誤り訂正可能な巡回符号を用いた
高信頼組込み自己テストに関する研究

2014年3月

広島市立大学大学院
情報科学研究科 博士後期課程
情報科学専攻

深澤 祐樹
目次

第1章 序論 3

第2章 論理回路のテストと高信頼組込み自己テスト 9
  2.1 論理回路のテスト .............................. 10
    2.1.1 論理回路の故障モデル ....................... 10
    2.1.2 テストパターン ................................. 11
  2.2 組込み自己テスト ................................. 12
    2.2.1 テスト生成器 ................................. 15
    2.2.2 応答圧縮器 ................................. 19
  2.3 高信頼設計 ........................................ 22
    2.3.1 静的マスク ..................................... 23
    2.3.2 動的マスク ..................................... 24
    2.3.3 符号化を利用した誤り訂正 ...................... 25
  2.4 高信頼組込み自己テスト ......................... 28
    2.4.1 BIST 回路故障の影響 ......................... 28
    2.4.2 高信頼組込み自己テスト ....................... 31
    2.4.3 テスト容易な応答圧縮器 [7] ................... 33
    2.4.4 誤り検出可能なテスト生成器 [2] .............. 36
  2.5 まとめ ............................................ 41

第3章 動的マスクを実現する耐故障テスト生成器 42
  3.1 はじめに .......................................... 42
  3.2 誤り検出機構を利用したテスト生成器 ............. 43
    3.2.1 待機冗長型テスト生成器 ..................... 43
    3.2.2 縮退利用型テスト生成器 ..................... 46
    3.2.3 テスト生成器のユニット数と面積および故障検出率の関係 48
  3.3 製造テストにおける評価 .......................... 52
    3.3.1 製造テストにおけるテスト生成器の信頼度評価式 .... 53
    3.3.2 テスト生成器の面積 ........................... 56
    3.3.3 テスト生成器の信頼度 ........................... 57
    3.3.4 テスト生成器の故障検出率 ..................... 60
<table>
<thead>
<tr>
<th>章目</th>
<th>内容</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>第4章</td>
<td>静的マスクを実現する耐故障テスト生成器</td>
<td>64</td>
</tr>
<tr>
<td>4.1</td>
<td>はじめに</td>
<td>64</td>
</tr>
<tr>
<td>4.2</td>
<td>巡回符号を利用した誤り訂正可能テスト生成器</td>
<td>65</td>
</tr>
<tr>
<td>4.3</td>
<td>フィールドテストにおける評価</td>
<td>69</td>
</tr>
<tr>
<td>4.3.1</td>
<td>フィールドテストにおけるテスト生成器の信頼度評価式</td>
<td>69</td>
</tr>
<tr>
<td>4.3.2</td>
<td>誤り訂正能力と面積および期待動作確率の関係</td>
<td>74</td>
</tr>
<tr>
<td>4.3.3</td>
<td>テスト生成器の面積と期待動作確率</td>
<td>77</td>
</tr>
<tr>
<td>4.3.4</td>
<td>オンライン BIST における信頼度評価</td>
<td>80</td>
</tr>
<tr>
<td>4.4</td>
<td>まとめ</td>
<td>86</td>
</tr>
<tr>
<td>第5章</td>
<td>誤り訂正を利用した耐故障応答圧縮器</td>
<td>87</td>
</tr>
<tr>
<td>5.1</td>
<td>はじめに</td>
<td>87</td>
</tr>
<tr>
<td>5.2</td>
<td>巡回符号を用いた誤り検出可能な応答圧縮器</td>
<td>88</td>
</tr>
<tr>
<td>5.2.1</td>
<td>符号化応答圧縮器</td>
<td>88</td>
</tr>
<tr>
<td>5.2.2</td>
<td>符号化応答圧縮器の構成法</td>
<td>92</td>
</tr>
<tr>
<td>5.2.3</td>
<td>自己判別率の期待値 $SE^E$</td>
<td>95</td>
</tr>
<tr>
<td>5.3</td>
<td>巡回符号を用いた誤り訂正可能な応答圧縮器</td>
<td>98</td>
</tr>
<tr>
<td>5.3.1</td>
<td>誤り訂正可能な応答圧縮器</td>
<td>98</td>
</tr>
<tr>
<td>5.3.2</td>
<td>誤り訂正可能な応答圧縮器における自己判別率</td>
<td>103</td>
</tr>
<tr>
<td>5.3.3</td>
<td>誤り訂正可能な応答圧縮器における期待動作確率</td>
<td>104</td>
</tr>
<tr>
<td>5.4</td>
<td>実験</td>
<td>111</td>
</tr>
<tr>
<td>5.4.1</td>
<td>符号化応答圧縮器の面積と自己判別率およびシグネチャ観測回数の関係</td>
<td>112</td>
</tr>
<tr>
<td>5.4.2</td>
<td>シグネチャ観測回数とテスト実行時間に関する考察</td>
<td>116</td>
</tr>
<tr>
<td>5.4.3</td>
<td>誤り訂正可能な応答圧縮器の自己判別率</td>
<td>117</td>
</tr>
<tr>
<td>5.4.4</td>
<td>誤り訂正可能な応答圧縮器の期待動作確率</td>
<td>123</td>
</tr>
<tr>
<td>5.4.5</td>
<td>オンライン BIST における耐故障応答圧縮器の信頼度評価</td>
<td>126</td>
</tr>
<tr>
<td>5.5</td>
<td>まとめ</td>
<td>128</td>
</tr>
<tr>
<td>第6章</td>
<td>結論</td>
<td>129</td>
</tr>
<tr>
<td>謝辞</td>
<td></td>
<td>133</td>
</tr>
<tr>
<td>参考文献</td>
<td></td>
<td>135</td>
</tr>
<tr>
<td>関連論文一覧</td>
<td></td>
<td>137</td>
</tr>
</tbody>
</table>
第1章 序論

近年、半導体製品は微細化・高性能化に伴い、大規模集積回路 (LSI: Large Scale Integrated circuit) のように身の回りに幅広く普及している。そのため半導体製品の信頼性確保が非常に重要な課題となっている。LSIの欠陥による故障が存在する場合、それを構成するシステムではサービスの停止・中断だけでなく、社会的に大きな影響を及ぼすことが想定される。信頼性の高いLSIの設計・製造するためには製造後のテスト(故障の有無を調べること)が必要不可欠である。しかしながら半導体製品は微細化・高性能化と共に大規模化・複雑化することでテストがますます困難、かつコストの増大が課題となっている。テスト品質は故障検出効率によって評価できる。これは仮定する故障のうち、被テスト回路 (CUT: circuit-under-test)に印加するテストパターンで検出できる故障と冗長故障(故障による誤りをLSIの外部出力に伝搬不可)の占める割合である。またテストコストは、テストパターンを生成する時間、CUTのテストをテストを用いて実行するテスト実行時間、またCUTをテストでどの程度一度にテストできるかで評価できる。

LSIのテスト品質の向上、およびテストコストの削減のために組込み自己テスト (BIST: Built-in Self-test) が広く利用されるようになってきた。故障検出効率の向上や、テストコスト削減のためにCUTを一部変更、もしくは回路機能を追加する手法をテスト容易化設計(DFT: Design for Testability)という。組込み自己テストは、テスト容易化設計手法の1つであり、テスト対象のCUTを含むLSIにテスト用回路であるBIST回路を組込み、自己テストを行わせることでテスタ利用時と同様のテスト品質の確保が期待できる。組込み自己テストでは一般に、テスタで最終テスト結果としてCUTの出力応答の圧縮結果であるシグネチャのみを比較すること
によるテストの簡略化やテストパターン生成が不要であることで、テストコストの削減が可能である。さらにLSI出荷後に組込まれたBIST回路によってフィールドテストを行うことが可能となる。

これまで、BISTに関連した数多くの研究では、BIST回路は製造テストでもフィールドテストにおいても故障がなく、正常に動作するとみなされてきた。しかしながら近年のCUT (circuit-under-test)の大規模化に対して、CUTに対するBIST回路の面積オーバヘッドはほぼ一定であり、BIST回路も大規模化している [1]。BIST回路の大規模化により、BIST回路中の故障発生率の増加が無視できなくなっている。文献 [2] ではBIST回路のテスト生成器への永久故障は、故障検出率の大幅な低下を招くことで、故障したCUTの誤りを見逃す可能性が指摘されている。また文献 [3] では、プロセスの立ち上げ時（低歩留まり時）や、CUTの故障検出率が98%以上と高いときに、BIST回路の故障の影響が大きくなることをモデルを用いて示している。

仮に故障したBIST回路でCUTのテストを実行した場合、大きく分けて2つの影響が考えられる。1つは正常なCUTを不良と判定すること、もう1つは不良LSIを正常と判定することである。前者は製造テストにおいて歩留まりの減少、フィールドテストにおいては不要な修復や交換が必要となり、コスト面への影響が考えられる。後者は製造テストにおいて市場不良の原因、フィールドテストにおいて障害を引き起こす原因となり、信頼性において影響が考えられる。さらにBIST回路が正常であることが、シリコンデバッグやウェハーレベルでのプロセス技術改善 [4]、FPGAを利用した機能テスト [5, 6] などのLSI初期製造時の診断において重要であることが示されている [7]。また近年は永久故障だけでなく、放射線衝突などが原因のソフトエラーなどの一時故障の発生確率の増加が指摘されている。文献 [8] では、製造プロセスの微細化が進むほどロジック部（論理ゲート）の一時故障発生回数が大きく増加することが示されている。組込み自己テストではLSI内部にBIST回路を組込むことで、LSI出荷後のフィールドテストに利用可能であるが、フィールドテストでは特にソフトエラーの一時故障によって、テスト実
行結果の判定を誤る可能性が考えられる。

組込み自己テストにおいて期待通りのテスト実行結果が得られることを保証するためには、BIST 回路が故障したとしても、その故障を BIST 回路自身の故障として正しく検出できる、または期待通りのテスト実行結果が得られるような機構が必要である。本論文では、このような BIST 回路を利用した組込み自己テストを高信頼組込み自己テストと定義し、高信頼組込み自己テストのための耐故障 BIST 回路を提案する。

これまで誤り検出可能なテスト生成器である EBIST が [2] が提案されている。EBIST ではテスト生成器に利用する LFSR (Linear Feedback Shift Register) を常に巡回符号の符号語をテストパターンとして生成するよう構成し、LFSR の生成するテストパターンが符号語か否かを判定する誤り検出器によって LFSR の故障の有無を判定する。誤り検出可能な BIST 回路による誤り検出の情報は CUT の故障と BIST 回路の故障とを区別できるため故障診断などに役立つが、BIST 回路故障時は別途 LSI テスタ利用などによって CUT のテストを実行する必要がある。

本論文の構成を図 1.1 に示す。本論文では EBIST の誤り検出機構を利用し、検出機構を利用した再構成や誤り訂正によって故障時であっても CUT のテストを期待通り実行可能な高信頼組込み自己テストのための耐故障 BIST 回路を提案する。耐故障テスト生成器の実現に対しては動的マスクと静的マスクの二通りのアプローチを行う。第 3 章では EBIST の誤り検出機構だけを利用した再構成可能テスト生成器を提案する。これは図 1.1 に示すとおり、動的マスクを実現する耐故障テスト生成器である。さらに第 4 章では EBIST で利用する誤り訂正可能な巡回符号の訂正能力を利用した誤り訂正可能テスト生成器を提案する。これは図 1.1 に示すとおり、静的マスクを実現する耐故障テスト生成器である。各アプローチでは救済実行方法の違いにより、テスト実行時間のオーバヘッドや再構成可能回数、誤り訂正可能回数などの救済可能回数が異なるため、それぞれ製造テスト、フィールドテストでの利用効果が高いことが想定される。一方応答圧縮器に対しては BIST 回路を利用した CUT のテスト実行時の応答圧縮器の誤り検出可能な手法は存在し
図 1.1: 本論文の構成

図 1.1: 本論文の構成

第 3 章では，テスト生成器を EBIST で構成した複数のユニットと，各ユニットの出力を切り替える切替器を組合せることでテスト生成器故障時も正常なテストバターンを生成可能な再構成可能な耐故障テスト生成器を提案する．提案するテスト生成器は 2 通りあるが，どちらも EBIST の誤り検出機構を利用することで各ユニットの故障を検出する．前者は再構成時に予め用意したスペアユニットの出力と置き換える待機冗長型テスト生成器である．後者は再構成時に故障したユニットと別の正常なユニットの出力を置き換える縮退利用型テスト生成器である．提案する 2 つのテスト生成器は故障箇所を再構成によって分離するため主に製造欠陥の検出を目的とした製造テストでの利用が想定される．しかしながら LFSR のみで構成した通常のテスト生成器と比較して，スペアユニットや切替器を付加し

以下，それぞれの章を簡単に説明する．
たことで面積オーバヘッドが増化し、再構成による救済効果以上に故障確率の増加と考えられる。そのため面積当たりの故障率を仮定し、無故障である、または再構成によって期待通りのテストパターンを生成可能な確率を信頼度として評価する。実験結果より、提案するテスト生成器は通常のテスト生成器よりも高い信頼度が得られることを示す。また提案するテスト生成器は複数のユニットで構成することから各ユニット内の LFSR のサイズが小さくなることで生成可能なテストパターンの種類の減少によって故障検出率の低下が考えられる。実験結果より、通常テスト生成器と同等の高い故障検出率の得られるユニット数の上限や、縮退利用型テスト生成器における縮退動作次での故障検出率の評価結果を示す。

第 4 章では、誤り訂正可能な巡回符号を用いた誤り訂正可能なテスト生成器を提案する。提案するテスト生成器は、誤り訂正可能な巡回符号を利用した EBIST から得られる誤り検出の情報を、誤り訂正に利用する。誤り訂正の回路はすべて組合せ回路で構成するため、LFSR 故障によるの内部状態の誤りを CUT へテストパター
ンとして出力する前に訂正が可能である。また訂正済みのテストパターンを LFSR
ヘ次状態として保存することで次のサイクルへ誤りが継続しないよう構成して
いる。再構成を利用したテスト生成器では救済回数に限りがあるが、誤り訂正可能
テスト生成器では訂正可能回数に上限はない。このためテスト生成器の動作中に
複数回の発生が考えられる一時故障に対する耐性が強いと考えられ、主にフィー
ルドテストでの利用が想定される。テスト生成器の誤り訂正能力が高いほどテス
ト生成器の故障時に無故障時と同様の期待動作可能なケースが多くなるが、訂正
回路の複雑化による面積増によって故障発生が増加し、期待通りのテストパター
ンを生成可能な確率が低下してしまうことが懸念される。テスト生成器の故障発
生時に期待通りのテストパターンを生成可能な確率を期待動作確率として定義し,
誤り訂正能力と期待動作確率の関係について実験を行った結果、1 ビット誤り訂
正可能なテスト生成器が最も期待動作確率が高くなるという結果が得られ、さら
に誤り訂正可能テスト生成器は EBIST や通常のテスト生成器より期待動作確率が
高いという結果が得られた。この傾向はメモリ部(FF)への一時故障発生率が高い
ほど顕著である。またCUTのテスト実行の最初から最後まで期待通りのテストパターンを生成可能な確率を期待テスト実行確率と定義し、EBISTとの比較実験を行った。誤り検出時にLFSRをリセットし、テストを最初から再実行するEBISTに対し、誤り訂正によって正しいテストパターンを生成し続ける誤り訂正可能テスト生成器の方が高い期待テスト実行確率が得られることを示す。

第5章では、巡回符号を用いた耐故障応答圧縮器を提案する。まずEBISTの符号を利用した誤り検出という考え方を応答圧縮器に適用することで誤り検出可能な応答圧縮器である符号化応答圧縮器を提案する。提案する応答圧縮器の評価尺度として応答圧縮器の故障を正しく検出可能である故障数の割合を自己判別率で定義することとする。実験結果では理論値と遜色ない自己判別率が得られることを考える。次に符号化応答圧縮器に誤り訂正回路を付加した誤り訂正可能応答圧縮器を提案する。誤り訂正可能応答圧縮器は誤り訂正可能な巡回符号を用いた符号化応答圧縮器で検出した誤りを訂正する機構を備えている。提案する応答圧縮器は、符号化応答圧縮器では応答圧縮器の故障と検出可能であった多くの故障に対して誤りの訂正が可能であるため、無故障時と同様の結果が得られる。製造テストを想定した評価では、符号化応答圧縮器に付加した誤り訂正回路の多くの永久故障も応答圧縮器の故障と判定可能であり、符号化応答圧縮器より高い自己判別率が得られることを示す。またフィールドテストを想定した評価ではシグネチャ観測時に期待通りのシグネチャを観測可能な確率を期待テスト実行確率と定義し、符号化応答圧縮器および一般的な応答圧縮器との比較実験を行った。実験結果より誤り訂正によって期待通りのシグネチャを生成可能な誤り訂正可能テスト生成器が最も高い期待テスト実行確率が得られることを示す。

第6章では本論文の結論と今後の課題について述べる。
第2章 論理回路のテストと高信頼組込み自己テスト

本章では論理回路のテストと高信頼組込み自己テストについて述べる。近年はVLSIのテストはシステムの信頼性を高めるために欠かせないが、VLSIの大規模化、複雑化によりテストコストの増加が課題となっている。組込み自己テスト（BIST）はテストを行う機構であるBIST回路を被テスト回路（CUT）に組込み、簡略化することでテストコストの削減ができるほか、LSI出荷後のフィールドテストにも利用できるという利点がある。しかしながら、CUTの大規模化とともにBIST回路もほぼ同じ比率で大規模化しており、故障発生の可能性を無視できなくなっていている。BIST回路の故障は、CUTのテスト実行結果に影響を与えることで、コスト面、信頼性の面の両方で不利益を被る。高信頼化設計には多重化、符号化、冗長回路付加などの様々な方法があるが、これらの高信頼化設計手法を取り入れることで、BISTによるテスト実行結果をより高信頼とする高信頼組込み自己テストという概念を提唱する。

2.1節では論理回路のテストに関する故障モデルおよびテストパターンについて述べる。2.2節では、製造されたVLSIが正常に動作するかを確認するためのテスト手法の1つである組込み自己テストについて述べる。2.3節では、高信頼設計手法について述べる。2.4節では、組込み自己テストのためのBIST回路を高信頼な設計とする高信頼組込み自己テストの考え方や、従来法として文献[2,7]の手法を交えて述べる。
2.1 論理回路のテスト

VLSIの巨大化、高集積化に伴い、VLSIに対するテストが重要となっている。本節では、VLSIに対するテストについて説明する。2.1.1節では、論理回路における故障モデルを説明し、2.1.2節では、論理回路の故障を検出するテストパターンについて説明する。

2.1.1 論理回路の故障モデル

回路を構成する要素に存在する物理的欠陥は、回路が誤動作する原因となり、期待通りの動作をしなくなる。このような物理的欠陥による正常回路と異なる振る舞いを故障といい、回路要素の出力に現れる故障の影響を誤りという。回路の誤りが外部出力などに伝搬し、提供するサービスに影響を与えることを障害という。VLSIの構成要素は論理回路（ディジタル回路）であるが、論理回路の論理機能が故障により別な論理機能に変化してしまう故障を論理故障（logical fault）という。

回路の物理的欠陥の要因は様々なあるため、通常は物理的欠陥による正常回路と異なる振る舞いを故障モデルとして表現する[9, 10]。論理故障として扱われる故障は縮退故障、ブリッジ故障、オープン故障など様々であり、また近年はVLSIの高速動作によって遅延故障も考慮される。中でも最もよく扱われる故障としては、信号線が論理値0に固定されてしまう縮退故障や、論理値1に固定されてしまう1縮退故障などの縮退故障があげられる。

例えば、図2.1のように2入力ORゲートと2入力NANDゲートを組み合わせた回路の入力線A, Bの論理値が0で、入力線Cの論理値が1のとき、正常な場合は出力線Zから論理値0が出力される。しかし、図2.1のようにORゲートの入力線Aに電源供給線と短絡という、物理的欠陥が原因による1縮退故障を仮定すると、出力値は誤りである1となる。

一般に、回路にn本の信号線が存在するならば、2n個の縮退故障が存在することになる。よって、VLSIの大規模化、複雑化に伴い信号線数が増加するので、想
定される故障の数も膨大になる．
また故障モデルはそのとらえ方によっていくつかに分類できる．まず回路への
故障を高々1つとする単一故障，回路に2つ以上の故障を同時に仮定する多重故障
があげられる．多重故障は故障箇所の組合せが爆発的に増加するが，一般には単
一故障よりも外部出力へ誤りを伝播しやすいとされている．次に故障発生後，再
現性のある永久故障と近年多くに課題となっている一時故障があげられる．一時
故障は宇宙放射線による信号線やメモリに保持された値の反転や，不安定な電源
電圧などによって誤った値が生成されるケースなどが考えられる．

2.1.2 テストパターン

製造された VLSI が正常に動作するかどうかを検査することをテストという．
VLSI の故障を検出するには，回路の出力応答を観測して期待値（正常値）と比較し，
相違を見つけることで行われる．そのためには故障箇所に対応し，回路の出力に
て期待値と異なる出力が得られる，テストパターンと呼ばれる入力信号パターン
が必要である．

図 2.2 に信号線 D 上の 0 縮退故障 (s-a-0) に対する例を示す．図 2.2(a) の例では，
入力信号パターン (A, B, C)=(0,1,1) を与えた結果，回路の出力にて期待値と異な
る出力 0 が得られている．そのため，入力信号パターン (A, B, C)=(0,1,1) は，信号
線 D 上の 0 縮退故障を検出するためのテストパターンといえる．一方，図 2.2(b)

欠陥

入力 A 0
入力 B 0
入力 C 1

VDD

故障

0/1

1/0

出力 Z

0/1

正常値/故障値

図 2.1: OR ゲートの 1 縮退故障とその伝播
の例では、入力信号パターン \((A, B, C) = (0,1,0)\) を与えた結果、回路の出力にて期待値と同じ出力 1 が得られている。そのため、入力信号パターン \((A, B, C) = (0,1,0)\) は、信号線 D 上の 0 緩退故障を検出するためのテストパターンとはいえない。

テストパターンに求められるのは故障の影響を外部出力に伝える事である。テストにおいて、故障を検出する能力はテスト品質として、表すことができる [10]。生成したテストパターンが、実際に発生した故障を検出可能ならば、テストの品質が高いと表すことができ、実際に発生した故障を見逃すなど検出不可能ならば、テストの品質が低いと表すことができる。

### 2.2 組込み自己テスト

VLSI のテストは VLSI の信頼性確保のために必須であるが、近年の VLSI は大規模化・複雑化する傾向が報告されている。LSI のテストコストは、テストパターン生成時間、LSI テスタの利用時間、LSI テスタと CUT の接続ピン数、LSI テスタで保持するテストに必要なデータ量などで決まるが、近年はテストパターン数の

![テストパターン図](image)

(a) 入力 \((A, B, C) = (0, 1, 1)\) の場合

(b) 入力 \((A, B, C) = (0, 1, 0)\) の場合

図 2.2: テストパターン
増大，CUT との接続ビン数の増加などが原因でテストコストの増大が課題となっている。文献 [1] では図 2.3 に示すように，将来にわたってテストデータ量が増化するという見込みが示されている。

本節では，テスト容易化設計の 1 つである組込み自己テスト（BIST）について説明する。一般には図 2.4 のように CUT と LSI テスタを接続し，印加したしたテストパターンに対する出力応答を LSI テスタで受取り，LSI テスタで保持する期待出力応答と比較し，一致すれば良品，不一致であれば不良と判定する。一方，組込み自己テストは図 2.5 のように，CUT を含む LSI と LSI テスタをわずかな制御信号線のみを接続する。LSI テストのためのテストパターンを生成するテスト生成器や出力応答を圧縮する応答圧縮器などからなる BIST 回路を LSI と同じチップ状に組込み，LSI テスタからの制御信号によって自己テストを行わせる。自己テストの結果，応答圧縮器から得られた出力応答の圧縮結果を LSI テスタで保持する期待圧縮結果と比較し，一致すれば良品，不一致であれば不良と判定する。

このような，組込み自己テストでは，BIST 回路でテストパターンを生成するためテストパターン生成時間が不要，LSI テスタと LSI の接続ビン数は制御信号のみで良いため大幅に減少，LSI テスタで保持するテストに必要なデータは期待値のみ

![図 2.3: テストデータ量の変化 [1]](chart.png)
で良いためごくわずかと，LSI テスタに必要な機能を大幅に削減することでテストコストの削減が可能となる。BIST 回路は一般にハードウェアで構成するため、テストパターンを格納するメモリと比較して面積が圧倒的に小さく、LSI の動作速度と同じ実動作でのテスト実行が可能であるというメリットの他に、ハードウェアで生成する疑似ランダムパターンを利用するため特定の故障モデルに限ったテストパターンでないということから未考慮の故障についての検出できる可能性がある。またテスト用の BIST 回路を LSI に組込むことから、LSI の製造テストだけでなく、LSI を出荷した後、市場にて行うフィールドテストにも適応可能であるという利点もある。

2.2.1 節では BIST 回路のうちテスト生成器について、2.2.2 節では応答圧縮器について

図 2.4: LSI テスタを利用したテストの概念図
図 2.5: 組込み自己テストの概念図
2.2.1 テスト生成器

テスト生成器は被テスト回路（CUT: circuit-under-test）に印加するテストパターンを生成する回路である。テスト生成器として考えられる手法の1つに、テストパターンをメモリに格納し、読み出すことでテスト生成器として機能させるメモリ読み出し方式がある。しかしながら、近年のLSIの大規模化・複雑化による総テストパターン量の増大により、すべてのテストパターンをメモリに格納し、チップ上にテスト生成器として組込むことはチップ面積の大幅な増大を招くほか、メモリからテストパターンを読み出すことで、テスト実行時間の増大が懸念される。そのため、高速にテストパターンを生成・印加可能なハードウェアで構成したテスト生成器が求められる。テスト生成器をハードウェアで実現する手法は様々で、

LFSR は図 2.6、2.7 に示すようにフリップフロップ (FF) に格納された値がクロックに同期して次の FF ヘシフト (移動) するシフトレジスタに XOR ゲートを用いたフィードバックをシフトレジスタの一部にかける構成をしており、あらかじめ設定されたフィードバック多項式 (LFSR のフィードバックを表す式) にしたがって、与えられた初期値 (シード) を展開することで疑似ランダムパターンを生成するテスト生成器である。疑似ランダムなパターンをハードウェアで生成することから、LFSR を用いることで、テストパターン生成を行う必要がなくなることも LFSR を用いる利点である。

図 2.6、2.7 に示す LFSR は、N ビットの出力を持つ LFSR を表しており、図 2.6 はガロア型の LFSR を、図 2.7 はフィボナッチ型の式の LFSR を表している [13]。2 種類の LFSR のフィードバック多項式 \( \phi(x) \) は以下のように表すことができる。

\[
\phi(x) = 1 + a_1 x^1 + a_2 x^2 + \ldots + a_{N-1} x^{N-1} + x^N \quad (a_k \in 0, 1) \quad (2.1)
\]

フィードバック多項式 \( \phi(x) \) の次数は LFSR の大きさ (与えるシードのビット数) を表し、その他の項は \( a_k = 1 \) (1 ≤ k ≤ N) ならばフィードバックがかかることを表し、\( a_k = 0 \) (1 ≤ k ≤ N) ならばフィードバックをかけないことを表している。

図 2.6、2.7 に示した LFSR ではクロック信号に同期して \( FF_i \) の値は右隣の \( FF_{i+1} \) に伝えられる。図 2.6 に示すガロア型 LFSR では、\( FF_1 \) には、\( FF_N \) の値が入力される。また \( a_{k-1} = 1 \) (2 ≤ k ≤ N) であれば、\( FF_k \) には \( FF_{k-1} \) と \( FF_N \) の値の排他的論理和が入力され、\( a_{k-1} = 0 \) (2 ≤ k ≤ N) であれば、\( FF_k \) には \( FF_{k-1} \) の値が入力され、それぞれの \( FF_i \) の値が出力値 \( output_i \) として出力される。一方、図 2.7 に示すフィボナッチ型 LFSR では \( FF_1 \) には、\( a_{k-1} = 1 \) (2 ≤ k ≤ N) である全ての \( FF_k \) の排他的論理和、その他の \( FF_k \) では \( FF_{k-1} \) の値が入力され、それぞれの \( FF_i \) の値が出力値 \( output_i \) として出力される。
ガロア式の LFSR が生成するパターンは、LFSR 中の FF の内部状態を多項式、
$S_k(x)$ で表すことで、LFSR のフィードバック多項式、$\phi(x)$ を用いて表すことができ
る [2, 14]。図 2.6, 2.7 に示す N ビット LFSR の FF,$FF_1$ から $FF_N$ の各内部状
態を $S_{k,1}$, $S_{k,2}$, ..., $S_{k,N-1}$, $S_{k,N}$ とすると、多項式 $S_k(x)$ は以下のように表せる。
なお $k$ は LFSR の動作開始後経過したクロックサイクル数を表すものとする。

$$S_k(x) = S_{k,1} + S_{k,2}x + \ldots + S_{k,N-1}x^{N-2} + S_{k,N}x^{N-1}$$  (2.2)

LFSR の FF にシードが格納されている初期状態とし、FF の内部状態を $S_0(x)$ とす
ると、LFSR の動作開始後 $k$ サイクル後の出力はフィードバック多項式 $\phi(x)$ と FF
の内部状態、$S_k(x)$ を用いて以下のように表すことができる。

$$S_k(x) = x^kS_0(x) \mod \phi(x)$$  (2.3)

一方、フィボナッチ型の LFSR が生成するパターンもガロア型の LFSR と同じ
く、LFSR 中の FF の内部状態を式 2.2 のように表すことができる。このフィボナッ
チ型 LFSR では $k$ サイクル後の各 FF の内部状態は以下の式で表すことができる。
なお、$\oplus$ は排他的論理和 (XOR) の演算を表すものとする。

$$S_{k,0} = a_{N-1}S_{k-1,2} \oplus a_{N-2}S_{k-1,3} \oplus \ldots \oplus a_3S_{k-1,N-2} \oplus a_2S_{k-1,N-1}$$

$$S_{k,i} = S_{k,i-1} \quad (2 \leq i \leq N)$$  (2.4)

LFSR は疑似ランダムパターンをハードウェアで生成することにより、ROM 等
のメモリに予めテストパターンを格納し、読み出す手法と比較して読み出し時間
の短縮によるテスト時間の高速化、テストパターンを格納しておくために必要な
メモリを節約可能等の利点があるが、生成されるテストパターンが乱数パターン
であることから常に多くの故障を検出できるとは限らず、また全て 0 となるパターン
は生成できない、他に LFSR の初期値 (シード) として全て 0 のパターンを用い
た場合は疑似ランダムなパターンを生成できない等の欠点がある。
図 2.9: 順序回路に対する BIST (STUMPS)
またテスト対象の回路によっては、疑似ランダムなテストパターンでは検出が困難な故障が存在する。そのため、検出困難な故障に対して ATPG にてテストパターンを生成し、BIST によるテストと外部テストを利用したテストを併用する手法 [10] や、検出困難な故障に対して有効なテストパターンを生成可能なシードを用意し、テスト時にシードを変えて再度 LFSR を利用した BIST によるテストを実行するリシーディング [15]、LFSR の出力ビットの XOR 演算を行う XOR ネットワークで構成した Phase Shifter [16, 17] を付加する手法などがある。

一般に BIST 回路は CUT が組合せ回路であっても順序回路であっても適用可能である。図 2.8, 2.9 にそれぞれ組合せ回路への BIST 回路を適用例、順序回路への BIST 回路を適用例を示す。図 2.8 では 1 パターンごとに CUT のテストを実行するが、図 2.9 のように複数の LFSR の出力を複数のスキャンチェーンを通して CUT に印加し、テストを実行する手法を特に STUMPS [18] という。

2.2.2 応答圧縮器

応答圧縮器は CUT の出力応答が期待値と一致・不一致かを調べ、LSI のテスト結果の合否を判定を簡略化するために CUT の出力応答を圧縮する回路である。近年の LSI の大規模化・複雑化による総テストパターン量の増大により、すべてのテストパターンに対する出力応答の期待値をメモリに格納し、チップ上に応答解析器として組込むことはチップ面積の大幅な増大を招くほか、メモリからテストパターンを読み出すことで、テスト実行時間の増大が懸念される。そのため BIST では図 2.5 に示すように、出力応答を圧縮する応答圧縮器を利用し、応答圧縮器から得られる圧縮結果をテスト結果の判定に利用することで、LSI テスタの機能を簡素化する手法が一般的である。

一般に圧縮手法としては XOR ベートなどの論理ゲートを使用して出力ビット幅を圧縮する空間圧縮器 (Space Compactor) が利用される。しかしながら BIST において出力応答を圧縮する手法としては、多入力シフトレジスタ (MISR, Multiple Input Signature Register) が一般に用いられる。
MISR は出力応答を時間軸方向に圧縮し，CUT のテスト終了後に MISR 内の FF の内部状態を圧縮結果（シグネチャ，Signature）として，LSI テスタ中の ROM に格納している。予め求めれば正常回路の期待値と比較し，一致すれば LSI は良品，不一致ならば不良と判定する。この MISR を用いたシグネチャによるテスト結果の判定をシグネチャ解析法（Signature Analyzer）という [9, 10]．

図 2.10: MISR の概念図

図 2.10 は，N ビットの入力を持つ MISR を表しており，この MISR のフィードバック多項式，\( \phi(x) \) は以下のよう表現することができる。

\[
\phi(x) = 1 + a_1 x^1 + a_2 x^2 + \ldots + a_{N-1} x^{N-1} + x^N \quad (a_k \in 0, 1)
\]  

(2.5)  

