LFSコード化を用いた入力ビット幅圧縮法

日大生産工(院)　○宇佐美　龍哉
日大生産工　細川　利典

１．はじめに

近年の大規模集積回路(LSI)のテストにおいては，従来挙げられてきた故障の検出率に関する問題だけでなく，そのテストにかかるコストの削減が重要である．LSI のテスト時のコストに直接影響してくる要素として，一般的にテスト実行時間，テストデータ，テスト時にかかる消費電力などが挙げられる．しかし最近では，前述のコストの問題だけでなく，LSI の高レベルな微細化，集積化により，内部コアのテストに必要な入出力数が，外部入出力の数では不足する事がある．そのことにより満足いくようなテストが行えない事態が発生している．

本稿ではこのような問題のうち，回路の入力ビット幅を減少させ，テストを効率よく行えるようにすることを考える．具体的には LFSR の論理を用いて入力するテスト系列を圧縮，符号化し，回路内に付加した展開器によって，本来回路に対して印加したいテスト系列を入力する手法を提案し，さらにより小さなビット幅での入力を可能にするため、改善する手法について検討する．

２．入力ビット幅の問題と解決方法

LSIの設計手法の一つにシステム・オン・チップ(SoC)がある．SoCとは，1つの半導体チップ上に必要とされる機能を集積する集積回路・LSIの設計手法である．それぞれの機能は，コアという単位でチップ上に配置されており，図1のようなイメージで表される．

SoC のテストにおいては，各コアの入力に，単体テストのパターンを入力し，コアごとの出力結果を評価する事でテストを行う．しかし，コアの入出力が SoC 自体の外部入出力の数を上回ってしまうと，完全なテストが困難となってしまう．図 1 の例だと，B，D，E のコアが直接外部入出力による単体テストが困難である．

解決する方法に，コアの入出力にスキャンフリップフロップを配置し，任意の入力を与えるようにするバウンダリースキャンという手法がある．

しかし，この方法には回路規模が膨大になるほどテスト実行時間が増大してしまうなどの問題がある．

本稿で提案する LFS コーディング(1)は，ビット幅削減のための難易度が回路規模に依存せず，テストパターン中のケアビットにのみ依存し，ケアビット数＋20 程度の入力で符号化が可能である．

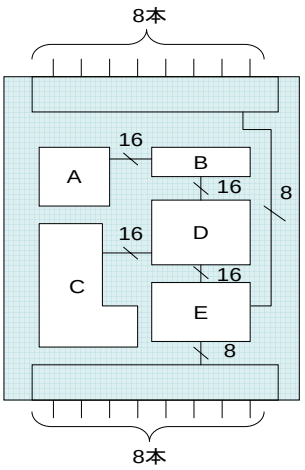

図 1:SoC の構成例

３．LFS 符号化の論理

本章では，提案する LFS 符号化の論理について述べる．LFSとはLinear Feedback Shiftの略で，線形フィードバックシフトレジスタの持つ論理を利用している．具体的に述べると，N ビットの LFSR に M ビットのシフトチェインが接続されている回路モデルを考える．　図 2 には LFS コーディングに使用するLFSR論理モデルの例を示す．この図では 4 ビット LFSR と 6 ビットシフトチェインで構成されている．

An input Space Compaction Method Using LFS Coding

Tatsuya USAMI,　Toshinori HOSOKAWA

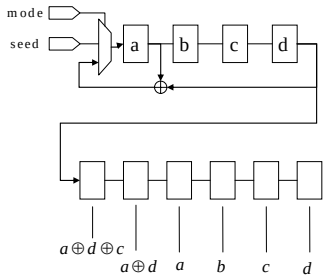

図 2:LFSR 論理モデル

シフトチェインの下に記されている式は，LFSR の初期値を(a, b, c, d)とした，6 時刻後の各シフトチェインの論理関数である．この論理関数は，符号化するべきパターンの連立方程式である．したがってこの連立方程式を解くことで， LFS コーディングのための符号化が可能となる．

$$\left.\begin{aligned} a \oplus d \oplus c &= 0 \\ a &= 0 \\ b &= 1 \\ d &= 1 \end{aligned}\right\} \cdots\cdots\cdots (1)$$

$$\because a, b, c, d \in \{0,1\}$$

例を挙げると，ある回路に対してのテストパターン(0, X, 0, 1, X, 1)の LFS 符号を求めるためには連立方程式(1)を解くことで求められる．これにより求められる a, b, c, d, すなわち符号化されたテストパターンは(a, b, c, d)＝(0, 1, 1, 1)である．

