速度・面積のトレードオフに適した多重波面型乗算器の構成と設計パラメータ

田丸啓吉[†] 品川正行[†,☆] FARHAD FUAD ISLAM[†,☆☆]

ASICの高位合成では，使用する機能ユニットとしてライブラリにあるユニットの中から選択する方法が使用されているが，つねに最適なユニットが存在するとは限らない．本論文では乗算器を取りあげ，速度仕様を満足する範囲でレイアウト面積の最小になるような，速度・面積トレードオフのとれる乗算器のアーキテクチャとその設計法について考察する．乗算器のアーキテクチャはこれまで高速で面積の大きいツリー型と低速で小面積のアレイ型に大別されてきたが，本論文では両者を両極端として含む，より一般的な乗算器として提案されている多重波面型乗算器の構成を検討し，演算時間の短縮化の改良をした改良多重波面型乗算器を新しく考案した．この乗算器では新しく階層構成を採用し従来の構成が1階層であったものを多階層に改良して高速化を可能にした．さらに高速乗算器で使用されているBoothのアルゴリズムに基づく構成に拡張し，より柔軟性のある乗算器の構成を実現する方法を示した．これらの構成をもとに，共通する精算用加算回路を除いた各乗算器の部分積加算部の演算時間と面積を比較評価し，基本演算回路の外部入力数 m を設計パラメータにして，最適構成を求めることができることを示している．その結果高位レベル自動設計における最適乗算器を生成する設計手法を提供することが可能になった．

Structure and Design Parameters of the Multiple Wave Front Multiplier Realizing Optimum Speed-area Trade off

KEIKICHI TAMARU,[†] MASAYUKI SHINAGAWA[†,☆] and FARHAD FUAD ISLAM[†,☆☆]

The Design of ASIC employing high level synthesis method uses function units selected from its library. But this method does not assure the selection of an optimum unit in speed and layout size. This paper describes the architecture and its design method of a speed-area trade off multiplier. Basic architecture of a high speed LSI multiplier is classified into two types, tree type and array type. As a new architecture of multiplier including wide performance range from tree type to array type, the modified multiple computation wave fronts configuration is proposed using the hierarchical structure of basic execution units. Then this configuration is extended to the well known Booth algorithm for high speed multiplication. The characteristics of these speed-area trade off multipliers is investigated. As a Design parameter, the number of inputs in an execution unit is selected and the optimum configuration of a multiplier which satisfies the design specification can be obtained using the value of this parameter.

1. まえがき

近年のASIC設計技術の進歩により，高位合成と呼ばれる機能レベルの自動設計手法が注目されている[1]．この設計では計算アルゴリズムをデータフローグラフに展開し，RT（レジスタトランスファ）レベルの構成を求めるもので，その中に加算器や乗算器などの機能ユニットを使用する．従来の手法ではこれらの機能ユニットは一定の形と性能のものがライブラリに登録されており，これを使用している[2]．しかしこの方法では上位の設計結果として与えられる要求仕様にライブラリのユニットが適合するとは限らない．一般的にはライブラリの適用範囲を広げるため，高性能のユニットが登録されており，オーバスペックでもこれを使用

† 京都大学工学部
Faculty of Engineering, Kyoto University
☆ 現在，シャープ株式会社
Presently with SHARP CORPORATION
☆☆ 現在，ヒューレットパッカード日本研究所
Presently with HEWLETT PACKARD Co.

せざるを得ない場合が多い．

しかしASICの設計をより精度の高い設計にするためには，オーバスペックを減らし，設計パラメータ間のトレードオフを最適にする必要がある．このような考えから，本論文では乗算器を取りあげ，性能（速度）仕様を満足する範囲でレイアウト面積の最小になるような，速度–面積トレードオフのとれる設計について考察する．最も基本的な乗算器のアーキテクチャはツリー型とアレイ型に分類される．ツリー型乗算器は[3]，ビット部分積を並列に加算する方式で，最も高速のアーキテクチャとして広く使用されている．しかし一般にツリー型は構造が不規則で，レイアウトしたとき面積が大きくなる傾向がある[4]．アレイ型乗算器は基本ユニットを規則的に配置した構造をしており，相互配線も規則的でコンパクトなレイアウトが可能であるが，信号の伝搬する基本ユニット数が大きくなるため動作速度の点でツリー型に劣る欠点がある[5]．しかしその規則的な繰返し構造がVLSI化に適しているため，多くのプロセッサで採用されてきた．これらの基本的な形式の欠点を改良するため，ツリー型乗算器の配置を規則的にする研究やアレイ型乗算器の動作速度の向上の研究[6)~10)]が数多く行われてきた．筆者らは従来のツリー型もアレイ型も含むより一般的な乗算器の構成として，分割アレイ構成を提案した[11]．アレイ構造は，アレイ上部から下部に向かって演算された信号が伝搬していくので，これを信号の波面と考えると，分割アレイ構造は多重波面型構成と呼ぶことができる．文献11)では m 個の多重波面型構成とすることにより，アレイ部の信号伝搬遅延時間を m 分の1にする構成手法が提案されている．

本論文は従来の多重波面型構成が1階層であったものを多階層に改良してより高速化を可能にするとともに，Boothのアルゴリズムに基づく構成に拡張し，より構成の柔軟性を実現した乗算器について述べている．本構成の乗算器により，各種の専用プロセッサの設計において，必要な動作速度に合致した最適乗算器をきめ細かく設計することが可能になる．その結果将来の自動設計において，最適性能の乗算器を自動生成するための設計手法を提供することができる．

本論文では，第2章で改良多重波面型アーキテクチャについて述べ，第3章でBoothのアルゴリズムを使用した構成に改良多重波面型アーキテクチャを拡張した場合について説明する．第4章で本アーキテクチャの乗算器の評価と設計例について説明する．

