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

2007－ＳＬＤＭ－132（14）
2007／11／21

種々の部分積加算構造に対応したテスト容易な乗算器の設計手法

鬼頭　信貴†　　高木　直史†

† 名古屋大学大学院 情報科学研究科　〒464-8603 名古屋市千種区不老町
E-mail: †{nkito,ntakagi}@takagi.i.is.nagoya-u.ac.jp

あらまし　テスト容易な乗算器の設計手法を示す．乗算器の部分積加算部を3種類のブロックを組み合わせて設計する．ブロックの組み合わせかたにより様々な部分積加算構造を実現できる．部分積加算部と共に乗算器の構成に必要な部分積生成部と最終加算部の構成についても示す．これらを組み合わせることで，様々な部分積加算構造をもつ乗算器を実現できる．提案手法による乗算器は単一セル機能故障の仮定のもとで演算のビット長にかかわりなく14個のテストパターンでテストできる．提案手法で設計した乗算器にはテストのために6本の追加の外部入力が必要となる．
キーワード　テスト容易化設計，Cテスト可能，ツリー型乗算器，配列型乗算器

A design method for easily testable multipliers adaptable to various structures of partial product addition

Nobutaka KITO† and Naofumi TAKAGI†

† Graduate School of Information Science, Nagoya University, Furo-cho, Chikusa-ku, Nagoya, 464–8603 Japan
E-mail: †{nkito,ntakagi}@takagi.i.is.nagoya-u.ac.jp

Abstract　We propose a design method for easily testable multipliers. We construct partial product adders of a multiplier with three types of adder blocks. Various combinations of adder blocks lead to various structures of partial product addition for multipliers. A partial product generator and a final adder are also shown. We can construct multipliers with them adaptable to various structures of partial product addition. These multipliers can be tested with 14 test patterns with respect to cell fault model irrespective of its bit width. They require 6 additional input terminals.
Key words　design for testability, C-testability, tree multiplier, array multiplier

1. はじめに

VLSI設計技術や製造技術の進展に伴い，VLSIチップの集積度は高まっている．多数のゲートを含むVLSIチップに短時間で高い品質のテストを行うのは一般に難しく，集積度が高まるにつれてその困難さも高まる．しかし，規則的な構造を持つ演算器などの回路ではその規則性を利用することでテストを容易化できる．

本稿では，乗算器の部分積加算部の規則性に注目し，単一セル機能故障の仮定において，種々の部分積加算構造をもつ乗算器のCテスト可能な設計手法を提案する．Cテスト可能とは，演算のビット長に依存せず定数個のテストパターンでテストができることを意味する．

乗算器は多くのデータパス回路に組み込まれている演算回路である．一般に乗算器の性能はシステム全体の性能に大きな影響を及ぼすため，システムの要求にあわせて，規則正しい回路構造で小面積だが高速とはいえない配列型乗算器と，高速だが回路構造が複雑で面積が大きくなりがちなツリー型の乗算器で選択が必要になる．

提案手法では配列型乗算器や，ツリー型の乗算器である4-2加算木を用いる乗算器[1]などをCテスト可能に設計でき，要求される性能にあわせた乗算器をCテスト可能に設計できる．

これまでの研究でCテスト可能な配列型乗算器の設計に関する研究が行われている([2],[3]等)．文献[4]ではツリー型の乗算器のテスト手法を述べている．文献[5]では，Wallace乗算器に用いられるWallace木をCテスト可能となるように設計する手法を述べている．しかしこれらの手法は，対象とする乗算器の構成に強く依存するため，同じ手法を他のタイプの乗算器に適用することは困難であった．

並列乗算器は一般的に部分積生成部，部分積加算部，最終加算器の3つの部分から構成される．部分積生成部は乗数と被乗数から部分積を生成し，部分積加算部はその部分積を足し合わせ，最終加算器により最終結果を得る．提案手法では，部分積加算部を3種類のブロックを組み合わせて構成する．ブロックの組み合わせ方により配列型，ツリー型両方の乗算器を設計で

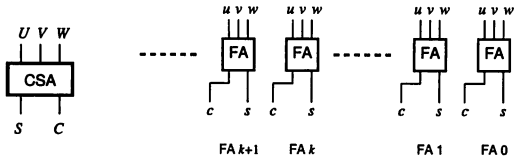

図 1　桁上げ保存加算器

きる．そして，これらの乗算器が演算のビット長に関わりなく 14 個のテストパターンでテストできることを明らかにした．

本稿の 2 節以降の構成を示す．2 節で準備として本稿で用いる故障モデルと並列乗算器の説明を行う．3 節では C-テスト可能な部分積加算部の構成法とテスト手法を示す．4 節では乗算器全体の設計法と，そのテスト方法を示す．5 節でまとめを行う．