フィードバック多項式 \( \phi(x) \) の次数は，MISR の大きさ（与えるシードのビット数）を表し，その他の項は，\( a_k = 1 \)（\( 1 \leq N \)）ならばフィードバックがかかったことを表し，\( a_k = 0 \)（\( 1 \leq N \)）ならばフィードバックをかけないことを表している。

図 2.10 に示した MISR では，クロック信号に同期して FFi の値は，右隣の FFi+1 に伝えられる。この MISR では，FFi に FFN の値と入力値 input1 の排他論理和が入力される。また \( a_k = 1 \)（\( 2 \leq k \leq N \)）であれば，FFk には，FFk−1，FFN と inputk の値の排他論理和が入力され，\( a_k = 0 \)（\( 2 \leq k \leq N \)）であれば，FFk には，FFk−1 と inputk の値の排他論理和が入力され，それぞれの FFi
の値が出力値 \( output_i \) として出力される．

図 2.10 に示す \( N \) ビット MISR の FF, \( FF_1 \) から \( FF_N \) の内部状態を \( S_{k, 1}, S_{k, 1}, \ldots, S_{k, N-1}, S_{k, N} \) とすると多項式 \( S_k(x) \) は以下のように表すことができる．なお，\( k \) はMISR の動作開始後経過したクロックサイクル数を表すものとする．

\[
S_k(x) = S_{k, 1} + S_{k, 2}x + \ldots + S_{k, N-1}x^{N-2} + S_{k, N}x^{N-1} \tag{2.6}
\]

また，MISR への \( N \) ビットの入力，\( input_1, input_2, \ldots, input_{N-1}, input_N \) をそれぞれ，\( I_{k, 1}, I_{k, 2}, \ldots, I_{k, N-1}, I_{k, N} \) で表すとすると，以下の多項式で表すことができる．

\[
I_k(x) = I_{k, 1} + I_{k, 2}x + \ldots + I_{k, N-1}x^{N-2} + I_{k, N}x^{N-1} \tag{2.7}
\]

LFSR の FF にシードが格納されている初期状態とし，FF の内部状態を \( S_0(x) \) とすると，LFSR の動作開始後 \( k \) サイクル後の出力はフィードバック多項式 \( \phi(x) \) と FF の内部状態 \( S_k(x) \) である MISR の出力である CUT の出力応答 \( I_k(x) \) を用いて以下のように表すことができる．

\[
S_k(x) = \{ x^{k-1}S_0(x) + x^{k-1}I_0(x) + x^{k-2}I_1(x) + \ldots + xI_{k-2}(x) + I_{k-1}(x) \} \mod \phi(x) \tag{2.8}
\]

なお式 2.8 を簡略化した式は以下のように表せる．

\[
S_k(x) = \{ x S_{k-1}(x) + I_k(x) \} \mod \phi(x) \tag{2.9}
\]

MISR は CUT の出力応答を圧縮するため，出力系列中の誤りがマスクされ，正常回路と故障回路のシグネチャが一致する，誤り見逃し（Aliasing）を生じる可能性がある．MISR では，出力系列からシグネチャがほぼランダムに得られることから，シグネチャ解析法の誤り見逃し率は非常に小さくなる．一般に，\( N \) ビット MISR を用いると，誤り見逃しの頻度は \( 2^N \) 分の 1 以下となるため，ほぼ無視できるくらい小さくなると考えられる [10, 14]．

応答圧縮器として利用される MISR や一般的なビット幅圧縮器である空間圧縮器では XOR ゲートが多く利用される．XOR ゲートはその性質上，入力値が 1 ま
たは0の固定値とする必要があり、不定値（ドントケアビット）が入力に含まれている場合は出力値を決定できない。不定値の発生する原因としては、メモリ、アナログ回路からの出力、FFの未初期化などが原因としてあげられる。応答圧縮器でのシグナル計算においてシグナルを正しく計算するためにはCUTの出力応答に不定値を含まないようマスクする必要がある。

2.3 高信頼設計

LSIに故障する故障は正常状態と異なる誤りを発生させ、誤りが外部へと伝搬することを提供するサービスに影響を及ぼす障害となる。高信頼なLSIを設計するには故障しないシステムを目指すフォールトアボイダンスという考え方があるが、故障の発生を防ぐことは難しい、フォールトアボイダンスだけでは信頼性の高いシステムを構築することは不可能である。LSIを高信頼とするためには、故障の発生を前提とし、故障による誤りが原因の障害を発生しないようにシステムを構成するフォールトトレランスという考え方が一般的である。このような考えに基づいたシステムをフォールトレラントシステムと呼ぶ[10]。近年はディベンダビリティというフォールトレランスとほぼ同義の言葉も一般的である。フォールトレランスは故障に耐えてサービスを続けることのあるのに対し、ディベンダビリティは信頼できるサービスを提供することという前向きな考え方であるという違いがある。しかしながら完全なフォールトレランスは実在しないため、対象とする故障を故障モデルとして明確に定めることが重要である。フォールトレランスの実現するにはサービスを利用するユーザからの障害の見え方に応じて、静的マスク、動的マスク、フェイルセーブの3つのレベルに分類できる。2.3.1節では静的マスクについて、2.3.2節では動的マスクについて、2.3.3節では静的マスクの一例である符号化を用いた誤り訂正について述べる。
2.3.1 静的マスク

静的マスクは障害を一切生じさせないというレベルであり、ユーザはサービス利用時に発生した故障や故障による誤りの発生に気づくことはない [10]。一例として図 2.11 に示す 3 重化多数決システムがある。このシステムはシステムを 3 重化し、同一のサービスを提供する各システムの出力多数決を取ることで故障は高々 1 つのユニットでのみ発生するという単一ユニット故障が発生しても、故障による誤りは多数決によって訂正されるためサービスへの障害は一切生じない。静的マスクは障害を生じさせないためリアルタイムな処理が必要となるため、演算を再計算するリトライが許されないシステムでは静的マスクを利用したフォールトトレランスが必要となることが多い。図 2.11 に示す 3 重化多数決システムでは 2 つのユニットの組合せごとに比較器によってシステムの出力比較を行うことで故障の発生したシステムを特定することが可能である。例えばユニット 1 の故障時は比較器 1 と比較器 2 から誤りが検出されたことを表す信号が出力され、比較器 3 か
らは誤りを検出していないことを表す信号が出力される。このようにして故障が発生したユニットを特定することで修復も容易となる。この多数決システムでは3重化したユニットの他に、3つの比較器、1つの多数決回路が必要となるなど、静的マスクを実現するためのコストは他のフォールトトレランスのレベルと比較して最も大きくなる。

2.3.2 動的マスク

動的マスクは一時的に障害を引き起こすが、自動的に回復するフォールトトレランスのレベルである [10]。例えば、図2.12に示すシステムでは単一ユニット故障に対してスペアユニットに切り替える機構を備えている。検出器1によってユニットの故障を検出した場合、出力切替器にスペアユニット1の出力に切替るよう信号が送られる。また検出器で誤りが検出されない場合は定期的に正常状態保持器にユニットの演算状態を保存する。正常状態保持器では検出器1からの検出信号を受信すると、スペアユニットに正常時のユニットの演算状態を復元し、スペアユニット1で演算を再開する。このように誤りを検出する際に一時的に出力に誤りが伝搬し、障害となってユーザに検知されるが、スペアユニットが演算を引き続きでサービスを再開することができる。動的マスクではこのような障害が生じる時間が比較的短い（アプリケーションにもよるが数分以内）ことが重要である。このように演算のリトライが許されるようなシステムで動的マスクは利用されることが多い。

図2.12に示すシステムは予め用意したスペアユニットへ切替することで回復するため待機冗長システムと呼ばれる。ほかにもユニットの故障検出時にほかの演算を実行しているユニットへ演算を引き継ぎ、システムを部分的に縮退させて回復する縮退利用型のシステムも存在する。

24
図 2.12: スペアユニットを持つ動的マスク可能なシステム

2.3.3 符号化を利用した誤り訂正

静的マスクの一例である符号化を用いた誤り訂正について述べる [21, 22]. 符号化とは情報を一定の規則に従ったデータに変換することである. 一定の規則に従ったデータを符号語と呼び, 異なる符号語以外のデータを非符号語と呼ぶ. 情報を符号化することで, 送信器の故障などで圧力に送信する情報に誤りが発生した場合でも, 情報の誤りを検知できる誤り検出, または情報の誤りを正しい情報へ構成できる誤り訂正が可能である. 一般に情報に付加する冗長ビット数が多いほど, 誤り検出・訂正可能なビット数が増加するが, データに対して情報の占める割合が減少する. また冗長ビットが多いほど, 符号化や誤り検出・訂正を行うソフトウェアやハードウェアは複雑化する.
ここで符号化手法の１つである \((n, k)\) 巡回符号を利用した誤り訂正について説明する。\((n, k)\) 巡回符号は線形符号の一種であり、\((n, k)\) 巡回符号の符号語を巡回シフトしたものまた符号語となる特徴を持つ。\((n, k)\) 巡回符号の符号語は \(n\) ビットの符号長であり、そのうち \(k\) ビットが情報ブロック、\(n - k\) ビットが誤り検出・訂正に使われる冗長ブロックとなっている。すべての符号語は、\(n - 1\) 次の多項式で表すと \(n - k\) 巡回符号の生成多項式で割り切ることができるので \((n, k)\) 巡回符号の生成多項式は以下の条件式を満たす \(n - k\) 次の多項式である [21, 22]。\(^{(2.10)}\)

\[
(x^n - 1) \mod g(x) = 0
\]

例として、\(n = 7\)、\(k = 4\) の１ビット誤り訂正が可能な \((7, 4)\) 巡回符号を利用した誤り訂正について述べる。この巡回符号の生成多項式は式 (2.10) より式 (2.11) から式 (2.12) で表される。

\[
x^7 - 1 = (1 + x + x^2 + x^4)(1 + x + x^3)
\]

\[
g(x) = 1 + x + x^3
\]

\(^{(2.11)}\)

\(^{(2.12)}\)

0, 1 で表される 7 ビットのデータ “0100011”は多項式表現によって \(x + x^5 + x^6\) と表すことができる。ここで 7 ビットの 0100011 は、式 (2.13) のように式 (2.12) で割り切ることができるため、(7, 4) 巡回符号の符号語である。

\[
0100011: (x + x^5 + x^6) \mod (1 + x + x^3) = 0
\]

\(^{(2.13)}\)

一方、7 ビットのデータ “0101111”は、式 (2.14) のように式 (2.12) で割り切ることができないため、(7, 4) 巡回符号の符号語でなく、非符号語である。

\[
0101111: (x + x^3 + x^4 + x^5 + x^6) \mod (1 + x + x^3) = 1 + x^2
\]

\(^{(2.14)}\)

式 (2.13)、式 (2.14) のように巡回符号の生成多項式である式 (2.12) での除算は図 2.13 に示すフィードバック多項式が式 (2.12) であるガロア型 LFSR で求めることができ、除算による余りはシンドロームと呼び、誤り訂正可能な巡回符号ではこの
図 2.13: (7,4) 循回符号の誤り検出器 (LFSR)

表 2.1: (7,4) 循回符号のシンドロームと誤りビット

<table>
<thead>
<tr>
<th>シンドローム</th>
<th>多項式表現</th>
<th>誤りビット</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1</td>
<td>1ビット目</td>
</tr>
<tr>
<td>010</td>
<td>x</td>
<td>2ビット目</td>
</tr>
<tr>
<td>001</td>
<td>x²</td>
<td>3ビット目</td>
</tr>
<tr>
<td>110</td>
<td>1 + x</td>
<td>4ビット目</td>
</tr>
<tr>
<td>011</td>
<td>x + x²</td>
<td>5ビット目</td>
</tr>
<tr>
<td>111</td>
<td>1 + x + x²</td>
<td>6ビット目</td>
</tr>
<tr>
<td>101</td>
<td>1 + x²</td>
<td>7ビット目</td>
</tr>
</tbody>
</table>

表 2.2: 図 2.13 の誤り検出器によるシンドロームの算出

<table>
<thead>
<tr>
<th>サイクル</th>
<th>scan-in</th>
<th>scan</th>
<th>各 FF の状態</th>
<th>scan-out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1 1 0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1 1 1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0 0 1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1 1 1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1 0 1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1 0 1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0 1 0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0 0 1</td>
<td>1</td>
</tr>
</tbody>
</table>

シンドロームを元に訂正の必要なビットを選択し、訂正を行う．表 2.1 に (7,4) 循回符号におけるシンドロームと誤りビットの関係を示す．表 2.1 は順にシンドローム、シンドロームの多項式表現、誤りビットを表している．

表 2.2 にデータ “0101111” を図 2.13 に示す誤り検出器でシンドロームを求めた際の動作を示す．表 2.2 にサイクル、図 2.13 の scan-in の値、図 2.13 の scan の値、図 2.13 の各 FF の状態、図 2.13 の scan-out をそれぞれ示す．誤り検出器中の LFSR
の FF をすべて 0 に初期化した後，サイクル 1 で scan=1 として scan-in からデータ“0101111”を 1 ビットずつ入力する．入力されたデータは LFSR で生成多項式による除算が実行され，7 サイクル後のサイクル 8 で FF にシンドロームが保持される．このときサイクル 8 からは再び scan=0 として scan-out からシンドロームを出力する．表 2.1 よりシンドローム“101”は 7 ビット目の誤りであることが分かる．つまりデータ“0101111”の 7 ビット目を 1 から 0 へと訂正することで符号語“0101110”へと訂正することができる．

このように符号化を利用した誤り訂正によってデータの訂正することで故障による誤りを訂正することができる．図 2.2 の例では順序回路である図 2.13 に示す除算器でシンドロームを求めたが，組合せ回路で求める方法も存在する．

2.4 高信頼組込み自己テスト

被テスト LSI(CUT) の大規模化・複雑化に伴って増大するテストコストに対応するため BIST が広く利用されている．しかしながら CUT の大規模化とともに BIST 回路も大規模化し，故障発生の可能性が無視できなくなっている．BIST 回路の故障の結果，CUT のテスト実行結果を誤ることで，コスト面，信頼性の面の両方で不利益を被る．本節では，提唱する BIST によるテスト実行結果をより高信頼とする高信頼組込み自己テストについて述べる．

2.4.1 BIST 回路故障の影響

近年，大規模化，複雑化が進む LSI のテスト品質低下や増加するテストコストに対応するために，組込み自己テスト (BIST) は製造テストだけでなくフィールドテストでも利用されている．これまで，BIST に関連した数多くの研究では，BIST 回路は製造テストでもフィールドテストにおいても故障がなく，正常に動作するとみなされてきた．しかしながら近年の CUT の大規模化に伴って BIST 回路自身も大規模化，複雑化する傾向がある．文献 [1] では，図 2.14 に示すように SoC の
図 2.14: LSI のトランジスタ数と BIST 回路の面積オーバヘッド


BIST 回路が故障すると，正常な CUT が不良品と判定される可能性，故障した CUT の誤り見逃しが起こる可能性が考えられる [2, 3, 4, 7, 20]．これらは製造テストにおいて，前者は歩留まり低下の原因となるため製造コストの面で，後者は市場不良の原因となるため LSI の信頼性の面で不利を被る．一方，フィールドテストにおいては，前者は不要な障害対策によるコストの増加につながり，後者は障害を検出するためにシステムの信頼性に重大な影響を与える [2]．さらに BIST 回路が正常であることが，シリコンデバッグやウェハーレベルでのプロセス技術改善 [4]，FPGA を利用した機能テスト [5, 6] などの LSI 初期製造時の診断において重要であることが示されている [7]．近年は永久故障だけでなく，放射線衝突などが原因の
図 2.15: 微細化に伴う一時故障発生回数の変化
ソフトエラーなどの一時故障の発生確率の増加が指摘されている。図 2.15 に製造プロセスと 10 億時間当たりのソフトエラー発生回数の関係を示す。文献 [8] では、図 2.15 に示すように、製造プロセスの微細化が進むほどロジック部（論理ゲート）の一時故障発生回数が大きく増加することが示されている。メモリ部（FF）のソフトエラー発生回数は製造プロセスにかかわらず多いが、微細化が進むことにロジック部の発生回数がメモリ部に匹敵するほど増加していることが分かる。BIST では LSI 内部に BIST 回路を組込むことで、LSI 出荷後のフィールドテストに利用可能であるが、フィールドテストでは特にソフトエラーなどの一時故障によって、テスト実行結果の判定を誤る可能性が考えられる。そのため、BIST 回路が故障したとしても、故障を検出できるまたは期待通りのテスト実行結果が得られるような機構が必要である。
2.4.2 高信頼組込み自己テスト

BIST 回路の大規模化により、被テスト回路 (CUT) と同様に無故障であるという仮定は非現実的であり、BIST 回路に対しても故障が発生することを前提として考慮すべきである [2, 3, 7]。高信頼組込み自己テストとは、このような前提に基づき、BIST 回路に発生した故障への対応策を取り入れた BIST 回路を利用した組込み自己テストである。具体的には、BIST 回路故障による誤りを BIST 回路の故障として正しく検出可能である、BIST 回路故障時には CUT が無故障時と同様に期待通りテスト実行可能な耐故障 BIST 回路などがあげられる。高信頼組込み自己テストを実現するためには、BIST 回路をテストする手法が必要である。BIST 回路のテストは大きく分けて 2 つの手法があり、一方は CUT のテスト用のテストパターンと異なる BIST 回路専用のテストパターンで BIST 回路単体でテストを実行する手法、もう一方は CUT のテストパターンや出力応答を BIST 回路のテストパターンとして BIST 回路のテストを実行する手法である。前者は、BIST 回路のテスト実行の後に CUT のテストを実行することが一般的であるため BIST 回路のプレテスト、後者は BIST 回路を利用した CUT のテスト実行時に、BIST 回路自体もテストするため BIST 回路の同時テストと呼ぶ。BIST 回路の同時テストは、CUT と BIST 回路を別々にテストする BIST 回路のプレテストと比較して、BIST 回路単体のテストが不要なためテスト実行時間が短いという利点がある。また CUT のテスト実行前は正常であっても、テスト実行中に BIST 回路に故障が発生する場合も考えられる。BIST 回路の同時テストではこのような故障も検出可能である。この利点はフィールドテストで BIST 手法を用いる場合に特に重要である。これは市場では回路にソフトエラー (α 線などの影響で一時的に回路が誤動作すること [8]) が起こる可能性が高いためである。

本研究で対象とする BIST 回路の例を図 2.16 に示す。BIST 回路を利用した CUT のテスト実行時は、テスト生成器 (LFSR) で生成した系列が CUT のテストパターンとして印加され、CUT から得られた出力応答は応答圧縮器 (MISR) にて圧縮さ
图 2.16: BIST 回路の例

LFSR は FF の初期化のために MUX, AND デート, NOT デートによるスキャン設計を施している。LFSR の初期化時は, \( scan = 1 \) とし, \( scan - in \) から初期値を FF に入力する。また MISR の入力部およびフィードバックループ上には MISR の制御のため AND デートを配置している。MISR の初期化時および SO から内部状態（シグネチャ）を出力する際は \( c1 = 0 \) とし, CUT のテスト実行時は \( c1 = 1 \) とする。

出力されたシグネータは LSI デスタに読み込まれ、あらかじめ求めていたシグネチャの期待値と比較され、一致すれば CUT は無故障、不一致であれば CUT は故障していると判定される。応答圧縮器では、CUT 故障時に得られたシグネータが期待値と一致しない場合があり、これをエイリアシングと呼ぶ。

なお論文中の議論ではエイリアシングが発生しないと仮定する。

高信頼組込み自己テストのためのテスト容易な BIST 回路としてこれまでプレテスト可能な応答圧縮器である MISR-NOT [7] や同時テスト可能なテスト生成器である EBIST [2] が提案されている。2.4.3 節ではプレテスト可能な応答圧縮器である MISR-NOT について、2.4.4 節では同時テスト可能なテスト生成器である EBIST
についてそれぞれ述べる。

2.4.3 テスト容易な応答圧縮器 [7]

文献 [7] では、プレテストによってテスト容易な応答圧縮器である MISR-NOT が提案されている。提案されている MISR は、図 2.17 に示すように、MISR のフィードバックループに NOT ゲートとフィードバック部とシフトリジスタ部の接続を制御する NAND ゲートを挿入した形をしている。またシグネチャを出力するための SO を持つ。図 2.17 に示す N ビット MISR-NOT のフィードバック多項式は以下のように表される。

$$\phi(x) = 1 + a_1x^1 + a_2x^2 + \ldots + a_{N-1}x^{N-1} + x^N (a_k \in 0, 1) \quad (2.15)$$

MISR-NOT をテストする手順は初期化、テスト、出力の 3 つの段階に分けることができる。まず初期化のため、図 2.17 に示す信号 $C_1 = 0$ とし、シフトリジスタ部とフィードバックを切断する。そして、N ビット MISR の場合、N クロックの間 MISR-NOT へ定数ベクトル $W$ を CUT から入力し続ける。次にテストのため、$C_1 = 1$ とし、シフトリジスタ部とフィードバックを再度接続する。そして、また初期化の際に用いたパターンと定数ベクトル $W$ を CUT から入力し続ける。この入力はテスト実行中の $t$ サイクルの間、続ける。最後に、再び信号 $C_1 = 0$ とし、SO から FF の内部の値を出力する。なお、テストの間、CUT からは定数ベクトル $W$ を出力し続けるようホールドする必要があり、また MISR-NOT はフィードバック多項式を $N$ 次の原始多項式とする。

図 2.18 に 4 ビット MISR-NOT を示す。図 2.18 の MISR-NOT はフィードバック多項式として以下に示す 4 次の原始多項式としている。

$$\phi(x) = 1 + x^3 + x^4 \quad (2.16)$$

表 2.3 に、定数ベクトル $W$ と、初期化後に SO から得られる出力を 20 サイクル分示す。MISR-NOT では、表 2.3 のように、SO から出力系列は PE と PO で表され
る，わずか2種類のみとなる。これは、入力する定数ベクトル\( W \)が偶数バリティであるか、奇数バリティであるかどうかに依存する。また、この出力系列は、\( N \)ビットMISR-NOTの場合、\( 2^N - 1 \)ビットごとに全く同じパターンが現れる。

MISR-NOTでは、入力が常に同じパターンであれば、出力系列が、PE、POのどちらかになる。そのため、出力系列が、PE、またはPOであればMISR-NOTは故障無しと判定できる。この際、期待値と異なるパターンであればこの時点で、CUTの故障と判定し、CUTのテストを行わない。

MISR-NOTは、応答圧縮器として用いられるMISRをフィードバック部、シフトレジスタ部を含めて完全にテストを行えるが、CUTが定数ベクトルを出力するよ

![図2.17: MISR-NOTの概念図](image1)

![図2.18: 4ビットのMISR-NOT](image2)
表2.3: 図2.18のMISR-NOTのSOから得られる出力

<table>
<thead>
<tr>
<th>定数ベクトルW</th>
<th>出力</th>
<th>初期化後のSOからの出力</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>偶数バリティ</td>
<td>PE 11110101100100011110</td>
</tr>
<tr>
<td>0001</td>
<td>奇数バリティ</td>
<td>PO 01010011011100001010</td>
</tr>
<tr>
<td>0010</td>
<td>奇数バリティ</td>
<td>PO 01010011011100001010</td>
</tr>
<tr>
<td>0011</td>
<td>偶数バリティ</td>
<td>PE 11110101100100011110</td>
</tr>
<tr>
<td>0100</td>
<td>奇数バリティ</td>
<td>PO 01010011011100001010</td>
</tr>
<tr>
<td>0101</td>
<td>偶数バリティ</td>
<td>PE 11110101100100011110</td>
</tr>
<tr>
<td>0110</td>
<td>偶数バリティ</td>
<td>PE 11110101100100011110</td>
</tr>
<tr>
<td>0111</td>
<td>奇数バリティ</td>
<td>PO 01010011011100001010</td>
</tr>
<tr>
<td>1000</td>
<td>奇数バリティ</td>
<td>PO 01010011011100001010</td>
</tr>
<tr>
<td>1001</td>
<td>偶数バリティ</td>
<td>PE 11110101100100011110</td>
</tr>
<tr>
<td>1010</td>
<td>偶数バリティ</td>
<td>PE 11110101100100011110</td>
</tr>
<tr>
<td>1011</td>
<td>奇数バリティ</td>
<td>PO 01010011011100001010</td>
</tr>
<tr>
<td>1100</td>
<td>偶数バリティ</td>
<td>PE 11110101100100011110</td>
</tr>
<tr>
<td>1101</td>
<td>奇数バリティ</td>
<td>PO 01010011011100001010</td>
</tr>
<tr>
<td>1110</td>
<td>奇数バリティ</td>
<td>PO 01010011011100001010</td>
</tr>
<tr>
<td>1111</td>
<td>偶数バリティ</td>
<td>PE 11110101100100011110</td>
</tr>
</tbody>
</table>

うに動作をホールド状態にするだけでよいため、CUTの設計に影響を与えない点が利点としてあげられる。一方で、この手法はプレテストであり、CUTのテストを行う前にMISR-NOTをテストするため、MISR-NOTとCUTを同時にテストすることはできない。そのためMISR-NOTを利用したCUTのテスト時には、MISR-NOTが故障すると、信頼できるテスト結果が得られない可能性がある。

また、LSI出荷後にプレストレステストでは、必ずしもプレテストが行えない可能性が考えられる。また今後、LSIの製造技術のさらなる微細化により、放射線衝突などが原因となる一時的に誤った値がFFに取り込まれるソフトエラー発生率増加が考えられることから[8]、プレテストによるMISRのテスト結果が必ずしもCUTのテスト中も信頼できることは限らない。
2.4.4 誤り検出可能なテスト生成器[2]

文献[2]では同時テスト可能なテスト生成器である誤り検出可能なテスト生成器(EBIST)が提案されている。EBISTは\( n \)ビットの出力を持つテスト生成器を、\((n, k)\)巡回符号の符号語を生成するLFSRと誤り検出器を組み合わせて構成している。EBISTでは、LFSRの生成するテストパターンが常に符号語となるように構成し、誤り検出器ではLFSRの出力が符号語か非符号語かを判定し、非符号語が出力された場合はLFSRの故障と判定する。

\((n, k)\)巡回符号を用いたEBISTを構成するにはまず生成多項式を求める必要がある。\((n, k)\)巡回符号の生成多項式は以下の条件式を満たす\( n - k \)次の多項式である[21,22]。

\[
(x^n - 1) \mod g(x) = 0 \tag{2.17}
\]

この式(2.17)を満たす\( g(x) \)を用いて、\( n \)ビットの出力を持つEBIST中の\((n, k)\)巡回符号を生成するLFSRのフィードバック多項式\( \phi(x) \)は以下のように表される。

\[
\phi(x) = g(x)p(x) \tag{2.18}
\]

式(2.18)のうち、\( g(x) \)は\((n, k)\)巡回符号の生成多項式を表し、\( p(x) \)は\( k \)次の多項式を表している。

ここで例として\((7,4)\)巡回符号を用いたEBISTについて考える。まずはじめに、
\((7,4)\)巡回符号を生成するLFSRを構成する。

\[
(x^7 - 1) = (1 + x + x^3)(1 + x^2 + x^3)(1 + x) \tag{2.19}
\]

式(2.19)より、\( 7 - 4 = 3 \)次の生成多項式は、式(2.20)のように決めることができる。\( \phi(x) \)を構成するための\( 4 \)次の多項式、\( p(x) \)は以下のように\( 4 \)次の原始多項式とする。

\[
g(x) = 1 + x + x^3 \tag{2.20}
\]

\[
p(x) = 1 + x + x^4 \tag{2.21}
\]
式(2.18)より\((n, k)\)巡回符号を生成するLFSRのフィードバック多項式は、式(2.20), (2.21)より、以下の式で表される。

\[
\phi(x) = 1 + x^2 + x^3 + x^5 + x^7
\]

(2.22)

次に、(7, 4)巡回符号を判定する誤り検出器を構成する。誤り検出器は、XORゲートを用いて構成するが、XORゲートの組み合わせをフィボナッチ型のLFSRを用いて求める [24]。一般に\((n, k)\)巡回符号を用いたEBISTの誤り検出器を決めるには、以下の式でフィードバック多項式\(\phi_e(x)\)を決める。

\[
\phi_e(x) = \frac{x^n - 1}{g(x)}
\]

(2.23)

(7, 4)巡回符号の例では、\(\phi_e(x)\)は式(2.20)を用いて以下のように求められる。

\[
\phi_e(x) = \frac{x^7 - 1}{1 + x + x^3} = 1 + x + x^2 + x^4
\]

(2.24)

式(2.24)をフィードバック多項式として構成したフィボナッチ型のLFSRを図2.19に示す。なお、\(\oplus\)は排他的論理和の演算を表わす。図2.19に示すLFSRから出力される値を元に誤り検出器を構成する。

LFSRの\(FF_i\)に格納されている値を\(V_i\)とおき、\(n\)クロック動作させる。図2.19に示すLFSRは(7, 4)巡回符号に対するLFSRのため、7クロック動作させ、その出力

図2.19: 誤り検出器の構成を求めるフィボナッチ型LFSR

<table>
<thead>
<tr>
<th></th>
<th>(FF_1)</th>
<th>(FF_2)</th>
<th>(FF_3)</th>
<th>(FF_4)</th>
<th>出力</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(V_1)</td>
<td>(V_2)</td>
<td>(V_3)</td>
<td>(V_4)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(V_2 + V_3 + V_4)</td>
<td>(V_1)</td>
<td>(V_2)</td>
<td>(V_3)</td>
<td>(V_4)</td>
</tr>
<tr>
<td>3</td>
<td>(V_1 + V_2 + V_3)</td>
<td>(V_2 + V_3 + V_4)</td>
<td>(V_1)</td>
<td>(V_2)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(V_1 + V_3 + V_4)</td>
<td>(V_1 + V_2 + V_3)</td>
<td>(V_2 + V_3 + V_4)</td>
<td>(V_1)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(V_1 + V_3 + V_4)</td>
<td>(V_1 + V_2 + V_3)</td>
<td>(V_2 + V_3 + V_4)</td>
<td>(V_1)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>(V_1 + V_3 + V_4)</td>
<td>(V_1 + V_2 + V_3)</td>
<td>(V_2 + V_3 + V_4)</td>
<td>(V_1)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>(V_1 + V_3 + V_4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

表2.4: 図2.19のLFSRの出力
図 2.20: (7, 4) 巡回符号を用いた EBIST

力を表 2.4 に示す。表 2.4 では、巡回符号の情報ブロックを表す、\( V_1, V_2, V_3, V_4 \)に対して、冗長ブロックを、\( V_1 \oplus V_3 \oplus V_4, V_1 \oplus V_2 \oplus V_3, V_2 \oplus V_3 \oplus V_4 \) と表せることを示している。

誤り検出器では、情報ブロックが出力されている LFSR の出力から作った冗長ブロックと、冗長ブロックが出力されている LFSR の出力と XOR ゲートを用いて除的論理和を取り、符号語となっているかを確認する。XOR ゲートの出力が 1 であれば、冗長ビットか情報ビットが誤っており、LFSR の故障の可能性が考えられる。誤り検出器では、XOR ゲートの出力を OR ゲートに入力し、いずれかの XOR ゲートから 1 出力された場合、OR ゲートの出力から 1 が出力され、非符号語が LFSR によって生成され、LFSR が故障していると判定する。(7, 4) 巡回符号を用いた EBIST の図を図 2.20 に示す。

表 2.5 に図 2.20 の EBIST の LFSR に初期値として "1101000" を与えた際の動作を示す。生成されるテストパターンは、誤り検出器の出力が常に 0 となっていることからすべて (7, 4) 巡回符号の符号語である。誤り検出器は、LFSR が生成したすべてのテストパターンを組合せ回路を用いた誤り検出器によって、符号語が非符
表 2.5: 図 2.20 のテスト生成器が生成するテストパターン

<table>
<thead>
<tr>
<th>テストパターン</th>
</tr>
</thead>
<tbody>
<tr>
<td>01101000</td>
</tr>
<tr>
<td>01101000</td>
</tr>
<tr>
<td>00110100</td>
</tr>
<tr>
<td>00011010</td>
</tr>
<tr>
<td>10110000</td>
</tr>
<tr>
<td>01011110</td>
</tr>
<tr>
<td>00101111</td>
</tr>
<tr>
<td>10100001</td>
</tr>
<tr>
<td>11000110</td>
</tr>
<tr>
<td>01110001</td>
</tr>
<tr>
<td>10001110</td>
</tr>
<tr>
<td>01000111</td>
</tr>
<tr>
<td>10010111</td>
</tr>
<tr>
<td>11111111</td>
</tr>
<tr>
<td>11001011</td>
</tr>
<tr>
<td>11010000</td>
</tr>
</tbody>
</table>

誤り検出器の出力: 0
号語かを判定可能である。そのため LFSR を利用した CUT のテストの際にも、誤り検出器は有効であり、同時テストが可能となる。
2.5 まとめ

本章では、VLSI のテストにおける故障モデルとテストパターンについて説明し、テスト手法の一つである組込み自己テストについて説明した。組込み自己テストはテストコストの削減に有効であるが、テスト用の機構である BIST 回路は被テスト回路と同様に故障の発生が考えられ、故障時には被テスト回路のテスト実行結果の判定を誤ることでコストや信頼性の面で不利益を被る。本章では、BIST 回路の故障時も、正しいテスト実行結果を得られる高信頼組込み自己テストの概念について説明し、従来法を交えて説明した。
第3章 動的マスクを実現する耐故障テスト生成器

