Counter Tree Diagram に基づく冗長加算器の系統的設計手法 ——冗長 2 進加算器設計の例

本間尚文[†,††] 崎山淳[†] 若松泰平[†]
青木孝文[†] 樋口龍雄[†††]

Counter Tree Diagram（CTD）と呼ばれる高速加算アルゴリズムの統一的な表現法に基づく冗長加算器の設計手法を提案する．CTD に基づく設計手法の特徴は，算術演算アルゴリズムに関する特別な知識を用いることなく，桁上げ伝搬の制限された冗長加算アルゴリズムを系統的に導出することにある．設計者は，導出されたアルゴリズムを論理素子で実現することで，高性能な冗長加算器を設計できる．本稿では，例として桁上げ伝搬の制限された冗長 2 進加算器を設計し，提案する設計手法の有効性を示す．

Systematic Design of Redundant Adders Using Counter Tree Diagrams ——An Example of Redundant-Binary Adder Design

Naofumi Homma,[†,††] Jun Sakiyama,[†] Taihei Wakamatsu,[†]
Takafumi Aoki[†] and Tatsuo Higuchi[†††]

This paper presents a design mothod of redundant adders based on a unified representation of fast addition algorithms called *Counter Tree Diagrams* (CTDs). An important feature of the CTD-based design is its capability to obtain possible constant-time addition algorithms in a systematic way without using specific knowledge about underlying arithmetic algorithms. We can derive high-performance redundant adders by mapping them onto physical logic devices. In this paper, we demonstrate the potential capability of CTD-based design through an experimental design of constant-time redundant-binary adders.

1. はじめに

近年，音声・画像・映像などのマルチメディア信号処理の分野に要求される演算能力は増加の一途をたどっており，用途に応じた多種多様な演算回路（データパス）の設計が要求されている．また，従来の 2 進数系にとらわれず，冗長数系や多進数系などの特殊数系を積極的に活用した演算アルゴリズムの有効性が示されており[1),2)]，2 進数 VLSI の性能限界が顕在化するにつれて，その必要性はますます高まると予想される．こうした算術演算回路の設計は，従来，豊富な知識と経験を持つ熟達した設計者により経験的に行われてきた．しかし，近年の応用範囲の拡大や設計手法の多様化にともない，用途ごとに最適な算術演算回路を設計することは，ますます困難な作業となっている．

この問題を本質的に解消するためには，算術演算のアルゴリズム自体を自動合成する手法の開発が急務である．すなわち，あらかじめ知識を用いずに，データ構造の代数的操作により最適な算術演算アルゴリズムを導出できれば，論理回路と同様に算術演算回路についても自動合成が可能となり，データパスの設計分野に多大な影響を与えるものと予想される．現行の論理合成の成功の鍵は，BDD（Binary Decision Diagram）に代表される系統的かつ集約的な論理演算の表現法の発明にあるが，算術演算アルゴリズムの自動合成の実現においても，論理合成における BDD と同様に算術演算アルゴリズムの系統的な表現法が必須であると考えられる．

著者らは，上記の観点から，算術演算アルゴリズムの基本となる加算アルゴリズムの統一的な表現法として，CTD（Counter Tree Diagram）[3),4)]を提案している．一般に，算術演算回路はいくつもの単純な基本演

† 東北大学
Tohoku University
†† 科学技術振興機構さきがけ
PRESTO, JST Japan
††† 東北工業大学
Tohoku Institute of Technology

算回路から構成され，その最も中心となるのが加算器もしくはカウンタと呼ばれる回路である．現在までにさまざまな高速加算アルゴリズムが独自の手法により開発されているが，数系の異なる加算アルゴリズムを統一的に取り扱う設計理論は存在しなかった．提案するCTDは，加算機能を抽象化したカウンタノードから構成され，さまざまな抽象度レベルにおける加算/カウンタのネットワークを表現する．たとえば，CTDにより，冗長2進（RB）加算器[5)~7)]，Signed-Digit（SD）加算器[8)]，桁上げ保存/Positive-Digit（PD）加算器[9)]，並列カウンタ（3-2カウンタ，4-2カウンタ[10)]など）およびその加算器やカウンタによって構成される加算木全般を取り扱うことが可能である☆．

本稿では，CTDの応用として，冗長数系に基づく加算器の設計手法を提案する．一般に，冗長数系を用いることで，桁上げ伝搬の制限された高速な加算器を実現できることが知られている．しかし，冗長数系に基づく加算アルゴリズム（冗長加算アルゴリズム）は従来より個別に取り扱われ，その回路構造は経験的に設計がなされてきた．これに対して，CTDによる冗長加算アルゴリズムの統一的な表現は，冗長加算器の系統的な設計を可能にする．提案する設計手法では，まず，桁上げ伝搬の制限された冗長加算アルゴリズムをCTDの等価変換操作により導出する．次に，導出されたCTDの各変数を回路の実装技術に合わせて符号化することで，高性能な冗長加算器を得る．本稿では，例として冗長2進加算器を設計し，提案する設計手法の有効性を示す．

2. Counter Tree Diagram

本章では，CTD（Counter Tree Diagram）について概説する．詳しくは文献3)を参照されたい．

CTDは，カウンタノードと呼ばれるノードの集合とノード間を結ぶ有向辺の集合からなる．カウンタノードは，加算機能（カウンタ機能）を抽象的に表現する．一方，有向辺は，オペランドどうしの依存関係を表現する．すべての有向辺はオペランドの定義域として重みつき区間を持つ．

定義1 重みつき区間 $w[a:b]$ は，次式で表される整数の集合である．

$$w[a:b] \stackrel{\text{def}}{=} \{wa, w(a+1), \cdots, wb\}$$

☆ CTDで表現できる回路は算術演算のみで構成されている加算器である．桁上げ先見加算器や桁上げ選択加算器のように，算術演算のほかにロジックを用いて高速な桁上げを実現するタイプの高速加算器は表現することができない．

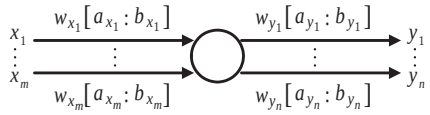

図1 カウンタノード
Fig. 1 Symbol of a counter node.

