計算機アーキテクチャ 110－4
設 計 自 動 化 73－4
（1995．1．19）

連想メモリを用いた高並列故障シミュレーション手法

大野慎介　前田志門　松下章　大附辰夫

早稲田大学　理工学部

あらまし

連想メモリ (CAM:Content Addressable Memory) は，ワード並列の一致検索機能，隣接ワード間での1ビットデータの双方向並列通信機能，およびワード並列書き込み機能を持つ．石浦と矢島は連想メモリのこれらの機能を利用した並列故障シミュレーション手法を提案し，1テストベクトル当たりのシミュレーションが $O(n)$(n:回路の信号線数) で実現可能であることを示した．しかし，このための連想メモリの必要記憶量は $O(n^2)$ であり，これをいかに削減するかが問題となっていた．そこで本稿では，より少ない記憶量で高い並列度を実現する連想メモリ上での並列故障シミュレーションアルゴリズムを提案し，ISCAS'85 のベンチマーク回路を用いて従来手法と比較した結果を示す．

A Massively Parallel Fault Simulation Algorithm Using Content Addressable Memory

Shinsuke OHNO, Shimon MAEDA, Akira MATSUSHITA and Tatsuo OHTSUKI

School of Science and Engineering, Waseda University
3-4-1 Ohkubo, Shinjuku-ku, Tokyo 169

Abstract

CAM(Content Addressable Memory) can operate word-parallel equivalence search, bilateral 1bit data shifting between consecutive words, and word-parallel writing. Ishiura and Yajima proposed the parallel fault simulation algorithm using these CAM operations, which takes only $O(n)$ simulation time per test vector, where n is the number of nets. But to maintain this $O(n)$ simulation time, the algorithm requires $O(n^2)$ storage size, which has been the main obstacle to apply it to large scale circuits. In this paper, a new massively parellel fault simulation algorithm requiring less CAM storage size is proposed and is compared with Ishiura and Yajima's algorithm by ISCAS'85 benchmark circuits.

1 はじめに

故障シミュレーションとは、故障の存在を仮定して、その時の論理回路の動作を調べるものである。故障検査系列の故障検出率の評価、故障検査系列の生成、故障辞書の作成などに用いられ、論理回路の故障検査に不可欠なものであるが、仮定された多くの故障に対してシミュレーションを行わなければならないため、回路規模の増大に伴う計算量、必要記憶量の増大が深刻な問題となっている。逐次計算機上のアルゴリズムの改良とともに、並列処理ハードウェアを用いた処理の大規模並列化を考える必要がある。また、このような並列処理ハードウェアは性能面の高さのほかに、コスト面から考えてある程度の汎用性を備えているものが望ましい。連想メモリはその機能から、細粒度のSIMD型並列計算機構と考えることができ、他のCADの分野への応用も広く提案されている [2][3]。

石浦と矢島は連想メモリを用いた故障シミュレーション手法を提案しており [1]、1テストベクトル当たりの故障シミュレーションの計算時間を、連想メモリを用いることによって $O(n)$ (n:回路の信号線数) にできることを示している（逐次計算機上では $O(n^2)$）。しかしこのために必要となる連想メモリの記憶量は $O(n^2)$ である。このことは、一定記憶量の場合のシミュレーション時間は $O(n^2)$ となることを意味する。大規模回路に対しては $O(n^2)$ の記憶量を確保するのは非常に困難となるので、一定記憶量でいかに高い並列度を維持するか、あるいは1テストベクトル当たりの $O(n)$ 実行を可能にする記憶量をいかに削減するかが問題となっていた。

これに対して本稿では、必要記憶量を削減し高並列化を実現した連想メモリ上の故障シミュレーション手法を提案する。以下、2章で [1] の手法について簡単に説明しその問題点を示す。3章では提案手法について述べる。まず最初に本方法の中心的な考えとなる連想メモリ内ビットの再利用について説明し、続いてこのためのゲートの評価順序の決定法について述べ、この考えを用いた故障シミュレーションのアルゴリズムについて解説する。4章では提案手法と [1] の手法との比較を行う。

