システムＬＳＩ設計技術　108－11
(2003. 1. 29)

多線2相式データ表現を用いた非同期式乗算器

西野　領†　阿部　公輝†

† 電気通信大学情報工学科
〒182–8585 調布市調布ヶ丘 1–5–1
E-mail: †{ike-ryo,abe}@cacao.cs.uec.ac.jp

あらまし　冗長2進加算木を用いた多線2相式乗算器を設計し，VLSIに実装するときの回路面積と速度を評価した．この乗算器は，非同期式プロセッサなどで使用することを目的とする．冗長2進値4線2相式データ表現 (RB4W-A)，冗長2進値3線2相式データ表現 (RB3W)，および，RB4W-Aの一部に2進値2線2相式データ表現を用いる場合 (RB4W-B) の3つの符号化方式を適用する場合それぞれについて，評価を行い比較した．乗算器全体では16ビット長において，RB4W-Aを適用した場合に対して，RB3Wを適用した場合12%，RB4W-Bを適用した場合17%の速度向上をもたらすことが分かった．乗算回路の性能は，主に冗長2進加算器の性能で決定され，冗長2進加算器の性能は主に中間値生成モジュールMedGenの性能で決定される．モジュールMedGenはRB4W-Bを適用する場合が，もっとも性能が優れていることが分かった．

キーワード　非同期回路，乗算器，冗長二進表現，多線2相式表現

Asynchronous Multipliers Based on Multi-Wire 2-Phase Data Representation

Ryo NISHINO† and Kôki ABE†

† Department of Computer Science, The University of Electro-Communications
1–5–1 Chofugaoka, Chofu-shi, Tokyo, 182–8585 Japan
E-mail: †{ike-ryo,abe}@cacao.cs.uec.ac.jp

Abstract　Asynchronous multipliers based on redundant binary addition tree, which are intended for being used in asynchronous processors, were designed and evaluated in terms of area and speed when implemented on VLSI. Three coding schemes, redundant-binary 4-wire 2-phase representation (RB4W-A), redundant-binary 3-wire 2-phase representation (RB3W), and modified version of RB4W-A (RB4W-B) where binary 2-wire 2-phase represetation is used in part, were applied to the multipliers and compared. The 16-bit redundant-binary asynchronous multipliers represented in RB3W and RB4W-B were found to be faster than in RB4W-A by 12% and 17%, respectively. The performance of the multipliers is mainly determined by that of redundant-binary adders, which is in turn determined by the module MedGen for generating intermediate values. Among the three representations, the representation RB4W-B was found to yield the module MedGen of the best performance.

Key words　Asynchronous circuits, Multiplier, Redundant binary representation, Multi-wire 2-phase representation

1. はじめに

非同期式回路は，環境変動に対する耐性や消費電力に優れ，非同期式の汎用プロセッサの開発事例もいくつか存在する[1],[2]．これらのプロセッサの算術演算回路の乗算器には逐次型のものや，Wallace木[3]が使用されている．

乗算器の構成法において，冗長2進加算木を用いた乗算器は，規則性と高速性の両面で優れた性質を持つことが知られている[4]．本研究では，非同期式プロセッサなどで使用することを目的とし，冗長2進加算木を用いた多線2相式乗算器を設計し，VLSIに実装するときの回路面積と速度を評価する．

クロックを使用しない非同期式回路では，値が有効であるか無効であるかを区別するデータ表現を用いて，2相のハンドシェイクを行う[5]．非同期回路で冗長2進数体系を用いる際の

表 1　冗長 2 進加算の第 1 ステップの生成規則

被加数 x_i	加数 y_i	下位桁の符号 (x_{i-1}, y_{i-1})	中間桁上げ c_{i+1}	中間和 s_i
1	1	任意	1	0
1	0	共に非負	1	−1
0	1	それ以外	0	1
0	0	任意	0	0
1	−1			
−1	1			
0	−1	共に非負	0	−1
−1	0	それ以外	−1	1
−1	−1	任意	−1	0

