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 変換を用いた半導体メモリの故障診断法

伊勢野　総[†]　　井口　幸洋[†]　　笹尾　勤[††,†††]

† 明治大学 理工学部
†† 九州工業大学 情報工学部
††† 九州工業大学マイクロ化総合技術センター
E-mail: †{iseno,iguchi}@cs.meiji.ac.jp, ††sasao@cse.kyutech.ac.jp

あらまし　半導体メモリのフェイル・ビットマップを Walsh 変換し, Walsh スペクトラムを得て, 故障診断を行う方法を提案する. 単一縮退故障の場合, 0 次と 1 次のスペクトラムの係数のみから故障の種類を判別できることを示す. アドレスのビット数が n のメモリを診断するための計算複雑度は $O(n\times 2^n)$ なので高速に故障を診断できる.

キーワード　メモリテスト, ウォルシュ変換, 故障診断, フェイル・ビットマップ, SRAM

Fault Diagnosis for RAMs using Walsh Transform

Atsumu ISENO[†], Yukihiro IGUCHI[†], and Tsutomu SASAO[††,†††]

† Department of Computer Science, Meiji University
†† Department of Computer Science and Electronics, Kyushu Institute of Technology
††† Center for Microelectronic Systems, Kyushu Institute of Technology
E-mail: †{iseno,iguchi}@cs.meiji.ac.jp, ††sasao@cse.kyutech.ac.jp

Abstract　In this paper, we show a method to locate single-stuck at fault for random access memory (RAM). From the fail-bitmaps of the memory, we obtain the Walsh transform of them. For single-stuck at faults, we show that the fault can be located by using only the 0-th and 1-st coefficients of the spectrum. The computation time is $O(n\times 2^n)$, where n is the number of bits in the address. Thus, we can quickly locate the faults.

Key words　memory test, Walsh transform, fault diagnosis, fail-bitmap, SRAM

1. はじめに

半導体設計・製造技術の進展により半導体メモリの微細化・大容量化が進展している. これに伴い, 検査や診断に要するコストも増大している.　半導体メモリは製造過程の様々な段階で検査され, この時に故障情報がテスタより出力される. この故障情報は製造プロセスの改善に役立ち, 歩留まりの向上などに利用される. メモリは, 単にコンピュータの記憶ユニットとして単体で使われるだけでなく, ディープサブミクロンの時代には, その規則性と高集積化容易な特性を利用し, 論理回路の構成部品として重要性が増している. また, SoC(System on Chip) では, 複数のメモリが一つのチップ上に集積され, この意味でも重要性が増している.

また, 微細化に伴い全く故障のないメモリチップの製造は難しく, 不良セルがあった場合, メモリ内部の代替用のセルを利用してメモリの歩留まりを上げる必要がある. この場合にも, どこが故障しているかを診断することが重要となる.

メモリのテスト・診断技術は重要でありさまざまな研究が行われている. 診断用の情報ではフェイル・ビットマップが最も重要である. これは, テストパターンを加えたとき, メモリが期待値どおりの値を出力するかを 0/1 で表したマップである. 検査を行う時, これをディスプレイに表示し, そのパターンから故障を推測する. これを自動的に判定する研究としては, フェイル・ビットマップのパターンを図形として分類することで診断を行う方法 [5] やテスト応答を圧縮し, これによって診断を行う組込み用 RAM 向きの方法 [1] などがある.

本稿では, フェイル・ビットマップが故障の位置と種類によって特徴的なパターンを出力することを利用し診断を行う. フェイル・ビットマップを論理関数として扱い, これを論理式簡単化ツールなどで圧縮すると各故障に特有な論理式が得られる. しかし, 論理式簡単化のための計算時間は長いので, これに代わる方法が必要となる. ここでは, フェイル・ビットマップの Walsh 変換を行い Walsh スペクトラムを得て, その 0 次と 1 次の係数で単一縮退故障を診断する方法を提案する.　本提

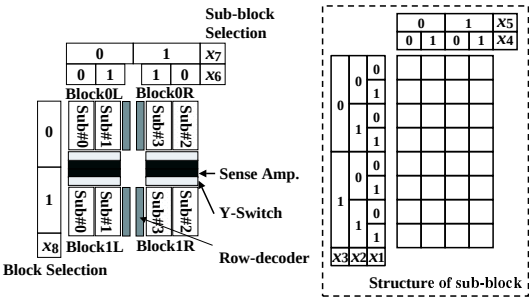

図 2.1　メモリの構造

案手法で必要な計算時間は, 論理式簡単化に要する計算時間に比べ短い.

第 2 章では, 半導体メモリの構造とテストについて, 第 3 章では Walsh 変換と Walsh スペクトラムを用いた故障診断の方法を提案する. 第 4 章では実験結果について, 第 5 章では, まとめを述べる.

