社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－SLDM－127 (11)
2006／11／28

メモリコアに対する組込み自己修復を考慮したSoCのテストスケジューリング

福田 雄介† 米田 友和† 藤原 秀雄†

† 奈良先端科学技術大学院大学，情報科学研究科
〒630-0192 けいはんな学研都市
E-mail: †{yuusu-fu,yoneda,fujiwara}@is.naist.jp

あらまし 本研究では，システムオンチップに搭載されている組込み自己修復機能を持つメモリコアに対する消費電力制約下でのテストスケジューリング法を提案する．提案手法では，組込み自己修復機能を持つメモリコアは，テスト，診断・修復，再テストの3つのステージによりテストされるとし，各ステージはその順序関係を満たしていれば連続して行われる必要はないとすることで，より効率的なテストスケジューリングを可能とする．また，故障が最初に検出された時点でシステムオンチップのテストを終了するというabort-on-failの概念を利用することで，テスト時間の期待値の削減が可能である．

キーワード システムオンチップ，組込み自己修復，消費電力，テストスケジューリング

Test Scheduling for SoCs with Built-In Self-Repairable Memory Cores

Yusuke FUKUDA†, Tomokazu YONEDA†, and Hideo FUJIWARA†

† Graduate School of Information Science, Nara Institute of Science and Technology
Kansai Science City, 630-0192, Japan
E-mail: †{yuusu-fu,yoneda,fujiwara}@is.naist.jp

Abstract This paper presents a power-constrained test scheduling mehtod for SoCs with built-in self repairable memories which are tested the following three stages:1)test, 2)diagnosis/repair and 3)re-test. The proposed method can acheive efficient and effective scheduling by considering that it is not necessary to test the three stages of each core consecutively as long as they are tested in correct order. Moreover, by using the concept of abort-on-fail which means that the test is aborted as soon as a fault is detected, we can reduce expected test time.

Key words System-on-Chip, Built-In Self-Repair, Power Constraints, Test Scheduling

1. まえがき

近年の半導体技術の向上に伴い，従来では複数のLSIにより構成していたシステムを，各LSIをコアと呼ばれる機能ブロックとして組み合わせることで一つのLSI上でシステムを実現するシステムオンチップ (System-on-Chip，以下SoC) が増えてきている．これにより，ボードに搭載するチップ数の低減による実装面積の縮小化，実装コストの低減，高速化といった効果を持つ．また設計済みのコアを再利用することで短期間で，大規模な回路の設計が可能になる．しかし，SoCのテストは従来のボード上に複数のLSIで構成していたシステムに比べ困難であり膨大な時間とコストが必要となるため，SoCに対するテスト容易化設計が必要不可欠である．

SoCにおいては，内部に埋め込まれたコアはSoC外部からは直接制御，観測することは出来ない．また，テスト対象コアは他のコアと論理的に切り離してテストする必要がある．これよりSoCのテストには，(1) テストパターン発生器およびテスト応答解析器，(2) テストアクセス機構 (Test Access Mechanism，以下TAM)，(3) ラッパーが必要となる [1]．代表的なテストパターン発生器およびテスト応答解析器として自動テスト装置 (Automatic Test Equipment，以下ATE) が用いられている．TAMとは，ATEとコアを接続するものであり，ラッパーを通じてコアとATE間でのテストデータの伝搬を行う．ラッパーとは他のコアからテスト対象のコアを論理的に独立させ，通常動作とTAMを利用したテスト動作とを切り替える機能を提供するものであり，IEEE1500 [2] として標準化されている．またSoCのテスト時間を短縮するために数多くのテストスケジューリング手法が提案されている [3]~ [6]．