本手法が対象とする回路の種類、レベルおよび故障のモデルは、[1] の手法と同じく、

1. ゲートレベルの組合せ回路または同期式順序回路
2. 信号は2値論理、遅延は0遅延モデル
3. 故障モデルは単一縮退故障モデル

とする。

2 文献 [1] の手法

文献 [1] では、連想メモリの持つワード並列の一致検索機能、隣接ワード間の1ビットデータの双方向並列通信機能、およびワード並列書き込み機能を用いることにより、1テストベクトル当たりの故障シミュレーションを $O(n)$ の手間で行えることが示されている。以下では連想メモリのこれら3つの機能について説明した後、文献 [1] の手法について簡単に述べることにする。

2.1 連想メモリの基本機能 [2]

図1に連想メモリの基本構成を示す。連想メモリは

- データを格納するメモリアレイ
- 一致検索、並列書き込みのためのキーを格納するインデックスレジスタ、キーをマスクするためのマスクレジスタ
- 検索結果、書き込み対象ワードを示すフラグとしてのレスポンスレジスタ

から構成される。通常のRAMの機能に加え、以下に示すような機能を併せ持つ。

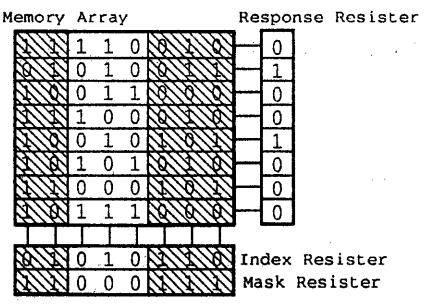

図 1: 連想メモリの基本構成

2.1.1 ワード並列の一致検索機能

メモリアレイの各ワードに格納されているデータのうち、インデックスレジスタに与えたキー（データ）と一致するデータを検索し、一致したデータを持つワードのレスポンスレジスタにフラグ1を立てる機能である。このとき、マスクレジスタに1が立っているビットは一致判定の対象から外す。この操作は、メモリセルアレイのワード数に依らない定数時間で行うことができる（ワード並列）。

2.1.2 隣接ワード間の1ビットデータの双方向並列通信機能

図2に示す通り、各ワードのレスポンスレジスタの内容を上位／下位隣接ワードのレスポンスレジスタへシフトする機能である．このシフトは，全ワードのレスポンスレジスタに対して一斉に行われる．

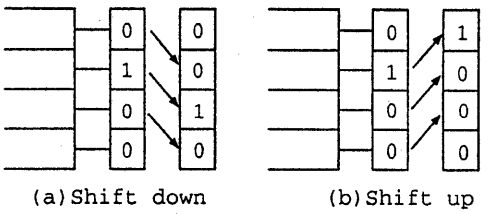

図 2: レスポンスレジスタのシフト

2.1.3 並列書き込み機能

書き込みたいデータをインデックスレジスタに与え，書き込みを行いたくないビットに対してはマスクレジスタに1を立てておけば，並列書き込み機能により，インデックスレジスタのデータのうちマスクを施されていない部分をワード並列に書き込むことが可能である．書き込み対象のワードは，(1) 全ワード (2) レスポンスレジスタが1のワード (3) レスポンスレジスタが0のワードのいずれかを指定することができる．

2.2 文献 [1] の手法

2.2.1 連想メモリ内のデータ格納法

文献 [1] では，図3に示すように，各ワードはタグフィールド，信号値格納用フィールド，ワード間通信用ビットに分けられる．

正常回路，あるいは1つの故障回路の信号線を格納しているワードをまとめてブロックと呼ぶことにする．タグフィールドには，各ブロックの先頭（最上位）ワードからのオフセットアドレス値が格納される．