2. 半導体メモリの構造とテスト

本章ではメモリの構造, 故障, および, テストパターンについて概説する.

2.1 メモリの構造と故障モデル

図 2.1に半導体メモリ (以後, 単にメモリと表記) の構造例を示す. メモリは, 1 ビットを記憶するセルを 2 次元アレイ上に並べた構造をしている. ただし, 一つの大きなアレイで実現すると動作速度や消費電力の点から不利なので, 実際には幾つかのセルをまとめて**サブブロック**を構成し, 幾つかのサブブロックで**ブロック**を構成し, さらに幾つかのブロックで一つのメモリを構成するというように階層的な構造となっている. サブブロックに注目すると, **ワード線**と呼ばれる水平方向の線と**ビット線**と呼ばれる垂直方向の線からなり, それらの交差点に**記憶セル**が配置される.

メモリでは製造過程においてさまざまな理由, 例えば, プロセスの不具合, 結晶の欠陥, 塵などのために起因する物理的欠陥が生じる. 外部からメモリにテスト・パターンを加え, 故障として判定されればそのチップは不良品として廃棄する. 初期ロットのメモリでは, 故障が頻発する可能性が高く, コストの増大を招くので, 故障の原因を早急に特定し, 製造過程にその結果をフィードバックし, 歩留まりを改善する必要がある.

物理的欠陥が生じた場所とその種類により, 外部から観測した様子が異なる. メモリの代表的な故障モデルを表 2.1に示す. 本稿では, 表 2.1に示す故障を取り扱うことにし, 図 2.1の構造のメモリを図 2.2に示すようにモデル化する. サブブロック 2 個で一つのブロックを構成する. 1 個の**行デコーダ**がこのブロックのワード選択を担当する. サブブロックにはそれぞれ 4 本のビット線があり, この 4 本のビット線の切替を **Y スイッチ**が担当する. Y スイッチには 1 個の**センスアンプ (SA)** がつながる. 従って, このメモリでは 2 個のサブブロックで構成されたブロックが 4 個あり, それを切り替えるために**ブロック・サブブロック選択用マルチプレクサ**がある. このメモリ・モデル

表 2.1　故障種別と故障モデル

故障種別	故障モデル
Cell	セルの 0(1) 縮退故障
Row Decoder	行デコーダの入力 0(1) 縮退故障
Bit Line	ビット線の 0(1) 縮退故障
Sense Amplifier	センス・アンプ出力の 0(1) 縮退故障
I/O	データ入出力線の 0(1) 縮退故障
Block, Sub block 選択用 MUX	ブロック, サブブロック選択用 MUX の セレクト入力の 0(1) 縮退故障
Y-Switch	Y スイッチのセレクト入力の 0(1) 縮退故障
Address Line	アドレス線の 0(1) 縮退故障

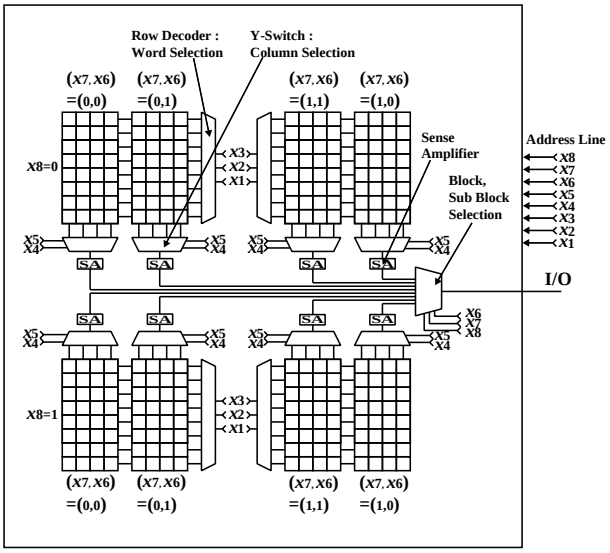

図 2.2　メモリモデル

上で表 2.1に示す 8 箇所の単一故障を本稿で取り扱う.

2.2 メモリのテスト

メモリが良品か不良品かを判定するために, メモリにテスト・パターンを加える. テスト・パターンには N 系, $N^{1.5}$ 系, N^2 系がある. テスト時間は N 系が一番短く, 次に $N^{1.5}$ 系で, 最も長いのが N^2 系である. テスト時間の関係から現在では多くの場合 N 系が使用される [2].

