Short Notes

Computation of the Fast Walsh–Fourier Transform

JOHN L. SHANKS, MEMBER, IEEE

Abstract—**The discrete, orthogonal Walsh functions can be generated by a multiplicative iteration equation. Using this iteration equation, an efficient Walsh transform computation algorithm is derived which is analogous to the Cooley-Tukey algorithm for the complex-exponential Fourier transform.**

Index Terms—**Algorithm, Cooley-Tukey, Hadamard-Fourier, orthogonal, transform, Walsh-Fourier.**

Optimization of the Discrete Walsh Transform

Several authors have treated the set of Walsh functions and the Walsh–Fourier transform [1]–[4], [7]. We shall discuss here the set of discrete, orthogonal Walsh functions and the derivation of a Walsh transform algorithm which is analogous to the Cooley–Tukey algorithm [5] for the complex-exponential Fourier transform.[1]

The discrete Walsh functions are sampled versions of the continuous set. As discussed here, the discrete functions are assumed to be infinite in extent, and are periodic with period M, where M is an integral power of two. Thus a complete orthogonal set will have M distinct functions. We shall designate these functions wal (m, n). The complete set is represented over the range $m=0, 1, 2, \cdots, M-1$ and $n=0, 1, 2, \cdots, M-1$.

The first two discrete Walsh functions are defined as

$$\text{wal } (0, n) = 1 \qquad \text{for } n = 0, 1, 2, \cdots, M-1 \tag{1}$$

$$\left.\begin{aligned} &\text{wal } (1, n) \\ &= 1 \quad \text{for } n = 0, 1, 2, \cdots, (M/2)-1 \\ &= -1 \quad \text{for } n = M/2, (M/2)+1, \cdots, M-1 \end{aligned}\right\}. \tag{2}$$

The remainder of the set can be generated by an iterative equation. Various iteration equations have been used to generate the Walsh functions, but for our development it is convenient to introduce the multiplicative iterative equation (3):

$$\text{wal } (m, n) = \text{wal } ([m/2], 2n) \cdot \text{wal } (m - 2[m/2], n), \tag{3}$$

where $[m/2]$ indicates the integer part of $m/2$. It is shown in the Appendix that (1), (2), and (3) generate an orthogonal set.

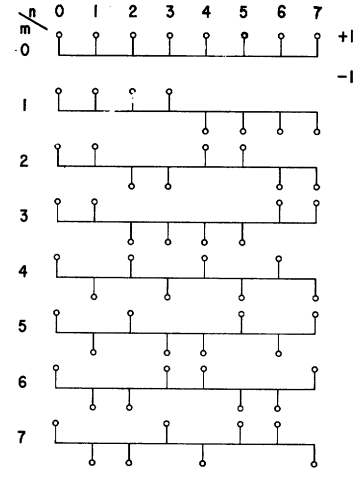

Fig. 1. The eight discrete Walsh functions of length 8.

Fig. 1 shows the eight discrete Walsh functions of length $M=8$ as generated by (1), (2), and (3). Note that the discrete Walsh functions as defined here are symmetric with respect to the argument (m, n). That is,

$$\text{wal } (m, n) = \text{wal } (n, m).$$

Given an M-length real array $f(n)$, we can define the Walsh transform as

$$F(m) = \sum_{n=0}^{M-1} f(n) \text{ wal } (m, n), \qquad m = 0, 1, 2, \cdots, M-1. \tag{4}$$

Similarly, the inverse transform is

$$f(n) = \frac{1}{M} \sum_{m=0}^{M-1} F(m) \text{ wal } (n, m), \qquad n = 0, 1, 2, \cdots, M-1. \tag{5}$$

Since the Walsh functions can have values of $+1$ and -1 only, computation of (4) or (5) requires no multiplications. Furthermore, using (3) we can derive a computation algorithm analogous to the Cooley–Tukey algorithm. This algorithm will require $M \log_2 M$ summations to compute a complete Walsh transform rather than M^2 as indicated by (4).

The derivation of the algorithm parallels the one given by Cooley and Tukey. Rather than present the general proof, we shall outline the derivation for the special case $M=8$. Anyone familiar with the derivation of the Cooley–Tukey algorithm should be able to extend the work presented here to the general case.