2. 準　　備

2.1 並列乗算器の構成

本稿では，符号なし整数乗算器を扱う．$N_x \times N_y$ ビット符号なし乗算器は乗数 $X = (x_{N_x-1} \ldots x_0)_2$ と非乗数 $Y = (y_{N_y-1} \ldots y_0)_2$ の積を 2 進数で出力する．一般的な並列乗算器は部分積生成部，部分積加算部と最終加算器の 3 つの部分から構成される．

$N_x \times N_y$ ビット符号なし乗算器の部分積生成部は，乗数と被乗数から部分積 $P_j = X \cdot y_j \cdot 2^j \quad (0 \leqq j < N_y)$ を生成する．

本稿では，部分積加算部を桁上げ保存加算器 (CSA) を組み合わせることで構成する．CSA とは 3 つの 2 進数の加算を行い，桁上げと中間和の 2 つの 2 進数を出力する加算器であり，本稿では図 1 の構成を用いる．部分積加算部の出力は 2 つの 2 進数として得られる．最終加算器は 2 つの 2 進数を足し合わせ，乗算結果を出力する．

CSA を構成する全加算器 (FA) には，図 1 のように入力端子に u, v, w，出力端子に c, s と名前付けする．CSA についても入力端子に U, V, W，出力端子に S, C と名前付けする．端子 U, V, W, S, C はそれぞれ CSA を構成する FA の u, v, w, s, c 端子からなる．CSA のなかの各 FA には入力ビットの重みに応じて番号をつける．入力ビットの重みが 2^k の FA を番号 $k (k \geqq 0)$ とする．全加算器の入力端子について，重みが 2^k の入力端子 u を u^k と表記する．

CSA を組み合わせてつくる部分積加算部には図 2 のように出力側をレベル 0 とし，入力側に向かって 1 つずつ大きくなるように段にレベル付けする．部分積加算部の出力 CSA を根，部分積加算部の CSA で別の CSA の出力が入力しないものを葉と呼ぶ．各レベルにおいて，各 CSA には区別のために 0 から順に重なりなく任意の順番でラベル付けをする．

2.2 故障モデル

本稿では，単一セル機能故障を仮定する．単一セル機能故障では，特定の機能を実現するモジュールをセルとして扱い，テスト対象回路をセルを組み合わせて構成する．

単一セル機能故障の仮定においては次にあげる故障を扱う．

- テスト対象回路で故障するのは高々1 つのセルのみである．
- 故障したセルの出力は現在の入力のみで決まり，正常なセルと出力値が異なるような入力が存在する．

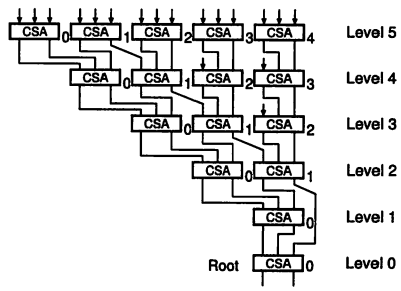

図 2　部分積加算部へのレベル付け

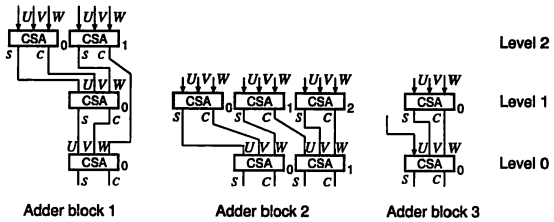

図 3　加算ブロック

単一セル機能故障を検出するため，次の 2 つの条件を満たすようにテスト集合を生成する．

- テスト対象回路を構成する全てのセルに網羅的にパターンを入力できる．
- テスト対象回路内のセル故障の影響が回路出力まで伝搬し，回路出力で観察できる．

本稿では FA 等をセルとして扱い，乗算器を構成する．単一セル機能故障では各セルの実現法にテスト方法は依存しない．

3. 部分積加算部の構成法とテスト生成

3.1 部分積加算部の構成法

部分積加算部を 3 種類のブロックを組み合わせることにより生成する．各ブロックは CSA を用いて構成されており，ブロックの組み合わせかたにより配列型乗算器の部分積加算部や 4-2 加算木 [1]，Overturned-Stairs 木 [6]，バランス木 [7] などを構成できる．

3 種類のブロックを図 3 に示す．本稿では図 3 の 3 つのブロックをそれぞれ加算ブロック 1，加算ブロック 2，加算ブロック 3 とする．各加算ブロックについて，図のように部分積加算部と同様のレベル付けを行う．CSA には各加算ブロックの各レベルで番号付けをして，CSA の右下に番号を記している．

加算ブロックを接続する際には，加算ブロックの出力部にある CSA を他の加算ブロックの入力部にある CSA と併合することで接続する．加算ブロック 1 と加算ブロック 3 を接続した場合を図 4 に示す．破線で囲んだ部分が各加算ブロックを表している．図では，レベル 2 において加算ブロック 1 の出力を加算ブロック 1 に接続し，レベル 4 において加算ブロック 3 の出力を加算ブロック 1 に接続している．

