計算機アーキテクチャ　98－2
設　計　自　動　化　65－2
（1993. 1. 21）

不良組合せ回路の単一修正方法

伊藤 雅樹　高嶺 美夫　清水 嗣雄

(株) 日立製作所 中央研究所
〒185 東京都国分寺市東恋ヶ窪1-280

あらまし

不良を含む組合せ回路を自動修正する方法を提案する。本稿では、一箇所のみを修正することにより正しい回路が得られる場合を対象とする。処理は、修正位置の決定、修正条件の計算、修正の実施、修正の確認の四段階からなる。修正位置の決定では、論理関数を変更することにより正しい回路が得られるような内部信号線を修正位置と決める。修正条件の計算では、修正位置の信号線が満たすべき条件を計算する。修正の実施では、修正位置の論理関数が変わるように回路を変更し、修正の確認では、回路変更後の論理関数が修正条件を満たしているか否かを判定する。全ての処理を論理関数処理により行うため、二分決定グラフを用いて効率的に実現できる。

和文キーワード　論理設計　論理不良　自動修正　組合せ回路　論理関数

A Single Rectification Method of An Erroneous Combinational Circuit

Masaki Ito, Yoshio Takamine, and Tsuguo Shimizu

Central Research Laboratory, Hitachi Ltd.
1-280, Higashi-koigakubo, Kokubunji-shi, Tokyo 185, Japan

Abstract

This paper presents a new method to rectify an erroneous combinational circuit. It finds a line to be modified such that a correct circuit can be obtained by replacing the logic function of the line and gets a necessary and sufficient condition for it to get a correct circuit. Then, the circuit is modified to meet the condition. This method can be applied in the case that a correct circuit can be obtained by modifying only one place in the erroneous circuit. Since this method utilizes only logic operations on logic functions, it can be implemented efficiently using binary decision diagram.

英文 key words　Logic Design, Logic Error, Error Correction, Combinational Circuit, Logic Function

1. はじめに

VLSIの開発において、論理検証により設計に不良が混入していることが判明した場合、不良を修正しなければならない。現状では、不良の修正は自動化がなされておらず、全て手作業で行われている。本稿では、組合せ回路を対象とし、一箇所のみを修正することにより正しい回路が得られる場合に適用可能な自動修正アルゴリズムを提案する。

不良の修正は、修正位置の決定、修正の条件の計算、修正の実施の三段階でなされる。修正位置の決定では、回路内のどの信号線を修正するかを決定し、修正の条件の計算では、その信号線をどのように直さなければならないかを表わす条件を求める。修正の実施では、修正位置を修正の条件を満たすように変更する。

これまで、不良位置指摘、不良修正の方法として、[1]-[4]などが提案されている。[1]と[2]は単一出力回路を対象としており、多出力回路を扱うためには各出力ごとに回路を分割して考えなければならない。[3]では、多段回路の簡単化手法であるトランスダクション法[5]のうち、error compensationを用いた不良修正法が提案された。トランスダクション法では、各信号線の許容関数集合を外部出力側から順次求めて行かなければならないが、再収斂する分岐に不良が影響すると、再収斂ゲート以降の許容関数集合が計算できなくなることがある。[4]ではブール単一化を用いた再設計手法が提案されたが、回路内での修正位置の具体的な指摘方法が示されていない。

本稿で提案する方法の特徴を次に記す。

・多出力回路を扱える。
・修正位置の決定にはブール単一化を応用。
・修正条件は位置決定に用いる論理関数を流用。
・修正の実施はトランスダクション法に従う。

以下、第2章で本稿で提案するアルゴリズムを解説し、その効率的実現方法を第3章で説明する。第4章では計算例を示し、評価結果を第5章で示す。

2. アルゴリズム

2.1 定義

本アルゴリズムが対象とする回路は組合せ回路のみである。以後、単に回路というときは、組合せ回路を意味する。また、論理式では、論理否定、論理積、論理和を表わす記号として、それぞれ¯、・、+を用いる。

