計測自動制御学会東北支部 第187回研究集会 (2000.6.2)
資料番号 187-13

高速冗長2進加減算器の開平器への応用

Application to Square Rooting Circuit for High-Speed Redundant Binary Adder-Subtractor

○日野杉充希*，恒川佳隆*，三浦守*

○Mitsuki Hinosugi*, Yoshitaka Tsunekawa*, Mamoru Miura*

*岩手大学

*Iwate University

キーワード： 冗長2進表現 (redundant binary representation)，高速 (high-speed)，加減算器 (adder-subtractor)，VLSI評価 (VLSI evaluation)，開平器 (square rooting circuit)

連絡先： 〒020-8551 盛岡市上田4-3-5 岩手大学 工学部
日野杉 充希，Tel.: (019)621-6468，Fax.: (019)621-6468，E-mail: mitsuki@cis.iwate-u.ac.jp

1. はじめに

算術演算の中で加算・減算は最も基本となるものであり，乗算や除算をはじめあらゆる算術演算がこれらの繰り返しで行われる．そのため，加減算器の性能は，実現するハードウェアの処理能力に大いに依存してくる．しかし，ディジタルシステムにおけるこれまでの数値表現は補数表現が中心であり，その特徴から減算も加算に置き換えられて行われることが一般的であった．

減算から加算への変換は，減数の符号を反転させることにより実現することができ，この操作は符号変換器を加算器の前に取り付けることで行われる．この方法により処理速度は低下するが加算器だけで加減算両方の処理を行うことができ，規模の小さい回路が実現できる．このように，これまでの回路設計は高速性よりも回路規模を重視した構成が一般的であった．

しかし，近年の集積回路技術の発達に伴い大規模回路の実現が可能となってきているため，回路規模よりも高速性を重視した回路構成も今後有効になってくるものと考えられる．実際に，マルチメディアのようなリアルタイム処理を必要とする分野においては，より高速性が要求されている．また，近年は高速性ばかりでなく，情報セキュリティなど種々の分野において高精度演算の重要性が高まっている．

高速性および高精度性を同時に要求される問題に対応する数体系の1つとしてSD (Signed Digit) 表現がある．本報告ではこのSD表現の一種である冗長2進数を用いた演算について検討を行う．この表現法は桁上げの伝搬に依存しない各桁独立の演算が可能となるため桁数に関係なく一定時間で加減算を行える．しかし，これまでこの表現法においても減算は加算器を用いて行われることが一般的であり，減算器について検討は行われてこなかった．

そこで，われわれはこの問題に対し，これまで従来の減算法を用いた加減算器よりも極めて高速な加減算器を提案してきた[1]．これは，符号変換器を必要とせず，しかも，われわれが既に提案してきた1桁2ビット/3ビット混合表現を用いた高速冗長2進加算器と全く同等の処理時間での演算を可能にする[2]．

本稿では，本加減算器の応用として，開平器にこれを適用した場合の検討を行う．これまで開平器の構成法としてはいくつか提案されてきている

Table 1 Computation rules for first step in addition without carry propagation.

a_i	b_i	a_{i+1}, b_{i+1}	c_i	s_i
$\bar{1}$	$\bar{1}$	—	$\bar{1}$	0
0	$\bar{1}$	Neither is negative	0	$\bar{1}$
$\bar{1}$	0	Otherwise	$\bar{1}$	1
$\bar{1}$	1			
1	$\bar{1}$	—	0	0
0	0			
1	0	Neither is negative	1	$\bar{1}$
0	1	Otherwise	0	1
1	1	—	1	0

が，本報告では冗長2進表現を用いた開平用ハードウェアアルゴリズムに基づく高速化手法について考察する．そして，ここで提案された構成法に対してVLSI設計システムPARTHENONを用いてVLSI設計および評価を行い，本加減算器の有効性を明らかにする．