N 系の代表的なテスト・パターンとしてマーチ・パターンが知られている [2]. マーチ・パターンにも多くの種類があり, 図 2.3にテスト長が最短の MATS+ を示す. MATS+ は, **ステップ M0** ですべてのアドレスに対して 0 を書き込む (0W: Zero Write). 次に**ステップ M1** で 0 番地より 1 アドレス毎にメモリの内容を読む. 期待値である 0 が読み出されたら正常を表す 0 を出力し, もし 1 が読み出されたら異常を表す 1 を出力する. 読み出しが終わったらその番地に 1 を書き込む. これを最大アドレスまで行う (0R-1W: Zero Read - One Write). 次の**ステップ M2** では, これを逆に最大アドレスから 0 番地まで 1 を読み 0 を書くことを繰り返す (1R-0W). ステップ M1 と M2 でそれぞれ出力される正常か異常かを示すパターンをそれぞれ**フェイル・ビットマップ 1** および**フェイル・ビットマップ 2** と呼ぶ. これらのフェイル・ビットマップで表 2.1に示す故障をすべて検出できる.

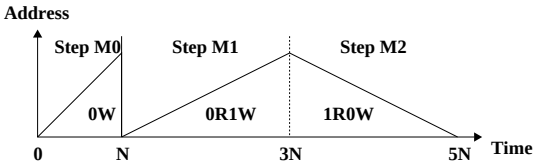

図 2.3　テストパターン MATS+

［例 2.1］ 表 2.1で示した故障の内, セル, 行デコーダ, ビット線に単一故障を有する各メモリに対し, MATS+ を加えたときのフェイル・ビットマップの例を図 2.4(a)-(d) に示す.

- 図 2.4(a) は Block0R 内のサブブロック Sub#2 内のセルに 0 縮退故障が存在している例である. この時, MATS+ のステップ M1 では, 異常セルに対しても 0 が読み出されるので異常を示す 1 はフェイル・ビットマップ 1 には出力されず, フェイル・ビットマップ 2 には異常が出力される.
- 図 2.4(b) と (c) は, ブロック 0L 用の行デコーダの入力 x_1 が 0 縮退故障を起こしている例である. Block0L 内のサブブロック Sub#0 および Sub#1 内の半分のセルにのみアクセスされる. 例えばステップ M1 で 0 番地から期待値 0 が読み出され 1 を書き込む. 次に 1 番地を読むと, 行デコーダの選択線に故障が生じているので再び 0 番地を記憶するセルの内容が読み出される. この時, このセルには 1 が書き込まれているので期待値 0 と一致せず異常を示す 1 がフェイル・ビットマップに出力される. つまり, Sub#0 と Sub#1 内の奇数番地のセルが異常を示す. ステップ M2 では, アクセスがアドレスの降順に行われるので今度は偶数番地のセルが異常を示すように見える.
- 図 2.4(d) は, Block1R 内のサブブロック Sub#2 のビット線 1 が 0 縮退故障を起こしている例である. この時, フェイル・ビットマップ 1 には異常は出力されない. ∎

メモリ・モデルを解析することにより次のことがわかる.

（1） 単一故障でも複数セルが故障しているように見える.

（2） 故障の種類によってフェイル・ビットマップが異なる.

（3） フェイル・ビットマップ 1 とフェイル・ビットマップ 2 は一般に異なる.

（4） フェイル・ビットマップより故障診断可能である.

以上より次の定理を得る.

［定理 2.1］ 表 2.1の故障モデルの下で図 2.2のメモリの単一縮退故障は, 図 2.3の MATS+ により得られるフェイル・ビットマップより診断可能である.

3. Walsh スペクトラムを用いた故障診断

2.章では故障の種類によってフェイル・ビットマップが異なることを示した. 本章では, これらのフェイル・ビットマップから Walsh 変換 [10] を用いて故障の箇所を特定する方法を提案する. まず, 3. 1節で Walsh 変換について概説する. 次に, 3. 2節で単一縮退故障時のフェイル・ビットマップの特徴を述べる. 3. 3節では, Walsh スペクトラムを用いたメモリの単一縮退故障時の故障診断法を提案する.

3. 1 Walsh 変換と Walsh スペクトラム [3], [7], [8]

$\mathbf{W}(n)$ を次式で定義される Walsh 変換行列とする.

$$\mathbf{W}(n) = \bigotimes_{i=1}^{n} \mathbf{W}(1), \quad \mathbf{W}(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

ここで $\bigotimes$ はクロネッカー積を示す. また, $\mathbf{W}(1)$ を, 基本 Walsh 変換行列と呼ぶ. 特に $\mathbf{W}(1)$ を記号で表現すると

$$\mathbf{W}(1) = \begin{bmatrix} 1 & 1-2x_i \end{bmatrix}$$

となる. ここでの計算は整数演算である.