回路の各外部入力は入力変数と呼ぶそれぞれ異なる論理変数x1, x2, …, xnに対応付けられる。回路の信号線gはgと外部入力の間の部分回路の実現する、入力変数上の完全指定の論理関数に対応付けることができる。この論理関数を信号線gの信号線関数(Signal Function)と呼び、SF[g]と書く。

関数仕様(Functional Specification)とは、各信号線が満たさなければならない条件を記述した不完全指定の論理関数である。信号線gの関数仕様をFS[g]と書く。外部出力の関数仕様は、通常外部仕様と呼ばれ、本アルゴリズムの入力として与えられる。不完全指定の論理関数は、完全指定の論理関数の集合を表わすため、信号線gが関数仕様を満たすとは、SF[g]がFS[g]に含まれることである。

与えられた関数仕様を満たさない外部出力(不良外部出力と呼ぶ)が少なくとも一つ存在するとき、回路に不良が存在するという。不良が存在する回路を不良回路と呼ぶ。信号線gの信号線関数SF[g]を、完全指定の論理関数RF[g]に置き換えた結果の回路に不良が存在しないとき、gは単一修正可能であるといい、RF[g]をgの修正関数(Rectification Function)という。SF[g]をどのような論理関数に置き換えても正しい回路が得られないとき、gは単一修正不能であるという。

信号線gが単一修正可能であるとき、gの修正関数は一般には複数あり得る。考えられる全ての修正関数の集合を表現する一つの不完全指定の論理関数こそが、gの関数仕様FS[g]にほかならない。

以上の修正関数、関数仕様は、トランスダクション法[5]の許容関数、MSPF(Maximum Set of Permissible Functions)を不良回路に対して定義したものである。

2.2 アルゴリズムの詳細

本稿で提案するアルゴリズムは、単一修正可能な信号線を探索し、その関数仕様を求め、関数仕様を満たすように、回路を部分的に変更する。具体的なアルゴリズムsingle_rectifyを図2.1に示す。入力として、回路CKTと各外部出力の関数仕様(外部仕様)FS_of_PO、及び不良修正モデルの集合ERMが与えられる。不良修正モデルとは、関数仕様を満たすべく回路を修正する際、具体的にどのように回路を変更するかを記述したモデルである。詳細は後述する。

```
1: single_rectify(CKT, FS_of_PO, ERM) {
2:   CKT中の全信号線の信号線関数を求める;
3:   for (CKT中の各信号線g) {
4:     if (single_rectifiable(CKT, g))
5:       modify(CKT, g, ERM);
6:   }
7: }
```

図2.1 単一不良修正アルゴリズム

まず、全ての信号線の信号線関数を求め、保持しておく。次に、回路の信号線を一つづつ取り上げ、単一修正可能性判定処理single_rectifiableを行う。single_rectifiableは、gが単一修正可能であると

き、gの関数仕様FS[g]を求めて真を返し、単一修正不能であるときは偽を返す。modifyでは、FS[g]を用いて、gが与えられた不良修正モデル集合ERMによって修正可能か否かを判定し、可能であれば修正例を提示する。詳細を次節以下に解説する。

2.3 単一修正可能性判定

単一修正可能性判定のアルゴリズムを図2.2に示す。

```
1: single_rectifiable(CKT, g){
2:   gに論理変数eを割り当てる;
3:   CKT中の全信号線の信号線関数を求める;
4:   gの仕様決定関数SG[g]を計算;
5:   if ((SG[g](e=0)・SG[g](e=1))が恒偽) {
6:     FS[g] = (SG[g](e=1), SG[g](e=0));
7:     return TRUE;
8:   } else return FALSE;
9: }
```

図2.2 単一修正可能性判定

まず、gにどの入力変数とも異なる変数eを割り当て、各信号線の変数eを含む信号線関数を求める。ただし、必ずしも全ての信号線にeが影響するわけではないので、eを含まない信号線関数をもつ信号線も存在する。

