設計自動化 72－9
(1994. 10. 27)

大規模LSI用分散型BIST構成法とその設計支援環境

池永 剛　　小倉 武

NTT LSI研究所

神奈川県厚木市森の里若宮3-1

あらまし　BISTインプリメントの短TAT化、BIST付加回路のハードオーバヘッド削減を目的とした、BIST設計支援環境並びにBIST回路を提案する。分散型BISTを対象とすることにより、短TATインプリメントを可能とし、各ブロックに配置するパターン発生器、圧縮器のハード量を削減を行なうことにより、BISTトータルのオーバヘッド削減を行なっている。画像処理用LSIの計110k gateの演算ブロックを対象として評価を行なった結果、5日のTATで、ハードオーバヘッド1%程度、故障検出率96%以上のBISTが実現可能なことを確認した。本環境を用いることにより、開発期間重視でかつ量産を目的としたLSIのテストコストを削減できる。

和文キーワード　分散型BIST、設計支援環境、TAT、ハードオーバヘッド、量産LSI

A distrubuted BIST Technique and its Test Design Platform for VLSIs.

Takeshi IKENAGA　　Takeshi OGURA

NTT LSI Laboratories

3-1, Morinosato Wakamiya, Atsugi-Shi, Kanagawa

Abstract　This paper proposes a test design platform for shorter turn-around-time (TAT) implementation of a distributed built-in self-test (BIST). This distributed BIST technique has lower hardware-overhead pattern generators and compressors. Experimental results for the 110k-gate arithmetic execution blocks of an actual image-processing LSI show that this environment enables the BIST to achieve about a 1% hardware overhead and more than 96% fault coverage within five days. This platform will significantly reduce testing costs for time-to-market and mass-production LSIs.

英文 key words　distributed BIST, test design platform, TAT, hardware overhead, mass-production LSIs

1 はじめに

LSIの不良品出荷率を減らすためには、高い故障検出率を持つテストベクタでテストする必要があるが、LSIの大規模化にともない、高い故障検出率を達成するのが困難になりつつある。このため、LSI設計時においてテスト容易性を考慮した、テスト容易化設計 (DFT: Design for Testability) が必要不可欠になってきている [1] [2]。代表的なテスト容易化手法として、SCAN手法 [3] が考案されており、実用に供されている。SCAN手法は、デザインルールチェッカ、SCAN自動挿入ツール、ATPG等のツールが整備されており、人手をほとんど介することなくインプリメントすることが可能である。しかし、面積、速度オーバヘッドが大きく、量産を前提としたフルカスタムLSIには不向きである。

オーバヘッドの少ないテスト容易化手法として、パターン発生器、圧縮器から構成されるBIST (Built-In Self Test) [4] が注目されてきており、マイクロプロセッサ等のLSIに対する適用例が報告されている [5] – [8]。これらのBIST手法は、集中管理型 (Centralized BIST) と分散型 (Distributed BIST) の2つに大きく分類することができる [2]。このうち、集中管理型は、1組のパターン発生器、圧縮器を用い、機能ブロックをバス等を切替えながら順々にテストしていく手法であり、オーバヘッド削減効果がおおきいため、適用例が多い [5] – [7]。しかし、集中管理型は、一般に複雑なテストシーケンスが必要なため、BIST制御回路が複雑となり、BISTの組み込み工数が多く必要になるという問題がある。また、BISTの評価単位が大きくかつテストベクタ長が長くなるため、故障検出率算出等が困難で、BISTの評価工数を多く要するという問題がある。

一方、分散型BISTは、各機能ブロック毎にパターン発生器、圧縮器を配置し、各機能ブロックを並列にテストする手法である。分散型BISTは、各ブロックを一括制御可能なため、BIST制御回路の作成、組み込みが容易である。また、BIST対象ブロック毎に並行して、故障検出率算出等が行なえるため、評価時間短縮が図れる。このため、BISTの短TATインプリメントが可能である。しかし、各機能ブロック毎に配置するパターン発生器、圧縮器としてハード量の大きなLFSR (Linear feedback shift register)、MISR (Multiple Input Signature Register) 等を用いた場合 [8]、BIST付加回路によるハードオーバヘッドが増大するという欠点がある。