また，LFSR はそのレジスタ数，シフトチェインが可変長ではない場合．シフトする時刻数がシフトチェイン数と等しい場合．といった 2 点を満たすとき，組み合わせ回路により等価な論理を持つ回路を作成することが可能である．このことにより生成した図 2 と等価な論理を持つ回路を図 3 に示す．この回路においては，6 ビットの入力が 4 ビットに削減されている．

本手法では，図.3 のような回路を構成し，展開器として使用することで，回路の入力ビット幅調整の技術として利用する．

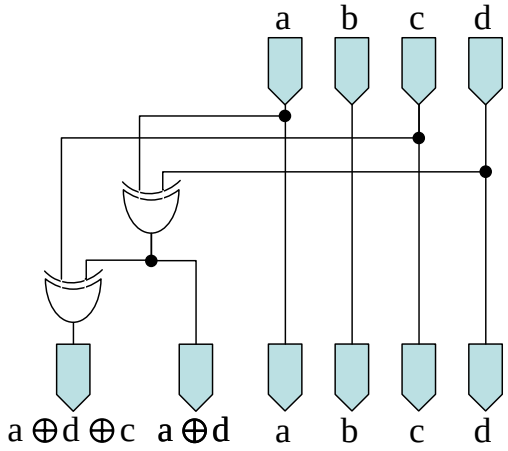

図. 3:LFS 符号復号器

４．LFS 符号化のアルゴリズム

４．１．LFS 符号化実施の上での問題点

本章では，実際に LFS 符号化の実施方法について述べる．LFS 符号化は 2 章でも述べた通り，入力ビット幅をテストのために削減するためのものである．そのために連立方程式を解くわけであるが，その入力パターンと復号器の構成によっては連立方程式が解なしとなる場合が存在する．

例として，図 2，図 3 の論理によって，テストパターン(1, X, 0, X, 1, 1)を LFS 符号化することを考える．このパターンの符号化には連立方程式(2)の解を求めればよい．

$$\left.\begin{aligned} a \oplus d \oplus c &= 1 \\ a &= 0 \\ b &= 1 \\ d &= 1 \end{aligned}\right\} \cdots\cdots\cdots (2)$$

$$\because a, b, c, d \in \{0,1\}$$

しかしながら，この連立方程式の解を求めようとすると，解は存在しないことが解る．このような場合，復号器の入力ビット幅を大きくする，複号器の接続を変更してやるなどの必要がある．しかしながら，SoC 中のコアのテストを考えた場合，SoC の外部入力数は限られているため，外部入力数を超えない事が前提である．

４．２．コード化可能性判定

LFS 符号化は，復号器の構成を変化させる事により，符号化可能かどうかが変化する．また符号器の構成を変化させる事なく，復号器の出力ビットの割り当てを変更させることのみで，これを実現する事ができる．

そのためには，あるテストパターンが，どの復号器の接続パターンにより符号化可能かを調べる必要がある

図 4 に符号化可能判定処理の処理手順を示す．

```
T' = X-Identification (T)
Collect Decoder Assignment (P,T')
FOR {All Test Pattern}
    FOR {All Decoder Assignment}
        ti = SAT-Solver (C,T')
        If {SAT}
            Add Decoder Assignment Information (SAT , ti)
        If {UNSAT}
            Add Decoder Assignment Information (UNSAT , NULL)
    ENDFOR
ENDFOR

T : Test Vector (Original)
T' : Test Vector (X-included)
P : Generating polynomial (For LFS Coding)
ti : Encoded Test Pattern
```

図 4:コード化可能判定処理

まず，テスト生成により得られたテストパターン T に対して X 抽出処理を行い， ドントケアを含むテストパターン T'を得る．

T'のパターンの X のあるビットの位置から，生成多項式 P に従って，復号器とその出力ビットの接続関係の候補を決定する．

パターン毎に全ての接続情報において，SAT-Solver を用いて符号化可能かどうかを調べ，接続情報表に追加する．

５．展開器接続問題

５．１．展開器の出力ビットの割当

前章で述べたとおり，LFSR モデルのシフトチェインの入力側のビットの復号論理の検討が難しい．そこで困難な要因の一つである，元となるテストパターンの中で，複雑な論理関数が割り当てられる部分に着目する．

一般に論理回路のテストパターンは 0 もしくは 1 により表記されているが，故障を検出する際に必要でない論理値というものが存在する．この値をドントケア(3)と呼び，X で表される．ドントケアは通常テスト生成時にランダムな値により 0 か 1 どちらかに設定されている．このドントケアが複雑な論理関数の部分に多く割り当てられれば，符号化時の失敗終了する可能性が減少すると考えられる．