次に、仕様決定関数(Specification Generator)SG[g]を計算する。仕様決定関数とは、全ての外部出力の信号線関数の関数仕様に対する包含性を表わす論理関数であり、次のように定義する(Σは論理和)。

$$SG =_{def} (\Sigma_{po}\bar{}(SF[po] \in FS[po])) \quad (式1)$$

すなわち、SG[g]が恒偽であることが、回路に不良が存在しないことの必要十分条件である。SG[g]は、一般にはeと入力変数に依存する。eはgに割り当てた変数であるから、実際には入力変数の論理関数となるべきものである。したがって、「SG[g]が恒偽になるような入力変数上の論理関数eが存在する」が「gが単一修正可能である」と同値である。SG[g]が恒偽になるような論理関数eが存在するか否かは、

$$SG[g](e=0) \cdot SG[g](e=1) \quad (式2)$$

が恒偽か否かによって判定できる[4]。(式2)が恒偽でない場合は、eがどのような論理関数であっても、SG[g]の値が1となるx1, x2, …, xnの値の組合せが存在することを表わすため、単一修正不能である。逆に恒偽であれば、SG[g]の値が1となる場合は常にeの値に依存することを表わす。このときは、以下に示すようにgの修正関数を構築できるので、gは単一修正可能である。

以上により、gが単一修正可能であると判定されたとき、gの関数仕様FS[g]を求める。FS[g]はSG[g]が恒偽となるような全ての論理関数eを表わす不完全指定の論理関数である。これは、次の3条件が同時に成立するような論理関数である。

(1) SG[g]が0となるようなx1, x2, …, xnの値の組合せに対しては、FS[g]はdon't care。
(2) SG[g]がeとなるようなx1, x2, …, xnの値の組合せに対しては、FS[g]は0。
(3) SG[g]が~eとなるようなx1, x2, …, xnの値の組合せに対しては、FS[g]は1。

ここで、(式2)が恒偽であるから、SG[g]が1となるようなx1, x2, …, xnの値の組合せは存在しない。したがって、SG[g](e=1)が1となる入力変数の値の組合せに対しては0、SG[g](e=0)が1となる入力変数の値の組合せに対しては1であるような論理関数が、SG[g]を恒偽とする論理関数であり、この他にはない。そこで、SG[g](e=1)と、SG[g](e=0)の組がFS[g]を表わすと考えてよいので、この組をgの関数仕様FS[g]として保持する。

2.4 修正の実施

修正の実施では、具体的に回路を変更し、不良の修正を試みる。図2.3に、全体の流れを示す。

```
1: modify(CKT, g, ERM) {
2:   for (ERMの各要素erm) {
3:     ermに基づきCKTを変更しnew_CKTを得る;
4:     new_CKTでのSF[g]を計算;
5:     if (SF[g] ∈ FS[g])
6:       output(g, erm, new_CKT);
7:   }
8: }
```

図2.3 修正の実施

修正の実施は、与えられた不良修正モデルの集合ERM中の不良修正モデルermが表わす回路の変更、修正の確認、結果の出力という三つのステップを繰り返す。ここで、不良修正モデルermには、次のような制限が科される。まず、ermが複雑であると、修正に多大な時間を要してしまい、設計し直す方が計算コストが低くなる可能性がある。よって、不良修正モデルは極力簡単でなければならない。次に、ermが表わす回路変更により信号線関数が変化するが、変化が外部出力へ影響する際、必ずgを通らなければならない。例えば、gにNOTゲートを挿入する(既存の場合は削除する)、gを出力とするゲートの種類を変更する、そのゲートの入力信号線を変更するなどがこの条件を満たす。

後者の条件を満たせば、修正の確認は回路全体の論理検証を必要とせず、SF[g]とFS[g]を用いることにより、局所的に計算できるため高速に実施できる。

全てのermの試行が終了しても、正しい回路が一度も得られない場合は、本手法では修正できない。

3. 実現方法

3.1 信号線関数の計算

信号線関数の計算は、single_rectifyの2行めと、single_rectifiableの3行めに現われる。いずれにおいても、記号シミュレーションを用いれば、容易に計算できる。ただし、前者では、全ての信号線の信号線関数が必要なため、外部入力側から全てのゲートを評価しなければならないが、後者では、変数eの影響の及ぶ範囲のみをイベント法により計算すればよい。

3.2 信号線の選択

single_rectifyの3行めで、単一修正不能であると容易に判定できる信号線は、single_rectifiableの処理対象からはずすことにより、全体の処理量を大幅に削減できる。本節では、single_rectifiableの処理対象信号線の選択方法を説明する。