ここで，$a, b, w \in \mathbf{Z}$，$a \le b$ であり，$\mathbf{Z}$ は整数全体の集合を表す．□

定義2 次式を重みつき区間の加算と定義する．

$$w_1[a_1:b_1] + w_2[a_2:b_2] \stackrel{\text{def}}{=} \{x+y \mid x \in w_1[a_1:b_1] \text{ and } y \in w_2[a_2:b_2]\}$$

□

ここで，$w_2 = cw_1$（$c \in \mathbf{Z}$）かつ $|c| \le |[a_1:b_1]|$（$|c|$ は c の絶対値，$|[a_1:b_1]|$ は集合 $[a_1:b_1]$ の要素数を表す）を満たす場合，上式の加算結果は次式のように単一の重みつき区間として表現することができる．

$$w_1[a_1:b_1] + w_2[a_2:b_2] = w_1[a_1 + \min(ca_2, cb_2) : b_1 + \max(ca_2, cb_2)]$$

一方，m 入力 n 出力のカウンタノード（**図1**）は，m 個の入力変数を n 個の出力変数に変換する素子として，次のように定義される．

定義3 m 個の入力変数を $x_1, \cdots, x_m$，n 個の出力変数を $y_1, \cdots, y_n$ とし，各々の重みつき区間を $w_{x_1}[a_{x_1}, b_{x_1}], \cdots, w_{x_m}[a_{x_m}, b_{x_m}]$，$w_{y_1}[a_{y_1}, b_{y_1}], \cdots, w_{y_n}[a_{y_n}, b_{y_n}]$ とする．このとき，次式を満たすように入力変数を出力変数に変換する素子をカウンタノードと呼ぶ．

$$\sum_{i=1}^{m} x_i = \sum_{j=1}^{n} y_j \tag{1}$$

$$\sum_{i=1}^{m} w_{x_i}[a_{x_i} : b_{x_i}] \subseteq \sum_{j=1}^{n} w_{y_j}[a_{y_j} : b_{y_j}] \tag{2}$$

□

式(1)は，カウンタノードにより変数の総和が変化しないことを表す．また，式(2)は，任意の入力の組合せを出力側で表現できることを意味し，カウンタノード条件と呼ばれる．以上のように定義されたカウンタノードは，多進数系や冗長数系を含むあらゆる重み数系の加算を表現可能である．CTDは，カウンタノードのネットワークとして，加算アルゴリズムをさまざまな抽象度レベルで表現する．

高い抽象度で記述されたCTDは，分解と呼ばれる操作により抽象度の低いCTDに変換される．CTDの分解操作には，有向辺の分解操作とカウンタノードの

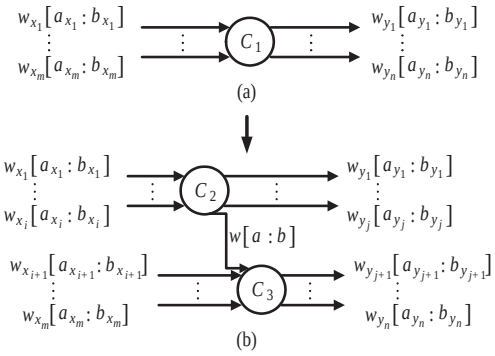

図 2 カウンタノードの分解
Fig. 2 Decomposition of counter nodes.

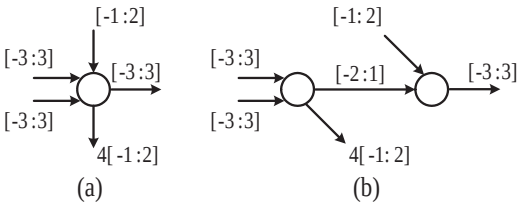

図 3 冗長加算器の CTD 表現：(a) 抽象度の高い表現，(b) (a) のカウンタノードを分解して得られた抽象度の低い表現
Fig. 3 CTD representation of a redundant adder: (a) 1-digit redundant adder at a high level of abstraction, (b) 1-digit redundant adder at a low level of abstraction after node decomposition.

分解操作がある．まず，有向辺の分解操作は，重みつき区間 $w[a:b]$ の有向辺を，次式を満足する $n\,(n>1)$ 本の有向辺に変換する操作である．

$$w[a:b] \subseteq \sum_{i=1}^{n} w_i[a_i:b_i] \tag{3}$$

ここで，$w_1[a_1:b_1],\cdots,w_n[a_n:b_n]$ は分解後の重みつき区間とする．一方，カウンタノードの分解操作は，**図 2** (a) のカウンタノードを図 2 (b) のように複数のカウンタノードに変換する操作である．このとき，以下の包含関係を満たさなければならない．

$$\sum_{k=1}^{i} w_{x_k}[a_{x_k}:b_{x_k}] \subseteq \sum_{l=1}^{j} w_{y_l}[a_{y_l}:b_{y_l}] + w[a:b] \tag{4}$$

$$\sum_{k=i+1}^{m} w_{x_k}[a_{x_k}:b_{x_k}] + w[a:b] \subseteq \sum_{l=j+1}^{n} w_{y_l}[a_{y_l}:b_{y_l}] \tag{5}$$

ここで $a=b=0$ の場合，有向辺 $w[a:b]$ は不要である．以上のように，それぞれの分解操作は，CTD 全体が表現する加算機能を保存する等価変換操作として定義される．分解操作の可否は，分解後の CTD においてカウンタノード条件 (2) を調べることで判定する．

3. CTD に基づく冗長加算器の設計手法

本章では，CTD に基づく冗長加算器の設計手法について述べる．前章で述べた CTD は，桁上げ伝搬の制限された冗長加算アルゴリズムを統一的に表現する．また，その統一的な表現は，算術アルゴリズムに関する専門的な知識を用いることなく，CTD への分解操作により導出できる．提案する設計手法は，上記の特徴を用いることで冗長加算器の系統的な設計を実現する．以下にその概略を示す．