部分積加算部の設計には次の 3 つの制約を設ける．

- 加算ブロック 2 同士を接続する際には，一方の加算ブ

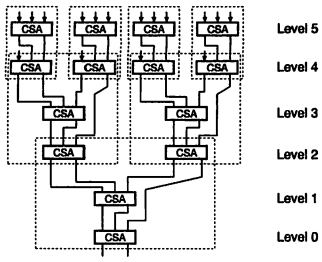

図 4　加算ブロックの接続 (4-2 加算木)

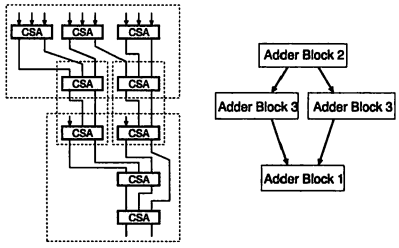

図 5　加算ブロックの出力の再収斂 ((a) 加算ブロックの接続，(b) 加算ブロック間の接続)

ロックのレベル 1 の CSA0 および CSA1 を他方のブロックのレベル 0 の CSA0 と CSA1 にそれぞれ併合することで接続する．

- 加算ブロック 2 の出力に加算ブロック 1 を接続する際には，加算ブロック 2 のレベル 0 の CSA0 と CSA1 をそれぞれ加算ブロック 1 のレベル 2 の CSA0 と CSA1 に併合することで接続する．
- 加算ブロックを組み合わせる際には，図 5 のような，加算ブロックの出力が再収斂する経路ができないようにする．図 5 では，加算ブロック 2 の出力が 2 つの加算ブロック 3 に枝分かれして入力し，加算ブロック 1 において再収斂している．

Overturned-Stairs 木は図 2 のように構成でき，4-2 加算木は図 4 のように構成できる．このほかにも，バランス木等の様々な構成を設計できる．

3.2　加算ブロックのテストのためのパターン集合

3 つの加算ブロックについて，テストのための入力パターン集合を示す．これらの加算ブロックのテストのためのパターンから，部分積加算部全体のテストのためのパターンを生成する．

加算ブロックの入力部にある CSA に，奇数番目の FA の入力ビットパターンが偶数番目の FA の入力ビットパターンの反転となるようなパターンを入力することを考える．このとき，加算ブロック 1 と 2 の入力部，そして加算ブロック 3 のレベル 1 に位置する入力部において，ある $\tilde{u}, \tilde{v}, \tilde{w} \in \{0,1\}$ が存在して以下の関係を持つ．

$$(u^j, v^j, w^j) = \begin{cases} (\tilde{u}, \tilde{v}, \tilde{w}) & (j \text{ is even}) \\ (\overline{\tilde{u}}, \overline{\tilde{v}}, \overline{\tilde{w}}) & (j \text{ is odd}) \end{cases}$$

加算ブロック 3 のレベル 0 の CSA 0 のビット入力 u では以下の関係を持つ．

$$u^j = \begin{cases} \tilde{u} & (j \text{ is even}) \\ \overline{\tilde{u}} & (j \text{ is odd}) \end{cases}$$

このようなパターンを隣接反転パターンとよぶことにし，$(\tilde{u}\ \tilde{v}\ \tilde{w})_{ai}$ や $(\tilde{u})_{ai}$ で表す．

全加算器は入力のビットパターンが反転すると出力のビットパターンも反転する性質をもつため，各加算ブロックにおいて，入力部の CSA に隣接反転パターンを入力したとき，入力段の次のレベルの CSA においても隣接反転パターンが入力される．そのため，加算ブロック内の全ての CSA に隣接反転パターンが入力される．なお，CSA の両端のビット位置の扱いについては 4.2 節で議論する．

加算ブロック 1 のテストのためのパターン集合として，隣接反転パターンを用いた表 1 の t_1^1 から $t_{\bar{5}}^1$ の入力パターンを用いる．表 1 では，レベル 2 の CSA に入力する入力パターンと，そのときレベル 1，レベル 0 の CSA に入力されるパターンを示している．隣接反転パターン $(000)_{ai}, \ldots, (111)_{ai}$ にそれぞれ $p_0, \ldots, p_7$ と名前を付けている．さらに，レベル 2 の CSA0 と CSA1 に入力されるパターンの組に $p_a, \ldots, p_{\bar{c}}$ と名前を付けている．

同様に，加算ブロック 2 のテストのためのパターン集合として表 2 の t_1^2 から $t_{\bar{5}}^2$ のパターンを用い，加算ブロック 3 には表 3 の t_1^3 から $t_{\bar{5}}^3$ のパターンを用いる．以後，特にブロックを指定して考えない場合，表 1, 表 2, 表 3 の入力パターンを t_1 のように肩の数字を省いて記す．