BISTインプリメントの短TAT化、BIST付加回路のハードオーバヘッド削減の両者を満たすことが、開発期間重視 (time to market) でかつ量産を目的としたLSI向きのDFT手法を実現する上で重要なポイントとなる。そこで、これらを目的としたBIST設計支援環境並びにBIST回路を提案する。分散型BISTを対象とすることにより、短TATインプリメントを可能とし、各ブロックに配置するパターン発生器、圧縮器のハード量を削減を行なうことにより、BISTトータルのオーバヘッド削減を行なっている。本稿では、プロトタイプを行なったBIST設計支援環境の構成について述べ、短TAT化を可能とする分散型BISTモデル、ハードオーバヘッドが小さいパターン発生器、圧縮器構成を示す。最後に、本設計支援ツールを実LSIに適用し、TAT、ハードオーバヘッド、故障検出率等の評価結果を示す。

2 分散型BISTの設計支援環境

BISTは、どの種の回路でも適用可能であるというオールラウンドの手法ではない。このため、そのインプリメントには、BIST回路を組み込み、その有効性を調べるという試行を繰り返すことが必要不可欠となる。よって、分散型BISTを効率良く実現するためには、

(2.1) パターン発生器、圧縮器、BIST制御回路等の各種BIST回路の作成、組み込みの短TAT化

(2.2) ハードオーバヘッド、故障検出率等のBIST有効性評価の短TAT化

が必要となる。そこで、DEBUG (Distributed BIST units generator) とBEST (BIST evaluation system) から構成される分散型BISTの設計支援環境を構築した。DEBUGは(2.1)に、BESTは(2.2)に寄与する。

DEBUGは、図1に示すように、リスト形式で与えられるビット幅等のパラメータを与えることにより、各種パターン発生器、圧縮器を自動生成する。また、テストサイクル等のパラメータを与えることにより、BIST制御回路を自動生成する。

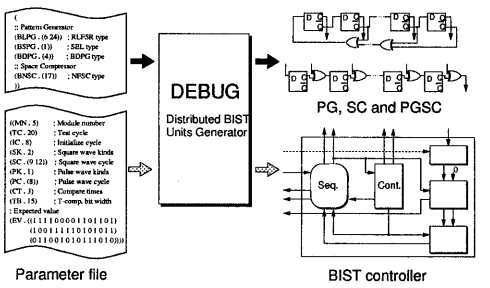

図1: DEBUGの構成

BIST回路の設計は、テストに関する高度な知識を必要とし、多くの設計、検証工数を必要としていたが、DEBUGを用いることにより、LSI設計者自ら、各種BIST回路を短TATに生成できる。一方、生成されたBIST回路の組み込みは、現時点では人手で行なう必要がある。しかし、DEBUGが生成するBIST回路は全て論理合成可能な機能記述言語であるため、機能記述段階で効率的なBIST組み込みが可能である。

BESTは、BIST有効性評価に特化したフレームワークであり、種々のベンダーツールと自社ツールから構成されている。現在、BIST有効性評価に有用なツールとして、各種シミュレータ、論理合成ツール等が開発されているが、実際にこれらのツールを組み合わせて、種々BISTの評価を行なうためには、ツール起動用のスクリ

プト作成並びに起動、各種ツール間のファイル (ネットリスト、テストパターン等) 変換等の繁雑な作業が必要であった。BEST は、BIST 評価の各項目別に、これらの作業を全て自動で行なうことにより、BIST 有効性評価の短 TAT 化を実現している。例えば、BIST 対象ブロックの機能記述から故障検出率を算出する場合、図 2 の一連の作業を自動で行なう。

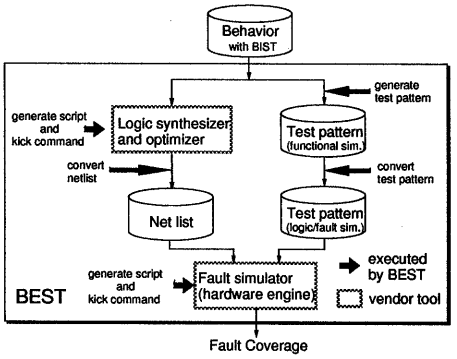

図 2: BEST 機能の例 (故障検出率算出)