(1) 設計する冗長加算アルゴリズム全体を抽象度の高い CTD で表現する．
(2) 抽象度の高い CTD に有向辺およびカウンタノードの分解操作を行い，抽象度の低い（論理ゲートレベルで実装可能な）CTD に変換する．
(3) 導出した CTD の変数を回路の実装技術に合わせて符号化する．
(4) 符号化により決定された論理関数を回路に変換する．

ここで，2 値論理回路による実装であれば，手順 (4) には論理合成ツールを利用可能である．以下では，(1)〜(3) の手順について説明する．

3.1 冗長加算アルゴリズムの導出

本節では，桁上げ伝搬の制限された冗長加算アルゴリズムを CTD の分解操作により導出する手法を示す．提案する設計手法では，まず，設計する冗長加算アルゴリズム全体を抽象度の高い CTD で表現する．ここでは，並列加算器として実現することを想定し，**図 3** (a) に示す 1 桁の冗長加算アルゴリズムを考える．図 3 (a) の CTD は，入出力の重みつき区間のみが与えられ，桁上げ伝搬に関するいかなる情報も持たない．そのため，この CTD から実現される並列加算器は，最大で入力語長分だけ桁上げ伝搬を発生する可能性がある．一方，図 3 (b) は，図 3 (a) の CTD をより低い抽象度で表現した CTD であり，桁上げ伝搬が 1 桁先までに制限された冗長加算アルゴリズムを表す．図 3 (b) の各カウンタノードを任意の論理素子を用いて実現すれば，桁上げ伝搬の制限された冗長加算器を得られる．以下では，図 3 (a) から図 3 (b) のような CTD の分解を数学的に導出する手法を示す．

ここでは，例として，冗長 2 進（RB）加算アルゴリズムの導出について考える．RB 数系は，基数 2 および桁集合 $[-1:1]\,(=\{-1,0,1\})$ によって定義され

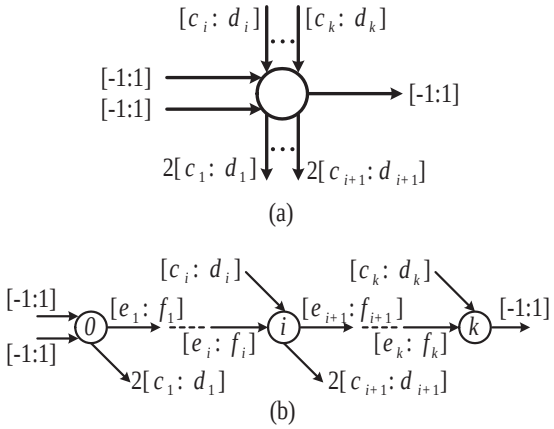

図 4 冗長 2 進（RB）加算器の分解：(a) 1 桁の RB 加算器，(b) (k+1) 段分解された 1 桁の RB 加算器

Fig. 4 CTD decomposition of an Redundant-Binary adder with the radix 2 and the digit set $[-1:1]$: (a) 1-digit adder, and (b) (k+1)-stage decomposition of the 1-digit adder.

る重み数系である．まず，1 桁の RB 加算アルゴリズム全体は**図 4** (a) の CTD で表される．このとき，この CTD を桁上げ伝搬が $k(\geq 1)$ 桁に制限された CTD（図 4 (b)）に分解することを考える．ここで，図 4 の c_i，d_i，e_i，$f_i\,(i \in \{1,2,\cdots,k\})$ は未知の整数パラメータである．これらパラメータの値は，式 (2) のカウンタノード条件から，次式を満足するように選択されなければならない．

$$\begin{cases} [-1:1]+[-1:1] \subseteq 2\,[c_1:d_1]+[e_1:f_1] \\ [c_i:d_i]+[e_i:f_i] \subseteq 2\,[c_{i+1}:d_{i+1}]+[e_{i+1}:f_{i+1}] \\ (i \in \{1,2,\cdots,k-1\}) \\ [c_k:d_k]+[e_k:f_k] \subseteq [-1:1] \end{cases} \tag{6}$$

ただし，第 2 式は $k=1$ の場合は不要である．また，実用的な並列加算器の場合，図 4 (b) の CTD を構成するすべてのカウンタノードの出力は単一の重みつき区間で表現され，以下の条件式が成り立つ．

$$1 \leq f_i - e_i \quad (i \in 1,2,\cdots,k) \tag{7}$$

式 (6)，(7) を満足する整数パラメータ c_i，d_i，e_i，$f_i\,(i \in \{1,2,\cdots,k\})$ が存在するとき，図 4 (a) の CTD を図 4 (b) のように分解することができる．

RB 加算器の場合，$k=1$ のとき式 (6) が解を持たないことから，桁上げ伝搬を 1 桁に制限した構造は実現できないことが分かる．一方，$k=2$ のとき，式 (6) は解を持ち，次の連立不等式が得られる．

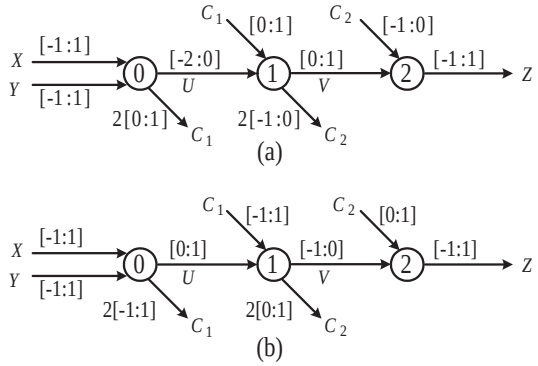

図 5 冗長 2 進加算器を表す 3 段の CTD：(a) Type 1，(b) Type 2

Fig. 5 3-stage CTD representation for Redundant-Binary adders: (a) Type 1, and (b) Type 2.