2. 従来の減算法を用いた冗長2進加減算器

2.1 冗長2進表現を用いた加算方法

冗長2進表現は，SD表現の一種である[3]．基数は2で，各桁は$\{\bar{1},0,1\}$の要素である．ここに$\bar{1}$は-1を表す．通常の2進表現と同様，小数点以下i桁目は2^{-i}の重みをもち，$[a_0.a_1\cdots a_n]_{SD2}(a_i \in \{\bar{1},0,1\})$は，$\sum_{i=0}^{n} a_i \cdot 2^{-i}$という数を表す．冗長2進表現には冗長性があり，1つの数を幾通りかに表すことができる．この冗長性を利用し，2つの冗長2進数の加算を桁上げの伝搬なしに高速に行うことができる．

桁上げ伝搬のない加算は，2つのステップで行う[4]．ステップ1では，各桁で，$2c_i + s_i = a_i + b_i$となるように，中間桁上げ$c_i(\in \{\bar{1},0,1\})$と中間和$s_i(\in \{\bar{1},0,1\})$を決める．ここに，a_iとb_iはそれぞれ被加数と加数の桁である．この時s_iとc_{i+1}が同時に$\bar{1}$あるいは1にならないように，1つ下位の桁のa_{i+1}とb_{i+1}も調べて，c_iとs_iを決める．表1にステップ1での計算規則を示す．ステップ2では，各桁で，$z_i = s_i + c_{i+1}$を計算し，和$z_i(\in \{\bar{1},0,1\})$を求める．このとき桁上げは生じない．このように，2つの冗長2進数の加算において桁上げの伝搬をなくすことができ，したがって，組合せ回路による並列加算を演算数の桁数に無関係な一定時間で行うことができる．

2.2 従来の減算法とその手法を用いた加減算器の構成

現在ディジタルシステムにおける数値表現は，2の補数形式が広く用いられている．この数体系は減算の操作を加算に置き換えて演算できるということから，減算器を用いる必要がないことが大きな特長となっている．このことから，減算器は加算器を用いて行うことが一般的となっていた．

また，本研究で扱う数体系の冗長2進数においても2の補数表現の場合と同様に，減算の方法は減数の符号を変換して加数にしてそれを加算するという手法が一般的であった．その従来の減算法を用いた冗長2進加減算器の構成を図1に示す．図1をみてわかるように，減数入力の前に符号変換器を取り付ける構成となるため，その分加算器よりも遅延時間が大きくなる．なお，この従来の加減算器の遅延時間は，われわれの検討結果より加算器の遅延時間よりも約1.3倍も大きくなることをすでに明らかにしている[1]．

3. 高速冗長2進加減算器

前章では，従来の減算法を用いた加減算器が符号変換用の回路を加算器の前に接続した構成となるため，処理時間が大きくなることを述べた．このことから，例えば加減算を繰り返す操作を行うような開平器または除算器に適用する場合を考えると，これが原因となって処理時間が大きく増加するという問題が生じる．

そこでわれわれはこの問題を解決するために，まずこれまで検討が十分行われてこなかった減算器について考察する．そして，その減算器の検討に基づいて高速加減算器を構成する．

3.1 新たな冗長2進数に基づく減算法

本節では，減算器を構成する上で基本となる減算用の計算規則を提案し，それについて述べていく．

まず，減算を加算と同じ処理時間で実行することを目的として，減算方法を加算と同様に2つのステップで考える．ステップ1では，各桁で$2e_i + d_i = a_i - b_i$となるように中間桁借り$e_i(\in \{\bar{1},0,1\})$と中間差$d_i(\in \{\bar{1},0,1\})$を決める．この時d_iとe_{i+1}が同