2. 改良多重波面型（MWF）アーキテクチャ

従来の多重波面型アーキテクチャについて簡単に紹介する[11]．n ビットの符号なし整数の乗算を考える．乗算器は $n \times n$ 個のビット部分積（以下PPBと記す）の和をとるコア部分とコアで生成した各桁の和と桁上げを精算する高速加算器CLAから構成される．いまある桁のPPBを上から m 個ごとに分けて，1列の m 個のPPBを加算する部分を単位計算ブロック（UCB）と呼ぶことにする．したがって $n \times n$ 個のPPBはUCBのアレイによって加算されることになる．UCBは**図1**の x 軸方向に $\lceil n/m \rceil$ 個（$\lceil x \rceil$ は x より大きく，最も x に近い整数を示す），y 軸方向に $(2n-1)$ 個のアレイを構成する．UCBは内部で加算されるPPBの最も小さい (x, y) 座標で表すことにする．各UCBは m 個のPPBを加算するため**図2**に示すような m 個の全加算回路（FA）を直列に接続した構成をしている．各FAの和出力は隣りのFAに入力され，最終段のFAからのみUCBの和出力として外部へ出力される．一方，各FAの桁上げ出力は並列に外部へ出力される．PPB入力は初段のFAのみ2個，終段は0，その他の中間のFAは1個が加えられる．また桁上げ入力はすべてのFAに並列に入力される．同じ y 座標のUCBの和出力は上から下へ順次直列的に加算され，最後にその桁の和出力がつくられる．桁上げ出力はUCBの中のFAごとに，上位桁（y 座標の1

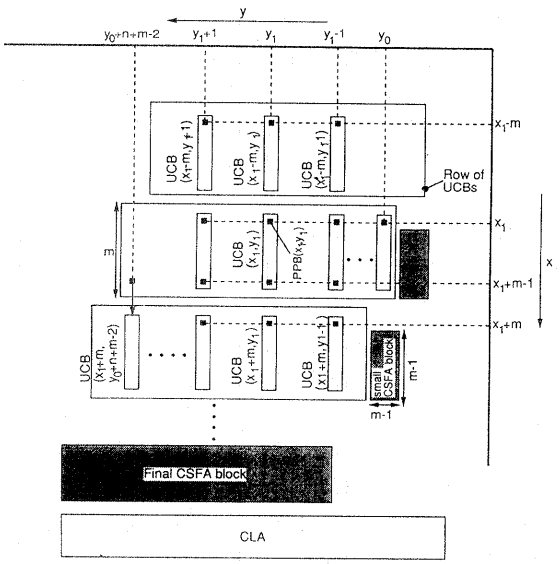


図1 多重波面型乗算器の構成

Fig. 1 Block diagram of the multiple wave front (MWF) multiplier.

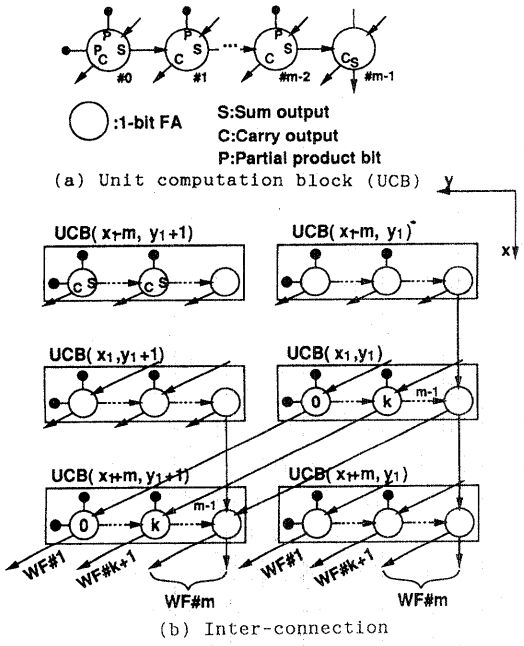


図2 UCBの構成と接続
Fig. 2 Structure and interconnection of UCB.

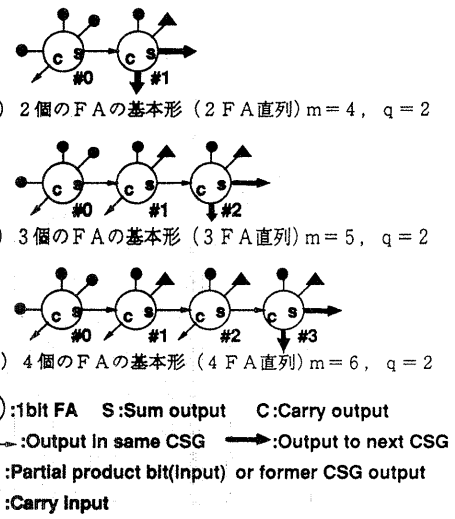


図3 演算回路ECの構成
Fig. 3 Structure of basic execution circuits EC.

大きい）の一つ下の行（x 座標の m 大きい）のUCBの対応するFAに加えられる．このため一番上の行のUCBの各FAから左下に向かって m 個の桁上げ信号がFA1段の遅延時間の差を持って次々に生成されて伝搬していくことになる．これはちょうど m 個の波面が時間差を持って次々に伝搬していくことに似ているので，多重波面型（Multiple Wave Front type, MWF）と名付けたものである．$(n+m-1)$ 個の最下段UCBアレイからは，各UCBにつき $(m+1)$ 個の出力が出る．これを加算するため桁上げ保存全加算器アレイ（CSFA）ブロックを設ける．このアーキテクチャの最大伝搬遅延時間は，UCBアレイ，CSFA，CLAの伝搬時間の和として得られ，文献11)より次式のようになる．

$$T=(m+\lceil n/m\rceil-1)T_{\mathrm{FA}}+HT_{\mathrm{FA}}+T_{\mathrm{CLA}} \quad (1)$$

ここで T_{FA} は全加算回路の遅延時間，T_{CLA} は高速加算器の遅延時間，H は $m=2$ のとき $H=1$，$m>2$ のとき $H=2$ である．また $\lceil X\rceil$ は X より大きく最も X に近い整数を表す．

上述のMWFアーキテクチャには，動作速度の向上の点で大きな制約がある．それは同じ y 座標の各UCBの和出力が直列に加算されるため，下段（x の座標の大きい）のUCBほど上段からの和出力の到着が遅くなり出力が遅れることである．この点を改良するため，計算ステージという概念を導入して多階層の構成をとるような新しい構成を考案した．計算ステージは同一桁位置の同じレベルの入力を加算して出力する1段分の演算回路の集まりである．信号のレベルはPPB入力が何段の演算回路を通過したかを示す最も小さい数で，レベル0はPPBそのもの，レベル1は1段の演算回路を通った後の信号を意味する．計算ステージは入力数より出力数が減少する性質を持つので，多段の計算ステージを通ると出力数は次第に減少し，最終段では通常の高速加算回路の入力となる2出力まで減る．前述のMWFアーキテクチャは1計算ステージの構成をしている．