$$\begin{cases} 2 & \leq & 2\cdot d_1 + f_1 \\ -2 & \geq & 2\cdot c_1 + e_1 \\ d_1 + f_1 & \leq & 2\cdot d_2 + f_2 \\ c_1 + e_1 & \geq & 2\cdot c_2 + e_2 \\ d_2 + f_2 & \leq & 1 \\ c_2 + e_2 & \geq & -1 \\ 1 & \leq & f_1 - e_1 \\ 1 & \leq & f_2 - e_2 \end{cases} \tag{8}$$

式 (8) における未知の整数パラメータ c_i，d_i，e_i，$f_i\,(i \in \{1,2\})$ は，以下に示す必要条件から簡単に求めることができる．

$$\begin{cases} 4 & \leq & 2\alpha_1 + \beta_1 \\ \alpha_1 + \beta_1 & \leq & 2\alpha_2 + \beta_2 \\ \alpha_2 + \beta_2 & \leq & 2 \\ 1 & \leq & \beta_1 \\ 1 & \leq & \beta_2 \end{cases} \tag{9}$$

変数 α_i，$\beta_i (i \in \{1,2\})$ は，それぞれ $\alpha_i = d_i - c_i$ および $\beta_i = e_i - f_i$ と与えられる．式 (9) の必要条件から，次の解が得られる．

- Type 1　$\alpha_1 = 1, \beta_1 = 2, \alpha_2 = 1, \beta_2 = 1$
- Type 2　$\alpha_1 = 2, \beta_1 = 1, \alpha_2 = 1, \beta_2 = 1$

ここで，c_i，d_i，e_i，$f_i\,(i \in \{1,2\})$ の値を適当に決定すると，**図 5** に示す Type 1 と Type 2 の RB 加算アルゴリズムが得られる．

このように，CTD を用いることで，数系や算術演算回路特有の構造に関する知識を用いることなく，高速な冗長加算アルゴリズムを数学的に導出することができる．著者らは，よく知られた RB 加算器の回路実現例がすべて Type 1 の CTD に分類できることを確

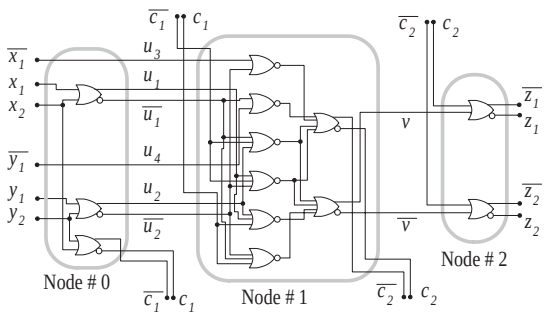

図 6 文献 5) の冗長 2 進加算器構造
Fig. 6 Circuit implementation of the Redundant-Binary adder from Ref. 5).

表 1 図 6 の冗長 2 進加算器における CTD 変数の 2 値符号化
Table 1 Binary coding of CTD signals in the Redundant-Binary adder shown in Fig. 6.

$X/Y/Z$	$x_1x_2/y_1y_2/z_1z_2$
1	10
0	00
-1	01

U	$u_1u_2u_3u_4$
0	110∗ or 1110
-1	10∗1 or 011∗
-2	0011 or 1111

V	v
1	0
0	1

C_1	c_1
1	1
0	0

C_2	c_2
0	0
-1	1

認している．Type 1 の CTD 変数 X，Y，Z，U，V，C_1 および C_2 をそれぞれ適当な 2 値信号で符号化することで，文献 5)〜7) の各回路が得られる．例として，文献 5) で報告されている RB 加算器を**図 6** に示す．この回路構造は，**表 1**（表中の ∗ は 0 もしくは 1 を表す）のように CTD 変数を 2 値符号化した Type 1 の RB 加算器と解釈できる．

3.2 冗長加算アルゴリズムの 2 値符号化

本節では，冗長加算アルゴリズムの論理関数を決定する手法について述べる．CTD で表現された加算アルゴリズムの論理関数は，その CTD 変数への符号化により決定される．そのため，高性能な冗長加算器を設計するには，無数に存在する符号化の中から，いかに適切な符号化を選択するかが重要となる．本稿では，2 値論理による冗長加算器の実装を想定し，優れた回路構造を実現するための 2 値符号化手法について述べる．

本稿で示す 2 値符号化手法は，回路の配線数や素子数を削減するため，CTD の各変数を最低限のビット数で符号化することを基本とする．たとえば，5 値をとる変数は 3 ビット，10 値をとる変数は 4 ビットに符号化する．そのうえで，入出力変数の場合，当該数系と 2 進数系との変換を簡単化するように符号を割り当てる．以下では，図 5 に示す RB 加算アルゴリズム

表 2 図 5 (a) における CTD 変数の 2 値符号化例
Table 2 Example of binary coding for CTD variables in Fig. 5 (a).

$X/Y/Z$	$x_1x_2/y_1y_2/z_1z_2$
1	10
0	00
-1	01

U	u_1u_2
0	00 or 01
-1	10
-2	11

V	v
1	1
0	0

C_1	c_1
1	0
0	1

C_2	c_2
0	1
-1	0

を例に，上記の観点に基づく 2 値符号化手法を示す．

まず，RB 数表現の入出力変数 X，Y および Z への符号化を考える．入力変数 X は，-1，0，1 の値をとるため，2 ビットの信号 x_1x_2 で表される．本稿の符号化手法では，x_1，x_2 をそれぞれ正/負信号に分けて，次のように符号化する．

$$\begin{aligned} X = 1 &\quad\leftrightarrow\quad x_1x_2 = 10 \\ X = 0 &\quad\leftrightarrow\quad x_1x_2 = 00 \\ X = -1 &\quad\leftrightarrow\quad x_1x_2 = 01 \end{aligned} \tag{10}$$