表 1, 表 2, 表 3 から，それぞれのブロックに対応するパターンを全て入力すると，ブロック内の全ての CSA には p_0, p_7 がそれぞれ 2 回，p_1 から p_6 までがそれぞれ 1 回ずつ入力される．このとき，CSA に含まれる各 FA には 000 から 111 までが全て入力される．また CSA の性質から，FA が故障すると演算結果は本来の値とは異なる値となるため故障の検出が可能である．このことから，3 つの表の t_1 から $t_{\bar{5}}$ は加算ブロックのテストのための入力パターンとなっている．

3.3　部分積加算部のためのテスト生成

加算ブロックに，前節で示した入力パターンを入力するとき，加算ブロック内の全ての CSA には隣接反転パターン p_0,p_7 がそれぞれ 2 回、p_1 から p_6 までがそれぞれ 1 度ずつ入力される．つまり，加算ブロックの入力部に存在する CSA と，出力部 (レベル 0) に存在する CSA では p_0 から p_7 までのそれぞれのパターンについて入力回数が両者の間で同じである．

また，加算ブロック 1,2 について，前節で示した入力パターンをすべて入力したとき，次の 3 箇所の入力パターンの組をつくると $p_a, \ldots, p_{\bar{c}}$ のパターンがそれぞれ 1 度ずつ出現する．

- 加算ブロック 1 の入力部の CSA0 と CSA1
- 加算ブロック 2 の入力部 (レベル 1) の CSA0 と CSA1
- 加算ブロック 2 の出力部 (レベル 0) の CSA0 と CSA1

これらの性質から，3.1 節に示した部分積加算部の設計法に則って部分積加算部を構成したとき，部分積加算部を構成する全て加算ブロックに，その加算ブロックに対応する入力パターンをすべて入力できる．部分積生成部のテストのための入力パターンは，部分積加算部の根の位置にある加算ブロックから葉に向かって順に求めてゆく．つまり，ある加算ブロックの入力 CSA にパターン p を入力する必要がある場合，その CSA の接

表 1　加算ブロック 1 への入力パターン集合

パターン	レベル 2 (入力)				レベル 1		レベル 0	
	CSA 0		CSA 1		CSA 0		CSA 0	
t_1^1	p_a	p_2: $(010)_{ai}$	p_5: $(101)_{ai}$		p_6: $(110)_{ai}$		p_0: $(000)_{ai}$	
$t_{\bar{1}}^1$	$p_{\bar{a}}$	p_5: $(101)_{ai}$	p_2: $(010)_{ai}$		p_1: $(001)_{ai}$		p_7: $(111)_{ai}$	
t_2^1	p_b	p_6: $(110)_{ai}$	p_0: $(000)_{ai}$		p_0: $(000)_{ai}$		p_3: $(011)_{ai}$	
$t_{\bar{2}}^1$	$p_{\bar{b}}$	p_1: $(001)_{ai}$	p_7: $(111)_{ai}$		p_7: $(111)_{ai}$		p_4: $(100)_{ai}$	
t_3^1	p_c	p_0: $(000)_{ai}$	p_4: $(100)_{ai}$		p_3: $(011)_{ai}$		p_1: $(001)_{ai}$	
$t_{\bar{3}}^1$	$p_{\bar{c}}$	p_7: $(111)_{ai}$	p_3: $(011)_{ai}$		p_4: $(100)_{ai}$		p_6: $(110)_{ai}$	
t_4^1	p_d	p_3: $(011)_{ai}$	p_6: $(110)_{ai}$		p_0: $(000)_{ai}$		p_2: $(010)_{ai}$	
$t_{\bar{4}}^1$	$p_{\bar{d}}$	p_4: $(100)_{ai}$	p_1: $(001)_{ai}$		p_7: $(111)_{ai}$		p_5: $(101)_{ai}$	
t_5^1	p_e	p_0: $(000)_{ai}$	p_0: $(000)_{ai}$		p_2: $(010)_{ai}$		p_7: $(111)_{ai}$	
$t_{\bar{5}}^1$	$p_{\bar{e}}$	p_7: $(111)_{ai}$	p_7: $(111)_{ai}$		p_5: $(101)_{ai}$		p_0: $(000)_{ai}$	