［定義 3.1］ 真理値表ベクトル $\mathbf{F}$ で与えられた関数 f に対して, Walsh スペクトラム $\mathbf{W}_f = [w_0, \ldots, w_{2^n-1}]^t$ を, 次式のように定義する.

$$\mathbf{W}_f = 2^{-n}\mathbf{W}(n)\mathbf{F}$$

Walsh 変換行列は, 定数 2^n を無視すると自分自身が逆行列となるので, Walsh 逆変換は次式のようになる.

$$f = \mathbf{X}_w\mathbf{W}_f, \quad \mathbf{X}_w = \bigotimes_{i=1}^{n} \begin{bmatrix} 1 & 1-2x_i \end{bmatrix}$$

3. 2 単一縮退故障時のフェイル・ビットマップの特徴

2.章で示したように, フェイル・ビットマップ中の 1 の個数および出現場所により故障の箇所を特定できる. 診断方法としては, フェイル・ビットマップ上の故障箇所の形状を矩形や点や線分として分類し, 数値化して記憶する方法がある [4]. これは情報圧縮としても意味がある. また, フェイル・ビットマップを一つの画像と考え, 画像圧縮法を応用し情報を圧縮し, 解析時に解凍・表示して故障診断する方法もある [9]. これらの方法では最後には, パターンを表示して人間が判断する必要がある.

ここでは, 単一故障を仮定したときのフェイル・ビットマップを用いた故障診断法について考える. 最も簡単な方法は, フェイル・ビットマップ中の 1 の個数 (**マップ関数の重み**) を数えることである. 例えば, セルの単一縮退故障では, フェイル・ビットマップ 1 か 2 のどちらか一方に 1 が 1 個出現する. 従って, マップ関数の重みが 1 の場合, セル故障と特定できる. しかし, この方法では, マップ関数の重みが同じになる故障は識別できない.

フェイル・ビットマップ 1 が表現する関数を**マップ関数 1**, フェイル・ビットマップ 2 が表現する関数を**マップ関数 2** と呼ぶ. 単一故障時の**マップ関数**は, 後に定理 3.1で示すように定数または単一の積項で表現可能である. 例えば, 図 2.4(a)-(d) のマップ関数を論理式で表現すると表 3.1に示すように単一の積項または定数で表現できる. 例えば, (a) の例のように行デコーダの選択信号線 x_1 に 0 縮退故障が起きると, マップ関数 1 は, 積項 $\bar{x}_8\bar{x}_7x_1$ で表現できる. 他の故障に対するマップ関数も表 3.1に示すように定数または単一の積項で表現可能である. 表の上段はマップ関数 1 を表す積項とマップ関数の重み, 下段はマップ関数 2 を表す積項とマップ関数の重みである. 例えば, 行デコーダの入力線の x_1 が 0 縮退故障を起こしているときは, マップ関数 1 の重みは 32 であり, マップ関数 1 を表す積項は $\bar{x}_8\bar{x}_7x_1$ となる. また, マップ関数 2 の重みは 32 であり, マッ

