smaller number of operations on the transformed data. Each time, 2^n binary/ternary data need to be stored in the memory. The sign Hadamard- γ -Haar- χ transform block may also be implemented as a parallel dedicated processor.

- The multi-polarity arithmetic-Walsh transform with its fast algorithms and computational costs and its STDDs are proposed. The new multi-polarity transform has been used for some MCNC benchmark functions, which shows that it is suitable for the applications in CAD for VLSI as well as in the other areas where the standard Walsh and arithmetic transforms are currently applied. In particular it would be interesting to use the introduced transform for analysis, synthesis and calculation of properties of discrete functions that represent internal high-level models of digital signal processing elements including the VLSI CAD tools. The fundamental

equalities of the new transform are very useful since either representation may be more convenient for local optimization, optimal covering and technology mapping during different stages of VLSI design process. The stated equations, fast algorithms and corresponding decision diagrams are a working tool to translate knowledge about discrete functions.

- The generalizations of multi-polarity arithmetic-Haar transform for any size are presented with its fast algorithms. By selecting the polarity and r of the multi-polarity arithmetic-Haar transform, we can find the optimal expression for a given logic function. The fast algorithms for the new transform are also developed which can calculate the transform more efficiently. The new multi-polarity transform shows both the computational advantage and better experimental results over multi-polarity Walsh and multi-polarity arithmetic transforms. The calculation of this transform through DDs can also be developed, which is more suitable for large discrete functions. The new type of STDD named arithmetic-Haar STDD is also developed and it corresponds to the multi-polarity arithmetic-Haar transform. It provides the data structure to represent the arithmetic-Haar spectra of the given discrete functions.
- A new coding scheme for the representation of Boolean functions named T coding is designed. T coding which utilizes the $GF(3)$ algebra is the first coding scheme that can provide a unified algebra for the Boolean logic with the consideration of don't care (DC) conditions. Hence, the operations on both completely and incompletely specified Boolean functions can be processed by the new proposed T coding.
- The ternary Walsh transform performed over $GF(3)$ is developed which

requires simpler arithmetic and less storage space than standard Walsh transform. Considering DC conditions in Boolean functions, the ternary Walsh transform keeps exactly the proper information of both completely and incompletely specified Boolean functions. Hence, the ternary Walsh transform is suited for processing both completely and incompletely specified Boolean functions in T coding while the other spectral transforms can only handle the completely specified Boolean functions. Based on the properties for Boolean functions of the ternary Walsh transforms, the decomposition and symmetry detection of completely and incompletely specified Boolean functions using ternary Walsh transform are shown.

9.2 Future Work

A number of new open questions arise from the developments presented in this dissertation. A list of future research directions and open questions is compiled below.

- Application of the convolution theorems corresponding to arithmetic, Reed-Muller and Haar transforms for efficient logic testing and verifications.
- Development of the ternary cryptographic system by applying sign Hadamard- γ -Haar- χ transform.
- Application of multi-polarity arithmetic-Walsh and multi-polarity arithmetic-Haar transforms in CAD for VLSI where the Walsh and arithmetic transforms are currently applied.
- Analysis of Boolean functions by using T coding for Boolean algebra,

especially for the analysis of the incompletely Boolean functions.

- Application of ternary Walsh transform in classification of completely and incompletely specified Boolean functions, logic verification and synthesis, and testing of logic circuits.

Author's Publications

Journals

- [J1] B. J. Falkowski and S. Yan, "Walsh-Hadamard optimization of fixed polarity Reed-Muller transform," *IEICE Electronics Express*, vol. 1, no. 2, pp. 39–45, Apr. 2004.
- [J2] B. J. Falkowski and S. Yan, "Application of modified sign Haar transform in logic functions," *IEICE Electronics Express*, vol. 1, no. 11, pp. 305–310, Sept. 2004.
- [J3] B. J. Falkowski and S. Yan, "Fixed sign Walsh hardware structure," *IEICE Electronics Express*, vol. 2, no. 3, pp. 91–96, Feb. 2005.
- [J4] B. J. Falkowski and S. Yan, "Properties of logic functions in spectral domain of sign Hadamard-Haar transform," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 11, no. 1, pp. 185–211, Mar. 2005.
- [J5] B. J. Falkowski and S. Yan, "Matrix decomposition and butterfly diagrams for mutual relations between Hadamard-Haar and arithmetic spectra," *IEEE Transactions on Circuits and Systems—Part I: Regular Papers*, vol. 53, no. 5, pp. 1119–1129, May 2006.
- [J6] B. J. Falkowski and S. Yan, "Hybrid multi-polarity arithmetic-Walsh transform," *IEE Proceedings—Circuits, Devices & Systems*, vol. 153, no. 6, pp. 591–598, Dec. 2006.