3.1 はじめに

2.4.4 節で述べた同時テスト可能なテスト生成器である EBIST [2] は LFSR の誤り検出が可能であるが、誤りを検出した場合は、外部テストを利用するなどして CUT の正常/不良判定を行う必要がある。本章では、高信頼組込み自己テストのための耐故障テスト生成器について述べる。提案するテスト生成器は EBIST による誤り検出を利用しているが、誤り検出後も CUT のテストを実行できる機構を備えている。耐故障テスト生成器を実現する手法は様々であり、この章で説明するテスト生成器は 2.3.2 で述べた動的マスク [10] を実現する。テスト生成器の再構成によって故障箇所の分離が可能な構成である。そのため主に製造欠陥が原因となる永久故障を検出する製造テストでの利用が想定される。一方で再構成実行時にはテスト生成器の再初期化が必要となるためテスト再実行の必要があり、テスト実行時間のオーバヘッドが必要となる。また付加するスペア回路や再構成のための切替機構などの回路面積の大きさによっては信頼性の低下が考えられる。このように提案する耐故障テスト生成器には回路面積と信頼性にトレードオフがあり、テスト生成器としての故障検出能力を備えつつ、信頼性が最も高くなる最適なテスト生成器の構成やユニット数が存在するかと考えられる。耐故障テスト生成器の信頼性が、再構成不可能な一般のテスト生成器よりも高い信頼性でなければ利用する価値はない。よって製造過程における面積当たりの永久故障発生率を仮定し、製造テストを想定した信頼性の評価を行う。

3.2 節では、提案する 2 つの耐故障テスト生成器について述べる。3.3 節では、3.2
節で提案する2つのテスト生成器の信頼性および故障検出率に関する評価を行う。

3.2 誤り検出機構を利用したテスト生成器

再構成を利用したテスト生成器では、誤り検出可能テスト生成器である EBIST を誤りの検出単位であるユニットとし、切替器によって故障ユニットと正常ユニットの切替を行う。以下では、再構成を利用した2つのテスト生成器として、待機冗長型テスト生成器、および縮退利用型テスト生成器それぞれのアーキテクチャと誤り検出時の回復方法について述べる。さらに、提案するテスト生成器の面積の大きさに応じた信頼性と故障検出率のトレードオフについて述べる。

3.2.1 待機冗長型テスト生成器

待機冗長型テスト生成器（冗長 TPG）は、再構成を利用したテスト生成器の1つであり、1個以上の通常ユニットと1つのスペアユニット、さらに通常ユニットとスペアユニットを適切に切り替える切替器から構成される。冗長 TPG が持つ通常ユニット数が k 個、スペアユニット数が s 個であり、1つのユニットの出力ビット数が j ビットの場合、出力ビット数 \( n = k \cdot j \) で表される。このような冗長 TPG を n ビット \((k + s)\) テスト生成器と呼ぶこととする。各ユニットは2.4.4 節で説明した EBIST であり、図 3.1 に (4,3) 回巡符号 (4 ビットパリティ符号) を用いた EBIST による誤り検出単位（ユニット）を示す。これは (4,3) 回巡符号を用いた EBIST の誤り検出器 (DET) 内部に FF と OR ゲートを付加し、ユニットをシリアルに接続して各ユニットの LFSR に初期値を設定するための scan – out ポートを付加したものである。LFSR 故障による誤りを出力すると同じユニット内の DET が誤りを検出し、誤り検出信号 (errorsignal) を出力する。図 3.1 に示すように DET 内部の FF と OR ゲートによって errorsignal は reset 信号が入力されるまで保持される。切替器はマルチプレクサ (MUX) と OR ゲートで構成され、故障を検出したユニットの errorsignal を受けて MUX の入力を切り替える。errorsignal は OR ゲートを
図 3.1: (4,3) 巡回符号を用いた EBIST による誤り検出単位 (ユニット)

通して伝搬し，故障ユニットを迂回するように再構成を行う．冗長 TPG では各ユニットが誤り検出の単位であり，切替器によって故障を検出したユニットを分離し，スペアユニットと適切に切り替えすることでテストを実行可能な状態に戻る．図 3.2 に図 3.1 に示す 4 ビット出力を持つ EBIST をユニットとした，12 ビット (3+1) テスト生成器を示す．

切替器によるテストパターンを生成するユニット切り替えにより，テストパターンを生成する LFSR が変化するため，初期値を再設定し，再びテストを実行する必要がある．このため，再構成によって故障を検出した LFSR を避けて各 LFSR に初期値を設定する必要がある．各ユニットに施されたスキャン設計を利用し，LFSR をシフトレジスタとして動作させ，LFSR の scan-out に接続された MUX によって各ユニットをシリアルに接続して初期値の再設定を行う．この MUX は errorsignal で入力である直前のユニットの MUX の出力か，現在のユニットの scan-out かを制御し，故障が発生したユニットを迂回するようにする．
図 3.2: 12 ビット (3+1) 待機冗長型テスト生成器
冗長 TPG の動作を以下に示す。まずテスト実行前にすべての通常ユニット内の LFSR に初期値を設定する。scan = 1 として LFSR をシフトレジスタとして動作させ、scan - in から LFSR に初期値を設定する。このとき reset = 1 として、DET 内の FF に対しても初期化を行う。テスト実行は、reset と scan を 0 とすることで開始する。すべての通常ユニットの LFSR がテストパターンを生成し、CUT に印加される。テスト実行中にいずれかの通常ユニットの errsignal が 1 となると、故障ユニットを分離するように、DET から errsignal が MUX に送られる。同時に故障ユニットの LFSR を迂回するように MUX が設定されるため scan = 1 として、scan - in から LFSR の初期値を再設定する。例えば、図 3.2 のテスト生成器で、テスト実行中にユニット 2 が故障した場合、ユニット 2 の errsignal が 1 となる。この errsignal は故障ユニット以後に伝搬され、MUX2 によってユニット 2 の出力はユニット 3 の出力に置き換えられる。同様に MUX3 によってユニット 3 の出力はスペアユニットの出力に置き換えられる。このとき、ユニット 1 は正常に動作しているため、MUX1 では出力の置き換えは起こらず、ユニット 1 の出力のままである。また初期値の再設定を行う際は、MUX4 と MUX6 の制御信号は 0 のままである。一方で MUX5 のみ制御信号が 1 に設定され、通常ユニット 1 と通常ユニット 3 がシリアルに接続され、通常ユニット 2 を迂回するように初期値の再設定が行われる。その後、scan = 0 とし、スペアユニットを含む各 LFSR が再度テストパターンを生成し、CUT のテストを再開する。

3.2.2 縮退利用型テスト生成器

縮退利用型テスト生成器 (縮退 TPG) は待機冗長型テスト生成器 (冗長 TPG) からスペアユニットを省いた構成である。縮退利用型テスト生成器が持つユニット数が k 個で、1 つのユニットの出力ビット数が j ビットの場合、出力ビット数 \( n = k \cdot j \) で表される。このような縮退 TPG を \( n \) ビット \((k + 0)\) テスト生成器と呼ぶこととする。縮退 TPG の各ユニットは 3.1 節で説明した EBIST と同様であり、通常ユニットを切り替えるための切替器を持つ。図 3.3 に図 3.1 に示す 4 ビット出力を持つ EBIST
図 3.3: 12 ビット (3+0) 縮退利用型テスト生成器

をユニットとした，12 ビット (3+0) テスト生成器を示す。

切替器によるテストパターンを生成するユニット切り替えにより，テストパターンを生成する LFSR が変化するため，初期値を再設定し，再びテストを実行する必要がある。このため，再構成によって故障を検出した LFSR を避けて各 LFSR に初期値を設定する必要がある。各ユニットに施されたスキャン設計を利用し，LFSR をシフトレジスタとして動作させ，LFSR の scan – out に接続された MUX によって各ユニットをシリアルに接続して初期値の再設定を行う。この MUX は errorsignal で入力である直前のユニットの MUX の出力が，現在のユニットの scan – out かを制御し，故障が発生したユニットを迂回するようにする。

縮退 TPG の動作を以下に示す。まずテスト実行前にすべての通常ユニット内の LFSR に初期値を設定する。scan = 1 として LFSR をシフトレジスタとして動作さ
せ、scan – in から LFSR に初期値を設定する。このとき reset = 1 として、誤り検出器 (DET) 内の FF に対しても初期化を行う。テスト実行は、reset と scan を 0 とすることで開始する。すべての通常ユニットの LFSR がテストパターンを生成し、CUT に印加される。テスト実行中にいずれかのユニットの errorsignal が 1 となると、故障ユニットを分離するように、DET から errorsignal が MUX に送られる。同時に故障ユニットの LFSR を迂回するように MUX が設定されるため scan = 1 として、scan – in から LFSR の初期値を再設定する。例えば、図 3.3 のテスト生成器で、テスト実行中にユニット 2 が故障した場合、ユニット 2 の errorsignal が 1 となる。この errorsignal は故障ユニット以後に伝播され、MUX2 によってユニット 2 の出力はユニット 3 の出力に置き換えられる。つまりユニット 3 の出力はテスト生成器全体の 12 ビットのうち、ユニット 2 2 つ分である 8 ビットを占めることとなる。なおユニット 1 は正常に動作しているため、MUX1 では出力の置き換えは起こらず、ユニット 1 の出力のままである。また初期値の再設定を行う際は、MUX4 の制御信号は 0 のままである。一方で MUX5 のみ制御信号が 1 に設定され、ユニット 1 とユニット 3 がシリアルに接続され、ユニット 2 を迂回するように初期値の再設定が行われる。その後、scan = 0 とし、スペアユニットを含む各 LFSR が再度テストパターンを生成し、CUT のテストを再開する。

このように縮退 TPG では、再構成後に生成されるテストパターンが再構成前と異なるため、再構成毎にシグネチャの期待値をテストに記憶しておく必要がある。またテスト生成器の機能縮退により、故障検出率の増減が起こる可能性があるが、スペアユニットがないため、冗長 TPG よりも小さい面積で構成可能であるという利点がある。

3.2.3 テスト生成器のユニット数と面積および故障検出率の関係

ここでは、テスト生成器のユニット数と面積、および故障検出率の関係について考察する。まず、冗長 TPG である n ビット (k + 1) テスト生成器 (出力ビット数 n, 通常ユニット数 k) を対象とする。なお TPG の出力ビット数 n は固定されてい
るものとする。

テスト生成器の面積は通常ユニット数 $k$ が大きくなると、小さくなる傾向がある。これは付加するスペアユニットの面積を決める、1 つのユニットに含まれる LFSR と誤り検出器 (DET) のビット数 $j = n/k$ に依存するためである。なお切替器の面積は $k$ にほとんど依存しない。図 3.4 に示すテスト生成器は図 3.2 に示すテスト生成器の出力ビット数 $n = 12$ のまま、$k$ を 2 とした 12 ビット (2+1) テスト生成器である。図 3.2 に示すテスト生成器では、スペアユニットのビット数は $12/3 = 4$ ビットであり、図 3.4 に示すテスト生成器では $12/2 = 6$ ビットのため、$k$ が小さい方がスペアユニットの面積が大きくなっている。一方で切替器の面積は、$k$ の大きさに対してほとんど変わらない。

次にテスト生成器が出力するテストパターン数とユニット数の関係を考える。1つのユニットが出力できるテストパターン数は $2^{n/k} - 1$ であり、3.2 節で説明した動作ではテスト生成器は $2^{n/k} - 1$ 種類のテストパターンを生成できる。よって $k$ が大きくなると、生成できるテストパターン数の種類が少なくなるため、CUT の故障検出率が低下する可能性がある。

このように冗長 TPG にはテスト生成器を構成するユニット数が増化すると主にスペアユニットの面積が小さくなることでテスト生成器としての面積が小さくなることで信頼性が高まるが、逆に 1 つあたりのユニットが小さくなることで生成可能なテストパターンの種類の減少に伴う故障検出率の低下というトレードオフが考えられる。

次に、縮退 TPG である $n$ ビット $(k+0)$ テスト生成器 (出力ビット数 $n$、ユニット数 $k$) を対象とする。なお出力ビット数 $n$ は固定されているものとする。

---

1) 厳密には、切替器は $k$ が大きくなると OR ゲートと各ユニット内の LFSR の初期値設定のための MUX の数や各ユニットを切り替える MUX を制御する OR ゲートが増えるためわずかに増加する。なおユニットの出力を切り替える MUX の数は増えるが、1 つの MUX は $n/k$ ビットであるためユニットの出力を切り替える MUX の占める面積は変わらない。また、LFSR も生成多項式が変化するため面積が多少増減する。
図 3.4: 12 ビット (2+1) 待機冗長型テスト生成器
図 3.5: 12 ビット (2+0) 縮退利用型テスト生成器

テスト生成器の面積は通常ユニット数 $k$ の大きさに対してほとんど変わらない。これは 1 つのユニットに含まれる LFSR と DET のビット数 $j = n/k$ とユニット数 $k$ で面積が決まるためである。なお冗長 TPG と同様に切替器の面積は $k$ にほとんど依存しない。図 3.5 に示すテスト生成器は図 3.3 に示すテスト生成器の出力ビット数 $n = 12$ のまま、$k$ を 2 とした 12 ビット (2+0) テスト生成器である。図 3.3 に示すテスト生成器では、1 つのユニットのビット数は $12/3 = 4$ ビットであり、図 3.5 に示すテスト生成器では $12/2 = 6$ ビットのため、$k$ が小さい方がユニット当たりの面積が大きくなっている。一方で切替器の面積は、$k$ の大きさに対して冗長 TPG と同様に、ほとんど変わらない。

次に縮退 TPG が出力するテストパターン数とユニット数の関係を考える。1 つのユニットが出力できるテストパターン数は $2^{n/k} - 1$ であり、3.2 節で説明した動作ではテスト生成器は $2^{n/k} - 1$ 種類のテストパターンを生成できる。縮退 TPG で
は故障による縮退動作時の際，1 つ目のユニットの故障時は残りの正常な \( k - 1 \) 個のユニットで \( n \) ビットのテストパターンを生成する。このときユニット数 \( k \) が小さい時は，1 つあたりのユニットの出力ビットが大きく縮退動作とはテストパターン中の多くのビットがほかのユニットの出力ビットと同じ値となる。一方 \( k \) が大きいときは，1 つ当たりの出力ビットは小さく，テストパターン中のほかのユニットの出力と同じ値となるビットは少なくなる。一般に LFSR は疑似ランダムパターンという 0 と 1 のランダム性を利用することで故障を検出するが，縮退動作によるランダム性の低下は縮退動作次の故障検出率の大幅な低下というトレードオフとなると考えられる。

このように冗長 TPG ではテスト生成器の信頼性と故障検出能力にトレードオフの関係があると考えられる。一方，縮退 TPG ではテスト生成器の面積と故障検出能力の間にトレードオフはないが，縮退後の故障検出率は \( k \) が小さいほど故障ユニットの出力をほかの正常ユニットが補うビット数が多くなるため故障検出率の低下が大きくなると考えられる。このため冗長 TPG，縮退 TPG のどちらにおいてもテスト生成器の信頼性の高さ，高い故障検出率を満たす最適なユニット数が存在すると考えられる。

3.3 節に示す実験では，面積当たりの故障発生率を仮定し，テスト生成器が期待通り動作する確率を求めることで信頼性の評価を行う。またユニット数 \( k \) を変化させ，ベンチマーク回路を対象とした故障検出率を求める実験を行う。

### 3.3 製造テストにおける評価

提案する再構成を利用したテスト生成器の信頼性評価を行う。再構成を利用したテスト生成器である待機冗長型テスト生成器（冗長 TPG）や縮退利用型テスト生成器（縮退 TPG）は，誤り検出時に正しい動作が可能な構成に再構成を行う。再構成によって故障回路を分離することで故障による誤りの影響がなくなるため永久故障に対する耐性が高いと考えられる。しかしながら冗長 TPG では出力ビット数が一定の場合，ユニット数 \( k \) が小さいほど生成可能なテストパターン数が多いこ
とから故障検出率が高いと考えられるが、テスト生成器の面積が増加することで故障発生率が増加し、信頼性が低下すると考えられる。一方、縮退 TPG では出力ビット数が一定の場合、ユニット数 が小さいほど再構成実行時の機能縮退の影響が大きく故障検出率が低下すると考えられるが、切替器の構成が簡素になることで面積が小さくなり、信頼性が高くなると考えられる。そのため再構成不可能な一般的なテスト生成器との比較してどちらがより高い信頼性が得られるかは未
知である。

よって冗長 TPG と縮退 TPG に対して製造テストを想定した永久故障仮定時の信頼度評価を行い、あわせて故障検出率についても評価・考察を行う。

3.3.1 製造テストにおけるテスト生成器の信頼度評価式

再構成を利用したテスト生成器である冗長 TPG と縮退 TPG の信頼度について考察する。ここで故障は製造時のみ発生すると仮定し、BIST 回路を利用した CUT のテスト実行時には故障は発生しないと仮定する。また各 TPG の LFSR，DET，切替器といった各要素に発生した故障は必ずその要素の出力に影響を与えるとしている。このとき各 TPG の信頼度 は故障が無く正常に動作する，または各 TPG の再構成機能により故障していても無故障時と同等の動作を行う確率と定義する。

信頼度  を考えるために、ユニット内の LFSR の信頼度を 、DET の信頼度を 、切替器の信頼度を で表すものとする。この 3 つの信頼度 は単位面積当たりの故障率 が与えられたとき，それぞれの面積に応じて決まると仮定する [25]。一般に故障率はシステムの面積や配線混雑度などで決まる [26] が，ここでは複雑度はどの要素でも同じであると仮定する。ここで 1 つの LFSR が占める面積を 1 つの DET が占める面積を で，切替器が

\footnote{縮退 TPG は故障時の再構成により，再構成前と TPG の構成が変化するため再構成前と同じテストパターンを発生することができない。冗長 TPG は再構成後も再構成前と同じテストパターンの生成が可能である。}
占める面積を $S_{SW}$ とすると，

$$R_{LFSR} = \exp(-p \cdot S_{LFSR}) \quad (3.1)$$

$$R_{DET} = \exp(-p \cdot S_{DET}) \quad (3.2)$$

$$R_{SW} = \exp(-p \cdot S_{SW}) \quad (3.3)$$

と表せる [27].

まずは冗長 TPG の信頼度 $R_{RED}$ について考える．冗長 TPG が正常に CUT のテスト実行をできるケースは少なくとも以下の 4 つのケースが考えられる．なお LFSR の故障は DET が正常ならば必ず検出され，DET 故障時は DET は正常時と異なる値を出力すると仮定する．テスト生成器により期待したテストパターンが生成できるのは以下の 4 つの場合となる．なお，ここでは LFSR が故障した場合は必ず検出器 (DET) により検出可能であると仮定している．また，DET が故障した場合の DET の出力は，不定（正常または異常のどちらを示すかわからない）と考えている．

- ケース 1: 通常ユニット，スペアユニット，切替器のすべてが正常な場合．
- ケース 2: ある 1 つの通常ユニット内の LFSR が故障していて，他の部分が正常な場合．この LFSR の故障は同じ通常ユニット内の DET により検出され，この故障した通常ユニットを正常なスペアユニットと置換することで，正常に動作する．
- ケース 3: ある 1 つの通常ユニット内の DET が故障していて，他の部分が正常な場合．この DET の故障により正常な LFSR を故障していると判断してしまう場合が考えられるが，この通常ユニットを正常なスペアユニットに置き換えるため，テスト生成器は正常に動作する．なお，故障した DET が故障した LFSR を見逃してしまうケースはこのケース 3 には含まれない．
- ケース 4: スペアユニットが故障していて，残りの部分が正常な場合．
$(k+1)$ 増長 TPG の信頼度 $R_{RED}$ は，式 (3.1)，(3.2)，(3.3) において定義した 3 つの信頼度 $R_{LFSR}$ , $R_{DET}$ , $R_{SW}$ を用いてそれぞれのケースが起こりうる確率の和として表現できる。各ケースの確率をそれぞれ $P_{R1}$ , $P_{R2}$ , $P_{R3}$ , $P_{R4}$ とするとき 4 つの確率は次のように表せる。

\[
P_{R1} = \exp(-p \cdot ((k+1) \cdot R_{LFSR} + (k+1) \cdot R_{DET} + R_{SW}))
\]

\[
P_{R2} = k \cdot (1 - \exp(-p \cdot R_{LFSR})) \cdot \exp(-p \cdot (k \cdot R_{LFSR} + (k+1) \cdot R_{DET} + R_{SW}))
\]

\[
P_{R3} = k \cdot (1 - \exp(-p \cdot R_{DET})) \cdot \exp(-p \cdot ((k+1) \cdot R_{LFSR} + k \cdot R_{DET} + R_{SW}))
\]

\[
P_{R4} = (1 - \exp(-p \cdot R_{LFSR})) \cdot (1 - \exp(-p \cdot R_{DET}))
\]

\[
\cdot \exp(-p \cdot (k \cdot R_{LFSR} + k \cdot R_{DET} + R_{SW}))
\]

この結果，信頼度 $R_{RED}$ は

\[
R_{RED} = P_{R1} + P_{R2} + P_{R3} + P_{R4}
\]  (3.4)

と表せる。

次に縮退 TPG の信頼度 $R_{STU}$ について考える。縮退 TPG が正常に CUT のテスト実行をできるケースは少なくとも以下の 3 つのケースが考えられる。なお信頼度を考える上での仮定はすべて増長 TPG と同じとする。

- ケース 1: すべてのユニット，切替器が正常な場合。

- ケース 2: ある 1 つのユニット内の LFSR が故障していって，他の部分が正常な場合。この LFSR の故障は同じユニット内の DET により検出され，この故障したユニットの出力を別の正常なユニットの出力と置換することで，正常に動作する。

- ケース 3: ある 1 つのユニット内の DET が故障していて，他の部分が正常な場合。この DET の故障により正常な LFSR を故障していると判断してしまう場合が考えられるが，このユニットを別の正常なユニットに置き換えるため，テスト生成器は正常に動作する。なお，故障した DET が故障した LFSR を見逃してしまうケースの場合はこのケース 3 には含まれない。
$(k + 0)$ 縮退 TPG の信頼度 $R_{STU}$ は，冗長 TPG と同様に 3 つの信頼度 $R_{LFSR}$, $R_{DET}$, $R_{SW}$ を用いてそれぞれのケースが起こりうる確率の和として表現できる．各ケースの確率をそれぞれ $P_{S1}, P_{S2}, P_{S3}$ とすると 3 つの確率は次のように表せる．

$$
P_{S1} = \exp(-p \cdot (k \cdot R_{LFSR} + k \cdot R_{DET} + R_{SW}))
$$

$$
P_{S2} = (1 - \exp(-p \cdot R_{LFSR})) \cdot \exp(-p \cdot ((k - 1) \cdot R_{LFSR} + k \cdot R_{DET} + R_{SW}))
$$

$$
P_{S3} = (1 - \exp(-p \cdot R_{DET})) \cdot \exp(-p \cdot (k \cdot R_{LFSR} + (k - 1) \cdot R_{DET} + R_{SW}))
$$

この結果，信頼度 $R_{STU}$ は

$$
R_{STU} = P_{S1} + P_{S2} + P_{S3} \quad (3.5)
$$

と表せる．

### 3.3.2 テスト生成器の面積

式 (3.4) と (3.5) を利用した信頼度評価のために，冗長 TPG と縮退 TPG を設計し，回路面積を評価する実験を行った．設計した冗長 TPG と縮退 TPG はテスト生成器の出力ビット幅を 64, 128, 256 ビットとして，ユニット数 $k$ は 64, 128 ビットでは 2, 4, 256 ビットでは 2, 4, 8, 16 とした．なお冗長 TPG に付加するスペアユニットの数は 1 つとしている．合わせて一般的なテスト生成器である LFSR のみで構成したテスト生成器 (通常 TPG) も合わせて設計し，評価の対象とした．面積の評価には Design Compiler (Synopsys, Inc.) を利用し，class ライブラリ [28] を利用した．

64, 128, 256 ビットの各テスト生成器の面積を表 3.1 に示す．面積は NOT ゲートを 1 とした時の値である．表 3.1 では左から順に，LFSR の面積，誤り検出器 (DET) の面積，切替器 (SW) の面積，各テスト生成器の総面積，および通常 TPG を基準としたときの面積比を示している．

表 3.1 より，すべてのビット幅において (2+1) 冗長 TPG の面積が最も大きく，通常 TPG のおよそ 2.44 倍から 2.50 倍を占めていることがわかる．これは 64 ビットでは 32 ビットの，128 ビットでは 64 ビットの，256 ビットでは 128 ビットである
表 3.1: 64, 128, 256 ビットの耐故障テスト生成器と通常テスト生成器の面積

<table>
<thead>
<tr>
<th></th>
<th>各回路の面積</th>
<th>総面積</th>
<th>面積比</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LFSR</td>
<td>DET</td>
<td>SW</td>
</tr>
<tr>
<td>64 ビット通常</td>
<td>588</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>64 ビット (2+1) 兀長</td>
<td>918</td>
<td>279</td>
<td>272</td>
</tr>
<tr>
<td>64 ビット (4+1) 兀長</td>
<td>765</td>
<td>275</td>
<td>284</td>
</tr>
<tr>
<td>64 ビット (2+0) 縮退</td>
<td>612</td>
<td>186</td>
<td>274</td>
</tr>
<tr>
<td>64 ビット (4+0) 縮退</td>
<td>612</td>
<td>220</td>
<td>282</td>
</tr>
<tr>
<td>128 ビット通常</td>
<td>1163</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>128 ビット (2+1) 兀長</td>
<td>1746</td>
<td>567</td>
<td>528</td>
</tr>
<tr>
<td>128 ビット (4+1) 兀長</td>
<td>1500</td>
<td>465</td>
<td>540</td>
</tr>
<tr>
<td>128 ビット (2+0) 縮退</td>
<td>1164</td>
<td>378</td>
<td>516</td>
</tr>
<tr>
<td>128 ビット (4+0) 縮退</td>
<td>1200</td>
<td>372</td>
<td>524</td>
</tr>
<tr>
<td>256 ビット通常</td>
<td>2301</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>256 ビット (2+1) 兀長</td>
<td>3462</td>
<td>1164</td>
<td>1040</td>
</tr>
<tr>
<td>256 ビット (4+1) 兀長</td>
<td>2910</td>
<td>945</td>
<td>1052</td>
</tr>
<tr>
<td>256 ビット (8+1) 兀長</td>
<td>2700</td>
<td>837</td>
<td>1076</td>
</tr>
<tr>
<td>256 ビット (16+1) 兀長</td>
<td>2499</td>
<td>935</td>
<td>1124</td>
</tr>
<tr>
<td>256 ビット (2+0) 縮退</td>
<td>2308</td>
<td>776</td>
<td>1028</td>
</tr>
<tr>
<td>256 ビット (4+0) 縮退</td>
<td>2328</td>
<td>756</td>
<td>1036</td>
</tr>
<tr>
<td>256 ビット (8+0) 縮退</td>
<td>2400</td>
<td>744</td>
<td>1052</td>
</tr>
<tr>
<td>256 ビット (16+0) 縮退</td>
<td>2352</td>
<td>880</td>
<td>1084</td>
</tr>
</tbody>
</table>

スペアユニットを構成する LFSR と DET の面積が大きいためである。通常ユニット数 k を増やすと兠長 TPG の面積は徐々に小さくなり、256 ビット (8+1) 兀長 TPG ではおよそ 2 倍の面積比までテスト生成器の面積が小さくなっていることがわかる。これは通常ユニット 1 つ当たりの面積が小さいため、スペアユニットも同様に面積が小さくなったためである。縮退 TPG はスペアユニットがないため兠長 TPG よりも小さい面積となっており、通常 TPG のおよそ 1.77 倍から 1.90 倍であることがわかる。これらの傾向はどの出力ビット幅のテスト生成器においても大きな変化はなかった。

3.3.3 テスト生成器の信頼度

表 3.2 に、表 3.1 で示した各テスト生成器の面積に基づいて、出力ビット数 64, 128, 256 ビットの兠長 TPG, 縮退 TPG, 通常 TPG の各テスト生成器の信頼度 R を示す。各テスト生成器の信頼度は式 (3.4), (3.5), (3.1) で表される。単位面積当たりの故障率として p は 1.0 × 10^{-4}, 5.0 × 10^{-5}, 1.0 × 10^{-5} を仮定している。ま
表 3.2: 64, 128, 256 ビットの耐故障テスト生成器と通常テスト生成器の信頼度

<table>
<thead>
<tr>
<th>ビット数</th>
<th>故障率 p</th>
<th>1.0 × 10^{-4}</th>
<th>5.0 × 10^{-5}</th>
<th>1.0 × 10^{-5}</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 ビット通常</td>
<td>0.9258</td>
<td>0.9622</td>
<td>0.9923</td>
<td></td>
</tr>
<tr>
<td>64 ビット (2+1) 元長</td>
<td>0.9682</td>
<td>0.9852</td>
<td>0.9972</td>
<td></td>
</tr>
<tr>
<td>64 ビット (4+1) 元長</td>
<td>0.9679</td>
<td>0.9848</td>
<td>0.9971</td>
<td></td>
</tr>
<tr>
<td>64 ビット (2+0) 緩退</td>
<td>0.9712</td>
<td>0.9860</td>
<td>0.9973</td>
<td></td>
</tr>
<tr>
<td>64 ビット (4+0) 緩退</td>
<td>0.9694</td>
<td>0.9852</td>
<td>0.9971</td>
<td></td>
</tr>
<tr>
<td>128 ビット通常</td>
<td>0.8910</td>
<td>0.9439</td>
<td>0.9885</td>
<td></td>
</tr>
<tr>
<td>128 ビット (2+1) 元長</td>
<td>0.9311</td>
<td>0.9691</td>
<td>0.9944</td>
<td></td>
</tr>
<tr>
<td>128 ビット (4+1) 元長</td>
<td>0.9333</td>
<td>0.9695</td>
<td>0.9945</td>
<td></td>
</tr>
<tr>
<td>128 ビット (8+1) 元長</td>
<td>0.9322</td>
<td>0.9687</td>
<td>0.9942</td>
<td></td>
</tr>
<tr>
<td>128 ビット (2+0) 緩退</td>
<td>0.9415</td>
<td>0.9720</td>
<td>0.9947</td>
<td></td>
</tr>
<tr>
<td>128 ビット (4+0) 緩退</td>
<td>0.9385</td>
<td>0.9710</td>
<td>0.9945</td>
<td></td>
</tr>
<tr>
<td>128 ビット (8+0) 緩退</td>
<td>0.9349</td>
<td>0.9694</td>
<td>0.9943</td>
<td></td>
</tr>
<tr>
<td>256 ビット通常</td>
<td>0.7944</td>
<td>0.8913</td>
<td>0.9773</td>
<td></td>
</tr>
<tr>
<td>256 ビット (2+1) 元長</td>
<td>0.8430</td>
<td>0.9318</td>
<td>0.9877</td>
<td></td>
</tr>
<tr>
<td>256 ビット (4+1) 元長</td>
<td>0.8540</td>
<td>0.9351</td>
<td>0.9888</td>
<td></td>
</tr>
<tr>
<td>256 ビット (8+1) 元長</td>
<td>0.8563</td>
<td>0.9353</td>
<td>0.9889</td>
<td></td>
</tr>
<tr>
<td>256 ビット (16+1) 元長</td>
<td>0.8530</td>
<td>0.9333</td>
<td>0.9883</td>
<td></td>
</tr>
<tr>
<td>256 ビット (2+0) 緩退</td>
<td>0.8764</td>
<td>0.9422</td>
<td>0.9893</td>
<td></td>
</tr>
<tr>
<td>256 ビット (4+0) 緩退</td>
<td>0.8702</td>
<td>0.9401</td>
<td>0.9891</td>
<td></td>
</tr>
<tr>
<td>256 ビット (8+0) 緩退</td>
<td>0.8645</td>
<td>0.9379</td>
<td>0.9889</td>
<td></td>
</tr>
<tr>
<td>256 ビット (16+0) 緩退</td>
<td>0.8572</td>
<td>0.9347</td>
<td>0.9883</td>
<td></td>
</tr>
</tbody>
</table>

た冗長 TPG，緩退 TPG それぞれで同じテスト生成器のピット幅，故障率で最も高い信頼度を太字で示している。表 3.2 より，緩退 TPG では信頼度が最も高くなるのはすべて (2+0) 緩退の構成であり，64 ビット冗長 TPG では (2+1) 冗長 TPG，同128 ビット，256 ビットではそれぞれ (4+1) 冗長 TPG, (8+1) 冗長 TPG であることがわかる。また通常 TPG と緩退 TPG および冗長 TPG では，故障率 p が大きくなるほど，また TPG の出力ビット幅が大きくなるほど信頼度の差が大きくなっている。各テスト生成器の信頼度は故障率 p の変化に対してほぼ同じ傾向が得られたため，図 3.6 に故障率 p = 5.0 × 10^{-5} における各テスト生成器の信頼度を示し，表 3.1 で示した各テスト生成器の面積を合わせて考察する。

図 3.6 より，すべての冗長 TPG，緩退 TPG で通常 TPG よりも高信頼となっていることがわかる。これはスペアユニットの付加や切替器の付加による面積増による信頼度の低下よりも，再構成によって故障時もテスト生成器による CUT のテストができることによる信頼度の増加が上回ったことを表している。特に緩退 TPG
図 3.6: 64, 128, 256 ビットの耐故障テスト生成器と通常テスト生成器の信頼度 ($p = 5.0 \times 10^{-5}$)

は冗長 TPG と比較して，ユニット数 $k$ が同じ場合，どのケースにおいても信頼度が上回っている．これはスペアユニットによる面積増がないため，式 (4.7) で表される無故障である確率が冗長 TPG と比較して高いためである．これは縮退 TPG と冗長 TPG において故障することが信頼度の低下を招く切替器が出力ビット数が大きい場合でもほぼ一定の面積増すむことが原因と考えられる．また図 3.6 より縮退 TPG ではユニット数 $k$ が少ないほど信頼度が高いことがわかる．これはユニット数の増加に対して，表 3.1 に示すように故障することが信頼度の低下を招く切替器の面積が $k$ が大きいほど増加しているためと考えられる．

