論　文

種々の部分積加算構造をもつテスト容易な乗算器の設計手法

鬼頭　信貴[†a)]　　高木　直史[†b)]

A Design Method of Easily Testable Multipliers with Various Structures of Partial Product Adder

Nobutaka KITO[†a)] and Naofumi TAKAGI[†b)]

あらまし　様々な部分積加算構造を構成できるテスト容易な乗算器の設計手法を示す．提案手法では，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ることで設計し，ブロックの組合せ方により様々な部分積加算構造を構成できる．部分積加算部ととも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．全加算器等をセルとする単一セル機能故障の仮定のもとで，演算数のビット長にかかわりなく 14 個のパターンでテストできる．

キーワード　テスト容易化設計，C テスト可能，トリー型乗算器，配列型乗算器

1. まえがき

VLSI 設計技術や製造技術の進展により，VLSI チップ上に集積される回路がますます大規模化し，そのテストに要するコストが増大している．テストのコストを低減するために VLSI を構成する部分回路をテスト容易に設計することは重要である．本論文では，データパス回路において多く用いられる並列乗算器について，テスト容易な設計法を示す．

並列乗算器は一般的に部分積生成部，部分積加算部，最終加算部の三つの部分から構成される．部分積生成部は被乗数と乗数から部分積を生成し，部分積加算部はその部分積をけた上げ保存形で足し合わせ，最終加算部は部分積加算部の出力として得られるけた上げ保存形のけた上げと中間和を足し合わせて最終結果を得る．

提案手法では，部分積加算部を 3 種類の加算器のブロックを組み合わせて構成する．ブロックの組合せ方により，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や，高速だが回路構造が複雑で面積がやや大きいトリー型の乗算器を設計できる．これにより，要求される性能に合わせたテスト容易な乗算器を設計できる．乗算器は，全加算器等をセルとする単一セル機能故障の仮定において，演算数のビット長にかかわりなく 14 個のパターンでテストできる．

演算数のビット長にかかわりなく定数個のパターンでテストできる回路は，C テスト可能と呼ばれる．これまでに 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]〜[4] 等）．文献 [5], [6] ではトリー型乗算器の一つである 4-2 加算木を用いた乗算器のテスト手法を述べている．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できない．そのため，要求性能が変化し，部分積加算の構造を他のタイプに変更する場合，同じ手法では対応できなかった．

本論文の **2.** 以降の構成を示す．**2.** で準備として並列乗算器と本論文で用いる故障モデルの説明を行う．**3.** 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.** では乗算器全体の設計法と，そのテスト方法を示す．**5.** でまとめを行う．

2. 準　　備

2.1 並列乗算器の構成

本論文では，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被乗数 $X = (x_{N_x-1} \ldots x_0)_2$ と乗数 $Y = (y_{N_y-1} \ldots y_0)_2$ の積を 2 進数で出力する．

† 名古屋大学大学院情報科学研究科，名古屋市
Department of Information Engineering, Nagoya University, Nagoya-shi, 464–8603 Japan
a) E-mail: nkito@takagi.i.is.nagoya-u.ac.jp
b) E-mail: ntakagi@takagi.i.is.nagoya-u.ac.jp

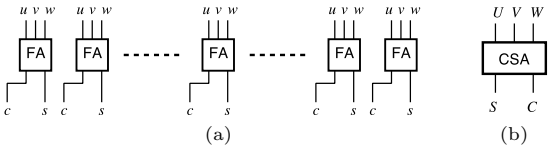

図 1　けた上げ保存加算器
Fig. 1　A carry save adder.

一般的な並列乗算器は部分積生成部，部分積加算部，最終加算部の三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，被乗数と乗数から部分積 $P_j = X \cdot y_j \cdot 2^j \quad (0 \leq j < N_y)$ を生成する．この部分積を部分積加算部と最終加算部で足し合わせることで演算結果が得られる．部分積加算部では，けた上げ伝搬による遅延が起こらない，けた上げ保存形で部分積の加算を行う．部分積加算部の加算結果はけた上げ保存形のけた上げと中間和の二つの 2 進数で出力され，最終加算器でこれらを足し合わせることで演算結果を得る．

部分積加算部はけた上げ保存加算器（CSA）を組み合わせて構成する．CSA は三つの 2 進数を足し合わせ，加算結果はけた上げ保存形でけた上げと中間和の二つの 2 進数で出力する．CSA は全加算器（FA）を用いて構成する．CSA の構成を図 1 (a) に示す．図の CSA では，三つの数の各ビットを FA を用いて足し合わせ，CSA の両端のけたを除いて，出力は各けたで 2 ビットずつ得られる．