Conferences

- [C1] B. J. Falkowski and S. Yan, "Sign Hadamard-Haar transform," in *Proceedings of 16th European Conference on Circuit Theory and Design*, vol. 2, Krakow, Poland, Sept. 2003, pp. 257–260.

- [C2] B. J. Falkowski and S. Yan, "Application of sign Hadamard-Haar transform in ternary communication system," in *Proceedings of 4th Workshop on Computational Intelligence and Information Technologies*, Nis, Serbia, Oct. 2003, pp. 135–138.
- [C3] B. J. Falkowski and S. Yan, "Comparison of computational costs of sign Hadamard-Haar, sign Walsh and ternary Reed-Muller transforms," in *Proceedings of 4th Workshop on Computational Intelligence and Information Technologies*, Nis, Serbia, Oct. 2003, pp. 139–142.
- [C4] B. J. Falkowski and S. Yan, "Hybrid arithmetic-Walsh transform," in *Proceedings of Joint Conference of the 4th International Conference on Information, Communications and Signal Processing, and the 4th Pacific Rim Conference on Multimedia*, vol. 3, Singapore, Dec. 2003, pp. 1831–1834.
- [C5] B. J. Falkowski and S. Yan, "Calculation of sign Hadamard-Haar transform by matrix approach," in *Proceedings of Joint Conference of the 4th International Conference on Information, Communications and Signal Processing, and the 4th Pacific Rim Conference on Multimedia*, vol. 3, Singapore, Dec. 2003, pp. 1845–1848.
- [C6] B. J. Falkowski and S. Yan, "Relationship between Hadamard-Haar and arithmetic transforms," in *Proceedings of 47th IEEE International Midwest Symposium on Circuits and Systems*, vol. 1, Hiroshima, Japan, July 2004, pp. 505–508.
- [C7] B. J. Falkowski and S. Yan, "Walsh-Hadamard spectral minimization of fixed polarity Reed-Muller expansions," in *Proceedings of 47th IEEE International Midwest Symposium on Circuits and Systems*, vol. 1, Hiroshima, Japan, July 2004, pp. 509–512.
- [C8] B. J. Falkowski and S. Yan, "Generalization of arithmetic-Haar transform for higher dimensions," in *Proceedings of 47th IEEE International Midwest Symposium on Circuits and Systems*, vol. 1, Hiroshima, Japan, July 2004, pp. 513–516.
- [C9] B. J. Falkowski and S. Yan, "Modified sign Haar transform," in *Proceedings of 47th IEEE International Midwest Symposium on Circuits and Systems*, vol. 2, Hiroshima, Japan, July 2004, pp. 133–136.
- [C10] B. J. Falkowski and S. Yan, "Comparison of sign Hadamard-Haar and sign Walsh transforms in ternary communication system," in *Proceedings of 10th International Symposium on Integrated Circuits, Devices and Systems*, Singapore, Sept. 2004, pp. 285–288.

- [C11] B. J. Falkowski and S. Yan, "Analysis of computational costs of sign Hadamard-Haar transform," in *Proceedings of 10th International Symposium on Integrated Circuits, Devices and Systems*, Singapore, Sept. 2004, pp. 471–474.
- [C12] B. J. Falkowski and S. Yan, "Ternary Walsh transform," in *Proceedings of 38th International Symposium on Circuits and Systems*, Kobe, Japan, May 2005, pp. 6202–6205.
- [C13] B. J. Falkowski and S. Yan, "Fixed sign Walsh transform and its iterative hardware architecture," in *Proceedings of 38th IEEE International Symposium on Circuits and Systems*, Kobe, Japan, May 2005, pp. 484–487.
- [C14] B. J. Falkowski and S. Yan, "Efficient calculations of composite spectra for arithmetic and Reed-Muller transforms," in *Proceedings of 48th IEEE International Midwest Symposium on Circuits and Systems*, Cincinnati, Ohio, Aug. 2005, pp. 1247–1250.
- [C15] B. J. Falkowski and S. Yan, "Properties and relations for hybrid multi-polarity arithmetic-Walsh transform," in *Proceedings of 48th IEEE International Midwest Symposium on Circuits and Systems*, Cincinnati, Ohio, Aug. 2005, pp. 583–586.
- [C16] B. J. Falkowski and S. Yan, "Development of multi-polarity arithmetic-Haar transform and its applications in logic design," in *Proceedings of 48th IEEE International Midwest Symposium on Circuits and Systems*, Cincinnati, Ohio, Aug. 2005, pp. 472–475.
- [C17] B. J. Falkowski and S. Yan, "New sign Hadamard- γ -Haar- χ transform in ternary communication system," in *Proceedings of 48th IEEE International Midwest Symposium on Circuits and Systems*, Cincinnati, Ohio, Aug. 2005, pp. 123–126.
- [C18] B. J. Falkowski and S. Yan, "Fast algorithms between arithmetic and Hadamard-Haar transforms," in *Proceedings of 48th IEEE International Midwest Symposium on Circuits and Systems*, Cincinnati, Ohio, Aug. 2005, pp. 1059–1062.
- [C19] B. J. Falkowski and S. Yan, "Arithmetic-Haar spectral transform decision diagrams," in *Proceedings of 36th IEEE International Symposium on Multiple-Valued Logic*, Singapore, May 2006.
- [C20] B. J. Falkowski and S. Yan, "Arithmetic-Walsh spectral transform decision diagrams," in *Proceedings of 2006 IEEE International Symposium on Information Theory*, Seattle, Washington, July 2006, pp. 2012–2016.