3.2.1 網羅コーン回路

単一修正可能とは回路全体の中で一つの信号線の信号線関数を変更するのみで正しい回路が得られるということである。したがって、変更すべき信号線は全ての不良外部出力のコーン回路の交わりの中になければならない。この範囲を網羅コーン回路と呼ぶことにすると、gが網羅コーン回路に含まれない場合は、即座に単一修正不能であると判定できる。

3.2.2 信号線の選択順

信号線gが属するファンアウトフリー領域の茎sの単一修正可能性と、gの単一修正可能性の関係を考える。SF[g]を他の関数に変更したとき、この変更の影響は必ずsを通る。すなわち、SF[g]の変更はSF[s]の変化を引き起こし、しかも外部出力へはSF[s]の変化を通してのみ影響する。いま、sが単一修正不能とすると、SF[s]をどのように変更しても正しい回路を得ることはできない。したがって、SF[g]をどのように変更しても正しい回路が得られないことは明らかであり、gもまた単一修正不能であることが分かる。

一方、sが単一修正可能であることが既知の場合は、FS[s]は既に計算されている。FS[s]は、正しい回路を得るための信号線sにおける必要十分条件を表わすから、gの単一修正可能性判定は、sをsと外部入力の間の部分回路の外部出力、FS[s]をその外部仕様とみなして行えばよい。すなわち、single_rectifiableの3行めの記号シミュレーションでは、sまででイベントの伝搬を終了し、sにおける信号線関数SF[s]を求め、このSF[s]とFS[s]を用いて、(式1)を計算することにより、gの仕様決定関数SG[g]を求めることができる。

以上から、外部出力に近い信号線から単一修正可能性を判定することにより、大幅に処理量を削減できることがわかる。外部出力側から判定すると、gがゲート入力のときは、そのゲートの出力信号線をsとみなせるので、single_rectifiableにおける記号シミュレーションは、そのゲートの評価だけを行えばよい。gがファンアウトの茎の場合は、外部出力までの記号シミュレーションが必要である。

3.3 仕様決定関数の計算

前節で述べたように、信号線gがファンアウトの茎の場合は、外部出力までの記号シミュレーションが必要である。このとき、SF[g]を変更することにより正しい回路を得るためには、少なくとも全ての不良外部出力にeの影響が及ぶ必要がある。したがって、eの影響が及ばなかった不良外部出力が一つでもあった場合は、即座にgは単一修正不能と判定することができる。

全不良外部出力にeの影響がおよぶ場合は、(式1)を計算して仕様決定関数SG[g]を求める。このとき、eの影響が及ばなかった外部出力については、(式1)の計算に加える必要はなく、(式1)の計算量を削減できる。

eの影響が及んだ外部出力は、記号シミュレーションにイベント法を用いれば、イベントの及んだ外部出力として容易に取り出すことができる。

3.4 論理関数

前節までに述べた方法は、あらゆる処理を論理関数処理を用いて行うため、論理関数の表現法がその効率を大きく左右する。ここでは、SBDD[7]を用いる。これは、多くの論理関数を比較的少ない記憶量で表現できる、論理関数間の論理演算が高速に行えるなどの特徴に加え、以下の点で有利であることが理由である。

(1) 記号シミュレーションに関して
- 記号シミュレーションは、各信号線の信号線関数を外部入力側から順次求めて行く処理である。SBDDを用いると、信号線関数が真に依存する入力変数のみしか演算の対象に現われないため、高速に行うことができる。
- イベント法の記号シミュレーションでは、イベントの消滅を検出し、それ以後のイベントの伝搬を抑止することにより、処理量の削減を図る。イベントの消滅とは、ある信号線において、論理変数eを割り当てたときの論理関数と元の論理関数が一致することである。論理関数の一致判定は、SBDDを用いればポインタの一致判定で行えるため、イベントの消滅を高速に検出できる。