次に計算ステージ（CSG）を使用した改良MWFアーキテクチャを説明する．基本演算回路としてFAを組み合わせた演算回路EC（従来のUCBに対応する）を考える．構成するFAの数 k，入力数を M，出力数を Q とすると $M=2k+1$，$Q=k+1$ となるが，この入出力信号数のうち同じ計算ステージの演算回路間で結線される桁上げ信号を除いた外部および計算ステージ間の入力数を m，次計算ステージへの出力数を q とすると，$m=M-(k-1)=k+2$，$q=2$ となる．このECの信号伝搬遅延時間を T_{EC} とすると $T_{\mathrm{EC}}=kT_{\mathrm{FA}}$ となる．ECを特徴づけるパラメータは m，q，T_{EC} である．最も簡単なECの構成は1個のFAで構成する場合である．これは $m=3$，$q=2$，$T_{\mathrm{EC}}=T_{\mathrm{FA}}$ の回路である．より大きいECはFAを接続して構成する．使用するFAは同じものを考える．**図3**に示すようにFAを直列に接続するとよ

り大きなECが構成できる．2個のFAでは同図 (a) のように $m=4$, $q=2$, $T_{EC}=2T_{FA}$ のECになる．さらに3個のFAでは同図 (b) のように $m=5$, $q=2$, $T_{EC}=3T_{FA}$，4個のFAでは同図 (c) のように $m=6$, $q=2$, $T_{EC}=4T_{FA}$ のECが構成できる．これらのECの構成は，初段のFA（番号0）にはすべて外部（PPBおよび前段の計算ステージ）からの入力を加え同時に動作可能とする，終段のFA（番号 $k-1$）は前段FAの和出力と1個の外部入力および桁上げ入力を入力し，和出力と桁上げ出力の2出力を次ステージに出す，中間のFAは接続されている前のFAの出力のほかに1個の外部入力と桁上げ入力を受け入れ，和出力を接続している後のFAに，桁上げ出力を同じステージの次の桁のECに出力するという原理によっている．**図4**に同一ステージおよび異なるステージのEC間の結線を示す．32個のPPBを入力とし，ECは3個のFAで構成されている場合の例である．第1計算ステージではECの各FAは入力としてPPBおよび下位ビット側のECのFAの桁上げ出力を入力し，出力として上位ビット側のECにつながる桁上げ出力または第2計算ステージのECにつながる和出力および桁上げ出力を出す．第2計算ステージのECは，第1計算ステージのECが出す和出力と下位ビットからの桁上げ信号のうち5個（図4の場合は3個の和出力と2個の桁上げ出力）と2個の第2計算ステージの下位ビット位置のECの出す桁上げ出力を入力として受け入れ，2個の同じ計算ステージ内の桁上げ信号と次の計算ステージへ出す和出力と桁上げ出力を出す．第2計算ステージのECの出す次の計算ステージへの和出力と桁上げ出力は第3計算ステージのECに入力される．

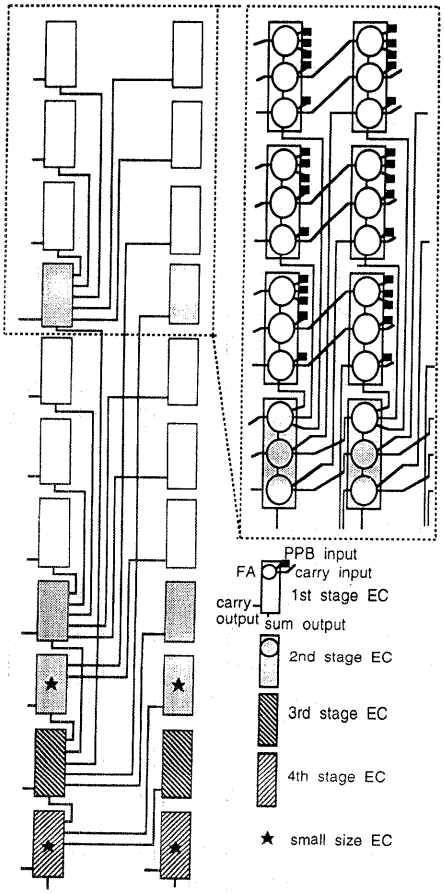


図4 EC間の接続の例（$n=32$, $m=5$, $k=3$, 4ステージ構成の場合）
Fig. 4 Inter EC connection ($n=32$, $m=5$, $k=3$, 4 stages).

いまビット位置 y_1 の第1計算ステージの入力数を n_1 とする．n_1 はこのビット位置のPPBの数である．入力数 n_1 を m で割り G_1 個のECに分割する．割り切れない場合には剰余分を NG_1 とする．このとき

$$n_1 = mG_1 + NG_1$$

が成り立つ．第1計算ステージの出力数は各ECより和と桁上げの2個 ($q=2$) の出力がある．ここで $NG_1 \neq 0$ のとき出力数は NG_1 のため $2G_1$ より増えることになる．ECは2出力であるから $NG_1=1$, 2のときには特に NG_1 を第2計算ステージの入力として直接利用するようにする．$NG_1 \geq 3$ のときEC1個を追加するが，追加されるECは NG_1 の数により内部の冗長なFAを省略した縮小ECとする．

第2計算ステージの入力数 n_2 は，第1計算ステージのECの和出力と下位ビット位置のECの桁上げ出力によってきまる．和出力は G_1 個または G_1+1 個（縮小ECのある場合）で，ほかに NG_1 個の直接入力がある場合がある．桁上げ入力は G_1 個または G_1+1 個（ビット位置により決まる）になる．したがって n_2 は

$$n_2 = \begin{cases} 2G_1 & NG_1 = 0 \\ 2G_1 + NG_1 & NG_1 = 1, 2 \\ \left.\begin{matrix} 2G_1+1 \\ 2(G_1+1) \end{matrix}\right\} & NG_1 \geq 3 \end{cases}$$

