Unified algorithm to generate Walsh functions in four different orderings and its programmable hardware implementations

B.J. Falkowski and T. Sasao

Abstract: This paper presents an algorithm to generate Walsh functions in four different orderings: Hadamard, Harmuth, Paley and strict sequency. By the analysis of the properties and mutual relations among these four orderings, the authors found a unified approach to generate any of the orderings from the primary set of Rademacher functions. By using these properties, the authors developed a programmable Walsh function generator for 64 outputs by both field programmable gate arrays and lookup table cascades to estimate the amount of hardware and performance. Such a programmable Walsh function generator can be used in VLSI testing, CDMA, pattern recognition, as well as image and signal processing.

1 Introduction

Walsh transforms are orthogonal, normal and complete [1–14]. They are important spectral representations of logic functions as the spectral Walsh domain with its global information provides much deeper insight into the logical structure of combinatorial networks than logic domain [1, 3, 6–9, 11, 12]. Spectral representation based on the Walsh transforms have been used in the classification of logic functions, functional decompositions, logic synthesis, multiplexer synthesis, prime implicant extraction, threshold logic synthesis, analysis of logic complexity, analysis of balanced functions, detection of symmetries, linearisation of decision diagrams, state assignment, cascade realisations, testing, and technology mapping [1, 3, 7–9, 11, 12, 15–24].

The renewed interest in applications of spectral methods in VLSI circuits is caused by their excellent design for testability and the recent development of efficient methods to operate on spectra of logic functions directly from reduced representations such as arrays of disjoint cubes or decision diagrams (DDs) [9, 11, 12, 17, 18, 20–22, 24–27]. VLSI testing based on signatures, that are, the correlations between output functions and test vectors, use the Walsh spectral coefficients [3, 6, 7, 11, 18–20].

In the applications of the Walsh transform to CDMA, VLSI testing, cryptography, and digital signal and image processing, the fastest approach is the hardware implementation [1–8, 10, 13, 14, 18–20, 28–34]. The Walsh functions of order 64 are extensively used in CDMA systems [31], and when applied to digital signal and image processing and pattern recognition, the manipulation of multidimensional signals involves operations on a large number of data values [1–8, 10, 13, 14, 32–34]. Since such a process generally involves high computational costs and substantial processing time, many attempts have been made to find efficient computation methods. One of them is to speed-up the data computation by using dedicated hardware.

Many different definitions for Walsh functions are known and used for various applications. They include: Walsh functions in strict sequency ordering related to sal and cal symmetries; Walsh functions in dyadic Paley ordering known also as Gray code ordered functions; Walsh functions in natural Hadamard ordering; Walsh functions in X-ordering; and Walsh functions in reverse Gray ordering [1–8, 11–13, 20, 24, 35]. Relationships between different orderings of Walsh functions have been investigated [1–8, 11–14, 24, 29, 31, 35–37].

Various Walsh function generators exist that use different methods [2, 3, 6, 31]. The sequence generators having the widest applicability are those generating a set of Walsh functions, although in some cases these functions are obtained by first generating Rademacher functions. Ideally, the generated functions should be orthogonal to each other and some designs are better in achieving this than others. Also the known generators operate either on the values $+1$ and -1 or these values are converted to 0 and 1 accordingly. For the second case, the Walsh orderings form a *group* under modulo 2 addition and such generators can be easily implemented using multiplicative EXOR gates. Most generators yield just one of many possible Walsh functions at a time. A different type of generator is required if one wants to produce several Walsh functions at the same time that come close to the ideal of mathematical orthogonality, and for example Besslich built such a generator for 16 Walsh functions with the minimum, orthogonality error [3, 6]. It should be noticed that all the generators discussed in [2, 3, 6, 31] are built just for one Walsh ordering or one Walsh function only.

In this paper, we show a unified algorithm to generate Walsh functions for four different orderings: Hadamard, Harmuth, Paley and strict sequency. It is an extension of

IEE Proceedings online no. 20045123

doi: 10.1049/ip-vis:20045123

Paper first received 23rd July 2004 and in revised form 4th March 2005. Originally published online 23rd June 2005

B.J. Falkowski is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Block S1, 50 Nanyang Avenue, 639798, Singapore

T. Sasao is with the Department of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka 820-8502, Japan

E-mail: efalkowski@ntu.edu.sg

our previous article [29] where we have shown the method of the Walsh function generator of order 64 only for one Walsh ordering using lookup table cascades. Here, we modified algorithms presented in [2, 3, 31] to compute Walsh functions in strict sequency ordering directly from the primary set of Rademacher functions, and to compute the Walsh-Hadamard as well as Walsh-Paley functions from strict sequency ordering, as well as presented a new algorithm to generate Harmuth functions from strict sequency ordering. Due to the fact that the Walsh functions in strict sequency ordering can be derived directly from the primary set of Rademacher functions, all other four orderings can also be generated directly from it too. Also the algorithm proposed here generates Walsh functions in strict sequency ordering directly from the primary set of Rademacher functions, and not from Walsh functions in Rademacher ordering as shown in [7]. Note that the ith Rademacher function in this algorithm corresponds to $(i-1)$th Rademacher function in [1–8, 10–14, 24, 31]. In the new algorithm for all four Walsh functions orderings, the original 1's are kept, but -1's are replaced by 0, and all the Walsh functions are generated in the complete interval between 0 and 1 rather than between $-\frac{1}{2}$ and $\frac{1}{2}$ as shown in [6]. In view of the above adjustments, our Walsh functions in strict sequency ordering correspond directly to the right half of Harmuth sequency functions and the whole Harmuth functions, but in the interval 0 to 1 are also generated by the algorithm. After the changing of original symbols the basic properties like a modulo 2 addition of two Walsh functions yields another Walsh function hold for all our Walsh functions. In addition, based on the known relations between Walsh, Paley and Hadamard orderings [1–8, 11–14, 24, 31], we will show a design method for a programmable Walsh function generator that produces any of four possible orderings. Our programmable Walsh function generator is useful for VLSI testing and other applications, especially for the IS-95 CDMA system where Walsh functions of order 64 are extensively used [1–8, 10–14, 18, 20, 31–34].

Since generation of Walsh functions involves considerable computational cost and substantial processing time, many attempts have been made to find efficient computation methods. One of them is to speedup the computation by hardware, such as field programmable gate arrays (FPGAs) widely used in digital signal processing (DSP) [28, 32, 33]. Recently, lookup table (LUT) cascades have been used for realisations of logic functions due to their short design time, and easy implementation [38–40]. Since the interconnections in a cascade are limited only to the adjacent cells, the area and the delay for interconnections are very small. Thus LUT cascades are suitable for VLSI implementation. In this paper, we implemented a 64-output programmable Walsh function generator that produces four different sequences using both an FPGA and LUT cascades.