Manuscript received November 1, 1968; revised February 6, 1969.
The author is with the Pan American Petroleum Corporation, Tulsa, Okla. 74102.

[1] At EASTCON, 1968, and at the IEEE Workshop of FFT Processing, October 6–8, 1968, J. E. Whelchel [9] presented the matrix derivation of the fast Fourier–Hadamard transform. The Hadamard [8], [10] and the discrete Walsh transforms are identical with the possible exception of a permutation in the order of functions.

Let us replace the indices in (4) with a set which can have only values of 0 and 1. That is, let

$$m = 4j_2 + 2j_1 + j_0, \qquad j_2, j_1, j_0 = 0 \text{ or } 1, \quad (6)$$

$$n = 4k_2 + 2k_1 + k_0, \qquad k_2, k_1, k_0 = 0 \text{ or } 1. \quad (7)$$

Using these new indices, the notation wal (m, n) becomes

$$\text{wal}\,(j_2, j_1, j_0; k_2, k_1, k_0).$$

Thus we rewrite (4) as

$$F(j_2, j_1, j_0) = \sum_{k_0=0}^{1}\sum_{k_1=0}^{1}\sum_{k_2=0}^{1} f(k_2, k_1, k_0) \cdot \text{wal}\,(j_2, j_1, j_0; k_2, k_1, k_0). \quad (8)$$

Also, we can restate the iteration equation (3) using the new indices. Note that the numbers j_2, j_1, j_0 are essentially the digits of the binary representation of m. Therefore, dividing m by two and keeping the integer part is equivalent to shifting the binary representation of m one bit to the right and dropping the fractional bit. That is, if

$$m \leftrightarrow j_2 j_1 j_0$$

then

$$[m/2] \leftrightarrow 0 j_2 j_1.$$

Similarly, multiplying n by two is equivalent to shifting the binary representation of n one bit to the left. If

$$n \leftrightarrow k_2 k_1 k_0$$

then

$$2n \leftrightarrow k_2 k_1 k_0 0.$$

An 8-length Walsh function is periodic with period 8. Thus any index (such as $2n$) can be evaluated modulo 8. This is equivalent to deleting any bits above the third bit and we have

$$2n(\text{modulo } 8) \leftrightarrow k_1 k_0 0.$$

Using these indices, (3) becomes

$$\text{wal}\,(j_2, j_1, j_0; k_2, k_1, k_0) = \text{wal}\,(0, j_2, j_1; k_1, k_0, 0) \cdot \text{wal}\,(0, 0, j_0; k_2, k_1, k_0). \quad (9)$$

Since j_0 can only be 0 or 1, wal $(0, 0, j_0; k_2, k_1, k_0)$ represents either wal $(0, n)$ or wal $(1, n)$. The function wal $(0, n)$ is 1.0 for all n. The function wal $(1, n)$ is 1.0 if $0 \leq n < 4$ and wal $(1, n)$ is -1.0 if $4 \leq n \leq 7$. Thus, from (7), wal $(1, n)$ is $+1.0$ if $k_2 = 0$ and wal $(1, n)$ is -1.0 if $k_2 = 1$. Therefore

$$\text{wal}\,(0, 0, j_0; k_2, k_1, k_0) = (-1)^{j_0 k_2}. \quad (10)$$

We can use (9) to factor wal $(0, j_2, j_1; k_1, k_0, 0)$ to get

$$\text{wal}\,(0, j_2, j_1; k_1, k_0, 0) = \text{wal}\,(0, 0, j_2; k_0, 0, 0) \cdot \text{wal}\,(0, 0, j_1; k_1, k_0, 0). \quad (11)$$

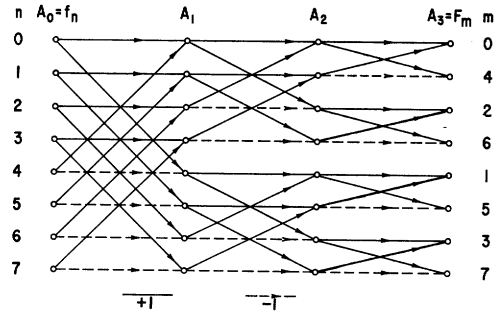

Fig. 2. Signal flow graph of 8-length discrete Walsh transforms. Multiplying factors are $+1$ and -1, as indicated by the solid and dotted branches, respectively.

