社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Walsh 係数を計算するハードウェアについて

井口　幸洋†　　笹尾　　勤††

† 明治大学理工学部情報科学科，神奈川県
†† 九州工業大学情報工学部電子情報工学科，福岡県

あらまし　論理関数の Walsh スペクトラムの係数の一部をハードウェアを用いて計算する方法を示す．まず, Walsh 変換木を定義し, Walsh 変換木から Walsh 係数を計算する方法を示す．次に, この操作をハードウェアで実現する方法を示す．1 個の係数を計算するハードウェア量は, $O(2^n)$ であり, 全ての係数を求めるハードウェア量は $O(n^2 \cdot 2^n)$ である．FPGA デバイス対して回路設計を行った結果, 市販 FPGA 上で計算可能な論理関数の変数の個数 n は 14 であった．また, マイクロプロセッサに比べて FPGA 実現は $n = 14$ のとき 1253 倍高速である．

キーワード　Walsh 変換, スペクトラム解析, FPGA, 二分決定グラフ, 論理関数.

Hardware to Compute Walsh Coefficients

Yukihiro IGUCHI† and Tsutomu SASAO††

† Department of Computer Science, Meiji University, Kawasaki, 214-8571 Japan
†† Dept. of Electronics and Computer Science, Kyushu Institute of Technology, Iizuka, 820-8502 Japan

Abstract　This paper presents a method to compute a fragment of the Walsh coefficients of logic functions using hardware. First, it introduces the Walsh transformation tree, and shows a method to compute Walsh coefficients using the Walsh transformation tree. Next, it shows the hardware realization for the Walsh tree. The amount of hardware to compute a coefficient and the entire coefficients are $O(2^n)$ and $O(n^2 \cdot 2^n)$, respectively. FPGA implementations show their feasibility up to $n = 14$. The FPGA realization is at least 1253 times faster than a software implementation on a microprocessor for $n = 14$.

Key words　Walsh transformation, spectrum analysis, FPGA, BDD, logic function.

1. はじめに

論理関数のスペクトラム解析 [13] は, 論理合成 [8], [10], [20], ブーリアンマッチング [5], [7], テスト [9], [11], [17], 検証 [18] などに応用されている．スペクトラムの計算法としては, 高速フーリエ変換 (FFT) に基づく方法 [3], キューブを用いた方法 [6], [20], 決定グラフを用いた方法 [5], [7] が知られている．

スペクトラル法の問題点として, スペクトラムを全て同時に表現すると表現が大きくなり過ぎることがあげられる．多くの応用では, 2^n 個のスペクトル係数のうち, 一部の係数の値が得られれば十分である．計算をスペクトル係数の一部に限定すれば, 必要な計算時間を削減できる．今までの殆どの研究は, スペクトラムを計算するソフトウエアに関するものである．特に, [7] や [12] では, 与えられた論理関数の Walsh スペクトラムの一部を計算する効率的な方法を考察している．

本論文では, これらとは異なり Walsh スペクトラムの一部をハードウェアで計算する方法について考察する．理論的には, FFT をそのまま率直に実現するハードウェアにより Walsh スペクトラムを全て同時に計算できる．しかし, 現実問題として, FFT を率直に実現するとハードウェアは, 大き過ぎて一つの FPGA 上には実装できない．そこで, 文献 [2] では, ビットシリアルに計算する方法を提案している．これはデジタル信号処理用のハードウェアであり, 設計条件は以下の通りである:

（1）　全てのスペクトラムを同時に計算する．
（2）　各入力は, 8 ビット (程度) の信号を考えている．

本論文では, Walsh スペクトラムの一部を計算するハードウェアについて考察する．ここでの設計条件は, 以下の通りである:

（1）　スペクトラムの一部を同時に計算する．
（2）　各入力は 1 ビット信号を扱う（単一出力論理関数のスペクトラムを計算する．　多出力関数への拡張は 4 章で述べる）．

このようなハードウェアは, 半導体メモリの故障診断 [11] やブーリアンマッチング [5] に適用できる．

2. 諸定義ならびに基本的性質

ここでは, まず Walsh スペクトラム, Walsh 変換木, Walsh 変

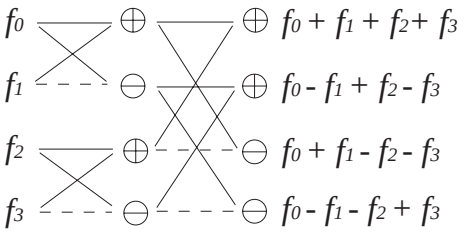


図 2.1　高速フーリエ変換 (バタフライ演算).

