設計自動化 55－13
(1990.12.13)

共有二分決定グラフを用いた多重故障の故障シミュレーション

高橋紀之　石浦菜岐佐　矢島脩三
京都大学 工学部

あらまし 従来、故障シミュレーションは単一故障が仮定された回路を対象としてきたが、大規模回路では、多重故障が起こる可能性が無視できないものとなる。多重故障の数が極めて多いため、多重故障の故障シミュレーションにおいては、計算量の削減が最大の課題となる。本稿では、多重縮退故障の仮定された組合せ回路のシミュレーションを、共有二分決定グラフを用いて効率良く行なう、演繹法に基づいた手法を提案する。この手法による多重故障シミュレータを実現し、いくつかの回路に対し実験を行なった結果、線形リストを用いる場合に比べ、格段に少ない領域でシミュレーションを行なうことができ、この手法の有効性が確認された。

Fault Simulation for Multiple Faults Using Shared Binary Decision Diagrams

Noriyuki TAKAHASHI, Nagisa ISHIURA, Shuzo YAJIMA
Faculty of Engineering, Kyoto University

Abstract In this paper a new fault simulation technique for multiple faults based on the deductive method is proposed. Main difficulty in multiple fault simulation is caused by the huge number of multiple faults. In order to cope with the difficulty sets of multiple faults are represented by Boolean functions. We use shared binary decision diagrams as internal representation of Boolean functions. We succeeded in simulating 4-ple faults of a 4-bit adder and double faults of the circuit of 2300 gates within 10MByte storage on Sun-4/60 workstation.

1 緒論

近年の集積回路技術の進歩に伴い、集積回路はより広範な分野で使用されるようになっており、故障検査による信頼性の確保が重要なものとなっている。通常、故障検査は、外部入力端子に一連の検査入力を印加した時の外部出力端子での応答を観測することにより行なわれる。効率の良い故障検査を行なうためには、検査入力の故障検出率、あるいはその入力で検出できない故障の同定を行なう必要がある。故障シミュレーション [1] は、故障を仮定された回路のある入力に対する外部出力を計算するものであり、上記のような検査入力の評価の他に、故障辞書の作成や故障した回路の振舞いの分析などさまざまな目的に用いられる。

従来、故障シミュレーションの多くは、故障のモデルとして単一縮退故障 (single stuck-at fault) モデルを採用してきた。単一縮退故障モデルは、扱いが容易であり、かつ、実際に起こる故障 (physical failure) の多くをこのモデルに帰着することが可能である [2]。しかし、実際の回路では、例えば CMOS ゲートの開放故障や遅延故障など、単純な単一縮退故障には帰着させることができない故障が現れることが報告されている [3]。また、単一故障の仮定は小規模の回路では合理的といえるが、数万ゲート規模の回路においては複数の故障が同時に起こる可能性も無視できない。多重縮退故障の性質についての研究報告は [4] が挙げられる程度であり、一般的な性質は示されていない。これは、多重故障の故障シミュレーション (多重故障シミュレーション) の計算量が大きく、現実的な計算時間、記憶量で行なうことが困難であることを理由としている。つまり、効率の良い多重故障シミュレーションを行なうことができれば、多重縮退故障の検出率と、回路や検査入力系列の性質の関係のような多重縮退故障の性質が明らかになり、得るところは多いと考えられる。[5] は、critical path の考え方に基づいた多重故障シミュレーション手法を提案しているが、外部出力まで伝播する多重故障のみをシミュレーションの対象としており、多重故障の性質の解明に用いるには不十分である。本稿では、多重縮退故障の仮定された組合せ回路に対し、共有二分決定グラフ [6] を用いて多重故障シミュレーションを効率良く行なう方法を提案する。本手法は、explicit simulation であり、仮定されたすべての多重故障がシミュレーションの対象となる。

多重故障のシミュレーションにおいては計算量の削減が最大の課題となる。m 重故障の数は、回路規模を n として $O(n^m)$ に達する。これを従来の単一故障を対象とした手法で扱った場合、仮定された故障を記憶するために必要な領域と、各故障回路の出力を計算するために必要な時間は、極めて大きなものになる。したがって、多重故障をコンパクトに表すデータ構造と、高速なシミュレーション手法を考える必要がある。本稿では、演繹法 [1] に基づき、故障集合の内部表現として共有二分決定グラフを用いた故障シミュレーション手法を提案する。本手法では、多重故障を符号化して論理関数の最小項と対応づけることにより、故障集合を論理関数で表す。論理関数の内部表現としては共有二分決定グラフを用いる。故障集合を表現する共有二分決定グラフの大きさは、必ずしも故障集合の大きさに比例せず、符号化の工夫により小さくすることができる。さらに、故障集合は、その本来の性格から相関性が比較的高い。これを共有二分決定グラフで表した場合、集合間の共通部分は、記憶領域上で実際に共有される。したがって、共有二分決定グラフを用いることにより、シミュレーション中に現れる複数の故障集合をコンパクトに表すことができると考えられる。多重故障の符号化の方法が問題となるが、本稿では、共有二分決定グラフとの相性が良い FNT 符号を提案する。