- [C21] B. J. Falkowski and S. Yan, “Properties and fast algorithms for ternary Walsh transform,” in *Proceedings of 37th IEEE International Symposium on Multiple-Valued Logic*, Oslo, Norway, May 2007.

Bibliography

- [AAE95] S. Agaian, J. T. Astola, and K. Egiazarian. *Binary Polynomial Transforms and Nonlinear Digital Filters*. Marcel Dekker, New York, 1995.
- [ABMR01] A. Amira, A. Bouridane, P. Milligan, and M. Roula. Novel FPGA implementations of Walsh-Hadamard transforms for signal processing. *IEE Proceedings—Vision, Image & Signal Processing*, 148(6):377–383, December 2001.
- [Aga80] S. Agaian. *Hadamard Matrices and Their Applications*, volume 1168 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1980.
- [Ake78] S. B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, 27(6):509–516, June 1978.
- [AR75] N. Ahmed and K. R. Rao. *Orthogonal Transforms for Digital Signal Processing*. Springer-Verlag, New York, 1975.
- [Bea75] K. G. Beauchamps. *Walsh Functions and Their Applications*. Academic Press, London, 1975.
- [Bea84] K. G. Beauchamps. *Applications of Walsh Functions and Related Functions with an Introduction to Sequency Theory*. Academic Press, London, 1984.
- [Bes78] Ph. W. Besslich. On the Walsh-Hadamard transform and prime implicant extraction. *IEEE Transactions on Electromagnetic Compatibility*, 20:516–519, November 1978.
- [BGG98] C. S. Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms*. Prentice Hall, Englewood Cliffs, 1998.
- [Bla74] N. M. Blachman. Sinusoids versus Walsh functions. *Proceedings of the IEEE*, 62(3):346–354, March 1974.
- [Bla98] R. E. Blahut. Decoding of cyclic codes on curves. In V. S. Pless and W. C. Huffman, editors, *Handbook of Coding Theory*, volume 2. Elsevier Science, Amsterdam, 1998.
- [BMS88] A. M. Buron, J. A. Michell, and M. Solana. Single chip fast Haar transform at megahertz rates. In C. Moraga, editor, *Theory and Application of Spectral Techniques*, pages 8–17. University Dortmund Press, Dortmund, 1988.
- [Boi81] L. L. Boiko. Generalized Fourier-Haar transform on finite Abelian groups. In L. P. Yaroslavsky, editor, *Digital Signal Processing and Applications*, pages 12–22. Nauka, Moscow, 1981. in Russian.

- [Boo54] G. Boole. *The Laws of Thought*. Macmillan, London, 1854. reprinted by Dover, New York, 1958.
- [Bri88] E. O. Brigham. *Fast Fourier Transform and Its Applications*. Prentice Hall, Upper Saddle River, 1988.
- [Bro90] F. Brown. *Boolean Reasoning*. Kluwer Academic, Boston, 1990.
- [Bry86] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, August 1986.
- [Bry91] R. E. Bryant. On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication. *IEEE Transactions on Computers*, 40(2):206–213, February 1991.
- [Bry92] R. E. Bryant. Symbolic Boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24(3):293–318, September 1992.
- [BSS99] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*, volume 265 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1999.
- [BT96] Ph. W. Besslich and E. A. Trachtenberg. Three-valued quasi-linear transformation for logic synthesis. *IEE Proceedings—Computers & Digital Techniques*, 143(6):391–400, November 1996.
- [Cal61] P. Calingaert. Switching function canonical forms based on commutative and associative binary operations. *Transactions of AIEE*, 52:808–814, 1961.
- [Cas96] K. R. Castleman. *Digital Image Processing*. Prentice Hall, Englewood Cliffs, 1996.
- [CFM⁺93] E. M. Clarke, M. Fujita, P. McGeer, J. Yang, and X. Zhao. Multi-terminal binary decision diagrams and hybrid decision diagrams. In *Proceedings of International Workshop on Logic Synthesis*, Tahoe City, California, May 1993.
- [CFZ95] E. M. Clarke, M. Fujita, and X. Zhao. Applications of multi-terminal binary decision diagrams. In *Proceedings of IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansions in Circuit Design*, pages 21–27, Makuhari, Chiba, Japan, August 1995.
- [CMZ⁺93] E. M. Clarke, K. L. Mcmillan, X. Zhao, M. Fujita, and J. Yang. Spectral transforms for large Boolean functions with applications to technology mapping. In *Proceedings of 30th ACM/IEEE Conference on Design Automation*, pages 54–60, Dallas, Texas, June 1993.
- [CMZ⁺97] E. M. Clarke, K. L. Mcmillan, X. Zhao, M. Fujita, and J. Yang. Spectral transforms for large Boolean functions with applications to technology mapping. *Formal Methods in System Design*, 10(2-3):137–148, April 1997.
- [Col61] R. P. Coleman. Orthogonal functions for the logical design of switching functions. *IRE Transactions on Computers*, 10:379–383, 1961.
- [Dam92] T. Damarla. Generalized transforms for multiple valued circuits and their fault detection. *IEEE Transactions on Computers*, 41(9):1101–1109, 1992.