(2) 単一修正可能性判定に関して
- 論理変数eをSBDDの最上位の順位とすることによ

り、e以下の部分グラフを既に保持している信号線関数を表わすグラフと共有できる可能性が高まるため、単一修正可能性判定に必要なSBDDのノード数はそれほど多くならないと考えられる。

・(式2)において、仕様決定関数SGに対して、SG中の変数eに0(1)を割り当てたときの論理関数SG(e=0)(SG(e=1))を求める必要がある。論理変数eをSBDDの最上位の順位としておけば、SG(e=0)、SG(e=1)は、それぞれSGを表わす部分グラフのeに対応するノードの0/1枝が指す部分グラフであるから、瞬時に求めることができる。

・単一修正可能性は、(式2)の計算の結果が恒偽関数か否かによって判定される。SBDDでは、ポインタの値を見るだけで恒偽関数か否かがわかるため、高速に判定できる。

3.5 不完全指定論理関数

SBDDはそのままでは不完全指定の論理関数は表現できない。上記の処理では、関数仕様が不完全指定論理関数であるため、これを扱う必要がある。ここでは、[6]と同様、一つの不完全指定論理関数fを二つの完全指定論理関数f0, f1の組で表現する。不完全指定論理関数は、全入力変数に値を割り当てたとき、出力として0/1/dの3値をとるものとみなすことができる(dはdon't care)。この3値を2ビットで符号化する。論理値と符号の対応を表3.1に示す。

表3.1 論理値の符号化

論理値	符号
0	10
1	01
d	00

符号の1ビット目が表わす完全指定の論理関数をf0、2ビット目が表わす完全指定の論理関数をf1とする。すなわち、f0(f1)は、fが0(1)を出力する入力値の組合せにおいて、かつそのときのみ1を出力する完全指定の論理関数である。

この符号化により表現された不完全指定の論理関数(f0, f1)に、完全指定の論理関数hが含まれるか否かは、

f0→~h　かつ　f1→h

が恒真か否かにより判定できる。すなわち、仕様決定関数SGの計算のための(式1)は、

$SG = (\Sigma_{po}(FS0[po] \cdot SF[po] + FS1[po] \cdot \tilde{}SF[po]))$　(式3)

と計算される。また、こうして計算されたSG[g]からFS[g]を生成するためには(2.3節参照)、

FS0[g] = SG[g](e=1)、FS1[g] = SG[g](e=0)　(式4)

として、容易に計算できる。

4. 計算例

本章では、前章までで述べた不良修正手法の計算例を示す。本章では、理解のしやすさを優先して、論理関数をカルノー図で示す。なお、説明の簡単のため、不良修正モデルとしてはゲート種の変更のみを考える。

外部仕様を図4.1に示す。これは、1ビットの全加算器にいくつかdon't careを含めた論理関数である。この外部仕様に対して、図4.2に示す回路を設計したものとする。図4.2の回路の外部出力の信号線関数を図4.3に示した。いずれの外部出力*out*、*cout*も外部仕様を満たしていない。そこで、図4.2の回路の不良を修正する。

図4.2の回路では、網羅コーン回路はg10、g20、c0から外部入力側の部分回路である。この網羅コーン回路中の信号線を、外部出力側から一つずつ取り上げながら各信号線の単一修正可能性判定single_rectifiableを行う。

(1) g10の単一修正可能性判定

信号線g10に対してsingle_rectifiableを適用する。まず、g10に論理変数eを割り当て、記号シミュレーションを行い、*out*、*cout*の信号線関数を求め、図4.1の外部仕様を用いて(式3)を計算すると図4.4に示すg10の仕様決定関数SG[g10]が求まる。SG[g10]に対して(式2)を計算すると、恒偽関数となり(図4.5)、g10は単一修正可能であることが分かる。

そこで、(式4)により、g10の関数仕様を求めた結果を図4.6(a)に示す。図4.6(a)は二つの完全指定の論理関数FS0[g10]とFS1[g10]の組で表わされた関数仕様である。これを一つの不完全指定論理関数FS[g10]で表わすと図4.6(b)となる。