換グラフの定義について述べる. 次に Walsh 変換木と Walsh 変換グラフとから Walsh スペクトラムの係数を計算する方法 [15] を紹介する.

2.1　Walsh 変換

［定義 2.1］ (Walsh 変換行列)

$$\mathbf{W}(n) = \begin{bmatrix} \mathbf{W}(n-1) & \mathbf{W}(n-1) \\ \mathbf{W}(n-1) & -\mathbf{W}(n-1) \end{bmatrix},$$

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

とするとき, $\mathbf{W}(n)$ を n **変数の Walsh 変換行列**といい, $\mathbf{W}(1)$ を**基本 Walsh 変換行列**という.

Walsh 変換行列 $\mathbf{W}(n)$ の逆行列は, $2^{-n}\mathbf{W}(n)$ である.

［定義 2.2］ n 変数論理関数 f の**真理値ベクトル**を $\vec{F} = (f_0, f_1, \ldots, f_{2^n-1})$, f の **Walsh スペクトラム**を $\vec{S} = (s_0, s_1, \ldots, s_{2^n-1})$ とすると, $\vec{S} = 2^{-n}\mathbf{W}(n)\vec{F}^t$, $\vec{F} = \mathbf{W}(n)\vec{S}^t$ が成立する. ここで, t はベクトルの転置を表している. s_i $(i = 0, 1, \ldots, 2^n - 1)$ を f の **Walsh 係数**という. この変換は **Walsh-Hadamard 変換**あるいは, **Hadamard 変換**と呼ばれることもあるが, 本論文では, **Walsh 変換**で統一する. Walsh 変換行列の各行は, **Walsh 関数**を表す.

論理関数のスペクトラムの計算では, 定数項 2^{-n} の乗算を省略することが多い. 本論文でも定数項は省略する. n 変数論理関数の Walsh 変換は, $\mathbf{W}(n)$ と $\vec{F}^t$ の積で計算できる. ただし, 率直な計算方法では行列が大きくなりすぎ, 計算量も多い. 図 2.1 のような高速フーリエ変換 (FFT 法) により, Walsh 変換を効率よく計算できる. 図 2.1 の計算方法はバタフライ演算と呼ばれる.

［例 2.1］ $f = \bar{x}_1 \vee x_2$ の真理値ベクトルは, $\vec{F} = (f_0, f_1, f_2, f_3) = (1, 1, 0, 1)$,

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \\ 1 \\ 1 \end{bmatrix}.$$

であるから, Walsh スペクトラムは $\vec{S} = (s_0, s_1, s_2, s_3) = (3, -1, 1, 1)$ となる. (例終り)

2.2　Walsh 変換木

［定義 2.3］ **二分決定木** (binary decision tree : BDT) は, 与えられた論理関数 f にシャノン展開 $f = \bar{x}f_0 \vee xf_1$ を繰り返し適用して得られる木である. この木では, 0 枝にラベル $\bar{x}_i$, 1 枝にラベル x_i が付いている. S と書かれた各節はシャノン展開を表している.

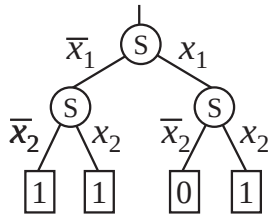


図 2.2　$f = \bar{x}_1 \vee x_2$ の二分決定木.

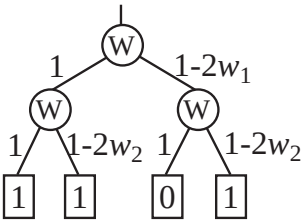


図 2.3　$f = \bar{x}_1 \vee x_2$ の Walsh 変換木.

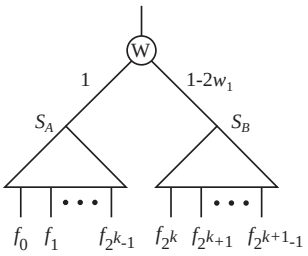


図 2.4　$k+1$ 変数の Walsh 変換木.

［定義 2.4］ BDT の根から終端節点への道において, 道の枝の全てのラベルと終端節点の値の積を**パス積**という. BDT の根から, 全ての終端節点への道のパス積 (path-of-product) の和を**パス積和 (sum-of-path-products: SOPP)** という.

f に対する BDT の場合, SOPP は f の論理和標準形に相当する.