- [DDT78] M. Davio, J. P. Deschamps, and A. Thayse. *Discrete and Switching Functions*. George and McGraw-Hill, New York, 1978.
- [Der65] M. L. Dertouzos. *Threshold Logic: A Synthesis Approach*. MIT Press, Cambridge, Massachusetts, 1965.
- [DiT91] N. DiToro. The sign transform: its application to fault detection in digital circuits and its efficient implementation. Master's thesis, Drexel University, Philadelphia, March 1991.
- [DM94] G. De Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, New York, 1994.
- [DMC97] B. K. Das, R. N. Mahapatra, and B. N. Chatterji. Modelling Hadamard Haar transform algorithm for omega connected multiprocessors. *Signal Processing*, 58(3):293–301, May 1997.
- [Dre00] R. Drechsler. *Formal Verification of Circuits*. Kluwer Academic, Boston, 2000.
- [Dry92] A. Drygajlo. On the Hadamard-Haar transforms and their multiresolution properties. In J. Vandewalle, R. Boite, M. Moonen, and A. Oosterlink, editors, *Signal Processing VI: Theories and Applications*, pages 1247–1250. Elsevier, Amsterdam, 1992.
- [Edw75] C. R. Edwards. The application of the Rademacher-Walsh transform to Boolean function classification and threshold logic synthesis. *IEEE Transactions on Computers*, 24(1):48–52, January 1975.
- [EH78] C. R. Edwards and S. L. Hurst. A digital synthesis procedure under function symmetries and mapping methods. *IEEE Transactions on Computers*, 27(11):985–997, November 1978.
- [ER82] D. F. Elliott and K. R. Rao. *Fast Transforms: Algorithms, Analysis, Applications*. Academic Press, London, 1982.
- [Eri78] E. Eris. Relationships between Rademacher-Walsh spectra of Boolean functions. *IEE Journal on Computers and Digital Techniques*, 1(2):45–48, May 1978.
- [Fal94] B. J. Falkowski. Properties and ways of calculation of multi-polarity generalized Walsh transforms. *IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing*, 41(6):380–391, June 1994.
- [Fal95] B. J. Falkowski. Recursive relationships, fast transforms, generalizations and VLSI iterative architecture for Gray code ordered Walsh functions. *IEE Proceedings—Computers & Digital Techniques*, 142(5):325–331, September 1995.
- [Fal99a] B. J. Falkowski. Haar transform, calculation, generalization and applications in logic design. In *Proceedings of 2nd International Workshop on Transforms and Filter Banks*, pages 101–120, Brandenburg, Germany, March 1999. Published as TICSP Series #4, March 2000.
- [Fal99b] B. J. Falkowski. A note on the polynomial form of Boolean functions and related topics. *IEEE Transactions on Computers*, 48(8):860–864, August 1999.