2 Basic properties and definitions of Walsh transform

Let $f(x)$ denote a completely specified Boolean function over a set of n input binary variables $X = (x_n, x_{n-1}, \ldots, x_2, x_1)$, where $x_i \in \{0, 1\}$ and $1 \leq i \leq n$. The truth vector of f and the encoded values of f are treated as column vectors $\vec{f}$ and $\vec{F}$, respectively. The Walsh spectra S and R of an n-variable Boolean function f are alternative canonical representations of f. In the S-encoding, the original elements of $\vec{f}$ confined to $\{0, 1\}$ are mapped to $\{1, -1\}$ of $\vec{F}$, where logic value '0' is denoted by 1, and logic value '1' is denoted by -1. In the R-encoding, the original elements of $\{0, 1\}$ are mapped to $\{0, 1\}$ of $\vec{F}$, where logic value '0' is denoted by 0, and logic value '1' is denoted by 1. In the following, we will show the definitions and properties of the spectrum for n-variable completely specified Boolean functions for the five well-known Walsh-type transforms matrices.

Definition 2.1: The Walsh-Rademacher spectrum $\vec{S}$ of $f(x)$ is a column vector given by, $\vec{S} = WR(n)\vec{F}$ where $\vec{S} = [s_\phi, s_0, s_1, s_2, \ldots, s_{n-1}, s_{01}, s_{02}, \ldots, s_{012\cdots n-1}]^T$, where T denotes the transpose operator. s_I are the spectral coefficients in Rademacher ordering, and I is the index of the coefficient, and $WR(n)$ is the Walsh-Rademacher matrix of order n.

Similar to definition 2.1, other Walsh orderings can be defined. The only differences among them are various reorderings of their rows in the matrices as well as the sign of rows in the matrices. The five matrices of order three for the Walsh orderings used in this paper are given below. In this article we compute Walsh functions in strict sequency ordering directly from the primary set of Rademacher functions and we generate the Hadamard, Paley and Harmuth orderings directly from strict sequency ordering. More explanation about our unified algorithm together with an example are given in the next Section. We also show the properties of their matrices. In definitions 2.2–2.6, we introduce five Walsh matrices. For each matrix, we can define corresponding functions. In definition 2.2, $\{\vec{R}_0, \vec{R}_1, \vec{R}_2\}$ is the primary set of Rademacher functions for $n = 3$.

Definition 2.2: The Walsh-Rademacher matrix of order three is given by

$$WR(3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} \vec{W}_0 \\ \vec{W}_1 \\ \vec{W}_3 \\ \vec{W}_7 \\ \vec{W}_2 \\ \vec{W}_6 \\ \vec{W}_4 \\ \vec{W}_5 \end{bmatrix} = \begin{bmatrix} \vec{1} \\ \vec{R}_0 \\ \vec{R}_1 \\ \vec{R}_2 \\ \vec{R}_0 \times \vec{R}_1 \\ \vec{R}_0 \times \vec{R}_2 \\ \vec{R}_1 \times \vec{R}_2 \\ \vec{R}_0 \times \vec{R}_1 \times \vec{R}_2 \end{bmatrix}$$

where the symbol '$\times$' denotes bit-wise product among vectors.

Definition 2.3: The Walsh-Hadamard matrix of order three is given by

$$WH(3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} \vec{W}_0 \\ \vec{W}_7 \\ \vec{W}_3 \\ \vec{W}_4 \\ \vec{W}_1 \\ \vec{W}_6 \\ \vec{W}_2 \\ \vec{W}_5 \end{bmatrix} = \begin{bmatrix} \vec{1} \\ \vec{R}_2 \\ \vec{R}_1 \\ \vec{R}_1 \times \vec{R}_2 \\ \vec{R}_0 \\ \vec{R}_0 \times \vec{R}_2 \\ \vec{R}_0 \times \vec{R}_1 \\ \vec{R}_0 \times \vec{R}_1 \times \vec{R}_2 \end{bmatrix}$$

Definition 2.4: The Walsh matrix in strict sequency of order three is given by

$$WS(3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} \vec{W}_0 \\ \vec{W}_1 \\ \vec{W}_2 \\ \vec{W}_3 \\ \vec{W}_4 \\ \vec{W}_5 \\ \vec{W}_6 \\ \vec{W}_7 \end{bmatrix} = \begin{bmatrix} \vec{1} \\ \vec{R}_0 \\ \vec{R}_0 \times \vec{R}_1 \\ \vec{R}_1 \\ \vec{R}_1 \times \vec{R}_2 \\ \vec{R}_0 \times \vec{R}_1 \times \vec{R}_2 \\ \vec{R}_0 \times \vec{R}_2 \\ \vec{R}_2 \end{bmatrix}$$

Definition 2.5: The Walsh-Paley matrix of order three is given by

$$WP(3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} \vec{W}_0 \\ \vec{W}_1 \\ \vec{W}_3 \\ \vec{W}_2 \\ \vec{W}_7 \\ \vec{W}_6 \\ \vec{W}_4 \\ \vec{W}_5 \end{bmatrix} = \begin{bmatrix} \vec{1} \\ \vec{R}_0 \\ \vec{R}_1 \\ \vec{R}_0 \times \vec{R}_1 \\ \vec{R}_2 \\ \vec{R}_0 \times \vec{R}_2 \\ \vec{R}_1 \times \vec{R}_2 \\ \vec{R}_0 \times \vec{R}_1 \times \vec{R}_2 \end{bmatrix}$$

Definition 2.6: The Walsh-Harmuth matrix of order three is given by

$$WHR(3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} \vec{W}_0 \\ -\vec{W}_1 \\ -\vec{W}_2 \\ \vec{W}_3 \\ \vec{W}_4 \\ -\vec{W}_5 \\ -\vec{W}_6 \\ \vec{W}_7 \end{bmatrix} = \begin{bmatrix} \vec{1} \\ -\vec{R}_0 \\ -[\vec{R}_0 \times \vec{R}_1] \\ \vec{R}_1 \\ \vec{R}_1 \times \vec{R}_2 \\ -[\vec{R}_0 \times \vec{R}_1 \times \vec{R}_2] \\ -[\vec{R}_0 \times \vec{R}_2] \\ \vec{R}_2 \end{bmatrix}$$

Definition 2.7: The Walsh functions inside the Walsh-Rademacher matrix are called Walsh-Rademacher functions. The Walsh functions in other Walsh matrix orderings are defined in similar ways.