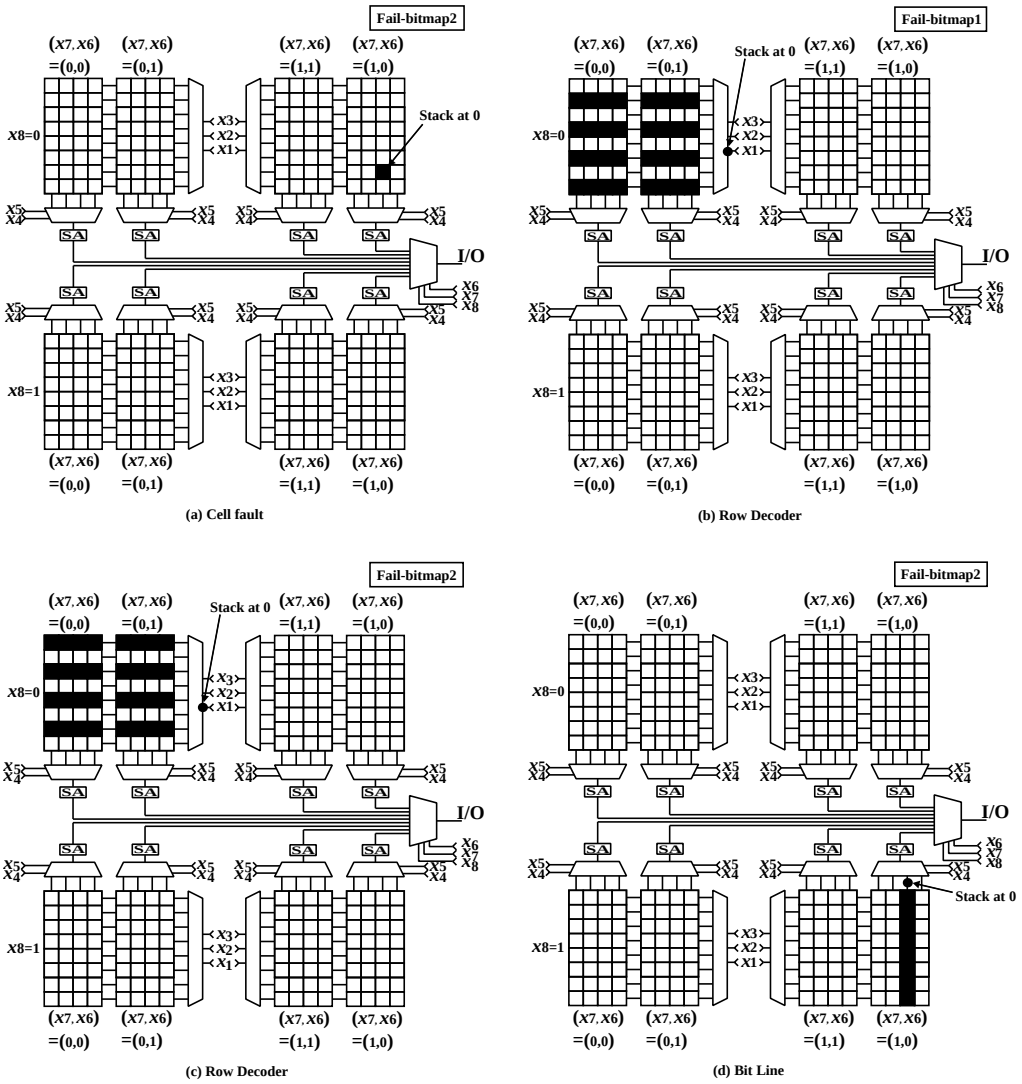

図 2.4　フェイル・ビットマップ

表 3.1　マップ関数による故障の表現例

故障種別	0 縮退故障		1 縮退故障	
	マップ関数	重み	マップ関数	重み
Cell	0	0	$\bar{x}_8x_7\bar{x}_6x_5\bar{x}_4x_3x_2\bar{x}_1$	1
	$\bar{x}_8x_7\bar{x}_6x_5\bar{x}_4x_3x_2\bar{x}_1$	1	0	0
Row Decoder	$\bar{x}_8x_7\bar{x}_1$	32	$\bar{x}_8x_7\bar{x}_1$	32
	$\bar{x}_8x_7x_1$	32	$\bar{x}_8x_7x_1$	32
Bit Line	0	0	$x_8x_7\bar{x}_6x_5\bar{x}_4$	8
	$x_8x_7\bar{x}_6x_5\bar{x}_4$	8	0	0
Sense Amplifier	0	0	$\bar{x}_8x_7\bar{x}_6$	32
	$\bar{x}_8x_7\bar{x}_6$	32	0	0
I/O	0	0	1	256
	1	256	0	0
Y-Switch	$\bar{x}_8x_7\bar{x}_6x_4$	16	$\bar{x}_8x_7\bar{x}_6x_4$	16
	$\bar{x}_8x_7\bar{x}_6\bar{x}_4$	16	$\bar{x}_8x_7\bar{x}_6\bar{x}_4$	16
Address Line	x_4	128	x_4	128
	$\bar{x}_4$	128	$\bar{x}_4$	128

プ関数 2 を表す積項は $\bar{x}_8\bar{x}_7\bar{x}_1$ となる．ここで表 2.1にあるブロック・サブブロック選択用 MUX の選択入力の縮退故障を表 3.1の故障種別に載せていないのは, この故障がアドレス線の故障に含まれるからである．

表 3.1に示すようにマップ関数 1, および, マップ関数 2 の重みは, 故障毎に異なるので単一故障の場合, 故障種別の特定が可能である．本稿の例ではマップ関数の重みはすべて異なっているが, メモリの構成によってはマップ関数の重みが同じ時でも故障が異なる場合がある．この時は, リテラルを調べることにより故障箇所を特定できる．このように論理式を用いれば故障診断が可能である．ただし, 論理式の処理時間が長いという欠点がある．

すべての単一縮退故障を検討することにより次の定理を得る．

［定理 3.1］ 表 2.1の故障モデルの下で図 2.2のメモリの単一縮退故障のマップ関数は定数または単一の積項で表現可能である．また, この積項により故障診断可能である．

3. 3　Walsh スペクトラムを用いた単一縮退故障の診断法

本節では Walsh スペクトラムを用いた単一縮退故障の診断法について述べる．特に, スペクトラムの 0 次と 1 次の係数の計算はアルゴリズムが簡単なため, 高速に実行可能である．

［定義 3.2］ Walsh スペクトラムの第 j 成分を w_j とする．j の 2 進表現を $(k_n, k_{n-1}, \ldots, k_1)$ とし, $k_l = 1$ となる要素からなる集合を $R \subseteq \{1, 2, \ldots, n\}$ とする．このとき, $s_R = w_j$ とする．s_ϕ を **0 次の係数**, s_i を **1 次の係数**, s_{ij} を **2 次の係数**, s_{ijk} を **3 次の係数**という．

以下では, Walsh スペクトラムを考えるとき係数 2^{-n} は無視

することにする.