冗長 TPG では表 3.1 に示すように，テスト生成器のビット幅が同じ場合はユニッ
ト数 \( k \) が小さいほど面積が大きくなるが、図 3.6 に示す信頼度は、64 ビットでは \( k = 2 \), 128 ビット、256 ビットではそれぞれ \( k = 4 \), \( k = 8 \) の構成が最も信頼度が高くなっている。これはユニット数 \( k \) が小さい冗長 TPG では、小さい \( k \) はスペアユニットが大きな面積を占めることを表しており、TPG 全体の面積オーバーヘッド増加により式 (4.7) で表される無障害である確率が低下したと考えられる。故障する信頼度の低下につながる切替器は \( k \) が大きいほど面積がわずかに増加するがこの切替器の面積をとスペアユニットの面積（テスト生成器全体の面積）のトレードオフの結果、64 ビット、128 ビット、256 ビットでそれぞれ \( k = 2 \), \( k = 4 \), \( k = 8 \) の構成が最も信頼度が高くなったと考えられる。

### 3.3.4 テスト生成器の故障検出率

次にベンチマーク回路に対する冗長 TPG と縮退 TPG をテスト生成器として利用した際の故障検出率について考察する。故障検出率を調べるために故障シミュレータとして HOPE [29] を用いた。被テスト回路（CUT）としては ISCAS’85 ベンチマーク回路 [30] の c7552、および ISCAS’89 ベンチマーク回路 [31] の s13207 と s15850 を用いた。ベンチマーク回路の FF はすべてスキャン FF とし、またベンチマーク回路の外部入出力もすべてバウンダリスキャン化し STUMPS アーキテクチャでスキャンチャーンへテストパターンを印加した。なおスキャンチャーンの本数は TPG の出力ピット数 (128 または 256) であり、各テスト生成器の LFSR が生成するテストパターン数は s15850 のみ 600,000 パターン、その他は 500,000 パターンとした①。

表 3.3 に 256 ビットの通常 TPG および冗長 TPG、表 3.4 に 128 ビットの通常 TPG および冗長 TPG をテスト生成器とした際の各ベンチマーク回路に対する故障検出率を示す。表 3.3, 3.4 より通常ユニット数 \( k = 8 \) までは通常 TPG とほぼ同等の故障検出率を得られているが、\( k = 16 \) の通常 TPG では故障検出率が大きく減少して

①通常 TPG で故障検出率がそれ以上上昇しなかったテストパターン数が 600,000 または 500,000 パターンだったためこれを基準とした。
表 3.3: ベンチマーク回路に対する故障検出率

<table>
<thead>
<tr>
<th>テスト生成器</th>
<th>故障検出率 [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 ビット通常</td>
<td>s15850</td>
</tr>
<tr>
<td>256 ビット (2+1) 兀長</td>
<td>95.55</td>
</tr>
<tr>
<td>256 ビット (4+1) 兀長</td>
<td>95.55</td>
</tr>
<tr>
<td>256 ビット (8+1) 兀長</td>
<td>95.54</td>
</tr>
<tr>
<td>256 ビット (16+1) 兀長</td>
<td>91.08</td>
</tr>
</tbody>
</table>

表 3.4: ベンチマーク回路における故障検出率 (128 ビット，500,000 パターン)

<table>
<thead>
<tr>
<th>テスト生成器</th>
<th>故障検出率 [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>128 ビット通常</td>
<td>s15850</td>
</tr>
<tr>
<td>128 ビット (2+1) 兀長</td>
<td>93.06</td>
</tr>
<tr>
<td>128 ビット (4+1) 兀長</td>
<td>93.33</td>
</tr>
<tr>
<td>128 ビット (8+1) 兀長</td>
<td>93.37</td>
</tr>
</tbody>
</table>

表 3.5: s15850 における縮退動作時の故障検出率

<table>
<thead>
<tr>
<th>テスト生成器</th>
<th>無故障時</th>
<th>縮退動作時</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 ビット通常</td>
<td>95.55</td>
<td>-</td>
</tr>
<tr>
<td>256 ビット (2+0) 縮退</td>
<td>95.47</td>
<td>92.88</td>
</tr>
<tr>
<td>256 ビット (4+0) 縮退</td>
<td>95.55</td>
<td>94.69</td>
</tr>
<tr>
<td>256 ビット (8+0) 縮退</td>
<td>95.55</td>
<td>94.68</td>
</tr>
<tr>
<td>256 ビット (16+0) 縮退</td>
<td>91.08</td>
<td>89.50</td>
</tr>
</tbody>
</table>

いることがわかる。これは通常ユニットの LFSR のビット数が小さくなったことで、出力できる固有なテストパターン数が減少した事によるものと考えられる。よってテスト生成器としての信頼度の高さと高い故障検出率を満たす最適なユニット数は、128 ビットテスト生成器では $k = 4$、256 ビットテスト生成器では $k = 8$ であることがわかった。

次に縮退 TPG に対する故障検出率について考察する。表 3.5 に s15850 に対して 256 ビットの出力を持つ縮退 TPG をテスト生成器として利用した際の故障検出率を求めた実験結果を示す。表 3.5 の無故障時の縮退 TPG での故障検出率は冗長 TPG と同じである。これは無故障時の縮退 TPG は通常 TPG と同じく $k$ 個のユニットの LFSR の出力をテストパターンとするためである。縮退動作時の故障検出率
は$k$個のユニットそれぞれが故障したとし、別の正常なユニットに置き換えた際の故障検出率の平均値を表している。表3.5より、(2+0)縮退TPGでの故障検出率は(4+0)縮退TPGや(8+0)縮退TPGでの故障検出率より低くなっていることがわかる。これは縮退動作によりユニット数$k$が小さな縮退TPGではいずれかの正常ユニットが障害ユニットの出力の代わりとなるビット幅が大きく、テストパターンの質が低下したためと考えられる。縮退動作時の故障検出率はテストパターン数の増加により、無故障時の故障検出率と同等またはそれ以上となるケースも存在した。

さらに縮退動作時のCUTに印加するテストパターン数を変化させた際の故障検出率について考察する。表3.6にs15850に対して256ビットの出力を持つ縮退TPGをテスト生成器として利用した際の縮退動作時の故障検出率とテストパターン数の関係を求めた実験結果を示す。表はそれぞれテスト生成器、縮退TPG無故障時の故障検出率、$k$個のユニットそれぞれの縮退動作時にはテストパターン数を600k、750k、900kとした際の故障検出率の平均値を表している。表3.6より、テストパターン数を750k、900kと無故障時のそれぞれ25%，50%多くした際は、(4+0)縮退TPGと(8+0)縮退TPGでは無故障時とほぼ同等の故障検出率を得られている。一方、(16+0)縮退TPGではテストパターン数の増加に対して故障検出率が変わっていないが、これは生成可能なテストパターンの種類数を超えるテストパターン数を生成しているためである。また(2+0)縮退TPGではテストパターン数を増加させても無故障時の故障検出率を下回っている。これは縮退TPGの機能縮退によって故障ユニットの出力ビットを正常なユニットの出力に置き換えた結果、いずれかの正常ユニットが障害ユニットの出力の代わりとなるビット幅が大きく、テストパターンの質が大きく低下したためと考えられる。

3.4まとめ

本章では、高信頼組込み自己テストのための耐故障テスト生成器として、動的マスクを実現するテスト生成器であるテスト生成器、および縮退利用型テスト生成器を提案し、その有効性について評価を行った。提案する2つのテスト生成器は
表 3.6: s15850 における縮退動作時の故障検出率とテストパターン数の関係

<table>
<thead>
<tr>
<th>テスト生成器</th>
<th>無故障時 (600k)</th>
<th>600k</th>
<th>750k</th>
<th>900k</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 ビット通常</td>
<td>95.55</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>256 ビット (2+0) 縮退</td>
<td>95.57</td>
<td>92.88</td>
<td>93.11</td>
<td>93.12</td>
</tr>
<tr>
<td>256 ビット (4+0) 縮退</td>
<td>95.55</td>
<td>94.69</td>
<td>95.07</td>
<td>95.12</td>
</tr>
<tr>
<td>256 ビット (8+0) 縮退</td>
<td>95.54</td>
<td>94.68</td>
<td>94.98</td>
<td>95.07</td>
</tr>
<tr>
<td>256 ビット (16+0) 縮退</td>
<td>91.08</td>
<td>89.50</td>
<td>89.50</td>
<td>89.50</td>
</tr>
</tbody>
</table>

一般にテスト生成器として利用される LFSR を複数の小さな LFSR で構成し、それぞれの LFSR に誤り検出器と誤り検出器の誤り検出情報を受けて LFSR(ユニット)の出力を切り替える切替器を備えている。実験結果から、動的マスクを実現する縮退利用型テスト生成器は、製造テストを想定した永久故障を仮定した環境下において、LFSR のみで構成した通常のテスト生成器と比較して、高い信頼度が得られることが分かった。また動的マスクを実現する待機冗長型テスト生成器は、同環境下において、再構成による故障検出率の低下なしに、通常のテスト生成器よりも高い信頼度が得られることが分かった。さらに縮退利用型テスト生成器では、再構成によって故障検出率が低下が見られるが、ユニット数が少ないほど故障検出率の低下が大きいが、一方ユニット数が多すぎて生成可能なテストパターンの種類が減少することで故障検出率が低下するという結果が得られ、実験結果からはおよそ 50%のテストパターン数増加でほぼ再構成前と同等の故障検出率が得られることが分かった。また実験結果からはテスト生成器の信頼性の高さと、高い故障検出率の両方を満たす最適なユニット数が存在することが分かった。
第4章 静的マスクを実現する耐故障
テスト生成器

4.1 はじめに

同時テスト可能なテスト生成器であるEBIST [2] はLFSRの誤り検出が可能であるが，誤りを検出した場合は，外部テストを利用することによってCUTの正常/不良判定を行う必要がある．本章では，2.3.1で述べた静的マスクを実現する耐故障テスト生成器について述べる．提案するテスト生成器は静的マスク [10] によって耐故障テスト生成器を実現する．3章の手法はEBISTの誤り検出能力のみを利用してているが，ここでは誤り訂正可能な巡回符号を用いたEBISTを利用し，巡回符号の誤り訂正能力を利用する．また3章で提案した再構成実行時にはテスト生成器の再初期化が必要となるため，テスト実行時点のオーバーヘッドが想定されるが，誤り訂正可能テスト生成器では誤り訂正による静的マスクを行うことで，誤りを含むテストパターンを出力する前に訂正できることから，故障発生時のテスト再実行は不要である．提案するテスト生成器では誤り訂正を行うことで，主に放射線衝突原因となる一時故障の影響が考えられるフィールドテストでの利用が想定される．そのためテスト生成器動作中に複数回の故障が発生した場合でも誤り訂正によってテスト生成器が期待通り動作可能である．しかしながら，誤り訂正を行う回路の面積オーバーヘッドによっては訂正を行うことによる信頼性の向上を面積増によって打ち消すことが考えられる．耐故障テスト生成器であるからには誤り訂正不可の一般のテスト生成器より信頼性が高くなければならない．そのため面積当たりの一時故障発生率を仮定し，与えられた時間内にテスト実行を完了できる確率で信頼性の評価を行う．
4.2 節では，巡回符号を利用して誤り訂正可能テスト生成器について述べる。4.3
節では，フィールドテストにおけるテスト生成器の信頼性の評価について
議論し，フィールドテストを対象としたオンライン BIST 環境における信頼性評価
を行う。

4.2 巡回符号を利用した誤り訂正可能テスト生成器

誤り訂正可能テスト生成器 (訂正 TPG) は誤り検出可能なテスト生成器である
EBIST を誤り訂正可能な巡回符号を利用して構成し，誤り訂正回路を付加するこ
とで耐故障テスト生成器を実現する。3.2 で示した再構成可能なテスト生成器は，
テスト生成器の誤りを検出時は故障回路を分離する再構成を行うが，再構
成時にはテスト生成器の再初期化が必須であり，テストの再実行が必要となる。
誤り訂正可能テスト生成器ではテスト生成器の誤り検出時に誤り訂正を行うことで，
テスト実行を止めることなく継続できる。誤り訂正可能テスト生成器は，(n, k) 巡
回符号 [21, 22] の符号語を生成する LFSR(巡回符号 LFSR)，誤り検出器 (DET)，訂
正ビット決定器 (SYN)，訂正実行器 (COR) の 4 つからなる。(n, k) 巡回符号の符号
語のみを生成する n ビットの出力を持つ誤り訂正可能テスト生成器を (n, k) 訂正
TPG と呼ぶ。なお 1 ビット誤り訂正可能な (n, k) 訂正 TPG は，k^2 ≥ 1 + n を満たす必
要がある。訂正 TPG は符号化を利用して LFSR の誤り検出を行い，誤り検出の情
報からテストパターン出力前にその誤りを訂正することで，CUT へ誤りを含む期
待値と異なるテストパターンの印加を防ぐことを目的とする。誤り検出・訂正の
ための付加回路，DET，SYN，COR はそれぞれ LFSR が生成する (n, k) 巡回符号
に対応し，組合せ回路として構成することで，LFSR のテストパターン生成と同時
刻内での誤り訂正を可能とする。なお，訂正 TPG では 2.4.4 節に示した EBIST と
同様の方法で LFSR と DET を構成する。EBIST では LFSR のみで構成される通常
のテスト生成器と同等の高い故障検出率が得られることから，訂正 TPG も同様に
高い故障検出率が得られると考えられる。(n, k) 巡回符号では，一般に n − k の大
きさが誤り検出・訂正能力を意味する。
図4.1に1ビット訂正可能な(7,4)巡回符号を利用した誤り訂正可能テスト生成器である(7,4)訂正TPGを示す。訂正TPGの動作を以下に示す。まずテスト実行前にLFSRに初期値を設定する。scan = 1としてLFSRをシフトレジスタとして動作させ、scan − inからLFSRに初期値を設定する。テスト実行は、scan = 0とすることで開始する。LFSRの生成するテストパターンは、DET、SYN、CORによって符号語かどうかをチェックし、非符号語であれば訂正を実行し、LFSRのFFに訂正済みのテストパターンを戻すと共にCUTにテストパターンを出力する。一般に(n, k)訂正TPGでは、LFSRの生成するnビットテストパターンに対してDETでn − kビットの誤り情報(シンドローム)を生成する。シンドロームがすべて0のベクトルであれば、LFSRの生成したテストパターンは符号語であり、COR内のセレクターによってMUXからnビットのゼロベクトル(どのビットも訂正が不要である)が出力され、訂正は実行されない。一方で、シンドロームがゼロベクトル以外であれば、SYNによってシンドロームからnビットの誤り位置情報ベクトルが生成される。このベクトルの各ビットは0であれば訂正が必要、1であれば訂正が必要であることを意味する。COR内部のセレクタではORゲートによってMUXのSYNからの入力が選択され、CORにて訂正が必要なビットの反転を行うことで訂正を実行する。

ここで図4.1の訂正TPGにおいてLFSRの左から5番目のFFに一時故障が発生し、FFに保持されているビットが1から0へと反転し、図4.2(a)のように誤ったベクトル"1011000"(正しくは"1011100")がLFSRから出力されたとする。図中の下線部は無故障時と異なる誤ったビットである。このときDETではLFSRの出力に対して3ビットの"011"というシンドロームが生成される。SYNはこのシンドロームを入力として対応する誤り位置情報ベクトルを生成する。シンドローム"011"からは誤り位置情報ベクトル"000100"が生成され、CORに入力される。CORでは誤り位置情報ベクトルを受けてLFSRの出力に対して5ビット目の訂正を行う。その結果、LFSRの出力ベクトルのうち5ビット目は0から1へ訂正され、LFSRの左から6番目のFFへと取り込まれる。また"1011100"へと訂正されたLFSRの出
図 4.1: (7,4) 巡回符号を用いた誤り訂正可能生成器

力は同時にテストパターンとして CUT へ印加される。この誤り訂正は組合せ回路である DET，SYN，COR により，一時故障の発生による誤りの出力と同じサイクルで行われる。

訂正 TPG は COR 内部にセレクターと呼ぶ機構を持ち，SYN の故障による誤りを訂正 TPG の出力へ伝搬するのを防ぐ役目を果たしている。ここで図 4.1 の訂正 TPG において SYN に一時故障が発生し，DET の生成したシンドロームがゼロベクトルであるにもかかわらず，図 4.2(b) のように“0101100”という間違った誤り位置情報ベクトルを生成したとする。このベクトルを参照することで COR では訂正を行うが，訂正 TPG ではセレクター内部の OR ゲートにより，シンドロームがゼ
ロベクトルであれば、SYN からの誤り位置情報ベクトルはマスクされるよう設計されている。そのためこのようなケースでは不要な訂正を間違って行うことはなく、訂正 TPG は“1011100”という期待通りのテストパターンを CUT に印加する。

一方、DET または COR の故障時は訂正 TPG は期待通り符号語のテストパターンを生成できないと考えられる。ここで図 4.1 の訂正 TPG において DET に一時故障が発生し、図 4.2(c) のように LFSR は無故障だがシンドロームとして“010”を生成したケースを考える。このケースでは SYN と COR は誤ったシンドロームを受けて“0100000”という誤り位置情報ベクトルによって LFSR の出力の 2 ビット目を反転させる。その結果、訂正 TPG は不要な訂正の実行により、非符号語をテストパターンとして出力する。

訂正 TPG ではこれらの例のように誤りを即時訂正することで、訂正 TPG に故障が発生しても無故障であるときと同じテストパターンを得ることができるので、訂正 TPG は一時故障の耐性が大幅に向上していると考えられる。4.3 節に示す実験では、訂正 TPG が正しくテストパターンを生成できる確率について評価する。
する。

4.3 フィールドテストにおける評価

提案する巡回符号を利用した誤り訂正可能テスト生成器の信頼性評価を行う。巡回符号を利用した誤り訂正可能テスト生成器（訂正 TPG）では検出した誤りを訂正するため、誤り訂正実行時に冗長 TPG や縮退 TPG と異なり、テストの再実行及び再初期化の必要が無くテストを継続することができるという利点がある。訂正 TPG では誤り訂正可能回数に制限がないため、一時故障が複数回発生する環境下では、冗長 TPG や縮退 TPG では再構成可能回数に限りがあるのに対し、訂正 TPG ではその都度訂正することで複数回の一時故障に対応可能である。そのためこのような環境下では信頼度が高くなると考えられる。訂正 TPG では誤り訂正能力が高いほど、多くの故障に対する誤りを訂正できることで信頼性が向上すると考えられるが、訂正回路の面積増化によるオーバヘッドによって信頼性が逆に低下することが考えられる。

よって、訂正 TPG に対してフィールドテストを想定した一時故障仮定時の信頼度評価を行う。

4.3.1 フィールドテストにおけるテスト生成器の信頼度評価式

訂正 TPG に付加した誤り訂正のための回路である誤り検出器 (DET)、訂正ピット決定器 (SYN)、訂正実行器 (COR) は、LFSR の誤り訂正を行うことで、一時故障に対する信頼性の向上が見込める。また利用する巡回符号の誤り訂正が高いほど、多くの故障による誤りを訂正できるため信頼性の向上が考えられる。しかし、ながら誤り訂正能力が大きいほど订正回路の面積増によってテスト生成器の面積オーバヘッドが大きくなり、テスト生成器自体の故障確率が上昇することで、逆に信頼性が下がることが懸念される。これらの相反する要素について議論するために、ここでは訂正 TPG の信頼性モデルを紹介し、面積当たりの故障発生率を利用した信頼性評価を行う。
ここで、TPGの信頼性を確率モデル、Test-reliability $R(t)$を用いて考えることにする。これにより、TPGが時刻$t$まで期待通り動作する、つまり時刻$t$から時刻$t$まで一時故障の有無にかかわらず、期待通りのテストパターンを生成できた確率を表す。ここで、TPGの時刻0でのTest-reliabilityは以下のよう表せる [26]。

$$R(0) = 1.$$  \hspace{0.5cm} (4.1)

これは時刻0で、TPGに一時故障が発生していない（無故障である）ことを意味する。

Test-reliability $R(t)$は、TPGが時刻$t-1$まで期待通りのパターンを生成し、さらに時刻$t$でも期待通りのテストパターンを生成できる確率で表すことができる。ここで、TPGが時刻$t-1$まで期待通りのテストパターンを生成しているとの条件の下、時刻$t$でも期待通りのテストパターンを生成できる確率を期待動作確率$Q(t)$で表すとする。このとき、$Q(t)$を用いて、Test-reliabilityは、

$$R(t) = Q(t) \cdot R(t-1) = \prod_{i=0}^{t} Q(i).$$  \hspace{0.5cm} (4.2)

のように表すことができる。

次に、対象とする一時故障についての仮定を述べる。一時故障は1サイクル以内に高々1つ発生するものとする。一時故障による誤りは一時故障の発生と同一サイクルに、一時故障が発生した回路要素（FFや論理ゲートなど）から出力されるもののとする。また発生した一時故障は次の時刻には消えるものとする。さらに一時故障の発生確率は回路要素の面積に依存し、時刻には依存しないものとする。

このような仮定の下では、期待動作確率$Q(t)$は時刻$t$に依存しない。そのため$Q(t) = Q$のように、時刻に依存しない$Q$で表すことができる。その結果、式 (4.2)は、

$$R(t) = Q^t.$$  \hspace{0.5cm} (4.3)

のように書き換えられる。式 (4.3)において、$Q$はテスト生成器ごとに異なるが、どのようなテスト生成器においても適用できる一般的な表現である。

70
次に訂正 TPG の期待動作確率, \( Q_{EC} \) について議論する. ここで仮定する訂正 TPG は連続する \( c \) ビット誤り ( \( c \) ビットバースト誤り) 訂正可能な巡回符号を適用したものですとする. なお, 以下の \( Q_{EC} \) の議論は \( c = 1 \) とすることで単一ビット誤り訂正可能な符号を利用した訂正 TPG にも適用可能である. 訂正 TPG が期待通りのテストパターンを生成できるケースは, 以下の3つのケースに分けることができる.

- **Case 1**: 一時故障の発生はなく, 無故障である.
- **Case 2**: LFSR に一時故障が発生するケース. このケースではさらに2つのケースに分けることができる. それぞれのケースは, 以下の通りである. (a)FF(メモリ部) に一時故障が発生, (b)XOR ゲート (ロジック部) に一時故障が発生.
- **Case 3**: SYN に一時故障が発生するケース.

これらのケースは完全に排他的であるため, それぞれのケースの確率を \( P_{EC1} \), \( P_{EC2} \) そして \( P_{EC3} \) と表すこととする. そのため訂正 TPG の期待動作確率, \( Q_{EC} \) は以下のように表すことができる.

\[
Q_{EC} = P_{EC1} + P_{EC2} + P_{EC3}. \tag{4.4}
\]

以下ではそれぞれの確率 \( P_{EC1} \), \( P_{EC2} \), \( P_{EC3} \) について考える. 先の仮定の通り, 一時故障の発生確率は面積のみに依存し, 故障率とよぶ定数 \( \lambda \) で表すとする. 文献 [27] では, 面積 \( S \) の回路要素が期待通り動作する確率 \( P \) は,

\[
P = \exp(-\lambda \cdot S). \tag{4.5}
\]

と表される. 以下では式 (4.5) に基づいて3つの確率を求める．

- 確率 \( P_{EC1} \): ケース 1 はすべての構成要素に故障が発生していないケースであるため, ケース 1 の発生確率は, 以下のように表すことができる.

\[
P_{EC1} = \exp(-\lambda \cdot (S_{LFSR} + S_{DET} + S_{SYN} + S_{COR})). \tag{4.6}
\]
ここで $S_{LFSR}$, $S_{DET}$, $S_{SYN}$, $S_{COR}$ はそれぞれ LFSR, DET, SYN, COR の面積を表す．一般に故障率は回路のメモリ部 (FF) とロジック部（論理ゲート）で異なり，メモリ部の故障率はロジック部の故障率よりも大きい [8]．ここでは単位面積当たりの故障率を 2 つ仮定し，単位面積当たりのメモリ部の故障率は $\lambda_M$，単位面積当たりのロジックの故障率は $\lambda_L$ で表すとする．このとき式 (4.6) は，以下のように書き換えることができる．

$$P_{EC1} = \exp (-\lambda_L \cdot (S_{LFSR(L)} + S_{DET} + S_{SYN} + S_{COR}) - \lambda_M \cdot S_{LFSR(M)}). \quad (4.7)$$

ここで $S_{LFSR(L)}$ と $S_{LFSR(M)}$ はそれぞれ LFSR のロジック部（XOR ゲート，初期値設定のための AND ゲート，NOT ゲート，MUX）とメモリ部 (FF) の面積を表している．よって $S_{LFSR(L)}$ と $S_{LFSR(M)}$ はそれぞれ以下のように書き表せる．

$$S_{LFSR(L)} = N_{XOR} \cdot S_{XOR} + S_{SCAN},$$

$$S_{LFSR(M)} = N \cdot S_{FF}. \quad (4.8)$$

なお，$N_{XOR}$，$S_{XOR}$，$S_{SCAN}$，$N$ そして $S_{FF}$ はそれぞれ LFSR 内の XOR ゲート数，XOR ゲートの面積，初期値設定のための回路面積，LFSR の出力ビット数そして FF の面積を表している．なお $S_{LFSR(M)}$ と $S_{LFSR(L)}$ の関係式は以下の通りである．

$$S_{LFSR} = S_{LFSR(L)} + S_{LFSR(M)}.$$  

式 (4.6) と (4.7) は LFSR のみで構成されるテスト生成器 (通常 TPG) に対する訂正 TPG の面積オーバヘッド，DET，SYN，COR によって期待動作確率 $Q_{EC}$ が減少することを意味する．

• 確率 $P_{EC2}$: 上記の通り、ケース 2 はさらに 2 つのケースに分けられる．1 つは連続する FF が $c$ 個故障，もう 1 つは XOR ゲートが 1 つだけ故障のケースで
ある。前後のケースが起こる確率は以下のよう考えることができる。1つのFFに一時故障が発生する確率は、
\[1 - \exp(-\lambda_M \cdot S_{FF})\]
と表すことができ、さらに\(N\) 個のFFのうち1つだけが故障するケースは\(NC_1 = N\) と\(N\)通りを考えられる。よって、いずれかのFFが故障する確率は\(N \cdot (1 - \exp(-\lambda_M \cdot S_{FF}))\)
となる。同じく、連続する\(c\) 個のFFが故障するケースは\(NC_1 = N\) と\(N\)通りであり、
\(N \cdot (1 - \exp(-\lambda_M \cdot S_{FF}))^c\) で表される。また連続する\(c\) ビット誤り訂正可能であれば、連続する\(c - 1\) ビット誤りも訂正可能である\((c \geq 2)\)。
そのため連続する\(c\) 個以下のFFが故障し、その他の\(N - c\) 個のFFやLFSR中のXORゲート、DET、SYN、CORがすべて無故障である確率は、以下のよう表せる。

\[
\sum_{i=1}^{c} \exp(-\lambda_L \cdot (S_{LFSR(L)} + S_{DET} + S_{SYN} + S_{COR}) \\
- \lambda_M \cdot (S_{LFSR(M)} - i \cdot S_{FF})) \\
\cdot N \cdot (1 - \exp(-\lambda_M \cdot i \cdot S_{FF})).
\]

また、後者のいずれか1つのXORゲートのみが故障する確率も同様に、

\[
\exp(-\lambda_L \cdot (S_{LFSR(L)} + S_{DET} + S_{SYN} + S_{COR}) \\
- S_{XOR}) - \lambda_M \cdot S_{LFSR(M)}) \\
\cdot N_{XOR} \cdot (1 - \exp(-\lambda_L \cdot S_{XOR})),
\]

と表すことができる。よってケース2の確率\(P_{EC2}\)は以下のように表すことができる。

\[
P_{EC2} = \sum_{i=1}^{c} \exp(-\lambda_L \cdot (S_{LFSR(L)} + S_{DET} + S_{SYN} + S_{COR}) \\
- \lambda_M \cdot (S_{LFSR(M)} - i \cdot S_{FF})) \\
\cdot N \cdot (1 - \exp(-\lambda_M \cdot i \cdot S_{FF})) \\
+ \exp(-\lambda_L \cdot (S_{LFSR(L)} + S_{DET} + S_{SYN} + S_{COR}) \\
- S_{XOR}) - \lambda_M \cdot S_{LFSR(M)}) \\
\cdot N_{XOR} \cdot (1 - \exp(-\lambda_L \cdot S_{XOR})).
\] (4.9)
式（4.9）は期待動作確率 $Q_{EC}$ が訂正 TPG の誤り訂正によって上昇する確率を
表している。誤り訂正能力である $c$ が大きいほど確率 $P_{EC2}$ は大きくなると考えられるが、さらなる訂正回路の面積オーバヘッド増大により、確率 $P_{EC1}$
は減少すると考えられる。

・確率 $P_{EC3}$：ケース 3 は故障した SYN が誤った誤り位置情報ベクトルを生成す
るが、COR は誤って本来不要な LFSR の出力の訂正を行わないことで、故障
した SYN による $Q_{EC}$ への悪影響を与えないと確率を表している。ってケー
ス 3 は訂正 TPG の期待動作確率 $Q_{EC}$ に加えることができる。$P_{EC3}$ は SYN
のみが障害し、その他の LFSR, DET, COR は正常である確率であり、以下
のように表すことができる。

$$
P_{EC3} = (1 - \exp(-\lambda_L S_{SYN})) 
\cdot \exp(-\lambda_L (S_{LFSR(L)} + S_{DET} + S_{COR})) 
- \lambda_M S_{LFSR(M)}.
$$

(4.10)

確率 $P_{EC1}$ は期待動作確率 $Q_{EC}$ の DET, SYN, COR の面積オーバヘッドによる
減少を表しているが、$P_{EC2}$ と $P_{EC3}$ は訂正 TPG の誤り訂正機能によって $Q_{EC}$ を
向上させることができる。つまり $P_{EC1}$ と $P_{EC2}$, $P_{EC3}$ は DET, SYN, COR のよ
うな追加要素に対する相互変数確率を表している。なお、この期待動作確率の解
析は訂正 TPG が DET または COR への一時障害では誤り訂正ができないという悲
観的な解析であることは事実である。

4.3.2 誤り訂正能力と面積および期待動作確率の関係

提案する誤り訂正可能テスト生成器（訂正 TPG）の誤り訂正能力と面積および期
待動作確率の関係について考察する。式（4.7）, （4.9）, （4.10）によって表される訂
正 TPG の面積オーバヘッドによって期待動作確率を低下させる要素と向上させる要素
の解析のためには、訂正 TPG を構成する各要素の面積が必要である。

74
提案する巡回符号を用いた訂正 TPG の誤り訂正能力と面積および期待動作確率の関係を求めるため、テスト生成器の出力ビット幅を 15, 35 とした訂正 TPG を設計した。なお 15 ビットの訂正 TPG は 1 ビット誤り訂正可能、2 ビットバースト誤り訂正可能な巡回符号を利用し、35 ビットの訂正 TPG は 1 ビット誤り訂正可能、3 ビットバースト誤り訂正可能な巡回符号を利用して構成した。あわせて、一般的なテスト生成器である LFSR のみで構成したテスト生成器（通常 TPG）設計し、評価を行った。面積の評価には Design Compiler (Synopsys, Inc.) を利用し、nangate ライブラリ [32] を用いた。与えた遅延制約はすべて 6.6ns とされている。

表 4.1 に各テスト生成器の面積の内訳および通常テスト生成器に対する面積比を示す。表 4.1 より 1 ビット誤り訂正可能な訂正 TPG は通常 TPG のおよそ 2.04 から 2.26 倍の面積であるが、2 ビットバースト誤り訂正可能な 15 ビット訂正 TPG では 3.42 倍、3 ビットバースト誤り訂正可能な 35 ビット訂正 TPG では 3.03 倍と非常に大きな面積を占めていることがわかる。表 4.1 より、これらの多重ビット誤り訂正可能な訂正 TPG では、LFSR, DET, COR の面積増を比較して SYN の面積が大きく増大していることがわかる。これは多重ビット誤り訂正を行うために DET で生成される誤り情報（シンドローム）の種類が増えたことで、SYN が生成するシンドロームに対応する誤り位置情報ベクトルの種類が増大し、SYN を構成する論理ゲートの数が増加したためである。

ここで表 4.1 の面積を利用し、式 (4.4) よりそれぞれの訂正 TPG の期待動作確率を求めた結果を表 4.2 に示す。表 4.2 ではロジック部の故障率 $\lambda_L = 10^{-7}$ とし、メモリ部の故障率をロジック部と同じ、5 倍、10 倍、50 倍、100 倍としたときそれぞれの期待動作確率を示している。なお表中の期待動作確率は、同じ出力ビット

<table>
<thead>
<tr>
<th>情報</th>
<th>訂正能力</th>
<th>LFSR</th>
<th>DET</th>
<th>SYN</th>
<th>COR</th>
<th>総面積</th>
<th>面積比</th>
</tr>
</thead>
<tbody>
<tr>
<td>(15,11) 訂正</td>
<td>1 ビット</td>
<td>94.70</td>
<td>-</td>
<td>28.73</td>
<td>-</td>
<td>181.41</td>
<td>2.04</td>
</tr>
<tr>
<td>15 ビット通常</td>
<td>2 ビットバースト</td>
<td>98.03</td>
<td>45.22</td>
<td>118.43</td>
<td>42.40</td>
<td>304.08</td>
<td>3.42</td>
</tr>
<tr>
<td>(35,29) 訂正</td>
<td>1 ビット</td>
<td>208.28</td>
<td>97.13</td>
<td>106.21</td>
<td>88.05</td>
<td>448.25</td>
<td>2.26</td>
</tr>
<tr>
<td>(35,27) 訂正</td>
<td>3 ビットバースト</td>
<td>221.84</td>
<td>54.80</td>
<td>184.34</td>
<td>88.05</td>
<td>602.61</td>
<td>3.03</td>
</tr>
<tr>
<td>35 ビット通常</td>
<td>-</td>
<td>198.70</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>198.70</td>
<td>1.00</td>
</tr>
</tbody>
</table>
表 4.2: $\lambda_L = 10^{-7}$ における表 4.1 に示すテスト生成器の期待動作確率