［例 2.2］ 図 2.2 に $n = 2$ の BDT の例を示す. SOPP は

$$1 \cdot \bar{x}_1\bar{x}_2 + 1 \cdot \bar{x}_1 x_2 + 0 \cdot x_1\bar{x}_2 + 1 \cdot x_1 x_2 = \bar{x}_1\bar{x}_2 + \bar{x}_1 x_2 + x_1 x_2.$$

である. (例終り)

［定義 2.5］ **多端子二分決定木** (multi-terminal binary decision tree : **MTBDT**) は, 二分決定木の葉の部分を m ビット $(m \geqq 2)$ にしたものである. MTBDT は多出力論理関数 $(f^0, f^1, \ldots, f^{m-1})$ を表現し, その終端節点の各ビットが各関数 $f^i (i = 0, 1, .., m-1)$ を表す.

［定義 2.6］ [15] **Walsh 変換木** (Walsh transform tree: **WTT**) は, 二分決定木 (BDT) において 0 枝にラベル 1 を, 1 枝にラベル $(1 - 2w_i)$ を付加したものである. W と書かれた各節は, Walsh 展開を表している.

図 2.3 に $n = 2$ の場合の WTT の例を示す.

［定義 2.7］ WTT の SOPP を **Walsh 式**という. 関数 f を表現する BDT の葉の値を $f_0, f_1, f_2, \ldots f_{2^n-1}$ としたとき,

$$s(f : w_1, w_2, \ldots, w_n)$$

$$
\begin{aligned}
&= f_0 + f_1 \cdot (1-2w_n) + f_2 \cdot (1-2w_{n-1}) \\
&\quad + f_3 \cdot (1-2w_{n-1}) \cdot (1-2w_n) + \cdots \\
&\quad + f_{2^n-1} \cdot (1-2w_1) \cdot (1-2w_2) \cdot \cdots \cdot (1-2w_n).
\end{aligned}
$$

が f の Walsh 式となる.

Walsh 式で $\vec{w} = (w_1, w_2, \ldots, w_n)$ の値を指定すると, 任意の Walsh 係数を計算できる. つまり, WTT は $\mathbf{W}(n)$ の行を表し, 値 $(w_1, w_2, \ldots, w_n)$ は行を指定している. $w_1 = w_2 = \ldots = w_n = 0$ のとき, SOPP は, $f_0 + f_1 + \ldots + f_{2^n-1} = s_0$ を表す. これは, 論理関数の真理値ベクトルと Walsh 変換行列の第 1 行目との内積を計算したことに対応する. また, $w_1 = w_2 = \ldots = 0, w_n = 1$ のとき, SOPP は, $(f_0 - f_1) + (f_2 - f_3) + \ldots + (f_{2^n-2} - f_{2^n-1}) = s_1$ を表す. これは, 論理関数の真理値表ベクトルと Walsh 変換行列の第 2 行目との内積を計算したことに対応する. 以下同様に, $\vec{w} = (w_1, w_2, \ldots, w_n)$ の値を変えることにより, Walsh 変換行列 $\mathbf{W}(n)$ の任意の行と $\vec{F}$ との内積を計算できる.

［例 2.3］ 図 2.3 の Walsh 変換木から得られる Walsh 式は

$$
\begin{aligned}
s(w_1, w_2) &= f_0 \cdot 1 \cdot 1 + f_1 \cdot 1 \cdot (1-2w_2) + f_2 \\
&\quad \cdot (1-2w_1) \cdot 1 + f_3 \cdot (1-2w_1) \cdot (1-2w_2) \\
&= 1 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot (1-2w_2) + 0 \\
&\quad \cdot (1-2w_1) \cdot 1 + 1 \cdot (1-2w_1) \cdot (1-2w_2) \\
&= 3 - 2w_1 - 4w_2 + 4w_1 w_2.
\end{aligned}
$$

となる. これより, Walsh 係数は次のように計算できる.

$$
\begin{aligned}
s(0,0) &= f_0 + f_1 + f_2 + f_3 = 3 \\
s(0,1) &= f_0 - f_1 + f_2 - f_3 = -1 \\
s(1,0) &= f_0 + f_1 - f_2 - f_3 = 1 \\
s(1,1) &= f_0 - f_1 - f_2 + f_3 = 1
\end{aligned}
$$

(例終り)

［定理 2.1］ n 変数の Walsh 変換木 (WTT) において, その葉が論理関数 f の真理値を表すとする. このとき, SOPP が $\vec{w}(w_1, w_2, \ldots, w_n)$ で指定された Walsh 係数を表現する.