DEBUG と BEST は、図 3 に示すように、ブロック設計段階と全体設計段階に分かれた分散型 BIST の設計フロー全体をサポートする。

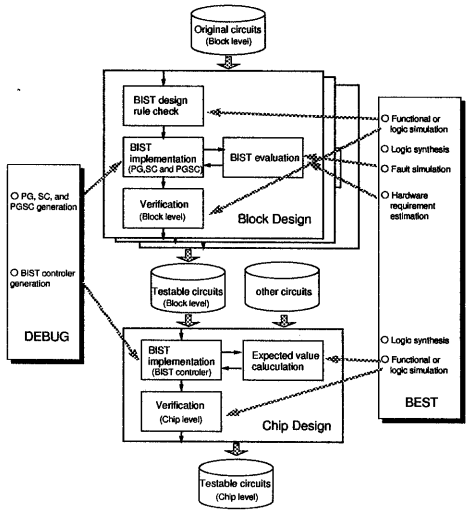

図 3: 分散型 BIST 設計フロー

DEBUG は、ブロック設計段階における各種パターン発生器、圧縮器の生成、全体設計段階における BIST 制御回路の生成といった BIST インプリメントの支援を行なう。BEST は、ブロック設計段階では、ハード量算出、故障シミュレーション、検証等を統一的に行なう環境を提供し、全体設計段階では、期待値作成、検証等を統一的に行なう環境を提供する。DEBUG と BEST を用いることにより、分散型 BIST 設計フローの中で特に工数を必要とする部分を効率良く行なうことが可能となり、分散型 BIST 設計全体の短 TAT 化が行なえる。

3 短 TAT 化を可能とする分散型 BIST

本設計支援環境で対象とする、一括制御タイプの分散型 BIST のモデルを図 4 に示す。

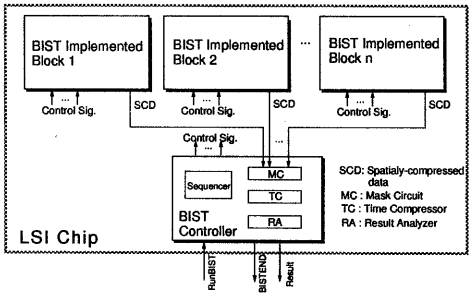

図 4: 分散型 BIST のモデル

各部の BIST インプリメントが、並行して効率良く行なえるように、分散型 BIST は、複数個の BIST 対象ブロック (BIST implemented block) と BIST 制御回路 (BIST controller) に明確に切り分けている。また、BIST 制御回路の構成をできるだけ簡易にするために、全 BIST 対象ブロックは、BIST 制御回路から生成される制御信号によって一括制御される。パターン圧縮は、空間 (各 BIST 対象ブロック) +時間 (BIST 制御回路) の 2 段階に分けて行なうことにより [9]、期待値、結果解析器を 1 つに集約可能な構成にし、BIST 制御回路の簡易化、小ハードオーバヘッド化を図っている。

3.1 BIST 対象ブロック

BIST 対象ブロックは、図 5 に示すように、機能ブロックの入力部にパターンを与えるパターン発生器 (PG : Pattern generator)、機能ブロックから出力パターンを圧縮する空間圧縮器 (SC : Space compressor)、機能ブロック内部の可制御、可観測性を高めるパターン発生器・空間圧縮器 (PGSC : Pattern generator/Space compressor) から構成される。