近年の SoC では，チップ面積に対するメモリコアの占める割合が増加している．国際半導体技術ロードマップ[10]によると，2011 年にはチップ面積の 90 %はメモリコアが占めると示されている．メモリコアに対するテスト手法としては，組込み自己テスト (Built-In Self Test，以下 BIST) が一般的に用いられる[11]．しかし，一般的にメモリコアの容量が増えるに従い，歩留まりが低下することが知られており，近年では，歩留まりの向上を目的として組込み自己修復 (Built-In Self Repair，以下 BISR) 機能を持つメモリコアが用いられている[12]~ [15]．BISR 機能を持つメモリコアは，冗長セル，BIST 回路および故障修復のための解析器を持つ．まず初めに，BIST 回路によるテストを行う．その結果を修復解析器で診断・解析し，搭載されている冗長セルで修復可能かどうかを判定する．修復可能ならば冗長セルを用いてメモリコアの再構成を行い，再度 BIST 回路によるテストを行う．[16] では，製造プロセスが 0.18um である DRAM に関して，1 行 1 列の冗長セルを搭載した場合に比べて 2 行 2 列の冗長セルを搭載した場合では，歩留まりが 3 割程度改善されることが示されている．これまでにメモリコアを含む SoC に対して数多くのテストスケジューリング手法が提案されているが，この BISR 機能を持つメモリコアを対象とした手法はこれまでに提案されていない．

本論文では組込み自己修復機能を持つメモリコアを搭載した SoC に対する消費電力制約下のテストスケジューリング法を提案する．本研究では各メモリコアはテスト，診断および修復，および再テストの 3 ステージで構成され，各ステージの実行時間および実行時の消費電力が与えられるとする．このときステージはその順序関係を満たしていれば連続して行われる必要はないとすることで，効率的なスケジューリングを可能とする．また故障を最初に検出した時刻で SoC のテストを終了する Abort-on-Fail の概念を利用する．各ステージに対して，そのステージで故障が検出されない確率または修復可能である確率を利用することで，テスト時間の期待値の削減が可能である．実験結果では，3 つのステージが連続して行わなければならない場合と比較して，テスト時間の期待値が削減可能であることを示す．

以下，2 章ではメモリコアの組込み自己修復，3 章でテスト時間の期待値について説明する．4 章で問題の定式化を行い，5 章でテストスケジューリング法の説明をする．6 章で実験結果を示し，7 章でまとめを行う．

2. メモリコアの組込み自己修復

本研究で対象とする組込み自己修復機能を持つメモリコアおよびそのテストフローを図 1，2 にそれぞれ示す．メモリコアは，複数の冗長セル，BIST 回路，マルチプレクサ，修復解析器 (Repair Analyzer) および Fuse Box で構成される．まず初めに第 1 ステージでは，BIST 回路を用いたテストを行う．BIST 回路でテストパターンを生成し，マルチプレクサを制御することによりメモリにそのパターンを印加する．その応答を BIST 回路に取り込み，期待値との比較を行う．期待値と異なる値を検出した場合，そのデータを故障情報として保存する．第 1 ステージで，故障が検出されなかった場合はパスと判定しテストを終了する．故障が検出された場合は，第 2 ステージへ移り，故障情報をもとに修復解析器を用いて診断・修復を行う．搭載している冗長セルでは修復不可能な場合はフェイルと判定しテストを終了する．搭載している冗長セルで修復可能な場合は，その修復情報が Fuse Box に書き込まれ，メモリコアの再構成が行われる．再構成後，第 3 ステージに移り，再度 BIST 回路によるテストが行われ，パスまたはフェイルの判定が行われる．

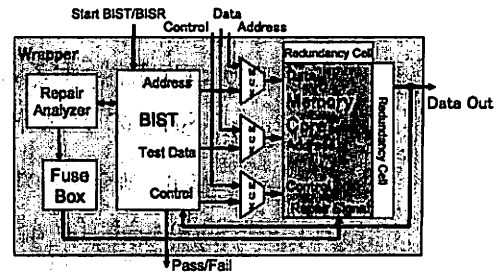

図 1　組込み自己修復機能を持つメモリコア

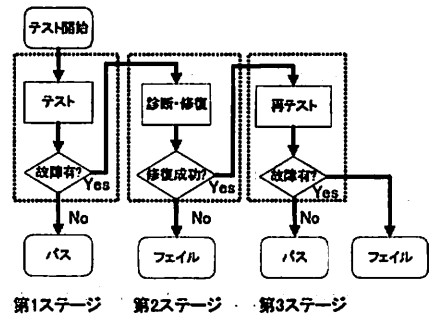

図 2　メモリコアのテストフロー

BISR 機能を持つメモリコアでは，そのサイズや目標とする歩留まりによって様々な数の冗長セルを搭載することが可能である．同様に，テストおよび診断・修復に用いられるアルゴリズムもその目的によって複数考えられる．そこで，本研究では特定の冗長セル数やアルゴリズムを対象とするのではなく，一般的に次の情報が与えられるとする．