実際にこの手法による多重故障シミュレータを C 言語で実現し、性能評価を行なったところ、線形リストを用いる場合に比べ、格段に少ない領域でシミュレーションを行なうことができ、この手法の有効性が確認された。また、いくつかの回路について、単一縮退故障を対象として生成されたテストセットが多重縮退故障に対しても高い検出率を持つことが、シミュレータを用いた実験により示された。

以下、2 章では、準備として、多重故障モデルと、故障シミュレーション、特に本手法の基本アルゴリズムである演繹法について述べる。3 章では、故障の集合の論理関数による表現とその共有二分決定グラフ表現について、4 章では、多重故障の FNT 符号化について、述べる。5 章では、多重故障シミュレータの実現とそれを用いた実験について述べる。6 章では、結びとして、本研究の成果と今後の展望を述べる。

2 多重故障と故障シミュレーション

2.1 故障シミュレーション

故障シミュレーションは、回路情報 C、回路に仮定される (起こりうる) 故障の集合 F、回路に対する入力パタンの系列 P を入力として、C に F の各故障を仮定した回路 (このような回路を故障回路という) の P に対する出力系列を計算するものであり、次のような目的に用いられる。

1. F に含まれる故障のうち P で検出できる (できない) 故障を識別する。
2. P の故障検出率 (F のうち P で検出できる故障の割合) を求める。
3. P の有効な部分を選んで、検査入力系列をつくる。
4. 故障辞書 (故障とそれを検出する入力パタンの対応表) を作成する。

故障シミュレーションでは、対象とする回路の種類、回路記述のレベル、遅延の扱いについて種々の計算法があり、目的に応じた選択が行なわれる。本稿では、ゲートレベル、零遅延でモデル化された組合せ回路の故障シミュレーションを考える。

さらに、回路に仮定される故障に関して、さまざまなモデルが考えられるが、多くの場合、縮退故障に帰着できるので縮退故障を考えることが多い。縮退故障 (stack-at fault) はゲートの入出力端子の信号値が 0 または 1 に固定される故障として定義され、その固定値 (縮退値) σ によって、σ 縮退故障という。また、回路中の故障は一箇所であるとする単一故障モデル、複数箇所が同時に故障し得る多重故障モデルという分類がある。本稿では、多重縮退故障の故障シミュレーションを考える。

2.2 多重故障

以下では、(多重)故障を f あるいは f_i のように表す。多重故障は複数箇所(故障点)の故障の集合と考えることができ、個々の故障を**要素故障**(prime fault)と呼ぶ。回路中の要素故障の数を M、すべての要素故障の集合を $P = \{p_0, p_1, \cdots, p_M\}$ で表す。この時、m 重故障は要素故障の集合として $f = \{p^1, p^2, \cdots, p^m\}$ のように表現できる。

多重故障シミュレーションにおいては計算量の削減が最大の課題となる。ゲートレベルの組合せ回路を対象とする故障シミュレーションの計算量の上界は、与えられた故障集合と入力パタン集合に対して単純に各故障回路の論理シミュレーションを行う場合の計算量であり、[故障数] × [入力パタン数] × [ゲート数] に比例する。単一縮対故障を対象とする場合、故障数は n に比例し、一つの入力パタンについての故障シミュレーションの計算量の上界は $O(n^2)$ である。m 重縮退故障の場合は、故障の数は、${}_2{}_mC_m$ 個であり、$M = O(n)$ であるから、$m \ll n$ の時、n^m に比例する。したがって、一つの入力パタンについての故障シミュレーションの計算量の上界は $O(n^{m+1})$ である。下界は明らかではないが、これより良い上界を与えるアルゴリズムは知られていない。このように多重故障の計算量は極めて大きく、数千〜数万ゲートの回路をシミュレーションするには、何らかの効率化の手段を考える必要がある。

2.3 演繹法

従来の単一故障を対象とするシミュレーションについては、処理時間を短縮するために種々の計算手法が提案されている。主なものとしては、並列法[7]、単一故障伝播(SFP)[8]、**演繹法**[1]、同時法[9]がある。多重故障を仮定故障とする場合は、各要素故障が複数の故障回路に含まれており、故障回路間の類似性が大きい。効率のよい多重故障シミュレーションを行なうためには、この類似性を利用することが不可欠と考えられる。同時法と演繹法は、故障リストの伝播によりすべての故障回路について一度にシミュレーションを行なっているため、この類似性を考慮したシミュレーションを行なえる可能性がある。本稿では、記憶量の観点から多重故障シミュレーションの基本アルゴリズムとして演繹法を用いることにする。

演繹法(deductive fault simulation)は、以下に挙げる2つの操作を、外部入力側から外部出力に向け繰り返し施して各信号線で検出可能な故障集合を演繹的に求め、最終的に外部出力で検出できる故障の集合を求めることによりシミュレーションを行う手法である。