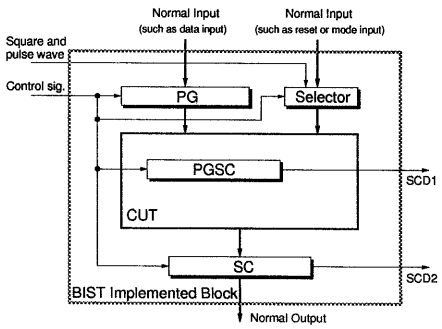

図 5: BIST 対象ブロックのモデル

PG あるいは SC としては、機能ブロックの入出力部に通常レジスタが存在する場合、通常モード時は通常レジスタとして、BIST モード時はそれぞれパターン発

生器、空間圧縮器として動作する共有型 (shared type) を用いる。存在しない場合は、PG、SC を独立に設ける (add-on type)。両者は、BIST 制御回路から与えられる BIST モード信号、BIST 初期化信号により制御される。また、SC は、BIST 制御回路に対し、空間圧縮データ (SCD) を渡す。機能ブロックの入力としては、乱数では制御し難いモード切替え入力、リセット入力等があるが、これらの入力に対しては、BIST 制御回路から与えられる任意サイクルの方形波信号、パルス波信号をセレクタで切替えて与える。

ブロックの入出力部のみに PG、SC を配置するだけでは十分な故障検出率が得られない場合は、機能ブロック中のレジスタを、通常レジスタ、パターン発生器、空間圧縮器共用の PGSC に置き換え、可観測性、可制御性を高める。PGSC としては、通常レジスタ共有型の MISR を用いる。PGSC に置き換えるレジスタは、トグルシミュレーション、あるいは乱数伝搬度 [10] 等のメジャを用いて特定する。

以上の構成により、種々の機能ブロックに対し、高い故障検出率を持った BIST 対象ブロックを構成可能である。またこれらは、4 種制御信号によって動作するため、制御が容易である。

3.2 BIST 制御回路

分散型 BIST の制御回路は、シーケンサ (Sequencer)、マスク回路 (MC : Mask Circuit)、時間圧縮器 (TC : Time Compressor)、結果解析器 (RA : Result Analyzer) から構成される。

シーケンサは、各 BIST 対象ブロックに対し 4 種の制御信号を生成する。対象 LSI によって、必要なテストサイクル、初期化サイクル等は変わってくるため、これらの信号は、全てパラメータによって可変な構成にしている。

マスク回路は、特定の BIST 対象ブロックからの空間圧縮データのみを時間圧縮器に渡し、残りは固定値 (0) を渡す機能を持つ。これにより、各 BIST 対象ブロックを独立してテストすることが可能となり、故障ブロックの特定がおこなえる。

時間圧縮器は、各 BIST 対象ブロックからの空間圧縮データを時間圧縮する機能を持つ。時間圧縮器としては、MISR を用いることにより、故障マスク率を減らしている。

結果解析器は、時間圧縮器の圧縮結果と期待値を比較し、結果をチップ外へ出力する機能を持つ。時間圧縮器 (MISR) のビット幅が小さい (20 ビット以下) 場合、故障マスク率が顕著になるが、この場合期待値比較を複数回行なう [11] ことにより、故障マスク率を低減可能な機構を持っている。

4 オーバヘッドの小さな PG、SC 構成

分散型 BIST において、PG、SC は各機能ブロック毎に配置するため、それらのハードオーバヘッド削減が重要な課題となる。PG、SC のオーバヘッド削減手法として、BILBO 手法 [12] の様に、PG、SC を通常レジスタと共有化する手法が有効であり、**3.1** で示したように本手法で用いる PG、SC においても採用している。本稿では、ハードオーバヘッドの小さな PG、SC 構成法として、さらに以下の 4 つを提案する。

反転シフト型の LFSR　通常の LFSR は、オール 0 パターンに初期化 [13] できないため、リセットあるいはプリセット付きのレジスタを shared type の PG に置き換える場合において、通常系と BIST 系で初期化機構を別々に設ける必要があった。そこで、オール 0、1 パターンに初期化可能な反転シフト型の LFSR (RLFSR : reverse shift LFSR) を考案し、初期化機構の共有化を可能にした (図 6)。

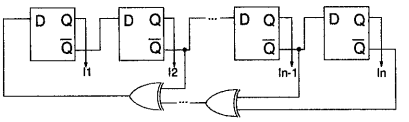

図 6: RLFSR の構成

4 ビット反転シフト型 LFSR(特性多項式 $= x^4 + x + 1$、初期値 $= 0000$) から生成されるパターンを図 10 に示す。4 ビット反転シフト型 LFSR は、1010 パターンを除く 15 パターンを生成可能である。

