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

2006－ＳＬＤＭ－125（12）
2006／5／12

整数乗算器の消費電力と遅延について

–Wallace Tree, Dadda Treeの比較–

橘 昌良†

† 高知工科大学工学部電子光システム工学科 〒782-8502 高知県香美市土佐山田町宮の口 185
E-mail: †tachibana.masayoshi@kochi-tech.ac.jp

あらまし 8から３２ビットの整数乗算器の部分積加算回路を Wallace Tree, Dadda Tree, およびその中間の形式で作成し，最終加算回路と共にシミュレーションにより遅延，面積の比較を行った．回路の作成は，生成アルゴリズムに基づく 1024 ビットまでの乗算器の生成が可能なプログラムにより行った．また，Wallace Tree 型の乗算器については，LSI での実装を 0.35m ルールで行い，消費電力の実測を試みた．さらに，部分定数化した乗算器との比較を試みた．
キーワード Wallace Tree, Dadda Tree, 部分乗数乗算器，整数乗算器

Delay and Power Consumption of Integer Multipiler

– Comparison of Wallace and Dadda tree –

Masayoshi TACHIBANA†

† Department of Electronics and Photonic System Engineering, Kochi University of Tecnology Miyanokuchi 185, Tpsayamada, Kochi, 782-8502 Japan
E-mail: †tachibana.masayoshi@kochi-tech.ac.jp

Abstract In this report, We compare Delay and Area of Integer Multiplier with both Wallace Tree type and Dadda tree type partial product adder in 8 bit to 32 bit range. Final adder of these multipliers are composed of RCA, CLA, combination of RCA and CLA fashion. Also, We implemented 8 bit Multiplier with Wallace Tree partial product adder in 0.35μm rule CMOS LSI and measured actual power consumption. And we compare that with 8 bit Partial-Constant Multiplier's actual power consumption which is implemented in the same LSI chip.
Key words Wallace Tree, Dadda Tree, Integer Multiplier, Partial-Constant Multiplier

1. まえがき

部分積加算回路と最終加算回路を複数の形式で構成した，8 から 32 ビットの整数乗算器の遅延と面積の比較結果，8 ビットの Wallace Tree 型整数乗算器について，通常の形式と部分定数化を行い低消費電力化したものを LSI 化し，消費電力の比較を行った結果について報告する．

整数演算器の部分積加算回路の構成としては，Wallace Tree, Dadda Tree 型の構成が用いられることが多い．しかしながら，この二つの形式は部分積の加算に関して木構造をどのように構成するかという観点で比較されることが多く，遅延時間や面積について定量的な比較を行っていることは少ない．[2], [3], [4] 本報告では，２つの形式とその中間的な形式について回路を生成することの出来る生成プログラムを作成し，8 ビットから 32 ビットの構成についてシミュレーションにより遅延時間と面積の比較を行った．この際，最終加算回路の形式も RCA，CLA，RCA と CLA の組み合わせの 3 通りの回路を生成し，比較を行った．

加算器や乗算器の消費電力は，なにを代表値として考えるべきかを含めて議論のあるものである．本報告では，シミュレーションではなく実際に作成した LSI を用いて消費電力の測定を試みた．対象としたのは Wallace Tree 型の部分積加算回路をもつ乗算器で，同時に，信号処理に特化した乗算器として部分定数化 [1] した乗算器についても実装を行い，実測により消費電力の比較を行った．

2. 整数乗算器の比較

設計する乗算器は，高速乗算器として代表的な Wallace Tree 型乗算器と Dadda Tree 型乗算器，その 2 つを合わせたハイブリッド型乗算器とし，積を算出するための最終加算回路は，桁

表 1　Dadda Tree における部分積の個数制限

Table 1　xxx

段数	部分積	段数	部分積	段数	部分積
0	2	7	28	14	474
1	3	8	42	15	711
2	4	9	63	16	1066
3	6	10	94	17	1599
4	9	11	141	18	2398
5	13	12	211	19	3597
6	19	13	316	20	5395