符号化方式としては，除算器で 3 線を用いた例が報告されている [6]．本研究では，3 線 2 相符号化に加え，符号と絶対値それぞれを 2 値の 2 線 2 相式表現を用いる 4 線 2 相符号化，および，2 値変数を導入し，4 線 2 相符号化に一部 2 進値 2 線 2 相式表現を用いる符号化の 3 種類の方式について，乗算器に適用した場合の性能を比較評価する．

以下では冗長 2 進加算木を用いた乗算器，非同期回路に適用する場合の符号化方式，符号化方式による乗算器の性能の比較結果，および評価と考察を述べる．

2. 冗長 2 進加算器

冗長 2 進加算木を用いた乗算法は，冗長 2 進数体系に基づく．冗長 2 進数 $X = x_{n-1} \cdots x_0$, $x_i \in \{-1, 0, 1\}$ は，$\sum_{i=0}^{n-1} x_i \cdot 2^i$ を表す．

2 つの冗長 2 進数 $X = x_{n-1} \cdots x_0$ と $Y = y_{n-1} \cdots y_0$ の加算は 2 つのステップで行う [4]．第 1 ステップでは，各桁で，$x_i + y_i = 2c_{i+1} + s_i$ を満たすように，中間桁上げ c_{i+1} と s_i を求める．第 2 ステップでは，各桁で，中間和 s_i と 1 つ下位の桁からの中間桁上げ c_i を加え合わせ，和 z_i を得る．

第 1 ステップでは，冗長性を利用して，第 2 ステップで新たな桁上げが生じないように，表 1 のように c_{i+1} および s_i を定める．

さらに，次のような 2 値の変数 $P_i, u_i, v_i \in \{0, 1\}$ を導入することにより，回路性能を向上させる方法も報告されている [7]．

$$\begin{aligned} u_i &= P_{i-1} - s_i \\ v_{i-1} &= P_{i-1} + c_i \end{aligned} \tag{1}$$

ここに，変数 P_i は，x_i, y_i ともに非負のとき 0，そうでないとき 1 をとる．和は次式から得られる．

$$z_i = s_i + c_i = v_{i-1} - u_i \tag{2}$$

これらの変数の真理値表を表 2 に示す．

図 1 に冗長 2 進加算の 3 桁分を示す．1 桁分の冗長 2 進加算器を RBA と呼ぶ．PGen は符号判定に用いられる変数 P_i を生成する．MedGen は，中間桁上げ c_{i+1} と中間和 s_i を生成する．表 2 の 2 値変数 v_i, u_i を導入する場合には，c_{i+1} と s_i の代りに v_i, u_i を生成する．SumGen は，和 z_i を生成する．

表 2　冗長 2 進加算器:2 値変数の導入

x_i	y_i	P_{i-1}	P_i	c_{i+1}	s_i	v_i	u_i
1	1	0	0	1	0	1	0
		1	0	1	0	1	1
1	0	0	0	1	-1	1	1
0	1	1	0	0	1	0	0
0	0	0	0	0	0	0	0
		1	0	0	0	0	1
1	-1	0	1	0	0	1	0
-1	1	1	1	0	0	1	1
0	-1	0	1	0	-1	1	1
-1	0	1	1	-1	1	0	0
-1	-1	0	1	-1	0	0	0
		1	1	-1	0	0	1

3. 冗長 2 進数の多線 2 相式符号化

クロックを使用しない非同期式回路では，組合せ回路での計算が完了したことがデータを見て分かるように，値が有効であるか無効であるかの区別をする．たとえば，2 進数体系では，表 3 のように，1 桁を 2 本の信号線で表す 2 線 2 相式表現が用いられる [5]．ここでは，この表現を B2W と呼ぶ．

表 3　2 進値 2 線 2 相式

状態	B2W
無効	00
0	01
1	10

冗長 2 進数体系においては，表 4 のように，符号と絶対値それぞれを，2 値の 2 線 2 相式表現で表すことが考えられる．この符号化をここでは冗長 2 進値 4 線 2 相式データ表現と呼び，RB4W と表す．