各メモリコア c_i は以下の 3 つのステージ $s_{i,1}, s_{i,2}, s_{i,3}$ で構成される．

- $s_{i,1}$：テスト
- $s_{i,2}$：診断修復
- $s_{i,3}$：再テスト

また，各ステージ $j(1 \leq j \leq 3)$ に対して，以下の情報が与えられるとする．

- $time(s_{i,j})$：$s_{i,j}$ の実行時間
- $power(s_{i,j})$：$s_{i,j}$ の実行時における消費電力量
- $pp(s_{i,j})$：$s_{i,j}$ のパス確率
  - $s_{i,j}$ で故障が検出されない確率 $(j = 1, 3)$
  - $s_{i,j}$ で修復可能である確率 $(j = 2)$

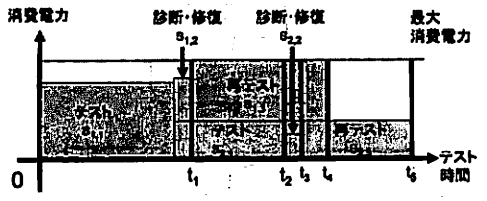

図 3 スケジューリング例

3. テスト時間の期待値

本研究では，複数の組込み自己修復機能を持つメモリコアを対象とし，各メモリコアのテスト，診断・修復および再テストの 3 つのステージの順序関係を満たすようにテストスケジューリングを行う．図 3 に 2 つのメモリコアに対するスケジューリング例を示す．前節で示したように，各メモリコアは最悪の場合 3 つの全てのステージを実行する必要がある．したがってテスト時間は，全てのメモリコアの全てのステージが終了するまでの時間となる．図 3 の例では，テスト時間は t_5 である．しかし，メモリコア c_1 がステージ $s_{1,1}$ でパスと判定され，メモリコア c_2 もステージ $s_{2,1}$ でパスと判定されれば時刻 t_2 でテストを終了することが可能である．同様に，メモリコア c_1 がステージ $s_{1,3}$ でパスと判定され，メモリコア c_2 がステージ $s_{2,1}$ でパスと判定とされていれば時刻 t_4 でテストを終了することも可能である．また，あるコアがフェイルと判定された時刻で対象としている SoC のテストを終了するという abort-on-fail [19] の概念を用いると，図 3 の例においては，t_1, t_3, t_4, t_5 でもテストを終了できる可能性がある．そこで，本研究ではテスト時間の最小化を目的とするのではなく，テスト時間の期待値の最小化を目的としたテストスケジューリングを行う．以下にテスト時間の期待値の計算方法を示す．

メモリテストがパスと判定されるのは，全てのメモリコアでパスと判定される場合である．また，メモリテストがフェイルと判定されるのは，あるメモリコアでフェイルと判定される場合である．各ステージ $s_{i,j}$ のテスト終了時刻を $end(s_{i,j})$，すべてのメモリコアの集合を $C = \{c_1, c_2, \cdots, c_N\}$ とする．メモリテストがパスまたはフェイルと判定される可能性のある時刻は以下のとおりである．

- メモリテストでパスと判定される可能性のある時刻
  - $max_i(end(s_{i,1}))$
  - $end(s_{j,3})$ ただし，$end(s_{j,3}) > max_i(end(s_{i,1})), (1 \leq j \leq N)$
- メモリテストでフェイルと判定される可能性のある時刻
  - $end(s_{i,2}), (1 \leq i \leq N)$
  - $end(s_{i,3}), (1 \leq i \leq N)$

メモリテストがパスと判定される可能性のある時刻を終了時刻に持つステージの集合を S_p とする．図 3 の例では，$S_p = \{s_{2,1}, s_{1,3}, s_{2,3}\}$ となる．このとき，ステージ $s_{i,j}$ の終了時刻 $(= end(s_{i,j}))$ でメモリコア c_k がパスと判定されている確率 $P_p(s_{i,j}, k)$ は以下の式で表現される．

$$P_p(s_{i,j},k) = \begin{cases} \bullet\ end(s_{k,3}) > end(s_{i,j}) \text{ の時} \\ pp(s_{i,1}) \\ \bullet\ \text{その他の時} \\ pp(s_{i,1}) + (1 - pp(s_{i,1})) \\ \cdot pp(s_{i,2}) \cdot pp(s_{i,3}) \end{cases} \quad (1)$$