信号値格納用フィールドは，回路の信号値を格納するのに用いられる．各信号線の信号値は，回路のレベルソート順に従って先頭ワードから順に下位ワードに向かって格納位置が割り当てられる．このとき，後述する連想メモリ内での論理演算を行うために，回路の各ゲートの入出力線は全て同一ワードに割り当てられる必要がある．現在割り当て

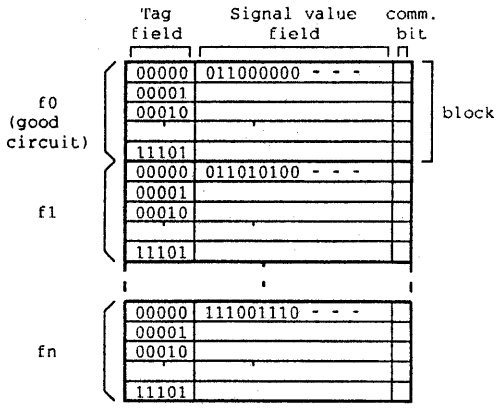

図 3: 文献 [1] のデータ格納法

を行っているワードにそれだけの空きビットがない場合には，ゲートを分解するか，次のワードに割り当てられる．このようにして正常回路と各故障回路の信号値の格納位置を決定する．各回路の同一信号線の割り当て位置は，各ブロック内での相対位置が同じとなる．

2.2.2 論理演算の方法

例としてANDゲートの評価法を考える．このゲートの入力線および出力線の格納位置は，各ブロック内でオフセットが p であるワードのそれぞれ第 $\{i_1, i_2, \ldots, i_n\}$ ビット，第 o ビットであるとする．このときこのゲートの評価は連想メモリの機能を用いて以下のように行われる．

1. インデックスレジスタのタグフィールドに対応するビットに2進数で p を書き込み，マスクレジスタ内の残りのフィールドに1を書き込む．
2. 一致検索を行う．各ブロック内のオフセットが p であるワードのレスポンスレジスタに1が立てられる．
3. インデックスレジスタの第 o ビットを0とする．マスクレジスタの残りのビットに1を立てる．
4. 並列書き込みにより，レスポンスレジスタが1のワードに対して書き込みを行う．これにより，ゲートの出力線の値は0に初期化される．
5. インデックスレジスタの第 $\{i_i, i_2, \ldots, i_n\}$ ビットを1とする．マスクレジスタの残りのビットに1を立てる．
6. 一致検索を行う．入力線の値が全て1であるゲートを格納するワードが選択される．

7. インデックスレジスタの第 o ビットを 1 とする．マスクレジスタの残りのビットに 1 を立てる．

8. 並列書き込みにより，レスポンスレジスタが 1 のワードに対して書き込みを行う．入力線の値が全て 1 であるゲートの出力線の格納位置に 1 が書き込まれる．

他の種類のゲートに対しても同様に評価することができる．この方法により，正常回路および各故障回路内の同一ゲートの評価は並列に行われることになる．

2.2.3 ファンアウト通信の方法

同一ワードにないゲート間では，信号値をワード間通信用ビットを用いて転送する．先に述べた連想メモリの 3 つの機能を組み合わせることにより，ワード間通信用ビットの値を上位／下位隣接ワードに並列にシフトすることができ，文献 [1] ではこの機能を用いたファンアウトの「パイプライン伝搬」を行うことによって，ファンアウト通信による計算量の増大を防いでいる．

パイプライン伝搬法は，先のデータ格納法によればファンアウト先は必ずファンアウト元の格納ワード以下のアドレスのワードになることに着目して，以下のように行われる．

1. 評価したゲートのファンアウト先が他ワードの場合，そのゲートの出力値を通信用ビットに書き込んだ後，通信用ビットの値を隣接下位ワードにシフトする．

2. ゲート評価を行うワードが移った際にも，通信用ビットの値を隣接下位ワードにシフトする．これにより，移り先ワードの通信用ビットの上書きを防ぐ．

3. 通信用ビットの値をシフトした結果，所望の値が転送先まで到達したワードは，通信用ビットの値を転送先に取り込む．この取り込み操作も連想メモリの機能を組み合わせて実現することができる．