の場合がある．図4の例は $n_1=32$，$m=5$ の場合で，$G_1=6$, $NG_1=2$，したがって第1ステージ出力は $2G_1+NG_1=14$，第2ステージ入力は和入力6，$NG_1=2$，桁上げ入力6，したがって $n_2=14$ となり，1個の縮小ECを含めて3個のECで構成され

ている．

第2計算ステージでは第1計算ステージと同様に入力数 n_2 より EC 数 G_2 と NG_2 を計算し，G_2 個の EC と1個の縮小 EC（ない場合もある）を結線して入力を加算する．各 EC の次計算ステージへの出力は第3計算ステージの入力となる．この構造をステージの EC 数が1個（縮小 EC の場合も含む）になり，計算ステージの出力数の合計が2になるまで繰り返すとビット位置 y_1 の構造が得られる．同様の構造を他のビット位置についても求める．

このアーキテクチャの遅延時間は次のようになる．EC の遅延時間を T_{EC}，最大の PPB のある桁の計算ステージ数を h とすると，アレイ部の全遅延時間 T_{AR} は

$$T_{\mathrm{AR}} = \sum_{i=1}^{h} \cdot T_{\mathrm{EC}_i}$$

となる．いま EC の構成として k 個の FA を直列に接続した場合を考えると，$T_{\mathrm{EC}} = kT_{\mathrm{FA}}$ となる．実際は最終計算ステージが縮小 EC になる可能性があるので遅延時間は次式のようになる．

$$T_{\mathrm{EC}_i} = \begin{cases} kT_{\mathrm{FA}} & n_i/m \geq 1 \\ (n_i - 2)T_{\mathrm{FA}} & n_i/m < 1 \end{cases} \quad (2)$$

ただし i はステージの番号で $i = 1, 2, 3, \ldots h$

したがってアレイ部の遅延時間 T_{AR} は

$$T_{\mathrm{AR}} = \sum_{i=1}^{h} T_{\mathrm{EC}_i} = \sum_{i=1}^{h-1} KT_{\mathrm{FA}} + (n_h - 2)T_{\mathrm{FA}}$$
$$= (h-1)(m-2)T_{\mathrm{FA}} + (n_h - 2)T_{\mathrm{FA}} \quad (3)$$

T_{AR} に高速加算回路の遅延時間 T_{CLA} を加えて，全体の演算時間 T は

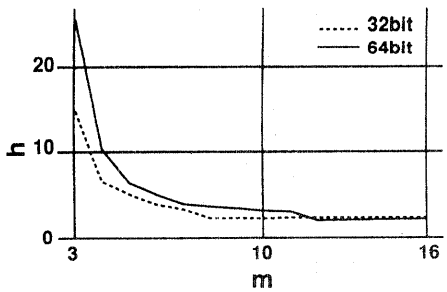


図5 改良型が高速になる限界の計算ステージ数 h
Fig. 5 Number of stages h in case that a modified MWF can get higher speed.

$$T = T_{\mathrm{AR}} + T_{\mathrm{CLA}}$$
$$= [h(m-2) - (m - n_h)]T_{\mathrm{FA}} + T_{\mathrm{CLA}} \quad (4)$$

となる．この式を式 (1) と比べると，

$$h(m-2) - (m - n_h) < (m + \lceil n/m \rceil - 1 + H) \quad (5)$$

が成立すれば，本アーキテクチャの方が高速になる．$m \geq 3$ の範囲では $H = 2$ となるから，次式のようになる．

$$h < [(m + \lceil n/m \rceil + 1) + (m - n_h)]/(m-2) \quad (6)$$

図5 に式 (6) の右辺すなわち本アーキテクチャが従来の MWF アーキテクチャより高速になる限界の h の値を示す．与えられた n に対してこの曲線より h が小さいか等しければより高速にできる．実際に設計した場合の h の値（**表1** に示す）を比べると，実用的な m の値（例えば $m \leq 8$）の範囲で h は曲線上かその下にある．m の値が大きくなると，この条件をは

表1 改良 MWF 乗算器の設計パラメータ
Table 1 Design parameters of the modified MWF multiplier.

$n = 32$

m	k	h	$G_i + (P_i)$	T_{EC}	T_{AR}
3	1	8	10^{++}, 7^{+}, 5, 3^{+}, 2^{+}, 1^{++}, 1^{+}, 1	1	8
4	2	4	8, 4, 2, 1	2	8
5	3	4	6^{++}, 2+(2), 1^{+}, (1)	3	10
6	4	3	5^{++}, 2, (2)	4	10
7	5	3	4+(2), 1+(1), (2)	5	12
8	6	2	4, 1	6	12
10	8	2	3^{++}, (6)	8	14
12	10	2	2+(6), (4)	10	14
16	14	2	2, (2)	14	16
32	30	1	1	30	30

$n = 64$

m	k	h	$G_i + (P_i)$	T_{EC}	T_{AR}
3	1	10	21^{+}, 14^{+}, 9^{++}, 6^{++}, 4^{++}, 3^{+}, 2^{+}, 1^{++}, 1^{+}, 1	1	10
4	2	5	16, 8, 4, 2, 1	2	10
5	3	5	12+(2), 5^{+}, 2^{+}, 1	3	12
6	4	4	10+(2), 3+(2), 1^{++}, (2)	4	14
7	5	3	9^{+}, 2 + (3), (4)	5	14
8	6	3	8, 2, (2)	6	14
9	7	3	7^{+}, 1+(4), (2)	7	16
10	8	3	6+(2), 1+(2), (2)	8	18
12	10	2	5+(2), 1	10	20
16	14	2	4, (6)	14	20
32	30	2	2, (2)	30	32
48	46	2	1+(14), (2)	46	48
64	62	1	1	62	62