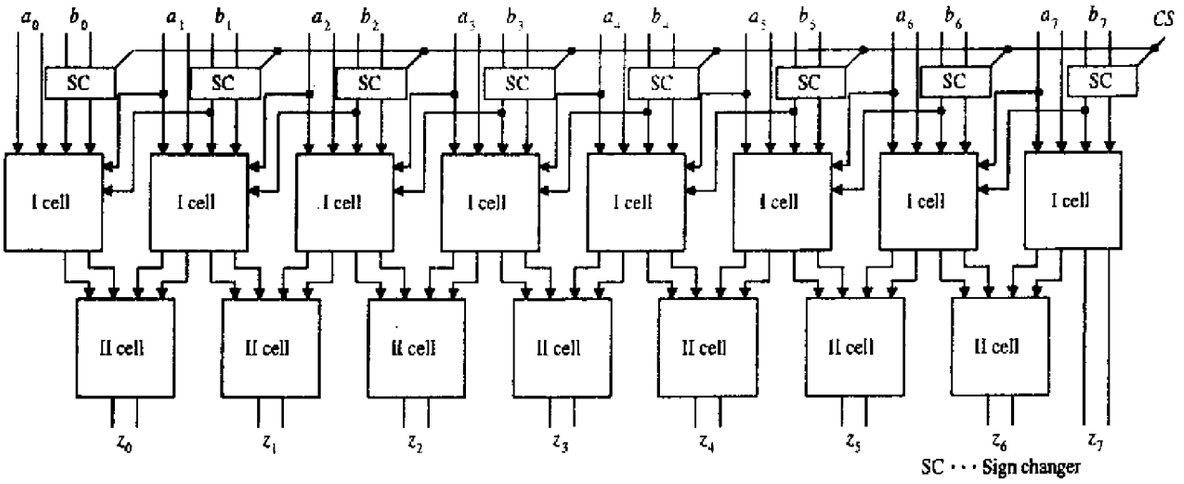

Fig. 1 Structure of redundant binary adder-subtractor using conventional subtraction method.

Table 2 Computation rules for first step in subtraction without borrow propagation.

a_i	b_i	a_{i+1}, b_{i+1}	e_i	d_i
1	1	—	1	0
1	0	$a_{i+1}=1$ or $b_{i+1}=1$	0	1
0	1	Otherwise	1	1
1	1	—	0	0
1	0			
0	1			
0	0			
0	1	$a_{i+1}=1$ or $b_{i+1}=1$	1	1
1	0	Otherwise	0	1
1	1	—	1	0

時に1あるいは1にならないように，1つ下位の桁のa_{i+1}とb_{i+1}も調べて，e_iとd_iを決める．表2にステップ1での計算規則を示す．そして，ステップ2ではステップ1で生成されたe_iとd_iに対して$w_i = d_i + e_{i+1}$を計算し，差w_iを求める．ステップ2に関しては加算器と同じ処理で行うことができる．つまり，減算を加算と同じように2つのステップに分けて，ステップ1で中間差，中間桁借りを生成する方法をとることにより，ステップ2の処理を行うIIセルを共有化させることが可能となる．このことにより，後に述べる加減算器を効率よく構成することができる．

3.2 1桁2ビット/3ビット混合表現の適用

本加減算器の構成法の目的は，冗長2進加減算器の高速化にある．そこで，減算器の具体的な構成法として，われわれがこれまでに提案してきた冗長2進加算器で用いた表現法，すなわち，1桁2ビット/3ビット混合表現の適用について考える．

従来では，1桁を2ビットで表現することが一般的であった．これは，1桁の要素$\{\bar{1}, 0, 1\}$の3つを表現するのに十分だからである．しかし，われわれのこれまでの検討によって，1桁を2ビットで表現することはどの方法を用いたとしてもある1つの要素を検出するのに必ず2ビット参照しなければならない．したがって，従来の表現法ではハードウェア量および処理時間が増加することがわかっている．そこで，この問題を解決するために，われわれは入出力を3ビット，そして，中間桁上げおよび中間和に対しては2ビットを参照する表現法を提案している[2]．この表現法を用いることによって，従来の2ビット表現に対しハードウェア量を抑えつつも高速な加算器が実現できる．

そこで，本報告で述べる減算器においてもこの表現法を適用する．この表現法を用いた減算器の構成を図2(a)に示す．図2(a)から遅延時間およびゲート数を算出するとわれわれがすでに提案した高速加算器と全く同じである[1]．また，われわれの高速加算器(図2(b))と比べ，非常によく似た構成となることもわかる．これは，本加算器および本減算器が1桁3ビット表現の0を表現するビットを容易に検出できる特長をうまく利用した構成法だからである．なお，この特徴は次節で述べる加減算器を構成する上において，有益なものとなることに注目していただきたい．