<table>
<thead>
<tr>
<th>TPG</th>
<th>$\lambda_M$</th>
<th>誤り訂正能力</th>
<th>$\lambda_L$</th>
<th>$5 \times \lambda_L$</th>
<th>$10 \times \lambda_L$</th>
<th>$50 \times \lambda_L$</th>
<th>$100 \times \lambda_L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(15.11) 訂正</td>
<td>(35.29) 訂正</td>
<td>1 ビット</td>
<td>0.999993</td>
<td>0.999993</td>
<td>0.999993</td>
<td>0.999993</td>
<td>0.999994</td>
</tr>
<tr>
<td>(35.29) 訂正</td>
<td>(35.27) 訂正</td>
<td>2 ビットパート</td>
<td>0.999991</td>
<td>0.999991</td>
<td>0.999991</td>
<td>0.999992</td>
<td>0.999992</td>
</tr>
<tr>
<td>(35.27) 訂正</td>
<td>(35.29) 訂正</td>
<td>3 ビットパート</td>
<td>0.999981</td>
<td>0.999981</td>
<td>0.999981</td>
<td>0.999982</td>
<td>0.999983</td>
</tr>
</tbody>
</table>

表 4.3: 表 4.1 に示す 35 ビット訂正 TPG の期待動作確率の内訳

<table>
<thead>
<tr>
<th>$\lambda_M$</th>
<th>訂正 TPG</th>
<th>$P_{EC1}$</th>
<th>$P_{EC2}$</th>
<th>$P_{EC3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_L$</td>
<td>(35.29) 訂正</td>
<td>0.999955</td>
<td>0.000021</td>
<td>0.000005</td>
</tr>
<tr>
<td>(35.27) 訂正</td>
<td>0.999940</td>
<td>0.000022</td>
<td>0.000118</td>
<td></td>
</tr>
<tr>
<td>$5 \times \lambda_L$</td>
<td>(35.29) 訂正</td>
<td>0.999877</td>
<td>0.000099</td>
<td>0.000005</td>
</tr>
<tr>
<td>(35.27) 訂正</td>
<td>0.999862</td>
<td>0.000100</td>
<td>0.000118</td>
<td></td>
</tr>
<tr>
<td>$10 \times \lambda_L$</td>
<td>(35.29) 訂正</td>
<td>0.999779</td>
<td>0.000197</td>
<td>0.000005</td>
</tr>
<tr>
<td>(35.27) 訂正</td>
<td>0.999764</td>
<td>0.000198</td>
<td>0.000005</td>
<td></td>
</tr>
<tr>
<td>$50 \times \lambda_L$</td>
<td>(35.29) 訂正</td>
<td>0.998998</td>
<td>0.000979</td>
<td>0.000005</td>
</tr>
<tr>
<td>(35.27) 訂正</td>
<td>0.998982</td>
<td>0.000980</td>
<td>0.000118</td>
<td></td>
</tr>
<tr>
<td>$100 \times \lambda_L$</td>
<td>(35.29) 訂正</td>
<td>0.998022</td>
<td>0.001956</td>
<td>0.000005</td>
</tr>
<tr>
<td>(35.27) 訂正</td>
<td>0.998006</td>
<td>0.001958</td>
<td>0.000118</td>
<td></td>
</tr>
</tbody>
</table>

幅のテスト生成器で高い期待動作確率が得られたものを太字で示している。表 4.2 よりどちらの訂正 TPG でも 1 ビット誤り訂正可能訂正 TPG の方が高い期待動作確率となっている。またメモリ部の故障率の増大とともに期待動作確率はほぼ同じか、わずかに増大していることがわかる。各訂正 TPG の期待動作確率について解析するため、期待動作確率の内訳を表 4.3 に示す。

表 4.3 は各 $\lambda_M$ ごとの 35 ビット訂正 TPG の期待動作確率の内訳、$P_{EC1}$、$P_{EC2}$、$P_{EC3}$ を示す。なお内訳 $P_{EC1}$、$P_{EC2}$、$P_{EC3}$ それぞれに対して高い値を太字で示している。確率 $P_{EC1}$ を比較すると、$\lambda_M$ が大きくなるほど低下し、それに 3 ビットパート誤り訂正可能な (35.27) 訂正 TPG の方が $P_{EC1}$ が低いことがわかる。これは多重ビットパート誤りを訂正するための面積増によるものである。しかしながら確率 $P_{EC2}$ からは (35.29) 訂正 TPG と (35.27) 訂正 TPG には大きな差は見られない。これは (35.27) 訂正 TPG ではより多くの故障時において誤りを訂正できるが、表 4.1 に示す面積増により、その効果が薄れたものと考えられる。一方、メモリ部の故障率が大きくなるほど確率 $P_{EC2}$ は増加していることがわかる。これはメモリ部の故障率が高ければ、LFSR の誤り訂正の効果が大きいことを示している。また確率 $P_{EC3}$ では (35.27) 訂正 TPG が (35.29) 訂正 TPG よりも大きな値を
なっている。これは故障しても誤りをマスク可能である SYN の大幅な面積増により、SYN のみが故障しているケースの確率が大きく上昇したためと考えられる。

つまり誤り訂正能力とテスト生成器の期待動作確率にはトレードオフが存在し、これからの結果から１ビット誤り訂正可能な巡回符号を利用した訂正 TPG で最も高い期待動作確率となるといえる。

4.3.3 テスト生成器の面積と期待動作確率

誤り訂正可能テスト生成器（訂正 TPG）においては１ビット誤り訂正可能訂正 TPG が最も期待動作確率が高くなることが分かったが、実際に、誤り訂正によって誤り訂正を行わない通常のテスト生成器（通常 TPG）や誤り検出可能なテスト生成器（EBIST）との期待動作確率の関係は明らかになっていない。そのため提案する訂正 TPG や通常 TPG および EBIST の面積および期待動作確率を評価する実験を行った。テスト生成器の出力ビット幅は 31, 63, 70, 127, 240, 255 とした。なお訂正 TPG はすべて１ビット誤り訂正可能な巡回符号を利用して構成した。あわせて、一般的なテスト生成器である LFSR のみで構成したテスト生成器（通常 TPG）とバリティ符号を用いた EBIST を設計し、評価を行った。面積の評価は先ほどと同様に Design Compiler (Synopsys, Inc.) を利用し、nangate ライブラリ [32] を用いた。与えた遅延制約は同じく 6.6ns としている。

ここで通常 TPG と EBIST の期待動作確率について考える。通常 TPG は LFSR のみで構成し、耐故障の要素は持ち合わせていないため、誤り検出および訂正はできない。一方 EBIST は誤り検出可能なテスト生成器である。これは巡回符号を生成する LFSR と誤り検出器（DET）で構成する。つまり、EBIST は訂正 TPG のうち、SYN と COR を省いた構成であると言える。そのため通常 TPG と EBIST が期待動作可能なケースは訂正 TPG ではケース 1 に該当する。よって、通常 TPG および EBIST の期待動作確率，$Q_{Gen}$, $Q_{EB}$ は以下のように表すことができる。
表 4.4: 1 ビット誤り訂正可能な訂正 TPG、通常 TPG および EBIST の面積

<table>
<thead>
<tr>
<th></th>
<th>LFSR</th>
<th>DET</th>
<th>SYN</th>
<th>COR</th>
<th>総面積</th>
<th>面積比</th>
</tr>
</thead>
<tbody>
<tr>
<td>(31,26) 訂正</td>
<td>191.52</td>
<td>78.20</td>
<td>39.10</td>
<td>76.87</td>
<td>385.70</td>
<td>2.12</td>
</tr>
<tr>
<td>31 ビット 通常</td>
<td>181.52</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>181.52</td>
<td>1.00</td>
</tr>
<tr>
<td>31 ビット EBIST</td>
<td>184.72</td>
<td>53.47</td>
<td>-</td>
<td>-</td>
<td>238.19</td>
<td>1.31</td>
</tr>
<tr>
<td>(63,57) 訂正</td>
<td>365.22</td>
<td>181.94</td>
<td>75.81</td>
<td>156.94</td>
<td>779.91</td>
<td>2.20</td>
</tr>
<tr>
<td>63 ビット 通常</td>
<td>354.49</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>354.49</td>
<td>1.00</td>
</tr>
<tr>
<td>63 ビット EBIST</td>
<td>357.94</td>
<td>104.54</td>
<td>-</td>
<td>-</td>
<td>462.48</td>
<td>1.30</td>
</tr>
<tr>
<td>(70,63) 訂正</td>
<td>401.93</td>
<td>232.14</td>
<td>89.38</td>
<td>171.70</td>
<td>895.14</td>
<td>2.25</td>
</tr>
<tr>
<td>70 ビット 通常</td>
<td>398.28</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>398.28</td>
<td>1.00</td>
</tr>
<tr>
<td>70 ビット EBIST</td>
<td>399.87</td>
<td>115.71</td>
<td>-</td>
<td>-</td>
<td>515.58</td>
<td>1.29</td>
</tr>
<tr>
<td>(127,120) 訂正</td>
<td>727.51</td>
<td>405.60</td>
<td>133.27</td>
<td>313.08</td>
<td>1579.45</td>
<td>2.19</td>
</tr>
<tr>
<td>127 ビット 通常</td>
<td>720.00</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>720.00</td>
<td>1.00</td>
</tr>
<tr>
<td>127 ビット EBIST</td>
<td>724.79</td>
<td>206.68</td>
<td>-</td>
<td>-</td>
<td>931.47</td>
<td>1.29</td>
</tr>
<tr>
<td>(240,232) 訂正</td>
<td>1376.28</td>
<td>813.96</td>
<td>425.33</td>
<td>594.78</td>
<td>3210.35</td>
<td>2.36</td>
</tr>
<tr>
<td>240 ビット 通常</td>
<td>1360.32</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1360.32</td>
<td>1.00</td>
</tr>
<tr>
<td>240 ビット EBIST</td>
<td>1365.11</td>
<td>387.03</td>
<td>-</td>
<td>-</td>
<td>1752.14</td>
<td>1.29</td>
</tr>
<tr>
<td>(255,247) 訂正</td>
<td>1460.87</td>
<td>873.01</td>
<td>434.64</td>
<td>632.02</td>
<td>3400.54</td>
<td>2.35</td>
</tr>
<tr>
<td>255 ビット 通常</td>
<td>1448.70</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1448.70</td>
<td>1.00</td>
</tr>
<tr>
<td>255 ビット EBIST</td>
<td>1451.89</td>
<td>410.97</td>
<td>-</td>
<td>-</td>
<td>1862.86</td>
<td>1.29</td>
</tr>
</tbody>
</table>

\[ Q_{Gen} = \exp(-\lambda_L \cdot S_{LFSR(L)} - \lambda_M \cdot S_{LFSR(M)}) \]  
\[ Q_{EB} = \exp(-\lambda_L \cdot (S_{LFSR(L)} + S_{DET}) - \lambda_M \cdot S_{LFSR(M)}) \]  

なお，EBIST の方がテスト生成器としての面積が大きくため，明らかに \( Q_{Gen} > Q_{EB} \) が成り立つ。

表 4.4 に訂正 TPG，通常 TPG，EBIST それぞれの面積を示す。表の各項目はそれぞれの TPG の種類およびビット幅，TPG を構成する各要素である LFSR，DET，
SYN，COR それぞれの面積，TPG の総面積，また通常 TPG を基準とした面積比である。表 4.4 より EC-TPG は通常 TPG の 2.19～2.36 倍の面積となっていることがわかる。また EBIST は通常 TPG のおよそ 1.26～1.30 倍である。

表 4.4 に示す論理合成によって求めた面積より各 TPG の期待動作確率を式 (4.4)，(4.11)，(4.12) より求めたものを表 4.5 に示す。なお式 (4.4) のうち，\( P_{EC2} \) については式 (4.9) に対して \( c = 1 \) とすることで求められる。一時故障の発生確率はロジック部の発生確率 \( \lambda_L \) を \( 10^{-7} \) として計算した。表 4.5 の各行はメモリ部の一時故障発
生確率 \( \lambda_M \) を \( 10^{-7} (\lambda_L \text{と等しい}) \times 10^{-7}, 5 \times 10^{-7}, 10^{-6} \) としたときの期待動

78
作確率を表している。表4.5において3つのTPGのうち、最も高い期待動作確率のものを太字で示している。

表4.5より、$\lambda_M$が増大すると、通常TPGとEBISTの期待動作確率は低下するが、
訂正TPGでは高い期待動作確率を保っている。これは$\lambda_M$の増大が、式(4.11)と
(4.12)において$\lambda_M \cdot S_{LFSR(M)}$の確率を大きく低下させたことで、通常TPGとEBIST
のLFSRの信頼性を大きく低下させたことが原因と考えられる。一方、訂正TPG
では$\lambda_M$の増大が同じく式(4.7)に示すようにLFSRの信頼性を低下させたが、式
(4.9), (4.10)に示すようにメモリ部をまったく含まないDET, SYN, CORの回路が
低下したLFSRの信頼性を補っている。特に式(4.9)のうち、$N \cdot (1 - \exp(-\lambda_M \cdot S_{FF}))$
の数値の増大が低下したLFSRの信頼性を補っている。

表4.5より、TPGの出力ビット数$n$が増大すると期待動作確率は低下している
ことがわかる。これは$n$の増大により、TPGの各要素の面積が増加し、式(4.7),
(4.11), (4.12)に示すように期待動作確率が低下したためである。例えば、式(4.8),
(4.11)では$n$の増大によって$S_{LFSR(M)}$が大きくなることがわかるが、それによっ
て$Q_{Gen}$は減少することとなる。表より訂正TPGではほとんどのケースにおいて

<table>
<thead>
<tr>
<th></th>
<th>$\lambda_M \cdot S_{LFSR(M)}$</th>
<th>$2 \times \lambda_M \cdot S_{LFSR(M)}$</th>
<th>$5 \times \lambda_M \cdot S_{LFSR(M)}$</th>
<th>$10 \times \lambda_M \cdot S_{LFSR(M)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(31,26)</td>
<td>訂正</td>
<td>0.999985</td>
<td>0.999985</td>
<td>0.999985</td>
</tr>
<tr>
<td>31ビット通常</td>
<td>0.999982</td>
<td>0.999965</td>
<td>0.999913</td>
<td>0.999826</td>
</tr>
<tr>
<td>31ビットEBIST</td>
<td>0.999976</td>
<td>0.999959</td>
<td>0.999907</td>
<td>0.999820</td>
</tr>
<tr>
<td>(63,57)</td>
<td>訂正</td>
<td>0.999966</td>
<td>0.999966</td>
<td>0.999966</td>
</tr>
<tr>
<td>63ビット通常</td>
<td>0.999965</td>
<td>0.999930</td>
<td>0.999824</td>
<td>0.999648</td>
</tr>
<tr>
<td>63ビットEBIST</td>
<td>0.999954</td>
<td>0.999919</td>
<td>0.999813</td>
<td>0.999637</td>
</tr>
<tr>
<td>(70,63)</td>
<td>訂正</td>
<td>0.999959</td>
<td>0.999959</td>
<td>0.999959</td>
</tr>
<tr>
<td>70ビット通常</td>
<td>0.999959</td>
<td>0.999921</td>
<td>0.999804</td>
<td>0.999608</td>
</tr>
<tr>
<td>70ビットEBIST</td>
<td>0.999948</td>
<td>0.999928</td>
<td>0.999813</td>
<td>0.999637</td>
</tr>
<tr>
<td>(127,120)</td>
<td>訂正</td>
<td>0.999928</td>
<td>0.999928</td>
<td>0.999928</td>
</tr>
<tr>
<td>127ビット通常</td>
<td>0.999930</td>
<td>0.999857</td>
<td>0.999644</td>
<td>0.999290</td>
</tr>
<tr>
<td>127ビットEBIST</td>
<td>0.999907</td>
<td>0.999836</td>
<td>0.999623</td>
<td>0.999269</td>
</tr>
<tr>
<td>(240,232)</td>
<td>訂正</td>
<td>0.999858</td>
<td>0.999858</td>
<td>0.999858</td>
</tr>
<tr>
<td>240ビット通常</td>
<td>0.999864</td>
<td>0.999730</td>
<td>0.999328</td>
<td>0.998658</td>
</tr>
<tr>
<td>240ビットEBIST</td>
<td>0.999825</td>
<td>0.999691</td>
<td>0.999289</td>
<td>0.998619</td>
</tr>
<tr>
<td>(255,247)</td>
<td>訂正</td>
<td>0.999848</td>
<td>0.999848</td>
<td>0.999848</td>
</tr>
<tr>
<td>255ビット通常</td>
<td>0.999855</td>
<td>0.999713</td>
<td>0.999286</td>
<td>0.998574</td>
</tr>
<tr>
<td>255ビットEBIST</td>
<td>0.999814</td>
<td>0.999671</td>
<td>0.999244</td>
<td>0.998533</td>
</tr>
</tbody>
</table>
最も高い期待動作確率となっている．これは$Q_{EC}$において$n$の増化によって，訂正TPGにおいてLFSRの誤りを訂正できるケースの増化によって確率$P_{EC2}$が増化したためである．一方，メモリ部の故障率$\lambda_M=\lambda_L=10^{-7}$の時はTPGの出力ビット数が70ビット以上において訂正TPGの期待動作確率がわずかに通常TPGよりも小さくなっていることがわかる．これは訂正TPGを構成する各要素の面積の増加によって訂正TPGが無故障である確率$P_{EC1}$が大きく低下し，確率$P_{EC2}$によるLFSRの誤り訂正ができることによる$Q_{EC}$の増加分では補えなくなったためである.

4.3.4 オンラインBISTにおける信頼度評価

提案する誤り訂正可能テスト生成器（訂正TPG）はリアルタイムシステムのためのオンラインテストのように，テスト実行時間に対する制約が大きいケースにおいてとくに一時故障の耐性が効果を発揮すると考えている．これは訂正TPGは一時故障による誤りを訂正することで待機冗長型テスト生成器などと異なり，テスト再実行が不要なため，テスト実行時間が一定となるためである．この効果を示すために，訂正TPGと誤り訂正能力の無いTPG（誤り検出可能なTPG，検出TPG）をリアルタイムシステムのためのオンラインテストに適用するケースを考える．なお誤り訂正能力の無いTPG（検出TPG）は訂正TPGからSYNとCORを省いた構成であり，誤り検出可能なテスト生成器であるEBISTに相当する．ここでは対象とするオンラインテストの構造は，文献[33]を対象とする．この構造では，テスト対象回路であるリアルタイムアプリケーションの通常動作と通常動作の間の短い時間にテストを実行する．このような短い時間のテストを繰り返し実行することで，求められる故障検出率を達成する．

ここで$L$を連続する通常動作の間に設けられたテスト実行用に与えられたサイクル数とする．次にテスト実行で与えられた$L$サイクルにおいて，CUTにはテスト生成器（訂正TPGまたは検出TPG）から$T$パターンが印加されるものとする．このとき，テストはtest-per-cycleで実行するものとする．つまり，テスト生成器は$L$
サイクル以内に連続して $T$ サイクルの間，期待通りのテストパターンを CUT に印加できればテストは成功とし，できなければフェイルとする．なお，$T < L$ である．なお，この $T$ と $L$ の関係は一般にあまり大きな差はなく，$L < 2T$ を想定するものとする．さらにこれらの $T$ と $L$ の制約において，訂正 TPG または検出 TPG がテストパターンの生成に成功する確率を期待ステスト実行確率と表すとし，それぞれ $\hat{R}_{EC}(T, L)$, $\hat{R}_{ED}(T, L)$ で表されるものとする．このテストでは訂正 TPG や通常 TPG では，シンプルなテスト実行のケースのみが考えられる．つまり訂正 TPG がテストを開始し，$T$ サイクルでテストを期待通り終える確率である．図 4.3 にテスト実行の例を示す．図 4.3 の例 (a) は無故障時のテスト実行を示している．仮に $T$ サイクルのテスト実行中に一時障害が発生した場合，訂正 TPG では図 4.3 の例 (b) のように誤りを訂正し，期待通りのテストパターンを生成することで，テストを中断することがない「テスト継続」いうケースが考えられる．よって訂正 TPG の期待テスト実行確率は $\hat{R}_{EC}(T, L) = R_{EC}(T)$ で表される．

一方，検出 TPG ではテスト実行において異なる「テスト再実行」が考えられる．これは検出 TPG では一時障害による誤りの訂正はできないが，検出はできるためである．ここで検出 TPG がサイクル $k$ で一時障害による誤りを検出したケースを考える．このとき，与えられた $L$ サイクルのうち，テスト実行時間の残りサイク
ルとテスト実行に必須な $T$ サイクルの関係が $L - k \geq T$ であれば、検出 TPG はテスト再実行が可能である。 図 4.3 の例 (c)–(e) に示すテスト再実行では、例 (c) と(d) ではテスト再実行があっても $L$ サイクル以内に $T$ サイクルのテストが完了しているが、例 (e) では、$L$ サイクル以内にテスト再実行が完了せず、テスト再実行に失敗しているケースを示している。

このように、検出 TPG の期待テスト実行確率 $\hat{R}_{ED}(T, L)$ は以下のように考えることができる。まず、検出 TPG が図 4.3 の例 (c), (d) のように各サイクル $h (T < h \leq L)$ においてテスト再実行を完了する確率 $TC(T, h)$ を考える。これは検出 TPG が $h - T$ サイクルで誤りを検出した後、$h - T + 1$ サイクルからテストを再実行したのち $T$ サイクルの間、一時故障が一切発生せずテスト再実行に成功するケースを表している。このとき、サイクル 0 からサイクル $h - T - 1$ においては、さまざまなテスト実行のケースが考えられる。例えば、サイクル 0 から $h - T - 1$ まで一時故障が一切発生しないケース、サイクル 0 とサイクル 1 で一時故障が発生し、その後サイクル $h - T + 1$ サイクルまで一時故障が発生しなかったケースなどである。つまり、検出 TPG がサイクル 0 からサイクル $h - T + 1$ の間の各サイクル $j (j = 0, 1, 2, \ldots h - T - 1)$ で、誤りを検出した回数を数え上げることで、確率 $TC(T, h)$ を求めることができる。なお検出 TPG の期待動作確率 $Q_{ED}$ より、一時故障が発生する確率を $1 - Q_{ED}$ と求めることができる。このとき、$TC(T, h)$ は

---

1 場所には、誤りを検出した検出 TPG 中の LFSR への初期値の再設定のためにいくつかのサイクルが必要だが、ここでは誤りを検出した後のサイクルで、初期値の再設定とテスト再実行を行えるものと仮定する。
以下のよう表すことができ、

\[
TC(T, h) = h_{-T-1}C_0 \cdot Q_{ED}^{h-T-1} \cdot (1 - Q_{ED}) \cdot Q_{ED}^T \\
+ h_{-T-1}C_1 \cdot Q_{ED}^{h-T-2} (1 - Q_{ED}) \\
\cdot (1 - Q_{ED}) \cdot Q_{ED}^T \\
+ \ldots \\
+ h_{-T-1}C_h \cdot (1 - Q_{ED})^{h-T-1} \\
\cdot (1 - Q_{ED}) \cdot Q_{ED}^T \\
= \sum_{j=0}^{h-T-1} h_{-T-1}C_j \cdot Q_{ED}^{h-T-1-j} \cdot (1 - Q_{ED})^j \\
\cdot (1 - Q_{ED}) \cdot Q_{ED}^T. \tag{4.13}
\]

さらに式 (4.13) を簡化することで以下のように表せる。

\[
TC(T, h) = (1 - Q_{ED}) \cdot Q_{ED}^T. \tag{4.14}
\]

よって期待テスト実行確率 \( \hat{R}_{ED}(T, L) \) は、時一故障が全く発生しないケース（図 4.3 的例 (a)) の確率, \( Q_{ED}^T \) と考え得るすべてのテスト再実行の確率, \( TC(T, T+1), TC(T, T+2) \ldots TC(T, L) \) から,

\[
\hat{R}_{ED}(T, L) = Q_{ED}^T + \sum_{h=T+1}^{L} TC(T, h) \\
= Q_{ED}^T \cdot (1 + (L - T) \cdot (1 - Q_{ED})). \tag{4.15}
\]

と表すことができる²

式 (4.15) より、サイクル \( L \) と \( T \) の差が大きいほど、検出 TPG の期待テスト実行確率が高くなると考えられる。一方、差が小さいときは、テスト再実行できるケースが少なくなることで期待テスト実行確率が低いと考えられる。

²オンライン BIST において通常動作と通常動作の間でテストを実行する時間をテストセッションとおく。厳密には複数のテストセッションを経て求められる故障検出率を得るため、1 セッションだけで期待テスト実行確率を判断すべきではないが、ここでは一時故障を対象とし、各テストセッション毎に BIST 回路へ初期値を読み込むと仮定するため、テストセッションにおいてテスト開始時には式 (4.1) に示すように一時故障による影響を受けていないものと考えることができる。
2つの期待テスト実行確率，\( \hat{R}_{ED}(T, L) \) と \( \hat{R}_{EC}(T, L) \) を比較すると，テスト実行時間の制約のあるオンライン BIST では，誤り訂正によるテスト継続が可能な訂正 TPG の方が，誤り検出時にはテスト再実行を行う検出 TPG より優位であると考えることができる．検証のため，テスト再実行に利用できるサイクル数 \( L - T \) が小さいケースと比較的大きなケースの 2 通りを考える．

テスト再実行のための時間的余裕が少ないケース：このケースではサイクル数 \( L \) と \( T \) の差は比較的少なく，\( L = 540, T = 450 \) とした \( ((L - T)/L=20\%) \)．表 4.6 に，ロジック部の故障率 \( \lambda_L = 10^{-7} \) としたときの検出 TPG と訂正 TPG の期待テスト実行確率を示す．期待していたとおり，表 4.6 より訂正 TPG がほぼすべてのケースで検出 TPG より高い期待テスト実行確率が得られていることがわかる．検出 TPG の方が高い期待テスト実行確率を示しているのは，240 ビットと 255 ビットのテスト生成器においてロジック部とメモリ部の故障率が等しい \( (\lambda_L = \lambda_M) \) ときのみである．表 4.5 に示した結果では，ロジック部とメモリ部の故障率が等しい \( (\lambda_L = \lambda_M) \) ときはすべてのビット幅で検出 TPG よりも訂正 TPG の方が高い期待動作確率を得られていたため，テスト再実行によってこの 2 つのケースは訂正 TPG を上回る期待テスト実行確率が得られたことがわかる．しかしながらこのケースではテスト再実行による検出 TPG に対する効果は小さく，訂正 TPG が優位と言える．

テスト再実行のための時間的余裕が多いケース：このケースではサイクル数 \( L \) と \( T \) の差は比較的多く，\( L = 540, T = 360 \) とした \( ((L - T)/L=50\%) \)．しかしながらこのような大きな差はテスト実行時間の限られるオンライン BIST では非現実的である可能性が高い．このケースでは，テスト再実行が可能な検出 TPG が優位であるにもかかわらず，ロジック部とメモリ部の故障率が等しい \( (\lambda_L = \lambda_M) \) ときを除いて，訂正 TPG の方が高い期待テスト実行確率が得られていることがわかる．

これら 2 つのケースから，テスト再実行が可能な検出 TPG と比較して，テスト継続が可能な訂正 TPG はリアルタイムシステムのためのオンライン BIST において非常に有効であると言える．
表 4.6: テスト再実行のための時間的余裕が少ないケースにおける期待テスト実行確率

<table>
<thead>
<tr>
<th>TPG (\lambda_M)</th>
<th>(\lambda_L)</th>
<th>(2 \times \lambda_L)</th>
<th>(5 \times \lambda_L)</th>
<th>(10 \times \lambda_L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(31,26) 訂正 31bit 検出</td>
<td>0.9932</td>
<td>0.9932</td>
<td>0.9932</td>
<td>0.9932</td>
</tr>
<tr>
<td>(63,57) 訂正 63bit 検出</td>
<td>0.9848</td>
<td>0.9848</td>
<td>0.9848</td>
<td>0.9848</td>
</tr>
<tr>
<td>(70,63) 訂正 70bit 検出</td>
<td>0.9819</td>
<td>0.9819</td>
<td>0.9819</td>
<td>0.9819</td>
</tr>
<tr>
<td>(127,120) 訂正 127bit 検出</td>
<td>0.9880</td>
<td>0.9880</td>
<td>0.9880</td>
<td>0.9880</td>
</tr>
<tr>
<td>(240,232) 訂正 240bit 検出</td>
<td>0.9379</td>
<td>0.9380</td>
<td>0.9380</td>
<td>0.9380</td>
</tr>
<tr>
<td>(255,247) 訂正 255bit 検出</td>
<td>0.9338</td>
<td>0.9339</td>
<td>0.9339</td>
<td>0.9343</td>
</tr>
</tbody>
</table>

表 4.7: テスト再実行のための時間的余裕が多いケースにおける期待テスト実行確率

<table>
<thead>
<tr>
<th>TPG (\lambda_M)</th>
<th>(\lambda_L)</th>
<th>(2 \times \lambda_L)</th>
<th>(5 \times \lambda_L)</th>
<th>(10 \times \lambda_L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(31,26) 訂正 31bit 検出</td>
<td>0.9945</td>
<td>0.9945</td>
<td>0.9945</td>
<td>0.9945</td>
</tr>
<tr>
<td>(63,57) 訂正 63bit 検出</td>
<td>0.9878</td>
<td>0.9878</td>
<td>0.9878</td>
<td>0.9878</td>
</tr>
<tr>
<td>(70,63) 訂正 70bit 検出</td>
<td>0.9855</td>
<td>0.9855</td>
<td>0.9855</td>
<td>0.9855</td>
</tr>
<tr>
<td>(127,120) 訂正 127bit 検出</td>
<td>0.9743</td>
<td>0.9743</td>
<td>0.9743</td>
<td>0.9744</td>
</tr>
<tr>
<td>(240,232) 訂正 240bit 検出</td>
<td>0.9500</td>
<td>0.9500</td>
<td>0.9501</td>
<td>0.9503</td>
</tr>
<tr>
<td>(255,247) 訂正 255bit 検出</td>
<td>0.9467</td>
<td>0.9467</td>
<td>0.9468</td>
<td>0.9470</td>
</tr>
</tbody>
</table>
4.4 まとめ

本章では，高信頼組込み自己テストのための耐故障テスト生成器として，巡回符号を利用した誤り訂正可能テスト生成器を提案し，その有効性について評価を行った。提案するテスト生成器は，誤り検出・訂正能力のない通常テスト生成器と誤り検出可能なテスト生成器である検出 TPG(EBIST) との比較において，高い期待動作確率が得られ，ロジック部の故障率に対してメモリ部の故障率が高いときは特に誤り訂正可能なテスト生成器が優位となることが分かった。また提案するテスト生成器と検出 TPG に対して，フィールドテストを想定したオンライン BIST での信頼度評価を行った。実験結果から，誤り訂正可能テスト生成器は，フィールドテストを想定した一時故障の発生を仮定した環境において，検出 TPG と比較して高い期待テスト実行確率が得られることがわかった。
第5章 誤り訂正を利用した耐故障応答圧縮器

5.1 はじめに

3章と4章では高信頼組込み自己テストのためのBIST回路のうち、テスト生成器について述べた。一方、高信頼なBIST回路を実現するためにはテスト生成器だけでなく、応答圧縮器についても考慮する必要がある。本章ではEBISTの誤り検出能力を応答圧縮器に適用した符号化応答圧縮器および、誤り訂正可能応答圧縮器について述べる。提案する応答圧縮器は符号化を利用し、応答圧縮器の誤り検出・訂正を行う。具体的にはテスト対象回路（CUT）の出力応答を巡回符号を利用して符号化し、同じ巡回符号に対応したMISRで応答圧縮器が無故障であれば、常にシグネチャが巡回符号の符号語となるよう構成している。これにより、応答圧縮器から得られたシグネチャが期待値と異なる符号語であればCUTの故障、非符号語であれば応答圧縮器の故障であると判定できる。特に誤り訂正可能応答圧縮器では、応答圧縮器のうち一部の故障による誤りを訂正可能であるため、無故障時と同様の期待通りのテスト実行結果が得られることが期待できる耐故障性を備えた耐故障応答圧縮器である。さらに、提案する2つの応答圧縮器はどちらもCUTの設計変更を必要としない。そのためCUTの出力応答中の不定値（ドントケアビット）を一意に0または1の固定値とするX-masking手法[34, 35]とも併用が可能である。

提案法では多くの応答圧縮器故障による誤りの検出または訂正が可能だが、すべての故障に対して検出または訂正ができるわけではない。つまり誤りの検出、訂正が可能となることによる応答圧縮器の信頼性の向上を面積増によって打ち消す
ことが考えられる。特に耐故障応答圧縮器は誤り訂正不可の一般の応答圧縮器よりも高い信頼性を備える必要がある。そのため本章では提案する誤り検出可能な応答圧縮器に対して、製造テストでのテスト実行時を想定、また耐故観応答圧縮器に対しては製造テスト、フィールドテストの2つのテスト実行時を想定し、それぞれに対する評価を行う。製造テストを想定した評価では応答圧縮器への永久故障発生時に、どれだけ多くの故障の誤りを応答圧縮器の故障として判定できるかを自己判別率によって評価する。またフィールドテストを想定した評価では応答圧縮器の動作中に、複数回の一時故障が発生した場合でも誤り訂正によって応答圧縮器が期待通り動作する確率によって評価する。このとき、一時故障の発生確率は面積に比例するものとする。