[補題 3.1] n 変数論理関数 $f(x_1, x_2, \ldots, x_n)$ において, 積項 $x_1x_2\cdots x_{n-t}$ に対する Walsh スペクトラムの係数の値の絶対値は, 2^t 又は, 0 である. また, 非零の係数は, $x_{n-t+1} = x_{n-t} = \cdots = x_n = 0$ に対応する部分である。

[補題 3.2] 積項 α を表現する論理関数の Walsh スペクトラムを $\mathbf{W}_\alpha$ とする. 積項 α において, ある変数の否定 (Negation) と変数の置換 (Permutation) を行った積項を β とし, それに対する Walsh スペクトラムを $\mathbf{W}_\beta$ とする. この時, $\mathbf{W}_\beta$ は, $\mathbf{W}_\alpha$ の要素を置換し符号を変換したものである.

[定義 3.3] $|f|$ を論理関数 f の重み, つまり, $f(\vec{a}) = 1$ となる $\vec{a}$ の個数とする.

[例 3.1] $n = 3$ のとき, $|x_2x_3| = 2$, $|x_1 \vee x_2 \vee x_3| = 7$.

[補題 3.3] 論理関数の Walsh スペクトラムを $\mathbf{W}_\alpha$ とする. この時, 各係数について $s_i = |f| - 2v_i$, ここで, $v_i = |x_i f|$ が成立する.

[補題 3.4] 積項を表す論理関数 $f = \hat{x}_1\hat{x}_2\cdots\hat{x}_{n-t}, 0 \leqq t \leqq n$ の Walsh スペクトラムを考える. $s_i = 0$ ならば f は x_i に依存しない. $s_i > 0$ ならば f の積項はリテラル $\bar{x}_i$ を含む. $s_i < 0$ ならば f の積項はリテラル x_i を含む. ここで $\hat{x}_j$ はリテラル x_j または $\bar{x}_j$ を示す.

定理 3.1および補題 3.4より, 次を得る.

[定理 3.2] 表 2.1の故障モデルの下で, 図 2.2のメモリの単一縮退故障は, マップ関数の第 0 次, および第 1 次の係数 $(s_\phi; s_1, s_2, \ldots, s_n)$ を用いて診断可能である.

Walsh スペクトラムを用いた故障診断法では, 0 次と 1 次の係数 $(s_\phi; s_1, s_2, \ldots, s_n)$ を用いる. 0 次および 1 次の係数はそれぞれ 1 個, n 個存在する. そのため, ステップ M1 で得られる**スペクトラム 1** の $n+1$ 個の係数と, ステップ M2 で得られる**スペクトラム 2** の $n+1$ 個の係数の併せて $2n+2$ 個の係数が必要である.

スペクトルを用いて, 故障診断のための故障辞書を作成する. 各故障に対し, フェイル・ビットマップはメモリモデルより導出できる. マップ関数を Walsh 変換し Walsh スペクトラムを求め, 0 次と 1 次の係数を記載する. 定理 3.1から明らかなように, 単一縮退故障時には, マップ関数のスペクトラムの非零要素の絶対値はすべて等しくなる. 例えば, ビット線に 0 縮退故障が起きるとマップ関数 2 の Walsh スペクトラムのうち s_ϕ, s_4, s_5, s_6, s_7, s_8 の値が 8 となり, 残りの係数は全て 0 となる. 従って, $(s_\phi; s_1, s_2, \ldots, s_8)$ の値は, $(8; 0, 0, 0, 8, 8, 8, 8, 8)$ のようになるが, 紙面を節約するために表 3.2では, $8(1; 0, 0, 0, 1, 1, 1, 1, 1)$ のように略記している. 故障箇所は, 1 次の各係数の値の正負で判定可能である.

以上のことを利用して次の診断法が得られる.

診断プログラムの生成

（1） **故障辞書の生成**

メモリモデルからフェイル・ビットマップを導出し, マップ関数を Walsh 変換して Walsh スペクトラムを求める. 0 次と 1 次の係数を記載することで, 表 3.2のような故障辞書を生成する.

（2） **多値決定木の作成**

（a） 決定木の各節点での分岐条件はスペクトラム $s_{i,j}(i = \phi, 1, \ldots, n, j = 1, 2)$ の絶対値とする. ここで $s_{i,j}$ は, スペクトラム j の i 番目の係数 s_i を表す.