関数仕様が求まったので、回路変更modifyを実行し、実際に不良を修正することを試みる。いま、不良修正モデルはゲート種の変更のみとしているため、考えられる変更はゲートG1の種類NORを他のゲートに変更することである。G1をAND、OR、XOR、NAND、XNORに変更したときの信号線関数SF[g10]をそれぞれ図4.7(1)～(5)に示した。図4.6と図4.7から(式3)のpoをg10として計算した結果が恒偽関数となるのは、G1をAND、XNORに変更した場合である。したがって、次の二つの修正方法が提示される。

・ゲートG1をANDに変更する。

・ゲートG1をXNORに変更する。

(2) g20の単一修正可能性判定

次に信号線g20に対してsingle_rectifiableを適用する。g20に論理変数eを割り当て、記号シミュレーションを行い、*out*、*cout*の信号線関数を求め、図4.1の外部仕様を用いて(式3)を計算すると図4.8に示すg20

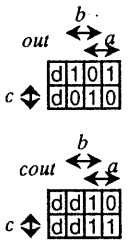


図4.1 外部仕様

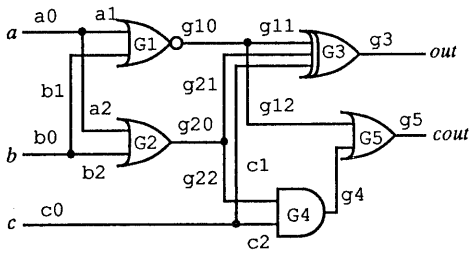


図4.2 不良回路

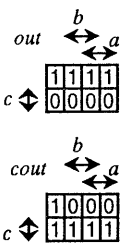


図4.3 外部出力の信号線関数

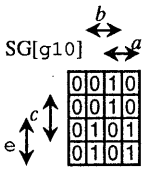


図4.4 g10の
仕様決定関数

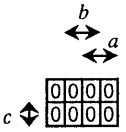
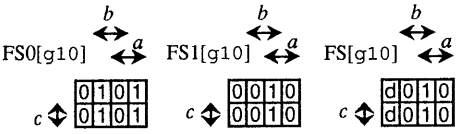
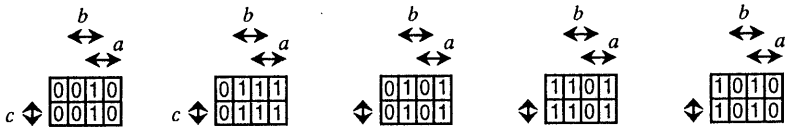


図4.5 g10の
修正可能性判定



(a) 完全指定関数表現　(b) 不完全指定関数表現

図4.6 g10の関数仕様



(1) AND　(2) OR　(3) XOR　(4) NAND　(5) XNOR

図4.7 g10の修正後の信号線関数

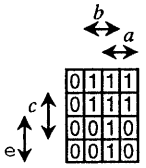


図4.8 g20の
仕様決定関数

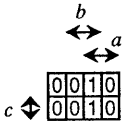


図4.9 g20の
修正可能性判定

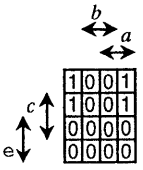


図4.10 g10の
信号線関数

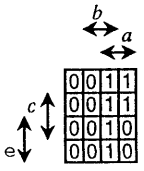


図4.11 a1の
仕様決定関数

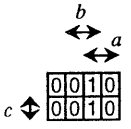


図4.12 a1の
修正可能性判定

の仕様決定関数SG[g20]が求まる。SG[g20]に対して(式2)を計算した結果を図4.9に示した。図4.9は恒偽関数ではないから、g20は単一修正不能である。

(3) c0の単一修正可能性判定

信号線c0に対する処理はg20と同様に進み、やはり単一修正不能であることが分かる。

(4) a1の単一修正可能性判定

信号線a1に対してsingle_rectifiableを適用する。a1はゲートG1のゲート入力でありG1の出力g10の関数仕様FS[g10]は既知であるため、ゲートG1の評価のみを行う。その結果のg10の信号線関数は図4.10となる。図4.10のSF[g10]と図4.6の関数仕様FS0[g10]、FS1[g10]を用いて(式3)を計算すると図4.11に示すa1の仕様決定関数SG[a1]が求まる。SG[a1]に対して(式2)を計算した結果(図4.12)は恒偽関数ではないから、a1は単一修正不能であることが分かる。