5.2 節では、誤り検出可能な応答圧縮器である符号化応答圧縮器について述べる。5.3 節では、誤り訂正可能な応答圧縮器について述べる。5.4 節では、提案する応答圧縮器の評価として、応答圧縮器に製造テストを想定した永久故障の発生を仮定、フィールドテストを想定した一時故障の発生をと、2つのケースにおいてそれぞれ自己判別率、期待テスト実行確率による評価を行う。また自己判別率については自己判別率の期待値を導出し、実測値と期待値の比較・考察も行う。

5.2 巡回符号を用いた誤り検出可能な応答圧縮器

5.2.1 符号化応答圧縮器

応答圧縮器の同時テスト、つまり BIST 回路を利用した CUT のテスト実行と同時に応答圧縮器のテストを可能とするためには、応答圧縮器から得られるシグネチャから CUT のテスト結果の合否だけでなく、応答圧縮器のテスト結果の合否も判定可能となる方法が考えられる。本章では、同時テストによって誤り検出可能な応答圧縮器である巡回符号を用いた符号化応答圧縮器 (符号化 RA) を提案する。さらに、一般の応答圧縮器が目的としているように、提案する応答圧縮器のアーキテクチャが原因で故障した CUT の誤りを見逃すことの無いような設計を目指す。
図 5.1: (7,4) 巡回符号を用いた符号化応答圧縮器

提案する符号化 RA は、巡回符号を利用した巡回符号化回路と巡回符号の符号化に対応した巡回符号 MISR を組み合わせて構成する。符号化回路は CUT の出力応答を符号化し、巡回符号の符号語を出力する。符号化に対応した MISR では符号化された CUT の出力応答を入力とし、シグネチャが常に符号化回路で利用した巡回符号の符号語となるよう圧縮する。

符号化 RA の例として、図 5.1 に (7,4) 巡回符号を用いた符号化 RA を示す。この符号化 RA は (7,4) 巡回符号化回路と (7,4) 巡回符号 MISR から構成される。なお n, k はそれぞれ符号化回路の出力ビット数 (MISR のサイズ)，CUT の外部出力ビット数に対応する (n > k)。

符号化 RA の動作を以下に示す。まず巡回符号 MISR のすべての FF を 0 で初期化する。これは制御信号 c1=c2=0 することで実行する。次に c1=c2=1 として CUT のテストを開始する。符号化回路によって符号化された CUT の出力応答を MISR によって圧縮し、シグネチャを観測する際は c1=0, c2=1 とすることで Scan-out からシグネチャを読み出す。読み出されたシグネチャは LSI テスタ内部でテスト実
行結果の判定が行われる。このとき、MISR ではシグネチャ観測後も MISR の内部状態を保持することが可能であるため、テスト実行途中で複数回シグネチャを観測し、観測後のシグネチャを保ったまま、再び $c_1 = c_2 = 1$ として CUT のテストを再開することが可能である。

符号化 RA では、CUT の故障の有無にかかわらずシグネチャは必ず符号化応答圧縮器で利用した巡回符号の符号語となり、さらに CUT が正常な場合は期待した符号語となるよう設計する（具体的な設計方法は 5.2.2 で説明する）。ここで図 5.1 に示す CUT が正常な場合と故障した場合を考える。表 5.1 は、正常時の CUT の出力系列 (2 列目), 故障 $j_0$ (信号線 $l_0$ の 0 縮退故障) を仮定した CUT の出力系列 (4 列目) を示しており、さらにそれぞれの出力系列に対する各サイクルでのシグネチャを示している (3, 5 列目)。表 5.1 よりどちらもすべての時刻でシグネチャは符号語であり、CUT 故障時は期待値と異なっていることがわかる。これは言い換えれば、シグネチャが非符号語となるのは符号化 RA の故障時のみであることになる。以上のことから、提案する符号化 RA のシグネチャと故障箇所に関して、次の 2 つの事実が成り立つ。

- シグネチャが期待値と一致する場合は、CUT・応答圧縮器共に正常である
- シグネチャが非符号語の場合は、応答圧縮器の故障である

一方でシグネチャが期待値と異なる符号語となるのは、CUT の故障時だけでなく符号化 RA 故障時にも起こりうる。故障した CUT を見逃さないため、このような場合は以下のような判定を行う。

- シグネチャが期待値と異なる符号語である場合は、CUT の故障と判定する。

このことから、符号化 RA の能力はシグネチャから CUT の故障か、応答圧縮器の故障かを明確に判定できる割合によって定義することができる。シグネチャが非符号語となるのは符号化 RA の故障時のみであることから、シグネチャが非符号

90
表 5.1: CUT の動作期待値と図 5.1 の符号化応答圧縮器のシグネチャ

<table>
<thead>
<tr>
<th>CUT の出力期待値</th>
<th>シグネチャ</th>
<th>CUT の出力期待値</th>
<th>シグネチャ</th>
</tr>
</thead>
<tbody>
<tr>
<td>正常時</td>
<td>故障</td>
<td>正常時</td>
<td>故障</td>
</tr>
<tr>
<td>1</td>
<td>0100</td>
<td>01000000</td>
<td>0100</td>
</tr>
<tr>
<td>2</td>
<td>0100</td>
<td>00000000</td>
<td>0100</td>
</tr>
<tr>
<td>3</td>
<td>1010</td>
<td>00101010</td>
<td>0100</td>
</tr>
<tr>
<td>4</td>
<td>1101</td>
<td>00000000</td>
<td>0101</td>
</tr>
<tr>
<td>5</td>
<td>1100</td>
<td>10111000</td>
<td>0100</td>
</tr>
<tr>
<td>6</td>
<td>1110</td>
<td>00000000</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>00101111</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>0001</td>
<td>00000000</td>
<td>0001</td>
</tr>
<tr>
<td>9</td>
<td>0010</td>
<td>11100101</td>
<td>0100</td>
</tr>
<tr>
<td>10</td>
<td>1111</td>
<td>01011110</td>
<td>1111</td>
</tr>
<tr>
<td>11</td>
<td>1111</td>
<td>11010000</td>
<td>0111</td>
</tr>
<tr>
<td>12</td>
<td>0101</td>
<td>10100001</td>
<td>0101</td>
</tr>
</tbody>
</table>

表 5.2: 図 5.1 の符号化応答圧縮器のシグネチャ

<table>
<thead>
<tr>
<th>シグネチャ</th>
<th>サイクル 12</th>
<th>サイクル 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>正常時</td>
<td>1010001</td>
<td>期待値</td>
</tr>
<tr>
<td>故障時</td>
<td>1101101</td>
<td>非符号語</td>
</tr>
<tr>
<td>f_1(信号線 l_1 の 0 繰返故障)</td>
<td>1101101</td>
<td>非符号語</td>
</tr>
<tr>
<td>f_2(信号線 l_2 の 0 繰返故障)</td>
<td>1010000</td>
<td>非符号語</td>
</tr>
<tr>
<td>f_3(信号線 l_3 の 1 繰返故障)</td>
<td>0101110</td>
<td>符号語</td>
</tr>
</tbody>
</table>

語となる符号化 RA の故障の割合を自己判別率 (SD: self-distinguishability) で表す。

$$ SD = \frac{|F_{CON}|}{|F_{ERA}|} \quad (5.1) $$

$|F_{ERA}|$ は符号化 RA の総故障数を、$|F_{CON}|$ は符号化 RA の故障時にシグネチャが非符号語となる故障数を表す。$|F_{CON}|$ が多いほど、$SD$ は大きな値となり、その結果、符号化 RA の故障であるにもかかわらず CUT の故障と判定される故障が減ることになる。

自己判別率 $SD$ はどのような $(n, k)$ 回巡符号に従って設計したかに依存する。特に符号語の冗長ビットのサイズ $n - k$ に大きく依存する。さらに一般に BIST 手法ではシグネチャを複数の異なるサイクルで観測するマルチサイクルシグネチャという手法がある [14, 36] が、自己判別率 $SD$ はシグネチャを何回観測するか、そしてどの時刻で観測するかにも大きく依存する。

表 5.2 は，図 5.1 に示す CUT と応答圧縮器が共に正常である場合と符号化 RA が単一繰返故障 $f_1, f_2, f_3$ のいずれかを持つ場合の，サイクル 12 とサイクル 7 で
得られるシグネチャを表している。例えば表 5.2 の 4 行目より，応答圧縮器に故障 \( f_1 \)（信号線 \( l_1 \) の 1 線間故障）を仮定した際のサイクル 12 で得られるシグネチャは 1101101（非符号語），サイクル 7 では 0110110（非符号語）であることがわかる。

表 5.2 より，故障 \( f_1 \) はサイクル 7 および 12 のどちらのシグネチャを観測しても非符号語であるため，応答圧縮器の故障と判定できる。一方で，故障 \( f_2 \) と \( f_3 \) はそれぞれサイクル 7 およびサイクル 12 のシグネチャが符号語となるため，どちらか一方のサイクルだけで観測した場合（シングルシグネチャ），どちらかの故障は応答圧縮器の故障と判定されない。そこで，サイクル 7 と 12 の両方のシグネチャを観測した場合（マルチサイクルシグネチャ）は，故障 \( f_2 \) と \( f_3 \) はどちらかのサイクルで非符号語となるため，応答圧縮器の故障と判定できる。このようにマルチサイクルシグネチャによって観測回数を増やせば，自己判別率 \( SD \) がより大きな値となることが期待できる。

以下の 5.2.2 節では符号化 RA の構成法を示し，5.2.3 節では符号語の冗長ビットの大きさ \( n - k \) とマルチサイクルシグネチャにおけるシグネチャの観測回数 \( l \) から得られる自己判別率 \( SD \) の期待値について議論する。

### 5.2.2 符号化応答圧縮器の構成法

\((n, k)\) 巡回符号を用いた符号化応答圧縮器（符号化 RA）の構成法を説明する。なお，以下の多項式は全て GF(2) 上の多項式であるとする。用いる \((n,k)\) 巡回符号の \( n - k \) 次の生成多項式を \( g(x) \) とする。なお \( g(x) \) は巡回符号の条件 [21, 22] を満たすため，

\[
(x^n - 1) \mod g(x) = 0 \tag{5.2}
\]

を満たす必要がある。さらに，\((n,k)\) 巡回符号のすべての符号語は，巡回符号の生成多項式 \( g(x) \) との積のみで表せる，すなわち \( g(x) \) で割り切れる。例えば，先ほど示した \((7, 4)\) 巡回符号の符号語 1101000 は多項式では \( 1 + x + x^3 \) と表現でき，巡回符号の生成多項式 \( g(x) = 1 + x + x^3 \) に対して，\((1 + x + x^3) \mod g(x) = 0 \) であることがわかる。
符号化 RA の符号化回路は、CUT の出力応答 \( O(x) \) に対する \( g(x) \) の剰余多項式 \( R(x) \) を求め（つまり \( R(x) = O(x) \mod g(x) \)）、出力 \( I(x) = x^{n-k}O(x) + R(x) \) を求める組合せ回路である。一方、(n, k) 回巡符号に対応する MISR のフィードバック多項式 \( \phi(x) \) は、式 (5.3) のように \( k \) 次の多項式 \( p(x) \) と符号化回路で用いた \( g(x) \) の積となるように構成する。

\[
\phi(x) = g(x)p(x)
\]  (5.3)

図 5.1 の符号化 RA は以下のように構成されている。まず (7, 4) 回巡符号 MISR を構成する。はじめに、(7, 4) 回巡符号の 3 次の生成多項式 \( g(x) \) を求める。\( g(x) \) は式 (5.2) を満たす必要があり、例えば、

\[
x^7 - 1 = (1 + x + x^2 + x^4)(1 + x + x^3)
\]  (5.4)

であるから

\[
g(x) = 1 + x + x^3 \quad \text{(5.5)}
\]

\[
p(x) = 1 + x + x^4 \quad \text{(5.6)}
\]

として符号化 RA を構成する。この例では \( p(x) \) として、式 (5.6) に示す 4 次の原始多項式を用いており、(7, 4) 回巡符号 MISR のフィードバック多項式 \( \phi(x) \) は式 (5.3) より式 (5.5), (5.6) を用いて式 (5.7) のようになる。

\[
\phi(x) = 1 + x^2 + x^3 + x^5 + x^7
\]  (5.7)

また (7, 4) 回巡符号化回路は式 (5.5) の生成多項式に対する剰余多項式を求める XOR ツリー (符号化回路の出力左 3 ビット) と、4 ビットの入力をそのまま出力する部分から構成されている。具体的には 2.4.4 節で説明した EBIST の誤り検出器を構成する情報ビット (符号化 RA の入力) から求めた冗長ビットと冗長ビットの XOR 演算を行う回路から、情報ビットから冗長ビットを求める回路と情報ビットをそのまま出力する回路を組合せる。

これらの手順で設計した符号化 RA は以下の定理を満たす。

93
定理 1:  

$(n,k)$ 巡回符号を用いた符号化応答圧縮器を考える。符号化応答圧縮器が正常でありかつ MISR の初期値 $S_0(x)$ が $(n,k)$ 巡回符号の符号語であるならば，時刻 $t$ のシグネチャー $S_t(x)$ は必ず $(n,k)$ 巡回符号の符号語となる。

証明: 時刻 $t$ での MISR の入力を高々 $n - 1$ 次多項式 $I_t(x)$ とすると，時刻 $t - 1$ と時刻 $t$ の MISR のシグネチャーの関係は，MISR の $n$ 次フィードバック多項式 $\phi(x)$ を用いて式 (5.8) のように表される [14]。

$$S_t(x) = (I_t(x) + xS_{t-1}(x)) \mod \phi(x) \quad (5.8)$$

ここで $I_t(x)$ は高々 $n - 1$ 次の多項式であるため，高々 $k - 1$ 次の多項式 $\alpha^I_t(x)$ と高々 $n - k - 1$ 次の多項式 $\epsilon^I_t(x)$ を用いて，以下のように表せる。

$$I_t(x) = g(x)\alpha^I_t(x) + \epsilon^I_t(x) \quad (5.9)$$

また $S_{t-1}(x)$ も高々 $n - 1$ 次の多項式であるため，高々 $k - 1$ 次の多項式 $\alpha^S_{t-1}(x)$ と高々 $n - k - 1$ 次の多項式 $\epsilon^S_{t-1}(x)$ を用いて，以下のように表せる。

$$S_{t-1}(x) = g(x)\alpha^S_{t-1}(x) + \epsilon^S_{t-1}(x) \quad (5.10)$$

なお $\epsilon^I_t(x)$, $\epsilon^S_{t-1}(x)$ はどちらも剰余多項式である。ここで $I_t(x)$ は符号化回路の出力のため符号語であるため，式 (5.11) のように $g(x)$ との積のみの式で表すことができる ($\epsilon^I_t(x) = 0$)。

$$I_t(x) = g(x)\alpha^I_t(x) \quad (5.11)$$

ここで，$S_{m-1}(x)$ を符号語と仮定すると，式 (5.12) のように $g(x)$ の積のみの式で表せる ($\epsilon^S_{m-1}(x) = 0$)。

$$S_{m-1}(x) = g(x)\alpha^S_{m-1}(x) \quad (5.12)$$

よって式 (5.8) より $S_m(x)$ は次のように表すことができ，

$$S_m(x) = g(x)((\alpha^I_m(x) + x\alpha^S_{m-1}(x)) \mod p(x)) \quad (5.13)$$

$g(x)$ の積の式であることから，$S_m(x)$ は符号語となる。
時刻0のシグネチャ$S_0(x)$は符号語であり，$S_{m-1}(x)$を符号語と仮定すると $S_m(x)$が符号語となることから，時刻$t$のシグネチャ$S_t(x)$は，数学的帰納法より任意の時刻$t$において符号語である。（証明終わり）

定理1は，符号化RAが正常であれば，CUTの故障の有無にかかわらず，シグネチャは必ず符号語となることを保証する．これはシグネチャが非符号語となるのは符号化RAの故障時のみであることを意味する．

5.2.3 自己判別率の期待値 $SD^E$

5.2.2で述べたように，自己判別率 $SD$は用いる巡回符号と，マルチサイクルシグネチャにおけるシグネチャの観測時刻と観測回数に大きく影響を受ける．ここでは，符号語の冗長ビットの大きさ $(n - k)$ とシグネチャの観測回数 $l$に着目し，これらの値に対して応答圧縮器故障時のシグネチャが符号語となる確率について議論し，その確率からシグネチャが非符号語となる故障の期待値，つまり自己判別率 $SD$の期待値を求める．$(n, k)$巡回符号に基づく符号化応答圧縮器（符号化RA）において，マルチサイクルシグネチャにおけるシグネチャの観測回数を$c$回とするとき，自己判別率 $SD$の期待値，$SD^E$は式 (5.14) のようになる．

$$SD^E = 1 - \left( \frac{2^k}{2^n} \right)^l$$ (5.14)

ここで式 (5.14) が成り立つ理由を説明する．自己判別率の期待値 $SD^E$ を求めるため，符号化RA故障時に時刻$t$のシグネチャが符号語となる確率 $P_t$を考える．符号化RAの故障は符号化回路の故障とMISRの故障の2通りが考えられる．

1. 符号化回路故障時に時刻$t$のシグネチャが符号語となる確率 $P_t^E$：$n$-bitの出力を持つ符号化回路故障時には出力が符号語か，非符号語かは不明であり，ここでは起こりうる$2^n$通りの出力が一様に起こると仮定する．

時刻$t$でのシグネチャは式 (5.3)，(5.8)，(5.9)，(5.10) より

$$S_t(x) = g(x)((\alpha^I_t(x) + x\alpha^S_{t-1}) \mod p(x))$$

$$+ \epsilon^I_t(x) + x\epsilon^S_{t-1}(x)$$ (5.15)
となる．式 (5.15) の $S_t(x)$ が符号語となるのは

$$ (e_t^I(x) + x e_{t-1}^S(x)) \mod g(x) = 0 \quad (5.16) $$

のときだけである．ここで $S_{t-1}(x)$ が符号語のときと非符号語のときの 2 通りに分けて考える．

(1) $S_{t-1}(x)$ が符号語である場合: $e_{t-1}^S(x) = 0$ のため，式 (5.16) は式 (5.17) のように表せる．

$$ e_t^I(x) \mod g(x) = 0 \quad (5.17) $$

$e_t^I(x)$ の次数は $g(x)$ の次数よりも小さいため，式 (5.17) を満たすのは $e_t^I(x) = 0$ のときのみである．$e_t^I(x) = 0$ となる $I_t(x)$ は式 (5.11) のように $g(x)$ の積ののみで表されることは意味し，$\alpha_t^I(x)$ と同数だけ存在する．$\alpha_t^I(x)$ は高々 $k - 1$ 次のため，このような $I_t(x)$ は $2^k$ 通り存在する．よって $S_{t-1}(x)$ が符号語かつ，$S_t(x)$ が符号語となる条件付き確率 $P_t^{\text{code}}$ は式 (5.18) で表せる．

$$ P_t^{\text{code}} = \frac{2^k}{2^n} \quad (5.18) $$

(2) $S_{t-1}(x)$ が非符号語である場合: このとき $S_{t-1}(x)$ は $e_{t-1}^S(x) \neq 0$ である 1 つの多項式となる．ここで $e_{t-1}^S(x)$ の次数が $n - k - 1$ 次の場合と $n - k - 2$ 次以下の場合を考える．

(a) $e_{t-1}^S(x)$ の次数が $n - k - 1$ 次

$x e_{t-1}^S(x)$ の次数は $g(x)$ の次数と等しいため，

$$ x e_{t-1}^S(x) = g(x) + e_{t-1}^S(x) \quad (5.19) $$

と表せる．ここで $e_{t-1}^S(x)$ は高々 $n - k - 1$ 次かつ $e_{t-1}^S(x) \neq 0$ である．よって式 (5.16) は式 (5.19) より

$$ (e_t^I(x) + g(x) + e_{t-1}^S(x)) \mod g(x) = 0 $$

$$ (e_t^I(x) + e_{t-1}^S(x)) \mod g(x) = 0 \quad (5.20) $$

96
と変形できる．ここで $e_t^l(x)$ も $e_t^S(x)$ も高々 $n - k - 1$ 次と，$g(x)$ の次数より低い
ため，式 (5.20) を満たすのは，

$$e_t^l(x) = e_t^{S^S}(x)$$  \hspace{1cm} (5.21)

のときである．ここで，$e_t^S(x)$ は $n - k - 1$ 次式のため，式 (5.21) を満たす $I_t(x)$
は $2^{k-1}$ 通り存在する．

(b) $e_t^S(x)$ の次数が高々 $n - k - 2$ 次

$e_t^l(x)$ も $x e_t^S(x)$ も高々 $n - k - 1$ 次式であり，$g(x)$ の次数より小さいため式 (5.16)
を満たすのは，

$$e_t^l(x) = x e_t^S(x)$$  \hspace{1cm} (5.22)

のときである．ここで，$e_t^S(x)$ は高々 $n - k - 2$ 次式のため，式 (5.22) を満たす
$I_t(x)$ は $2^{k-1}$ 通り存在する．このため，$S_t(x)$ が非符号語かつ，$S_t(x)$ が符号語と
なる条件付き確率 $P_t^{non}$ は式 (5.23) のとおりであり，

$$P_t^{non} = \frac{2^{k-1}}{2^n} + \frac{2^{k-1}}{2^n} = \frac{2^k}{2^n}$$  \hspace{1cm} (5.23)

よって符号化回路故障時に時刻 $t$ のシグネチャが符号語となる確率 $P_t^E$ は式 (5.25)
のように表される．

$$P_t^E = P_t^{non}(1 - P_t^E) + P_t^{code} P_t^E = \frac{2^k}{2^n}$$  \hspace{1cm} (5.24)

2. 巡回符号 MISR 故障時に時刻 $t$ のシグネチャが符号語となる確率 $P_t^M$：MISR の
故障時には，故障箇所によって MISR の振る舞いが大きく異なるため，ここでは
時刻 $t$ でのシグネチャ $S_t(x)$ が取り得るすべての場合が等しく起こりうると仮定す
る．$n$-bit の MISR で $S_t(x)$ が取り得るすべての場合は $2^n$ 通り，うち符号語は $2^k$ 通
りのため，$P_t^M$ は式 (5.25) のように表される．

$$P_t^M = \frac{2^k}{2^n}$$  \hspace{1cm} (5.25)

符号化回路または巡回符号 MISR の故障時に，観測回数 $l$ 回のマルチサイクル
シグネチャを行った場合，1 度でもシグネチャが非符号語となる確率はそれぞれ
$1 - (P_t^E)^l, 1 - (P_t^M)^l$ となる．自己判別率の期待値を求めた結果として，式(5.25)で表される巡回符号 MISR 故障時に時刻 $t$ のシグネチャが符号語となる確率 $P_t^M$ は，式(5.24)の符号化定数関数の確率 $P_t^E$ と同じ式になっているが，確率 $P_t^M$ を導くために用いた議論はそのまま確率 $P_t^E$ を求める議論としても適応可能である（ここでは確率 $P_t^E$ を求めるために，より厳密な議論を行っている）．そのため，この2つの確率が同じになることは自然である．ここで符号化 RA のうち，符号化回路の故障数を $|F_{ENC}|$，巡回符号 MISR の故障数を $|F_{MISR}|$ とすると，シグネチャが非符号語となる応答圧縮器の故障数 $|F_{CON}|$ の期待値は，

$$
|F_{ENC}| \cdot (1 - (P_t^E)^l) + |F_{MISR}| \cdot (1 - (P_t^M)^l) \\
= \left( 1 - \left( \frac{2^k}{2^n} \right)^l \right) \cdot |F_{ERA}| 
$$

(5.26)

であり，自己判別率 $SD$ の期待値 $SD^E$ は式(5.26)より，以下のように表される．

$$
SD^E = \left( 1 - \left( \frac{2^k}{2^n} \right)^l \right) \cdot \frac{1}{|F_{ERA}|} \\
= 1 - \left( \frac{2^k}{2^n} \right)^l 
$$

(5.27)

式(5.14)より，自己判別率 $SD$ は $n$ の大きい巡回符号を利用するとほど，またシグネチャ観測回数 $l$ を増やすほど大きな値となることがわかる．

### 5.3 巡回符号を用いた誤り訂正可能な応答圧縮器

#### 5.3.1 誤り訂正可能な応答圧縮器

5.2 節で示した符号化応答圧縮器(符号化 RA) では，テスト実行結果として得られるシグネチャから CUT の故障，または応答圧縮器の故障と判定ができる．一方，符号化 RA は応答圧縮器自身の故障検出のみのため，応答圧縮器故障時は別途，LSI テスタなどを用いて CUT をテストする必要がある．応答圧縮器に耐故障性を持たせることで，応答圧縮器の故障時も期待度通りのテストが実行できると考えられる．そこで耐故障性を持つ応答圧縮器として応答圧縮器中の MISR の内部状態の
誤り訂正，一部の故障による誤りのマスクが可能な誤り訂正可能応答圧縮器（訂正 RA）を提案する．提案する応答圧縮器は 5.2 節で示した符号化 RA の拡張であり，4.2 節で示した誤り訂正回路を符号化 RA に付加する．また 5.2 節で示した自己判定率を誤り訂正および誤りマスクが可能な訂正 RA に適用するために再定義および 4.3 節で示した期待動作確率を誤り RA に対して適用するための議論を行う．

訂正 RA では符号化 RA を構成する符号化回路（ENC），MISR に対して，MISR 内部状態の誤り検出・訂正を 3 つの付加回路，誤り検出器（DET），訂正ビット決定器（SYN），訂正実行器（COR）によって実現する．3.3 節の誤り訂正可能テスト生成器と同様に，DET，SYN，COR は符号化 RA と同じ巡回符号に対応するよう構成するため，（n，k）巡回符号を用いた訂正 RA を（n，k）訂正 RA と呼ぶ．仮に訂正 RA が無故障であれば，符号化 RA と同様に MISR は常に巡回符号の符号語を内部状態として保持する．（n，k）巡回符号の n − k は巡回符号の誤り訂正能力を表す．（n，k）巡回符号は n ビットの符号語で構成され，k ビットの情報ブロックと n − k ビットの誤り検出・訂正のための冗長ブロックからなる [21, 22]．

訂正 RA では，k は CUT の出力ビット数，n は MISR の大きさ（FF 数）を表す．

ここで，図 5.2 に示す訂正 RA において，MISR 内の左から 5 番目の FF の出力信号線が故障したケースを考える．この結果，図 5.3(a) に示すように故障値を含む“100111”（正常値“1001011”）が MISR から出力される．DET はこの誤りを検出し，誤りに対応する 3 ビットのシンドローム “011” を生成する．SYN はこのシンドロームを 7 ビットの誤り位置情報ベクトルに対応付け，誤りのあるビットには 1，それ以外は 0 とする誤り位置情報ベクトル “0000100” を生成する．このベクトルは 5 ビット目のみに誤りが存在することを意味する．COR は SYN の生成したベクトルを受けて，MISR の 5 ビット目の出力を XOR ゲートによって 1 から 0 へと反転させることで訂正を実行する．訂正された MISR の出力は MISR へ再び入力され，同じく MISR の入力ベクトルである CUT の出力応答 “0001” を符号化した “1010001” と合わせて内部状態を更新する．この結果 “0101110” が MISR の次状態として内部状態を更新する．この誤り訂正は訂正回路である DET，SYN，COR が
図 5.2: (7,4) 巡回符号を用いた誤り訂正可能な応答圧縮器
図 5.3: (7,4) 巡回符号を用いた誤り訂正可能な応答圧縮器の動作例

組合せ回路で構成されていることから，故障による誤りが出力されたサイクルと同じサイクルで実行される。

このように訂正 RA は誤りを即時訂正することで，期待値と一致するシグネチャが得られる確率を大きく向上させることができると考えられる。しかしながら，訂正 RA は MISR のすべての故障に対する故障による誤りを訂正できるわけではない。加えて，MISR 以外の ENC，DET，SYN，COR といった他の構成要素に対しても故障発生時について自己判別率などを利用して訂正 RA の能力を考慮する必
要がある。

以下ではDET, SYN, COR それぞれに単一故障が発生したケースを考える。

まずDETに故障が発生したケースについて考える。このとき，故障したDETは入力であるMISRの内部状態が符号語であるにもかかわらず，図5.3(b)に示すように“111”のような非ゼロのシンドロームを生成することが考えられる。または，入力が符号語であるにもかかわらず，ゼロベクトルをシンドロームとして生成するケースも考えられる。その結果，前者ではSYNとCORがMISRの内部状態が符号語であるにもかかわらず，必要な訂正を実行することで非符号語に訂正される。一方後者では，MISRの内部状態が非符号語であるが訂正が実行されず，MISRの内部状態が非符号語のままとなる。よって，DET故障時は非符号語のシグネチャが得られると考えられる。

次にSYNに故障が発生したケースについて考える。例えば図5.2の訂正RAでは図5.3(c)に示すように，故障したSYNがDETの出力がゼロベクトルであるにもかかわらず“0010101”という非ゼロの誤り位置情報ベクトルを生成するケースが想定される。しかしながら訂正RAでは，COR内部にセレクタというMISRの内部状態が符号語であればSYNの誤った出力ベクトルをMISRへ伝搬することを防ぐ回路を持つ。よってこのケースではDETの生成したシンドローム“000”によってセレクタのMUXが0に制御され，その結果MISRの内部状態に対して訂正が実行されない。結果として，MISRの内部状態はSYNの故障にもかかわらず，符号語のままとなる。

次にCORの故障について考える。このケースでは，CORはDETが誤りを検出していないにもかかわらず，MISRの内部状態に対して特定のビットを反転させる不要な訂正を実行することが考えられる。その結果，図5.3(d)の例では，MISRの左から4ビット目のFFの出力を反転させる不要な訂正を実行している。一方で，故障によっては図5.3(d)の例と異なり，訂正RAが期待通り動作するケースが存在する。例えばセレクタ内部のORゲートの故障時には，CORはSYNの出力ベクトルを参照し，訂正を実行するがCOR以外の構成要素が無故障のためSYNはゼロ

102
ベクトルを生成し、COR では訂正は実行されない。よって、COR 故障時のうち多くのケースでは非符号語もしくは期待値と一致するシグネチャが得られる。

結果として、訂正 RA の故障時は符号化 RA のように非符号語シグネチャを生成するだけでなく、訂正 RA が故障しているにもかかわらず期待値と一致するシグネチャを生成する。特に、後者の期待値シグネチャが得られるケースは、符号化 RA と異なって非常に多いと考えられる。これは訂正 RA の自己判別率の向上に効果がある。以下では、訂正 RA に符号化 RA の評価指標である自己判別率の適用について議論する。

5.3.2 誤り訂正可能な応答圧縮器における自己判別率

上記の通り、誤り訂正可能な応答圧縮器（訂正 RA）では、故障時でも期待値と一致するシグネチャを得られる。この訂正 RA の能力を自己判別率として表すために、訂正 RA 故障時に期待値と一致するシグネチャを得られる訂正 RA は無故障であるとみなす。したがって、訂正 RA の自己判別率は故障時に期待値と異なるシグネチャを生成した故障数のうち、応答圧縮器故障を正しく応答圧縮器の故障と判定できた故障数の割合で表すこととする。よって訂正 RA および符号化応答圧縮器（符号化 RA）の応答圧縮器の自己判別率は、

\[ SD = \frac{|F_{CON}|}{|F_{RA}| - |F_{EXP}|} \quad (5.28) \]

と表すことができる。\( F_{RA} \) は応答圧縮器の総故障数を表し、\( F_{CON} \) と \( F_{EXP} \) はそれぞれ応答圧縮器の故障と正しく判定出来た故障数および応答圧縮器故障時に期待値と一致するシグネチャを生成した故障数を表している。なお、符号化 RA では故障時に期待値と一致するシグネチャを生成することは困難なため、符号化 RA では \( F_{EXP} = \phi \) となる。そのため符号化 RA では式 (5.28) は式 (5.1) と一致する。自己判別率の大きさは、訂正 RA や符号化 RA を含む応答圧縮器の故障のうち、いかに多くの故障を正しく応答圧縮器の故障と判定できたかを表す。
5.4 節に示す実験では、訂正 RA および符号化 RA の自己判別率を式 (5.28) より求め、訂正 RA の有効性を明らかにする。

5.3.3 誤り訂正可能な応答圧縮器における期待動作確率

提案する誤り訂正可能な応答圧縮器（訂正 RA）では、誤り訂正によって故障時に期待値と一致するシグネチャを得られる。とくにフィールドテストにおいては一時故障の発生確率増加によってテスト実行結果が変化することが報告されている。訂正 RA では故障による誤りを検出し、その誤りを訂正することでテストを再実行及び BIST 回路の再初期化なしにテストを継続し、期待通りのテスト実行結果を得ることが可能となる。訂正 RA は 4 章で示した誤り訂正可能テスト生成器と同様に誤り訂正可能回数に制限がないため、一時故障が複数回発生する環境下であっても、その都度訂正することで複数回の一時故障に対応可能である。そのためこのような環境下では信頼度が高くなると考えられる。しかしながら、訂正回路の付加による面積オーバヘッドによって訂正能力の無い応答圧縮器よりも信頼性が逆に低下することが考えられる。よって、訂正 RA に対してフィールドテストを想定した一時故障仮定時の信頼度評価が必要である。

ここで、対象とする一時故障についての仮定を述べる。4 章の仮定と同様に、一時故障は 1 サイクル以内に高々 1 つ発生するものとする。一時故障による誤りは一時故障の発生と同一サイクルに、一時故障が発生した回路要素 (FF や論理ゲートなど) から出力されるものとする。また発生した一時故障は次の時刻には消えるものとする。さらに一時故障の発生確率は回路要素の面積に依存し、時刻には依存しないものとする。