ゲートにおける故障伝播　ゲート入力で検出できる故障の集合から、ゲート出力で検出できる故障の集合を集合演算を用いて計算する。例として、2入力ANDゲートの場合を考える。入力信号線を x, y、出力信号線を z とし、その正常値(故障のない場合の信号値)をそれぞれ0,1,0とする。また、x, y, z に伝播する、つまり、信号値を反転させるような故障の集合を、それぞれ X, Y, Z とする。この場合、x に故障が伝播してその信号値が1に反転し、y が0に留まる時にのみ、z が1に反転する(故障が伝播する)。すなわち、X に含まれ Y には含まれない故障だけが z に伝播することから、Z は $Z = X \cap \overline{Y}$ なる式で計算できる。この操作は、単一故障と多重故障の両者で共通である。

故障点における故障の挿入　信号線 l に0縮退故障 l_0 と1縮退故障 l_1 が要素故障として仮定されている場合を考える。l の正常値を0、l の入力側および出力側に伝播する故障の集合をそれぞれ L, L' とする。単一故障の場合は、$L' = L \cup \{l_1\}$ により L' が計算される。多重故障が仮定されている場合は、l_0 を含む故障が L に含まれ得るため、$L' = (L \cup \{l_1$を含むすべての故障$\}) \cap \{l_0$を含むすべての故障$\}$ となる。$l_0(l_1)$ を含むすべての故障の集合を、信号線 l の **0(1) 縮退局所故障集合**(site-local fault set)と呼ぶ。

3 故障集合の表現

従来の単一故障を対象とした演繹法による故障シミュレーションでは、故障集合は線形リストで表現されていた。多重故障を線形リストで表現すると、リストが極めて長くなると考えられる。例えば、ISCAS85のベンチマーク回路の一つであるc1908(端子数2436)について、線形リストを用いて演繹法で2重縮退故障のシミュレーションを行なうと、600Mbyte以上の領域が必要になる。計算時間は平均リスト長に比例するので[10]、これも極めて大きくなり、実用的でない。本章では、線形リストに変わって、論理式で故障集合を表現し、さらに内部表現として共有二分決定グラフを用いる手法を提案する。

3.1 故障集合の論理関数による表現

故障の全体集合を $U = \{f_0, f_1, \cdots, f_{N-1}\}$ とし、故障 $f_k \in U$ に、v ビット($\lceil \log N \rceil \leq v$)の互いに異なる符号語 $c_k = c_k^0 c_k^1 \cdots c_k^{v-1} \in \{0,1\}^v$ を割り当てる。U の部分集合 F に対して、特徴関数 $\Phi_F : \{0,1\}^v \to \{0,1\}$ を以下のように定める。

$$\Phi_{\{f_k\}}(x_0, x_1, \cdots, x_{v-1}) = \xi_0 \cdot \xi_1 \cdot \cdots \cdot \xi_{v-1},$$
$$\text{where } \begin{cases} \xi_i = \overline{x}_i & \text{if } c_k^i = 0, \\ \xi_i = x_i & \text{if } c_k^i = 1, \end{cases}$$
$$\Phi_{F \cup G} = \Phi_F + \Phi_G.$$

例えば、$U = \{f_0, f_1, \cdots, f_7\}$ のとき、c_k として k の2進表現を用いれば、

$$\Phi_{\{f_0\}} = \bar{x}_2\bar{x}_1\bar{x}_0, \Phi_{\{f_1\}} = \bar{x}_2\bar{x}_1x_0,$$
$$\cdots, \Phi_{\{f_7\}} = x_2x_1x_0,$$
$$\Phi_{\{f_0,f_1\}} = \bar{x}_2\bar{x}_1\bar{x}_0 + \bar{x}_2\bar{x}_1x_0 = \bar{x}_2\bar{x}_1,$$
$$\Phi_U = 1.$$

となる。特徴関数の定義より、

- $\Phi_{F(c_k)} = 1 \Leftrightarrow f_k \in F$,
- $\Phi_{F \cup G} = \Phi_F + \Phi_G$, $\Phi_{F \cap G} = \Phi_F \cdot \Phi_G$,
  $\Phi_{\overline{F}} = \overline{\Phi}_F \cdot \Phi_U$.