上げ伝播加算器（Ripple Carry Adder）と先見桁上げ加算器（Carry Look-ahead Adder），その 2 つをつなげた加算器の 3 つとする．そして，生成された回路の速度と面積を比較することで，各アルゴリズムの乗算器の特性を把握することができる．本研究では並列乗算器として，アルゴリズムを実現しやすい Wallace Tree 型乗算器と Dadda Tree 型乗算器，そして，その 2 つを合わせたハイブリッド型乗算器を対象とした．

2.1　Wallace Tree, Dadda Tree の構成

Wallace Tree を用いた部分積加算では，全加算器（以下 FA）と半加算器（以下 HA）を桁上げ保存加算器として考える．各列で 3 個以上の部分積が FA で加算され，最後に残った 2 個の部分積は HA を用いて加算している．和と桁上げの出力が部分積の個数に関係なく，FA，HA1 段分の計算が可能となる．

部分積が n 個の場合，$n/3$ 個の FA を用いて上記の処理を並列に行えば，部分積の個数は $2n/3$ 個となる．

この処理を $\log_{2/3} n$ 回繰り返せば，部分積は 2 個になるので，回路の段数は $\log n$ に比例し，高速な演算が可能となる．8 ビット入力の乗算器では，部分積が 64 個となるので，FA と HA の 5 段分の計算時間ですべての列の部分積加算を行えることになる．

Dadda Tree のアルゴリズムは Wallace Tree のアルゴリズムと似ている．違うのは Wallace Tree が部分積が 3 以下になるまで，すべて FA を使用するのに対し，Dadda Tree は，各列の部分積の個数が表 1 の数になるように，FA だけでなく，HA を使用することである．　Wallace Tree と Dadda Tree を合わせる．部分積が 6 より大きい間は，Wallace Tree のアルゴリズムを用いて FA のみで部分積を減らしていき，すべての列が 6 以下になってから，Dadda Tree のアルゴリズムを用いて部分積を加算していく．　このアルゴリズムでは 9 ビット以下の入力の乗算器では Dadda Tree と同じ動作をすることになり，9 ビットより大きくなければならない．

2.2　最終加算回路

最終加算回路は，Ripple Carry Adder，Carry Look-ahead Adder，途中まで Ripple Carry Adder その後を Carry Look-ahead Adder という回路とした．

木状の部分積加算回路では，部分積を 2 以下にするために各列で使用する加算器の数が異なるので，各列ごとに遅延が異なる．Wallace Tree，Dadda Tree 共に，1 列目から真ん中の列までは通過する加算器が増えていくことになるので，その分の遅延が生じる．従って，最終加算回路で RCA を用いても，CLA と比較して最終的な遅延は変わらないと考えられる．真ん中の列から最後の列にかけては，通過する加算器が減っていくと考えられるので，CLA を用いて最終加算器の遅延を最小に押さえる．

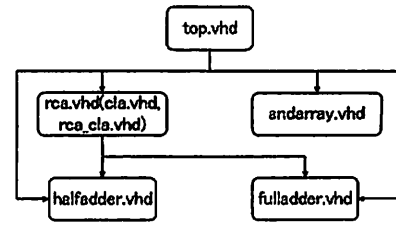

図 1　各 VHDL の関係

Fig. 1　xxx

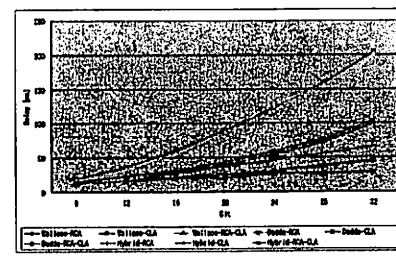

図 2　乗算器の遅延

Fig. 2　xxx

2.3　乗算回路生成ツール

開発したツールのプログラムは，以下のような動作をする．（1）必要なビット数を指定し，部分定数化に対応するために乗数のビットパターンを入力．（2）回路や信号をデータ構造を用いてモデル化する．（3）乗算器のアルゴリズムに沿ってデータを処理する．（4）結果を VHDL で出力する．