Using (10) and (11) back in (9), we get the completely factored expression for the 8-length Walsh function

$$\text{wal}(j_2, j_1, j_0; k_2, k_1, k_0) = (-1)^{j_2 k_0}(-1)^{j_1 k_1}(-1)^{j_0 k_2}. \quad (12)$$

Using (12), (8) becomes

$$F(j_2, j_1, j_0) = \sum_{k_0=0}^{1}(-1)^{j_2 k_0}\sum_{k_1=0}^{1}(-1)^{j_1 k_1} \cdot \sum_{k_2=0}^{1}(-1)^{j_0 k_2} f(k_2, k_1, k_0). \quad (13)$$

We can define

$$A_1(j_0, k_1, k_0) = \sum_{k_2=0}^{1}(-1)^{j_0 k_2} f(k_2, k_1, k_0). \quad (14)$$

This A_1 array becomes the first result in a set of calculations to compute $F(m)$. From A_1 we can compute an $A_2(j_0, j_1, k_0)$ array and so forth until the last sum (over k_0) has been evaluated. The final result $A_3(j_0, j_1, j_2)$ will contain all the coefficients of the 8-length Walsh transform. However, just as in the Cooley–Tukey algorithm, the order of the values will be in bit-reversed form.

Fig. 2 shows the signal flow graph for the 8-length Walsh transform. Its similarity to the Cooley–Tukey algorithm is obvious [6]. In fact, one can convert some Cooley–Tukey transform programs to the Walsh transform by setting all the trig values to 1.0, or by removing the steps which multiply the array values by the trig values. Also, the complex part of the Cooley–Tukey operations can be removed since only the Walsh transform is real. Since wal $(n, m) = \text{wal}\,(m, n)$, the inverse Walsh transform is identical to the Walsh transform except that all the values are divided by M.

For the general case when $M = 2^p$, the intermediate Walsh transform arrays are defined by

$$A_l(j_0, j_1, \cdots, j_{l-1}, k_{p-l-1}, \cdots, k_0) = \sum_{k_{p-l}=0}^{1} A_{l-1}(j_0, j_1, \cdots, j_{l-2}, k_{p-l}, k_{p-l-1}, \cdots, k_0) \cdot (-1)^{j_{l-1} k_{p-l}}, \quad (15)$$

where $l=1, 2, \cdots, p$, and

$$A_0(k_{p-1}, k_{p-2}, \cdots, k_0) = f(k_{p-1}, k_{p-2}, \cdots, k_0). \quad (16)$$

The general equation for the $M=2^p$-length discrete Walsh function is

$$\text{wal}\,(j_{p-1}, j_{p-2}, \cdots, j_0; k_{p-1}, k_{p-2}, \cdots, k_0) = \prod_{i=0}^{p-1} (-1)^{j_{p-1-i}k_i}. \quad (17)$$

From (15) we can build a program for the general M-length discrete Walsh transform. With (17) we can evaluate a Walsh function for any m and n. Also, (17) can be used to prove the orthogonality (see Appendix) and the symmetry of the set of discrete Walsh functions.

Appendix

The orthogonality of the discrete Walsh functions as generated by (1), (2), and (3) can be proved by using the general Walsh function (17). Equation (17) is derived from the iterative equation (3) and the definitions (1) and (2). The dot product of any two M-length Walsh functions is as follows.

$$\phi(r_{p-1}, r_{p-2}, \cdots, r_0; j_{p-1}, j_{p-2}, \cdots, j_0)$$

$$= \sum_{k_{p-1}} \sum_{k_{p-2}} \cdots \sum_{k_0=0}^{1} \quad (18)$$

$$\cdot \text{wal}\,(r_{p-1}, \cdots, r_0; k_{p-1}, \cdots, k_0)$$

$$\cdot \text{wal}\,(j_{p-1}, \cdots, j_0; k_{p-1}, \cdots, k_0) \quad (19)$$

$$= \sum_{k_{p-1}} \sum_{k_{p-2}} \cdots \sum_{k_0=0}^{1} \prod_{i=0}^{p-1} \left\{(-1)^{r_{p-1-i}k_i}(-1)^{j_{p-1-i}k_i}\right\} \quad (20)$$