2.2.4 レジスタからのフィードバック

レジスタからのフィードバックは各テストベクトルのシミュレーションの後に行われ，これも回路内の全レジスタ値に対するパイプライン伝搬を行うことにより，計算量の増大を防いでいる．詳細は文献 [1] を参照されたい．

2.3 利点と問題点

汎用計算機による並列故障シミュレーションと本手法の違いは，前者の並列度が計算機内のレジスタ長に固定されるのに対して，本手法は連想メモリの増設により比較的容易に並列度を上げることができるところにある．これにより，連想メモリの記憶量が十分にあれば，全故障回路のシミュレーションを並列に行えることになり，この場合のシミュレーション時間（1 テストベクトル当たり）は 1 個の故障回路のシミュレーション時間に等しいことになる．先に述べたように，この時間は回路の信号線数を n としたとき $O(n)$ である．

しかし，このために必要な記憶量は，

1. 1 故障回路のシミュレーションに必要な記憶量は，回路の全信号線数分だけのビット数（$O(n)$）である．

2. シミュレーションを行う故障数は一般に $O(n)$ である．

ことから，全体として $O(n^2)$ となってしまう．回路の大規模化に伴ってこれだけの記憶量を確保することは現実的には不可能である．また，記憶量を一定と考えた場合には，回路規模の増大に伴って並列度が $O(n)$ で減少することになるので，シミュレーション時間は $O(n^2)$ となってしまう．したがって，必要記憶量を削減し，一定記憶量での並列度を高く維持することが問題となっていた．

3 提案手法

3.1 基本アイデア

本節では，連想メモリの必要記憶量の削減法について新たに提案する．例として図 4に示す回路を考える．文献 [1] の手法では，1 故障回路当たりに必要な連想メモリのビット数は回路の信号線数に等しく，17 ビットである．しかし，例えば信号線 0 の値はゲート a の評価時のみ参照され，信号線 1 の値はゲート c の評価時のみ参照されるといった具合に，ほとんどの信号線の値はシミュレーションの途中から参照されなくなる．これら参照されなくなった値を格納していたビットに，新たに評価したゲートの出力線の値を格納すれば，連想メモリの記憶量はかなり節約されることになる．図 4の回路をレベルソート順に評価した場合に，シミュレーションの進行に応じて各ビットに割り当てられる信号線が変化する様子を図 5に示す。図より，各ビットは複数の信号線の値を格納するのに利用され，必要なビット数は 17 ビットから 5 ビットに削減されているのがわかる．

この時に必要となるビット数はゲートの評価順序によって変化する．また，ゲート評価のためには，そのゲートの入力線は，

1. 外部入力線

2. 記憶素子からのフィードバック線（疑似外部入力線）

3. それまでに評価したゲートの出力線

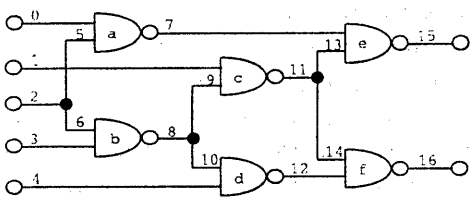

図 4: 回路例

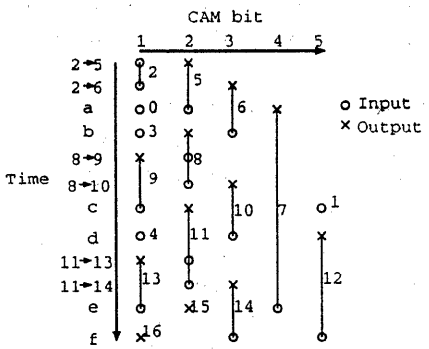

図 5: ビットの再利用

4. それまでに評価したゲートの出力線の分岐線

のいづれかでなければならない. これをゲート評価の半順序関係と呼ぶ.