(証明) f の変数の個数 n に関する数学的帰納法を用いる.

（1） $n = 1$ のとき, 明らかに定理が成立する. 即ち, $w_1 = 0$ のとき, WTT の SOPP は $f_0 + f_1$ を表現し, $w_1 = 1$ のとき, WTT の SOPP は $f_0 - f_1$ を表現する.

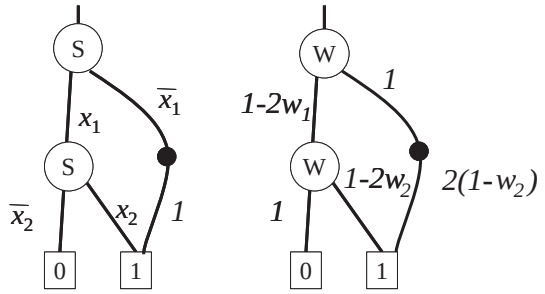
（2） $n = k$ のとき成立すると仮定する.

（3） 次に $n = k + 1$ の場合を考える. このとき, WTT は図 2.4 のように表現できる. $w_1 = 0$ のとき, WTT の SOPP は $s_A(w_2, w_3, \ldots, w_{k+1}) + s_B(w_2, w_3, \ldots, w_{k+1})$ を表現する. また, $w_1 = 1$ のとき, WTT の SOPP は $s_A(w_2, w_3, \ldots, w_{k+1}) - s_B(w_2, w_3, \ldots, w_{k+1})$ を表現する. これは, WTT が

$$
\mathbf{W}(k+1) = \begin{bmatrix} \mathbf{W}(k) & \mathbf{W}(k) \\ \mathbf{W}(k) & -\mathbf{W}(k) \end{bmatrix}
$$

を表現していることを示す. これより, $n = k + 1$ の場合も定理が成立することが明らかとなる.

(1), (2), (3) より, 全ての自然数 n で定理が成立する. (証明終)



(a)BDD (b)WTD

図 2.5 $f = \bar{x}_1 \vee x_2$ の BDD と WTD.

2.3 Walsh 変換グラフ

［定義 2.8］ [4] **二分決定グラフ (BDD)** とは, BDT において,

（1） BDT のある節点 v が同じ子供につながっている場合, 節点 v を除去する.

（2） 二つの節点 v_1, v_2 が同じ関数を表現するときには, いずれか一方の節点を除去する.

という操作を可能な限り繰り返し行ない, 節点を削減したものである.

［定義 2.9］ **Walsh 変換グラフ (WTD)** [15] とは, WTT において,

（1） WTT の w_i に対応する節点 v が同じ子供につながっている場合, 節点 v を除去する. ただし, 子供につながる枝のラベルを $2(1 - w_i)$ とする.

（2） 二つの節点 v_1, v_2 が同じ関数を表現するときには, いずれか一方の節点を除去する.

という操作を可能な限り繰り返し行い, 節点を削減したものである.

BDD で節点 v の子供が同じ場合, 縮約したグラフのラベルは, $\bar{w}_i + w_i = 1$ となる. 一方, WTD で節点 v の子供が同じ場合, 縮約したグラフのラベルは, $1 + (1 - 2w_i) = 2(1 - w_i)$ となる.

［定理 2.2］ n 変数の Walsh 変換グラフ (WTD) において, 終端節点が論理関数 f の真理値を表すとする. このとき, WTD の SOPP が Walsh 式を表現する.

［例 2.4］ 図 2.5(a) の二分決定グラフで, x_2 に対応する節点が除去されている. これを**クロスポイント** [15] といい, 図では • で表わす. これに対して, 図 2.5(b) のような WTD を考える. Walsh 式は

$$
\begin{aligned}
s(w_1, w_2) &= 1 \cdot 1 \cdot 2(1-w_2) + 0 \cdot (1-2w_1) \\
&\quad \cdot 1 + 1 \cdot (1-2w_1) \cdot (1-2w_2) \\
&= 3 - 2w_1 - 4w_2 + 4w_1 w_2.
\end{aligned}
$$

となる. 本式は, 例 2.3 の $s(w_1, w_2)$ に等しいことに注意. これより, Walsh 係数を計算できる. (例終り)