$$= \sum_{k_{p-1}} \sum_{k_{p-2}} \cdots \sum_{k_0=0}^{1} \prod_{i=0}^{p-1} (-1)^{k_i(r_{p-1-i}+j_{p-1-i})} \quad (21)$$

$$= \prod_{i=0}^{p-1} \sum_{k_i=0}^{1} (-1)^{k_i(r_{p-1-i}+j_{p-1-i})} \quad (22)$$

$$= \prod_{i=0}^{p-1} \left\{1 + (-1)^{(r_{p-1-i}+j_{p-1-i})}\right\}. \quad (23)$$

Suppose each $r_k=j_k=0$ or 1. Then (23) becomes

$$\phi = \prod_{i=0}^{p-1} 2 = 2^p = M.$$

Suppose at least one $r_k \neq j_k$. Then at least one term in the product in (23) is 0 and $\phi=0$. In terms of the decimal indices, this means that

$$\phi(m, l) = \sum_{n=0}^{M-1} \text{wal}\,(m, n)\cdot\text{wal}\,(l, n) = \begin{cases} M & \text{for } m = l \\ 0 & \text{for } m \neq l \end{cases} \quad (24)$$

Thus (1), (2), and (3) generate an orthogonal set.

References

[1] J. L. Walsh, "A closed set of orthogonal functions," *Am J. Math.*, vol. 55, pp. 5–24, January 1923.
[2] J. L. Hammond and R. S. Johnson, "Orthogonal square wave functions," *J. Franklin Inst.*, vol. 273, pp. 211–225, March 1962.
[3] K. W. Henderson, "Some notes on the Walsh functions," *IEEE Trans. Electronic Computers* (Correspondence) vol. EC-13, pp. 50–52, February 1964.
[4] H. F. Harmuth, "A generalized concept of frequency and some applications," *IEEE Trans. Information Theory*, vol. IT-14, pp. 375–382, May 1968.
[5] J. W. Cooley, and J. W. Tukey, "An algorithm for the machine computation of complex Fourier series," *Math. of Computation*, vol. 19, pp. 297–301, April 1965.
[6] G-AEC Subcommittee on Measurement Concepts, "What is the Fast Fourier transform?" *Proc. IEEE*, vol. 55, pp. 1664–1674, October 1967.
[7] N. J. Fine, "The generalized Walsh functions," *Trans. Amer. Math. Soc.*, vol. 69, pp. 66–77, 1950.
[8] S. W. Golomb and L. D. Baumert, "The search for Hadamard matrices," *Am. Math. Monthly*, vol. 70, p. 12, January 1963.
[9] J. E. Whelchel and D. F. Guinn, "The fast Fourier–Hadamard transform and its use in signal representation and classification," Melpar, Inc., Fall's Church, Va., Tech. Rept. PRC 68-11, 1968.
[10] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proc. IEEE*, vol. 57, pp. 58–68, January 1969.

Threshold Logic Design of Pulse-Type Sequential Networks

RAYMOND M. KLINE, MEMBER, IEEE, AND DONALD F. WANN, MEMBER, IEEE

***Abstract*—A modification of the conventional design method for pulse-type sequential logic is developed in which threshold gates are used in all of the combinational portions of the circuit. The canonical configuration, memory element employed, tabular design aids, etc., found in the conventional method were changed as necessary in order to use more effectively the unique properties possessed by threshold elements. A theorem providing insight for threshold gate realization is given and a classification of sequential functions is discussed in order to facilitate the understanding and use of the new process. Finally, examples are presented showing a significant advantage of the present method in both gate and transistor economy over conventional design.**

***Index Terms*—Logical design, realizability, sequence classification, sequential networks, switching circuit synthesis, threshold logic.**

I. Introduction

Results of extensive research have been reported concerning the application of threshold logic principles in the design of combinatorial logic circuits [1]–[5]. In these papers, attention has been focused on general properties of threshold functions, on techniques for determining whether a given Boolean function is realizable by means of a single gate, and on various applications such as the development of learning

Manuscript received October 25, 1968; revised November 18, 1968.

The authors are with the Department of Electrical Engineering, School of Engineering and Applied Science, Washington University, St. Louis, Mo. 63130.