ここで，$x_1x_2 = 11$ は X の値に割り当てない．本符号化は，RB 数表現と 2 進数表現との変換を簡単な回路で実現できるため，実用上広く用いられる[1)]．他の入出力変数 Y，Z についても同様に 2 値符号化する．

次に，内部変数 U，V，C_1 および C_2 では，3 値をとる内部変数を 2 ビット，2 値をとる内部変数を 1 ビットに符号化する．Type 1 の場合，-2，-1，0 の値をとる U が 2 ビットの信号 u_1u_2 に符号化される．また，V，C_1 および C_2 の値は，それぞれ 1 ビットの信号 v，c_1，c_2 に符号化される．ここで，U の符号化には，ある U の値に u_1u_2 の値を冗長に割り当てる符号化と，U と u_1u_2 の値を 1 対 1 に割り当てる符号化とが存在する．前者は，1 段目のカウンタノードから得られる回路を縮小する反面，次段のカウンタノードから得られる回路を増大する．一方，後者はその逆である．その優劣は実装により異なる．Type 1 の各変数を 2 値符号化した例を**表 2** に示す．ここで，U の符号化では，U の値 0 に u_1u_2 の値が冗長に割り当てられている．

一方，Type 2 の場合，-1，0，1 の値をとる C_1 が 2 ビットの信号 $c'_{11}c'_{12}$ に符号化される．また，U，V および C_2 の値は，それぞれ 1 ビットの信号 u'，v'，c'_2 に符号化される．Type 2 の各変数を 2 値符号化した例を**表 3** に示す．ここで，C_1 の符号化では，C_1 と $c'_1c'_2$ の値を 1 対 1 に割り当てられている．

上記の符号化手法から得られる RB 加算器は，Type 1 で 5760 通り，Type 2 で 1536 通りある．ここで，総

表 3 図 5 (b) における CTD 変数の 2 値符号化例
Table 3 Example of binary coding for CTD variables in Fig. 5 (b).

$X/Y/Z$	$x_1x_2/y_1y_2/z_1z_2$
1	10
0	00
-1	01

C_1	$c'_{11}c'_{12}$
-1	00
0	10
1	11

U	u'
1	1
0	0

V	v'
0	0
-1	1

C_2	c'_2
0	1
1	0

数の違いは，Type 1 の方が Type 2 よりも RB 加算器を実現する CTD 変数の組合せが多いことに起因する．2 値論理による実装の場合，本稿で示す 2 値符号化から得られる回路構造が 1 つの基準となると考えられる．ただし，最適な符号化は用途や実装技術により異なるため，設計者は必要に応じて符号化手法を検討する必要がある．

4. 冗長 2 進加算器の設計

本章では，冗長加算器の実験的な設計から，提案する設計手法の有効性を示す．ここでは，前章で示した 2 値符号化手法を用いた冗長 2 進（RB）加算器の設計を考える．実験では，まず，提案する設計手法により，RB 加算器を系統的に設計できることを示す．次に，前章で示した符号化手法から得られる RB 加算器を網羅的に設計し，従来の RB 加算器と性能を比較する．

4.1 実 験 1

実験 1 では，CTD に基づく設計手法により，桁上げ伝搬の制限された冗長加算器を系統的に設計できることを示す．従来，冗長加算器は，図 4 (a) に示す 1 段の CTD（入出力の数系情報のみ）から設計されてきた．そのため，論理式の簡単化を基本とする論理合成の適用は難しく，設計者は桁上げ伝搬が発生しないよう論理レベルで手設計する必要があった．一方，提案する設計手法では，図 5 に示す 3 段の CTD を設計に用いるため，桁上げ伝搬の制限された冗長加算器をつねに論理合成から得ることができる．本実験では，1 段の CTD と 3 段の CTD から得られる HDL（Hardware Description Language）記述をそれぞれ論理合成し，合成される回路構造を比較する．

3 段の CTD を HDL 記述へ変換する手順を以下に示す．まず，前章で示した符号化手法により，CTD の各変数に 2 値の符号を割り当てる．符号化により，各カウンタノードの論理関数（真理値表）が決定される．次に，得られた論理関数を VHDL で記述する．本実験では，論理関数を最小論理和形で表し，それをテクノロジ非依存の論理演算子と信号代入文により記述した．ここでは，カウンタノードごとに独立したエンティティとして記述する．一方，加算器全体を表すエンティティでは，各カウンタノードをコンポーネントインスタンスとして宣言し，1 桁の RB 加算器を 3 段の CTD で表す．ここで，1 段目のキャリー出力は次桁 2 段目のキャリー入力，2 段目のキャリー出力は次桁 3 段目のキャリー入力とする．

同様の手順により，1 段の CTD も HDL 記述に変換する．ただし，キャリーの入出力は，-1，0，1 の値をとる変数として 2 ビットに符号化する．また，加算器全体を表すエンティティでは，1 桁の RB 加算器を 1 段の CTD で表す．

1 段と 3 段の CTD から得られる HDL 記述をランダムに 50 通りずつ選択し，それぞれ入力語長 8 桁と 16 桁の場合について論理合成を行う．論理合成には，Synopsys 社の Design Compiler を用いる．本実験では，回路構造を大域的に最適化するため，Design Compiler のコンパイル時に “-ungroup - all -flatten” のオプションを付ける．各 HDL 記述に対して，目標とする遅延時間を 0.5 ns から 2.2 ns まで 0.1 ns 刻みに設定し，それぞれの設定値で合成する．消費電力については，つねに最小を目標とした．

回路性能の評価には，ROHM 社の CMOS 0.35 μm プロセス（京都大学提供の低電力版スタンダードセルライブラリ．電源電圧 3.0 V，温度 25 度[11]）を利用する．本実験では，遅延時間と消費電力により得られた回路を評価する．ただし，それらの解析には，回路シミュレーションにより得られるスイッチング情報は考慮しない．すなわち，遅延時間は，ライブラリの非線形遅延モデルにより概算される．また，消費電力は，ライブラリのゲート容量と配線負荷モデルにより概算される動的電力消費と貫通電流による電力消費の和とする．ここで，スイッチングアクティビティは 1 GHz，論理値の遷移確率は 0.5 とする．参考までに，本実験で使用したセルライブラリの全加算器セル（標準駆動力）の遅延時間と消費電力は，それぞれ 0.59 ns，1.32 mW となる．