以上より,

> ゲート評価の半順序関係のもとで, 必要なビット数を最小にするようなゲートの評価順序を決定する

という問題を解く必要があることがわかる. しかしこの問題は NP 困難であることが予想されるので, 以下ではこの問題を解くためのヒューリスティックを示す.

3.2 ゲート評価順序の決定

本方法では, ゲートの評価順序を決定する前に, 回路内の FFR(Fanout Free Region: 分岐のない信号線で結ばれたゲートの集合, 図 6) の評価順序を決定し, その後に各 FFR 内でのゲートの評価順序を決定する. FFR 内の信号線の値はその FFR 内のゲートの評価時にのみ参照されるので, これらのゲートをまとめて評価すれば, FFR 内の信号線の値を格納するビットを早く再利用 (他の信号線の値の格納に用いること) できるからである.

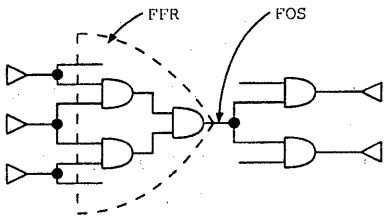

図 6: FFR と FOS

FFR の評価順序の決定は, 「次に評価する FFR」を, 評価可能となっている (FFR の入力線の値が全て評価済みである) FFR の中から以下の基準で選択することによって行われる.

選択基準 評価可能となっている各 FFR についてその入力線を調べ, 仮にその FFR を選択した場合に, それ以降のシミュレーションで参照されなくなる入力線の本数を数える. この本数が最大の FFR を選択する.

これは, このような FFR を選択すれば, 以降で再利用できるビットが最も多くなるからである. また, FFR の評価順序が決まった後, 各 FFR 内でのゲートの評価順序の決定も, 上の選択基準を評価可能ゲートの集合に対して適用することにより行われる.

例として図 4の回路について考える. FFR1 = {a, e}, FFR2 = {b}, FFR3 = {c}, FFR4 = {d, f} とする。

1. 最初に評価可能である FFR は FFR2 のみであるので, FFR2 を選択する.
2. FFR2 を評価した結果, 新たに FFR3 が評価可能となるので FFR3 を選択する.
3. FFR3 を評価した結果, FFR1 と FFR4 が評価可能となる. FFR1 を選択した場合, 以降で参照されなくなる信号線は 0, 5, 13 の 3 本, FFR4 を選択した場合も 4, 10, 14 の 3 本である. この場合はランダムに FFR1 を選択する.
4. 以上から, FFR の評価順序は FFR2, FFR3, FFR1, FFR4 の順になる. FFR1, FFR4 内のゲートは半順序関係からそれぞれ a, e と d, f の順となる. したがって最終的に決まるゲートの評価順序は b, c, a, e, d, f の順となる.

定理 1 1 個の FFR から成る回路の場合, 外部入力線数を NI としたとき, 1 故障回路に必要な記憶量は高々 $2NI$ ビットである.

証明 1段目のゲート評価を行うのに必要なビット数があれば十分であるからである.

系 1 回路内の FFR の入力線数のうち最大のものを NI とし, 回路内の FOS(Fanout Stem:分岐線の幹) の数を NFO としたとき, 1 故障回路に必要な記憶量は高々 $2NI + NFO$ ビットである.

実装法の詳細については省略するが, この方法は $O(n)$ の計算量で実行することが可能である.

3.3 アルゴリズム

3.3.1 提案手法のデータ格納法

連想メモリ内の信号線の割り当ての様子を図 7に示す. 1 故障回路に用いられるブロックは, バッファ領域 (Buffer field), ゲート評価領域 (Gate eval. field), キャッシュ領域 (Cache field), 内部信号線格納領域 (Internal net value field), 外部入出力線格納領域 (PI/PO net value field) に分けられる. 外部入力線をすべて外部入出力線格納領域に割り当てた後, 上述の方法で決められたゲートの評価順に従いゲートの入出力線を割り当ててゆく.