$G_i + (P_i)$ で 0 の項は省略
T_{EC}, T_{AR} は T_{FA} で規格化

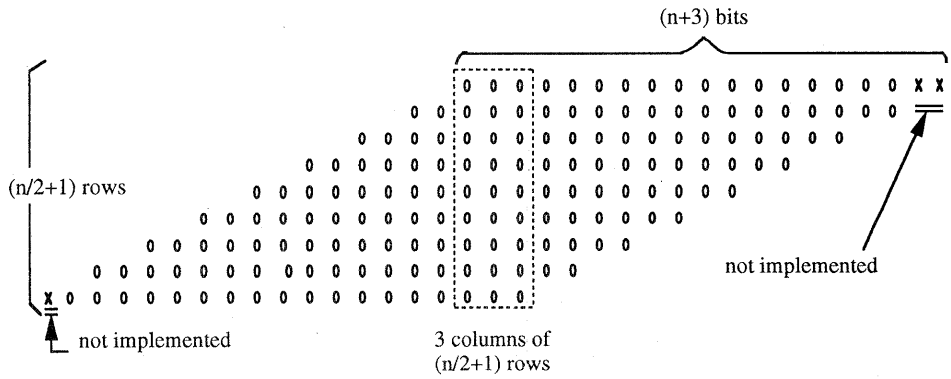


図6 Sign extention prevention による Booth のアルゴリズムを用いた乗算器の原理
Fig. 6 Principle of Booth algorithm multiplier using sign extention prevention.

ずれる場合があるが一般的には改良 MWF アーキテクチャの方が高速であることが分かる．$n=32$，64 ビットに対して改良 MWF 乗算器の構成を与える設計パラメータを表 1 に示す．

表の中で m は EC の外部入力数，k は EC の中の FA の数，h は計算ステージ段数，G_i は各計算ステージの EC の数，(P_i) は各計算ステージの縮小 EC が含む FA 数を示す．肩の $+$ の印は $NG=1$，$++$ 印は $NG=2$ であることを示している．T_{EC} は 1 個の EC の遅延時間（FA の遅延時間で規格化してある），T_{AR} は式 (3) によるアレイの遅延時間（規格化した値）を示したものである．

3. Booth のアルゴリズムを使用した乗算器への拡張

乗算器の高速化に使用されている 2 次の Booth のアルゴリズムを使用した乗算器へ，改良 MWF アーキテクチャを拡張することを考える．Booth エンコーダからは正数の他に負数も出力されるため，符号部分の加算器数が増える．これを避けるため sign extension prevention の方法[12]を使用する．いま乗数，被乗数を n ビット（n は偶数）とすると，アレイ部分の構成は**図 6** に示すように加算器の段数が $(n/2)+1$ 段となる．追加される 1 段は負数に対する符号拡張部を表す数値を加算するための加算器である．また各段の加算器数は，負数をとるための最下位桁の 1 加算，その桁からの桁上りの加算，正数に対する符号ビットの補正加算を考慮して，第 1 段は $n+1$ 個，最下段は $n+2$ 個，その他は $n+3$ 個となる．最も加算項数の多い桁では $(n/2)+1$ 個の加算を行うから，これを改良 MWF 乗算回路にあてはめると，乗数のビット数が $(n/2)+1$ の場合と同じになる．**表 2** は表 1 と同じ方法で計算した $n=32$，64 ビットに対する Booth のアルゴリズムを使用した乗算器の設計パラメータである．

表 2 Booth のアルゴリズムを用いた MWF 乗算器の設計パラメータ
Table 2 Design parameters of the Booth algorithm MWF multiplier.

$n=32$

m	k	h	$G_i+(P_i)$	T_{EC}	T_{AR}
3	1	6	5^{++}, 4, 2^{++}, 2, 1^{+}, 1	1	6
4	2	4	4^{+}, 2^{+}, 1^{+}, (1)	2	7
5	3	3	3^{++}, 1+(1), (2)	3	8
6	4	2	2+(3), 1	4	8
8	6	2	2^{+}, (3)	6	9
16	14	2	1^{+}, (1)	14	15
17	15	1	1	15	15

$n=64$

m	k	h	$G_i+(P_i)$	T_{EC}	T_{AR}
3	1	8	11, 7^{+}, 5, 3^{+},2^{+} 1^{++}, 1^{+}, 1	1	8
4	2	5	8^{+}, 4^{+}, 2^{+}, 1^{+}, (1)	2	9
5	3	4	6+(1), 2+(2), 1^{+}, (1)	3	10
6	4	3	5+(1), 2, (2)	4	10
7	5	3	4+(3), 1+(1), (2)	5	12
8	6	3	4^{+}, 1^{+}, (1)	6	13
10	8	2	3+(1), 6	8	14
16	14	2	2^{+}, (3)	14	17
32	30	2	1^{+}, (1)	30	31
33	31	1	1	31	31

$G_i+(P_i)$ で 0 の項は省略
T_{EC}，T_{AR} は T_{FA} で規格化

4. 評価と設計例

4.1 遅延時間の評価

図7は m と T_{AR} の関係を示したもので，非Boothのカーブが改良MWF乗算器（以下非Boothと記す），Boothのカーブが Boothのアルゴリズムを使用した乗算器（以下 Boothと記す）の場合の遅延時間である．m が増加するとともに遅延も増加することが分かる．m のある値で T_{AR} が下がる現象は，m が変わるとき各ステージのECの中の直列FA数と並列になるEC数の関係が変わることにより，全体の遅延時間 T_{AR} が小さくなるために生じるものである．Boothではツリーの場合（$m=3$）にFA2段分の時間短縮になる．m が変わると m によって32ビットでは1～4段，64ビットでは1～6段の短縮になる．しかしBoothのエンコーダの遅延時間 $1.57T_{FA}$（回路図より計算）を加えると，実際の時間短縮は32ビットで最大2.4段，64ビットで4.4段になる．m の値によっては0.6段分逆転する場合がある．

4.2 面積の評価

次に面積の計算について説明する．面積は配置配線手法により異なるので，ここでは簡単なモデルを使用する．このモデルは次の仮定による．

(1) 乗算回路の各ビット位置の配置配線は同じ形式で行う．全体の面積は n 個のPPBのある中央のビット位置の面積 S に比例するとし，この面積について評価する．CLAは m に関係なく一定であるから除外する．