Definition 2.8: In the Walsh-Rademacher matrix $WR(n)$ of order n, the first row denotes the constant 1 function. The next n rows denote the primary Rademacher functions. The next $\binom{n}{2}$ rows denote the products of two primary Rademacher functions. The next $\binom{n}{3}$ rows denote the products of three primary Rademacher functions, etc. And, the last row denotes the product of n primary Rademacher functions. Any two different primary Rademacher functions are mutually orthogonal, but the products of the Walsh-Rademacher functions are not complete.

Property 2.1: The Walsh transform matrix of each sequence in definitions 2.3–2.6 is orthogonal and complete; therefore, the S and R spectra contain all the information of the Boolean function $\vec{f}$. Each row depends on different subset of the primary Rademacher functions.

Property 2.2: The sequency of a Walsh function is defined as the number of zero crossings in one cycle. In strict sequency order, each row has one more crossings between 1's and −1's than the row above.

Property 2.3: Only the Walsh-Hadamard matrix in definition 2.3 has the recursive Kronecker product structure. Other matrices in definitions 2.2, 2.4–2.6, do not have this property.

Property 2.4: The Walsh-Rademacher matrix in definition 2.2 and the Walsh-Harmuth matrix in definition 2.6 are non-symmetric. All other matrices in definitions 2.3, 2.4, and 2.5 are symmetric, so that, disregarding a scaling factor, the same matrix can be used for both the forward and inverse transformations.

Property 2.5: The Walsh functions in strict sequency can be generated from the primary set of Rademacher functions as shown in algorithm 3.1. To generate, we use EXOR operators, add a constant vector and convert indices of the natural binary code of the Walsh-Rademacher functions into Gray code sequence (see Tables 3.1 and 3.3).

Property 2.6: The Walsh-Paley, the Walsh in strict sequency as well as the Walsh-Hadamard functions are related as shown in algorithm 3.1 through reversing bit positions of the indexes, through a conversion between Gray code and natural binary code, or through combination of both of these (see Tables 3.1 and 3.2).

Property 2.7: The Walsh matrix in strict sequency is similar to the Walsh-Harmuth matrix: the only difference between them is addition of minus signs for the vectors having two least significant indices $[b_1, b_0] = [0, 1]$ *or* $[1, 0]$ (see Table 3.4). From definitions 2.4 and 2.6, we can confirm that when the Walsh matrix in strict sequency order is divided into two: $WS = [WS_{LEFT} : WS_{RIGHT}]$, we have the Walsh-Harmuth matrix as follows: $WHR = [WS_{RIGHT} : WS_{LEFT}]$.

Property 2.8: When the classical matrix multiplication method is used to generate the spectral coefficients for different Walsh transforms (different matrices in definitions 2.2–2.6 represent different Walsh functions in different orderings), the only difference in the result is the ordering of the spectral coefficients. The coded vector $\vec{F}$ corresponding to the original truth vector of a Boolean function f is the same for all the orderings of the Walsh functions. The values of the spectral coefficients s_I and r_I with the same indices are the same for every ordering of the Walsh transforms. In the ordering of the Walsh functions in strict sequency, most of the neighbouring rows are related to each other. This property applies not only to the neighbouring rows in the matrix, but also to the groups of neighbouring rows.

3 Algorithm to generate four orderings of Walsh functions

Based on the properties and definitions in Section 2, we will present the algorithm to generate four different Walsh functions where in the Walsh functions the original 1's are kept, but $-1'$s' are replaced by 0. Walsh functions in strict sequency ordering are directly generated from the primary Rademacher functions, and the other three orderings are generated from Walsh functions in strict sequency ordering, so in general, all of them can be generated from the primary Rademacher functions $R_i (i = 0, 1, 2, \ldots, n-1)$. Fig. 3.1 gives the relations among four Walsh orderings and transforms of order n, and the algorithm below gives the details.

Note that in definitions 2.2–2.6, we use the $\{-1, 1\}$ coding, while in the algorithm, we will use the $\{0, 1\}$ coding. A product of the rows of the matrices described by definitions 2.2–2.6 corresponds to an EXOR or an EXNOR of the primary Rademacher functions in the algorithm.

Algorithm 3.1:

1. Let $\vec{B} = [b_{n-1}, b_{n-2}, \ldots, b_i, \ldots, b_0]$ be a binary vector representing an index of the Walsh function, where $0 \leq i \leq n-1$.
2. Obtain the Gray code $\vec{G} = [g_{n-1}, g_{n-2}, \ldots, g_i, \ldots, g_0]$ from the natural binary code $\vec{B} = [b_{n-1}, b_{n-2}, \ldots, b_i, \ldots, b_0]$, where $g_i = b_{i+1} \oplus b_i$ for $0 \leq i \leq n-2$, and $g_{n-1} = b_{n-1}$. (Table 3.1 shows the relation between the natural binary code and the Gray code for $n = 5$, and a similar Table can be obtained for $n > 5$) [2–4, 6, 31].
3. Generate the Walsh functions in strict sequency ordering as follows: $\vec{W}_{\vec{B}} = \vec{1} \times \bar{b}_0 \oplus \sum_{i=0}^{n-1} \oplus g_i \vec{R}_i$, where $\vec{R}_i$ denotes the ith Rademacher function, and $\vec{1}$ is the binary vector consisting of n 1's. Note that the element $\vec{1} \times \vec{b}_0$ disappears from this equation when $\vec{B}$ represents an odd number since $\bar{b}_0 = 0$.
4. To generate the Walsh-Paley functions $\overrightarrow{WP}_{\vec{G}}$ from the Walsh functions in strict sequency ordering, perform the following operations: $\overrightarrow{WP}_{\vec{G}} = \vec{W}_{\vec{B}}$. In this case instead of using the operations in step 2, we can convert the Gray code $\vec{G} = [g_{n-1}, g_{n-2}, \ldots, g_i, \ldots, g_0]$ to the natural binary code $\vec{B} = [b_{n-1}, b_{n-2}, \ldots, b_i, \ldots, b_0]$ by using Table 3.2 or its extension for $n > 5$.
5. To generate the Walsh-Hadamard functions $\overrightarrow{WH}_{\vec{G}}$ from the Walsh functions in strict sequency ordering, perform the following operations: $\overrightarrow{WH}_{\vec{G}} = \vec{W}_{\vec{B}}$. In this case, given the Gray code $\vec{G} = [g_{n-1}, g_{n-2}, \ldots, g_i, \ldots, g_0]$, first reverse the order of bits to obtain $\vec{G}^r = [g_0, g_1, \ldots, g_i, \ldots, g_{n-1}]$, and then obtain the natural binary code $\vec{B} = [b_{n-1}, b_{n-2}, \ldots, b_i, \ldots, b_0]$, for which we can use Table 3.2 or its extension.
6. To generate Harmuth ordering $\vec{H}_{\vec{B}}$ from the Walsh functions in strict sequency ordering, perform the following operations:

$$\begin{array}{lllll} \vec{H}_{\vec{B}} = \vec{W}_{\vec{B}} & \text{if} & [b_1, b_0] = [0, 0] & \text{or} & [1, 1] \\ \vec{H}_{\vec{B}} = 1 \oplus \vec{W}_{\vec{B}} & \text{if} & [b_1, b_0] = [0, 1] & \text{or} & [1, 0] \end{array}$$