（b） 故障辞書のスペクトラムに従って, 決定木を構成する. 終端節点に故障種別 (表 3.2の左の項目) を記載する. 終端節点は 14 個, 各節点は 3 本以上の分岐を有するので MTMDT(multi-terminal multi-valued decision tree) となる.

（3） **診断プログラムの生成**

多値決定木を簡単化し, ブランチング・プログラムを生成する.

故障診断の実行

（1） メモリにテストパターンを加え, マップ関数 1 およびマップ関数 2 を得る.

（2） マップ関数 1 および 2 を Walsh 変換し, 0 次と 1 次の係数を計算する.

（3） ステップ 2 で得られた各係数の絶対値を用いて, 診断プログラムを実行し, 故障の種別を判定する. 次に, ステップ 2 で得た各係数の正負からリテラルを求め, 故障位置を特定する.

表 3.2 スペクトラムを用いた故障辞書の例

故障種別	0 縮退故障 スペクトラム 1 スペクトラム 2	1 縮退故障 スペクトラム 1 スペクトラム 2
Cell	1(0; 0, 0, 0, 0, 0, 0, 0, 0) 1(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, 0)
Row Decoder[1]	32(1; 1, 0, 0, 0, 0, 0, 1, 1) 32(1; 1, 0, 0, 0, 0, 0, 1, 1)	32(1; 1, 0, 0, 0, 0, 0, 1, 1) 32(1; 1, 0, 0, 0, 0, 0, 1, 1)
Row Decoder[2]	32(1; 0, 1, 0, 0, 0, 0, 1, 1) 32(1; 0, 1, 0, 0, 0, 0, 1, 1)	32(1; 0, 1, 0, 0, 0, 0, 1, 1) 32(1; 0, 1, 0, 0, 0, 0, 1, 1)
Row Decoder[3]	32(1; 0, 0, 1, 0, 0, 0, 1, 1) 32(1; 0, 0, 1, 0, 0, 0, 1, 1)	32(1; 0, 0, 1, 0, 0, 0, 1, 1) 32(1; 0, 0, 1, 0, 0, 0, 1, 1)
Bit Line	1(0; 0, 0, 0, 0, 0, 0, 0, 0) 8(1; 0, 0, 0, 1, 1, 1, 1, 1)	8(1; 0, 0, 0, 1, 1, 1, 1, 1) 1(0; 0, 0, 0, 0, 0, 0, 0, 0)
Sense Amplifier	1(0; 0, 0, 0, 0, 0, 0, 0, 0) 32(1; 0, 0, 0, 0, 0, 1, 1, 1)	32(1; 0, 0, 0, 0, 0, 1, 1, 1) 1(0; 0, 0, 0, 0, 0, 0, 0, 0)
I/O	1(0; 0, 0, 0, 0, 0, 0, 0, 0) 256(1; 0, 0, 0, 0, 0, 0, 0, 0)	256(1; 0, 0, 0, 0, 0, 0, 0, 0) 1(0; 0, 0, 0, 0, 0, 0, 0, 0)
Y-Switch[1]	16(1; 0, 0, 0, 1, 0, 1, 1, 1) 16(1; 0, 0, 0, 1, 0, 1, 1, 1)	16(1; 0, 0, 0, 1, 0, 1, 1, 1) 16(1; 0, 0, 0, 1, 0, 1, 1, 1)
Y-Switch[2]	16(1; 0, 0, 0, 0, 1, 1, 1, 1) 16(1; 0, 0, 0, 0, 1, 1, 1, 1)	16(1; 0, 0, 0, 0, 1, 1, 1, 1) 16(1; 0, 0, 0, 0, 1, 1, 1, 1)
Address Line[1]	128(1; 1, 0, 0, 0, 0, 0, 0, 0) 128(1; 1, 0, 0, 0, 0, 0, 0, 0)	128(1; 1, 0, 0, 0, 0, 0, 0, 0) 128(1; 1, 0, 0, 0, 0, 0, 0, 0)
Address Line[2]	128(1; 0, 1, 0, 0, 0, 0, 0, 0) 128(1; 0, 1, 0, 0, 0, 0, 0, 0)	128(1; 0, 1, 0, 0, 0, 0, 0, 0) 128(1; 0, 1, 0, 0, 0, 0, 0, 0)
⋮	⋮	⋮
Address Line[8]	128(1; 0, 0, 0, 0, 0, 0, 0, 1) 128(1; 0, 0, 0, 0, 0, 0, 0, 1)	128(1; 0, 0, 0, 0, 0, 0, 0, 1) 128(1; 0, 0, 0, 0, 0, 0, 0, 1)