また，$end(s_{i,j})$ でメモリテストがはじめてパスと判定される確率 $P_p(s_{i,j})$ は以下の式で表される．

$$P_p(s_{i,j}) = \begin{cases} \bullet\ j = 1 \text{ の時} \\ \prod_{k=1}^{N} P_p(s_{i,j},k) \\ \bullet\ j = 3 \text{ の時} \\ (1 - pp(s_{i,1})) \cdot pp(s_{i,2}) \cdot pp(s_{i,3}) \\ \cdot \prod_{\{c_k\} \in C - \{c_i\}} P_p(s_{i,j},k) \end{cases} \quad (2)$$

以上より，メモリテストがパスと判定されるまでの時間の期待値 ET_p は，以下の式で表現される．

$$ET_p = \sum_{s \in S_p} end(s) \cdot P_p(s) \quad (3)$$

同様に，フェイルと判定される可能性のある時刻を終了時刻に持つステージの集合を S_f とする．図 3 の例では，$S_f = \{s_{1,2}, s_{2,2}, s_{1,3}, s_{2,3}\}$ となる．このとき，$end(s_{i,j})$ においてメモリコア c_i 除くメモリコア c_k がフェイルしていない確率 $P_{nf}(s_{i,j},k)$ は以下の式で表現される．

$$P_{nf}(s_{i,j},k) = \begin{cases} \bullet\ end(s_{i,j}) < end(s_{k,2}) \text{ の時} \\ 1 \\ \bullet\ end(s_{k,2}) \leq end(s_{i,j}) \text{ かつ} \\ \quad end(s_{i,j}) < end(s_{k,3}) \text{ の時} \\ 1 - (1 - pp(s_{k,1})) \cdot (1 - pp(s_{k,2})) \\ \bullet\ \text{その他の時} \\ pp(s_{k,1}) + (1 - pp(s_{k,1})) \\ \cdot pp(s_{k,2}) \cdot pp(s_{k,3}) \end{cases} \quad (4)$$

一方，メモリコア c_i が $end(s_{i,j})$ ではじめてフェイルと判定される確率 $P_{cf}(s_{i,j})$ は以下の式で表現される．

$$P_{cf}(s_{i,j}) = \begin{cases} \bullet\ j = 2 \text{ の時} \\ (1 - pp(s_{i,1})) \cdot (1 - pp(s_{i,2})) \\ \bullet\ j = 3 \text{ の時} \\ (1 - pp(s_{i,1})) \cdot pp(s_{i,2}) \cdot (1 - pp(s_{i,3})) \end{cases} \quad (5)$$

式 (4) および (5) より，$end(s_{i,j})$ でメモリテストがはじめてフェイルと判定される確率 $P_f(s_{i,j})$ は以下の式で表現される．

$$P_f(s_{i,j}) = P_{cf}(s_{i,j}) \cdot \prod_{\{c_k\} \in C - \{c_i\}} P_{nf}(s_{i,j},k) \quad (6)$$

以上を用いると，メモリテストがフェイルと判定されるまでの時間の期待値 ET_f は，以下の式で表現される．

$$ET_f = \sum_{s \in S_f} end(s) \cdot P_f(s) \quad (7)$$

上記の式 (6), (7) を用いると，テスト時間の期待値 ET は以

下の式で表現される．

$$ET = ET_f + ET_p \tag{8}$$

また，テスト時間 T は以下の式で表現される．

$$T = max_i(end(s_{i,3})) \tag{9}$$

4. 問題の定式化

消費電力制約下において，テスト時間の期待値の最小化を目的としたメモリコアにおける組込み自己修復を考慮したテストスケジューリング問題を以下の最適化問題として定式化する．

[定義] 組込み自己修復を考慮したテストスケジューリング問題

入力

- メモリコアの集合 $C = \{c_1, c_2, \cdots, c_N\}$
- 最大許容消費電力量 P_{max}
- 各メモリコア c_i $(1 \leqq i \leqq N)$ のステージ $s_{i,j}(1 \leqq j \leqq 3)$ に対して，
  - $time(s_{i,j})$
  - $power(s_{i,j})$
  - $pp(s_{i,j})$

出力

- テストスケジュール
  - $s_{i,j}$のテスト開始時刻 $start(s_{i,j})$ $(1 \leqq i \leqq N, 1 \leqq j \leqq 3)$