3. ハードウェア量

3.1 Walsh 係数を一個だけ計算する場合

WTT の各非終端節点を加減算器で置換すると, Walsh 係数

を計算する回路が得られる．k ビット加減算器は，$y(w, s_a, s_b) = s_a + (1-2w)s_b$ を計算する．ここで s_a と s_b は k ビットの二進数とする．$y(w, s_a, s_b)$ は，$w = 0$ の時，加算 $s_a + s_b$ を表し，$w = 1$ の時，減算 $s_a - s_b$ を表す．ここで，w は制御入力である．k ビットの加減算器の入力数は $(2k+1)$，出力数は $(k+1)$ である．k ビットの加減算器のハードウェアのコストを αk と仮定する，ここで α は定数とする．

WTT を実現するハードウェアは，2進木構造をしている．ただし根に近づくにつれて節点の加減算器のコストは増える．しかし，全体のコストは丁度 $O(2^n)$ となることを証明できる．

［補題 3.1］ $\sum_{i=1}^{n} i2^{i-1} = (n-1)2^n + 1$

［定理 3.1］ n 変数の WTT を実現するハードウェアのコストは $O(2^n)$ である．

(証明) 根に対応する変数を w_1，葉に対応する変数を w_n とすると，変数 w_i に対応する加減算回路は，2^{i-1} 個必要である．変数 w_i に対応する加減算回路一個のコストは，$\alpha(n-i+1)$ となる．従って，WTT を実現するコストは，

$$\begin{aligned}
&\sum_{i=1}^{n} \alpha(n-i+1)2^{i-1} \\
&= \alpha(n+1)\sum_{i=1}^{n} 2^{i-1} - \alpha\sum_{i=1}^{n} i2^{i-1} \\
&= \alpha(n+1)(2^n-1) - \alpha[(n-1)2^n+1] \\
&= \alpha(2^{n+1}-n-2) = O(2^n)
\end{aligned}$$

となる． (証明終)

3.2 全ての Walsh 係数を計算する場合

［定理 3.2］ n 変数の Walsh 係数を全部同時に計算するハードウェアのコストは $O(n^2 2^n)$ である．

(証明) バタフライ演算の各節点を加算器，及び，減算器で置換すると，全ての Walsh 係数を同時に計算できる．各段に対して 2^{n-1} 個の加算器と減算器が必要である．また，第 i 段目に対応する加算器，及び，減算器 1 個のコストは βi と仮定する．ここで，β は定数．従って，ハードウェア全体のコストは

$$\sum_{i=1}^{n} \beta i 2^{n-1} \times 2 \;=\; \beta 2^n \sum_{i=1}^{n} i = \beta 2^n \frac{n(n-1)}{2} = O(n^2 2^n)$$

となる． (証明終)

4. 多出力関数の場合

整数関数 $Z = 2^{m-1}f^{m-1} + 2^{m-2}f^{m-2} + \cdots + 2^0 f^0$ を用いると，WTT や WTD は，多出力関数 $(f^{m-1}, f^{m-2}, \ldots, f^0)$ を表現できる [15]．(注1)

［定理 4.1］ $s(f^i : w_1, w_2, \ldots, w_n)$ を $f^i (i = 0, 1, \ldots, m-1)$ の SOPP とするとき，Z の SOPP は，

$$\sum_{i=0}^{m-1} s(f^i : w_1, w_2, \ldots, w_n) \cdot 2^i.$$

(注1)：f_0 は，$f(0, 0, \ldots, 0)$ を表し，f^0 は，第 0 番目の関数を表現することに注意．

表 5.1 加減算器の符号

符号			2 の補数	加減算器での数
0	0	0	0	0
0	0	1	1	1
0	1	0	2	2
0	1	1	3	3
1	0	0	-4	4
1	0	1	-3	-3
1	1	0	-2	-2
1	1	1	-1	-1

となる．

定理 4.1 より，多出力関数の Walsh 係数は，f^i の Walsh 係数を 2^i 倍したものを全ての $i = 0, 1, \ldots, m-1$ に関して加算すれば求まる．

多出力関数の Walsh 係数は，MTBDT を用いても計算できるが，率直に実現すると各非終端節点のハードウェア量が増える．定理 4.1 の方法を用いると，ハードウェアのほとんど部分が出力数 m に依存しない．出力数に依存するのは，最後の加算回路の部分だけとなり，ハードウェア量を大幅に削減できる．ただし，計算時間は出力数 m に比例する．

5. 実験結果

5.1 1 個の係数を計算する回路

第 2.2, 2.3 章で WTT, WTD に基づいた Walsh 係数の計算方法を述べた．本章では，WTT のハードウェア化についてのみ考察する．なぜなら，WTD を用いる事ができるのは関数が固定されている場合のみだからである．