CSA を構成する FA には，図 1 (a) 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．図 1 (b) のように，CSA の入力に U，V，W，出力に S，C と名前付けする．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 の中の各 FA には入力ビットの重みに応じて番号を付ける．乗算器の中で入力ビットの重みが 2^k の FA を番号 k $(k \geq 0)$ とする．FA 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 を組み合わせて設計す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，入力側に向かってレベル付けする．CSA の属するレベルは，その CSA の出力側の CSA が属するレベルの最大値より一つ大きい値とする．部分積加算部をトリーとしてみることにし，レベル 0 の CSA を根，他の CSA の出力が入力とならない CSA 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序でラベル付けをする．

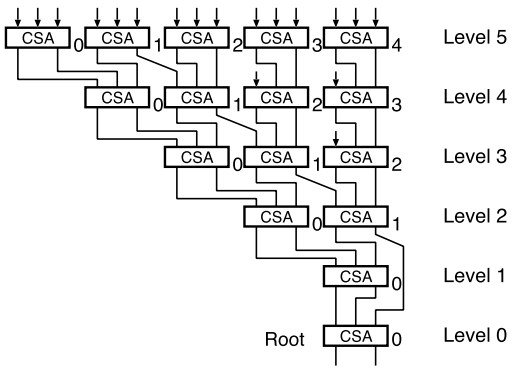

図 2　部分積加算部へのレベル付け
Fig. 2　Levels of carry save adders.

2.2　故障モデル

本論文では，単一セル機能故障を仮定する．本論文では FA 等をセルとして扱い，これらのセルを用いて乗算器を構成する．単一セル機能故障では，次の故障を扱う．

- テスト対象回路で故障するのはたかだか一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．

単一セル機能故障を検出するため，次の二つの条件を満たすようにテストを設計する．

- テスト対象回路を構成するすべてのセルの各々に全入力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．

3.　C テスト可能な部分積加算部の構成法とテスト設計

3.1　部分積加算部の構成法

部分積加算部を CSA で構成した 3 種類のブロックを組み合わせることにより構成する．ブロックの組合せ方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 [7]，バランス木 [8] などを構成できる．

3 種類のブロックを図 3 に示す．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，レベル付けと各レベルでの CSA へのラベル付けをしている．図において CSA のラベルを CSA の右下に記している．

部分積加算部は根から葉に向かって加算ブロックを

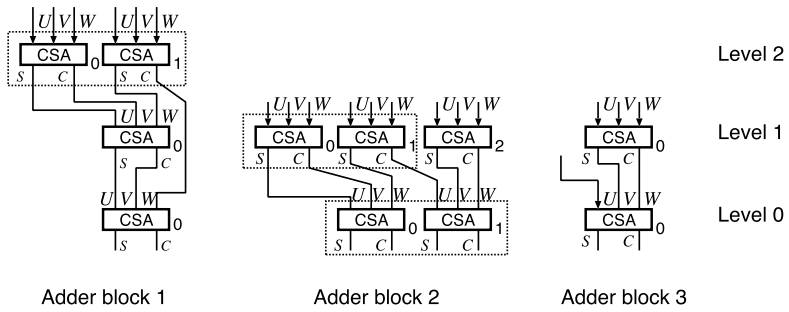

図 3　加算ブロック
Fig. 3　Adder blocks.

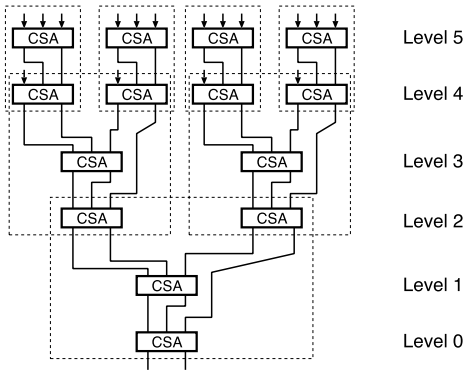

図 4　部分積加算部の設計例（4-2 加算木）
Fig. 4　An example of a partial product adder. (4-2 adder tree)