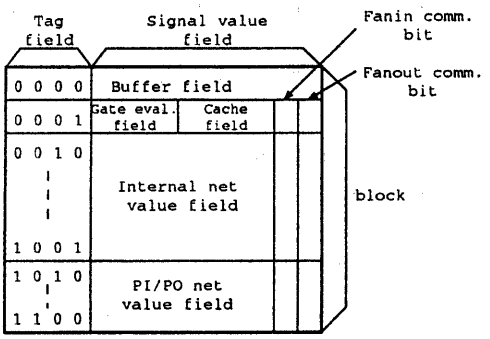

図 7: 提案手法のデータ格納法

信号線の割り当て領域は以下の基準で決められる.

1. 内部信号線の場合, その信号線が最初に割り当てられてから参照されなくなるまでの期間が一定値より長い場合, 内部信号線格納領域に割り当てる. 一定値より短い場合, キャッシュ領域に空きビットがあればキャッシュ領域に割り当て, 無ければ内部信号線格納領域に割り当てる.
2. 外部出力線, および疑似外部出力線の場合, 外部入出力線格納領域に割り当てる.

各領域に割り当てられている信号線のうち, 参照されなくなった信号線の値を格納するビットは, 空きビットとして他の信号線を割り当てに用いられる.

ゲートの論理演算はすべてゲート評価領域で行われる. これは, ビットの再利用を行った場合には同一ゲートの入出力線がすべて同一ワードに割り当てられる保証がないからである. ゲート評価の際には, 入力線の値がその割り当て位置からゲート評価領域まで転送された後にゲート評価が行われ, 出力はその信号線の所定の割り当て位置まで転送される. このため, 各ワードは入力線転送ビット (Fanin comm. bit), 出力線転送用ビット (Fanout comm. bit) の 2 ビットを持つ.

今回はゲートの入力線の最大本数を 9 としたので, ゲート評価領域の 10 ビット (入力線格納用 9 ビット + 出力線格納用 1 ビット) を除いた同一ワードの残りの部分はキャッシュ領域として用いられる. キャッシュ領域に割り当てられる信号線の値はゲート評価の際に転送を行う必要がないので, 転送のオーバヘッドを削減するにはキャッシュ領域に割り当てられる信号線を多くすればよい. 先の基準は, キャッシュ内ビットが再利用される頻度を高くして, 多くの信号線を割り当てられるようにするための条件である. 基準内にある「一定値」としては, キャッシュ領域のビット数の 10 倍程度が経験的に適当であった.

また, 外部入出力線の格納領域を内部信号線から切り離すことにより, 外部入力線の値 (テストベクトル) はこの領域のワードのみへの書き込みで行うことができ, また外部出力線の値もこの領域のワードのみから読み出すことができる. 疑似外部出力線の値は, 次のテストベクトルにおいて対応する疑似外部入力線の値として用いられる.

バッファ領域には外部入出力線格納領域と内部信号線格納領域に必要なワード数に等しいビット数が確保される. この領域は, 後述するゲート入出力線のパイプライン伝搬を行う際の RAW (Read After Write) ハザードを防ぐために用いられる.

系 2 本アルゴリズムで 1 故障回路当たりに必要な記憶量は $O(NI + NFO)$ である.

3.3.2 ゲート入出力線のパイプライン伝搬

キャッシュ領域の利用により, ゲート入出力線のゲート評価用領域への転送 (シフト) 回数はある程度抑えることができるが, ここではこの回数をさらに抑えるためのパイプライン伝搬法について述べる.