であり、Φ_F は F の membership を表現している。集合演算を論理演算に置き換えることにより、演繹法による故障シミュレーションを論理関数処理として実現できる。

3.2 論理関数の共有二分決定グラフ表現

本節では、故障集合と対応づけられた論理関数を計算機内部で表現するデータ構造について考える。以下では、このようなデータ構造を内部表現と呼ぶ。論理関数の内部表現として、真理値表表現、オンセット表現および積和型表現(キューブ表現)が良く知られている。さらに、比較的新しいものとして共有二分決定グラフ表現(Shared Binary Decision Diagrams, SBDD's)[6]がある。真理値表で表現する場合には、真理値表の中の'1'の数の割合に関わりなく一定の領域を必要とする。オンセット表現は、真理値表における'1'の位置を整数と対応づけることにより、線形リストで表現することができる(以下、オンセット表現のことをリスト表現という)。すでに述べたように、線形リストで多重故障の集合を表した場合、リストが非常に長くなり領域および計算量の観点から実用的でない。本稿では、論理関数の内部表現として、共有二分決定グラフを採用する。

共有二分決定グラフは多出力論理関数を効率良く表現するために、二分決定グラフ(Binary Decision Diagrams, BDD's)[11]を拡張したものである。二分決定グラフはグラフによる論理関数の表現法である。これは、シャノンの展開を再帰的に繰り返すことによって得られる二分木のグラフに対し、冗長なノード('0'エッジと'1'エッジが同じサブグラフを指しているノード)の削除と、同型なサブグラフの共有を行なうことにより、簡約化したものである。例えば、図1(a)に示す二分木を簡約化すると、(b)のような二分決定グラフが得られる。二分決定グラフは、次の特徴を持つ。

1. 多くの実用的な論理関数を、現実的な記憶量で表現できる[12]。
2. 入力変数の順序を固定すれば、論理変数に対してグラフの形が一意に定まる。
3. 関数同士の論理演算が、グラフのノード数にほぼ比例する時間で行なえる。

二分決定グラフを改良して、複数の関数を表すグラフの間においてもサブグラフの共有を行なったものを共有二分決定グラフと呼ぶ。共有二分決定グラフは、二分決定グラフの特徴に加えて以下の特徴を持つ。

1. 複数の関数を、少ない記憶量で同時に表せる。
2. 関数の一致判定がポインタの一致判定だけで行なえる。

図1(c)に共有二分決定グラフの例を示す。各変数を表すノード数の最大の値を共有二分決定グラフの幅と呼ぶ。図1(c)に示した共有二分決定グラフの幅は3である。共有二分決定グラフの大きさおよび幅は、変数の順序によって大きく左右されることが知られている[13]。故障集合を表す共有二分決定グラフの変数順序については4.3節で述べる。

各信号線で検出される故障の集合は、前段のゲートから伝播してきた故障集合から計算されるものであるから、各信号線で検出できる故障の集合間の相関性が比較的高い。また、特に多重故障の場合は、故障の現れ方が良く似た回路が多く、これら回路間の対応する信号線で検出される故障の集合間の相関性も高いと考えられる。共有二分決定グラフで故障集合を表した場合、かなりの部分を複数の集合間で共有できる可能性があり、領域および計算量の削減が期待できる。

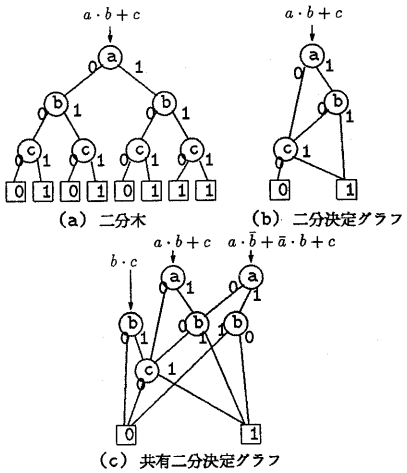

図 1: 共有二分決定グラフ

4 多重故障の FNT 符号化

4.1 多重故障の符号化

故障集合を論理関数で表現するためには、故障の符号化、すなわち故障 f_k の符号語 c_k への対応づけが必要となる。符号化は、故障集合を表す論理関数の形を決め、さらに共有二分決定グラフの形を、ひいては必要となる記憶量を定めることになる。以下、いくつかの符号化をあげ、その得失を吟味する。回路中の要素故障の数を M、最大故障多重度を m とし、m 重以下の多重度を持つ故障を論理関数で表すことを考える。

詰込み符号化 (Stuffed Coding) すべての故障の集合を U で表す。U の各故障に、0から始まる続き番号を割り当て、その2進表現を故障の符号語とする方法を詰込み符号化と呼ぶ。この符号化は最小の符号長を与え、従って、論理変数の数も最小となる。しかし、故障相互の関係が符号に反映されないため、特徴関数およびそれを表現する共有二分決定グラフが複雑なものになり、局所故障集合の構成に多大な時間が要求される。

BPF 符号化 (bit per fault coding) この符号化では、各故障は M ビットの符号語で表現され、各ビットが別々の要素故障に対応する。故障 f_k に対する符号語を c_k とし、c_k の第 i ビットを c_k^i で表すと、f_k が要素故障 p_i を含む時(その時に限り)、$c_k^i = 1$ とする。例えば、$P = \{p_1, p_2, p_3\}$ の時、単一故障 $\{p_1\}$ は 100 に、二重故障 $\{p_2, p_3\}$ は 011 に、3重故障 $\{p_1, p_2, p_3\}$ は 111 に、それぞれ符号化される。論理変数のうち m 個以下が'1'となった時に'1'となるような対称関数と特徴関数との論理積をとることにより、m 重以下の故障を容易に切り出すことができる。一方、最大故障多重度 m に関わりなく、M 個の論理変数が必要であるため、m が小さい場合には m 重以上の故障に対応する多くの符号語が無駄になる。