8 桁および 16 桁 RB 加算器の合成結果をそれぞれ **図 7**，**図 8** に示す．ここで，×印は 1 段の CTD から合成された回路，+印は 3 段の CTD から合成された回路を表す．横軸と縦軸はそれぞれ合成された回路の遅延時間と消費電力である．図 8 では，遅延時間と消費電力の範囲が図 7 の 2 倍であることに注意されたい．図 7 と図 8 から，3 段の CTD から合成された回路がつねに桁上げ伝搬最小の RB 加算器となることが

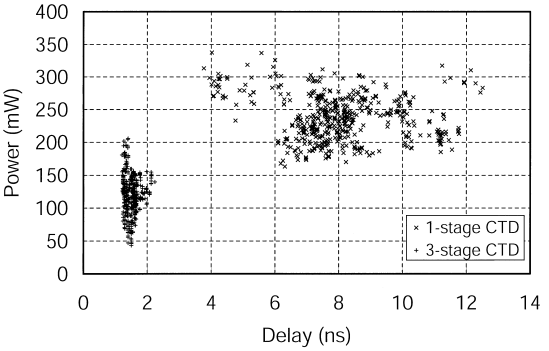

図 7 実験 1 における 8 ビット冗長 2 進加算器の合成結果
Fig. 7 Synthesis results of 8-bit Redundant-Binary adders in Experiment 1.

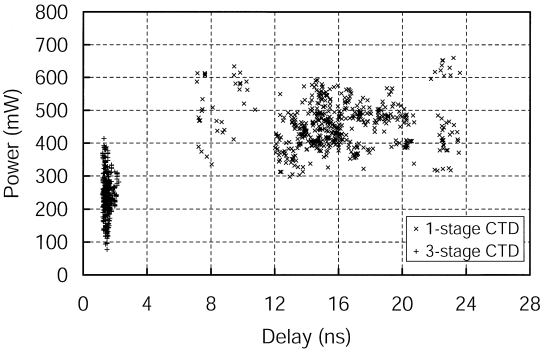

図 8 実験 1 における 16 ビット冗長 2 進加算器の合成結果
Fig. 8 Synthesis results of 16-bit Redundant-Binary adders in Experiment 1.

確認できる．一方，1 段の CTD から合成される RB 加算器には，論理合成による最適化にもかかわらず，入力語長分だけ桁上げ伝搬の発生する構造が含まれた．このように，提案する設計手法は，従来の設計手法では適用の困難だった論理合成を用いて，高性能な冗長加算器を合成できる．すなわち，設計者は，冗長加算アルゴリズムに関する知識を用いることなく，冗長加算器を系統的に設計することが可能となる．

4.2 実 験 2

実験 2 では，本稿で提案する設計手法から得られる冗長加算器の性能を評価する．提案する設計手法の特長は，CTD の Type とその符号化を組み合わせることにより，多様な回路構造を設計できることにある．そこで，前章で示した符号化手法から得られる RB 加算器を網羅的に設計し，CTD の Type およびその 2 値符号化が RB 加算器の性能に与える影響を確認する．また，得られる RB 加算器の性能を従来の RB 加算器と比較する．

本実験では，本稿で示す符号化手法から得られる Type1 と Type2 の符号化をすべて用いて，入力語長 4 桁の RB 加算器を設計する．各 RB 加算器は，前節で示した手順に従って論理合成される．実験方法（Design Compiler のコンパイルオプションや消費電力の評価方法など）は実験 1 と同一とする．

Type 1 および Type 2 の合成結果をそれぞれ**図 9**, **図 10** に示す．本実験では遅延時間の最適化が消費電力よりも優先される．そのため，合成された回路の遅延時間はおおむね設定した目標値の近傍となる．合成された回路が各目標値に集中して存在する様子は図 9, 図 10 から確認できる．ここで，ある符号化から得られる回路の電力遅延積が一定であるとすると，符号化による性能差は消費電力にして 4 倍程度にまで達することになる．たとえば，図 9 の遅延時間 1.3 ns のとき，消費電力はおおよそ 25 mW から 100 mW までの値をとる．このように，同じ Type に分類される RB 加算器でも，わずかな符号化の違いにより性能に大きな差が生じる．なお，本実験では消費電力と回路遅延の関係を評価したが，回路の面積についても消費電力と同程度の影響を受けるものと予想される．さらに，本実験結果により，RB 加算器に関して次のことが確認できた．

- 従来知られていなかった Type 2 の RB 加算アルゴリズムでも実用的な回路を実現できる．
- 本実験条件では，Type 1 の方が Type 2 よりも回路遅延，消費電力ともに優位な傾向にある．

従来の RB 加算器[5)~7)]と提案する設計手法から得られた RB 加算器の最高性能を**表 4** に示す．ここで，文献 5)~7) の RB 加算器は，文献に示される回路構造のゲートレベルネットリストから合成された．ここで，ネットリストの HDL 記述は，回路内部の信号を中間信号として宣言し，テクノロジ非依存の論理演算子と信号代入文で与えた．合成方法および評価方法は実験 1 と同一である．表 4 より，本実験で得られた Type 1 の RB 加算器が消費電力，遅延時間および電力遅延積の面で従来の RB 加算器よりも優れていることを確認できる．参考までに，本実験で得られた電力遅延積最小の 2 値符号化を**表 5** に示す．

5. む す び

本稿では，Counter Tree Diagram（CTD）と呼ばれる高速加算アルゴリズムの統一的な表現法に基づく冗長加算器の設計手法を提案した．例として冗長 2 進（RB）加算器を設計し，数系やその演算アルゴリズムに関する専門的な知識を用いることなく，冗長加算器