スペクトラムを計算する際，入力のエンコーディングとして $(0, 1)$ と $(1, -1)$ の二通りの方法が存在する [10]．本論文では $(0, 1)$ エンコーディングを用いる．この場合，n 変数関数のスペクトラムの最大値は 2^n，最小値は -2^{n-1} となる．

図 5.1(a) に $n = 3$ の場合の**組合せ回路方式**の回路図を示す．また，図 5.1(b) に $\vec{w} = (w_1, w_2, w_3) = (1, 1, 0)$ の場合の計算例を示す．図 5.1(c) に示すように，WTT で用いる加減算器は，$(2k+1)$ 入力 $k+1$ 出力である．本実現では，ハードウェア量を節約するため，$(1, 0, 0, \ldots, 0)$ は，2^k (正の数) を表現し，それ以外の数は，2 の補数表現という，特殊な符号を利用している．$k = 2$ の場合を表 5.1 に示す．ここで $(1, 0, 0)$ は 4 を表現することに注意．

表 5.2 に実験環境と実験条件，表 5.3 に組合せ回路方式の場合の使用論理ブロック (ALUT) の個数や遅延時間の実験結果を示す．**組合せ回路方式**の場合，ピン数の制限から $n = 10$ までの回路が実現可能である．この場合，ALUT の使用率はわずか 2 パーセントであった．n が増えるに従って，ALUT 数は 2^n に比例して増えている．

n が 11 以上の回路を実現するにするために，図 5.2 に示すような**時分割方式**を用いた．本方式では，

（1） 論理関数の真理値ベクトル $(f_0, f_1, \ldots, f_{2^n-1})$ を幾つかのグループに分割する．

（2） 各グループ毎に，$Load_i$ 信号を 1 とすることにより，レ

ジスターに論理関数値を取り込む.

（3） すべての論理関数値をレジスタに取り込んだ後, Walsh 関数の値を計算する.

図 5.2 は, 入力のデータ幅が 4 ビットの例を示しているが, 表 5.2 のデバイスでは, 入力のデータ幅は 1024 ビットまで拡張可能である. 表 5.4 に, 時分割方式の場合の実験結果を示す. 本デバイスで図 5.2 の回路を実現する場合, 最大 $n = 14$ まで実現できる.

表 5.2 実験環境と実験条件

使用デバイス: Stratix II	
型名:	EP2S180F1508C4
ALUTs の個数:	143520
IO pins:	1173 (全部で 1508 pin)
Memory bits:	9383040
DSP block 9-bit elements:	768
使用計算機	
PC: IBM ThinkPad T41 Pentium M (1.6GHz), RAM 1GB	
論理合成・フィッティングツール	
Altera 社 Quartus II V4.1	
ツールの最適化パラメータ	
Fitter setting:	
Physical Synthesis Optimization,	
Perform physical synthesis for combinational logic Extra	
Timing-driven compilation:	
Optimize timing. Extra effort. Standard Fit (highest effort)	
Analysis Synthesis Settings:	
Speed	

5.2 全ての係数を計算する回路

一度に全ての係数を計算する回路も実現した. 回路は, 図 2.1 のバタフライ演算をそのまま実現した. 全ての係数を同時に求める $n = 7$ までの組合せ回路を表 5.2 の FPGA で実現できた. 表 5.5 に ALUT の個数と遅延時間を示す. $n \geqq 8$ については, FPGA の端子数が不足したので時分割方式を用いた. 表 5.6 に実験結果を示す. これらの表から, ALUT の個数は $O(n^2 \cdot 2^n)$ で増加していることがわかる.

表 5.3 組合せ回路方式の実験結果.

n	Pins	ALUTs		Delay[ns]
6	77	230	(<1%)	13.3
7	143	471	(<1%)	15.2
8	273	954	(<1%)	18.7
9	531	1924	(1%)	21.0
10	1045	3965	(2%)	26.2

表 5.4 時分割方式の場合の実験結果.

n	Pins	ALUTs		Registers		Delay[ns]
11	1050	8764	(6%)	2048	(1%)	39.3
12	1054	17547	(12%)	4095	(2%)	43.0
13	1060	35118	(24%)	8192	(5%)	47.0
14	1070	70256	(48%)	16374	(10%)	51.7