表 2　加算ブロック 2 への入力パターン集合

パターン	レベル 1 (入力)			レベル 0		
	CSA0	CSA1	CSA2	CSA0		CSA1
t_1^2	p_e :	p_0	p_0	p_0	p_a :	p_2 p_5
$t_{\bar{1}}^2$	$p_{\bar{e}}$:	p_7	p_7	p_7	$p_{\bar{a}}$:	p_5 p_2
t_2^2	p_a :	p_2	p_5	p_5	p_b :	p_6 p_0
$t_{\bar{2}}^2$	$p_{\bar{a}}$:	p_5	p_2	p_2	$p_{\bar{b}}$:	p_1 p_7
t_3^2	p_b :	p_6	p_0	p_6	p_c :	p_0 p_4
$t_{\bar{3}}^2$	$p_{\bar{b}}$:	p_1	p_7	p_1	$p_{\bar{c}}$:	p_7 p_3
t_4^2	p_c :	p_0	p_4	p_7	p_d :	p_3 p_6
$t_{\bar{4}}^2$	$p_{\bar{c}}$:	p_7	p_3	p_0	$p_{\bar{d}}$:	p_4 p_1
t_5^2	p_d :	p_3	p_6	p_3	p_e :	p_0 p_0
$t_{\bar{5}}^2$	$p_{\bar{d}}$:	p_4	p_1	p_4	$p_{\bar{e}}$:	p_7 p_7

統元の加算ブロックの出力部 CSA の入力が p となるようにパターンを選択する．このとき，10 パターンで部分積加算部のすべての加算ブロックにそれぞれに対応するパターンを入力できる．

図 6 の配列型乗算器の部分積加算部を例にして，テストのための入力パターンの設計法を示す．配列型乗算器の部分積加算部は加算ブロック 3 を繰り返し接続することで構成できる．部分積加算部の根の加算ブロックに，t_1^3 から t_5^3 までの全てを入力するようにパターンの集合を生成する．図では，根の加算ブロックに t_1^3 を入力する場合を示す．根の加算ブロックに t_1^3 を入力する場合，レベル 1 の CSA には $(000)_{ai}(=p_0)$ を入力する必要があり，表 3 より，加算ブロックの出力部 CSA のパターンが p_0 となるものを選ぶ．t_2^3 又は t_5^3 を入力することでレベル 1 の CSA に p_0 を入力できる．この例では t_5^3 を用いている．レベル 0,1 の加算ブロックに t_4^3 を入力する際にも，レベル 1 の CSA に p_0 を入力するが，その際には t_2^3 を用いる．

残った部分についても，入力側に向かって入力パターンを求められ，部分積加算部の入力パターンが決まる．このとき，3.1 節の制約から再収斂を起こす経路はないので，求まったパターンで不整合は起きない．

4. C テスト可能な乗算器の設計

3 節で示した部分積加算部を用いた乗算器の構成法とテスト方法を示す．本稿で提案する乗算器の構成を図 7 に示す．図中の d_1, d_2, r, r_0, r_1, r_2 は本節で導入するテストのための外部入力である．

表 3　加算ブロック 3 への入力パターン集合

パターン	レベル 1 (入力)	レベル 0	
	CSA0	u (入力)	CSA0
t_1^3	p_0	$(1)_{ai}$	p_5
$t_{\bar{1}}^3$	p_7	$(0)_{ai}$	p_2
t_2^3	p_2	$(1)_{ai}$	p_7
$t_{\bar{2}}^3$	p_5	$(0)_{ai}$	p_0
t_3^3	p_6	$(1)_{ai}$	p_4
$t_{\bar{3}}^3$	p_1	$(0)_{ai}$	p_3
t_4^3	p_0	$(0)_{ai}$	p_1
$t_{\bar{4}}^3$	p_7	$(1)_{ai}$	p_6
t_5^3	p_3	$(0)_{ai}$	p_0
$t_{\bar{5}}^3$	p_4	$(1)_{ai}$	p_7

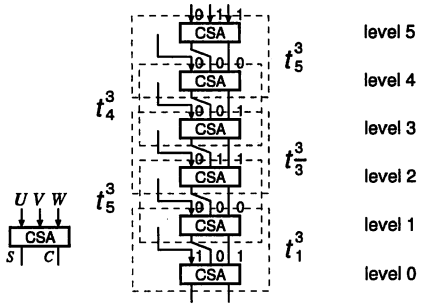

図 6　配列型乗算器の部分積加算部の入力パターン生成

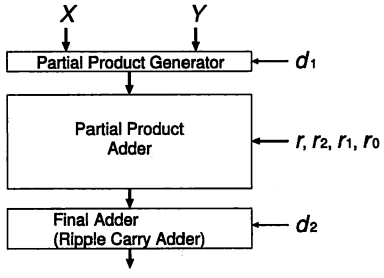

図 7　乗算器の構成

4.1 部分積生成部の構成

本稿では，外部入力信号 d_1 を導入して $0 \leqq i, j < N$ について次の式のように部分積のビットを生成する．

$$pp_{ij} = \begin{cases} x_i \cdot y_j & (i+j \text{ is even}) \\ x_i \cdot (y_j \oplus d_1) & (i+j \text{ is odd}) \end{cases}$$