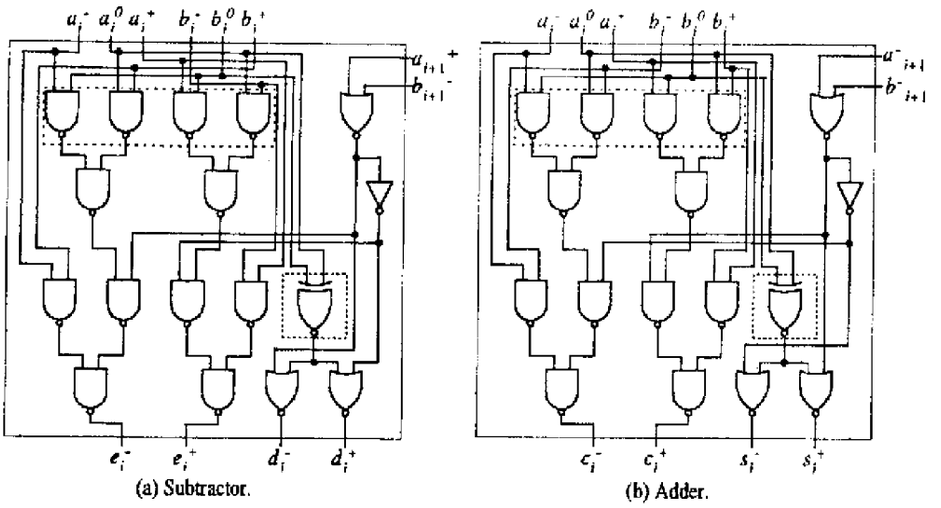

Fig. 2 Structure of each I cell representing each digit by hybrid 2 bits / 3 bits.

3.3 高速加減算器の構成法

本節では，前節の減算器の検討結果を基に構成した高速冗長2進加減算器の構成について述べる．これは，われわれがすでに提案してきた高速冗長2進加算器と全く同等の処理時間で加減算を行うことが可能である．

本報告で提案する加減算器は，われわれがすでに提案した高速加算器と前節の高速減算器を並列に組み合わせた構成である．しかし，単に組み合わせただけでは，高速加算器よりも処理時間が大きくなり，また，ハードウェア量も大きくなってしまう．そこで，われわれは以下のように加減算器を構成する．

最初に，ハードウェア量の問題に関しては，加算器および減算器のIセルだけを組み合わせてハードウェア量の効率化を図る．そして，加算器および減算器を組み合わせる際に，前節で述べた加算器と減算器の構成が似ているという特徴を用いる．すなわち，共有できるゲートを利用することである(図2の波線で囲まれた部分)．

次に，処理時間の問題に関しては，本加減算器の雛形として図3に示すような加算器および減算器の出力を制御信号によって選択するセレクタを用いた構成について最初に考える．ここで，容易に演算が決定できるよう本加減算器では制御信号を2ビット設けることにする．制御信号CSを加算信号CS^+，減算信号CS^- として，図3に示すセレクタを用いた場合の中間桁上げC_iおよび中間和S_i

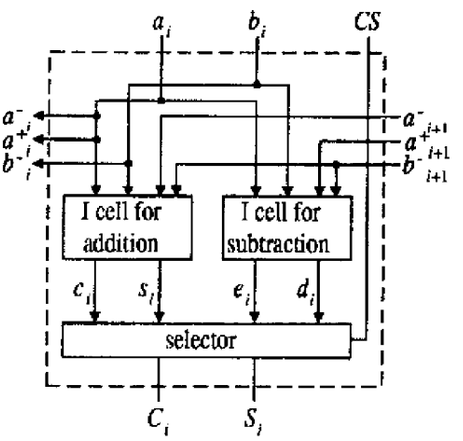

Fig. 3 I cell of adder-subtractor combined I cell of adder / subtractor.