(5) その他の信号線の単一修正可能性判定

信号線b1に対する単一修正可能性判定は、(4)のa1と同様に進み、b1も単一修正不能であることが分かる。信号線a2、b2はゲート入力であり、そのゲート出力g20が単一修正不能であることから、即座に単一修正不能であることが分かる。信号線a0、b0に対する単一修正可能性判定は、(3)のc0と同様に行い、いずれも単一修正不能であることが分かる。

以上から、不良修正モデルとしてゲート種の変更のみを用いたときの図4.2の回路の修正方法は、

・ゲートG1をANDに変更する。

・ゲートG1をXNORに変更する。

の二通りであり、このいずれかの変更により図4.2の回路は図4.1の外部仕様を満足するものとなる。

5. 評価

第2章で述べたアルゴリズムを第3章で示した手法により実現したプログラムをSPARC Station 2(28.5MIPS, 主記憶48MBytes, SunOS4.1.1(JLE1.1.1))上にC言語で開発した。用いたコンパイラはSunOS標準のccであり、オプション-Oを指定してコンパイルした。SBDDの処理には、湊氏の二分決定グラフ処理パッケージSBDD Package[7]を使用した。また、BDDの入力変数の順序づけは、回路記述に現われた外部入力の順序、[8]で我々が提案した順序づけ手法のうち、仕様を表わすBDDのノード数が少ない方を用いた。

評価実験に用いた回路はISCAS'85で提案された、テスト生成用ベンチマーク回路[9]である。ベンチマーク回路のデータは、それぞれ一つの回路記述しかないため、評価ではこの回路記述を仕様とみなし、回路中のゲートをランダムに一つ選んで別のゲート種に変更した回路を不良回路とみなした。不良修正モデルもゲート種の変更のみとした。

評価結果を表5.1に示す。表5.1において、試行回数とは本プログラムの実行回数であり、その内、処理が最後まで終了した回数を終了回数に示した。終了しなかった原因は、SBDDのノード数がSBDD Packageの扱える100万個を越え、異常終了したことである。ただし、異常終了までに修正可能な信号線を指摘し、その信号線を出力とするゲートの素子種を変更することにより修正が達成できたものも多い。それらの値は表には記載されていない。

分類の欄のmin(max)の行は、処理が最後まで終了した試行のうち、全体の処理時間が最小(最大)であった試行の値である。aveの行には、処理が最後まで終了した全ての試行の値の平均値を示した。なお、C7552では、処理が最後まで終了した試行が一回しかなかったため、そのときの値を記載してある。

まずCPU時間について考察する。表5.1中、Totalは全ての処理、S_rectはsingle_rectifiable、Modifyはmodifyに要したCPU時間(秒)である。

表から、特に処理時間を要するC1355を除くと、Totalは高々数百秒であり、SBDDのノード数が溢れない限り、十分実用的な範囲である。C1355は、single_rectifiableの処理対象として選ばれる信号線数が多く(cand欄参照)、さらに信号線関数を表わすBDDも複雑である(ノード数欄参照)などが原因で、処理時間が多くなっている。

minとmaxを比較すると、2～40倍の違いがある。実験結果では、修正位置が外部出力に近いものほど処理時間が少なく、外部入力に近いものほど多くの処理時間を要した。これは単一修正可能性判定は外部出力側から行うことの当然の帰結である。外部出力に近いほど論理が複雑になるため、実際の設計では外部出力に近いほど不良が入り込みやすく、本手法が有効に働くと考えられる。

次にSBDDのノード数について考える。表5.1において、nd_spc(nd_dsn)は元の回路(不良を挿入した回路)の全信号線の信号線関数を表現するために要したノード数、nd_rctはsingle_rectifiableで必要としたSBDDのノード数である。ただし、nd_ rctはnd_dsnを保持したままの値である。

nd_dsnとdn_spcには大きな違いは見られず、不良の存在のSBDDのノード数への影響は小さい。一方、nd_rctはnd_spc、nd_dsnのノード数の最大30倍程度まで大きくなる。従って、single_rectifiableに必要となる