Note that the ith Rademacher function in this algorithm corresponds to $(i-1)$th Rademacher function in algorithms in [1–8, 11–13, 24, 31]. However, the basic properties such that a modulo 2 addition of two different Walsh functions yields another Walsh function holds for all our Walsh functions, too. Thus we can construct the Walsh functions in strict sequency ordering as well as in Harmuth ordering by using the primary Rademacher functions or its extensions. In the method described in [31], each Walsh ordering is generated just from one equation. However, we can generate each Walsh function by either using the set of the primary Rademacher functions or already generated other Walsh functions in strict sequency ordering. Tables 3.3 and 3.4 show the four primary Rademacher functions as well as calculation of the first 16 Walsh functions in sequency and Harmuth orderings for $n = 4$ based on our algorithm 3.1. Note that sometimes we need to add 1 in our unified algorithm and it is not the case in known algorithms based on other representations of Walsh functions. For example, in Table 3.3, to generate the second Walsh function $\vec{W}_2$ in strict sequency ordering, we can use the first two Rademacher functions. In generating the Walsh functions of higher dimensions, we have more ways to generate a given Walsh function from the primary Rademacher functions. Table 3.4 also shows the relation between the Walsh-Harmuth functions and the Walsh functions in strict sequency ordering. Note that half of them are exactly the same, and the another half has opposite signs that is the result of property 2.7. Since we can generate all four Walsh orderings, i.e. Hadamard, Harmuth, Paley and strict sequency, directly from the primary Rademacher functions, or we can generate Hadamard, Harmuth and Paley orderings from the Walsh functions in strict sequency ordering that is obtained from Rademacher functions, we can build a programmable function generator with a small amount of hardware. In example 3.1, we show an application of our method to generate two selected Walsh functions in strict sequency, Paley, Hadamard and Harmuth orderings. We can extend Tables 3.3 and 3.4 for any size of n based on algorithm 3.1.

Example 3.1: Let us generate two functions in four Walsh orderings for $n = 5$ using algorithm 3.1. For the first

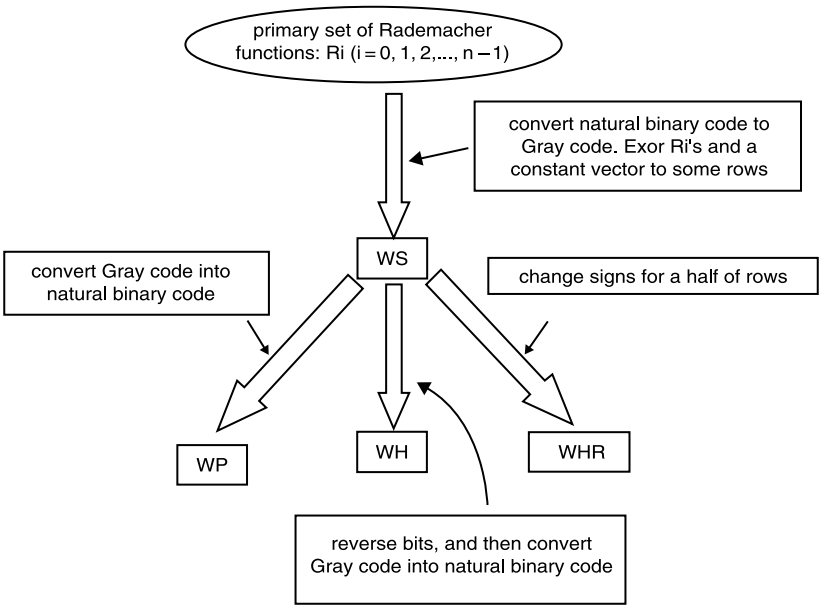

Fig. 3.1 *Derivation of four Walsh orderings from primary set of Rademacher functions*

Table 3.1: Conversion between natural binary code and Gray code for $n = 5[2, 3, 4, 6, 31]$

Natural binary code $\vec{B} = [b_4, b_3, b_2, b_1, b_0]$	Gray code $\vec{G} = [g_4, g_3, g_2, g_1, g_0]$
00000	00000
00001	00001
00010	00011
00011	00010
00100	00110
00101	00111
00110	00101
00111	00100
01000	01100
01001	01101
01010	01111
01011	01110
01100	01010
01101	01011
01110	01001
01111	01000
10000	11000
10001	11001
10010	11011
10011	11010
10100	11110
10101	11111
10110	11101
10111	11100
11000	10100
11001	10101
11010	10111
11011	10110
11100	10010
11101	10011
11110	10001
11111	10000

Table 3.2: Conversion between Gray code and natural binary code for $n = 5[2, 3, 4, 6, 31]$

Gray code $\vec{G} = [g_4, g_3, g_2, g_1, g_0]$	Natural binary code $\vec{B} = [b_4, b_3, b_2, b_1, b_0]$
00000	00000
00001	00001
00010	00011
00011	00010
00100	00111
00101	00110
00110	00100
00111	00101
01000	01111
01001	01110
01010	01100
01011	01101
01100	01000
01101	01001
01110	01011
01111	01010
10000	11111
10001	11110
10010	11100
10011	11101
10100	11000
10101	11001
10110	11011
10111	11010
11000	10000
11001	10001
11010	10011
11011	10010
11100	10111
11101	10110
11110	10100
11111	10101

Table 3.3: Generation of first 16 Walsh functions in strict sequency order for n = 4 based on algorithm 3.1