S0	0000 (Initial state)	S8	1110
S1	0111	S9	1000
S2	0100	S10	0011
S3	0101	S11	0110
S4	1101	S12	1100
S5	1001	S13	0001
S6	1011	S14	1111
S7	0010	S15=S0	0000

図 7: 4-bit RLFSR の出力パターン

ビット分配型パターン発生器　機能ブロックの入力部に通常レジスタが存在しない場合、add-on type の PG を用いる必要があるが、add-on type は、乱数発生器を構成するためのレジスタを新たに追加する必要があるため、ハードオーバヘッドが大きい。そこで、BIST 対象ブロックの入力数よりも少ないビット幅の疑似乱数パターン発生器を用い、その出力を共有して対象ブロック入力に与えることでハードオーバヘッド削減を行なったビット分配型パターン発生器 (BDPG : bit distributed pattern generator) を考案した。シミュレーションにより、BDPG は、演算器等の回路に対し、故障検出能力を落さずに、BIST 回路のハード量削減が行なえることを確認している [9]。

No Feedback 型空間圧縮器　SC としては、MISR を用いた例[8]が知られているが、多数のフィードバックループを有するため、オーバヘッドが大きい。そこで、No Feedback 型空間圧縮器 (NFSC : no feedback space compressor) を考案した (図 10)。

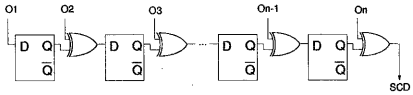

図 8 : NFSC の構成

No Feedback 型は、フィードバックループを持たないため、少ないハード量で実現できる。また、No Feedback 型、MISR とも空間圧縮器として用いる場合、故障マスクを生じさせないためには、空間圧縮データ (SCD) に一度でも故障の影響を伝搬させればよい。よって、故障マスク率は極めて小さな値となり、実用上問題ない。

多段空間圧縮器　機能ブロックの出力部に通常レジスタが存在しない場合、add-on type の SC を用いる必要があるが、MISR 等を用いた場合、オーバヘッドが大きい。そこで、故障マスクが生じないビット幅までハード量の小さな ExOR 圧縮器で空間圧縮を行ない、その後、NFSC 等で 1 本に空間圧縮する多段空間圧縮器 (MSSC : multi-stage space compressor) を考案した。シミュレーションにより、多段空間圧縮器は、演算器等の回路に対し、故障検出能力を損なうことなく、BIST 回路のハード量削減が可能であることを確認している[9]。

5　実 LSI への適用、評価

5.1　BIST 対象ブロック

画像処理用 LSI の 4 種の演算ブロック (DTPATH …乗算器、加算器等のパイプライン演算パス、PEARY1-3 …ハード量の異なるデータパスの 2 次元アレイ) を用い、BIST 対象ブロックの評価を行なう。

まずはじめに、DTPATH に対する BIST 対象ブロックの設計例を図 11 に示す。

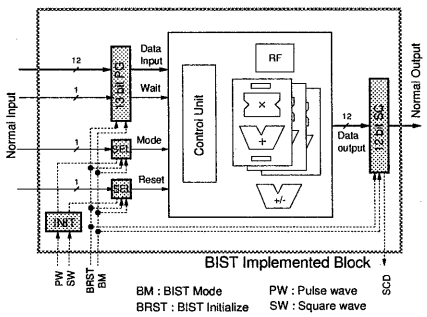

図 9: BIST 対象ブロックの設計例 (DTPATH)

DTPATH は、データ入力 (12 ビット)、ウエイト・モード切替え・リセット入力、データ出力 (12 ビット) の各入出力を持つ。データ入力、ウエイト入力に対しては、13 ビットの RLFSR から生成される乱数を与える。モード切替え入力、リセット入力に対しては、それぞれ 2^{14} 周期の SW、2^{12} 周期の PW をセレクタで切替えて与える。モード切替え入力、リセット入力に対し乱数を与えた場合、制御ユニットの状態遷移が行なわれないため、故障検出率はあがらない。データ出力に対しては、12 ビットの NFSC を用いる。INIT は、初期化モード時に、SW、PW を組み合わせて、DTPATH 内のレジスタの初期化を行なう。DTPATH に対しては、内部のデータフローが比較的簡易であるため、入出力部のみ BIST 化することで、十分な故障検出率が得られた。