目的

- テスト時間の期待値の最小化

制約

- 各コア $c_i(i \leqq i \leqq N)$ に対するステージ間の順序制約
  - $end(s_{i,1}) \leqq start(s_{i,2})$
  - $end(s_{i,2}) \leqq start(s_{i,3})$
- 消費電力制約
  - 任意の時刻における動作中のメモリコアの合計消費電力量 $\leqq P_{max}$

5. テストスケジューリング法

本章では消費電力制約下におけるメモリコアに対する組込み自己修復機能を考慮したテストスケジューリング法について述べる．テスト時間の期待値の最小化のためには，短時間で終了する確率が高いスケジュールが望ましい．テストを終了できる可能性がある時刻は，すべてのメモリコアのテストがパスした時刻，およびあるメモリコアでフェイルと判定された時刻が挙げられる．しかし[16]で示されているように，BISR機能を搭載する理由としては歩留まりの低さが挙げられるので，本研究では早い時刻にフェイルと高い確率で判定されるスケジュールを生成することでテスト時間の期待値の最小化を図る．また，テスト時間を最小化することでテスト時間の期待値も削減可能である．テスト時間の最小化を目的とした数多くのスケジューリング法においては，テスト時間の長い，または消費電力量の大きいコアからスケジュールするのが効果的であると知られている．そこで提案手法では，ステージ $s_{i,j}$ のパス確率 $pp(s_{i,j})$ を用いてそのステージ終了時のフェイル確率を計算し，フェイル確率に基づいたクラスタリングを行う．各クラスタに属するステージ $s_{i,j}$ は同程度のフェイル確率を持つようにクラスタリングを行い，スケジューリングを行う際は高いフェイル確率を持つクラスタを優先する．一方，各クラスタ内では，各ステージ $s_{i,j}$ に対して面積 $(= time(s_{i,j}) \cdot power(s_{i,j}))$ を計算し，その面積の降順にスケジュールすることでテスト時間の最小化を図る．図4に提案するテストスケジューリング法の概要を示し，以下にその詳細を示す．

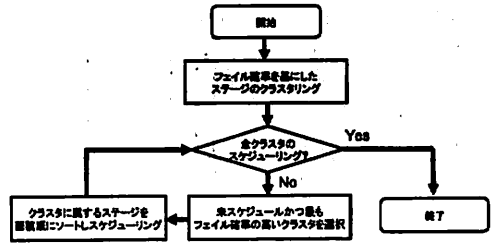

図4 提案スケジューリング法の概要

ステップ1：クラスタリング

本ステップでは，各ステージ $s_{i,j}$ のパス確率 $pp(s_{i,j})$ を用いてクラスタリングを行う．まず初めに，各メモリコア c_i に対して各ステージ $s_{i,j}$ 終了後にフェイルと判定される確率であるフェイル確率 $fp(s_{i,j})$ を計算する．$fp(s_{i,j})$ は，以下の式で表現される．

- $fp(s_{i,1}) = 0$
- $fp(s_{i,2}) = (1 - pp(s_{i,1}))(1 - pp(s_{i,2}))$
- $fp(s_{i,2}) = (1 - pp(s_{i,1}))pp(s_{i,2})(1 - pp(s_{i,3}))$

全てのステージの集合 S を上記のフェイル確率の降順にソートする．3つのメモリコアの合計9ステージをフェイル確率の降順にソートした例を図5に示す．S から下記の条件を満たすステージを取り除いた集合を S_g とする．

- $s_{i,1}$ $(1 \leqq i \leqq N)$
- $fp(s_{i,3}) > fp(s_{i,2})$ を満たす $s_{i,2}$ $(1 \leqq i \leqq N)$

図5の例では，ステージ $s_{2,2}, s_{1,1}, s_{2,1}, s_{3,1}$ が上記の条件を満たし，S_g には含まれない．以下では，S_g に対するクラスタリングを行う．S_g をクラスタリング後，除外されたステージ2は同一メモリコアのステージ3と同一クラスタに属するとし，同様に除外された各メモリコアのステージ1は同一メモリコアのステージ2と同一クラスタに属するとする．このようにすることで，続くステップ2において順序関係制約を満たすスケジュールの生成が保障される．