しかしながら訂正 RA の信頼性は 4 章で示した期待動作確率 \( Q \) を利用し、式 (4.3) のように求めることはできない。これは訂正 RA の MISR 部からテスト実行結果として得られる MISR の内部状態は前時刻の MISR 内部状態だけでなく CUT の出力応答を符号化した符号化回路 (ENC) の出力を必要とするためである。図 5.2 に示す (7, 4) 巡回符号を用いた誤り訂正可能な応答圧縮器では、ENC の出力は誤り
訂正のための付加回路 (DET, SYN, COR) を経由することなく MISR の FF に取り込まれる構成となっていることがわかる。つまり ENC 内の XOR ゲートに発生した一時故障により XOR ゲートの出力値が反転した場合、誤りはそのまま FF に取り込まれることで MISR の内部状態は一時的に非符号語となる。次サイクルでは非符号語の内部状態は訂正回路によって訂正されることで期待値と一致する符号語に訂正される。このため、ENC に発生した一時故障はテスト最終サイクルの 1 時刻前までの故障による誤りのみシグネチャとして観測するまでに訂正することが可能となる。よって時刻 t で期待値と一致するシグネチャが得られる確率は、式 (4.3) は各時刻における期待動作確率 Q と ENC に一時故障が発生した一時故障による誤りを次のサイクルで訂正可能な確率 P_{EM4} を用いて，

\[ R_M(t) = Q^{t-1} \cdot (Q - P_{EM4}). \] (5.29)

と表すことができる。式 (5.29) において，Q は応答圧縮器ごとに異なるが，どのような応答圧縮器においても適用できる一般的な表現である。

次に訂正 RA の期待動作確率，Q_{EM} について議論する。ここで仮定する訂正 RA は c ビット (c ビットバースト誤り) 誤り訂正可能な巡回符号を適用したものとする。なお，以下の Q_{EC} の議論は c = 1 とすることで単一ビット誤り訂正可能な符号を利用した訂正 RA にも適用可能である。訂正 RA が期待通のテストパターンを生成できるケースは，以下の 4 つのケースに分けることができる。

- **Case 1:** 一時故障の発生はなく，無故障である。

- **Case 2:** MISR に一時故障が発生するケース。このケースではさらに 2 つのケースに分けることができる。それぞれのケースは，以下の通りである。(a)FF(メモリ部) に一時故障が発生，(b)XOR ゲート (ロジック部) に一時故障が発生。

- **Case 3:** SYN に一時故障が発生するケース。

- **Case 4:** ENC に一時故障が発生するケース。
これらのケースは完全に排他的であるため，それぞれのケースの確率を $P_{EM1}$，$P_{EM2}$，$P_{EM3}$ そして $P_{EM4}$ と表すこととする．そのため，正 TPG の期待動作確率，$Q_{EM}$ は以下のように表すことができる．

$$Q_{EM} = P_{EM1} + P_{EM2} + P_{EM3} + P_{EM4}.$$

(5.30)

以下ではそれぞれの確率 $P_{EM1}$，$P_{EM2}$，$P_{EM3}$，$P_{EM4}$ について考える．先の仮定の通り，一時故障の発生確率は面積のみに依存し，故障率とよぶ定数 $\lambda$ で表すとする．なお４章の仮定と同様に定数 $\lambda$ は FF（メモリ部），および論理ゲート（ロジック部）それぞれに対して $\lambda_M$，$\lambda_L$ とする．以下では４つのケースそれぞれの確率を求めめる．

- 確率 $P_{EM1}$: ケース 1 はすべての構成要素に故障が発生していないケースであるため，ケース 1 の発生確率は，以下のように表すことができる．

$$P_{EM1} = \exp(-\lambda \cdot (S_{ENC} + S_{MISR} + S_{DET} + S_{SYN} + S_{COR})).$$

(5.31)

ここで $S_{ENC}$，$S_{MISR}$，$S_{DET}$，$S_{SYN}$，$S_{COR}$ はそれぞれ ENC，MISR，DET，SYN，COR の面積を表す．4 章と同様に単位面積当たりのメモリ部の一時故障発生率を $\lambda_M$，単位面積当たりのロジックの一時故障発生率は $\lambda_L$ で表すとする．このとき式 (5.31) は，以下のように書き換えることができる．

$$P_{EM1} = \exp(-\lambda_L \cdot (S_{ENC} + S_{MISR(L)} + S_{DET} + S_{SYN} + S_{COR})$$

$$- \lambda_M \cdot S_{MISR(M)}).$$

(5.32)

ここで $S_{MISR(L)}$ と $S_{MISR(M)}$ はそれぞれ MISR のロジック部（XOR ゲート，内部状態出力のための AND ゲート）とメモリ部（FF）の面積を表している．よって $S_{MISR(L)}$ と $S_{MISR(M)}$ はそれぞれ以下のように書き表せる．

$$S_{MISR(L)} = N_{XOR} \cdot S_{XOR} + S_{SCAN};$$

$$S_{MISR(M)} = N \cdot S_{FF}.$$

(5.33)
なお，$N_{XOR}$，$S_{XOR}$，$S_{SCAN}$，$N$ そして $S_{FF}$ はそれぞれ MISR 内の XOR ゲート数，XOR ゲートの面積，内部状態出力のための回路面積，MISR の FF 数そして FF の面積を表している．なお $S_{MISR(M)}$ と $S_{MISR(L)}$ の関係式は以下の通りである．

\[ S_{MISR} = S_{MISR(L)} + S_{MISR(M)}. \]

式 (5.31) と (5.32) は MISR のみで構成される応答圧縮器（通常 RA）に対する

訂正 TPG の面積オーバヘッド，ENC，DET，SYN，COR によって期待動作

確率 $Q_{EM}$ が減少することを意味する．

• 確率 $P_{EM2}$：上記の通り，ケース 2 はさらに 2 つのケースに分けられる．1 つ

は連続する FF が $c$ 個故障，もう 1 つは XOR ゲートが 1 つだけ故障のケース

である．1 つの FF に一時故障が発生する確率は，$1 - \exp(-\lambda_M \cdot S_{FF})$ と表す

ことができ，さらに $N$ 個の FF のうち 1 つだけが故障するケースは $N_C = N$

と $N$ 通り考えられる．よって，いずれかの FF が故障する確率は $N \cdot (1 -

\exp(-\lambda_M \cdot S_{FF}))$ とする．同じく，連続する $c$ 個の FF が故障するケースは

$N_{C} = N$ と $N$ 通りであり，$N \cdot (1 - \exp(-\lambda_M \cdot S_{FF}))^c$ で表される．また連続

する $c$ ビット誤り訂正可能であれば，連続する $c - 1$ ビット誤りも訂正可能で

ある ($c \geq 2$)．そのため連続する $c$ 個以下の FF が故障し，その他の $N - c$

個の FF や MISR 中の XOR ゲート，DET，SYN，COR がすべて無故障であ

る確率は，以下のように表せる．

\[
\sum_{i=1}^{c} \exp(-\lambda_L \cdot (S_{ENC} + S_{MISR(L)} + S_{DET} + S_{SYN} + S_{COR}) - \lambda_M \cdot (S_{MISR(M)} - i \cdot S_{FF})) \\
\cdot N \cdot (1 - \exp(-\lambda_M \cdot i \cdot S_{FF})).
\]
また，後者のいずれか1つのXOR ゲートのみが故障する確率も同様に，

$$
\exp \left( -\lambda_L \cdot (S_{ENC} + S_{MISR(L)} + S_{DET} + S_{SYN} + S_{COR} - S_{XOR}) - \lambda_M \cdot S_{MISR(M)}) \right) \\
\cdot N_{XOR} \cdot (1 - \exp(-\lambda_L \cdot S_{XOR})),
$$

と表すことができる．よってケース2の確率 $P_{EM2}$ は以下のように表すことができる．

$$
P_{EM2} = \sum_{i=1}^{c} \exp(-\lambda_L \cdot (S_{ENC} + S_{MISR(L)} + S_{DET} + S_{SYN} + S_{COR}) \\
- \lambda_M \cdot (S_{MISR(M)} - i \cdot S_{FF})) \\
\cdot N \cdot (1 - \exp(-\lambda_M \cdot i \cdot S_{FF}))) \\
+ \exp(-\lambda_L \cdot (S_{ENC} + S_{MISR(L)} + S_{DET} + S_{SYN} + S_{COR} \\
- S_{XOR}) - \lambda_M \cdot S_{MISR(M)}) \\
\cdot N_{XOR} \cdot (1 - \exp(-\lambda_L \cdot S_{XOR})).
$$

式(5.34)は期待動作確率 $Q_{EM}$ が訂正RAの誤り訂正によって上昇する確率を表している．確率 $P_{EM2}$ によって期待動作確率は上昇するが符号化回路，訂正回路の面積オーバヘッド増大により，確率 $P_{EM1}$ は減少すると考えられる．

• 確率 $P_{EM3}$: ケース3は故障したSYN が誤った誤り位置情報ベクトルを生成するが，COR は誤って本来不要なMISR の出力の訂正を行わないことで，故障したSYN による $Q_{EM}$ への悪影響を与えない確率を表している．よってケース3は訂正TPGの期待動作確率 $Q_{EM}$ に加えることができる．$P_{EM3}$ はSYN のみが故障し，その他のENC，MISR，DET，COR は正常である確率であり，以下のように表すことができる．

$$
P_{EM3} = (1 - \exp(-\lambda_L \cdot S_{SYN})) \\
\cdot \exp(-\lambda_L \cdot (S_{ENC} + S_{MISR(L)} + S_{DET} + S_{COR}) \\
- \lambda_M \cdot S_{MISR(M)})).
$$

108
• 確率 $P_{EMA}$: ケース4は故障したENCがCUTの出力応答を正しく符号化できず非符号語を生成し、MISRに取り込まれた後、次のサイクルで訂正されることで期待通りのシグネチャが1サイクル後に得られる確率を表している。よってケース4はテスト実行時の最終サイクルに故障が発生するケースを除き、訂正RAの期待動作確率 $Q_{EM}$ に加えることができる。$P_{EMA}$はENCを構成する論理ゲートが故障し、その他のENC、MISR、DET、CORは正常である確率である。一方でENC内部には分歧が存在するため、確実にcビット誤りとなるケースは、ENCのXORツリーの最終段（ANDゲートの入力となるXORゲート）の $N-k$ 個のXORゲートもしくは $N$ 個のANDゲート故障時である。これらのケースを2つに分けて考えるとすると、1つはXORゲートがc個だけ故障、もう1つはANDゲートがc個だけ故障のケースである。1つのXORゲートに一時故障が発生する確率は、$1 - \exp(-\lambda_L \cdot S_{XOR})$と表すことができ、さらに $N-k$ 個のXORゲートのうち1つだけが故障するケースは $N-kC_1 = N-k$ と $N-k$ 通り考えられる。よって、いずれかのXORが故障する確率は $(N-k) \cdot (1 - \exp(-\lambda_L \cdot S_{XOR}))$ となる。同じく、連続するc個のFFが故障するケースは $N-k-cC_1 = N$ と $N-k-c$ 通りであり、$(N-k-c) \cdot (1 - \exp(-\lambda_L \cdot S_{XOR}))^c$ で表される。また連続するcビット誤り訂正可能であれば、連続するc−1ビット誤りも訂正可能である ($c \geq 2$)。そのため連続するc個以下のXORゲートが故障し、その他の $N-k-c$ 個のXORゲートやMISR中のXORゲート、ENC、DET、SYN、CORがすべて無故障である確率は、以下のように表せる。

$$
\sum_{i=1}^{c} \exp(-\lambda_L \cdot (S_{ENC} + S_{MISR(L)} + S_{DET} + S_{SYN} + S_{COR} - i \cdot S_{XOR}) - \lambda_M \cdot S_{MISR(M)} - (N-k-c) \cdot (1 - \exp(-\lambda_L \cdot i \cdot S_{XOR})).
$$

1つのANDゲートに一時故障が発生する確率は、$1 - \exp(-\lambda_L \cdot S_{AND})$ と表すことができ、さらに $N$ 個のANDゲートのうち1つだけが故障するケースは
$N C_1 = N$ と $N$ 通り考えられる。よって、いずれかの AND ゲートが故障する確率は $N \cdot (1 - \exp(-\lambda_L \cdot S_{AND}))$ となる。同じく、連続する $c$ 個の AND ゲートが故障するケースは $N C_1 = N$ と $N$ 通りであり、$N \cdot (1 - \exp(-\lambda_L \cdot S_{AND}))^c$ で表される。また連続する $c$ ビット誤り訂正可能であれば、連続する $c - 1$ ビット誤りも訂正可能である ( $c \geq 2$ )。そのため連続する $c$ 個以下の AND ゲートが故障し、その他の $N - c$ 個の AND ゲートや ENC のその他のゲート、MISR、DET、SYN、COR がすべて無故障である確率は、以下のように表せる。

$$
\sum_{i=1}^{c} \exp(-\lambda_L \cdot (S_{ENC} + S_{MISR(L)} + S_{DET} + S_{SYN} + S_{COR} - i \cdot S_{AND}) - \lambda_M \cdot S_{MISR(M)})
\cdot N \cdot (1 - \exp(-\lambda_L \cdot i \cdot S_{AND})).
$$

そのため連続する $c$ 個以下の XOR ゲートまたは AND ゲートだけが故障し、その他の ENC 内部の論理ゲート、MISR、DET、SYN、COR がすべて無故障である確率は、以下のように表せる。

$$
P_{EM4} = \sum_{i=1}^{c} \exp(-\lambda_L \cdot (S_{ENC} + S_{MISR(L)} + S_{DET} + S_{SYN} + S_{COR} - i \cdot S_{XOR}) - \lambda_M \cdot S_{MISR(M)})
\cdot (N - k - c) \cdot (1 - \exp(-\lambda_L \cdot i \cdot S_{XOR}))
+ \sum_{i=1}^{c} \exp(-\lambda_L \cdot (S_{ENC} + S_{MISR(L)} + S_{DET} + S_{SYN} + S_{COR} - i \cdot S_{AND}) - \lambda_M \cdot S_{MISR(M)})
\cdot N \cdot (1 - \exp(-\lambda_L \cdot i \cdot S_{AND})).
$$

(5.36)

確率 $P_{EM1}$ は期待動作確率 $Q_{EM}$ の ENC、DET、SYN、COR の面積オーバヘッドによる減少を表しているが、$P_{EM2}$、$P_{EM3}$ と $P_{EM4}$ は訂正 RA の誤り訂正機能によって $Q_{EM}$ を向上させることができる。つまり $P_{EM1}$ と $P_{EM2}、P_{EM3}、P_{EM4}$ は
ENC, DET, SYN, COR のような追加要素に対する相反する確率を表している。なお、この期待動作確率の解析は訳正 RA が DET または COR への一時故障では誤り訂正ができないこと、また ENC 内部のうち一部の XOR ゲート故障時のみ訂正可能であるとするという悲观的な解析であることは事実である。

5.4 実験

本節では提案する応答圧縮器を製造テスト、フィールドテストそれぞれの利用時を想定した評価を行う。提案する符号化応答圧縮器は誤り訂正可能な応答圧縮器では符号化を利用することで CUT の故障か応答圧縮器自身の故障かをシグネチャから判定することが可能であるが、一部の故障に対しては故障した CUT の誤り見逃しを防ぐため処理マルソスとなる CUT の故障と判定する。製造テストを仮定した評価では、応答圧縮器にて利用する巡回符号と歩留まりスをの少なさを表す自己判別率を求め、さらに論理合成によって求めた応答圧縮器の面積と合わせて考察を行う。さらに提案する 2 つの応答圧縮器をフィールドテストでの利用を想定した評価を行う。特に誤り訂正可能な応答圧縮器は誤り訂正によって期待通りのシグネチャを得られる確率が高くなると考えられるが、誤り訂正のために付加した面積オーバヘッドによってその確率の上昇を上回る信頼性の低下の可能性が考えられる。実験では、応答圧縮器への一時故障仮定時に期待通りのシグネチャを得られる期待テスト実行確率を利用して評価を行う。

5.4.1 節では製造テストでの利用時を想定し、提案する符号化応答圧縮器へ永久故障を仮定し、応答圧縮器自体の故障を正しく応答圧縮器の故障と判定可能である自己判別率を利用して評価し、利用する巡回符号と面積オーバヘッドおよび自己判別率の関係性について考察する。5.4.2 節では、符号化応答圧縮器においてテスト実行時に指定されたシグネチャの観測回数とテスト実行時間の関係について考察する。5.4.3 節では、誤り訂正可能な応答圧縮器の面積を論理合成によって求め、さらに製造テストでの利用時を想定した永久故障の仮定時の自己判別率について評価する。5.4.4 節では、誤り訂正可能な応答圧縮器および符号化応答圧縮器の
にフィールドテストでの利用時を想定し，一時故障仮定時に期待通りのシグネチャが得られる期待テスト実行確率について評価する．

5.4.1 符号化応答圧縮器の面積と自己判別率およびシグネチャ観測回数の関係

提案する符号化応答圧縮器（符号化 RA）に対して面積オーバーヘッド，自己判別率 SD，シグネチャ観測回数の関係を確認する実験を行った．ITC’99 ベンチマーク回路 [37] の b03，b04，b05，b07，b12 を CUT とし，これらの回路に対応する巡回符号を用いた符号化 RA を構成し，ベンチマーク回路の出力系列を符号化 RA に入力した．対象故障は符号化 RA 中の全信号線の単一 0/1 緩退故障とし，評価対象は，巡回符号を用いた符号化 RA の自己判別率と，MISR だけで構成される応答圧縮器を基準とした面積比率である．

(n,k) 巡回符号 MISR では，k 次の原始多項式を式 (5.3) の p(x) として利用した．なお (n,n – 1) 巡回符号はバリティ符号を表している．符号化 RA の面積を見積もるため，通常の応答圧縮器として k 次の原始多項式をフィードバック多項式として持つ MISR を合わせして設計し，基準とした．実験では符号化 RA と通常の MISR の面積は FF と 2 入力 XOR ゲートと 2 入力 AND ゲートの面積比を 7:3:2 として計算した値を示している．また実験においては，エイリアシングは発生していない．

同じ (n,k) 巡回符号を用いた符号化 RA であっても生成多項式 g(x) を最大で 4 種類用意し，それぞれに対して応答圧縮器を構成し，最も面積が小さいものを用いた．またシグネチャの観測回数 l はテスト終了時のサイクルを必ず含む 1, 2, 3, 4 とし，テスト終了時以外のサイクルはランダムに選択した．CUT のテストパターン数は b03，b04，b05，b07 は 1000，b12 では 100000 とした．また，各 CUT に対して 4 つの異なる初期値から LFSR によって生成されたテストパターン系列に対する出力応答を符号化 RA の入力として自己判別率を求めた．なお自己判別率は 5.2.2 節で示した式 (5.1) で計算した．

表 5.3 に巡回符号を用いた符号化 RA に対する実験結果を示す．各項目は利用し
たベンチマーク回路、出力ビット数 (#PO)、テストパターン数 (#TEST)、利用した巡回符号、符号化RAを通過のMISRと比較した面積比、そしてシグネチャ観測回数 \( l = 1, 2, 3, 4 \) それぞれの自己判別率 SD の平均値 (AVE)、最大値 (MAX)、期待値 (\( SDE \)) を表している。

実験結果より、5.2.3 節で求めた自己判別率の期待値のとおり、\( n \) の値の大きな巡回符号を利用するとほど、またシグネチャの観測回数 \( l \) を増やすほど、自己判別率が大きな値となっていることがわかる。例えば、表 5.3 より b03 の (48, 34) 巡回符号、b04 の (88, 74) 巡回符号、b05 の (84, 70) 巡回符号の実験結果を見ると、自己判別率はほど同じ値を取っていることから、符号化 RA の面積オーバヘッドに関係なく、自己判別率の期待値のとおり、CUT の出力ビット数 \( k \) に対する \( n \) の値や、シグネチャ観測回数 \( l \) で自己判別率が決まることがわかる。また、自己判別率の期待値と実験によって求められた値はかなり近いため、自己判別率の期待値を求める際の故障した符号化回路または MISR からは取り得る \( 2^n \) 通りすべての出力値が一様に起こるというのは仮定は妥当なものであることがわかる。

また、自己判別率の最大値 (MAX) を見ると、実験結果からは 4 回の観測を行えば、一般的な MISR に比べて 1.27〜1.70 倍の面積比で 100 % の自己判別率が得られる構成が可能であることがわかる。

自己判別率は符号化 RA の故障を正しく判別できる割合を表し、高い値であるほどより多くの故障を符号化応答圧縮器の故障として判別可能であることを意味する。つまり符号化 RA の故障によって無故障の CUT を故障と判別する確率を低減することが可能である。なお表 5.3 から、ERA で用いる符号語の長さ \( n \) が増えた場合、ERA の面積が増加するにも関わらず、自己判別率は増加していることがわかる。このことから、符号語の長さ \( n \) を増やすことで、ERA 内の故障の判別能力を向上できることがわかる。

符号化 RA は、与えられた面積オーバヘッドや自己判別率などの制約に対して、利用する巡回符号やシグネチャ観測回数の選択によって制約を満たす応答圧縮器を構成可能である。例えば、CUT が b07 のとき、シグネチャ観測回数 \( l = 1 \) かつ自己
判別率が99%以上で面積オーバヘッド最小化の制約が与えられた場合は、表 5.3 より (72, 57) 巡回符号を用いた符号化 RA を利用すればよいことがわかる。また CUT が b12 のとき、面積オーバヘッドは 1.3 倍以下かつ自己判別率が 97%以上の制約が与えられた場合は、表 5.3 よりシグネチャ観測回数 I は 3 回以上とし、(129, 127) 巡回符号を用いた符号化 RA であれば制約を満たすことができる。

また実験では、シグネチャの観測をテスト終了時のサイクルを必ず含むランダムに選んだサイクルで行っているが、同じ観測回数であれば自己判別率の値に大きな差はなく、表 5.3 に示す平均値 (AVE.) に近い値となった。このことから観測サイクルの選び方は自己判別率に大きな影響を与えないといえる。
表 5.3: 実験結果

<table>
<thead>
<tr>
<th>回路</th>
<th>#PO</th>
<th>#TEST</th>
<th>(n, k)</th>
<th>面積比</th>
<th>自己判別率 $SD$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$l = 1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AVE.</td>
</tr>
<tr>
<td>b03</td>
<td>34</td>
<td>1000</td>
<td>(35, 34)</td>
<td>1.31</td>
<td>44.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(36, 34)</td>
<td>1.37</td>
<td>73.87</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(42, 34)</td>
<td>1.67</td>
<td>99.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(48, 34)</td>
<td>1.93</td>
<td>99.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(54, 34)</td>
<td>2.43</td>
<td>99.41</td>
</tr>
<tr>
<td>b04</td>
<td>74</td>
<td>1000</td>
<td>(75, 74)</td>
<td>1.31</td>
<td>46.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(76, 74)</td>
<td>1.33</td>
<td>77.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(96, 74)</td>
<td>2.10</td>
<td>99.86</td>
</tr>
<tr>
<td>b05</td>
<td>70</td>
<td>1000</td>
<td>(71, 70)</td>
<td>1.31</td>
<td>52.53</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(72, 70)</td>
<td>1.34</td>
<td>76.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(75, 70)</td>
<td>1.37</td>
<td>98.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(80, 70)</td>
<td>2.10</td>
<td>99.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(84, 70)</td>
<td>2.08</td>
<td>99.93</td>
</tr>
<tr>
<td>b07</td>
<td>57</td>
<td>1000</td>
<td>(58, 57)</td>
<td>1.32</td>
<td>57.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(60, 57)</td>
<td>1.35</td>
<td>85.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(63, 57)</td>
<td>1.71</td>
<td>98.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(72, 57)</td>
<td>1.62</td>
<td>99.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(80, 57)</td>
<td>2.01</td>
<td>100.00</td>
</tr>
<tr>
<td>b12</td>
<td>127</td>
<td>100000</td>
<td>(128, 127)</td>
<td>1.31</td>
<td>47.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(129, 127)</td>
<td>1.32</td>
<td>74.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(160, 127)</td>
<td>1.90</td>
<td>99.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(180, 127)</td>
<td>2.10</td>
<td>99.64</td>
</tr>
</tbody>
</table>
5.4.2 シグネチャ観測回数とテスト実行時間に関する考察

さらにプレテスト可能な応答圧縮器である MISR-NOT と符号化応答圧縮器（符号化 RA）に対してテスト実行時間の比較を行った。CUT の出力ビット数 \( k \)，CUT のテスト実行サイクル数を \( T \) とおく。さらにテスト実行における MISR-NOT の MISR 部および CUT のサイクル時間を \( \lambda_C \)、シグネチャを読み出すテストの動作サイクル時間を \( \lambda_S \) とする。なお文献 [7] に示された結果に従い、MISR-NOT の初期化、およびテスト実行にはそれぞれ \( k \)，\( k+2 \) サイクルかかるとし、さらに MISR-NOT のテ尔斯実行結果と CUT のテスト実行結果の出力にはどちらも \( k \) サイクルかかることから MISR-NOT のテスト実行時間 \( T_{MN} \) は式 (5.37) で表すことができる。

\[
T_{MN} = (k + k + 2 + T) \cdot \lambda_C + (k + k) \cdot \lambda_S
\] (5.37)

また、符号化 RA の巡回符号 MISR の初期化には \( n \) サイクル、CUT のテスト実行時間の出力毎に \( n \) サイクルかかることから、テスト実行時間 \( T_{ERA} \) は式 (5.38) で表すことができる。

\[
T_{ERA} = (n + T) \cdot \lambda_C + l \cdot n \cdot \lambda_S
\] (5.38)

表 5.4 は、表 5.3 のうち、b03 と b12 に対して自己判別率が最大で 100%となる符号化 RA と MISR-NOT に対してテスト実行時間を比較した結果である。各項目は回路、CUT の出力ビット数 (#PO)，応答圧縮器の種類、通常の MISR を基準とした面積比、シグネチャ観測回数 (#I)，符号化 RA の巡回符号の \( n \) の大きさ、CUT のテスト実行サイクル数、MISR-NOT を基準としたテスト実行時間の比を表している。なお \( \alpha \) は MISR シグネチャ観測時におけるサイクル時間に対する CUT のテスト実行時のサイクル時間の比を表しており、\( \alpha = \frac{\lambda_S}{\lambda_C} \) である。ここでは \( \alpha=1 \)，10 それぞれの場合についてテスト実行時間の比較を行った。表 5.4 より符号化 RA ではシグネチャ観測回数 \( l=1 \) の場合は MISR-NOT と比較してすべての場合でテスト実行時間が短縮できていることがわかる。シグネチャの観測回数が 2 以上になると、符号化 RA のテスト実行時間は MISR-NOT よりも長くなるものの、テスト実行サイクル数 \( T \) が大きくなるほど（\( T=100000 \) の場合）、また \( \alpha \) が小さくなるほど
表 5.4：テスト実行時間比較

<table>
<thead>
<tr>
<th>回路</th>
<th>#PO((k))</th>
<th>応答圧縮器</th>
<th>面積比</th>
<th>#l</th>
<th>n</th>
<th>T</th>
<th>テスト時間比</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MISR-NOT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(\alpha=1)</td>
</tr>
<tr>
<td>b03</td>
<td>34</td>
<td></td>
<td>1.01</td>
<td>1</td>
<td>-</td>
<td>1000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>符号化応答圧縮器</td>
<td>2.31</td>
<td>1</td>
<td>54</td>
<td></td>
<td>0.973</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.87</td>
<td>2</td>
<td>48</td>
<td></td>
<td>1.005</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.44</td>
<td>3</td>
<td>42</td>
<td></td>
<td>1.026</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.34</td>
<td>4</td>
<td>36</td>
<td></td>
<td>1.037</td>
</tr>
<tr>
<td>b12</td>
<td>127</td>
<td>MISR-NOT</td>
<td>1.00</td>
<td>1</td>
<td>-</td>
<td>100000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>符号化応答圧縮器</td>
<td>1.99</td>
<td>1</td>
<td>180</td>
<td></td>
<td>0.999</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.79</td>
<td>2</td>
<td>160</td>
<td></td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.35</td>
<td>3</td>
<td>135</td>
<td></td>
<td>1.001</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.35</td>
<td>4</td>
<td>135</td>
<td></td>
<td>1.002</td>
</tr>
</tbody>
</table>

(\(\alpha=1\)の場合。これはCUTのテスト実行におけるサイクル時間がシグネチャの観測時におけるサイクル時間と等しいことを意味する。)。このテスト実行時間の増加の影響は小さくなることがわかる。

5.4.3 誤り訂正可能な応答圧縮器の自己判別率

5.4.1 節では、誤り検出可能な応答圧縮器（符号化 RA）の自己判別率に対する理論値を導出し、実測値と比較してほぼ同じ値となることを確認した。本節では、誤り訂正可能な応答圧縮器（訂正 RA）の自己判別率を実験的に導出する。

訂正 RA と符号化 RA の自己判別率を比較するために、ISCAS’89 ベンチマーク回路 [31] および ITC’99 ベンチマーク回路 [37] を CUT とした訂正 RA と符号化 RA を実装し、CUT の出力応答をこれらの応答圧縮器に入力した。訂正 RA と符号化 RA の大きさは CUT のスキャンチェーン数と同じで、対象障害は CUT、訂正 RA、縮退 RA の単一縮退障害とした。訂正 RA および符号化 RA は nangate ライブラリ [32] を利用して Design Compiler (Synopsys, Inc.) によって論理合成を行い、遅延制約として 6.6ns を与えた。表 5.5 に対象 CUT としたベンチマーク回路の情報を示す。それぞれ対象とした CUT、スキャンチェーン数（#ScanChain）、テストパターン数（#Test）、故障検出率（FC(%)）を表している。なお、CUT のすべての FF および外部入出力をパウンドリスキャン化している。この実験においては、すべての CUT の故障による誤りはスキャンチェーンに伝搬され、エイリアシング無しに検
表5.5: テスト対象回路および故障検出率

<table>
<thead>
<tr>
<th>CUT</th>
<th>#Scanchain</th>
<th>#Test</th>
<th>FC(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b07</td>
<td>15</td>
<td>1,500</td>
<td>93.486</td>
</tr>
<tr>
<td>b13</td>
<td>21</td>
<td>2,000</td>
<td>96.362</td>
</tr>
<tr>
<td>b04</td>
<td>31</td>
<td>3,000</td>
<td>95.546</td>
</tr>
<tr>
<td>s5378</td>
<td>63</td>
<td>4,000</td>
<td>95.503</td>
</tr>
<tr>
<td>b201</td>
<td>127</td>
<td>40,000</td>
<td>92.954</td>
</tr>
<tr>
<td>s38584</td>
<td>255</td>
<td>30,000</td>
<td>95.419</td>
</tr>
</tbody>
</table>

出された。

表5.6に訂正RAと符号化RAの面積および自己判別率を示す。表はそれぞれ応答圧縮器の種類と利用した巡回符号、CUT、応答圧縮器の構成要素であるENC、MISR、DET、SYN、CORそれぞれの面積、応答圧縮器の総面積、符号化RAの面積を基準とした面積比、応答圧縮器の総故障数（|F_RA|）、応答圧縮器の故障と正しく判定できた故障数（|F_CON|）、応答圧縮器故障時に期待値と一致するシグネチャが得られた故障数（|F_EXP|）、自己判別率（SD%）を表している。なお自己判別率は式(5.28)によって算出し、訂正RAと符号化RAのうち自己判別率の高い結果を太字で示している。

表5.6より訂正RAは符号化RAのおよそ1.6倍程度の面積であることがわかる。訂正RAと符号化RAで同じ巡回符号を利用しているためENCとMISRの面積は同じだが、構成要素として追加したDET、SYN、CORによって応答圧縮器中の総故障数は著しく増化していることがある。しかしながら、すべての応答圧縮器で訂正RAの方が高い自己判別率が得られていることがわかる。例えば表5.6の5列目に示す(26,21)巡回符号を用いた符号化RAでは99.03%の自己判別率が得られているが、4列目に示す同じ巡回符号を用いた訂正RAでは99.32%の自己判別率が得られている。表5.6より、訂正RAと符号化RAにおいて期待値と異なる符号語のシグネチャが得られた（応答圧縮器の故障だがCUTの故障と判定した）ケースはごくわずかであることがわかる。さらに、訂正RAでは応答圧縮器の総故障数のうち、約半数の故障で期待値と一致するシグネチャが得られていることがわかる。これは多くの訂正RA故障時において無故障時と同様に期待通りのシグネチャ生成できたことを意味する。結果として、提案する訂正RAは約半分の故障に
よる誤りを訂正もしくはマスク可能であり，さらに残りのほとんどどの故障も訂正RAの故障と正しく判定できたことがわかる．なお符号化RAでは$|F_{EXP}|$はすべての応答圧縮器で0となっている．
| 表5.6: 誤り訂正可能応答圧縮器と符号化応答圧縮器の面積および自己判別率 |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
|                 | CUT  | ENC | MISR | DET | SYN | COR | Total | 面積比 | $|F_{Ra}|$ | $|F_{Exp}|$ | $|F_{Con}|$ | $SD(\%)$ |
| (20,15) 訂正    | (20,15) 符号化 | b07  |  64.69 |  154.15 |  52.67 |  32.19 |  52.34 |  356.04 |  1.63 |  766  |  435  |  328  | **99.09** |
|                 |                  | b13  |  85.05 |  197.78 |  65.44 |  38.84 |  67.23 |  454.34 |  1.61 |  984  |  543  |  438  | **99.32** |
| (37,31) 訂正    | (37,31) 符号化 | b04  | 134.36 |  279.97 | 106.93 |  54.26 |  94.37 |  669.89 |  1.62 | 1464  |  784  |  675  | **99.26** |
| (70,63) 訂正    | (70,63) 符号化 | s5378|  272.69 |  515.91 |  205.88 |  89.64 | 175.23 |1259.35  | 1.60 | 2904  | 1498  | 1402  | **99.72** |
| (135,127) 訂正  | (135,127) 符号化| b20.1|  585.07 |  991.79 |  464.44 | 181.41 | 334.30 | 2557.01 | 1.62 | 6602  | 3220  | 3380  | **99.94** |
| (264,255) 訂正  | (264,255) 符号化| s38584| 1200.85 | 1936.88 |  949.62 | 299.25 | 657.75 | 5044.35 | 1.61 |13654  | 6391  | 7260  | **99.96** |
次に訂正 RA と符号化 RA から得られたシグネチャの詳細について説明する。対象とする訂正 RA および符号化 RA は (264,255) 巡回符号を利用した応答圧縮器である。表 5.7, 5.8 に、訂正 RA および符号化 RA から得られたシグネチャと故障箇所の関係を示す。表 5.7, 5.8 では、応答圧縮器の故障時に得られたシグネチャが期待値 (符号語), 期待値と異なる符号語, 非符号語それぞれの内訳を表している。表 5.7 は、利用した巡回符号と応答圧縮器, ENC と MISR それぞれにおいてシグネチャが期待値であった故障数 (期待値), シグネチャが期待値と異なる符号語であった故障数 (非符号語) を示している。なお、符号化 RA では表 5.6 に示したとおり、シグネチャが期待値となった故障数は無かった。表 5.8 も同様に訂正 RA および符号化 RA の DET , SYN , COR それぞれの故障時のシグネチャの内訳を示している。