次に、4 種の演算ブロックに対し、BIST 対象ブロックを構成した場合のハード量、故障検出率評価結果を表 1 に示す。いずれも 1% 程度のハードオーバヘッドで 96% 以上の故障検出率 (冗長故障を含むため実際はさらに高い検出率となる) を得ることができた。

表 1: ハードオーバヘッド、故障検出率比較

Circuits	Gate 数	BIST オーバヘッド	故障検出率 †
DTPATH	40.3k	0.31%	96.0%
PEARY1	25.0k	0.82%	97.6%
PEARY2	22.4k	1.11%	97.7%
PEARY3	26.4k	1.00%	97.4%

† … 100 万パターンで算出、冗長故障を含む

図 10 に、上の演算ブロックに適用した 3 種の PG、SC に対し、本手法と従来手法 (レジスタ非共有の LFSR、MISR を想定) でハードオーバヘッド比較 (従来手法のオーバヘッドを 1 とする) を行なった結果を示す。4 で述べた、ハードオーバヘッド削減のための各種テクニックを組み合わせることにより、本手法は、従来手法と比較して、$\frac{1}{3}$ 程度のハードオーバヘッドで実現できることを確認した。

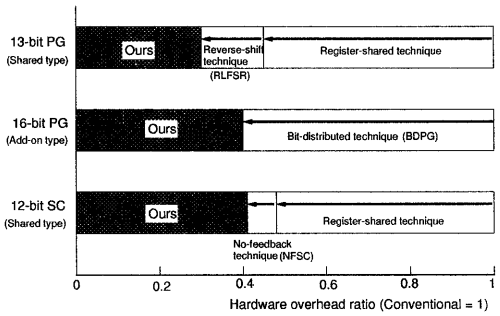

図 10: ハードオーバヘッドの比較

図 11 に DTPATH と PEARY1 の、ブロック設計段階の各種工程に要した TAT を示す。DPTATH は、規模が大きなことと検出率の立ち上がりが悪いため、故障シミュレーションに多く時間を要している。PEARY1 は、first try では、十分な故障検出率が得られなかったので、BIST 再構成 (パターン発生器を追加) を行なっ

ている。BIST 対象ブロックのインプリメントは、本 BIST 設計支援環境を用いることにより、数時間から 3 日 (4 つの演算ブロック合計で 4 日強) で実現できており、効率的に行なえることを確認した。

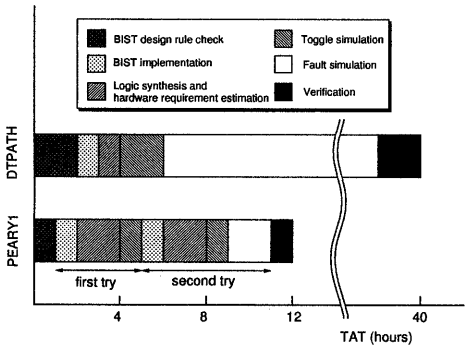

図 11: BIST 対象ブロックの TAT

5.2 BIST 制御回路

5.1 の 4 つの BIST 対象ブロックに対する BIST 制御回路 (テストサイクル = 2^{20}、時間圧縮器 = 15 ビット、期待値比較 = 3 回) を生成した結果、ハード量は約 500 ゲートとなった。各 BIST 対象ブロックのハードオーバヘッドと合わせて、BIST 付加回路のトータルのハードオーバヘッドは 1.2% となる。

図 12 に本設計支援環境を用いた場合と、用いなかった場合において、全体設計段階の各種工程に要した TAT を示す。DEBUG を用いない場合、BIST 制御回路の設計、検証に特に多くの工数を要したが、DEBUG を用いることにより、これらの工数を大幅に削減することができた。また、BEST を用いることにより、期待値算出等に必要な繁雑な作業を削減できた。この結果、BIST 全体設計は、期待値算出を含めて半日 (従来の $\frac{1}{6}$ 程度) の TAT で実現可能なことを確認した。