4. 実験結果

性能評価のため $1M\times4$ ビットの SRAM に対し本手法を適用した. 故障を模擬するシミュレータを作成し, ランダムに単

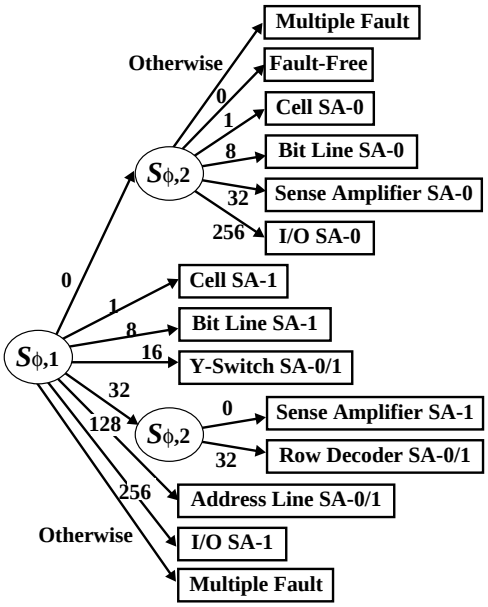

図 3.1　故障診断プログラム生成のための多値決定木

一故障を発生させ識別に要する時間を計測した．論理式簡単化法としてはESPRESSO MV [6] を用いた．

ESPRESSO MV は単一故障に関しては, 非常に高速に簡単化が行うことができた．しかし, アドレス線とセルが混在する故障などの, 一部の多重故障については簡単化に時間がかかり, 実用時間内に停止しない場合が存在した．使用計算機は PentiumII(1.13GHz) メモリ 384MByte, OS は WindowsXP を用いた．論理式簡単化にはセルの単一故障が最短の場合 28$[msec]$, アドレス線の単一故障が 21 秒, I/O 線の単一故障が最長の場合 23 秒の時間がかかった．また, Walsh スペクトラムの計算時間は, どのようなフェイル・ビットマップに対しても, 最大 9 秒程度で計算が終了した．

5. ま と め

本稿では, メモリに検査パターン (MATS+) を加えた際に生ずるフェイル・ビットマップを用いて故障の種類と位置を判定する方法について検討した．フェイル・ビットマップを論理関数と考えると, 各故障に対して固有の論理関数が対応する．従来の手法は, この論理関数を ESPRESSO 等の論理式簡単化プログラムで簡単化し, その論理式を解析する方法を用いていた．しかし, この方法は計算時間が大きくなることもあり, 不都合であった．本稿では, フェイル・ビットマップの Walsh スペクトラムの 0 次および 1 次の係数のみを用いて, メモリの単一縮退故障の診断が可能なことを示した．アドレスのビット数が n のとき, 計算複雑度は $O(n\times 2^n)$ なので, 論理式簡単化プログラムを用いる方法よりもはるかに高速である．

謝辞

本研究は一部日本学術振興会 科学研究費補助金および武田計測先端知財団補助金による．また, 北九州産業学術推進機構 SOC 設計センターの川本洋センター長, および九州工業大学の中村和之助教授にはメモリの検査法に関してご教授頂いた．

文　献

[1] J. T. Chen, J. Rajski, J. Khare, O. Kebichi, and W. Maly, “Enabling embedded memory diagnosis via test response compression,” *IEEE International Test Conference,* pp. 292-298, 2002.

[2] A. J. van de Goor, *Testing Semiconductor Memories —Theory and Practice—,* ComTex Publishing (Gouda), 1998.

[3] S. L. Hurst, D. M. Miller, J. C. Muzio, *Spectral Techniques in Digital Logic,* Academic Press Inc. (London) LTD. 1985.

[4] A. Iseno, and Y. Iguchi, “A method for storing fail bit maps in burn-in memory testers” in *Proc. International Workshop on Electronic Design, Test, and Applications,* pp. 142-145, Christchurch, Jan. 2002.

[5] W. Malzfeldt, W. Mohr, K. Kodalle, “Fast automatic failbit analysis for DRAMs, ” *IEEE International Test Conference,* pp. 431-438, 1989.

[6] R. Rudell, and A. Sangiovanni-Vincentelli, “Multiple-valued minimization for PLA optimization,” *IEEE Transactions on Computer Aided-Design,* vol.6, pp.727-750, Sep 1987.

[7] R. S. Stankovic, T. Sasao, and C. Moraga, “Spectral transform decision diagrams,” in T. Sasao and M. Fujita (ed.), *Representations of Discrete Functions,* Kluwer Academic Publishers 1996.

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

[9] J. Vollrath, U. Lederer, T. Hladschik, “Compressed bit fail maps for memory fail pattern classification,” in *IEEE European Test Workshop,* pp. 125-132, Cascais, May 2000.

[10] J. L. Walsh, “A closed set of normal orthogonal functions, ” *American Journal of Mathematics*, 1923.