- [Fal99c] B. J. Falkowski. Relationship between arithmetic and Haar wavelet transforms in the form of layered Kronecker matrices. *Electronics Letters*, 35(10):799–800, May 1999.
- [Fal02] B. J. Falkowski. Generalized multi-polarity Haar transform. In *Proceedings of 35th IEEE International Symposium on Circuits and Systems*, volume 5, pages 757–760, Phoenix, Arizona, May 2002.
- [Fal05] B. J. Falkowski. Relationship between Haar and Reed-Muller spectral and functional domains. *IEICE Electronics Express*, 2(1):37–42, January 2005.
- [FC97a] B. J. Falkowski and C. H. Chang. Forward and inverse transformations between Haar spectra and ordered binary decision diagrams of Boolean functions. *IEEE Transactions on Computers*, 46(11):1272–1279, November 1997.
- [FC97b] B. J. Falkowski and C. H. Chang. Properties and methods of calculating generalized arithmetic and adding transforms. *IEE Proceedings—Computers & Digital Techniques*, 144(5):249–258, October 1997.
- [FC98] B. J. Falkowski and C. H. Chang. Mutual conversions between generalized arithmetic expansions and free binary decision diagrams. *IEE Proceedings—Circuits, Devices & Systems*, 145(4):219–228, August 1998.
- [Fin72] B. J. Fino. Relations between Haar and Walsh-Hadamard transform. *Proceedings of the IEEE*, 60(5):647–648, 1972.
- [FK99] B. J. Falkowski and S. Kannurao. Identification of Boolean symmetries in spectral domain of Reed-Muller transform. *Electronics Letters*, 35(16):1315–1316, August 1999.
- [FK00] B. J. Falkowski and S. Kannurao. Algorithm to identify skew symmetries through Walsh transform. *Electronics Letters*, 36(5):401–402, March 2000.
- [FK02] B. J. Falkowski and S. Kannurao. Generation of sign Walsh spectra from disjoint cubes of Boolean functions. *Computers & Electrical Engineering*, 28(6):631–641, November 2002.
- [FMY97] M. Fujita, P. McGeer, and J. Yang. Multi-terminal binary decision diagrams: an efficient data structure for matrix representation. *Formal Methods in System Design*, 10(2-3):149–169, April 1997.
- [Fol92] G. B. Folland. *Fourier Analysis and Its Applications*. Wadsworth & Brooks/Cole, Pacific Grove, California, 1992.
- [FP90a] B. J. Falkowski and M. A. Perkowski. A family of all essential radix-2 addition/-subtraction multi-polarity transforms: algorithms and interpretations in Boolean domain. In *Proceedings of 23rd IEEE International Symposium on Circuits and Systems*, pages 2913–2916, New Orleans, Louisiana, May 1990.
- [FP90b] B. J. Falkowski and M. A. Perkowski. On the calculation of generalized Reed-Muller canonical expansions from disjoint cube representation of Boolean functions. In *Proceedings of 33rd IEEE Midwest Symposium on Circuits and Systems*, pages 1131–1134, Calgary, Canada, August 1990.

- [FR94] B. J. Falkowski and S. Rahardja. Sign Haar transform. In *Proceedings of 27th IEEE International Symposium on Circuits and Systems*, volume 2, pages 161–164, London, United Kingdom, May 1994.
- [FR03] B. J. Falkowski and S. Rahardja. Operations on logic functions and variables through sign Haar spectra. *Computers & Electrical Engineering*, 29(1):1–12, January 2003.
- [FS04] B. J. Falkowski and T. Sasao. Implementation of Walsh function generator of order 64 using LUT cascades. In *Proceedings of 47th IEEE International Midwest Symposium on Circuits and Systems*, volume 3, pages 467–470, Hiroshima, Japan, July 2004.
- [FSP92] B. J. Falkowski, I. Schaefer, and M. A. Perkowski. Effective computer methods for the calculation of Rademacher-Walsh spectrum for completely and incompletely specified Boolean functions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(10):1207–1226, October 1992.
- [FTY87] H. Fleisher, M. Tavel, and J. Yeager. A computer algorithm for minimizing Reed-Muller canonical forms. *IEEE Transactions on Computers*, 36(2):247–250, 1987.
- [GES91] B. Golubov, A. Efimov, and V. Skvortsov. *Theory and Applications of Walsh Series and Transforms*. Kluwer Academic, Boston, 1991.
- [Gra81] A. Graham. *Kronecker Products and Matrix Calculus with Applications*. John Wiley, New York, 1981.
- [Haa10] A. Haar. Zur theorie der orthogonalen Funktionssysteme. *Mathematische Annalen*, 69:331–371, 1910.
- [Har69] H. F. Harmuth. *Transmission of Information by Orthogonal Functions*. Springer-Verlag, Berlin, 1969.
- [Hei91] K. D. Heidtmann. Arithmetic spectrum applied to fault detection for combinational networks. *IEEE Transactions on Computers*, 40(3):320–324, March 1991.
- [HK95] T. Hanyu and M. Kameyama. A 200 MHz pipelined multiplier using 1.5 V-supply multiple-valued MOS current-mode circuits with dual-rail source-coupled logic. *IEEE Journal of Solid-State Circuits*, 30(11):1239–1245, November 1995.
- [HMM85] S. L. Hurst, D. M. Miller, and J. C. Muzio. *Spectral Techniques in Digital Logic*. Academic Press, London, 1985.
- [HR68] P. Hammer and S. Rudeanu. *Boolean Methods in Operations Research*. Springer-Verlag, Berlin, 1968.
- [HS84] T. C. Hsiao and S. C. Seth. Analysis of the use of Rademacher-Walsh spectrum in compact testing. *IEEE Transactions on Computers*, 33(10):934–937, October 1984.
- [HS96] G. D. Hachtel and F. Somenzi. *Logic Synthesis and Verification Algorithms*. Kluwer Academic, Boston, 1996.