部分積生成部は Y の各ビットにつき $y_j \oplus d_1$ を計算するための Controlable Inversion(CI) セルと，部分積ビットそれぞれに $x_i \oplus y_j$ あるいは $x_i \oplus (y_j \oplus d_1)$ を計算するための Partial Product Generator(PPG) セルを用いて構成する．CI セルは 2 入力 1 出力のセルで，排他的論理和を計算するセルであり，PPG セルは 2 入力 1 出力のセルで，入力の論理積を計算する．例として 8×8 ビット乗算器の部分積生成回路の，P_0, P_1 の生成部を図 8 に示す．$P_{j_e}(j_e$ は偶数) の生成回路は P_0 と同じ構成をとり，$P_{j_o}(j_o$ は奇数) の生成回路は P_1 と同じ構成をとる．通常時は d_1 に 0 を入力し，テスト時に 1 とする．$d_1 = 1, X = 11\ldots1$ とすると部分積加算部に隣接反転パターンを入力できる．

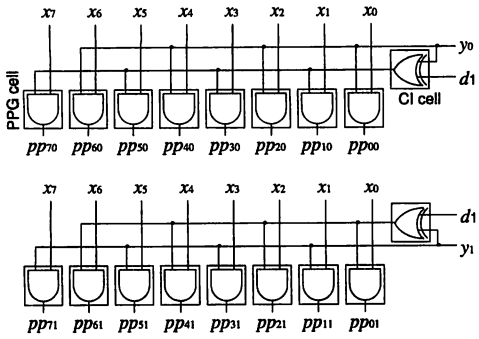

図 8　8×8 ビット符号なし乗算器の部分積生成部

部分積加算部の葉の各 CSA には 3 つの部分積を入力する．CSA に P_g, P_{g+1}, P_{g+2} を入力するとき，CSA の入力ビット幅は N_x+2 ビットで，最小の入力重みは 2^g になる．このとき，CSA 内部の各 FA(入力重み 2^f, $g \leqq f < g+N_x+2$) と部分積生成部を次のように接続する．

$$(u^f, v^f, w^f) \leftarrow (pp_{(f-g)g}, pp_{(f-g-1)(g+1)}, pp_{(f-g-2)(g+2)})$$

ただし，pp_{ij} ($i<0$ あるいは $i \geqq N_x$) の箇所は 4.2 節で述べる入力補完ユニットに接続する．$d_1 = 1, X = 11\ldots1$ のとき，FA の入力ビットパターンは次の式で表される．

$$(u^f, v^f, w^f) = \begin{cases} (y_g, y_{g+1}, y_{g+2}) & (f \text{ is even}) \\ (\overline{y_g}, \overline{y_{g+1}}, \overline{y_{g+2}}) & (f \text{ is odd}) \end{cases}$$

上の式より，$d_1 = 1, X = 11\ldots1$ としたとき，CSA に隣接反転パターンを入力できることが分かる．つまり，部分積 P_g, P_{g+1}, P_{g+2} を入力とする CSA には隣接反転パターン $(y_g\ y_{g+1}\ y_{g+2})_{ai}$ を入力できる．

加算ブロック 3 のレベル 0 に位置する CSA の入力についても同様で，P_g を入力とすると，隣接反転パターン $(y_g)_{ai}$ を入力できる．したがって，3.3 節で求めた部分積加算部のテストのための入力パターンは部分積生成部を通して全て入力できる．

4.2　桁上げ保存加算器の構成

各部分積は，ビット幅が等しく，最小の入力重みがそれぞれ異なっている．そのため，部分積を入力とする CSA の両端のいくつかの重みでは入力が 2 つや 1 つになる．これらの部分では入力が隣接反転のパターンとならない．また，部分積加算部の内部の CSA についても，CSA の両端の重みにおいて入力が 3 未満となる箇所がある．

このような入力が 3 未満の重みについて，隣接反転パターンの規則性を保つために入力を入れ，CSA の各重みで入力が必ず 3 つになるようにする．これらの入力を生成するために，各 CSA に以下で説明する入力ビット補完器を一つずつ接続する (図 9)．入力ビット補完器は通常時には 0 を出力するので演算結果に影響は与えない．

入力ビット補完器のために，4 本の外部入力 r, r_0, r_1, r_2 を導入する．r は，テスト時に，部分積加算部の根に位置する加算ブロックの入力パターンが $t_{\overline{z}}$ ($z \in \{1,2,3,4,5\}$) になるテストパターンのときだけ 1 にする．一方，r_0, r_1, r_2 には r_0, r_1, r_2 を 2 進数 $(r_2\ r_1\ r_0)_2$ と見立て，部分積加算部の根に位置する