VHDL は，各回路ごとに分割して出力する．半加算器と全加算器の VHDL（`halfadder.vhd`, `fulladder.vhd`），入力されたビット幅とビットパターンからなる AND ゲートを並べた部分積生成回路 `andarray.vhd`），最終加算器は 3 種類の加算回路（RCA, CLA, RCA-CLA）の VHDL（`rca.vhd`, `cla.vhd`, `rca_cla.vhd`），Wallace Tree，Dadda Tree の部分積加算回路を含み，回路全体をまとめるトップレベルの VHDL（`top.vhd`）を分割して出力する．図 1 に各 VHDL の関係を示す．

2.4　シミュレーション

生成された VHDL を Synopsys 社の Design Compiler で論理合成を行った．論理合成には，京大版 Rohm0.35μm ライブラリを使用した．最終加算回路は平坦化せず，別のモジュールとして論理合成を行った．論理合成後，Timing の予測値を出力した．各乗算器の遅延は，図 2 のようになる．このグラフから，Wallace Tree 型乗算器は Dadda Tree 型乗算器とハイブリッド型乗算器に比べて，遅延が大きいことがわかる．また，最終加算回路を RCA，CLA，RCA-CLA と変更することでは明確な差が認められない．最終加算回路が速度に影響を与えないとい

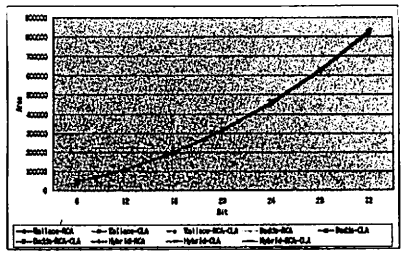

図 3　乗算器の面積
Fig. 3　xxx

うことから，部分積加算回路内の桁上げ伝播が下位列（LSB 付近）から上位列（MSB 付近）まで続くことで，上位列での部分積加算が遅くなっていることが考えられる．

Dadda Tree 型乗算器は 3 つの中で最も速いことがわかる．最終加算回路の違いで見ていくと，速い順に CLA，RCA-CLA，RCA となっている．特に，CLA については，部分積加算回路の段数に従って速度も階段状になっていることが良くわかる．また，24 ビットまでは CLA と RCA-CLA で大きな差はない．

ハイブリッド型乗算器は 8 ビットでは Dadda Tree と回路がまったく同じであるので，速度は同じである．16 ビットから Dadda-CLA，Dadda-RCA-CLA と差がつき始め，24 ビットからは Dadda-RCA より遅くなっている．そして，Wallace Tree 型乗算器と同じく，最終加算回路が変わっても速度に大きな変化が見られないことがわかる．

次に，面積についての比較を図 3 に示す．各乗算器で大きな差はないが，Wallace Tree 型乗算器が最も小さく，次いでハイブリッド型乗算器，Dadda Tree 型乗算器となっている．最終加算回路は RCA，RCA-CLA，CLA の順で大きくなっている．

生成された VHDL を見ると，すべてのビットで各乗算器の部分積加算回路を構成する FA，HA の数は等しくなっているので，各回路を別々にコンパイルし，平坦化を行わなければ，最終加算回路の面積以外に差はなくなる．

以上のことから，16 ビットまではハイブリッド型で最終加算回路を RCA にすれば，速度と面積のバランスが良いことがわかる．24 ビットでは Dadda Tree 型で最終加算回路を RCA-CLA に，20 ビット，24 ビット以上では CLA が最も速く，かつ面積の大きさを抑えることができることがわかる．

2.5 ま と め

ツールにより生成される VHDL のコードは，32 ビット時における `top.vhd` では 8539 行，信号線の本数は 930 本，全加算器と半加算器の使用数 899 個と 31 個．最終加算回路を桁上げ先見加算器にすると，`cla.vhd` の行数は 618 行になる．

今後の課題としては，今回研究結果から Dadda Tree 型乗算器が最も速く，面積も Wallace Tree 型乗算器と殆ど変わらないことが分かったので，Dadda Tree 型乗算器の最終加算回路を CLA 以外の高速な加算器（桁上げ選択加算器，並列プレフィックス加算器等）で設計できるように，自動生成ツールを作成することがあげられる．