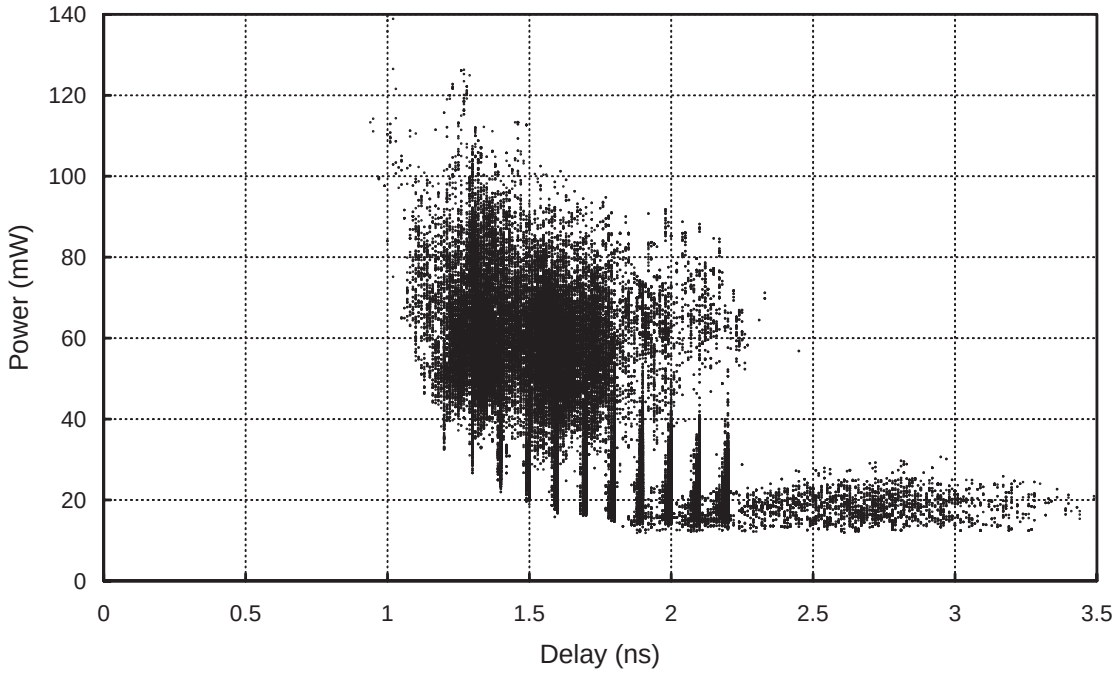

図 9 実験 2 における 4 ビット冗長 2 進加算器（Type 1）の合成結果
Fig. 9 Synthesis results of 4-bit Type-1 Redundant-Binary adders in Experiment 2.

表 4 実験 2 における冗長 2 進加算器の性能
Table 4 Performance of Redundant-Binary adders in Experiment 2.

	power-delay product (nsec∗mW)	delay (nsec)	power (mW)
Type 1	**22.28**	**0.94**	**11.84**
Type 2	28.87	1.19	14.43
RBA 5)	26.62	1.21	13.58
RBA 6)	23.45	1.20	12.93
RBA 7)	25.85	1.17	14.09

表 5 実験 2 における電力遅延積最小となる 2 値符号化
Table 5 Optimal binary coding of CTD signals in Experiment 2.

$X/Y/Z$	$x_1x_2/y_1y_2/z_1z_2$
1	10
0	00
-1	01

U	u_1u_2
0	00
-1	10 or 11
-2	01

V	v
1	1
0	0

C_1	c_1
1	1
0	0

C_2	c_2
0	0
-1	1

を系統的に設計できることを示した．また，提案する設計手法により RB 加算器を網羅的に設計し，多様な回路構造が得られることを確認した．今後は，さらに多くの冗長加算器の解析・設計を通して提案する設計手法の総合的な評価を行い，CTD に基づく冗長加算器の自動合成システムを構築する予定である．

参 考 文 献

1) Koren, I.: *Computer arithmetic algorithms, 2nd Edition*, A K Peters (2001).
2) Aoki, T. and Higuchi, T.: Beyond-binary arithmetic — Algorithms and VLSI implementations, *Interdisciplinary Information Sciences*, Vol.6, No.1, pp.75–98 (2000).
3) Sakiyama, J., Homma, N., Aoki, T. and Higuchi, T.: Counter Tree Diagrams: A unified framework for analyzing fast addition algorithms, *IEICE Trans. Fundamentals*, Vol.E86-A, No.12, pp.3009–3019 (2003).
4) Sakiyama, J., Aoki, T. and Higuchi, T.: Counter tree diagrams for design and analysis of fast addition algorithms, *Proc. 32nd IEEE Int. Symp. Multiple-Valued Logic*, pp.91–98 (May 2003).
5) Takagi, N., Yasuura, H. and Yajima, S.: High-speed VLSI multiplication algorithm with a redundant binary addition tree, *IEEE Trans. Computers*, Vol.34, No.9, pp.789–796 (1985).
6) Kuninobu, S., Nishiyama, T. and Taniguchi, T.: High speed MOS multiplier and divider using redundant binary representation and their implementation in a microprocessor, *IEICE*