- [HS97] J. P. Hansen and M. Sekine. Decision diagrams based techniques for the Haar wavelet transform. In *Proceedings of 1st IEEE International Conference on Information, Communication and Signal Processing*, volume 1, pages 59–63, Singapore, September 1997.
- [HSB02] S. Hassoun, T. Sasao, and R. K. Brayton, editors. *Logic Synthesis and Verification*. Kluwer Academic, Boston, 2002.
- [Hur77] S. L. Hurst. Detection of symmetries in combinatorial functions by spectral means. *IEE Journal on Electronic Circuits and Systems*, 1(5):173–180, September 1977.
- [Hur78] S. L. Hurst. *Logic Processing of Digital Signals*. Grawe Russak & Edward Adward Arnold, London, 1978.
- [Hur81] S. L. Hurst. The Haar transform in digital network synthesis. In *Proceedings of 11st IEEE International Symposium on Multiple-Valued Logic*, pages 10–18, Oklahoma City, Oklahoma, 1981.
- [Hur92] S. L. Hurst. *Custom VLSI Microelectronics*. Prentice Hall, New York, 1992.
- [JABF92] J. Jain, J. A. Abraham, J. Bitner, and D. S. Fussell. Probabilistic verification of Boolean functions. *Formal Methods in System Design*, 1(1):61–115, July 1992.
- [Kar76] M. G. Karpovsky. *Finite Orthogonal Series in the Design of Digital Devices*. John Wiley, New York, 1976.
- [Kar85] M. G. Karpovsky, editor. *Spectral Techniques and Fault Detection*. Academic Press, Orlando, 1985.
- [KB81] S. K. Kumar and M. A. Breuer. Probabilistic aspects of Boolean switching functions via a new transform. *Journal of the ACM*, 28(3):502–520, July 1981.
- [KSA00] M. G. Karpovsky, R. S. Stankovic, and J. T. Astola. Spectral techniques for design and testing of computer hardware. In *Proceedings of 1st International Workshop on Spectral Techniques and Logic Design for Future Digital Systems*, pages 9–42, Tampere, Finland, June 2000. published as TICSP Series #10, December 2000.
- [KSA03] M. G. Karpovsky, R. S. Stankovic, and J. T. Astola. Reduction of sizes of decision diagrams by autocorrelation functions. *IEEE Transactions on Computers*, 52(5):592–606, May 2003.
- [KSY91] G. A. Kukharev, V. P. Shmerko, and S. N. Yanushkevich. *Parallel Processing of Binary Data in VLSI*. Univ. Publisher, Minsk, Bielarus, 1991. in Russian.
- [KSZ90] G. A. Kukharev, V. P. Shmerko, and E. N. Zaitseva. *Multiple-Valued Data Processing Algorithms and Systolic Processors*. Science and Engineering, Minsk, 1990. in Russian.
- [Kul84] W. Kulesza. *Systems of Spectral Analysis of Digital Data*. WKL, Warsaw, 1984. in Polish.
- [Lab84] V. G. Labunets. *Algebraic Theory of Signals and Systems (Digital Signal Processing)*. University Press, Krasnojarsk, 1984. in Russian.

- [Lec71] R. J. Lechner. Harmonic analysis of switching functions. In A. Mukhopadhyay, editor, *Recent Developments in Switching Theory*. Academic Press, New York, 1971.
- [LPV94] Y.-T. Lai, M. Pedram, and B. K. Vrudhula. EVBDD-based algorithms for integer linear programming, spectral transformation, and function decomposition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(8):959–975, August 1994.
- [LPV96] Y.-T. Lai, M. Pedram, and B. K. Vrudhula. Formal verification using edge-valued binary decision diagrams. *IEEE Transactions on Computers*, 45(2):247–255, February 1996.
- [LZ83] Z. H. Li and Q. S. Zhang. Introduction to bridge functions. *IEEE Transactions on Electromagnetic Compatibility*, 25(4):459–464, 1983.
- [Mal82] V. D. Malyugin. Representation of Boolean functions by arithmetic polynomials. *Automation and Remote Control*, 43(4):496–504, 1982.
- [Mal97] V. D. Malyugin. *Parallel Logic Calculations Using Arithmetic Polynomials*. Russian Academy of Sciences, Moscow, 1997. in Russian.
- [MB04] U. Meyer-Baese. *Digital Signal Processing with Field Programmable Gate Arrays*. Springer-Verlag, Berlin, 2nd edition, 2004.
- [Mil87] D. M. Miller, editor. *Developments in Integrated Circuit Testing*. Academic, London, 1987.
- [Moh92] P. S. Moharir. *Pattern-Recognition Transforms*. John Wiley, New York, 1992.
- [Mor78] C. Moraga. Introducing disjoint spectral translation in multiple-valued logic design. *Electronics Letters*, 14:241–243, 1978.
- [Mor82] B. M. Moret. Decision trees and diagrams. *ACM Computing Surveys*, 14(4):593–623, December 1982.
- [Muz78] J. C. Muzio. Evaluation of the spectra of sum and product functions. *IEE Journal on Computers and Digital Techniques*, 1(3):113–118, August 1978.
- [Muz80] J. C. Muzio. Composite spectra and the analysis of switching circuits. *IEEE Transactions on Computers*, 29(8):750–753, 1980.
- [Ohn85] M. Ohnita. Approximation of analytic functions by Haar functions. *IEICE Transactions*, 105(5):101–108, 1985.
- [OLRPP93] J. Ortega, A. Lloris-Ruiz, A. Prieto, and F. J. Pelayo. Test-pattern generation based on Reed-Muller coefficients. *IEEE Transactions on Computers*, 42(8):968–980, 1993.
- [Par99] K. K. Parhi. *VLSI Digital Signal Processing Systems: Design and Implementation*. John Wiley, New York, 1999.
- [PM75] K. P. Parker and E. J. McCluskey. Probabilistic treatment of general combinational networks. *IEEE Transactions on Computers*, 24(6):668–670, 1975.
- [RF98] S. Rahardja and B. J. Falkowski. Symmetry conditions of Boolean functions in complex Hadamard transform. *Electronics Letters*, 34(17):1634–1635, August 1998.