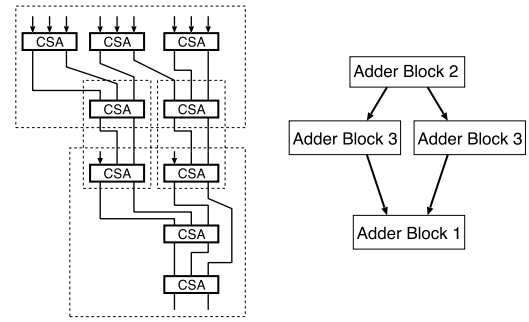

図 5　加算ブロック出力が再収れんする接続禁止の例（(a) 構成，(b) 加算ブロックレベルでの接続）
Fig. 5　An example of forbidden connections with reconvergence of an adder block's output. ((a) Configuration, (b) Connections of adder blocks)

接続することで構成する．部分積加算部の初期構成を加算ブロック 1 あるいは加算ブロック 3 とし，各時点の部分積加算部の葉に加算ブロックを接続していくことで目的の部分積加算部を得る．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 をその時点の部分積加算部の葉と併合することで接続する．

ただし加算ブロック 2 の接続については，その時点の部分積加算部の葉の CSA のうち，図 3 で破線で示した加算ブロック 1 のレベル 2 の CSA0，1 あるいは加算ブロック 2 のレベル 1 の CSA0，1 に，同じく破線で示した加算ブロック 2 のレベル 0 の CSA0，1 をそれぞれ併合することで接続し，これ以外の箇所に加算ブロック 2 を接続しない．

加算ブロック 1 と加算ブロック 3 を接続して部分積加算部を設計した例を図 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，部分積加算部のレベル 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．

加算ブロック 2 の接続についての制約により，図 5 のような，加算ブロックの出力が別の加算ブロックの入力において再収れんする部分積加算部は構成されない．図 5 では，加算ブロック 2 の出力が二つの加算ブロック 3 に枝分かれして入力し，加算ブロック 1 において再収れんしている．

本節の構成法を用いることで，トリー型の高速な乗算器の部分積加算部に用いられる 4-2 加算木は図 4 のように構成できる．加算ブロックの出力が再収れんすることのある Wallace 木は設計できないものの，バランス木や図 2 のような Overturned-Stairs 木等の様々な部分積加算部を設計できる．

3.2　加算ブロックのテストのためのパターン集合

三つの加算ブロックについて，個々のブロックをテストする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設計する．

加算ブロックの入力部にある CSA に，偶数番目のすべての FA の入力ビットパターンが同じで，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．言い換えると，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，パターンに応じて $\alpha, \beta, \gamma \in \{0, 1\}$ が存在して，以下の関係をもつ．

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

加算ブロック 3 のレベル 0 のビット入力端子 u では以下の関係をもつ．

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

このようなパターンを交互反転パターンと呼ぶことにし，$(\alpha\ \beta\ \gamma)_{ai}$ や $(\alpha)_{ai}$ で表す．図 6 (a)，(b) に加算ブロック 3 に交互反転パターンを入力した例を示す．図では加算ブロックのレベル 1 の CSA に $(010)_{ai}$，レベル 0 の入力端子に $(1)_{ai}$ を入力している．図 6 (a) の FA において，FA の番号が偶数ものを網かけで示している．

FA は入力のパターンが反転すると出力のパターンも反転する性質をもつ．そのため，各加算ブロックにおいて入力部の CSA に交互反転パターンを入力すると，入力部の次のレベルの CSA においても交互反転パターンが入力される．したがって，加算ブロックの入力部の CSA に交互反転パターンを入力すると，加算ブロック内のすべての CSA に交互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** で議論する．

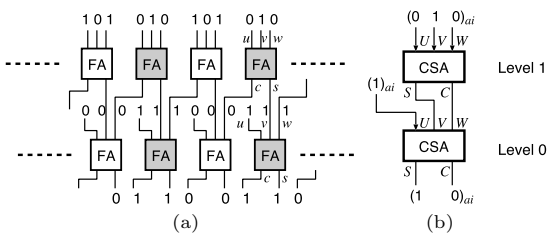

図 6　交互反転パターンの入力例
Fig. 6　An example of applying alternately inverted patterns.