入力線のパイプライン伝搬

ゲート評価順に、ゲートの入力線のうち、割り当てられているワードがキャッシュ領域以外であるようなものをリストに付け加えてゆく。このリストの順に信号線がゲート評価領域まで転送されれば、1ゲート当たりの入力線転送のためのシフトアップの回数は高々そのゲートの入力線数に等しい回数で済む。このためには、リストのi番目の信号線が割り当てられているワードの、ゲート評価領域のワードからのオフセットをL、信号線格納領域のワード数をWとしたとき、$i+W-L$回シフトアップを行った時点でその信号線の値を入力線転送用ビットにコピーすればよい。ただし最初のゲートに関してだけは、入力線がゲート評価用ワードに到達するまでに最高で信号線格納領域のワード数分だけのシフトアップを行う必要がある。

出力線のパイプライン伝搬

これは文献 [1] のパイプライン伝搬法とほぼ同じである。ゲート評価領域で評価されたゲートの出力線の値は、その格納位置がキャッシュ領域でない場合、出力線通信用ビットにコピーされ、出力線通信用ビットは1回シフトダウンされる。また、後述する理由により、隣接するワードにあるバッファ領域にも直接転送される。シフトダウンの結果、格納位置のあるワードに到達した信号線の値は格納位置にコピーされる。具体的には、j番目に評価されるゲートの出力線の値は、$j+L$回のシフトダウンが行われた時点で格納位置のあるワードに到達する。

パイプライン伝搬の RAW(Read After Write) ハザード

入出力線の値はパイプライン伝搬されるため、あるゲートの出力線の値が格納位置まで到達しないうちに、格納位置の値が後続のゲートの入力線の値として入力線通信用ビットにコピーされてしまう事態が生じる。これを RAW ハザードと呼ぶ。これを避けるためには、出力線の値が出力線通信用ビットを伝搬している間は、その値を同時にバッファ領域に保持しておき、そのような信号線を入力線とするゲートの評価の際には、入力線通信用ビット内の値の代わりにバッファ領域内の値を用いるようにすれば良い。バッファ領域からゲート評価領域までの信号値の転送にはパイプライン法を用いることはできないので、直接転送される。

4 実験結果

ISCAS'85 のベンチマーク回路ファイルを読み込んで連想メモリ上での本アルゴリズムの動作を記述するファイルを出力するプログラムを実装し、1故障回路当たりに必要なワード数、ワード間シフトの回数、およびこの動作記述ファイルを出力するまでの時間を測定した。ただし、連想メモリのワード長は1ワード 36 ビットを仮定した。

実装は C 言語を用いて行い、使用計算機は Sun Sparc Station 2. コンパイラは gcc である。各項目に関して、文献 [1] の手法と比較した結果を表1に示す。

表1より、ワード数は提案手法のほうが大幅に少なくなることがわかる。ワード数が少ないことは、一定記憶量で扱える故障回路数が多いことを意味する。大規模回路で1テストベクトル当たりの全故障のシミュレーションを複数のパスに分けて行う必要がある場合には、パスの回数が少なくなり、シミュレーション時間が高速化されることになる。ただし、文献 [1] の手法の実装では、信号線の割り当てを行っているワードの空きビットがこれから割り当てようとするゲートの入出力線の本数より少ない場合、ゲート分解を行わずに次のワードに割り当てを行うことにしている。また、同一レベルのゲートの評価順序を工夫することにより、ゲート分解を行わない場合でも必要ワード数を表1の値より少なくできる場合もある。しかしいづれにしても1故障回路当た信号線格納領域は最低 n ビット必要であり、この値は例えば C7552 では 291 ワードに相当するので、提案手法が大幅に記憶量を削減していることに変わりはない。

また、ゲートの評価回数は両手法とも同じであるので、シフト数、コピー数が少ないことはシミュレーション1パスの実行時間が削減されることを意味する。提案手法ではゲート入出力線のパイプライン伝搬法とキャッシュ領域の利用によって、これらの値が文献 [1] の手法と同程度かそれ以下になっていることがわかる。