の論理式は以下のようになる．

$$C_i^- = \overline{\overline{a_i^- \cdot b_i^- \cdot \overline{m1_i \cdot \overline{nn_i}} \cdot CS^+} \cdot \overline{a_i^- \cdot b_i^+ \cdot m2_i \cdot ns_i} \cdot CS^-} \quad (1)$$

$$C_i^+ = \overline{\overline{a_i^+ \cdot b_i^+ \cdot \overline{p1_i \cdot nn_i} \cdot CS^+} \cdot \overline{a_i^+ \cdot b_i^- \cdot \overline{p2_i \cdot \overline{ns_i}} \cdot CS^-}} \quad (2)$$

$$S_i^- = \overline{a_i^0 \oplus b_i^0 + \overline{nn_i \cdot CS^+} + \overline{ns_i} \cdot CS^-} \quad (3)$$

$$S_i^+ = \overline{a_i^0 \oplus b_i^0 + \overline{nn_i} \cdot CS^+ + ns_i \cdot CS^-} \quad (4)$$

ここで，式(3)および(4)に関しては，先程述べたゲートの共有化を考慮した式である．これらの式からわかるように，制御信号によって論理が多段化されたため，加算器よりも遅延時間が大きくなる．

そこで，最初に中間桁上げC_iに対し，制御信号

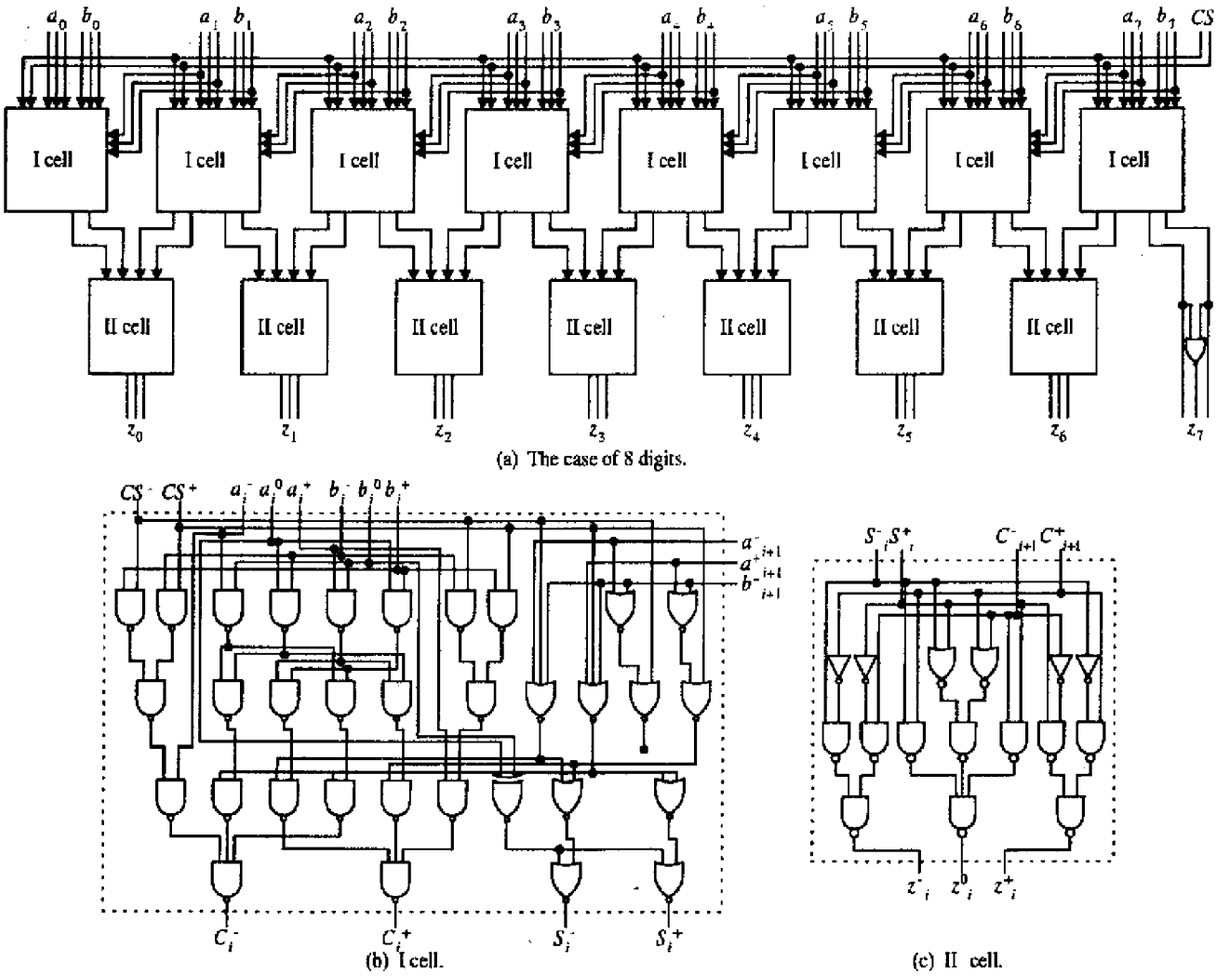

Fig. 4 Structure of high-speed redundant binary adder-subtractor.

による多段化を行わず一度式展開し，入力a_iに対して多段化を施す．

$$C_i^- = \overline{\overline{a_i^- \cdot b_i^- \cdot CS^+} \cdot \overline{b_i^+ \cdot CS^-} \cdot \overline{m1_i \cdot \overline{nn_i} \cdot CS^+} \cdot \overline{m2_i \cdot ns_i \cdot CS^-}} \quad (5)$$

$$C_i^+ = \overline{\overline{a_i^+ \cdot b_i^+ \cdot CS^+} \cdot \overline{b_i^- \cdot CS^-} \cdot \overline{p1_i \cdot nn_i \cdot CS^+} \cdot \overline{p2_i \cdot \overline{ns_i} \cdot CS^-}} \quad (6)$$