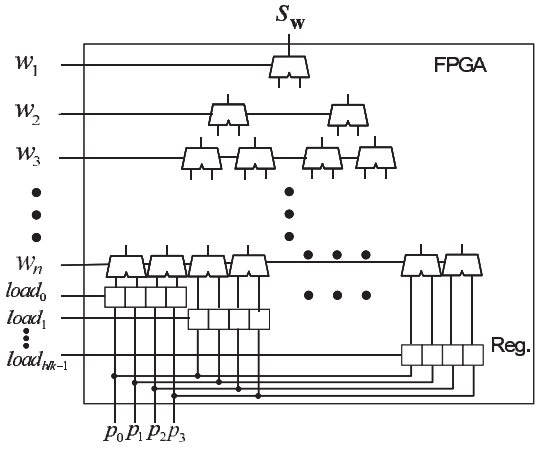


図 5.2 n が大きい場合の Walsh 係数計算回路 (時分割方式).

表 5.5 全ての係数を計算する組合せ回路方式の実験結果

n	Pins	ALUTs		Delay[ns]
4	96	218	(<1%)	8.4
5	224	676	(<1%)	11.4
6	512	1896	(1%)	15.4
7	1152	5072	(3%)	18.8

表 5.6 全ての係数を計算する時分割方式回路の実験結果

n	Pins	ALUTs		Delay[ns]
8	273	14398	(10%)	24.4
9	532	35496	(25%)	32.9
10	1046	85395	(59%)	44.7

表 5.7 マイクロプロセッサと FPGA での計算時間の比較

n	time ($n$sec)		Speed-up
	FPGA	MPU	
8	18.7	962	51
9	21.0	1903	91
10	26.2	3914	149
11	39.3	8181	208
12	43.0	16100	372
13	47.0	32200	685
14	51.7	64800	1253

5.3 マイクロプロセッサとの比較

Walsh 係数を計算するソフトウエアには, 種々のものがある. データ構造としては, 真理値ベクトルをアレイで表現したものを仮定する. 任意の係数を計算するためには, 真理値ベクトルの全ての 2^n 個の要素を参照し, 加算と減算を併せて $2^n - 1$ 回実行しなければならない. よって, Walsh 係数 1 個を計算するには少なくとも $f_0 + f_1 + \ldots + f_{2^n-1}$ を計算するための時間が必要となる. よって, 各 n に対して, $f_0 + f_1 + \ldots + f_{2^n-1}$ の値を計算するためのコードを生成した. 本論文では, 10^6 回実行し, 平均値を計算した. 表 5.2 に示す計算機と gcc コンパイラを実験に用いた.

表 5.7 は計算時間の比較を示す. マイクロプロセッサ (MPU) の場合, 計算時間は 2^n に比例して増えている. 表より, FPGA による実現は, $n = 14$ のとき MPU に比べ 1253 倍高速であることがわかる. 文献 [5], [12] のソフトウェアによる実現は, 固定

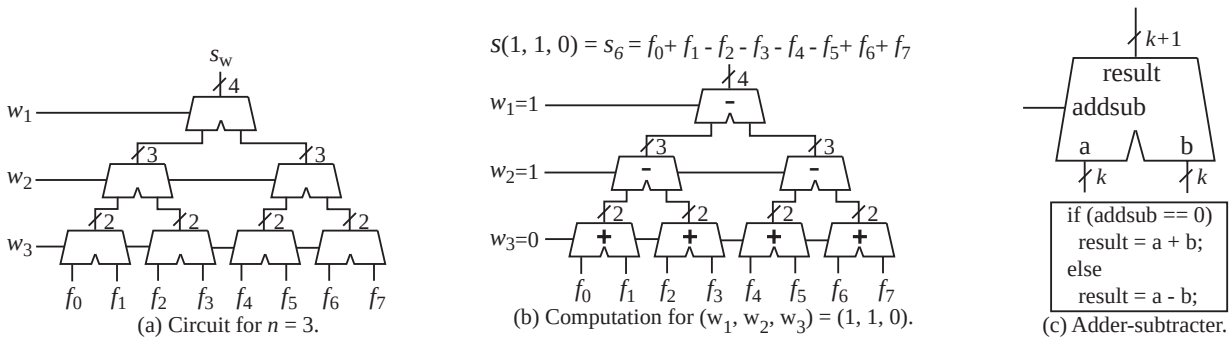


図 5.1　$n = 3$ に対する Walsh 係数計算回路 (組み合せ回路方式).

された関数の係数のみを計算可能で, 予備的な計算を必要とする. 一方, 本稿で提案した実現方法は, 予備的な計算なしに任意の関数に対する Walsh 係数を計算できる.