さらに、3.2 節で述べたゲート評価順序の有効性を示すために、提案手法において1故障回路に必要なワード数を、ゲートの評価順序をレベルソート順に行う場合と 3.2 節で示した方法とで比較した結果を表2に示す。表より、3.2 節の方法はレベルソート順に比べて必要なワード数を大幅に少なくできることがわかる。

5 おわりに

本稿では連想メモリを用いた新しい並列故障シミュレーションのアルゴリズムを提案した。この方法により、文献 [1] では1故障回路当たり $O(n)$ (n:回路の信号線数) だけ必要となる連想メモリの記憶量が、$O(NI+NFO)$ (NI:回路内の FFR の入力線数の最大値、NFO:回路内の分岐信号線の幹の本数) に削減される。また、キャッシュ領域の利用、ゲート入出力転送のパイプライン化によって、シミュレーション1パスの実行時間も提案手法と同程度あるいはそれ以下に

表 1: 文献 [1] の手法と提案手法の比較（1 ワード 36 ビット）

回路	文献 [1] の手法				提案手法			
	ワード数†1	シフト数†2	コピー数†3	処理時間 [s]†4	ワード数	シフト数	コピー数	処理時間 [s]
C17	1	0	0	0.51	3	12	18	1.48
C432	32	373	355	0.70	5	457	718	1.69
C499	38	438	409	0.66	5	208	363	1.77
C880	70	838	816	0.97	6	574	960	2.06
C1355	110	1377	1349	1.05	5	563	947	2.51
C1908	157	2012	1973	1.27	6	1227	1970	3.05
C2670	207	2558	2615	1.50	14	2454	3815	4.04
C3540	311	3755	3629	1.89	8	4012	5081	4.77
C5315	463	5457	5465	2.69	14	2842	4975	7.41
C6288	575	6695	6686	3.25	8	1663	2830	11.96
C7552	667	7904	7986	3.56	21	4318	7560	9.65

†1 1 故障回路当たりに必要となるワード数 †2 シミュレーション 1 パスで行われるワード間シフトの回数 †3 シミュレーション 1 パス内で行われる通信用ビットと信号線格納領域との信号線の受渡しの回数 †4 シミュレーションの動作記述ファイルを出力するまでの処理時間

表 2: ゲート評価順序の違いによる 1 故障回路に必要なワード数の比較

回路	レベルソート順にゲート評価を行った場合	提案手法（3.2 節の方法）でゲート評価を行った場合
C17	3	3
C432	7	5
C499	8	5
C880	11	6
C1355	11	5
C1908	13	6
C2670	22	14
C3540	21	8
C5315	42	14
C6288	23	8
C7552	44	21

削減されることが実験結果より明らかになった.

連想メモリを用いた並列故障シミュレーションは, 連想メモリが比較的容易に増設可能であることから, 汎用計算機に比べて高い並列度を達成することができる. しかしその反面, 連想メモリ-ホスト計算機のデータ転送というボトルネックを伴う. これを本質的に解消するには転送データ量を削減する必要があり, 故障シミュレーションの場合には等価故障解析による挿入故障数の削減, テストベクトルコンパクションによるシミュレーションパス数の削減などが考えられる. 今後はこれらの問題について検討を行う.

謝辞

本研究に進めるに当たり, 貴重な御指導, 御助言を賜わりました本学佐藤 政生助教授に深謝致します.

参考文献

[1] 石浦 菜岐佐, 矢島 脩三: "連想記憶を用いた線形時間故障シミュレーション," 第 4 回回路とシステムに関する軽井沢ワークショップ, pp.63–68 (Apr. 1991).

[2] T. Ohtsuki: "A New Content Addressable Memory Chip for VLSI Design," *SASIMI '93*, pp.209–216 (Oct. 1993).

[3] 佐藤 政生, 久保田 和人: "特集 機能メモリのアーキテクチャとその並列計算への応用 6. LSI CAD 分野への応用," 情報処理, pp.1276–1286 (Dec. 1991).