$\vec{R}_0 = [1111111100000000]$
$\vec{R}_1 = [1111000011110000]$
$\vec{R}_2 = [1100110011001100]$
$\vec{R}_3 = [1010101010101010]$
$\vec{W}_0 = \vec{1} = [1111111111111111]$
$\vec{W}_1 = \vec{R}_0 = [1111111100000000]$
$\vec{W}_2 = \vec{1} \oplus \vec{R}_1 \oplus \vec{R}_0 = [1111000000001111]$
$\vec{W}_3 = \vec{R}_1 = [1111000011110000]$
$\vec{W}_4 = \vec{1} \oplus \vec{R}_2 \oplus \vec{R}_1 = [1100001111000011]$
$\vec{W}_5 = \vec{R}_2 \oplus \vec{R}_1 \oplus \vec{R}_0 = [1100001100111100]$
$\vec{W}_6 = \vec{1} \oplus \vec{R}_2 \oplus \vec{R}_0 = [1100110000110011]$
$\vec{W}_7 = \vec{R}_2 = [1100110011001100]$
$\vec{W}_8 = \vec{1} \oplus \vec{R}_3 \oplus \vec{R}_2 = [1001100110011001]$
$\vec{W}_9 = \vec{R}_3 \oplus \vec{R}_2 \oplus \vec{R}_0 = [1001100101100110]$
$\vec{W}_{10} = \vec{1} \oplus \vec{R}_3 \oplus \vec{R}_2 \oplus \vec{R}_1 \oplus \vec{R}_0 = [1001011001101001]$
$\vec{W}_{11} = \vec{R}_3 \oplus \vec{R}_2 \oplus \vec{R}_1 = [1001011001101001]$
$\vec{W}_{12} = \vec{1} \oplus \vec{R}_3 \oplus \vec{R}_1 = [1010010110100101]$
$\vec{W}_{13} = \vec{R}_3 \oplus \vec{R}_1 \oplus \vec{R}_0 = [1010010101011010]$
$\vec{W}_{14} = \vec{1} \oplus \vec{R}_3 \oplus \vec{R}_0 = [1010101001010101]$
$\vec{W}_{15} = \vec{R}_3 = [1010101010101010]$

Table 3.4: Generation of first 16 Walsh functions in Harmuth order for n = 4 based on algorithm 3.1

$\vec{H}_0 = \vec{W}_0 = 1 = [1111111111111111]$
$\vec{H}_1 = \vec{1} \oplus \vec{W}_1 = \vec{1} \oplus \vec{R}_0 = [0000000011111111]$
$\vec{H}_2 = \vec{1} \oplus \vec{W}_2 = \vec{R}_1 \oplus \vec{R}_0 = [0000111111110000]$
$\vec{H}_3 = \vec{W}_3 = \vec{R}_1 = [1111000011110000]$
$\vec{H}_4 = \vec{W}_4 = \vec{1} \oplus \vec{R}_2 \oplus \vec{R}_1 = [1100001111000011]$
$\vec{H}_5 = \vec{1} \oplus \vec{W}_5 = \vec{1} \oplus \vec{R}_2 \oplus \vec{R}_1 \oplus \vec{R}_0 = [0011110011000011]$
$\vec{H}_6 = \vec{1} \oplus \vec{W}_6 = \vec{R}_2 \oplus \vec{R}_0 = [0011001111001100]$
$\vec{H}_7 = \vec{W}_7 = \vec{R}_2 = [1100110011001100]$
$\vec{H}_8 = \vec{W}_8 = \vec{1} \oplus \vec{R}_3 \oplus \vec{R}_2 = [1001100110011001]$
$\vec{H}_9 = \vec{1} \oplus \vec{W}_9 = \vec{1} \oplus \vec{R}_3 \oplus \vec{R}_2 \oplus \vec{R}_0 = [0110011010011001]$
$\vec{H}_{10} = \vec{1} \oplus \vec{W}_{10} = \vec{R}_3 \oplus \vec{R}_2 \oplus \vec{R}_1 \oplus \vec{R}_0 = [0110100110010110]$
$\vec{H}_{11} = \vec{W}_{11} = \vec{R}_3 \oplus \vec{R}_2 \oplus \vec{R}_1 = [1001011001101001]$
$\vec{H}_{12} = \vec{W}_{12} = \vec{1} \oplus \vec{R}_3 \oplus \vec{R}_1 = [1010010110100101]$
$\vec{H}_{13} = \vec{1} \oplus \vec{W}_{13} = \vec{1} \oplus \vec{R}_3 \oplus \vec{R}_1 \oplus \vec{R}_0 = [0101101010100101]$
$\vec{H}_{14} = \vec{1} \oplus \vec{W}_{14} = \vec{R}_3 \oplus \vec{R}_0 = [0101010110101010]$
$\vec{H}_{15} = \vec{W}_{15} = \vec{R}_3 = [1010101010101010]$

Walsh function in strict sequency $\vec{W}_{21}$, in step 1, we have $\vec{B} = [b_4, b_3, b_2, b_1, b_0] = [1, 0, 1, 0, 1]$.

In step 2, we have $\vec{G} = [g_4, g_3, g_2, g_1, g_0] = [b_4, b_4 \oplus b_3, b_3 \oplus b_2, b_2 \oplus b_1, b_1 \oplus b_0] = [1, 1, 1, 1, 1]$. $\vec{B}$ represents an odd number and $\bar{b}_0 = 0$. In Table 3.1, we can verify that the natural binary code 21 corresponds to the Gray code 31.

In step 3, we have $\vec{W}_{21} = \vec{1} \times \bar{b}_0 \oplus \sum_{i=0}^{4} \oplus g_i \vec{R}_i = \vec{1} \times 0 \oplus g_4\vec{R}_4 \oplus g_3\vec{R}_3 \oplus g_2\vec{R}_2 \oplus g_1\vec{R}_1 \oplus g_0\vec{R}_0 = \vec{R}_4 \oplus \vec{R}_3 \oplus \vec{R}_2 \oplus \vec{R}_1 \oplus \vec{R}_0$.

In step 4, we have $\vec{G} = [g_4, g_3, g_2, g_1, g_0] = [1, 0, 1, 0, 1] = [1, 1 \oplus 1, 1 \oplus 0, 0 \oplus 0, 0 \oplus 1] = [b_4, b_4 \oplus b_3, b_3 \oplus b_2, b_2 \oplus b_1, b_1 \oplus b_0]$. Hence, $\vec{B} = [b_4, b_3, b_2, b_1, b_0] = [1, 1, 0, 0, 1]$ and $\overrightarrow{WP}_{21} = \vec{W}_{25}$. In Table 3.2, we can verify that the Gray code 21 corresponds to the natural binary code 25: $\overrightarrow{WP}_{21} = \vec{W}_{25}$.

In step 5, we have $\vec{G} = [g_4, g_3, g_2, g_1, g_0] = [1, 0, 1, 0, 1]$ and $\vec{G}^r = [g_0, g_1, g_2, g_3, g_4] = [1, 0, 1, 0, 1]$. In Table 3.2, we can verify that the Gray code 21 corresponds to the natural binary code 25 so that $\overrightarrow{WP}_{21} = \vec{W}_{25}$. Notice that $\overrightarrow{WH}_{21} = \overrightarrow{WP}_{21} = \vec{W}_{25}$.

In step 6, Harmuth ordering for the above function is generated. Since $[b_1, b_0] = [0, 1]$, we have $\vec{H}_{21} = \vec{1} \oplus \vec{W}_{21} = \vec{1} \oplus \vec{R}_4 \oplus \vec{R}_3 \oplus \vec{R}_2 \oplus \vec{R}_1 \oplus \vec{R}_0$. Note that instead of adding the constant vector $\vec{1}$, we can complement any Rademacher function.