また，部分積の並べ替えを実際の FA と HA のディレイに置き換えることで，並べ替えの更なる最適化が実現できると考えられる．

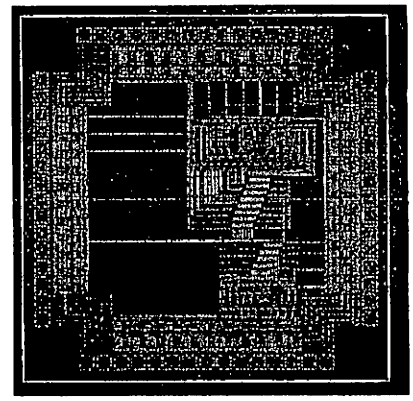

図 4　試作チップのレイアウト
Fig. 4　xxx

3. 部分定数化乗算器の消費電力

部分定数乗算器は，ある特定の演算を行う乗算器に着目し，冗長である部分を取り除くというものである．回路の一部分を取り除いてしまうため通常の乗算器としては使えなくなるが，特定の演算を行う場合は回路規模の縮小（小面積化）や高速化を実現することができる．

2 つの乗数，01011001 と 10010101 について考えると，1，2，5，6bit 目が等しいことがわかる．ここで，1 を固定した場合，部分積には被乗数をそのまま代入するだけで削減できる部分としては部分積生成回路のＡＮＤゲートのみであり，低消費電力化は見込めない．そこで，乗数の 0 が共通する桁のみを固定する部分定数化を行った．こうすると，乗数が 0 であるため生成される部分積はすべて 0 である．これならば，部分積加算の演算数自体を削減することができるため低消費電力化を期待することができる．

Tree 構造では，固定する位置により削減できる段数が列ごとに違うので，固定する bit 位置により削減量は変わってくる．必要な加算器数が少なくなるのは中央付近の bit を固定したときである．Tree 構造では，部分積の段数が多くなるのは中央付近であるため，LSB や MSB 側を固定するより少し削減できる加算器の数が増える．

3.1 チップ試作と消費電力の測定

8 ビットの Wallace Tree 型乗算器において 2，5，8 ビット目を 0 に固定した部分定数化乗算器と固定を行わない通常の乗算器について実際に作成し，その消費電力を比較することを試みた．

レイアウト設計に使用したセルは，チャネル長 0.4μm，P チャネル幅 20μm，N チャネル幅 10μm ですべて設計した．また，入出力ピンには全て VDEC より提供される I/O バッファを使用している．図 4 は，Cadence 社の Layout Plus 4.46 を用いてレイアウト設計したものである．

通常乗算器，部分定数乗算器の各レイアウト図面から，Ca-

表 2　消費電力シミュレーション結果

Table 2　xxx

乗算器	Avr. Power [μW]	RMS Power[μW]
通常乗算器	1371.89	7630.81
部分定数乗算器	869.31	5053.67

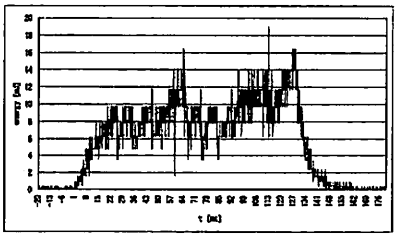

図 5　通常乗算器の消費エネルギー (01101101)

Fig. 5　xxx

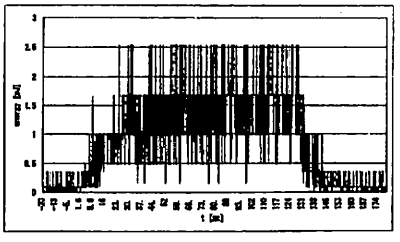

図 6　部分定数乗算器の消費エネルギー (01101101)