S_g のフェイル確率の分散 V_g を以下の式を用いて計算する．

$$V_g = \frac{\sum_{s \in S_g} fp(s)^2}{|S_g|} - \left(\frac{\sum_{s \in S_g} fp(s)}{|S_g|} \right)^2 \tag{10}$$

さらに，クラスタを決定するための分散の閾値 V_t を以下の式によって決定する．

$$V_t = \alpha \cdot V_g \tag{11}$$

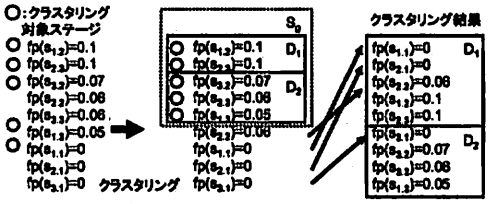

図 5 クラスタリング例

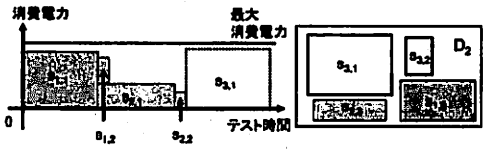

図 6 ステージのスケジュール例

ここで，α は定数とし，実験では α の値によるテスト時間の期待値の影響を評価する．次に要素が空であるクラスタ D_1 を形成し，集合 S_g よりフェイル確率の最も大きいステージ $s_{i,j}$ をクラスタ D_1 に追加した場合のクラスタ D_1 のフェイル確率の分散 V_{D_1} を計算する．$V_{D_1} <= V_t$ ならば $s_{i,j}$ をクラスタ D_1 に追加する．同様に S_g より降順にステージを選択し，$V_{D_1} <= V_t$ が満たされる間処理を繰り返す．$V_{D_1} <= V_t$ を満たさない場合は，新しいクラスタ D_2 を形成し，同様の処理を繰り返す．クラスタリングの対象となっている全てのステージがある 1 つのクラスタに属するまで上記の処理を繰り返す．図 5 の例では，2 個のクラスタが形成されている．

ステップ 2：クラスタ内のスケジューリング

ステップ 1 のクラスタリングが終了すると，フェイル確率の最も高いステージで形成されたクラスタ D_1 を選択し，D_1 に属するステージのスケジューリングを行う．まず初めに，選択されたクラスタに属する各ステージ $s_{i,j}$ に対して面積 $a_{i,j}$ を以下の式で計算する．

$$a_{i,j} = time(s_{i,j}) \cdot power(s_{i,j}) \qquad (12)$$

次に，クラスタ内の未スケジュールのステージで，以下の条件を満たすステージから最大の面積を持つステージ $s_{i,j}$ を選択する．

- $s_{i,1}$ $(1 \leq i \leq N)$
- $s_{i,1}$がスケジュール済みである $s_{i,2}$ $(1 \leq i \leq N)$
- $s_{i,2}$がスケジュール済みである $s_{i,3}$ $(1 \leq i \leq N)$

選択された $s_{i,j}$ のテスト開始時刻 $start(s_{i,j})$ は，消費電力制約およびメモリコア c_i の順序関係制約を満たす最も早い時刻とする．図 6 の例では，スケジュール対象となるクラスタ D_2 で最大の面積を持つステージ $s_{3,1}$ が選択され，消費電力およびメモリコア c_3 の順序関係を満たす最も早い時刻である $end(s_{2,2})$ にスケジューリングされる．上記の処理をステージの面積の降順に繰り返し，クラスタ D_1 に属する全てのステージのスケジュールを決定する．クラスタ D_1 に対する処理が終了すると，以降同様にフェイル確率の高いステージで形成されたクラスタを順に選択し，同様の処理を行う．

6. 実験結果

本章では，表 1 に示す実験用メモリコアに対して提案するテストスケジューリング法を適用した結果を示す．提案手法は C 言語で実装し，PentiumM 1.6GHz, 512MB のメモリ搭載の計算機上で実験を行った．提案手法によるテストスケジューリング結果は，全て 2 秒以下で得ることが出来た．表 1 に実験の対象としたメモリコアの特性を示す．A から H までの 8 種類のメモリコアをそれぞれ“個数”に示す数だけ用いた合計 47 個のメモリコアを対象とした．“$time(s_{i,j})$”，“$power(s_{i,j})$”，“$pp(s_{i,j})$”は，それぞれステージ $s_{i,j}$ のテスト時間，消費電力，パス確率を表す．テスト時間の単位はサイクル数とし，消費電力は メモリコア A の消費電力を 100 ユニットとした場合の相対値を表す．