For the second Walsh function in strict sequency $\vec{W}_{22}$, in step 1 we have $\vec{B} = [1, 0, 1, 1, 0]$.

In step 2, we have $\vec{G} = [g_4, g_3, g_2, g_1, g_0] = [b_4, b_4 \oplus b_3, b_3 \oplus b_2, b_2 \oplus b_1, b_1 \oplus b_0] = [1, 1, 1, 0, 1]$. $\vec{B}$ represents an even number, and $\bar{b}_0 = 1$. In Table 3.1, we can verify that the natural binary code 22 corresponds to the Gray code 29.

In step 3, $\vec{W}_{22} = \vec{1} \times \bar{b}_0 \oplus \sum_{i=0}^{4} \oplus g_i \vec{R}_i = \vec{1} \times 1 \oplus \sum_{i=0}^{4} \oplus (g_i \vec{R}_i) \oplus (\vec{1}\bar{b}_0) = \vec{1} \oplus g_4\vec{R}_4 \oplus g_3\vec{R}_3 \oplus g_2\vec{R}_2 \oplus g_1\vec{R}_1 \oplus g_0\vec{R}_0 = \vec{1} \oplus \vec{R}_4 \oplus \vec{R}_3 \oplus \vec{R}_2 \oplus \vec{R}_0$. Also in this case, we can complement any Rademacher function instead of adding the constant vector $\vec{1}$.

In step 4, we have $\vec{G} = [g_4, g_3, g_2, g_1, g_0] = [1, 0, 1, 1, 0] = [0 \oplus 1, 1 \oplus 1, 1 \oplus 0, 0 \oplus 1, 1 \oplus 1]$. Hence, $\vec{B} = [b_4, b_3, b_2, b_1, b_0] = [1, 1, 0, 1, 1]$ and $\overrightarrow{WP}_{22} = \vec{W}_{27}$.

In Table 3.2, we can verify that the Gray code 22 corresponds to the natural binary code 27: $\overrightarrow{WP}_{22} = \vec{W}_{27}$.

In step 5, we have $\vec{G} = [g_4, g_3, g_2, g_1, g_0] = [1, 0, 1, 1, 0]$ and $\vec{G}^r = [g_0, g_1, g_2, g_3, g_4] = [0, 1, 1, 0, 1]$. In Table 3.2, we can verify that the Gray code 13 corresponds to the natural binary code 9: $\overrightarrow{WH}_{22} = \vec{W}_9$. Notice that $\overrightarrow{WP}_{22} \neq \overrightarrow{WH}_{22}$.

In step 6, we obtain Harmuth ordering. Since $[b_1, b_0] = [1, 0]$, we have $\vec{H}_{22} = \vec{1} \oplus \vec{W}_{22} = \vec{1} \oplus \vec{1} \oplus \vec{R}_4 \oplus \vec{R}_3 \oplus \vec{R}_2 \oplus \vec{R}_0 = \vec{R}_4 \oplus \vec{R}_3 \oplus \vec{R}_2 \oplus \vec{R}_0$. (End of example)

4 Hardware implementation

4.1 Architecture

In this paper, we defined five matrices for the Walsh functions, but implemented only four of them: we did not implement the Walsh-Rademacher functions in the whole Rademacher-Walsh matrix but only the primary Rademacher functions related to our algorithm as shown in Fig. 3.1. To implement all the Walsh-Rademacher functions based on definition 2.2, we need a very large LUT cascade. In the Walsh-Rademacher matrix, the adjacent rows tend to depend on different primary Rademacher functions, as shown in the properties. On the other hand, in the other Walsh matrices, i.e. Walsh-Hadamard matrix (WH), Walsh matrix in strict sequency (WS), Walsh-Paley matrix (WP), and Walsh-Harmuth matrix (WHR), the adjacent rows tend to depend on many common primary Rademacher functions. This is the reason why the Walsh function generator for four selected orderings by us has compact LUT cascade realisations.

For each ordering, there exist 2^n different Walsh function of n variables. When n is large, it is impractical to generate all the functions at the same time. So, we generate 2^{n-p} functions at a time by using the time multiplexing as shown in Fig. 4.2 and 4.3. It shows an implementation of a programmable Walsh function generator of order 64. It consists of the binary counter of six bits and the combinational part. The combinational part has six inputs from the counter, and p bits from the Select_Function inputs, two bits from the Select_Ordering inputs, and 2^{6-p}

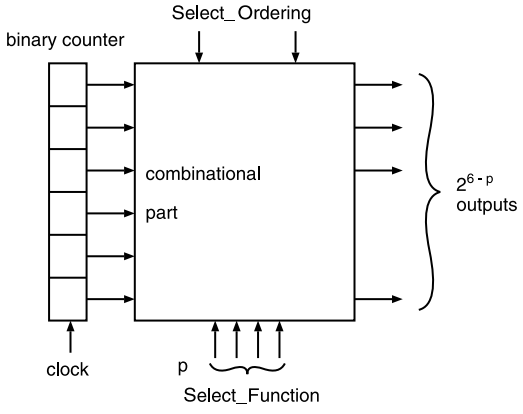

Fig. 4.2 *Programmable Walsh function generator for 64 outputs*

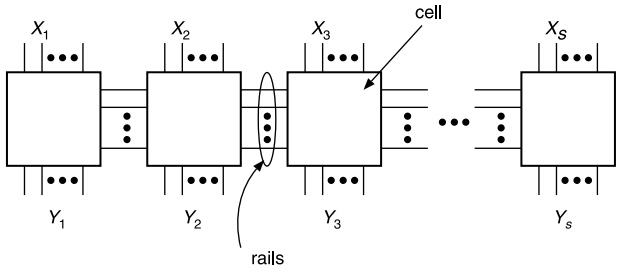

Fig. 4.3 *LUT cascade*

outputs. In the basic design (i.e. $p = 0$), the number of outputs is $2^6 = 64$. To reduce the number of outputs, we introduced p control signals to select 2^p different outputs. Thus, the combinational part has $p + 8$ inputs, and 2^{6-p} outputs. For the circuits with $p = 2, 3$ and 4, the numbers of outputs are 16, 8, and 4, respectively. The Select_Ordering inputs specify one of the four orderings: Hadamard, Harmuth, Paley and strict sequency.