表 5.7 より、非符号語シグネチャが得られた符号化 RA の MISR の 1104 個の故障は訂正 RA において期待値シグネチャとなった故障数は 1027 個、非符号語シグネチャとなった故障数は 77 個であることを示している。さらに符号化 RA において CUT の故障と判定する符号語シグネチャが得られた 4 個の故障は、すべて訂正 RA において非符号語シグネチャとなり、応答圧縮器の故障と正しく判定できたことがわかる。これらの結果から、訂正 RA での MISR の誤り訂正は有効であると言える。また ENC においても、符号化 RA において CUT の故障と判定する符号語シグネチャが得られた 3 個の故障は、すべて訂正 RA において非符号語シグネチャとなった。しかしながら符号化 RA では非符号語シグネチャとなった故障のうち 2 つの故障については、訂正 RA では期待値と異なる符号語が得られたことで CUT の故障と判定する結果となっている。これは訂正 RA の故障によって、不要な誤り訂正が実行された結果であると考えられる。
表 5.7: ENC と MISR の故障に対するシグネチャの内訳

<table>
<thead>
<tr>
<th></th>
<th>ENC</th>
<th>MISR</th>
</tr>
</thead>
<tbody>
<tr>
<td>(264,255)符号化</td>
<td></td>
<td></td>
</tr>
<tr>
<td>非符号語</td>
<td>3427</td>
<td>1104</td>
</tr>
<tr>
<td>符号語</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>(264,255)訂正</td>
<td>121</td>
<td>1027</td>
</tr>
<tr>
<td>期待値</td>
<td>3304</td>
<td>77</td>
</tr>
<tr>
<td>非符号語</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>符号語</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

表 5.8: DET, SYN, COR の故障に対するシグネチャの内訳

<table>
<thead>
<tr>
<th></th>
<th>DET</th>
<th>SYN</th>
<th>COR</th>
</tr>
</thead>
<tbody>
<tr>
<td>(264,255)訂正</td>
<td>121</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>期待値</td>
<td>3481</td>
<td>4030</td>
<td>1213</td>
</tr>
<tr>
<td>非符号語</td>
<td>1</td>
<td>0</td>
<td>391</td>
</tr>
<tr>
<td>符号語</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
表 5.8 より，SYN に仮定した 4030 個すべて故障において期待値と一致するシグネチャが得られたことがわかる。これは COR 内部のセレクタによる誤りマスクがうまく機能したこと意味する。また COR に仮定した 1604 個の故障の結果，1213 個の期待値シグネチャ，391 個の非符号語シグネチャが得られている。つまり，COR の故障時は多くの場合期待値と一致するシグネチャが得られ，訂正 RA を無故障とみなすことが可能となっている。さらに DET に仮定した 3482 個の故障の結果として，3481 個の非符号語シグネチャ，CUT の故障と判定する 1 個の符号語シグネチャが得られている。この結果より，訂正 RA では DET の故障時はほぼすべての故障を応答圧縮器の故障と正しく判定できたことがわかる。結果として，MISR の誤り訂正が可能な訂正 RA では，誤り検出のみ可能な符号化 RA と比較して高い自己判別率が得られた。つまり訂正 RA では訂正回路の付加により，符号化 RA よりも応答圧縮器故障時のテスト実行結果がより高信頼であり，高信頼な RA を実現できたといえる。

5.4.4 誤り訂正可能な応答圧縮器の期待動作確率

提案する誤り訂正可能な応答圧縮器（訂正 RA）の期待動作確率について考察する。式 (5.32) によって表される訂正 RA の面積オーバヘッドによって期待動作確率を低下させる要素と，式 (5.34)，(5.35)，(5.36) 向上させる要素の解析のために，訂正 RA を構成する各要素の面積を利用する。また一般的な誤り検出・訂正能力を持たない一般的応答圧縮器 (MISR) を設計し，同様に論理合成によって面積を求めた。面積の評価には Design Compiler (Synopsys, Inc.) を利用し，nangate ライブラリ [32] 用いた。与えた遅延制約はすべて 6.6ns となっている。

ここで一般 RA と符号化 RA の期待動作確率について考える。一般 RA は MISR のみで構成し，耐故障の要素は持ち合わせていないため，誤り検出および訂正はできない。一方符号化 RA は誤り検出可能な応答圧縮器である。これは巡回符号のシグネチャを生成する MISR と符号化回路 (ENC) で構成する。つまり，符号化 RA は訂正 RA のうち，DET，SYN と COR を省いた構成であると言える。そのた
表 5.9: 誤り訂正可能応答圧縮器と符号化応答圧縮器および一般 RA の面積

<table>
<thead>
<tr>
<th></th>
<th>ENC</th>
<th>MISR</th>
<th>DET</th>
<th>SYN</th>
<th>COR</th>
<th>Total</th>
<th>面積比</th>
</tr>
</thead>
<tbody>
<tr>
<td>(20,15) 訂正</td>
<td>64.69</td>
<td>154.15</td>
<td>52.67</td>
<td>32.19</td>
<td>52.34</td>
<td>356.04</td>
<td>3.23</td>
</tr>
<tr>
<td>(20,15) 符号化</td>
<td>64.69</td>
<td>154.15</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>218.84</td>
<td>1.98</td>
</tr>
<tr>
<td>15 ビット MISR</td>
<td>-</td>
<td>110.26</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>110.26</td>
<td>1.00</td>
</tr>
<tr>
<td>(26,21) 訂正</td>
<td>85.05</td>
<td>197.78</td>
<td>65.44</td>
<td>38.84</td>
<td>67.23</td>
<td>454.34</td>
<td>2.95</td>
</tr>
<tr>
<td>(26,21) 符号化</td>
<td>85.05</td>
<td>197.78</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>282.83</td>
<td>1.84</td>
</tr>
<tr>
<td>21 ビット MISR</td>
<td>-</td>
<td>153.89</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>153.89</td>
<td>1.00</td>
</tr>
<tr>
<td>(37,31) 訂正</td>
<td>134.36</td>
<td>279.97</td>
<td>106.93</td>
<td>54.26</td>
<td>94.37</td>
<td>669.89</td>
<td>2.97</td>
</tr>
<tr>
<td>(37,31) 符号化</td>
<td>134.36</td>
<td>279.97</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>414.33</td>
<td>1.84</td>
</tr>
<tr>
<td>31 ビット MISR</td>
<td>-</td>
<td>225.71</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>225.71</td>
<td>1.00</td>
</tr>
<tr>
<td>(70,63) 訂正</td>
<td>272.69</td>
<td>513.91</td>
<td>205.88</td>
<td>89.64</td>
<td>175.23</td>
<td>1259.35</td>
<td>2.78</td>
</tr>
<tr>
<td>(70,63) 符号化</td>
<td>272.69</td>
<td>513.91</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>786.60</td>
<td>1.74</td>
</tr>
<tr>
<td>63 ビット MISR</td>
<td>-</td>
<td>452.46</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>452.46</td>
<td>1.00</td>
</tr>
<tr>
<td>(135,127) 訂正</td>
<td>585.07</td>
<td>991.79</td>
<td>464.44</td>
<td>181.41</td>
<td>334.30</td>
<td>2557.01</td>
<td>2.79</td>
</tr>
<tr>
<td>(135,127) 符号化</td>
<td>585.07</td>
<td>991.79</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1576.86</td>
<td>1.72</td>
</tr>
<tr>
<td>127 ビット MISR</td>
<td>-</td>
<td>915.18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>915.18</td>
<td>1.00</td>
</tr>
<tr>
<td>(264,255) 訂正</td>
<td>1200.85</td>
<td>1936.88</td>
<td>949.62</td>
<td>299.25</td>
<td>657.75</td>
<td>5044.35</td>
<td>2.75</td>
</tr>
<tr>
<td>(264,255) 符号化</td>
<td>1200.85</td>
<td>1936.88</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3137.73</td>
<td>1.71</td>
</tr>
<tr>
<td>255 ビット MISR</td>
<td>-</td>
<td>1833.94</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1833.94</td>
<td>1.00</td>
</tr>
</tbody>
</table>

一般 RA と符号化 RA が期待動作可能なケースは訂正 TPG ではケース 1 に該当する．よって，一般 RA および符号化 RA の期待動作確率，$Q_{EMGen}$，$Q_{EMEnc}$ は，式 (5.32) より以下のように表すことができる．

$$Q_{EMGen} = \exp(-\lambda_L \cdot S_{MISR(L)} - \lambda_M \cdot S_{MISR(M)}),$$  (5.39)

$$Q_{EMEB} = \exp(-\lambda_L \cdot (S_{ENC} + S_{MISR(L)}) - \lambda_M \cdot S_{MISR(M)}).$$  (5.40)

なお，符号化 RA の方が応答圧縮器としての面積が大きいため，明らかに $Q_{EMGen} > Q_{EMEB}$ が成り立つ．

表 5.9 に 15, 21, 31, 37, 63, 127, 255 本のスキャンチェーンを持つ CUT に対して適用する訂正 RA，符号化応答圧縮器 (符号化 RA)，MISR の各応答圧縮器の面積を示す．なお，訂正 RA および符号化 RA の構成および面積は表 5.6 で求めたものと同じである．表 5.9 より訂正 RA の面積は MISR のおよそ 2.75-3.23 倍と訂正回路の付加による面積オーバーヘッドが大きいことがわかる．以下ではこれらの面積を元に各応答圧縮器の期待動作確率について考察する．

表 4.4 に示す論理合成によって求めた面積より各 RA の期待動作確率を式 (5.30)，
表 5.10: $\lambda_L = 10^{-7}$ における表 5.9 に示す応答圧縮器の期待動作確率

<table>
<thead>
<tr>
<th>RA</th>
<th>$\lambda_M$</th>
<th>$\lambda_L$</th>
<th>$2 \times \lambda_L$</th>
<th>$5 \times \lambda_L$</th>
<th>$10 \times \lambda_L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(20,15)</td>
<td>(λM)</td>
<td>0.999984</td>
<td>0.999984</td>
<td>0.999984</td>
<td>0.999984</td>
</tr>
<tr>
<td></td>
<td>(20,15) 符号化</td>
<td>0.999978</td>
<td>0.999967</td>
<td>0.999933</td>
<td>0.999878</td>
</tr>
<tr>
<td></td>
<td>15 ビット MISR</td>
<td>0.999989</td>
<td>0.999978</td>
<td>0.999944</td>
<td>0.999888</td>
</tr>
<tr>
<td>(26,21)</td>
<td>(λM)</td>
<td>0.999979</td>
<td>0.999979</td>
<td>0.999979</td>
<td>0.999979</td>
</tr>
<tr>
<td></td>
<td>(26,21) 符号化</td>
<td>0.999972</td>
<td>0.999957</td>
<td>0.999914</td>
<td>0.999841</td>
</tr>
<tr>
<td></td>
<td>21 ビット MISR</td>
<td>0.999985</td>
<td>0.999970</td>
<td>0.999927</td>
<td>0.999854</td>
</tr>
<tr>
<td>(37,31)</td>
<td>(λM)</td>
<td>0.999967</td>
<td>0.999967</td>
<td>0.999967</td>
<td>0.999967</td>
</tr>
<tr>
<td></td>
<td>(37,31) 符号化</td>
<td>0.999959</td>
<td>0.999938</td>
<td>0.999876</td>
<td>0.999773</td>
</tr>
<tr>
<td></td>
<td>31 ビット MISR</td>
<td>0.999977</td>
<td>0.999957</td>
<td>0.999895</td>
<td>0.999791</td>
</tr>
<tr>
<td>(70,63)</td>
<td>(λM)</td>
<td>0.999936</td>
<td>0.999936</td>
<td>0.999936</td>
<td>0.999936</td>
</tr>
<tr>
<td></td>
<td>(70,63) 符号化</td>
<td>0.999921</td>
<td>0.999882</td>
<td>0.999765</td>
<td>0.999570</td>
</tr>
<tr>
<td></td>
<td>63 ビット MISR</td>
<td>0.999955</td>
<td>0.999916</td>
<td>0.999798</td>
<td>0.999603</td>
</tr>
<tr>
<td>(135,127)</td>
<td>(λM)</td>
<td>0.999862</td>
<td>0.999862</td>
<td>0.999862</td>
<td>0.999862</td>
</tr>
<tr>
<td></td>
<td>(135,127) 符号化</td>
<td>0.999842</td>
<td>0.999767</td>
<td>0.999541</td>
<td>0.999164</td>
</tr>
<tr>
<td></td>
<td>127 ビット MISR</td>
<td>0.999908</td>
<td>0.999833</td>
<td>0.999607</td>
<td>0.999230</td>
</tr>
<tr>
<td>(264,255)</td>
<td>(λM)</td>
<td>0.999718</td>
<td>0.999718</td>
<td>0.999718</td>
<td>0.999719</td>
</tr>
<tr>
<td></td>
<td>(264,255) 符号化</td>
<td>0.999686</td>
<td>0.999539</td>
<td>0.999097</td>
<td>0.998360</td>
</tr>
<tr>
<td></td>
<td>255 ビット MISR</td>
<td>0.999817</td>
<td>0.999669</td>
<td>0.999227</td>
<td>0.998491</td>
</tr>
</tbody>
</table>

(5.39), (5.40) より求めたものを表 5.10 に示す．なお式 (5.30) のうち，$P_{EM2}$ について（5.34）に対して $c = 1$ とすることで求められる．一時故障の発生確率はロジック部の発生確率 $\lambda_L$ を $10^{-7}$ として計算した．表 5.10 の各行はメモリ部の一時故障発生確率 $\lambda_M$ を $10^{-7}$ ( $\lambda_L$ と等しい)，$2 \times 10^{-7}$，$5 \times 10^{-7}$，$10^{-6}$ としたときの期待動作確率を表している．表 5.10 において 3 つの RA のうち，最も高い期待動作確率のものを太字で示している．

表 5.10 より，$\lambda_M$ が増加すると，一般 RA(MISR) と符号化 RA の期待動作確率は低下するが，訂正 RA は高い期待動作確率を保っていることがわかる．これは $\lambda_M$ の増加が，式 (5.39) と (5.40) において $\lambda_M \cdot S_{LFSR(M)}$ の確率を大きく低下させることで，一般 RA と符号化 RA の MISR の信頼性を大きく低下させたことが原因と考えられる．一方，訂正 RA では $\lambda_M$ の増加が同じく式 (5.32) に示すように MISR の信頼性を低下させたが，式 (5.34)，(5.35)，(5.36) に示すようにメモリ部をまったく含まない ENC，DET，SYN，COR の回路が低下した MISR の信頼性を補っている．特に式 (5.34) のうち，$N \cdot (1 - \exp(-\lambda_M \cdot S_{FF}))$ の数値の増加が低下した MISR の信頼性を補っている．
また表5.10より，RAの出力ピット値$N$が増加すると期待動作確率は低下していることがわかる。これは$N$の増加により，RAの各要素の面積が増加し，式(5.32)，(5.39)，(5.40)に示すように期待動作確率が低下したためである。例えば，式(5.33)，(5.39)では$N$の増加によって$S_{MISR(M)}$が大きくなることがわかるが，それによって$Q_{EMGen}$は減少することとなる。表より訂正RAではほとんどのケースにおいて最も高い期待動作確率となっている。これは$Q_{EM}$において$N$の増加によって，訂正RAにおいてMISRの誤りを訂正できるケースの増加によって確率$P_{EM2}$が増加したためである。一方，メモリ部の故障率$\lambda_M=\lambda_L=10^{-7}$の時はRAの出力ピット数にかかわらず訂正RAの期待動作確率がわずかに一般RAよりも小さくなっていことがわかる。これは訂正RAを構成する各要素の面積の増加によって訂正RAが無故障である確率$P_{EM1}$が大きく低下し，確率$P_{EM2}$によるMISRの誤り訂正ができることによる$Q_{EM}$の増加分では補えなくなったためである。

5.4.5 オンラインBISTにおける耐故障応答圧縮器の信頼度評価

提案する誤り訂正可能応答圧縮器(訂正RA)は，4章で提案した耐故障テスト生成器と同様に，リアルタイムシステムのためのオンラインテストのように，テスト実行時間に対する制約が大きいケースにおいてとくに一時故障の耐性が効果を発揮すると考えている。これは訂正RAは一時故障による誤りを訂正することで故障発生時も期待通りのテストを実行できることからテスト再実行が不要となり，テスト実行時間が一定となるためである。ここで対象とするオンラインテストの構造は，文献[33]を対象とする。この構造では，テスト対象回路であるリアルタイムアプリケーションの通常動作と通常動作の間の短い時間にテストを実行する。このような短い時間のテストを繰り返し実行することで，求められる故障検出率を達成する。

一般に応答圧縮器はテスト実行中のすべてのサイクルでテスト実行結果であるシグネチャを観測せず，テスト終了後またはテスト実行途中の決められたサイクルでのみ観測する。そのためオンラインBISTにおける応答圧縮器の信頼度評価は，
<table>
<thead>
<tr>
<th>RA \ α_M</th>
<th>λ_L</th>
<th>2 × λ_L</th>
<th>5 × λ_L</th>
<th>10 × λ_L</th>
</tr>
</thead>
<tbody>
<tr>
<td>(20,15) 訂正</td>
<td>0.9926</td>
<td>0.9926</td>
<td>0.9926</td>
<td>0.9926</td>
</tr>
<tr>
<td>(20,15) 符号化</td>
<td>0.9902</td>
<td>0.9852</td>
<td>0.9705</td>
<td>0.9464</td>
</tr>
<tr>
<td>15 ビット MISR</td>
<td>0.9951</td>
<td>0.9901</td>
<td>0.9752</td>
<td>0.9510</td>
</tr>
<tr>
<td>(26,21) 訂正</td>
<td>0.9905</td>
<td>0.9905</td>
<td>0.9905</td>
<td>0.9905</td>
</tr>
<tr>
<td>(26,21) 符号化</td>
<td>0.9874</td>
<td>0.9809</td>
<td>0.9619</td>
<td>0.9310</td>
</tr>
<tr>
<td>21 ビット MISR</td>
<td>0.9931</td>
<td>0.9866</td>
<td>0.9675</td>
<td>0.9364</td>
</tr>
<tr>
<td>(37,31) 訂正</td>
<td>0.9852</td>
<td>0.9852</td>
<td>0.9852</td>
<td>0.9852</td>
</tr>
<tr>
<td>(37,31) 符号化</td>
<td>0.9815</td>
<td>0.9724</td>
<td>0.9457</td>
<td>0.9027</td>
</tr>
<tr>
<td>31 ビット MISR</td>
<td>0.9899</td>
<td>0.9807</td>
<td>0.9537</td>
<td>0.9104</td>
</tr>
<tr>
<td>(70,63) 訂正</td>
<td>0.9714</td>
<td>0.9714</td>
<td>0.9714</td>
<td>0.9714</td>
</tr>
<tr>
<td>(70,63) 符号化</td>
<td>0.9652</td>
<td>0.9484</td>
<td>0.8996</td>
<td>0.8239</td>
</tr>
<tr>
<td>63 ビット MISR</td>
<td>0.9798</td>
<td>0.9628</td>
<td>0.9133</td>
<td>0.8363</td>
</tr>
<tr>
<td>(135,127) 訂正</td>
<td>0.9397</td>
<td>0.9397</td>
<td>0.9397</td>
<td>0.9397</td>
</tr>
<tr>
<td>(135,127) 符号化</td>
<td>0.9315</td>
<td>0.9004</td>
<td>0.8133</td>
<td>0.6863</td>
</tr>
<tr>
<td>127 ビット MISR</td>
<td>0.9597</td>
<td>0.9276</td>
<td>0.8378</td>
<td>0.7071</td>
</tr>
<tr>
<td>(264,255) 訂正</td>
<td>0.8808</td>
<td>0.8808</td>
<td>0.8809</td>
<td>0.8812</td>
</tr>
<tr>
<td>(264,255) 符号化</td>
<td>0.8683</td>
<td>0.8126</td>
<td>0.6659</td>
<td>0.4779</td>
</tr>
<tr>
<td>255 ビット MISR</td>
<td>0.9208</td>
<td>0.8617</td>
<td>0.7061</td>
<td>0.5067</td>
</tr>
</tbody>
</table>

与えられたテスト実行時間終了時に期待通りのシグネチャを生成できる確率で表示することができる。評価においては、シグネチャの観測回数は L サイクルかけて行うテスト終了時の 1 回のみとする。このとき記号 RA が期待通りのシグネチャを生成可能な期待テスト実行確率は、式 (5.29) で表すことができる。一方、一般的な応答圧縮器（一般 RA、MISR）や符号化応答圧縮器（符号化 RA）の期待テスト実行確率は、式 (4.3) より、式 (5.39)、(5.40) を用いてそれぞれ式 (5.41)、(5.42) のように表せる。

\[
R_{MISR}(L) = Q^L. \quad (5.41)
\]

\[
R_{EMISR}(L) = Q^L. \quad (5.42)
\]

つまり各応答圧縮器の期待テスト実行確率は、表 5.10 に示した期待動作確率 Q でほぼ決まると言える。

訂正 RA、符号化 RA および一般 RA の期待テスト実行確率をそれぞれ式 (4.3)、(5.41)、(5.42) で表したものを表 5.11 に示す。表 5.11 よりロジック部とメモリ部の故障が等しい（\(λ_L = λ_M\)）ときを除いて、訂正 RA の方が最も高い期待テスト実行確率が得られていることがわかる。これは訂正 RA に付加した訂正回路の面積
増による信頼性低下を訂正による信頼性の上昇分で回復できなかったためと考えられる。一方で、メモリ部の故障率が高いときは、MISR自体の信頼性低下が大きくなり、訂正機構を持たない一般RAの信頼性が低下することで期待テスト実行確率の差が大きくなったと考えられる。これらのことから、訂正RAが備える誤り訂正機構はリアルタイムシステムのためのオンラインBISTにおいて有効であるといえる。

5.5 まとめ

本章では、BIST回路の応答圧縮器に対して故障時に応答圧縮器の故障を正しく判定可能な符号化応答圧縮器を提案し、さらにその発展系である誤り訂正可能な応答圧縮器を提案した。提案する応答圧縮器では、巡回符号による符号化を利用し、CUTの誤り検出をできる限り防ぐという応答圧縮器本来の機能を保ちつつ、CUT故障を応答圧縮器の故障と判定することはない。一方、応答圧縮器の故障時はほとんどの故障を応答圧縮器の故障と判定可能だが、一部の故障はCUT故障時と同じテスト実行結果が得られるため安全側の判定としてCUTの故障と判定する。この応答圧縮器の故障を正しく判定可能な割合を自己判別率と定義し、実験によって符号化応答圧縮器、誤り訂正可能応答圧縮器では99%以上の高い自己判別率を得ることが分かった。また符号化応答圧縮器では、採用する巡回符号、シグネチャ観測回数の選択によって面積オーバヘッド、自己判別率などの設計制約を柔軟に満たすことができることが分かった。さらに誤り訂正可能な応答圧縮器では、製造テストを仮定した永久故障発生時に誤り訂正によって応答圧縮器故障時も無故障時と同じ結果が得られるケースが大きく増加し、さらに符号化応答圧縮器より高い自己判別率が得られることが分かった。またフィールドテストを想定した一時故障発生時も、誤り訂正能力の無い一般の応答圧縮器よりも無故障時と同様に期待通りのシグネチャを生成可能な確率である高い期待テスト実行確率を得ることができた。今後の課題として、提案する誤り訂正可能な応答圧縮器で利用する誤り訂正能力と期待動作確率の関係性の考察があげられる。
第6章 結論

大規模化，複雑化の進む大規模集積回路 (VLSI) の信頼性を確保するために故障の有無を調べるテストは必須であるが，テストにかかるコストの増加が課題となっている．組込み自己テストはテストコストの削減に有効であるほか，テストの機構を BIST 回路として組み込むため製造テスト，出荷後のフィールドテストの両方に適用可能である．しかしながら，被テスト LSI と同様に BIST 回路も大規模化し，無故障であるとの仮定は非現実的となっている．BIST 回路の故障は被テスト LSI のテスト実行結果を誤る原因となり，コストや信頼性の面で不利益を被る．とくに近年は，微細化によってソフトエラーなどの一時故障の発生率増加によって，フィールドテストにおける影響が懸念される．本論文では，BIST 回路の故障時も BIST 回路自身の故障と正しく検出できる，または BIST 回路故障時も無故障時と同様に期待通りのテスト実行が可能な高信頼組込み自己テストを提唱し，高信頼組込み自己テストのための耐故障 BIST 回路の提案を行った．

第3章では，テスト生成器を誤り検出単位であるユニットと誤り検出時の切替機構を組み合わせた耐故障テスト生成器を提案した．提案するテスト生成器は動的マスクの考え方を利用しており，誤り検出時にスペアユニットと故障ユニットを置き換える待機冗長型テスト生成器（冗長 TPG）と，故障ユニットの出力をほかの正常なユニットの出力で代替する縮退利用型テスト生成器（縮退 TPG）の2つがある．提案するテスト生成器はテストパターンを生成するユニットの切替によって故障時も期待通り動作することが可能なため，信頼性が高まると考えられるが，切替器などの面積オーバヘッドによりテスト生成器の故障する確率が高まることも考えられる．これらのテスト生成器は故障箇所を分離するよう再構成を行うことが
ら、おもに永久故障の検出を目的とする製造テストでの利用が考えられる。提案するテスト生成器の評価として、面積当たりの故障発生率を定義し、LFSRのみで構成した通常のテスト生成器（通常 TPG）との信頼性比較を行った。その結果、提案するテスト生成器の面積は冗長 TPG では通常 TPG の 1.98 倍から 2.50 倍、縮退 TPG では 1.77 倍から 1.90 倍となったが、すべてのテスト生成器において通常 TPG を上回る信頼性を得ることができた。また冗長 TPG と無故障時の縮退 TPG では通常 TPG と遜色ない故障検出率が得られ、故障による縮退動作時の縮退 TPG でも約 50% のテストパターン数増加で再構成による縮退動作前と同等の故障検出率が得られることがわかった。よって、製造テストにおいては冗長 TPG および縮退 TPG が有効であるといえる。

第 4 章では、通常 TPG である LFSR に対して誤り訂正を行う回路を付加した誤り訂正可能なテスト生成器（訂正 TPG）を提案した。このテスト生成器は静的マスクの考え方を利用しており、誤り訂正可能な巡回符号を利用して構成し、故障による誤りを訂正することで誤りを含んだテストパターンを出力せず、故障時もテストを中断せず継続できる。これは誤り検出時はテストパターンを生成するユニット切り替え後にテストを再実行する冗長 TPG と縮退 TPG にはない利点である。また冗長 TPG と縮退 TPG では誤り検出時に行う再構成の実行可能回数に上限があるが、訂正 TPG では何度も誤り訂正が可能である。そのため訂正 TPG での誤り訂正能力が高いほど誤り訂正によって多くの故障に対して期待通りのテストパターンを生成でき考えられるが、誤り訂正回路の肥大化によって故障発生確率が増加し、期待通りのテストパターンを生成可能である期待動作確率が低下すると考えられる。誤り訂正能力に対する訂正 TPG の実験では 1 ピット訂正可能な訂正 TPG が最も期待動作確率が高いことがわかった。1 ピット訂正可能な訂正 TPG は通常 TPG のおよそ 2.12 倍から 2.36 倍の面積となるが、通常 TPG よりも高い期待動作確率となることを確認した。また、特にロジック部（論理ゲートなど）に対してメモリ部（FF）の一時故障率が高いほど、訂正 TPG が EBIST を大きく上回る期待動作確率を得られることを確認した。またフィールドテストを想定したオンライン
イン BIST 環境において、従来法である誤り検出可能なテスト生成器（EBIST）との比較の結果、訂正 TPG の方が与えられた時間内にテストを完了できる確率である期待テスト実行確率が得られることができた。EBIST に優位と考えられるテスト再実行に利用可能なテスト実行時間の余裕が大きいときも同様の結果だったため、フィールドテストにおいては訂正 TPG が有効であるといえる。

第 5 章では、従来法である誤り検出可能なテスト生成器を BIST 回路の応答圧縮器に適用した符号化応答圧縮器（符号化 RA）、さらに符号化 RA を発展させた誤り訂正可能な工巡符号を用いた誤り訂正可能な応答圧縮器（訂正 RA）を提案した。提案する応答圧縮器は、CUT の故障と応答圧縮器の故障をテスト実行結果として得られるシグネチャから区別することが可能であり、また CUT の故障を応答圧縮器の故障と判定することはない。しかしながら、一部の応答圧縮器の故障は安全側の判定により CUT の故障と判定する。このような故障数の少なさを自己判別率によって表すと定義し、実験の結果符号化 RA、訂正 RA ともにほぼ 99% 以上の自己判別率を得られた。とくに訂正 RA では符号化応答圧縮器に誤り訂正回路を付加した面積オーバヘッドおよび総故障数の大きな応答圧縮器であるが、訂正 RA 内の約半数の故障仮定時は無故障時と同じシグネチャが得られ、さらに残りの故障に対しても高い自己判別率を得られている。また符号化 RA に対しては利用する巡回符号およびシグネチャ観測回数を変えることで設計制約として与えられる面積オーバヘッド、自己判別率などの制約を柔軟に満たすことが可能であることがわかった。またフィールドテストを想定したオンライン BIST 環境においても、誤り検出可能な応答圧縮器である符号化 RA や誤り検出・訂正能力を持たない一般の RA の比較の結果、訂正 RA が最も高い期待テスト実行確率が得られることがわかった。

本論文で示した研究成果は、テストコスト削減に有効な組込み自己テストによる被テスト LSI のテスト実行結果をより高信頼とすることができる。今後ますます微細化の進行と共に一時故障の発生率増加が考えられる中、フィールドテストの重要性がさらに高まることが予想される。提案する BIST 回路はこのような想定にも
十分対応可能であると考えられる。

本論文で示した研究成果から、提案する BIST 回路の利用によるコスト面（実行歩留まりの向上）や信頼性（市場不良率の低減）の点における効果を求めることがさらなる研究の発展としてあげられる。またユーザが安心して利用できるサービスを提供可能な LSI システムの実現のために、提案する BIST 回路を含めた LSI 全体で消費電力、面積、パフォーマンスなどの設計制約を満たす LSI の統合設計手法の提案が今後の研究の発展としてあげられる。

132
謝辞

本研究の全過程において，研究を進める上での懇切なご指導，研究会発表に向けての論文執筆，発表準備に際してのご助言を頂きましたこと，また研究発表者助成をはじめとする研究を進める上での有益な情報を提供して頂きましたことを，主指導教員の井上智生教授に深く感謝の意を表します．

本研究の遂行にあたり，国際会議発表準備に際しての投稿論文査読やご助言，ご指導を頂きました，副指導教員の北村俊明教授に深く感謝の意を表します．

本研究の全過程において，研究活動の初歩から研究を進める上での懇切なご指導，研究を進める上での課題解決のための議論，研究会発表に向けての論文執筆，発表準備に際してのご助言を頂きました，副指導教員の市原英行准教授に深く感謝の意を表します．

本研究の遂行にあたり，有益な議論および貴重なご助言，ご指導を頂きました，若林真一教授に深く感謝の意を表します．

本研究の遂行にあたり，研究を進める上での懇切なご指導，研究会発表に向けての論文執筆，発表準備に際してのご助言を頂きました，岩垣剛助教に深く感謝の意を表します．

本研究の遂行にあたり，研究を進める上での懇切なご指導，研究会発表準備に際してのご助言を頂きました，広島市立大学院所属で現呉高等専門学校所属の吉川祐樹講師に深く感謝の意を表します．

最後に，本論文にあたり，同じ研究チームに所属し，日頃より活発な議論に参加し，数多くの有益なご協力，ご助言を頂きました，上田健司君，大元将一君，佐々木孝洋君，富田孝道君をはじめとする広島市立大学院情報科学研究科および広島
市立大学情報科学部のコンピュータデザイン研究室の諸氏、およびコンピュータデザイン研究室卒業生で現マツダ株式会社所属の志水昂君に感謝いたします。
参考文献


136
関連論文一覧

[論文雑誌]

[国際会議]


[その他査読付き論文 (国際ワークショップ・国内会議)]

RTL and High-Level Testing (WRTLT’12), pp.3.3.1–3.3.4, November, 2012(ショート).

3. 志水昂，深澤祐樹，吉川祐樹，市原英行，井上智生，“組込み自己テストにおける救済可能テスト生成器”，第9回情報科学技術フォーラム(FIT)，2010年9月。


[研究会発表]


2. 深澤祐樹, 市原英行, 井上智生, “テスト可能な応答圧縮器におけるマルチサイクルシングネチャの効果について”, 第65回 FTC 研究会, 2011年7月。


5. 深澤祐樹, 吉川祐樹, 市原英行, 井上智生, “組込み自己テストにおける応答圧縮器の自己診断能力について”, 第60回 FTC 研究会, 2009年1月。