6. 結　　論

本論文では, 論理関数の Walsh 係数を Walsh 変換木より直接, 計算するハードウェアの構成法を示した. また, この回路を FPGA を用いて設計した. 本方法では $n \leqq 14$ までを市販の FPGA で実現可能だった. マイクロプロセッサを用いたソフトウェア実現より本方法は, $n = 14$ の時 1253 倍高速であった. 係数を 1 個求める回路と全ての係数を同時に求める回路のハードウェア量は, それぞれ $O(2^n)$, $O(n^2 \cdot 2^n)$ であることも示した.

謝　　辞

本研究は, 一部, 文部科学省・科学研究費補助金, および, 文部科学省・北九州地域知的クラスター創成事業の補助金による. また, 有益なご討論を頂いた永山忍氏に感謝する.

文　　献

[1] Altera: http://www.altera.com/

[2] A. Amira, A. Bouridane, P. Milligan, and M. A. Roula, "Novel FPGA implementations of Walsh Hadamard transforms for signal processing," *IEE Proceedings on Vision, Image and Signal Processing*, pp. 377-383, Vol. 148, No. 6, December 2001.

[3] K. G. Beauchamp, *Applications of Walsh and Related Functions*, New York: Academic Press, 1984.

[4] R. E. Bryant, "Graph-based algorithms for Boolean functions manipulation," *IEEE Trans. Computers,* vol. 35, no. 8, pp. 667-691, Aug. 1986.

[5] E. M. Clarke, K. L. McMillan, X. Zhao, M. Fujita, and J. Yang, "Spectral transforms for large Boolean functions with application to technology mapping," *Proc. Design Automation Conf.*, pp. 54-60, June, 1993.

[6] 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 Trans. Computer-Aided Design*, vol. 11, no. 10, pp. 1207-1226, Oct. 1992.

[7] M. Fujita, J. C.-Y. Yang, M. Clarke, X. Zhao, and P. McGeer, "Fast spectrum computation for logic functions using binary decision diagrams," *Proc. Int'l Symp. Circuit and Systems (ISCAS '94)*, pp. 275-278, 1994.

[8] J. P. Hansen and M. Sekine, "Synthesis by spectral translation using Boolean decision diagrams," *Proc. 33rd Design Automation Conf.*, pp. 248-253, June, 1996.

[9] T-C. Hsiao and S. C. Seth, "An analysis of the use of Rademacher-Walsh spectrum in compact testing," *IEEE Trans. Computers*, vol. 33, pp. 934-938, Oct. 1984.

[10] S. L Hurst, D. M. Miller, and J. C. Muzio, *Spectral Techniques in Digital Logic*, Academic Press, 1985.

[11] A. Iseno, Y. Iguchi, and T. Sasao, "Fault diagnosis for RAMs using Walsh spectrum," *IEICE Trans. Information and Systems*, Vol. E87-D, No.3, March 2004, pp. 592-600.

[12] D. Jankovic, R. S. Stankovic, and R. Drechsler, "Decision diagram method for calculation of pruned Walsh transform," *IEEE Transactions on Computers*, Vol. 50, No. 2, Feb. 2001, pp. 147-157.

[13] R. J. Lechner, "Harmonic analysis of switching functions," *In A. Mukhopadhyay, editor, Recent Developments in Switching Theory,* New York, Academic Press, 1971.

[14] Synplify: http://www.synplicity.com/

[15] R. S. Stankovic, T. Sasao, and C. Moraga, "Spectral transform decision diagrams," *Representations of Discrete Functions*, T. Sasao and M. Fujita, eds., pp. 55-92, Kluwer Academic, 1996.

[16] R. Stankovic and J. Astola, *Spectral Interpretation of Decision Diagrams, Springer-Verlag*, May 16, 2003.

[17] A. K. Susskind, "Testing by verifying Walsh coefficients," *IEEE Transactions on Computers*, Vol. 32, No.2, pp. 198-201, Feb. 1983.

[18] M. A. Thornton, R. Drechsler, and D. M. Miller, *Spectral Techniques in VLSI CAD*, Kluwer Academic Publishers, July, 2001.

[19] M. A. Thornton and V. S. S. Nair, "Efficient calculation of spectral coefficients and their applications," *IEEE Trans. Computer-Aided Design Integrated Circuits and Systems*, Vol. 4, no. 11, pp. 1328-1341, Nov. 1995.

[20] D. Varma and E. A. Trachtenberg, " Design automation tools for efficient implementation of logic functions by decomposition," *IEEE Trans. CAD*, Vol. 8, pp. 901-916, Aug. 1989.