5B－4　検出能力による重み付けを用いた故障検出入力集合の準最小化法について

吉田清明　朱雀保正　元石浩二
久留米工業大学

1．はじめに

組合せ回路の故障表(fault table)より，最小故障検出入力集合[1]を求める問題は，最小被覆問題(minimum cover problem)と等価であり，ＮＰ完全であることが証明されている[2]．さらにＬＳＩ，ＶＬＳＩ等の大規模回路に対しては，多項式時内にその故障表を得ることさえ困難と考えられる．

そこで本稿では，最小化を断念する代りに，必要とされる検出率を保証した故障検出入力集合を多項式時間内に効率よく縮小する方法について述べる．本手法は与えられた故障表において，仮定した故障に対する故障検出入力数を重みとして用い，その重みの関数表を操作することにより故障検出入力数を減らしていくものである．本手法をISCAS-85の回路データから同時故障シミュレーションを用いて作成した故障検出入力集合に適用したところ，平均圧縮率は３７．２％であった．

2．最小故障検出入力集合

本手法について述べる前に，最小故障検出入力集合について簡単に説明しておく．

表１　故障表

Input No.	代表故障											
	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	
T_0	0	0	0	0	0	0	1	0	0	0	0	S_1
T_1	0	0	0	0	0	1	0	1	1	0	1	S_2
T_2	0	0	0	0	0	0	1	0	0	0	0	S_3
T_3	0	0	0	1	0	1	0	1	1	0	1	S_4
T_4	0	0	0	0	0	0	1	0	0	0	0	S_5
T_5	0	0	0	0	1	1	0	1	1	0	1	S_6
T_6	0	1	0	0	0	0	1	0	0	0	0	S_7
T_7	0	1	1	0	0	0	0	0	0	0	0	S_8
T_8	0	0	0	0	0	0	1	1	0	0	0	S_9
T_9	0	0	0	0	0	1	0	0	0	0	1	S_{10}
T_{10}	0	0	0	1	0	0	1	1	0	0	0	S_{11}
T_{11}	0	0	0	0	0	1	0	0	0	0	1	S_{12}
T_{12}	0	0	0	0	1	0	1	1	0	0	0	S_{13}
T_{13}	0	0	0	0	0	1	0	0	0	0	1	S_{14}
T_{14}	1	0	1	0	0	0	0	0	0	1	1	S_{15}
T_{15}	1	0	0	0	0	0	0	0	0	0	1	S_{16}
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	

表１のように入力パターンの印加に対して，故障の発生している回路が正常信号値を出力するか故障信号値を出力するかの対応関係を，正常信号値の場合は０，故障信号値の場合は１を用いて示した表を故障表と呼ぶ．このような故障表において，どの故障（冗長故障は除く）の列にも少なくとも１つの１が存在する入力パターンの行の組合せのうち，その数が最少となるものを最小故障検出入力集合という．　表１の場合は，

{T3, T5, T6, T14}, {T5, T6, T10, T14},
{T3, T6, T12, T14}, {T6, T10, T12, T14}

が最小故障検出入力集合となる．

最小故障検出入力集合の生成法としてはプライムインプリカント法(prime implicant method)[1]やテスト群交差法(method of test set intersection)[1]などがよく知られているが，扱う故障表が大規模化すると，その計算量は指数級数的に増大し実用的でなくなる．これに対して本手法は，故障表が大規模化しても適用可能である．

3．本手法

本手法のアルゴリズムは以下の通りである．表１のような故障表において，

[0] どの代表故障も検出しない，あるいは個々の代表故障に対する検出能力が他の入力パターンと等価，非包含関係にある入力パターンを故障表から削除する．例えば表１のT11とT13は各々の代表故障に対する検出能力が全く等しいのでどちらか一方を残せば十分であり，T15はT14に包含されるので削除可能である．

[1] もし代表故障f_jを検出可能な入力パターンが唯一ならば(表１の網の部分)，そのパターンT_iを選択し，T_iによって検出可能な全ての代表故障を故障表より削除する．そのようなT_iがなければ[2]へ行く．