(2) 面積はFAの面積を基準として正規化する．FAは正方形とし，面積 A_{FA} は $A_{FA}=1$ とする．

(3) EC内部の直列に接続されているFAは，1列に密着して配置される．

(4) ECの周囲に入出力線のEC近傍分の配線および同一計算ステージ内の桁上げ配線のための領域をとる．この配線領域はECの中のFAの1辺あたり A_w とする．ただし密着している辺については配線領域をとらない．$A_w=\alpha A_{FA}$ とする（$0<\alpha<1$）．

(5) ECの外部入力と最終ステージ出力および計算ステージ間の信号の配線領域として信号線数に比例する配線領域をとる．配線面積は配線1本あたり βA_{FA} とする（$0<\beta<1$）．

以上の仮定より計算ステージの全ECの面積 S_{EC} は次式のようになる．

$$S_{EC}=\sum_{i=1}^{h}[G_iA_{EC}+A_{p_i}] \tag{7}$$

ここに A_{EC}は1個のECの面積（配線領域を含む）で

$$\begin{aligned}A_{EC}&=kA_{FA}+(2k+2)A_w\\&=kA_{FA}+(2k+2)\alpha A_{FA}\end{aligned} \tag{8}$$

となる．A_{p_i} は i 段目の縮小ECの面積で，その中のFA数を P_i としたとき

$$A_{p_i}=\begin{cases}P_iA_{FA}+(2P_i+2)\alpha A_{FA} & p_i\neq 0\\ 0 & p_i=0\end{cases} \tag{9}$$

となる．h は計算ステージの段数である．次に計算ステージ間の配線領域について説明する．

ECのステージ間結線による配線面積 S_w は，ECの外部入力および前段ステージからの入力本数と，最終段ステージから出る2本の出力に比例する面積となる．

$$S_w=\left(\sum_{i=1}^{h}n_i+2\right)\beta A_{FA} \tag{10}$$

ここに

$$n_i=\begin{cases}n & i=1\\ 2G_{i-1}+NG_{i-1} & i=2,\ldots,h\ \text{かつ}\\ & NG_{i-1}=0,1,2\\ 2(G_{i-1}+\gamma_{i-1}) & i=2,\ldots,h\ \text{かつ}\\ & NG_{i-1}\geq 3\end{cases}$$

$$\gamma_{i-1}=\begin{cases}0 & P_{i-1}=0\\ 1 & P_{i-1}\neq 0\end{cases}$$

したがって中央ビット位置の合計面積 S は次式の

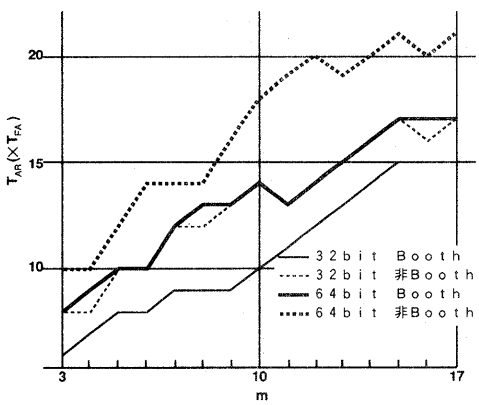


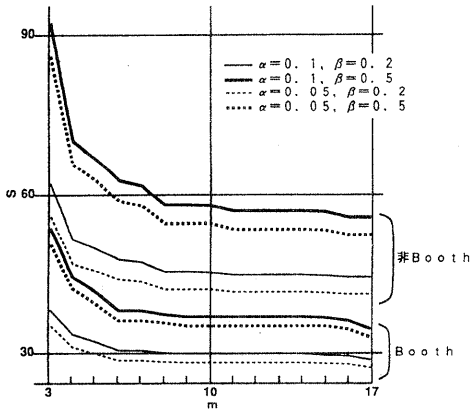
図7 アレイ部の遅延時間 T_{AR}
Fig. 7 Latency time T_{AR} of array part.

ようになる．

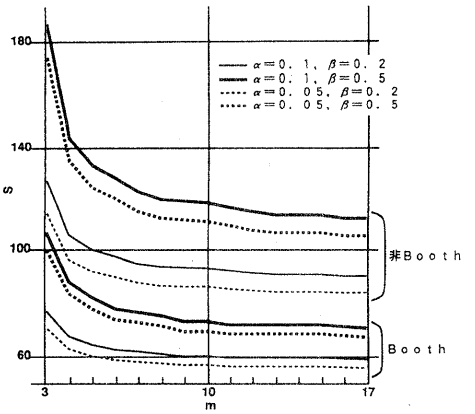
$$S = S_{EC} + S_w$$
$$= \sum_{i=1}^{h} [G_i\{k + (2k+2)\alpha\}A_{FA}]$$
$$+ \sum_{\substack{j=1 \\ p_j \neq 0}}^{h} [\{P_j + (2P_j+2)\alpha\}A_{FA}]$$
$$+ \left[\left(\sum_{i=1}^{h} n_i + 2\right)\beta A_{FA}\right] + S_B \qquad (11)$$

ただし S_B は1ビット分のBoothのエンコーダの面積で，非Boothの場合には $S_B = 0$ とする．

面積の比較には $A_{FA} = 1$ とした正規化面積で比較する．$n = 32$, 64ビットについて式(11)を計算した正規化面積と m の関係を**図8**(a)（32ビット）と(b)（64ビット）に示す．Boothでは，Boothのエンコーダの面積をレイアウトより求め，$S_B = (0.87/n + 0.39)A_{FA}$ として加算してある．計算は表1，表2によって行い，パラメータとして $\alpha = 0.05, 0.1$, $\beta = 0.2, 0.5$ と変えて計算した．この図より正規化面積は $m = 3$ の場合が最も大きく，m が増加するにしたがって減少し，$m = n$ （枠外になるので明示されてない）で最も小さい値になる．$m = 3$ はツリー型であり，$m = n$ はアレイ型に相当する．使用するFAの個数は一定であるから，m の値によって変化する面積は配線面積に相当する部分である．ツリー型の配線は一般的に複雑で配線面積がアレイ型より大きくなることが，このカーブでも示されている．特に β の値が大きい場合に配線面積の比率が大きくなるためこの傾向が顕著に見られる．しかし面積の減少は m が10以上で非常にゆるやかになるので，実用上有効な範囲は m が3～10の範囲である．



(a) 32 bits



(b) 64 bits

図8 乗算器1行分の正規化面積
Fig. 8 Normalized area of 1 bit column of multiplier.