表 2 に提案手法によるテスト時間およびテスト時間の期待値を示す．各コアのステージは順序関係を満たしていれば連続して実行する必要が無い提案手法に対し，3 つのステージは連続して実行されなければならないとした場合を“制約付手法”とした．最大消費電力 P_{max} は，600，1200，1800 の 3 通りを設定し，α は 10，1，0.1，0.01 の 4 通りに対して実験を行った．提案手法では全ての場合において制約付き手法に比べ短いテスト時間の期待値を達成している．各消費電力制約における最小のテスト時間の期待値を比較すると，提案手法は制約付き手法に比べ平均で約 16%短い．さらに，テスト時間に関しても $\alpha = 10$ を除いた全ての場合で，提案手法は制約付き手法より短い時間を達成している．これらより，各コアのステージは順序関係を満たしていれば連続して実行する必要が無いとすることで，効率的なスケジューリングを達成していることが分かる．また α に関しては，その値を小さく設定するとクラスタ数が多くなりフェイル確率の高いステージの優先順位が上がり，式 (12) で表される面積の大きいステージの優先順位が下がる．与えられた消費電力制約に対して適切な α を選択することにより，フェイル確率と面積を共に考慮したテストスケジューリングが可能となることが分かる．

7. まとめ

本論文は，SoC に搭載される組込み自己修復機能を持つメモリコアに対する消費電力制約下でのテストスケジューリング法を提案した．提案したスケジューリング法では，メモリコアをテスト，診断・修復および再テストの順序関係を持つ 3 つの独立したステージで構成されると考えることで柔軟かつ効率的なスケジューリングを可能とした．また，abort-on-fail の概念を用いてテスト時間の期待値を定義し，短いテスト時間の期待値を実現するアルゴリズムを提案した．評価実験では，各コアにおいてステージが連続して実行されなければならない場合と比較し，テスト時間の期待値を平均で約 16%削減することに成功した．提案法では，各メモリコアが BISR を実現するためのハードウェアを持つとしているが，複数のメモリコアで BISR 回路を共有している場合のスケジューリングが今後の課題とし

表 1 実験用メモリコア

		ステージ 1			ステージ 2			ステージ 3		
メモリコア	個数	$time(s_{i,1})$ [cycles]	$power(s_{i,1})$ [units]	$pp(s_{i,1})$	$time(s_{i,2})$ [cycles]	$power(s_{i,2})$ [units]	$pp(s_{i,2})$	$time(s_{i,3})$ [cycles]	$power(s_{i,3})$ [units]	$pp(s_{i,3})$
A	15	448	100	0.95	500	50	0.92	448	100	0.9
B	10	896	100	0.93	500	50	0.93	896	100	0.95
C	6	1792	100	0.9	500	50	0.88	1792	100	0.97
D	5	3584	200	0.88	1000	100	0.98	3584	200	0.94
E	4	7168	200	0.82	1000	100	0.97	7168	200	0.98
F	4	14336	200	0.81	1000	100	0.95	14336	200	0.94
G	2	28672	400	0.78	2000	200	0.94	28672	400	0.92
H	1	57344	400	0.75	2000	200	0.92	57334	400	0.95

表 2 テスト時間およびテスト時間の期待値

$P_{max} = 600$								
α	10		1		0.1		0.01	
	制約付き手法	提案手法	制約付き手法	提案手法	制約付き手法	提案手法	制約付き手法	提案手法
テスト時間	242752	245712	242752	237116	242752	237960	242752	237960
テスト時間の期待値	218619	201736	212026	202927	212315	**193032**	**211917**	195357
$P_{max} = 1200$								
α	10		1		0.1		0.01	
	制約付き手法	提案手法	制約付き手法	提案手法	制約付き手法	提案手法	制約付き手法	提案手法
テスト時間	121688	152528	121688	119292	121688	119688	121688	119688
テスト時間の期待値	115022	107998	113422	109977	113364	**105193**	**113268**	109301
$P_{max} = 1800$								
α	10		1		0.1		0.01	
	制約付き手法	提案手法	制約付き手法	提案手法	制約付き手法	提案手法	制約付き手法	提案手法
テスト時間	116688	138192	116688	116688	116688	116688	116688	116688
テスト時間の期待値	94177	75680	91939	67245	**91832**	64369	91972	**62401**

て挙げられる.