In a similar manner, we can design other function generators with the binary counter of N-bits. In this case, the combinational part has $N + p + 2$ inputs and 2^{N-p} outputs. In the following subsections, we implemented the combinational part by LUT cascades and FPGAs. First, we defined Walsh functions $\vec{W}_0$ through $\vec{W}_{63}$ using algorithm 3.1 and then selected required sequence by multiplexers. Finally, we used additional multiplexers to select the required Walsh functions. To generate LUT cascades we described the circuit by our original HDL. To generate FPGAs, we converted the HDL description into Verilog HDL description by our tools.

4.2 LUT cascade implementation

An LUT cascade consists of cells implemented by memory. Unlike ordinary FPGAs, an LUT cascade uses cells with k inputs and many outputs, where k is often larger than 5. By using the algorithm in [39], we mapped the combinational part of the function generator into LUT cascades, where each cell has at most $k = 6$ inputs and 6 outputs.

For $p = 2$, we have a 10-input, 16-output circuit shown in Fig. 4.4*a*. It requires 20 cells and 91 LUT outputs. The total number of bits in the LUTs is $2^6 \times 91 = 5824$ bits.

For $p = 3$, we have an 11-input, 8-output circuit shown in Fig. 4.4*b*. It requires 21 cells and 77 LUT outputs. The total number of bits in the LUTs is $2^6 \times 77 = 4928$ bits.

For $p = 4$, we have a 12-input, 4-output circuit that shown in Fig. 4.4*c*. It requires 12 cells and 37 LUT outputs. The total number of bits in the LUTs is $2^6 \times 37 = 2368$ bits.

If the circuits were implemented by a single memory, then we need the following number of bits: For $p = 2$, $16 \times 2^{10} = 16384$ bits; for $p = 3$, $8 \times 2^{11} = 16384$ bits; and for $p = 4$, $4 \times 2^{12} = 16384$ bits. Thus, we need the same number of bits for all three cases. Note, that LUT cascade realisation requires many fewer bits than the single memory realisation.

Since LUT cascades have regular structures, they are easy to lay out, and the interconnection is simple. The delay of each cell in the LUT cascade is at most 4 ns [40], so the total delay is about 20 ns. Thus, the operating speed is about 50 MHz.

4.3 FPGA implementation

As for the target device, we used Altera FLEX10K EPF10K10LC84-3. It has 576 LE (logic elements) and 59 user I/O pins. In this case, we optimised the design by Sinplify Pro [41], and mapped by Quartus II [42]. Since each LE has a 4-input 1-output LUT, it requires 16 bits.

For $p = 2$, it required 122 LEs and the delay was 31.1 ns. The total number of bits used in the LEs is $122 \times 16 = 1952$ bits.

For $p = 3$, it required 102 LEs and the delay was 25.7 ns. The total number of bits used in the LEs is $102 \times 16 = 1632$ bits.

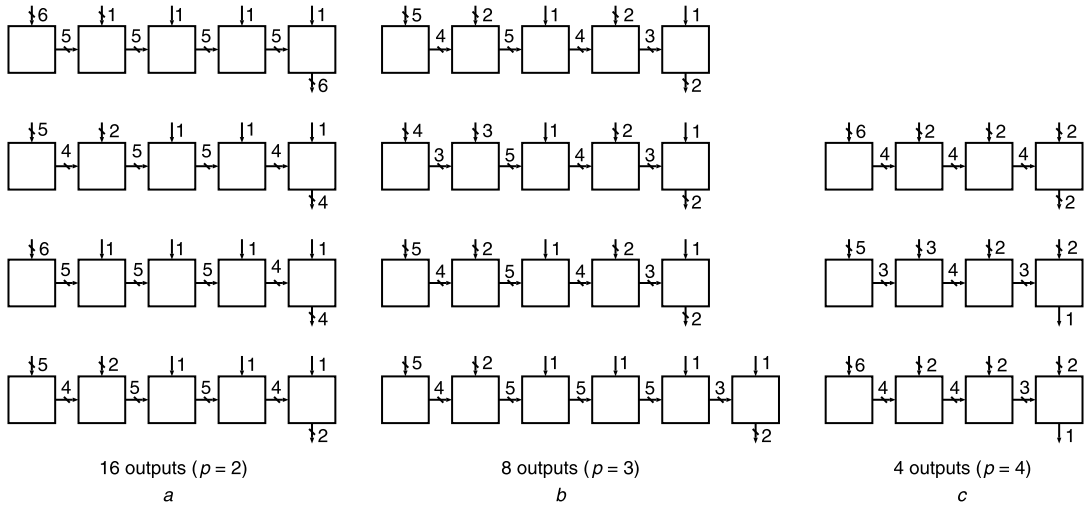

Fig. 4.4 *LUT cascades realisation*

For $p = 4$, it required 76 LEs and the delay was 30.5 ns. The total number of bits used in the LEs is $76 \times 16 = 1216$ bits.

In the FPGA design, we did not use any special technique to reduce the delay. Although the FPGA implementations require fewer bits in LUTs than LUT cascades, the interconnections in FPGAs are much more complex than in LUT cascades. Thus, we need a considerable amount of chip area for interconnections. The operating speed is between 30 to 40 MHz. These data show that a small FPGA is sufficient for this application.

5 Conclusion

In this paper, we have presented a unified algorithm to generate Walsh functions in four different orderings. Walsh functions in strict sequency can be generated directly from the primary Rademacher functions and their extensions, and from these Walsh functions remaining three orderings, Paley, Hadamard and Harmuth, are derived. Using these properties, we also presented the design method for a Walsh function generator. We designed the Walsh function generator of order 64 with LUT cascades and an FPGA, and estimated the amount of hardware and operating speed. We confirmed that it operates at the speed of 50 MHz when it is implemented by LUT cascades, and 30 to 40 MHz when it is implemented by FPGAs.

Various problems in digital system design and testing can be expressed as a sequence of operations on discrete functions or its corresponding spectra [9, 24]. A major advantage of the approach presented here is the convenience of the hardware implementation for the Walsh functions of order n in four most important orderings. However, the presented unified algorithm can be implemented not only in hardware but also in software. By using the presented algorithms 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 [1–8, 10, 13, 14, 18, 20, 24, 28, 30–33].

6 Acknowledgments

This work was supported by a grant from the Japanese Ministry of MEXT via the Kitakyushu Innovative Cluster Project, and the Aid for Scientific Research of the Japan Society for the Promotion of Science (JSPS).