加算ブロック 1 のテストのために，交互反転パターンを用いた表 1 の t_1^1 から $t_{\overline{5}}^1$ の 10 個の入力パターンを用いる．各パターンの上付き文字は加算ブロックを表し，下付き文字はパターンの番号を表している．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．表に示すように交互反転パターン $(000)_{ai}, \ldots, (111)_{ai}$ にそれぞれ $p_0, \ldots, p_7$ と名前付けする．更に，表 1 においてレベル 2 の CSA0 と CSA1 に入力されるパターンの組に $p_a, \ldots, p_{\overline{e}}$ と名前付けする．

同様に，加算ブロック 2 のテストのために表 2 の t_1^2 から $t_{\overline{5}}^2$ の 10 個のパターンを用い，加算ブロック 3 には表 3 の t_1^3 から $t_{\overline{5}}^3$ の 10 個のパターンを用いる．以降，特に特定のブロックに依存しない文脈では，これらのパターンを記す際，ブロック名を表す肩の数字を省き，$t_1, \ldots, t_{\overline{5}}$ と表す．

表 1，表 2，表 3 から，三つの各ブロックについて，ブロックに対応するパターン集合のパターンをすべて

表 1　加算ブロック 1 への入力パターン集合
Table 1　Input pattern set for the adder block 1.

	Inputs for Level 2			Level 1	Level 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: $(000)_{ai}$
$t_{\overline{1}}^1$	$p_{\overline{a}}$	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: $(011)_{ai}$
$t_{\overline{2}}^1$	$p_{\overline{b}}$	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: $(001)_{ai}$
$t_{\overline{3}}^1$	$p_{\overline{c}}$	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: $(010)_{ai}$
$t_{\overline{4}}^1$	$p_{\overline{d}}$	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: $(111)_{ai}$
$t_{\overline{5}}^1$	$p_{\overline{e}}$	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
Table 2　Input pattern set for the adder block 2.

	Inputs for level 1			Level 0	
	CSA0	CSA1	CSA2	CSA0	CSA1
t_1^2	p_e : p_0	p_0	p_0	p_a : p_2	p_5
$t_{\overline{1}}^2$	$p_{\overline{e}}$: p_7	p_7	p_7	$p_{\overline{a}}$: p_5	p_2
t_2^2	p_a : p_2	p_5	p_5	p_b : p_6	p_0
$t_{\overline{2}}^2$	$p_{\overline{a}}$: p_5	p_2	p_2	$p_{\overline{b}}$: p_1	p_7
t_3^2	p_b : p_6	p_0	p_6	p_c : p_0	p_4
$t_{\overline{3}}^2$	$p_{\overline{b}}$: p_1	p_7	p_1	$p_{\overline{c}}$: p_7	p_3
t_4^2	p_c : p_0	p_4	p_7	p_d : p_3	p_6
$t_{\overline{4}}^2$	$p_{\overline{c}}$: p_7	p_3	p_0	$p_{\overline{d}}$: p_4	p_1
t_5^2	p_d : p_3	p_6	p_3	p_e : p_0	p_0
$t_{\overline{5}}^2$	$p_{\overline{d}}$: p_4	p_1	p_4	$p_{\overline{e}}$: p_7	p_7

表 3　加算ブロック 3 への入力パターン集合
Table 3　Input pattern set for the adder block 3.

	Level 1	Level 0	
	Inputs for CSA0	Input for u	CSA0
t_1^3	p_0	$(1)_{ai}$	p_5
$t_{\overline{1}}^3$	p_7	$(0)_{ai}$	p_2
t_2^3	p_2	$(1)_{ai}$	p_7
$t_{\overline{2}}^3$	p_5	$(0)_{ai}$	p_0
t_3^3	p_6	$(1)_{ai}$	p_4
$t_{\overline{3}}^3$	p_1	$(0)_{ai}$	p_3
t_4^3	p_0	$(0)_{ai}$	p_1
$t_{\overline{4}}^3$	p_7	$(1)_{ai}$	p_6
t_5^3	p_3	$(0)_{ai}$	p_0
$t_{\overline{5}}^3$	p_4	$(1)_{ai}$	p_7

入力すると，ブロック内のすべての CSA に 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 までがすべて入力される．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，乗算器の出力において故障の影響を観測できる．このことから，三つの表の t_1 から $t_{\overline{5}}$ 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設計

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，加算ブロック内のすべての CSA には交互反転パターン 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の CSA と，出力部（レベル 0）の CSA の両者で，p_0 から p_7 までのそれぞれのパターンについての出現回数が同じである．

また，加算ブロック 1 について前節で示した入力パターンを入力するとき，入力部の CSA0，CSA1 に $p_a, \ldots, p_{\overline{e}}$ のパターンが 1 度ずつ入力される．同様に，加算ブロック 2 についても，前節で示した入力パターンを入力するとき，入力部（レベル 1）と出力部（レベル 0）の CSA0，CSA1 に $p_a, \ldots, p_{\overline{e}}$ のパターンが 1 度ずつ入力される．