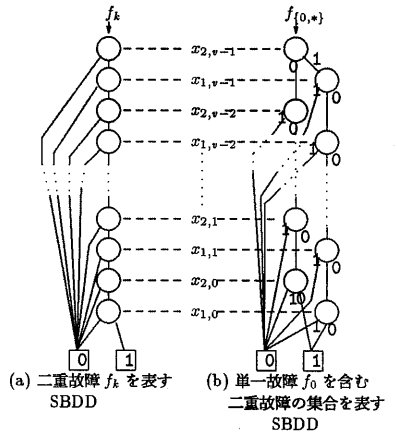

図 2: FNT 符号による故障集合の表現

FNT 符号化 (fault number tuple coding) m 重故障 $\{p^1, p^2, \cdots, p^m\}$ は、要素故障の m 個組 (m-tuple) として $(p^1, p^2, \cdots, p^m)$ のように表現できる。p^i $(1 \leq i \leq m)$ を第 i 要素故障と呼ぶ。m 重未満の故障も、$p^1 = p^2 = \cdots$ とすることにより m 個組で表現できる。FNT 符号では、要素故障の組を符号化する。

$0 \leq k_i < M$, $u = \lceil \log M \rceil$ に対し、$f_k = (p_{k_1}, p_{k_2}, \cdots, p_{k_m})$、$\gamma_{k_i}$ を k_i の u ビット 2 進表現とする。この時、f_k を、γ_{k_i} の連接、すなわち $c_k = \gamma_{k_1}\gamma_{k_2}\cdots\gamma_{k_m}$ なる $(u \times m)$ ビットの符号語 c_k に対応づける。例えば、$m = 2$, $f_k = (p_1, p_6)$, $u = 3$ の時、$\gamma_1 = 001$, $\gamma_6 = 110$ であるから、$c_k = 001110$ である。すなわち、各要素故障に対して符号語が定まれば、故障の集合を $(u \times m)$ 変数論理関数で表現できる。上記の例では、第 i 要素故障の第 j ビットに対する変数を $x_{i,j}$ として、$\Phi_{\{f_0\}} = \overline{x}_{1,1}\overline{x}_{1,2}x_{1,3}x_{2,1}x_{2,2}\overline{x}_{2,3}$ となる。単一の多重故障からなる故障集合 $\Phi_{\{f_k\}}$ は、図 2(a) のように、幅 1、ノード数 $u \times m$ の共有二分決定グラフで表される。

本手法の特徴は、局所故障集合がコンパクトに表現できるということである。例えば、$m = 2$ の時、単一故障 p_0 が発生する端子で挿入すべき故障の集合は $\{(p_0, p) \mid p \in P\} \cup \{(p, p_0) \mid p \in P\}$ であり、$2N - 1$ 個の故障を含む。この故障集合は、特徴関数 $\overline{x}_{1,1}\overline{x}_{1,2}\cdots\overline{x}_{1,u} + \overline{x}_{2,1}\overline{x}_{2,2}\overline{x}_{2,u}$ で表すことができ、対応する共有二分決定グラフは、図 2(b) のようになる。一般に、局所故障集合を表す共有二分決定グラフの大きさは $m \times u = (m \log M)$ であり、線形リストを用いた場合のリスト長 $O(M^{m-1})$ に比べ、領域の面で有利である。

本稿では、多重故障の故障番号の符号化として、FNT 符号化を用いる。

4.2 故障集合の正規化

FNT 符号では、多重故障を要素故障の組として表現しているため、要素故障の順列に対応する複数の符号語が一つの故障に割り当てられ得る。例えば、3 重故障 $\{p, p', p''\}$ は次の 6 つの 3 つ組のいずれによっても表現できる。

$$(p, p', p''), (p, p'', p'), (p', p, p''),$$
$$(p', p'', p), (p'', p, p'), (p'', p', p).$$

同様に、2 重故障 $\{p, p'\}$ に対する等価な 3 つ組表現は、

$$(p, p, p'), (p, p', p), (p', p, p),$$
$$(p, p', p'), (p', p, p'), (p', p', p).$$

である。組表現におけるこのような一意性の欠如は、故障集合中の故障を数え上げる際に障害となるだけでなく、複雑な特徴関数と大きな共有二分決定グラフをもたらす。故障挿入の時点で各故障に対してただ一つの組表現を定めたとしても、補集合演算により他の組表現が集合の要素となり得るため、一意性を保つことは困難である。我々の手法では、組表現の一意性を保つために、まず、fault tuple の正規形を定義する。

定義 (fault tuple の正規形) 要素故障の m 個組 $f = (p^1, p^2, \cdots, p^m)$ において、すべての $2 \leq k \leq m$ について $p^{k-1} < p^k$ または $p^{k-1} < p^k$ or $p^1 = \cdots = p^{k-1} = p^k$ が成り立つならば (またその時に限り)、f は正規である。ただし、要素故障間に全順序を仮定する。 □