7 References

1 Agaian, S.S., Astola, J.T., and Egiazarian, K.: 'Binary polynomial transforms and nonlinear digital filters' (Marcel Dekker, New York, 1995)
2 Ahmed, N., and Rao, K.R.: 'Orthogonal transforms for digital signal processing' (Springer-Verlag, Berlin, 1975)
3 Beauchamps, K.G.: 'Applications of Walsh functions and related functions with an introduction to sequency theory' (Academic Press, London, 1984)
4 Elliott, D.F., and Rao, K.R.: 'Fast transforms: algorithms, analysis, applications' (Academic Press, London, 1982)
5 Golubov, B., Efimov, A., and Skvortsov, V.: 'Theory and applications of Walsh series and transforms' (Kluwer Academic, Boston, 1991)
6 Harmuth, H.F.: 'Sequency theory foundations and applications' (Academic Press, New York, 1977)
7 Hurst, S.L.: 'The logical processing of digital signals' (Crane-Russak, New York, 1978)
8 Karpovsky, M.G.: 'Finite orthogonal series in design of digital devices' (Wiley, New York, 1976)
9 Sasao, T., and Fujita, M. (Eds.): 'Representation of discrete functions' (Kluwer Academic, Boston, 1996)
10 Sid-Ahmed, M.A.: 'Image processing: theory, algorithms, and architectures' (McGraw-Hill, New York, 1995)
11 Stankovic, R.S., Stoic, M.R., and Stankovic, M.S.: 'Recent developments in abstract harmonic analysis with applications in signal processing' (Nauka, Belgrade, Yugoslavia, 1996)
12 Stankovic, R.S., Stankovic, M., and Jankovic, D.: 'Spectral transforms in switching theory, definitions and calculations' (Nauka, Belgrade, Yugoslavia, 1998)
13 Yaroslavsky, L.P.: 'Digital picture processing an introduction' (Springer-Verlag, Berlin, 1985)
14 Yaroslavsky, L.P.: 'Digital holography and digital image processing: principles, methods, algorithms' (Kluwer Academic, Boston, 2003)
15 Falkowski, B.J., and Kannurao, S.: 'Algorithm to identify skew symmetries through Walsh transform', *Electron. Lett.*, 2000, **36**, (5), pp. 401–402
16 Falkowski, B.J., and Kannurao, S.: 'Analysis of disjoint decomposition of balanced Boolean functions through the Walsh spectrum', *IEE Proc. Comput. Digit. Tech.*, 2001, **148**, (2), pp. 71–78
17 Hansen, J., and Sekine, M.: 'Synthesis by spectral translation using Boolean decision diagrams'. Proc. 33rd IEEE/ACM Design Automation Conf., June 1996, pp. 248–253
18 Hurst, S.L.: 'Custom VLSI microelectronics' (Prentice Hall, London, 1993)
19 Iseno, A., Iguchi, Y., and Sasao, T.: 'Fault diagnosis for RAMs using Walsh spectrum', *IEICE Trans. Inf. Syst.*, 2004, **E87-D**, (3), pp. 601–608
20 Karpovsky, M.G. (Ed.): 'Spectral techniques and fault detection' (Academic Press, Orlando, FL, 1985)
21 Karpovsky, M.G., Stankovic, R.S., and Astola, J.T.: 'Reduction of sizes of decision diagrams by autocorrelation functions', *IEEE Trans. Comput.*, 2003, **52**, (5), pp. 592–606
22 Meinel, C., Somenzi, F., and Theobald, T.: 'Linear sifting of decision diagrams and its applications in synthesis', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2000, **19**, (5), pp. 521–533
23 Sasao, T.: 'Cascade realizations of two-input multiple-valued output functions using decomposition of group functions'. Proc. 33rd IEEE Int. Symp. on Multiple-valued Logic, Tokyo, Japan, May 2003, pp. 125–132
24 Stankovic, R.S., and Astola, J.T.: 'Spectral interpretation of decision diagrams' (Springer-Verlag, New York, 2003)
25 Falkowski, B.J., Schaefer, I., and Perkowski, M.A.: 'Effective computer methods for the calculation of Rademacher-Walsh spectrum for completely and incompletely specified Boolean functions', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 1992, **11**, (10), pp. 1207–1226
26 Thornton, M.A., and Nair, V.S.S.: 'Efficient calculation of spectral coefficients and their applications', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 1995, **14**, (11), pp. 1328–1341
27 Jankovic, D., Stankovic, R.S., and Drechsler, R.: 'Decision diagram method for calculation of pruned Walsh transform', *IEEE Trans. Comput.*, 2001, **50**, (2), pp. 147–157
28 Amira, A., Bouridane, A., Milligan, P., and Roula, M.: 'Novel FPGA implementations of Walsh-Hadamard transforms for signal processing'. *IEE Proc. Vis., Image Signal Process.*, 2001, Vol. 148(6), pp. 377–383
29 Falkowski, B.J., and Sasao, T.: 'Implementation of Walsh function generator of order 64 using LUT cascades'. Proc. 47th IEEE Midwest Symp. on Circuits and Systems, Hiroshima, Japan, July 2004, **3**, pp. 447–450
30 Jaehne, B.: 'Digital image processing' (Springer-Verlag, Berlin, 2002, 5th revised and extended edn.)
31 Lee, J.S., and Miller, L.E.: 'CDMA systems engineering handbook' (Artech House, Boston, MA, 1998)
32 Meyer-Baese, U.: 'Digital signal processing with field programmable gate arrays' (Springer-Verlag, Berlin, 2001)
33 Parhi, K.K.: 'VLSI digital signal processing systems design and implementation' (John Wiley, New York, 1999)
34 Schneier, B.: 'Applied cryptography, protocols, algorithms and source code in C' (John Wiley, New York, 1996)
35 Falkowski, B.J.: 'Recursive relationships, fast transforms, generalisations and VLSI iterative architecture for Gray code ordered Walsh functions', *IEE Proc. Comput. Digit. Tech.*, 1995, **142**, (5), pp. 325–331
36 Falkowski, B.J., and Rahardja, S.: 'Walsh-like functions and their relations', *IEE Proc. Vis. Image Signal Process.*, 1996, **143**, (5), pp. 279–284
37 Lee, Z.H., and Zhang, Q.S.: 'Ordering of Walsh functions', *IEEE Trans. Electromagn. Compat.*, 1983, **25**, pp. 115–119
38 Sasao, T., Matsuura, M., and Iguchi, Y.: 'A cascade realization of multiple-output function for reconfigurable hardware'. Int. Workshop on Logic Synthesis, Lake Tahoe, CA, USA, June 2001, pp. 225–230
39 Sasao, T., and Matsuura, M.: 'A method to decompose multiple-output logic functions'. Proc. 41st IEEE/ACM Design Automation Conf., June 2004, pp. 428–433
40 Qin, H., Sasao, T., Matsuura, M., Nagayama, S., Nakamura, K., and Iguchi, Y.: 'A realization of multiple-output functions by look-up table rings', *IEICE Trans. Fundam.*, December 2004, **E87-A**, pp. 3141–3150
41 Sinplify: http://www.sinplicity.com/
42 Altera: http://www.altera.com/