表 4　冗長 2 進値 4 線 2 相式データ表現

状態	RB4W	
	符号	絶対値
無効	00	00
0	01	01
1	01	10
-1	10	10

有効なデータは 3 値であるから，表 5 のように，3 線で表現することもできる．これを冗長 2 進値 3 線 2 相式データ表現と呼び，RB3W と表す．

表 5　冗長 2 進値 3 線 2 相式データ表現

状態	RB3W
無効	000
0	001
1	010
-1	100

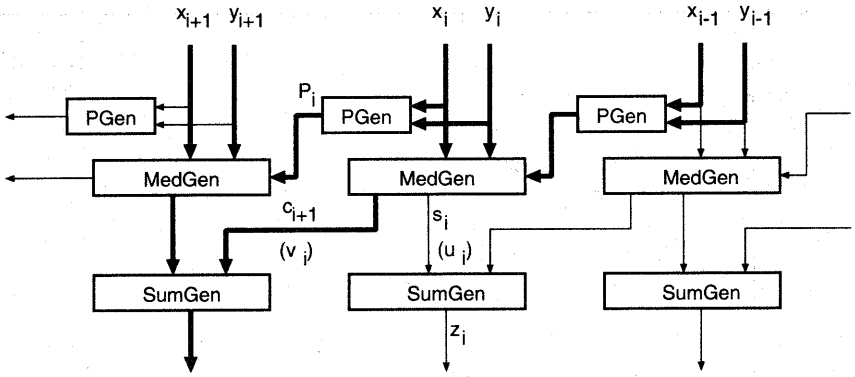

図 1　冗長 2 進加算器 (括弧内は、表 2 の 2 値変数を導入する場合)

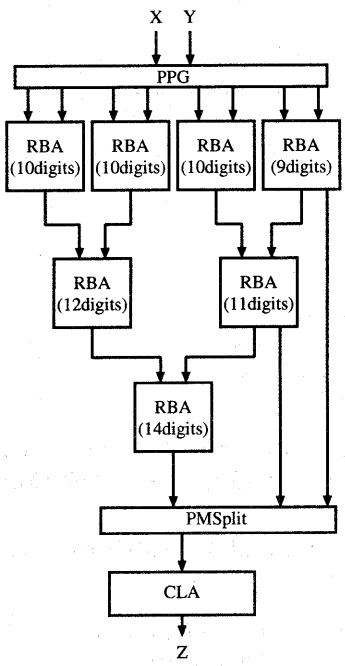

図 2　8 ビット冗長 2 進乗算器の構成図

4. 冗長 2 進乗算器

冗長 2 進加算木を用いた乗算 [4] では，部分積を 2 つずつ 2 分木状に冗長加算器で加え合わせ，冗長 2 進表現で表された積を得，それを 2 進表現に変換する．

例として 8 ビットの場合の構成を図 2 に示す．図において，PPG は部分積生成部で，8 ビット長の部分積 8 個を生成する．RBA は冗長 2 進加算器である．8 ビットの場合，RBA は $\log_2 8 = 3$ 段である．初段の RBA は，入力が 2 進値である点が後段の RBA と異なる．RBA の桁数は後段になるにしたがい増加する．PMSplit は正の数と負の数への分離回路，CLA は桁上げ先見加算器を表す．PMSplit と CLA により，冗長 2 進値から 2 進値への変換を行う．符号化 RB4W と RB3W を冗長 2 進乗算器に適用する．

RB4W の適用に当たっては，表 1 の中間値生成規則を用いる場合と，新たに 2 値変数を導入する表 2 を用いる場合とが考えられる．後者では，2 値の変数 u_i, v_i をそれぞれ B2W で表現できる．冗長 2 進乗算器の前者による回路構成を multRB4W-A，後者による回路構成を multRB4W-B と呼ぶ．