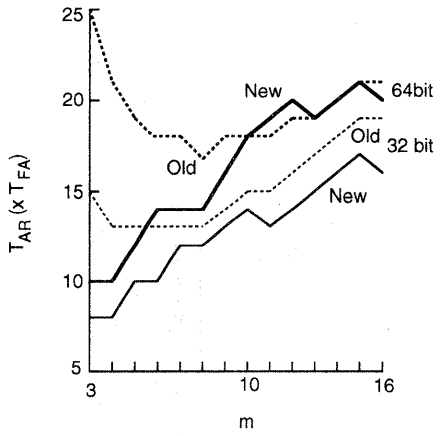
乗算器の面積 S_M は，アレイの構造に従って S を累積したアレイ部の面積とCLAの面積を加算して得られる．アレイの構造は，改良MWF乗算器では中央に n 個のFAがある単純な形になるが，Boothの乗算器では図6に示すようにFAの総数は $[(n+3)\times(n/2+1)-3]$ 個であり，その配列は中央に $(n/2)+1$ 個の列が3列，その左右に2列ずつ $n/2$ 組の列が三角形にできる．ただし第1行の右端に2個，最終行の左端に1個の空席ができる．改良MWF乗算器の中央の1列の面積を S，Boothの乗算器の中央の1列の面積を S' とすると，近似的に次式が得られる．

$$S_M \begin{cases} = nS + S_{CLA} \\ \qquad \text{（改良MWF乗算器）} \\ = [S'(n+3) - 3A_{FA}] + S_{CLA} \\ \qquad \text{（Booth乗算器）} \end{cases} \qquad (12)$$

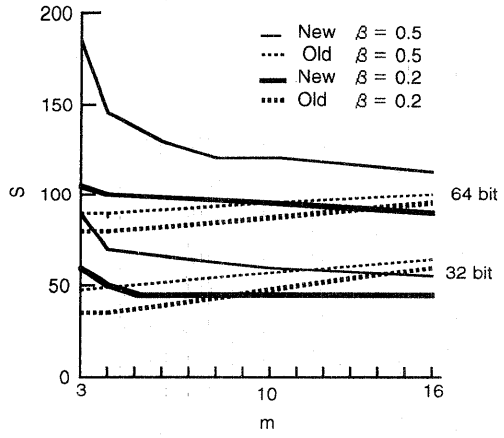
ここで S' は $n/2+1$ のときのSに相当し，近似的には $S/2$ に近い．また S_{CLA} はCLAの面積である．

4.3 従来型MWFとの比較

改良MWFを従来型MWFと比較した結果を**図9**に示す．アレイ部の遅延時間については従来型は式(1)より T_{CLA} を除いた値を使用する．これはアレイ部とCSFA部を含む値である．図9(a)に示すように改良型は32ビットでは従来型より遅延時間は小さく，64ビットでは m の値が9以下の時に従来型より小さくなる．このことより m の実用的な範囲である3～10の範囲内で，m の値が小さいほど改良型の方が高速である．次に面積について比較する．前と同じモデルのもとに，$\beta = 0.2$ と 0.5 について比較した結果を図9(b)に示す．従来型はCSFA部を含む値である．従



(a) Latency time



(b) Normarized area

図9 改良型と従来型MWF乗算器の比較（アレイ部の遅延時間と1行分の面積，従来型はCSFA部を含む）

Fig. 9 Comparison of new and old MWF multiplier (latency time and 1 column area, old type includes CSFA part).

来型のUCBとCSFAに含まれるFAの総数と改良型のECに含まれるFAの総数は，前者の方が$(m+1)$個多い．これはCSFAに$(m-1)$個必要であることによる．一方，配線面積はUCB間の桁上げ配線とステージ内のEC間桁上げ配線が同程度と考えると，大幅に異なるところは改良型のステージ間配線の増加である．配置のやり方により細部は異なるが，概略のところステージ間配線は1個のECの外部入力数に相当するm本と横を通過する配線数の和になり，$m+1$本から$2m$本程度になる．従来型では同じビット位置のUCB間は1本，CSFAに入る配線が$m+1$本であるから，配線面積は改良型の方が1.2～2倍程度大きくなる．その差はmが大きくなるほど減少する．図でnが小さく，nに対してmが大きい場合には改良型の方が従来型より面積が小さくなる場合がある．これは改良型のECの数が少なくなりステージ間配線が簡単になるのに対し，従来型ではCSFAの面積が大きくなるためである．しかし実用的なmの範囲では，改良型の方が面積は大きい．またβの値が小さい場合でも，ステージ間配線面積が小さくなるのに対し，CSFAの面積は変わらないため改良型の方が面積が小さくなる．これらの結果より，改良型は遅延時間では有利であるが面積では従来型より大きくなるため，面積の増加を減らす方策として，できるだけmを大きくとる工夫とβを小さくするためのステージ配置の工夫が必要である．

4.4 設計例

与えられる乗算器の仕様は，ビット数n，乗算時間の最大値T_{spec}である．この条件を満足する構成は数多くあり得るので，その中から面積の最小のものを最適とする．いま$n=64$ビット，T_{spec}として$T_{AR}=10T_{FA}$の乗算器を考える．図7より，この条件を満たすmの値は非Boothの場合は$m=3\sim4$，Boothの場合は$m=3\sim6$である．しかしBoothの場合はエンコーダの遅延があるので，実質的に$m=3$となる．一方，図8(b)より面積はmの値の大きい方が小さくなるので，非Boothの場合は$m=4$で$S=97\sim144$程度になり，Boothの場合には$m=3$で70～106程度で両者の比は前者が後者の1.39～1.36倍となる．時間を遅らせて$T_{AR}=15T_{FA}$とすると非Boothでは$m=8$，Boothでは$m=9$（Boothのエンコーダを含める）とすることができ，面積は非Boothで85～120で，面積比は0.88～0.83倍，Boothでは57～73で0.81～0.69倍となる．また前者と後者の比は1.49～1.64倍となり，高速の場合に比べて後者がより有利となる．mが10以上になると面積の減少傾向が小さくなり，ほとんどきかなくなる．したがってmを10以上にとっても時間が遅くなるだけで面積的には効果がなく，意味のないことが分かる．mの有効な範囲は3～10の間である．