つまり、正規な fault tuple は以下を満たす。

$$\begin{aligned}
& p^1 < p^2 < p^3 < \cdots < p^{m-1} < p^m, \\
\text{or}\quad & p^1 = p^2 < p^3 < \cdots < p^{m-1} < p^m, \\
\text{or}\quad & p^1 = p^2 = p^3 < \cdots < p^{m-1} < p^m, \\
& \vdots \\
\text{or}\quad & p^1 = p^2 = p^3 = \cdots = p^{m-1} < p^m, \\
\text{or}\quad & p^1 = p^2 = p^3 = \cdots = p^{m-1} = p^m).
\end{aligned}$$

以上の定義に従えば、m 重以下の任意の故障に対して正規な組表現が必ず一つ (だけ) 対応することが、ただちに導かれる。すなわち、上記の条件はすべての正規な組表現の集合を定める。この条件は論理関数 (以下では、R で表す) として表現できる。

FNT 符合化のもう一つの問題点は、同一の故障点における 0 縮退故障と 1 縮退故障を同時に含むような多重故障 (inconsistent multiple fault) が故障集合に含まれ得るという点である。このような矛盾した故障組表現も、論理関数 (以下、S で表す) を用いて除去することができる。

以上に述べたことから、$\Phi_U = R \cdot S$ とすることにより、正規かつ無矛盾な故障組表現に対応する符号語だけを対象に故障シミュレーションを行なうことができる。

4.3 変数の順序

論理関数を表す共有二分決定グラフの大きさは変数の順序に依存する。多重故障シミュレーションの内部データ構造として共有二分決定グラフを用いる場合、変数の順序は (故障集合そのものではなく) 正規化関数 R を表すグラフの大きさに大きな影響を与える。

第 i 要素故障の第 j ビットに対する変数を $x_{i,j}$ とする。単純な順序付けは、

$$\underbrace{x_{1,1}, x_{1,2}, \cdots\cdots, x_{1,u}}_{\text{bits for 1st prime fault}}, \quad \underbrace{x_{2,1}, x_{2,2}, \cdots\cdots, x_{2,u}}_{\text{bits for 2nd prime fault}},$$
$$\cdots\cdots, \quad \underbrace{x_{m,1}, x_{m,2}, \cdots\cdots, x_{m,u}}_{\text{bits for } m\text{-th prime fault}}.$$

表 1: 実験結果 (50 パターンに対する平均)

回路	ゲート数	故障点数	多重度	仮定故障数	ノード数	時間 (秒)
adder4	24	69/90	4	1.426×10^7	6.980×10^4	34.33
adder16	96	273/354	3	2.698×10^7	7.047×10^4	40.31
mult4	88	237/293	3	1.764×10^7	8.579×10^4	88.21
mult8	400	1049/1289	2	2.201×10^6	2.521×10^4	40.35
dec8	17	66/87	4	1.191×10^7	3.301×10^4	15.43
enc8	22	77/89	4	2.225×10^7	3.258×10^4	22.49
c17	6	17/25	1	3.400×10^1	1.326×10^1	0.01
			5	2.421×10^5	6.278×10^3	1.26
			10	4.211×10^7	7.497×10^4	44.91
			13	1.123×10^8	1.204×10^5	89.07
c432	160	432/539	2	3.732×10^5	2.864×10^3	3.84
c499	202	499/683	2	4.980×10^5	5.801×10^3	8.08
c880	383	880/1198	2	1.549×10^6	1.594×10^4	9.15
c1355	546	1355/1683	2	3.672×10^6	2.012×10^4	29.55
c1908	880	1908/2436	2	7.281×10^6	4.072×10^4	56.98
c2670	1193	2670/3642	2	1.426×10^7	5.807×10^4	58.27
c3540	1669	3540/4680	2	2.506×10^7	7.467×10^4	135.97
c5315	2307	5315/6994	2	5.650×10^7	1.441×10^5	155.50
c6288	2416	6288/7280	1	1.258×10^4	6.048×10^2	5.88
c7552	3512	7552/9971	1	1.510×10^4	1.017×10^3	2.64

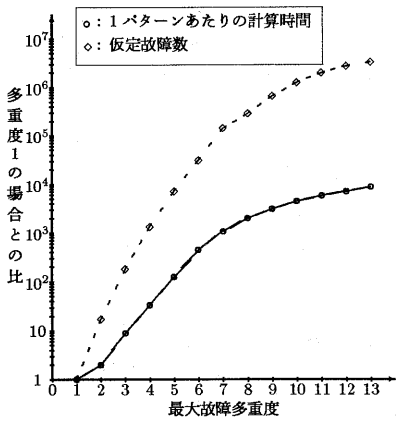

図 3: 1パタンあたりの計算時間と多重度の関係 (c17)