符号化 RB3W を適用する場合は，表 1 の中間値生成規則ををそのまま用いる．冗長 2 進乗算器の RB3W による回路構成を multRB3W と呼ぶ．なお，いずれの符号化でも，B2W で表現された 2 値の変数 P_i を用いる．

第 2 節で述べた冗長 2 進加算器 RBA に対し，上の 3 つの符号化法をそれぞれ適用し，図 2 の構成の乗算器 multRB4W-A，multRB4W-B，および，multRB3W を設計した．

5. 結　　果

冗長 2 進乗算器の符号化方式による違いをみるため，回路構成 multRB4W-A，multRB4W-B，multRB3W について，Verilog-HDL で記述し，論理合成ツール DesignCompiler で合成して得られる速度と面積の値を用いて，比較した．合成に用いたライブラリは，ローム社の製造条件に基づき VDEC(東京大学大規模集積システム設計教育研究センター) で製作されたものを用いた．主な製造条件は，CMOS 0.35μm，PolySi 2 層，メタル配線 3 層，電源電圧 5V である．

まず，乗算器の主たる構成要素である冗長 2 進加算器 RBA について，速度と面積を調べた．RBA のクリティカルパスは図 1 の太線の経路となるので，3 桁分について，遅延時間の制約条件を変化させて論理合成を行なった．その結果を図 3 に示す．図 3 から，良い方から RB4W-B, RB3W, RB4W-A の順に，より高速かつ低面積の冗長 2 進加算器 RBA が得られることが分かった．

さらに，RBA の内訳を調べるため，構成モジュール PGen, MedGen, SumGen について，速度と面積を調べた．その結果を表 6 にまとめて示す．MedGen のクリティカルパスは，中間値 c_{i+1} または v_i を出力するパスとなる．

RBA の構成モジュールのうち，符号化方式の違いが大きく現れるのは，MedGen であることが分かる．

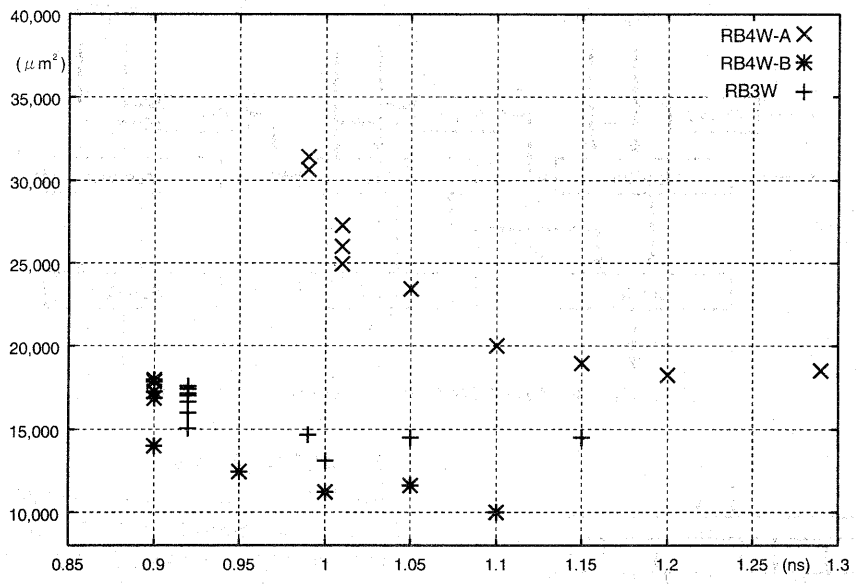

図 3　冗長 2 進加算器 RBA の 3 桁分の合成結果

表 6　RBA の内訳

	multRB4W-A		multRB4W-B		multRB3W	
	time (ns)	area (μm^2)	time (ns)	area (μm^2)	time (ns)	area (μm^2)
PGen	0.21	271	0.21	271	0.26	421
MedGen	0.65	3664	0.43	1212	0.55	2225
SumGen	0.32	1485	0.26	742	0.28	889

そこで，符号化方式が MedGen に与える影響を詳しく調べるため，遅延時間の制約条件を変えて合成を行った．その結果を図 4 に示す．3 つの符号化による違いが顕著に現れているのが分かる．