5. むすび

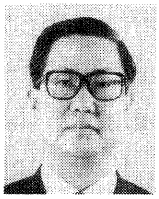
乗算時間とレイアウト面積のトレードオフをとることができる乗算器アーキテクチャとその設計手法について考案した．アーキテクチャとしてはFAを基本セルとする多重波面型乗算回路を高速化した改良多重波面型構成を提案し，さらにこの構成をBoothのアルゴリズムに適用した．この構成をもとにしてツリー型とアレイ型を両極端として含む中間の性能のBoothと

非 Booth の乗算器の構成について，共通する精算用 CLA を除く乗算器の部分積加算部の演算時間と面積を比較する方法を明らかにした．その結果基本演算回路として FA を直列に接続した回路を使用し，その外部入力数 m を設計パラメータにして演算時間と面積を表示すると m の増加に対し一方は増加し，他方は減少するトレードオフの関係が生じることを示した．したがって最適値を求めることができ，高位合成の乗算器モジュールジェネレーションに適用することができる．今後の問題としては，基本演算回路を FA でなく m に対応した最適回路化することがある．$m=4$ は最近使用されるようになってきた 4-2 加算器[10]であるが，$m=3$ の FA に対して 4-2 加算器の位置づけを明確にし，さらに他の回路についても最適構成をとることにより，最適な乗算器の生成を可能にする手法を開発することが必要である．

参考文献

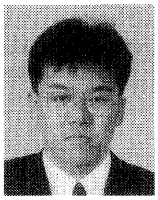
1) McFarland, M.C., Parker, A.C. and Camposano, R.: The High-level Synthesis of Digital Systems, *Proc. of the IEEE*, Vol.78, No.2, pp.301–318 (1990).
2) Moshnyaga, V., Mori, Y., Onodera, H. and Tamaru, K.: Layout-Driven Module Selection for Register-Transfer Synthesis of Submicron ASIC's, *Proc. IEEE/ACM Int. Conf. on CAD-93, Santa Clara*, pp.100–103 (1993).
3) Wallace, C.S.: A Suggestion for a Fast Multiplier, *IEEE Trans. Electron Comp.*, Vol.EC-13, No.2, pp.14–17 (1964).
4) Chu, K. and Sharma, R.: A Technology Independent MOS Multiplier Generator, *Proc. IEEE ACM DAC*, pp.90–97, Jun. (1984).
5) Singh, H.P., Burrier, R.A. and Sadler, R.A.: A 6.5-ns GaAs 20×20 bit Parallel Multiplier with 67ps Gate Delay, *IEEE J. Solid-State Circuits*, Vol.25, No.5, pp.1226–1231 (1990).
6) Hatamian, M. and Cash, G.: A 70-MHz 8 bit × 8 bit Parallel Pipelined Multiplier in 2.5 μm CMOS, *IEEE J. Solid-State Circuits*, Vol.21, No.4, pp.505–513 (1986).
7) Oowaki, Y., Numata, K., Tsuchiya, K., Tsuda, K., Takato, H., Takenouchi, N., Nitayama, A., Kobayashi, T., Chiba, M., Watanabe, S., Ohuchi, K. and Hojo, A.: A Sub-10-ns 16×16 Multiplier Using 0.6 μm CMOS Technology, *IEEE J. Solid-State Circuits*, Vol.22, No.5, pp.762–765 (1987).
8) Sharma, R., Lopez, A.D., Michejda, J.A., Hillenius, S.J., Andrews, J.M. and Studwell, A.J.: A 6.75 ns 16×16 bit Multiplier in Single-Level-Metal CMOS Technology, *IEEE J. Solid-State Circuits*, Vol.24, No.4, pp.922–926 (1989).
9) Stearns, C.C. and Ang, P.H.: Yet Another Multiplier Architecture, *IEEE 1990 CICC*, pp.24.6.1–24.6.4 (1990).
10) Goto, G., Sato, T., Nakajima, M. and Sukemura, T.: A 54 × 54-b Regularly Structured Tree Multiplier, *IEEE J. Solid State Circuits*, Vol.27, No.9, pp.1229–1236 (1992).
11) Islam, F.F. and Tamaru, K.: An Architecture for High Speed Array Multiplier, *IEICE Trans. Fundamentals*, Vol.E76-A, No.8, pp.1326–1333 (1993).
12) Faravi-Ardekani, J.: $M \times N$ Booth Encoded Multiplier Generator Using Optimized Wallace Trees, *IEEE Trans. VLSI Systems*, Vol.1, No.2, pp.120–125 (1993).

(平成 7 年 1 月 9 日受付)
(平成 7 年 11 月 2 日採録)



田丸 啓吉（正会員）

1936 年生．1958 年京都大学工学部電子工学科卒業．1960 年同大大学修士課程（電子工学専攻）修了．同年（株）東芝に入社．総合研究所勤務．1979 年京都大学工学部教授．この間，マイクロコンピュータのアーキテクチャ，LSI の設計手法，LSI 用 CAD などの研究に従事．工博．著書「ハードウェア技術」(オーム社),「理論回路の基礎」(工学図書),「マイクロコンピュータ入門」(日刊工業，共著）など．電気学会，情報処理学会，ACM，IEEE 各会員．



品川 正行

1970 年生．1993 年京都大学工学部卒業．1995 年同大学院修士課程修了．乗算器の小面積化・高速化の研究に従事．同年シャープ（株）入社．

Farhad Fuad Islam

1962年生．1986年バングラデシュ工科大学（B.U.E.T.）電気電子工学科卒業，1988年同修士課程修了．1993年京都大学大学院工学研究科電子工学専攻博士課程修了．博士（工学）．1993年4月よりNTTヒューマンインタフェース研究所にポストドクトラルフェローとして2年間勤務．1995年4月ヒューレットパッカード日本研究所に入社．VLSIハードウェウアルゴリズム，DSPのアーキテクチャ，マルチメディア通信などの分野の研究をしている．IEEE，電子情報通信学会，オーストラリアIEA各会員．