- [RF99] S. Rahardja and B. J. Falkowski. Application of linearly independent arithmetic transform in testing of digital circuits. *Electronics Letters*, 35(5):363–364, March 1999.
- [RJY77] K. R. Rao, A. Jalali, and P. Yip. Rationalized Hadamard-Haar transform. In *Proceedings of 11st Annual Asilomar Conference on Signals, Systems, and Computers*, pages 194–203, Pacific Grove, California, November 1977.
- [RJY80] K. R. Rao, A. Jalali, and P. Yip. Rationalized Hadamard-Haar transform. *Applied Mathematics and Computation*, 6(3):263–281, 1980.
- [RMB88] G. Ruiz, J. A. Michell, and A. Buron. Fault detection and diagnosis for MOS circuits from Haar and Walsh spectrum analysis: on the fault coverage of Haar reduced analysis. In C. Moraga, editor, *Theory and Application of Spectral Techniques*, pages 97–106. University Dortmund Press, Dortmund, 1988.
- [RMB92] G. Ruiz, J. A. Michell, and A. Buron. Switch-level fault detection and diagnosis environment for MOS digital circuits using spectral techniques. *IEEE Proceedings—Computers & Digital Techniques*, 139(4):293–307, July 1992.
- [RNR75a] K. R. Rao, M. A. Narasimham, and K. Revuluri. A family of discrete Haar transforms. *Computers & Electrical Engineering*, 2(4):367–388, November 1975.
- [RNR75b] K. R. Rao, M. A. Narasimhan, and K. Revuluri. Image data processing by Hadamard-Haar transform. *IEEE Transactions on Computers*, 24(9):888–896, 1975.
- [RZ01] K. Radecka and Z. Zilic. Relating arithmetic and Walsh spectra for verification by implicit error modeling. In *Proceedings of 5th International Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, pages 205–214, Mississippi, August 2001.
- [RZ04] K. Radecka and Z. Zilic. Design verification by test vectors and arithmetic transform universal test set. *IEEE Transactions on Computers*, 53(5):628–640, May 2004.
- [SA03] R. S. Stankovic and J. T. Astola. *Spectral Interpretation of Decision Diagrams*. Springer-Verlag, New York, 2003.
- [Sas93] T. Sasao, editor. *Logic Synthesis and Optimization*. Kluwer Academic, Boston, 1993.
- [Sas99] T. Sasao. *Switching Theory for Logic Synthesis*. Kluwer Academic, Boston, 1999.
- [Sch89] W. G. Schneeweiss. *Boolean Functions with Engineering Applications and Computer Programs*. Springer-Verlag, Berlin, 1989.
- [Sch98] W. G. Schneeweiss. On the polynomial form of Boolean functions: Derivations and applications. *IEEE Transactions on Computers*, 47(2):217–221, 1998.
- [SF96] T. Sasao and M. Fujita, editors. *Representation of Discrete Functions*. Kluwer Academic, Boston, 1996.
- [SF03] R. S. Stankovic and B. J. Falkowski. The Haar wavelet transform: its status and achievements. *Computers & Electrical Engineering*, 29(1):25–44, January 2003.