このような操作を行うことによって，加算時および減算時における中間桁上げの出力が制御信号を待たずに並列に実行することが可能となる．

次に，中間和S_iに対しては，遅延時間が大きくなる原因のもとが$nn_i \cdot CS^+$，$\overline{nn_i} \cdot CS^+$，$ns_i \cdot CS^-$，$\overline{ns_i} \cdot CS^-$の4つの項，すなわち，制御信号と下位桁の条件を組み合わせた項であるため，これらをそれぞれ新たな条件式pnn_i，mnn_i，pns_i，mns_iとし，以下のように導出する．

$$pnn_i = nn_i \cdot CS^+ \quad (7)$$
$$= \overline{a_{i+1}^- + b_{i+1}^- + \overline{CS^+}} \quad (8)$$
$$= \overline{a_{i+1}^- + b_{i+1}^- + CS^-} \quad (9)$$
$$mnn_i = \overline{nn_i} \cdot CS^+ \quad (10)$$
$$= \overline{nn_i + CS^-} \quad (11)$$
$$pns_i = ns_i \cdot CS^- \quad (12)$$
$$= \overline{a_{i+1}^+ + b_{i+1}^- + CS^+} \quad (13)$$
$$mns_i = \overline{ns_i} \cdot CS^- \quad (14)$$
$$= \overline{ns_i + CS^+} \quad (15)$$

ここで，pnn_iの式において，式(8)でCS^+を用いていたのに対し式(9)ではCS^-を用いているのは，$\overline{CS^+} = CS^-$の関係を利用したものである．したがって，この制御信号の関係を利用することにより，論理の多段化を減少させ，高速化を図る．

以上の検討結果から得られた本加減算器の出力論理式は以下のようになる．また，以下の論理式に基づいた本加減算器の構成を図4に示す．

$$C_i^- = \overline{\overline{a_i^- \cdot b_i^- \cdot CS^+} \cdot \overline{b_i^+ \cdot CS^-} \cdot \overline{m1_i \cdot mnn_i} \cdot \overline{m2_i \cdot pns_i}} \quad (16)$$

Table 3 Computation rules for first step in addition used in conventional square rooting circuit.