図 5 は，8 ビット冗長 2 進乗算器について，遅延時間の制約条件を変化させて合成させた結果である．

同様に，16 ビット冗長 2 進乗算器について，遅延時間の制約条件を変化させて合成させた結果を図 6 に示す．

6. 評価と考察

図 1 と表 6 から，RBA のクリティカルパスの遅延時間は，PGen, MedGen, SumGen の遅延時間の和となり，符号化方式の違いは中間値生成回路 MedGen にもっとも大きく現れることが分かった．

論理関数の複雑さは，入出力変数のビット数と，関数の縮約の容易さによる．MedGen の出力変数は，RB4W-B では 4 ビット，RB3W では 6 ビット，RB4W-A では 8 ビットである．また，入力変数は，RB4W-A および RB4W-B では 10 ビット，RB3W では 8 ビットである．出力変数のビット数は面積に直接関係し，入力変数のビット数は積項のリテラル数，すなわちゲートのファンインに影響を与える．これらのことと，関数の縮約の容易さの違いから，符号化方式による MedGen の面積と速度の違いが現れたと考えられる．

次に，冗長 2 進乗算器全体の性能に対し，冗長 2 進加算器 RBA がどのように影響を与えるかを遅延時間について考察する．

n ビットの冗長 2 進乗算器の遅延時間 $T_{\mathrm{delay}}(n)$ は次式で与えられる．

$$\begin{aligned} T_{\mathrm{delay}}(n) \quad = \quad & T_{\mathrm{PPG}} + T_{\mathrm{RBA1stLevel}} \\ & + (\log_2 n - 1) \cdot T_{\mathrm{RBA}} \\ & + T_{\mathrm{PMSplit}} + T_{\mathrm{CLA}}(n) \end{aligned} \tag{3}$$

ここに，$T_{\mathrm{PPG}}, T_{\mathrm{RBA1stLevel}}, T_{\mathrm{RBA}}, T_{\mathrm{PMSplit}}$ は，それぞれ，部分積生成部，初段の冗長 2 進加算器，冗長 2 進加算器，正負分離部の遅延時間で，桁数によらず一定である．$T_{\mathrm{RBA1stLevel}}$ は，通常の B2W 表現を入力に持つ冗長 2 進加算器の遅延時間である．また，$T_{\mathrm{CLA}}(n)$ は n 桁の桁上げ先見回路の遅延時間で $\log n$ に比例する．

冗長 2 進乗算器の各構成モジュールについて論理合成を行い，これらの遅延時間を求めた結果を表 7 に示す．各モジュールは，回路中で使用されるときの駆動対象の容量を指定して合成を行った．これらの各モジュールの遅延時間から式 (3) により求めた $T_{\mathrm{delay}}(n)$ の値も表に示してある．その値は図 5, 図 6 の乗算器全体の遅延時間とほぼ一致することが分かる．