これらの性質から，**3.1** に示した設計法で部分積加算部を構成したとき，10 パターンで部分積加算部を構成するすべて加算ブロックに，各加算ブロックに対応する t_1 から $t_{\overline{5}}$ の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パターン集合は，根の加算ブロックに t_1 から $t_{\overline{5}}$ のパターンを入力する 10 個の入力パターンから構成する．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 CSA の入力パターンを順に求めていくことで構成する．つまり，ある加算ブロックの入力部の CSA にパターン p を入力する必要がある場合，その CSA の接続元の加算ブロックの出力部の CSA の入力が p となるように，t_1 から $t_{\overline{5}}$ の中から接続元の加算ブロックへのパターンを選択する．このとき，**3.1** の制約から加算ブロックの出力が再収れん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．

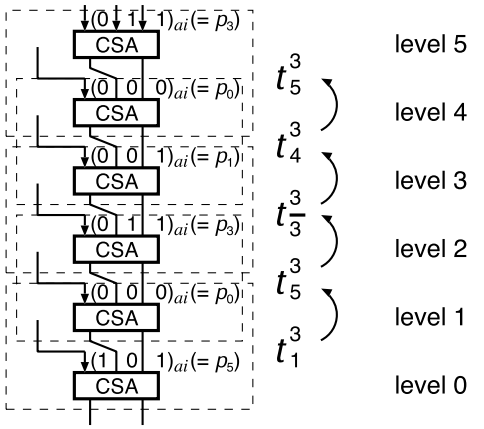

図 7　配列型乗算器の部分積加算部の入力パターン設計の例
Fig. 7　An example of input pattern design for an array multiplier.

図 7 の配列型乗算器の部分積加算部を例に，テストのための入力パターンの構成法を示す．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 $t_{\overline{5}}^3$ までのすべてを入力するようにパターンの集合を構成する．図では，根の加算ブロックに t_1^3 を入力する場合を示している．根の加算ブロックに t_1^3 を入力する場合，図 7 のレベル 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（レベル 0）の CSA のパターンが p_0 となるものを，図 7 のレベル 1，2 の加算ブロックへの入力割当とする．$t_{\overline{2}}^3$ または t_5^3 を入力することで図 7 のレベル 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．図 7 のレベル 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 $t_{\overline{2}}^3$ 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．図 7 のレベル 4 の CSA にも p_0 を入力するが，このときレベル 4，5 の加算ブロックには t_2^3 あるいは t_5^3 どちらを用いてもよく，例では t_5^3 を用いている．この場合，レベル 4 の CSA に p_0 を入力する別の入力パターンを構成する際にはレベル 4，5 の加算ブロックに t_2^3 を用いる．

4. C テスト可能な乗算器の設計手法

前章で示した部分積加算部設計手法を用いた，乗算器の構成法とテスト方法を示す．本論文で提案する乗算器の構成を図 8 に示す．図中の d_1，d_2，r，r_0，r_1，r_2 は本章で導入するテストのための外部入力である．

4.1 部分積生成部の構成

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

$$pp_{ij} = \begin{cases} x_i \cdot y_j & (i+j \ \text{ is even}) \\ x_i \cdot (y_j \oplus d_1) & (i+j \ \text{ is odd}) \end{cases} \quad (1)$$

部分積生成部は Y の各ビットにつき $y_j \oplus d_1$ を計算するための制御入力つき反転セル（CI セル）と，部分積ビットそれぞれに $x_i \cdot y_j$ あるいは $x_i \cdot (y_j \oplus d_1)$ を計算するための部分積生成セル（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 P_0，P_1 の生成回路を図 9 に示す．偶数の j について P_j 生成回路は P_0 と同じ構成をとり，奇数の j については P_j 生成回路は P_1 と同じ構成をとる．

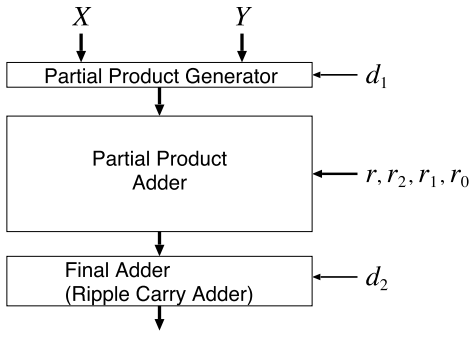

図 8　乗算器の構成
Fig. 8　Structure of a multiplier generated by the proposed method.