謝辞 本研究に際し，多くの貴重な意見を頂いた本学の井上美智子助教授，大竹哲史助手はじめコンピュータ設計学講座の諸氏に深く感謝します．本研究は一部，日本学術振興会科学技術研究費補助金・基盤研究 B(2)(課題番号 15300018) および若手研究 (B)(課題番号 18700046) の研究助成による．

文 献

[1] Y. Zorian, E. J. Marinissen and S. Dey, "Testing Embedded Core-Based System Chips," Proc. IEEE International Test Conference, pp.130-143, 1998.

[2] E. J. Marinissen, R. Kapur, M. Lousberg, T. McLaurin, M. Ricchetti and Y. Zorian, "On IEEE P1500's Standard for Embedded Core Test," Journal of Electronic Testing, Vol.18, Numbers 4/5, pp.365-383, Aug/Oct 2002.

[3] K. Chakrabarty, "Design of system-on-a-chip test access architectures using integer linear programming," Proc. 18th VLSI Test Symp., pp.127-134, May 2000.

[4] V. Iyengar, K. Chakrabarty and E. J. Marinissen, "On using rectangle packing for SOC wrapper/TAM co-optimization," Proc. 20th VLSI Test Symp., pp.253-258, Apr. 2002.

[5] Z. He and Z. Peng, "Power Constrained and Defect-Probability Driven SoC Test Scheduling with Test Set Partitioning," Proc. Degin, Automation, and Test in Europe, pp.1-6, Mar. 2006.

[6] W. Zou, S. M. Reddy, I. Pomeranz and Y. Huang, "SOC Test Scheduling Using Simulated Annealing," Proc. 21th VLSI Test Symp., pp.325-330, May 2003.

[7] B. H. Fang and N. Nicolici, "Power-Constrained Embedded Memory BIST Architecture," IEEE International Symp. on Defect and Fault Tolerance in VLSI Systems, pp.451-458, 2003.

[8] W. Wang, "March Based Memory Core Test Scheduling for SOC," Proc. of 13th Asian Test Symp., pp.248-253, 2004.

[9] E. J. Marinissen, B. Prince, D.Keitel-Schulz and Y.Zorian, "Challenges in Embedded Memory Design and Test," Proc. Degin, Automation, and Test in Europe, pp.722-727, Mar. 2005.

[10] International Technology Roadmap for Semiconductors 2005 Edition. (Design) http://www.itrs.net/

[11] A. J. van de Goor, Testing Semiconductor Memories Theory and Practice, John Wiley and Sons, 1991.

[12] E. Nelson, J. Dreibelbis and R. McConnell, "Test and Repair of Large Embedded DRAMs: part 2," Proc. IEEE International Test Conference, pp.173-181, 2001.

[13] S. Lu and S. Huang, "Built-in Self-Test and Repair (BISTR) Techniques for Embedded RAMs," International Workshop on Memory Technology, Design and Testing, pp.60-64 2004.

[14] I. Kim, Y. Zorian, G. Komoriya, H. Pham, F. P. Higgins and J. L. Lewandowski, "Built in Self Repair for Embeded High DensitySRAM," Proc. IEEE International Test Conference, pp.1112-1119, 1998.

[15] V. Schöber, S. Paul and O. Picot, "Memory Built-In Self-Repair using redundant words," Proc. IEEE International Test Conference, pp.995-1001, 2001.

[16] T. Kawagoe, J. Ohtani, M. Niiro, T. Ooishi, M. Hamada and H. Hidaka, "A Built-In Self-Repair Analyzer (CRESTA) for embedded DRAMs," Proc. IEEE International Test Conference, pp.567-574, 2000.

[17] M. Nicolaidis, "Theory of Transparent BIST for RAMS," IEEE Trans. on Computer, Vol.45, No.10, pp.1141-1156, Oct. 1996.

[18] Y. Zorian and S. Shoukourian, "Embeded Memory Test and Repair Infrastructure IP for SOC Yield," Proc. IEEE International Test Conference, pp.340-349. 2002.

[19] E. Larsson J. Pouget and Z. Peng, "Abort-on-Fail Based Test Scheduling," Journal of Electronic Testing, vol.21, pp.651-658, 2005.