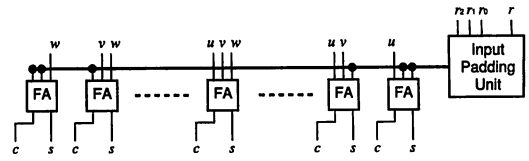

図 9　桁上げ保存加算器と入力ビット補完器

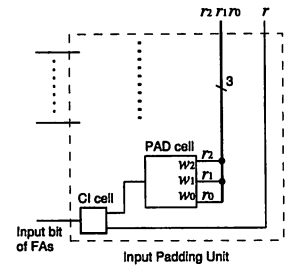

図 10　入力ビット補完器の構成

加算ブロックに t_1,$t_{\overline{1}}$ を入力するとき 1，t_5,$t_{\overline{5}}$ を入力するとき 5 というように根に入力するパターンに連動した値を入力する．通常動作時には r や r_1, r_2, r_3 は 0 とする．

入力ビット補完器は PAD セルと CI セルを用いて構成する．PAD セルは 3 入力 1 出力のセルで，一つ一つのセルの機能は異なる．その機能を PAD($\{n_1, n_2, \ldots, n_m\}$)($0 \leqq n_i < 8$) の形で記述し，このとき，入力 w_0, w_1, w_2 を 2 進数 $(w_2\ w_1\ w_1)_2$ と見立て，$n_1, n_2, \ldots, n_m$ に含まれれば 1 を出力し，それ以外で 0 を出力することを意味する．

3.3 節で設計した部分積加算部のテストのためのパターンでは，全ての CSA の入力パターンは根のブロックに入る入力パターンと対応関係がある．$(r_2 r_1 r_0)_2$ と r より根の加算ブロックへの入力パターンが分かるため，これらの外部入力を用いて任意の CSA 内の FA への入力パターンが生成できる．

入力ビット補完器の 1 ビット分を図 10 に示す．根の加算ブロックに $t_{z'}(z' \in Z' \subset \{1,2,3,4,5\})$ が入力されるとき 1 となる信号は，外部入力信号 r_2, r_1, r_0 と PAD セル (PAD(Z')) を 1 つ用いて生成できる．根の加算ブロックへの入力パターンが t_z から $t_{\overline{z}}$ になると，部分積加算部の内部の全ての FA の入力ビットは反転することから，PAD セルの出力を CI セルと r 信号を用いて反転させる．

なお，PAD セルや CI セルの故障の影響は部分積加算部に伝搬し，乗算器の出力でその影響は観測できる．

4.3　最終加算器の構成

最終加算器として本稿では図 11 の順次桁上げ加算器 (RCA) を用いる．図のように，FA をつなげた構造になっており，RCA の入力にはビット位置 1 つおきに CI セルを挟む．テストのために外部入力 d_2 を導入する．通常動作時，d_2 の入力は 0 とする．

最終加算器の入力は部分積加算部の根の CSA の出力である．3.3 節で示した部分積加算部のテストのためのパターンでは，部分積加算部内の全ての CSA に隣接反転パターン p_0, p_7 がそれぞれ 2 回，p_1 から p_6 までがそれぞれ 2 回入力される．この性質を利用し，部分積加算部のテストと同時に最終加算器もテストする．

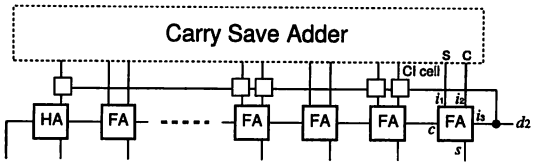

図 11 最終加算器の構成

表 4 最終加算器への入力パターン集合

	最下位ビットへの入力	d_2	最終加算器内の FA の入力 $i_1 i_2 i_3$		
			最下位 (0 個目)	偶数個目	奇数個目
t_{A1}	00	1	001	000	000
t_{A2}	00	0	000	001	110
t_{A3}	00	0	000	001	110
t_{A4}	01	0	010	010	100
t_{A5}	01	1	011	011	011
t_{A6}	10	0	100	100	010
t_{A7}	10	1	101	101	101
t_{A8}	11	1	111	111	111
t_{A9}	11	1	111	111	111
t_{A10}	11	0	110	110	001

CSA に隣接反転パターン p_0, p_7 がそれぞれ 2 回，p_1 から p_6 までがそれぞれ 1 回入力すると，出力も隣接反転パターンで得られ，CSA の各 FA は $(c,s)=(0,0),(1,1)$ をそれぞれ 2 回，$(0,1),(1,0)$ をそれぞれ 3 回出力する．そのため，最終加算器の各ビット位置で入力には 00,11 が 3 回ずつ，01,10 がそれぞれ 2 回ずつ入力される．最終加算器のテストのための入力パターンを表 4 に示す．

表 4 では，1 列目に最終加算器の最下位のビット位置での入力を示している．最終加算器の入力パターンも隣接するビット位置で反転するので全体の入力はこのパターンで表すことができる．2 列目に外部入力 d_2 の入力の値を示している．3 番目の列は最終加算器の最下位の位置にある FA の入力を示し，4,5 番目の列では偶数個目，奇数番目の FA の入力を示している．