部分積加算部の葉の各 CSA には三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，$g \le f < g + N_x + 2$）と部分積生成部を次のように接続する．

$$\begin{aligned} u^f &\leftarrow pp_{(f-g)g} \\ v^f &\leftarrow pp_{(f-g-1)(g+1)} \\ w^f &\leftarrow pp_{(f-g-2)(g+2)} \end{aligned} \quad (2)$$

ただし，$i < 0$ あるいは $i \ge N_x$ についての pp_{ij} は **4.2** で述べる入力補完ユニットに接続する．外部入力信号 d_1 には通常時に 0 を入力し，テスト時に 1 とする．$d_1 = 1$，$X = 11\ldots1$ のとき，FA の入力ビットパターンは式 (1)，式 (2) より次の式で表される．

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

上の式より，$d_1 = 1$，$X = 11\ldots1$ としたとき，部分積 P_g，P_{g+1}，P_{g+2} を入力とする CSA に y_g，y_{g+1}，y_{g+2} の値に応じた交互反転パターンが入力されることが分かる．

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，y_g の値に応じた交互反転パターンを入力できる．したがって，**3.3** 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すべて入力できる．

4.2 部分積加算部の構成

各部分積は，ビット幅が等しく，最小の入力重みが

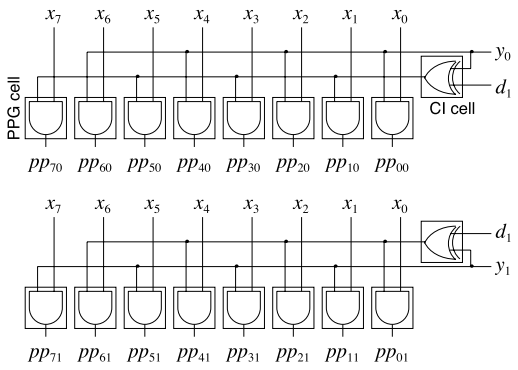

図 9　8×8 ビット乗算器の P_0，P_1 の生成回路
Fig. 9　Partial product generators for P_0 and P_1 in an 8×8 bit multiplier.

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けたでは入力ビットが二つ，あるいは一つになる．これらの部分では入力が交互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 の両端のけたにおいて入力ビットが 3 未満となる箇所がある．

このような入力ビットが 3 未満のけたについて，交互反転パターンの規則性を保つために入力ビットを補完して入力し，CSA の各けたで入力ビットが必ず三つになるようにする．これらの入力ビットを生成するため，図 10 のように各 CSA に以下で説明する入力ビット補完器（Input Padding Unit）を一つずつ接続する．入力ビット補完器は通常時には 0 を出力するため，通常時の演算結果に影響を与えない．

3. 3 では，根の加算ブロックに t_1 から $t_{\overline{5}}$ を入力するパターンをそれぞれ一つずつ設計し，テストのためのパターン集合を設計した．そのため設計したパターン集合において，部分積加算部の各 CSA の入力は，根の加算ブロックの入力パターンと対応関係をもつ．つまり，パターンにより根の加算ブロックに t_1 から $t_{\overline{5}}$ のどれが入力されるかが分かれば，各 CSA への入力が分かる．例えば図 7 では，根の加算ブロックにパターン t_1 が入力されると分かれば，レベル 2 の CSA の入力が $(0\ 1\ 1)_{ai}$ と分かる．そこで，外部入力線を用いて，入力パターンにより根のブロックに t_1 から $t_{\overline{5}}$ のどのパターンが入力されるかを与え，これらの外部信号線を用いて入力ビット補完器を設計する．

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，部分積加算部の根の加算ブロックへの入力パターンが $t_{\overline{z}}$ $(z \in \{1,2,3,4,5\})$ になるテストパターンのときだけ 1 を入力する．一方，テスト時に部分積加算部の根の加算ブロックへ t_z，$t_{\overline{z}}$ が入力される場合，r_0，r_1，r_2 には 2 進数 $(r_2\ r_1\ r_0)_2$ とみて z を入力する．つまり，根の加算ブロックへの入力パターンが t_4 となるテストパターンでは $(r_2, r_1, r_0, r) = (1, 0, 0, 0)$，$t_{\overline{1}}$ となるテストパターンでは $(r_2, r_1, r_0, r) = (0, 0, 1, 1)$ とする．通常動作時には r や r_1，r_2，r_3 は 0 とする．

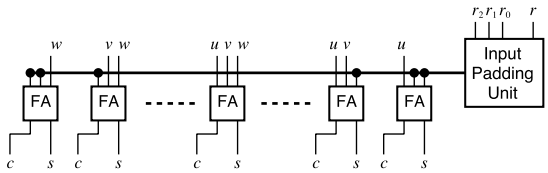

図 10　けた上げ保存加算器と入力ビット補完器
Fig. 10　A carry save adder and an input padding unit.