本章ではテストパターンに対するビット割り当てを変更し，

パターン中にドントケアの多い入力が複雑な論理関数のビットに割り当てられるようにする手法を提案する．

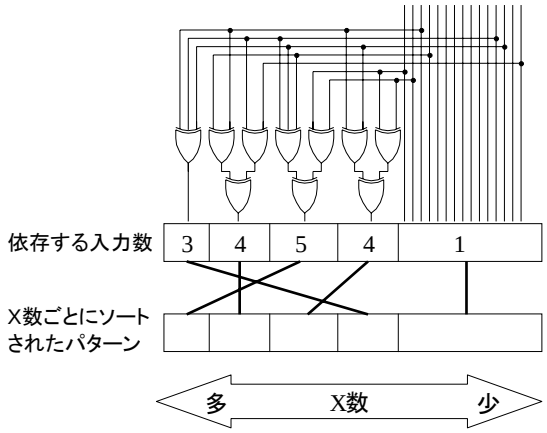

図.5:ビット割当変更イメージ

５．２．展開器接続問題の定式化

いくつか展開器の接続情報を用意するわけだが，出力ビットの接続を変化させる場合には，マルチプレクサを利用するため，あまり非効率に接続の種類を増加させてしまうと，マルチプレクサの入力が増加しすぎてしまうため，効率的に圧縮する事ができない．

そのため，用意した接続情報を入力として最も少ない接続の組み合わせ数で実現できるようにしたい．これを，最小被覆問題として定式化すると．

入力：展開器接続情報行列
（$T=t_{ij}$, $t_{ij}\in\{1,0\}$）
出力：各テストパターンを復号する時の展開器と，テスト対象回路の接続情報．
最小化：接続の組み合わせの数

この最小被覆問題を解くことにより，最小の組み合わせを得る事ができる．

展開器接続情報行列の例を以下に示す．

表：展開器接続行列

		接続1	接続2	・・・	接続j
テストパターン	t1	1	0	・・・	1
	t2	1	1	・・・	0
	・	・	・	・	・
	・	・	・	・	・
	・	・	・	・	・
	ti	0	1	・・・	0

＊行列中の 1＝符号化可能，0＝符号化不可能

具体的な処理は，展開器接続情報行列 T の列ベクトルの和が 1 になるようなできるだけ少ない数の列ベクトルを選択する．

５．３．展開器の構成

本章では，提案手法において用いる展開器の構成例を示す．

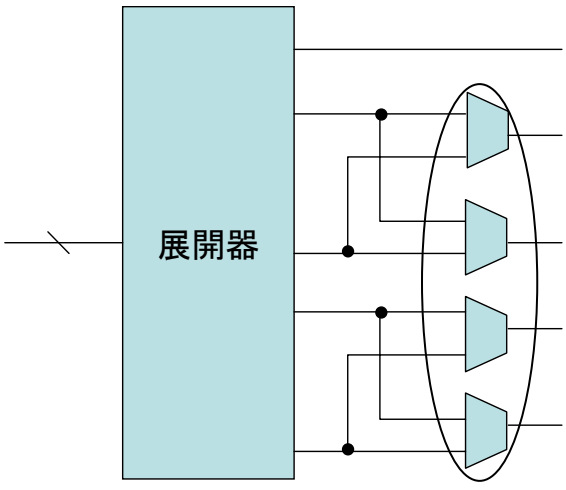

図 6：展開器アーキテクチャの例

前節で提案した展開器接続問題は，図 6 の丸部分で囲まれたマルチプレクサの数を最小化するものであると考える事ができる．

６．おわりに

本稿では LFS 符号化を用いて，テスト時の入力ビット幅を削減することを，最小被覆問題として定式化することができた．

今後，最小被覆問題の解を効率的に得る事ができる方法と，元となる展開器接続情報行列の構成法について検討していく．

参考文献

(1) 欧州特許番号：EP0481097 “Method and apparatus for testing a VLSI device” International Business Machines Corporation, 2003.

(2) S.Kajihara and K.miyase , “On Identifying Don’t Care Inputs of Test Patterns for Combinational Circuits , “ Proc. Of Int’l Conf. of CAD , pp. 364-369, 2001.

(3) K.Miyase et al. , ”Don’t-Care Identification on Specific Bits of Test Patterns”, ICCD, 2002.

(4) 藤原秀雄 著 “ディジタルシステムの設計とテスト” 工学図書 刊 2004 年 ISBN-4769204590