図 5, 6 から，冗長 2 進乗算器の性能は，良い方から multRB4W-B, multRB3W, multRB4W-A の順に向上するこ

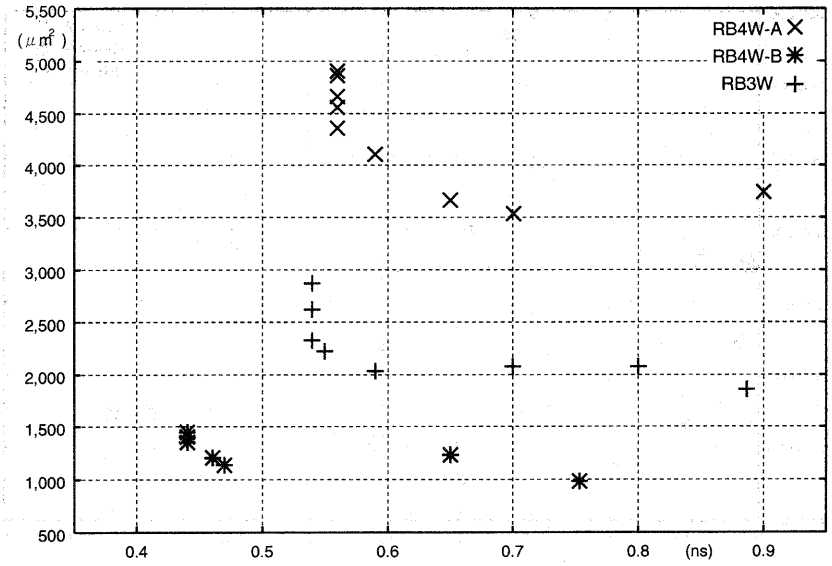

図 4　RBA の構成モジュール MedGen の合成結果

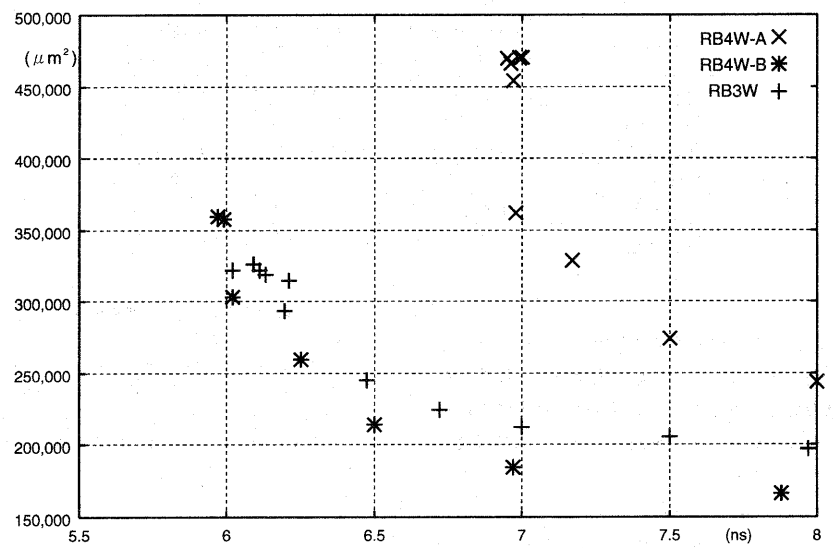

図 5　8 ビット冗長 2 進乗算器の合成結果

とが分かる.

図 6 の 16 ビット長においては，multRB4W-A に対して，multRB3W が 12%，multRB4W-B が 17%の速度向上をもたらしている．表 7 から，これは乗算器の構成要素 RBA の性能によること，および，この傾向は，式 (3) でよく説明できることが分かる.

7. おわりに

非同期式プロセッサなどで使用される冗長 2 進加算木を用いた多線 2 相式乗算器を設計し，VLSI に実装するときの回路面積と速度を評価した．冗長 2 進加算器には，3 つの符号化方式 RB4W-A, RB4W-B, RB3W を適用した．乗算回路の性能は，主に冗長 2 進加算器の性能で決定され，冗長 2 進加算器の性能は主に中間値生成モジュールの性能で決定される．中間値生成モジュールは，冗長 2 進値 4 線 2 相式の符号化に 2 値変数を導入し，一部 2 線 2 相式符号化を用いる方式 RB4W-B を適用する場合が，もっとも性能が優れていることが分かった．乗算器全体では 16 ビット長において，multRB4W-A に対して，multRB3W が 12%，multRB4W-B が 17%の速度向上をもたらす事が分かった.