[2] 故障表の各列ごとに１の総数を求めC_jとする．これはf_jを検出可能な入力パターンの総数を求めめることを意味する．

[3] 故障表の各々の１を，その列のC_jの値の２乗で割りP_{ij}とする．これはT_iの故障f_j検出への重要度

A Method to Compress Test Sets for Combinational Logic Circuits Using the Detectability Weight of Each Test. Kiyoaki YOSHIDA, Yasumasa SUJAKU and Kohji MOTOISHI. *Kurume Institute of Technology 2228 Kamitsu-machi Kurume, Fukuoka 830, JAPAN*

を表す.

[4]故障表の各行ごとのP_{ij}の総和をS_iとする.これは入力パターン全体に対するT_iの相対的な重要度を意味する.

[5][4]においてS_iの最大の値を持つT_iを選択し,T_iによって検出可能な全ての代表故障を故障表より削除する.

[6]全ての代表故障が故障表より削除されたら処理を終わる.そうでなければ[2]へ行く.

以上の処理過程で選択された入力パターンの集合が,本稿の方法による準最小故障検出入力集合である.

4. 実行結果

本手法をISCAS-85のベンチマーク回路データから作成した故障表データ(表2)に適用したところ,表3のような結果が得られた. 実験に使用した計算機は,久留米工業大学情報処理センタ内のSUN-4/670, model50 (320MB, 96.2MIPS, 17.2MFLOPS)である.故障表データは,同時故障シミュレーションにランダム入力パターンを与えることにより作成した.またその際,未検出故障は故障表より除去した.実行時間の計測にはUNIXシステムのtimeコマンドを使用した.

表2 故障表の特性

回路名	検出率 (%)	代表故障数 (個)	故障表サイズ
c432	98.3	524	84×515
c499	98.3	758	64×745
c880	97.0	942	79×914
c1355	95.4	1574	82×1501
c1908	90.6	1879	103×1702
c2670	83.6	2747	101×2296
c3540	92.5	3428	187×3172
c5315	98.2	5350	186×5255
c6288	99.6	7744	49×7709
c7552	91.6	7550	201×6912

表3の結果より,本手法は故障検出入力集合の準最小化法として,あるいは最小化の際の枝刈り法に有効であると考えられる.

表3 実行結果

回路名	圧縮前 (個)	圧縮後 (個)	圧縮率 (%)	CPU Time (sec)
c432	84	51	39.29	2.7
c499	64	48	25.00	3.2
c880	79	44	44.30	4.6
c1355	82	61	25.60	9.0
c1908	103	67	34.95	14.3
c2670	101	60	38.61	18.2
c3540	187	111	40.64	65.7
c5315	186	106	43.01	110.1
c6288	49	30	38.78	23.3
c7552	201	117	41.79	180.3

なお,表3の圧縮率は式(1)のように定義する.

$$\text{圧縮率} = \frac{\text{圧縮前} - \text{圧縮後}}{\text{圧縮前}} \times 100\ (\%) \qquad (1)$$

5. おわりに

多項式時間内に比較的効率よく故障検出入力集合を圧縮する手法を提案した.本手法の計算量は圧縮前の故障検出入力数をn,既検出故障数をmとすると,$O(n^2m)$である.本手法をISCAS-85のベンチマーク回路データに適用したところ,平均圧縮率は37.2%であった.今後は本手法により生成された故障検出入力集合が,与えられた故障表の最小故障検出入力集合に対して,どの程度効率がよいのかなどについて評価を行う予定である.

文献

[1]玉本英夫:論理回路の故障診断,日刊工業新聞社(1983).

[2]M. Garey: *Computers and Intractability, A Guide to the Theory of NP-completeness*, W.H. Freeman and Co. (1979).

[3]吉田,朱雀,元石: 重要度を用いる故障検出テスト入力集合の準最小化法,九州支部連合大会(1979).

[4]F.F. Sellers, M.Y. Hiao and C.L. Bearnson," Analyzing errors with Boolean difference, " IEEE Trans.on Comput., vol. C-17, pp.676-683, July (1968).