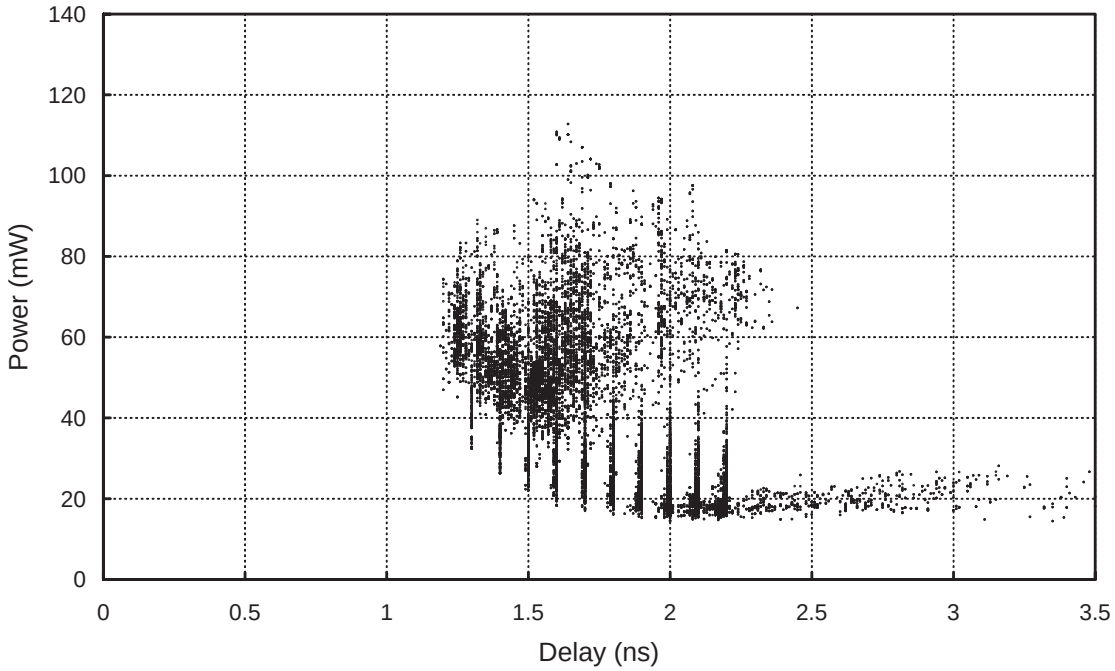

図 10 実験 2 における 4 ビット冗長 2 進加算器（Type 2）の合成結果
Fig. 10 Synthesis results of 4-bit Type-2 Redundant-Binary adders in Experiment 2.

Trans. Electron., Vol.E76-C, No.3, pp.436–445 (1993).
7) Makino, H., Nakase, Y., Suzuki, H., Morinaka, H., Shinohara, H. and Mashiko, K.: An 8.8-ns 54×54-bit multiplier with high speed redundant binary architecture, *IEEE J. Solid-State Circuits*, Vol.31, No.6, pp.773–783 (1996).
8) Kawahito, S., Kameyama, M., Higuchi, T. and Yamada, H.: A 32×32-bit multiplier using multiple-valued MOS current-mode circuits, *IEEE J. Solid-State Circuits*, Vol.23, No.1, pp.124–132 (1988).
9) Kawahito, S., Ishida, M., Nakamura, T., Kameyama, M. and Higuchi, T.: High-speed area-efficient multiplier design using multiple-valued current-mode circuits, *IEEE Trans. Computers*, Vol.43, No.1, pp.34–42 (1994).
10) Ohkubo, N., Suzuki, M., Shinbo, T., Yamanaka, T., Shimizu, A., Sasaki, K. and Nakagome, Y.: A 4.4 ns CMOS 54×54-b multiplier using pass-transistor multiplexer, *IEEE J. Solid-State Circuits*, Vol.30, No.3, pp.251–257 (1995).
11) Hashimoto, M., Fujimori, K. and Onodera, H.: Standard cell libraries with various driving strength cells for 0.13, 0.18, and 0.35 μm technologies, *Proc. Asia and South Pacific Design Automation Conference 2003*, pp.589–590 (2003).

(平成 15 年 10 月 21 日受付)
(平成 16 年 3 月 5 日採録)

本間 尚文（正会員）
平成 9 年東北大学工学部情報工学科卒業．平成 13 年同大学大学院情報科学研究科博士課程修了．博士（情報科学）．同年同大学院情報科学研究科助手．平成 14 年より科学技術振興機構さきがけ研究者兼任，現在に至る．ハードウェアアルゴリズム，LSI 設計手法に関する研究に従事．電子情報通信学会，IEEE 各会員．

崎山 淳
平成 11 年東北大学工学部情報工学科卒業．平成 13 年同大学大学院情報科学研究科修士課程修了．現在，同大学院情報科学研究科博士課程に在学中．算術演算アルゴリズムの解析，設計，自動合成手法に関する研究に従事．

若松　泰平

平成 15 年東北大学工学部情報工学科卒業．現在，同大学大学院情報科学研究科修士課程に在学中．冗長数系に基づく算術演算回路の設計に関する研究に従事．

青木　孝文（正会員）

昭和 63 年東北大学工学部電子工学科卒業．平成 4 年同大学大学院工学研究科博士課程修了．工学博士．同年同大学工学部電子工学科助手，平成 6 年同大学院情報科学研究科助手，平成 8 年同助教授，平成 14 年同教授．平成 9 年〜11 年科学技術振興事業団さきがけ研究 21 研究者兼任，現在に至る．超高速ディジタル計算の理論，画像センシング，映像信号処理，バイオメトリクス，VLSI 設計技術，多値論理，分子コンピュータの基礎理論に関する研究に従事．IEEE，電子情報通信学会，計測自動制御学会各会員．

樋口　龍雄（正会員）

昭和 37 年東北大学工学部電子工学科卒業．昭和 44 年同大学大学院工学研究科博士課程修了．工学博士．昭和 42 年同大学工学部電子工学科助手，昭和 45 年同助教授，昭和 55 年同教授．平成 5 年同大学院情報科学研究科教授．平成 6 年〜10 年同情報科学研究科長併任．平成 7 年〜13 年東北大学情報処理センター長併任．平成 15 年東北工業大学工学部電子工学科教授・東北大学名誉教授，現在に至る．この間，ディジタル信号処理，特に 1 次元および多次元ディジタルフィルタの統一的設計ならびに信号処理プロセッサのアーキテクチャ，および多値集積回路，超多値オプト・バイオコンピューティングの研究に従事．IEEE，電子情報通信学会，計測自動制御学会各フェロー．