Fig. 6　xxx

dence 社 Dracula の LPE 機能を用いて配線容量などのデータを抽出，その後 Synopsys 社の HSPICE を通して論理検証，Nanosim にて消費電力シミュレーションを行った．表 2 にその結果を示す．LSI 設計時に内部に 2 系統の電源を作っており，それぞれ通常乗算器部分，部分定数乗算器部分のみに電源を供給している．ここでは，各電源と LSI に直列に抵抗を挿入し，抵抗の両端の電圧より電流を求め電力を算出する方法をとった．しかし，乗算器の演算自体は数 ns で終了してしまうため 1 演算の瞬間的な動作電流を測定することは不可能である．そこで，入力信号をパターンジェネレータで LSI の入力に常に供給しておき，一定時間におけるエネルギーを算出することにした．

電流は前節のシミュレーションデータより数十 mA から数 mA 程度であると予想され，抵抗値を大きくしてしまうと LSI への供給電圧の変動が大きくなってしまうので，測定値 10.07Ω の抵抗を用いて測定を行った．

計測には，Hewlett Packard 社の 16702A を使用した．パターンジェネレータより入力信号を生成し，同じく，LogicAnalyzer にて出力が正常であるかを確認する．同時に，オシロスコープで抵抗の両端の電圧を測定し，測定データを読み出し消費されるエネルギーを算出した．測定時に乗算器の入力として非乗数側は常に全パターンを与え，乗数側は "01101101"，"00001101"，"00000000" と 3 パターンで固定した．図 5 に乗数に "01101101" を設定した場合の通常乗算器の測定結果のグラフを示す．X 軸（時間）にマイナスがあるのは，入力が乗数，非乗数ともに "00000000" 状態から信号が遷移したタイミングでトリガをかけているため，トリガ時が基準であるためである．

測定結果から非乗数の値が大きくなるにつれて，クリティカルな桁上げ信号の遅延が増えるため，消費されるエネルギーは大きくなっている．

次に，図 6 に部分定数乗算器についての測定結果を示す．

部分定数乗算器では，余計な部分積を削減することでそれに伴う冗長であった演算も削除できているため，通常乗算器のように右肩上がりにはなっていない．また，消費エネルギーは信号が遷移したときに突出しており，通常乗算器に比べて消費されるエネルギーは少なくなっている．

消費エネルギーの差はきわめて大きい．しかし，レイアウトのシミュレーション結果に比べて差が大きすぎることから，LSI の入出力ピンに挿入した I/O バッファによる消費エネルギーが偏っていると考えられる．

3.2 ま と め

乗算器を対象とした論理ゲート数の削減による低消費電力化の手法として，部分定数化による低消費電力化について検討した．シミュレーション結果よりわかるように，固定した桁数分の低消費電力化は見込めるので低消費電力化に有効であるといえる．また，実際にに LSI の試作を行い，消費エネルギーの測定を行った．バッファによる消費エネルギーも含まれているが，これらの結果からも部分定数化による乗算器の低消費電力化は十分有効であると考えられる．

試作した LSI では，入出力に全て I/O バッファをとりつけている．今回の測定方法では，この I/O バッファによる消費エネルギーも含まれてしまうために，乗算器単体での消費エネルギーは測定できていない．そのため I/O バッファによる消費エネルギーをキャンセルする必要がある．これについては，別の LSI に I/O バッファ単体を実装しており，この I/O バッファ単体の消費エネルギーを測定し，今回の測定結果より差し引くことで乗算器単体の消費エネルギーを求めることができるものと思われる．

4. おわりに

本研究は，東京大学大規模集積システム設計教育センターを通し，シノプシス株式会社，日本ケイデンス株式会社，及びローム株式会社の協力で行われたものである．

文　献

[1] 杉本 秀一, 遠藤 彬浩, 橘 昌良, 部分定数化による乗算器の低消費電力化について, 電子情報通信学会技術研究報告 [VLSI 設計技術], VLD2005-4, pp.19-24, 2005.

[2] Behrooz Parhami, Computer Arithmetic algorithms and hardware designs, Oxford University Press, 2000.

[3] Lsrael Koren, Computer Arithmetic Algorithms 2ND edition, A K Peters.Ltd, 2002.

[4] 高木直史, 算術演算の VLSI アルゴリズム, コロナ社, 2005.