- [Sha38] C. E. Shannon. A symbolic analysis of relay and switching circuits. *Transactions of AIEE*, 57:713–723, March 1938.
- [Sho73] J. E. Shore. On the application of Haar functions. *IEEE Transactions on Communications*, 21(3):209–216, 1973.
- [SMA01] R. S. Stankovic, C. Moraga, and J. T. Astola. From Fourier expansions to arithmetic-Haar expressions on quaternion groups. *Applicable Algebra in Engineering, Communication and Computing*, 12(3):227–253, July 2001.
- [SS01] R. S. Stankovic and T. Sasao. A discussion on the history of research in arithmetic and Reed-Muller expressions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(9):1177–1179, September 2001.
- [SSJ98] R. S. Stankovic, M. Stankovic, and D. Jankovic. *Spectral Transforms in Switching Theory, Definitions and Calculations*. Science Publisher, Belgrade, 1998.
- [SSS96] R. S. Stankovic, M. R. Stojic, and M. Stankovic, editors. *Recent Development in Abstract Harmonic Analysis with Applications in Signal Processing*. Science Publisher, Belgrade, 1996.
- [Sta82] R. S. Stankovic. A note of the relation between Reed-Muller expansions and Walsh transforms. *IEEE Transactions on Electromagnetic Compatibility*, 24(1):68–70, February 1982.
- [Sta95] R. S. Stankovic. Some remarks about spectral transform interpretation of MTB-DDs and EVBDDs. In *Proceedings of 1st IEEE Asia and South Pacific Design Automation Conference*, pages 385–390, Chiba, Japan, August 1995.
- [Sta98] R. S. Stankovic. Some remarks on terminology in spectral techniques for logic design: Walsh transform and Hadamard matrices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(11):1211–1214, November 1998.
- [Swi69] D. A. Swick. Walsh function generation. *IEEE Transactions on Information Theory*, 15(1):167–168, January 1969.
- [SWSP90] F. Schipp, W. R. Wader, P. Simon, and J. Pal. *Walsh Series - An Introduction to Dyadic Harmonic Analysis*. Adam Hilger, Bristol, 1990.
- [TDM01] M. A. Thornton, R. Drechsler, and D. M. Miller. *Spectral Techniques in VLSI CAD*. Kluwer Academic, Boston, 2001.
- [Tho97] M. A. Thornton. Modified Haar transform calculation using digital circuit output probabilities. In *Proceedings of 1st IEEE International Conference on Information, Communications and Signal Processing*, volume 1, pages 52–58, Singapore, September 1997.
- [Tho99] M. A. Thornton. Probabilistic equivalence checking using partial Haar spectral diagrams. In *Proceedings of 4th International Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, pages 123–132, Victoria, Canada, August 1999.

- [TMD01] M. A. Thornton, D. M. Miller, and R. Drechsler. Transformations amongst the Walsh, Haar, arithmetic and Reed-Muller spectral domains. In *Proceedings of 5th International Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, pages 215–225, Mississippi, August 2001.
- [TMS96] C. C. Tsai and M. Marek-Sadowska. Generalized Reed-Muller forms as a tool to detect symmetries. *IEEE Transactions on Computers*, 45(1):33–40, January 1996.
- [TMS97] C. C. Tsai and M. Marek-Sadowska. Boolean functions classification via fixed polarity Reed-Muller forms. *IEEE Transactions on Computers*, 46(2):173–186, February 1997.
- [TN95] M. A. Thornton and V. S. S. Nair. Efficient calculation of spectral coefficients and their applications. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(11):1328–1341, November 1995.
- [VK95] M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding*. Prentice Hall, Upper Saddle River, 1995.
- [Wal23] J. L. Walsh. A closed set of orthogonal functions. *American Journal of Mathematics*, 45:5–24, 1923.
- [Wol98] W. Wolf. *Modern VLSI Design, Systems on Silicon*. Prentice Hall, Upper Saddle River, 1998.
- [WSW72] W. D. Wallis, A. P. Street, and J. S. Wallis. *Combinatorics: Room Squares, Sum-free Sets, Hadamard Matrices*. Springer-Verlag, Berlin, 1972.
- [Yan91] S. Yang. Logic synthesis and optimization benchmarks user guide, version 3.0. Technical report, Microelectronics Center of North Carolina, Research Triangle Park, North Carolina, 1991.
- [Yar85] L. P. Yaroslavsky. *Digital Picture Processing - An Introduction*. Springer-Verlag, Berlin, 1985.
- [Yar03] L. P. Yaroslavsky. *Digital Holography and Digital Image Processing*. Kluwer Academic, Boston, 2003.
- [YM91] J. Yang and G. De Micheli. Spectral transforms for technical mapping. Technical Report CSL-TR-91-498, Stanford University, December 1991.
- [Zal89] L. A. Zalmanzon. *Fourier, Walsh, and Haar Transforms and Their Applications in Control, Communication and Other Fields*. Nauka, Moscow, 1989. in Russian.
- [Zha89] Q. S. Zhang. *New Methods of Signal Information Transfer*. Aeronautics and Astronautics Publishers, Beijing, 1989. in Chinese.
- [ZZ93] Q. S. Zhang and Y. G. Zhang. *Theory and Applications of Bridge Functions*. Defense Industry Publisher, Beijing, 1993. in Chinese.