であるが、この順序では R を表すグラフのサイズは回路規模に関して指数的に大きくなる。我々が用いるのは、次のような順序づけである。

$$\underbrace{x_{1,1}, x_{2,1}, \cdots\cdots, x_{m,1}}_{\text{1st bits of prime faults}}, \quad \underbrace{x_{1,2}, x_{2,2}, \cdots\cdots, x_{m,2}}_{\text{2nd bits of prime faults}}, \quad \cdots\cdots, \quad \underbrace{x_{1,u}, x_{2,u}, \cdots\cdots, x_{m,u}}_{u\text{-th bits of prime faults}}.$$

この順序付けでは、R は $2^m m \log s$ すなわち回路規模の対数に比例する大きさの共有二分決定グラフで表現される。

5 多重故障シミュレータ

5.1 シミュレータの実現と性能評価

前章までに述べた手法に基づき、多重故障シミュレータを Unix 上の C 言語で実現し、Sun4/60(実装記憶 16MB) 上で性能評価を行なった。対象回路はゲート数 6 から 3512 までの組合せ回路であり、すべての端子に 0/1 縮退故障が起こり得ることとして、50 ランダムパタンについて、与えられた多重度以下のすべての多重縮退故障のシミュレーションを行なった。故障のドロップは行なっていない。結果を表 1 に示す。「回路」欄は回路名を示し、adder$n$ は n ビット桁上げ伝播加算器、mult$n$ は n ビット配列型乗算器、enc8 は 8 ビットエンコーダ、dec8 は 8 ビットデコーダ、c で始まる回路は ISCAS85 のベンチマーク回路 [14] である。限定されたものではあるが、等価故障解析を行なって区別される故障点の数を減らしており、「故障点数」欄に [故障点数]/[端子数] を示す。「仮定故障数」欄は与えられた「多重度」以下の多重故障の総数である。「ノード数」は、少なくとも 1 つの外部出力で検出できる故障の集合を表す共有二分決定グラフのサイズを示す。「ノード数」は、シミュレーションに必要な領域量を直接示すものではなく、動的にはより多くの領域を必要とする。実行時の共有二分決定グラフのノード数を最大 50 万ノード (10.5MB) に制限しており、各回路とも、表に示した以上の多重度では、実行途中にノード数が 50 万を越えたため異常終了した。「時間」は、1 パターンあたりのシミュレーション時間である。

2 つの回路を除き、少なくとも 2 重までの故障のシミュレーションを行なうことができている。例えば、リスト表現では単純に見積もると 600MB 必要となるシミュレーション (c1908、2 重故障) が、10MB 以内で行なえており、共有二分決定グラフが多重故障シミュレーションのデータ構造として有効であることを示している。また、実験を行なった範囲では 1 パタンにつき最大でも 3 分以内でシミュレーションを行なうことができ、多重故障の性質の把握を目的とするならば、十分な性能を持つといえる。

多重故障シミュレーションに必要な時間は、回路の大きさや多重度だけでなく、回路の性質やパタンにも強く影響される。ほぼ同規模の回路である c880 と c1355、mult8 を比較することにより、回路の性質が計算時間に大きな影響を与えることがわかる。図 3 に、同一の回路 (c17) で多重度を高くした場合の 1 パタンあたりの計算時間の変化を示す。計算時間の増加は仮定故障数の増加に比べると非常に小さいといえ、本手法の有効性を示している。

5.2 多重故障の検出率

多重故障の性質の一端を明らかにするために、実現した多重故障シミュレータを用いて、ランダムパタンおよび単一故障検査入力による多重縮退故障の検出率の評価を行なった。表 2 に 500 個のランダムパタンによる 2 重故障 (単一故障を含む) の検出率を示す。比較のため、同一のパタンによる単一故障の検出率を併記した。単一故障の場合に比べ、2 重故障の場合は検出率は高くなるが、回路に仮定された故障の総数が多いため、多くの故障が未検出となっている。表 3 に、単一縮退故障を対象として生成されたテストパタン (冗長でないすべての単一縮退故障を検出する) による 2 重故障の検

表 2: 500 ランダムパタンによる検出率

回路	単一故障		2 重故障†	
	検出率	未検出故障数	検出率	未検出故障数
c432	97.69%	20(10)‡	99.94%	227
c499	98.30	17 (8)	99.97	165
c880	91.42	151 (0)	99.24	11780

† 単一故障を含む
‡ ( ) は冗長故障数

表 3: 単一故障検査入力による 2 重故障 † の検出率

回路	パタン長	検出率	未検出故障数
c432	77	99.98%	67
c499	89	99.99	36
c880	73	100.00	0

† 単一故障を含む

出率を示す。ランダムパタンの場合と比べ、短いパタン長でより多くの 2 重故障を検出している。冗長な単一故障の数を考慮すれば、未検出故障のうち少なくとも半数程度は冗長故障である可能性が高い。すなわち、多重故障を考慮せずにテスト生成を行なった場合にも、多くの多重故障の検出を期待できると考えられる。ただし、ここで用いたパタンはパタン長の圧縮をほとんど行なっておらず、このことが検出率に寄与していることも考えられる [4]。