表より，最終加算器の全ての FA，CI セルに網羅的にパターンを入力できることが確認できる．また，FA や CI セルが故障すると乗算器の出力が正しい値とならないため、最終加算器を構成するセルの故障は最終加算器の出力より確認できる．

4.4 C テスト可能な乗算器のテスト集合

提案手法により設計した乗算器において，部分積加算部のテストに隣接反転パターンを用いる．3.3 節で示したように，10 個の隣接反転パターンにより，部分積加算部をテストできる．

部分積生成部を通して部分積加算部に隣接反転パターンを入力する方法を 4.1 節で示した．部分積加算部は 10 個のテストパターンによりテストできる．また，4.3 節より最終加算器と部分積加算部は同時にテストできる．

部分積加算部のためのテストパターン入力時，外部入力信号 r_0, r_1, r_2 には $(r_2\ r_1\ r_0)_2$ が 1 から 5 までのみが入力される．そのため，入力ビット補完器内の PAD セルには入力されないパターンがある．また，外部入力信号 d_1 と X はそれぞれ $1,11\ldots1$ となっており，部分積生成部の PPG セルや CI セルには入力されないパターンがある．

そこで，以下の 4 つのパターンを追加する．“*” の箇所はドントケアを意味する．

$$
\begin{aligned}
(X, Y, d_1, d_2, r, r_2, r_1, r_0) &= (000\ldots0, 000\ldots0, 1, *, *, 0, 0, 0),\\
&\quad (000\ldots0, 111\ldots1, 1, *, *, 1, 1, 0),\\
&\quad (111\ldots1, 000\ldots0, 0, *, *, 1, 1, 1),\\
&\quad (111\ldots1, 111\ldots1, 0, *, *, *, *, *)
\end{aligned}
$$

この 4 つのパターンを先に述べた 10 個テストパターンと共に用いることで，部分積生成部の PPG セルや CI セル，PAD セルに全ての入力組合せを入力できる．また，部分積生成部の PPG セルや CI セルの故障の影響は乗算器の出力で観測できる．

部分積加算部と最終加算器のための 10 個のテストパターンと，上に示した 4 個のテストパターンを用いることで乗算器全体のテストができる．テストパターンの数は演算のビット長に依存せず 14 個のため，提案乗算器は C テスト可能である．このとき，テストのために必要な追加の外部入力は d_1, d_2, r, r_2, r_1, r_0 の 6 本である．

5. ま と め

C テスト可能な乗算器を設計手法を示した．提案手法では乗算器の部分積加算部を 3 つの加算ブロックを組み合わせて設計する．加算ブロックの組み合わせかたにより様々なタイプの乗算器を設計できる．提案手法で設計した乗算器は，14 個のテストパターンで乗算器をテストできた．提案手法で設計した乗算器にはテストのために 6 本の追加の外部入力が必要であった．最終加算器として遅延時間の大きい順次桁上げ加算器を用いたが，テストが容易で高速な桁上げ伝播加算器 ([8], [9] 等) を組み合わせて用いることも可能である．

文 献

[1] J. Vuillemin, “A very fast multiplication algorithm for VLSI implementation,” Integration the VLSI journal, vol. 1, pp 39–52, 1983.

[2] J.P. Shen and F.J. Ferguson, “The design of easily testable VLSI array multipliers,” IEEE Trans. on Comput., vol. C-33, pp 554–560, June 1984.

[3] D. Gizopoulos, D. Nikolos, A. Paschalis, and C. Halatsis, “C-testable modified-booth multipliers,” Journal of Electronic Testing, Vol. 8, pp. 241–260, June 1996.

[4] P. Zeng, Z. Mao, Y. Ye, and Y. Deng, “Test pattern generation for column compression multiplier,” Proc. Seventh Asian Test Symposium, pp. 500–503, 1998.

[5] A. Chatterjee and J.A. Abraham, “On the C-testability of generalized counters,” IEEE Trans. on CAD, Vol. CAD-6, pp.713–726, Sep. 1987.

[6] Z.-J. Mou, and F. Jutand, “Overturned-stairs adder trees and multiplier design,” IEEE Trans. on Computers, Vol. 41, pp. 940–948, Aug. 1992.

[7] S.D. Pezaris, “A 40-ns 17-bit by 17-bit array multiplier,” IEEE Trans. on Computers, Vol. C-20, pp. 442–447, Apr. 1971.

[8] R.D. Blanton, and J.P. Hayes, “Testability of convergent tree circuits,” IEEE Trans. on Computers, Vol. 45, pp. 950–963, Aug. 1996.

[9] D. Gizopoulos, M. Psarakis, A. Paschalis, and Y. Zorian, “Easily testable cellular carry lookahead adders,” Journal of Electronic Testing, Vol. 19, pp. 285–298, June 2003.