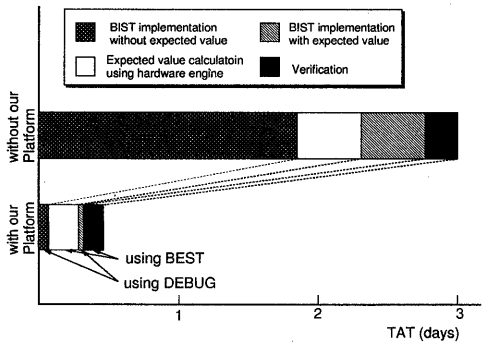

図 12: BIST 制御回路の TAT

6 おわりに

本稿では、BIST インプリメントの短 TAT 化、BIST 付加回路のオーバヘッド削減を目的とした、BIST 設計支援環境の検討を行なった。本設計支援環境は、各種 BIST 回路の自動生成を行なう DEBUG と、BIST 評価を効率良く行なう BEST から構成される。BIST モデルとしては、短 TAT 化が可能な分散型を取り上げた。また、各ブロックに配置するパターン発生器、圧縮器のハード量を削減を行なうことにより、BIST トータルのオーバヘッド削減を行なった。画像処理用 LSI の合計 110k gate の演算ブロックを対象として評価を行なった結果、BIST 制御回路のインプリメントが従来の $\frac{1}{6}$ の TAT に削減できたのをはじめ、BIST 全体は、5 日の TAT でインプリメント可能なことを確認した。また、インプリメントした BIST は、1% 程度のハードオーバヘッドで、96% 以上の故障検出率が得られることを確認した。本環境を用いることにより、開発期間重視でかつ量産を目的とした LSI のテストコストを削減できる。今後は、本環境を演算ブロック以外の回路に適用し、その有効性を調べていく。また、BIST 化するレジスタを判断するためのメジャ等の検討を行なうとともに、BIST 回路自動組み込みツール等を開発し、SCAN 並の使い勝手を持った BIST 設計支援環境を構築していく。

謝辞

本研究を遂行するにあたり、ご指導、ご鞭撻頂きました、LSI 研究所 高機能 LSI 研究部 唐津部長、信号処理 LSI 研究グループ笠井グループリーダーに感謝します。また、今回の検討に対し貴重なご意見を頂きました、信号処理 LSI 研究グループ各員にに感謝します。

参考文献

1. 菅野卓雄、堀口勝治 :"ULSI 設計技術 ", 電子情報通信学会, (1993).
2. H. Abramovici, et al., "Digital Systems Testing and Testable Design", Computer Science Press, pp. 457-540 (1990).
3. E. B. Eichelberger, et al., "A Logic Design Structure for LSI Testing", Proc. 14th DAC, pp. 462-468 (1977).
4. P. H. Bardell, et al., "Built-In Test for VLSI : Pseudorandom Techniques", John Wiley & Sons Inc., pp. 336-338 (1987).
5. J. Kuban, et al., "The MC6804P2 Built-In Self-Test", IEEE ITC, pp. 295-300 (1983).
6. Y. Nozuyama, et al., "Design for Testability of a 32-bit Microprocessor", IEEE ITC, pp. 172-182 (1988).
7. N. Sakashita, et al., "Built-In Self-Test in a 24 bit Floating Point Digital Signal Processor", IEEE ITC, pp. 880-885 (1990).
8. P. P. Gelsinger, " Design and test for the 80386", IEEE Design & Test of Comp., 14, 3, pp. 42-50 (1987).
9. 池永、高橋: 大規模 LSI の演算ユニット向き Built-In Self-Test 構成法、信学論 C-II、(1994.5 採録決定済)
10. 池永、高橋、小倉: 乱数伝搬度に着目した BIST 評価手法、第 45 回情処全大、1K-08、(1993)
11. 鮫島、北村、深沢: 最小故障見逃し率を保証する複数シグネチャ解析、信学技報 FTS91-32、(1991)
12. B. Konemann, et al., "Built-In Logic Block Observation Techniques", IEEE ITC, pp. 37-41 (1979).
13. W. W. Paterson, et al., "Error-Correcting Codes", MIT Press (1972).