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，3 変数の論理関数を実現する．各 CSA の入力ビット補完器で PAD セルが実現する論理関数は異なり，設計時に静的に決める．PAD セルが実現する論理関数を PAD(Z) ($Z \subset \{0,1,2,\ldots,7\}$) の形で記述する．セルの入力 n_0，n_1，n_2 を 2 進数 $(n_2\ n_1\ n_0)_2$ とみたとき，Z 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．

入力ビット補完器の 2 ビット分を図 11 に示す．根の加算ブロックに t_{z_i} $(z_i \in Z_i \subset \{1,2,3,4,5\})$ 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_i)）を一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すべての 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．図中の各 PAD セルの機能は異なるため，PAD セルを PAD_0，PAD_1 と区別している．一つの入力ビット補完器は最大で 6 個の PAD セルを含む．

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．PAD セルに対する網羅パターンの入力については **4. 4** で述べる．

4. 3　最終加算部の構成

最終加算部として図 12 の順次けた上げ加算器（RCA）を用いる．RCA の各ビットの入力ビット端子に a，b と名前付けする．RCA の FA には最下位より 0 から順番に番号を付け，m 個目の FA に対応するビット位置の a，b 入力端子を a_m，b_m で表す．FA の

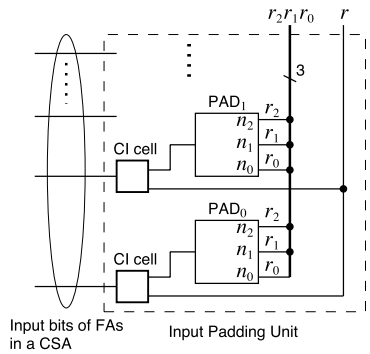

図 11　入力ビット補完器の構成
Fig. 11　An input padding unit.

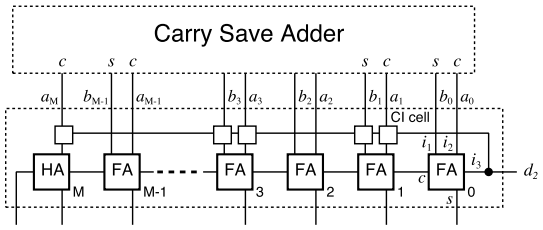

図 12　RCA を用いた最終加算部の構成
Fig. 12　Structure of the final adder with an RCA.

入力端子には i_1，i_2，i_3 と名前付けする．a，b 端子と奇数番目の FA の i_2，i_1 端子の間には CI セルを一つずつ挟む．テストのために外部入力 d_2 を導入し，CI セルの制御に用いる．通常時，d_2 の入力は 0 とする．

最終加算部の入力は部分積加算部の根の CSA の出力である．**3. 3** で示した部分積加算部のテストのためのパターンでは，部分積加算部内のすべての CSA に交互反転パターン 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部もテストする．

CSA に交互反転パターン p_0，p_7 をそれぞれ 2 回，p_1 から p_6 までをそれぞれ 1 回入力すると，出力も交互反転パターンで得られ，CSA の各 FA は出力端子 c，s に $(c, s) = (0, 0)$，$(1, 1)$ 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，最終加算部の各ビット位置で入力端子 a，b には $(a, b) = (0, 0)$，$(1, 1)$ が 3 回ずつ，$(0, 1)$，$(1, 0)$ がそれぞれ 2 回ずつ入力され，最終加算部の入力パターンも交互反転のパターンとなっている．最終加算部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 列目に最終加算部の最下位のビット位置での入力ビットパターンを示している．入力パターンは交互反転のパターンなので，最下位のビット位置の入力ビットパターンで最終加算部全体のパターンが一意に決まる．2 列目に外部入力 d_2 に入力する値を示している．3 列目は最終加算部の最下位の 0 個目の位置にある FA の入力を示し，4 列目では 2 個目以降の偶数個目の FA の入力，5 列目では奇数個目の FA の入力を示している．

表より，最終加算部のすべての FA，CI セルに網羅的にすべての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，最終加算部を構成するセルの故障は最終加算部の出力より確認できる．