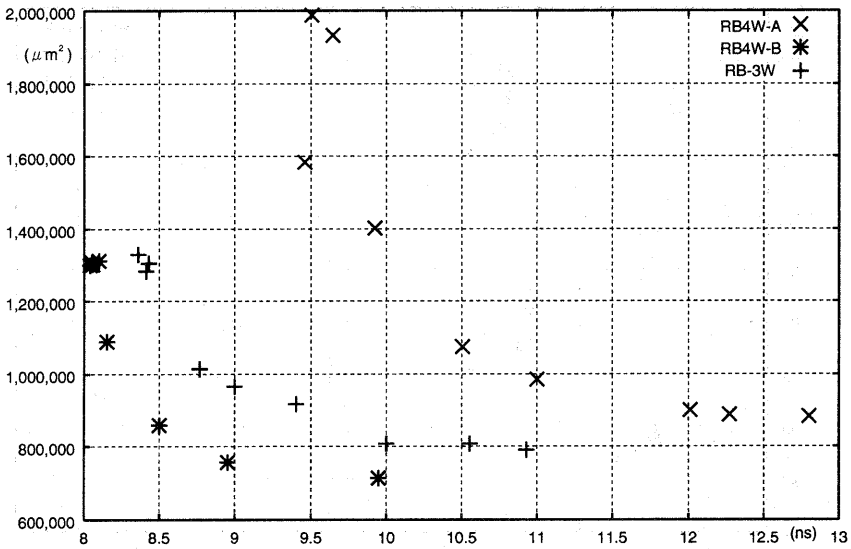

図 6　16 ビット冗長 2 進乗算器の合成結果

表 7　冗長 2 進乗算器を構成するモジュールの遅延時間 (ns)

	multRB4W-A		multRB4W-B		multRB3W	
	$n=8$	$n=16$	$n=8$	$n=16$	$n=8$	$n=16$
T_{PPG}	0.26		0.26		0.26	
$T_{\mathrm{RBA1stLevel}}$	0.79		0.79		0.74	
$T_{\mathrm{RBA}} \times (\log_2 n - 1)$	4.00	6.00	3.10	4.65	3.30	4.95
T_{PMSplit}	0.35		0.35		0.32	
$T_{\mathrm{CLA}}(n)$	1.89	2.20	1.89	2.20	1.89	2.20
$T_{\mathrm{delay}}(n)$	7.29	9.60	6.39	8.25	6.51	8.47

謝　辞

本研究を進めるにあたり，修士論文を参照させていただいた南慶浩氏に感謝いたします．また，有益な議論をし，貴重な助言をいただいた葛毅氏に感謝します．本研究は東京大学大規模集積システム設計教育研究センターを通しシノプシス株式会社の協力で行われたものである．

文　献

[1] J. K. Garside, S. B. Furber, and S.-H. Chung, "AMULET3 Revealed," Proc. 5th International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp.51–59, Apr. 1999.

[2] A. Takamura, M. Kuwako, M. Imai, T. Fjii, M. Ozawa, I. Fukusaku, Y. Ueno, and T. Nanya, "TITAC-2: A 32-bit Asynchronous Microprocessor based on Scalable-Delay-Insensitive Model," Proc. ICCD'97, pp.288–294, Oct. 1997.

[3] C. S. Wallace, "A Suggestion for a Fast Multiplier, " IEEE Trans. Elec. Computer, Vol.EC-13, No.1, pp.14–17, Feb. 1964.

[4] 高木直史、安浦寛人、矢島修三、"冗長 2 進加算木を用いた VLSI 向き高速乗算器," 電子情報通信学会論文誌、Vol.J66-D, No.6, pp.683–690, Jun. 1983.

[5] D. B. Armstrong, A. D. Friedman, and P. R. Menon, "Design of Asynchronnous Circuits Assuming Unbouded Gate Delays, IEEE Trans. Computers, Vol.C-18, No.12, pp.1110–1120, Dec. 1969.

[6] 中野栄治、今井雅、中村宏、南谷崇、"符号確定位置を考慮した非同期式非回復法除算器の設計," 情報処理学会第 58 回全国大会講演論文集, Vol.1, pp.I-51–52, Mar. 1998.

[7] S. Kuninobu, T. Nishiyama, and T. Taniguchi, "High Speed MOS Multiplier and Diveider Using Redundang Binary Representation and Their Implementation in a Microprocessor," IEICE, Vol.E76-C, No.3, pp.436–445, Mar. 1993.