6 結論

本稿では、論理回路に仮定された多重縮退故障の集合を共有二分決定グラフを用いて表現する方法と、これを用いた多重故障シミュレーションの手法を提案した。多重故障のシミュレーションは、扱う故障の数が極めて多くなるため、従来の手法では多大な領域と計算時間を必要とする。本稿で提案した手法では、故障集合を論理関数で表し、さらに、論理関数の内部表現として共有二分決定グラフを用いることにより、効率の良いシミュレーションを行なう。本手法に基づく多重シミュレータを実現し、実験を行なった結果、その有効性が示された。特に故障多重度あるいは回路規模が大きく、仮定される故障が多いほど、故障集合間の共有の効果が顕著になる。

多重故障の要求する領域量と計算量は非常に大きく、現状での数万ゲート規模の回路への適用は困難である。シミュレーションに必要な領域のほぼすべては故障集合を表す共有二分決定グラフを格納する領域であり、また、論理演算に要する時間もグラフの大きさの関数となっている。したがって、グラフのノード数を削減することが効率化に直接つながる。グラフのノード数は、回路やパタンの性質にも左右されるが、特に故障集合の符号化に用いる論理変数の数 (あるいは仮定故障の数) に強く影響される。この見地から、実験では限定的な等価故障解析を行なって故障の数を減らしたが、さらに、ゲートの性質を用いた等価故障解析 (AND ゲートの入力端子の 0 縮退故障は出力端子の 0 縮退故障で代表するなど) や回路の分割による仮定故障の限定などを行なうことによりグラフのノード数を削減できると考えている。また、故障点の隣接性を考慮した符号化により、故障変数の数は変わらないものの、グラフを小さくできる可能性がある。また、本研究で提案した故障集合の表現手法は、故障シミュレーションだけでなく、テスト生成や診断にも用いることができると考えられる。これらへの応用をも今後の課題としたい。

謝辞

多重故障シミュレータの実現に不可欠な、共有二分決定グラフに基づく論理関数処理プログラムの提供者である湊真一氏 (NTT) に感謝致します。また、御討論頂いた本学矢島研究室の諸氏に感謝致します。

参考文献

[1] M. A. Breuer, A. D. Friedman. *Diagnosis & Peliable Design of Digital Systems.* Computer Science Press, 1976.

[2] A. D. Friedman M. Abramovichi, M. A. Breuer. *Digital Systems Testing and Testable Design.* Computer Science Press, 1990.

[3] J. A. Abraham. *Fault Modeling in VLSI.* In Advances in CAD for VLSI, vol. 5, VLSI Testing, 1986.

[4] J. L. A. Hughes. *Multiple Fault Detection Using Single Fault Test Sets.* IEEE Trans. on CAD, vol. 7, no. 1, May 1988.

[5] S. Maker, E. McCluskey. *The Critical Path For Multiple Faults.* In Proc. of Int. Conf. CAD (ICCAD-89), pp. 162–165, Nov. 1989.

[6] S. Minato, N. Ishiura, S. Yajima. *Shared Binary Decision Diagram for Efficient Boolean Function Manipulation,* In Proc. of 27th Design Automation Conf., pp. 52–57, June 1990.

[7] 樹下, 浅田, 唐津. *VLSI* の設計 *II*, 岩波書店, 1987 年 10 月.

[8] P. Goel, T. E. Rosser, T. J. Stroh, E. B. Eichelberger. *LSSD Fault Simulation Using Conjunctive Combinational and Sequential Method.* In Proc. of Int. Test Conf., pp. 371–376, 1980.

[9] E. G. Ulrich, T. Baker. *The Concurrent Fault Simulation of Nearly Identicl Digital Networks.* In Proc. of 10th DA Workshop, vol. 6, pp. 145–150, June 1973.

[10] 武智. 故障集合の共有二分決定図表現を用いた演繹故障シミュレーション. 京都大学工学部情報工学教室特別研究報告書, 1989.

[11] R. E. Bryant. *Graph-Based Algorithms for Boolean Function Manipulation.* In IEEE Trans. on Comp., vol. 35, no. 8, pp. 677–691, Aug. 1986.

[12] S. Yajima, N. Ishiura. *A Class of Logic Functions Expressible by a Polynomial-Size Binary Decision Diagrams.* In Proc. of Synthesis and Simulation Meeting and Int. Interchange (SASIMI '90), Oct. 1990.

[13] H. Fujisawa, M. Fujita, N. Kawato. *Evaluation and Improvements of Boolean Comparison Method Based on Binary Decision Diagrams.* In Proc. of Int. Conf. CAD (ICCAD-88), pp 2–5, Nov. 1988.

[14] F. Braglez, H. Fujiwara. *A Neutral Netlist of 10 Combinational Benchmark Circuits and A Target Translator in FORTRAN.* In Proc. of Int. Symp. Circuits and Systems, June 1985.