表 4　最終加算部への入力パターン集合
Table 4　Input pattern set for the final adder.

	input values for the a_0b_0	d_2	Input values for $i_1i_2i_3$ of a FA		
			0-th	even numbered	odd numbered
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

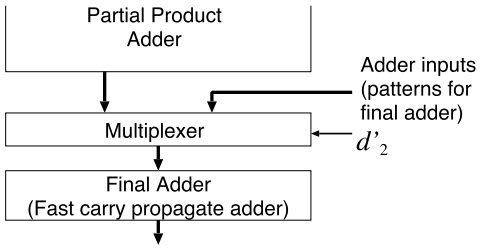

図 13　高速けた上げ伝搬加算器を最終加算部として用いた設計
Fig. 13　A design with a fast carry propagate adder as the final adder.

ここでは，最終加算部として遅延時間の大きい RCA を用いたが，[9], [10] 等のテストが容易で高速なけた上げ伝搬加算器を最終加算部として用いることも可能である．この場合，部分積加算部と最終加算部の間にマルチプレクサを挿入した図 13 の構成を用いる．最終加算部のテストは，マルチプレクサを外部入力端子 d'_2 で切り換え，テストパターンを入力して行う．

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交互反転パターンを用いる．**4. 1** で示した部分積生成部を用いることで部分積生成部で交互反転パターンを生成でき，**3. 3** で示した 10 個の交互反転パターンにより，部分積加算部をテストできる．また最終加算部として RCA を用いるとき，**4. 3** より，最終加算部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に，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，以下の四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．

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

この四つのパターンを先に述べた 10 個テストパターンとともに用いることで，部分積生成部の PPG セルや CI セル，PAD セルにすべ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部のための 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. むすび

乗算器の部分積加算部を 3 種類の加算ブロックの組合せで設計し，ブロックの組合せ方により様々なタイプの乗算器を設計できる C テスト可能な乗算器の設計手法を示した．提案手法で設計した乗算器は，最終加算部として順次けた上げ加算器を用いた場合，14 個のテストパターンでテストできる．

提案手法を用いることで，面積やスピードなどの要求に対して，性能に合わせたテスト容易な乗算器を設計できる．

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. Comput., vol.C-33, no.6, pp.554–560, June 1984.

[3] D. Gizopoulos, D. Nikolos, A. Paschalis, and C. Halatsis, “C-testable modified-booth multipliers,” J. Electron. Test., vol.8, pp.241–260, June 1996.

[4] K.O. Boateng, H. Takahashi, and Y. Takamatsu, “Design of C-testable modified-booth multipliers,” IEICE Trans. Inf. & Syst., vol.E83-D, no.10, pp.1868–1878, Oct. 2000.

[5] 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.

[6] 鬼頭信貴，花井健輔，高木直史，“4-2 加算木を用いたテスト容易な乗算器，” 信学技報，VLD2006-140，March 2007.

[7] Z.-J. Mou and F. Jutand, “Overturned-stairs adder trees and multiplier design,” IEEE Trans. Comput., vol.41, no.8, pp.940–948, Aug. 1992.

[8] S.D. Pezaris, “A 40-ns 17-bit by 17-bit array multiplier,” IEEE Trans. Comput., vol.C-20, no.4, pp.442–447, April 1971.

[9] R.D. Blanton and J.P. Hayes, “Testability of convergent tree circuits,” IEEE Trans. Comput., vol.45, no.8, pp.950–963, Aug. 1996.

[10] D. Gizopoulos, M. Psarakis, A. Paschalis, and Y. Zorian, “Easily testable cellular carry lookahead adders,” J. Electron. Test., vol.19, pp.285–298, June 2003.

（平成 20 年 3 月 4 日受付，5 月 8 日再受付）

鬼頭　信貴　（学生員）

平 16 名大・工・電気電子情報工卒，平 18 同大大学院情報科学研究科博士前期課程了．現在，同大学院情報科学研究科博士後期課程在学中．テスト容易化設計，算術演算回路等の研究に従事．

高木　直史　（正員）

昭 56 京大・工・情報工卒，昭 58 同大大学院修士課程了，昭 63 京大工博．昭 59 京大・工・情報工助手，平 3 同助教授，平 6 名大・工・情報工助教授，平 10 同教授，平 15 名大・情報科学・教授．算術演算回路，ハードウェアアルゴリズム，論理設計等の研究に従事．平 7 日本 IBM 科学賞，情報処理学会坂井記念特別賞等受賞．