表5.1 評価結果

回路名	試行回数	終了回数	分類	CPU時間[秒]			ノード数			信号線数		
				Total	S rect	Modify	nd spc	nd dsn	nd rct	cand	r-abl	r-ed
C432	160	160	min	33.29	26.03	0.86	6205	6205	30926	11	1	1
			ave	117.39	108.10	2.65		7458	42296	44	2	1
			max	168.60	160.97	0.87		8646	67593	48	1	1
C499	68	68	min	156.56	145.74	2.79	32577	32577	127545	20	2	2
			ave	223.36	209.32	5.05		42633	255031	21	3	1
			max	351.22	337.90	1.94		82516	458988	18	1	1
C880	71	20	min	11.09	1.77	0.88	81208	81208	81209	1	1	1
			ave	53.25	38.00	6.85		81210	120346	20	7	5
			max	69.21	51.27	9.74		81208	81242	29	11	20
C1355	11	11	min	1352.15	1341.13	0.95	103301	103301	103896	159	1	2
			ave	2950.93	2931.60	5.98		124363	663924	160	4	2
			max	4396.93	4371.04	9.01		172067	998524	160	6	2
C1908	15	15	min	694.33	678.20	1.10	62903	49002	244347	98	1	1
			ave	1050.91	1031.90	5.27		63941	366624	122	5	1
			max	1305.39	1287.26	6.52		59460	452787	154	6	2
C2670	20	11	min	13.03	3.52	1.77	33579	33579	33582	2	2	2
			ave	227.67	215.39	4.35		34550	543052	22	4	1
			max	547.79	529.42	10.51		34652	998431	54	12	3
C7552	30	1	-	22.62	5.64	1.97	198554	198554	198555	3	2	1

ノード数を如何に減らすかが大きな問題である。

最後に、処理した信号線数について考察する。表5.1の、candはsingle_rectifiableで処理した信号線数、r-ableは単一修正可能と判定された信号線数、r-edは修正されたゲート種数である。candは全体の信号線数(回路名の数値部分)に比べると多くとも1割程度であり、3.2節で述べた信号線の選択が効果的であることが分かる。さらに、単一修正可能と判定された信号線は数個であり、本手法は位置指摘能力が非常に高いことが分かる。

6. おわりに

不良組合せ回路の単一修正方法を提案し、その評価結果を述べた。今後の課題としては、単一修正可能性判定処理の対象とする信号線数の削減、実用的な不良修正モデルの考案、複数箇所の修正が必要な不良回路の自動修正手法の開発などが挙げられる。

また、本手法を正常回路に適用すると、MSPFを用いたトランスダクション法が実現できる。本手法をベースにした、より効率的な多段論理回路の簡単化手法の研究も今後の課題である。

謝辞: SBDD Packageを提供していただいたNTTの湊氏に感謝いたします。

参考文献

[1] M. Tomita他: An Algorithm for Locating Logic Design Errors; ICCAD90, pp.468-471 (1990).

[2] J. C. Madre他: Automating the Diagnosis and the Rectification of Design Errors with PRIAM; ICCAD89, pp.30-33 (1989).

[3] M. Fujita他 : Redesign and automatic error correction of combinational circuits; IFIP Workshop on Logic and Architecture Synthesis, pp.253-262 (1990).

[4] M. Fujita他: Application of Boolean Unification to Combinational Logic Synthesis; ICCAD91, pp.510-513 (1991).

[5] S. Muroga他 : The Transduction Method－Design of Logic Networks Based on Permissible Functions; Trans. Comp., Vol. 38, No.10, pp.1404-1424 (1989).

[6] S. Minato: Fast Generation of Irredundant Sum-of-Products Forms from Binary Decision Diagrams; SASIMI92, pp.64-73 (1992).

[7] S. Minato他: Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation; DAC90, pp.52-57 (1990).

[8] 伊藤他: enumerationによる二分決定グラフ構成法; 信学技報, Vol.91, No.376, pp.17-24 (1991).

[9] F. Brglez他: A neutral netlist of 10 combinational benchmark circuis and a target translator in FORTRAN; presented at ISCAS85 (1985).