(a) c_i, s_i

a_i	b_i	t_{i+1}	c_i	s_i
$\bar{1}$	$\bar{1}$	0	0	0
0	0	1	0	$\bar{1}$
$\bar{1}$	0	0	0	0
0	1	$\bar{1}$	$\bar{1}$	$\bar{1}$
$\bar{1}$	1	0	1	0
1	1	$\bar{1}$	$\bar{1}$	$\bar{1}$

(b) t_i

		a_i		
		$\bar{1}$	0	1
	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$
b_i	0	$\bar{1}$	0	0
	1	$\bar{1}$	0	0

$$C_i^+ = \overline{\overline{a_i^+ \cdot b_i^+ \cdot CS^+ \cdot b_i^- \cdot CS^-} \cdot \overline{p1_i \cdot pnn_i \cdot p\bar{2}_i \cdot mns_i}} \quad (17)$$

$$S_i^- = \overline{\overline{a_i^0 \oplus b_i^0} + \overline{pnn_i + mns_i}} \quad (18)$$

$$S_i^+ = \overline{\overline{a_i^0 \oplus b_i^0} + \overline{mnn_i + pns_i}} \quad (19)$$

4. 従来の冗長2進表現を用いた開平器

これまでに冗長2進表現を用いた開平器の構成法が提案されている[5]．これは，冗長2進表現の特長をいかした開平用アルゴリズムを用いて高速化を図っている．本章では，最初に従来の開平器で用いた冗長2進加減算器の構成について述べる．次に，開平アルゴリズムについて述べ，最後に，そのアルゴリズムに基づいた開平器の構成について述べる．

4.1 従来の開平器で用いた冗長2進加減算器

従来の開平器で用いた冗長2進加算器では，本加減算器で用いた加算方法と同様2つのステップを用いるが，ステップ1で算出する中間桁上げおよび中間和を求める計算規則が異なる．その計算規則を表3に示す．この計算規則の導出方法は，$2c_i + s_i = a_i + b_i - 2t_i + t_{i+1}$の関係を満足するように決めたものである．

表3(a)の計算規則を見てわかるように，中間桁上げc_iは$\{0, 1\}$，そして，中間和s_iは$\{\bar{1}, 0\}$とそれぞれ2つの要素しか取らないことがわかる．よって，この計算規則を用いて加算器を構成する場合，c_iおよびs_iは1ビットで十分表現できることがわかる．ゆえに，この計算規則を用いた加算器では，ハードウェア量を抑えることが可能となる．しかし，演算時間に関しては従来の1桁2ビット表現による加算器とほとんど変わらない．したがって，この加算器を開平器に適用する場合，さらに符号変換器も必要となることから，処理時間が大きく増加することが容易に推測できる．なお，この計算規則を用いた場合のIセルの構成を図5(a)に，IIセルの構成を図5(b)に示す．

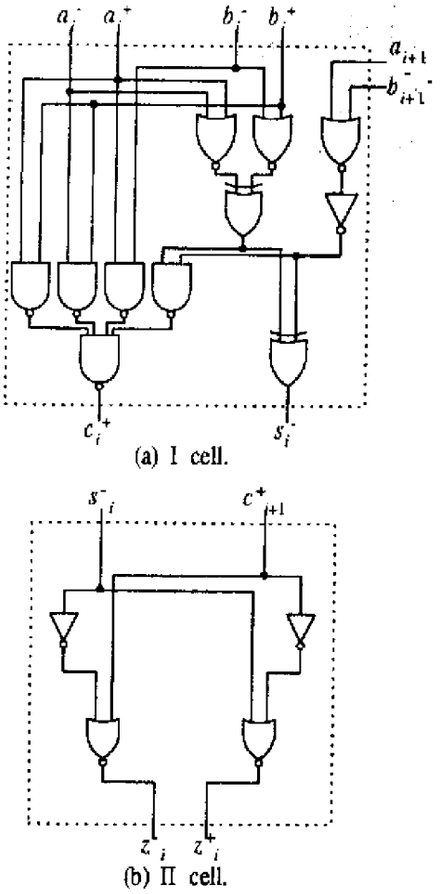

Fig. 5 Structure of redundant binary adder-subtractor used in conventional square rooting circuit.

4.2 開平法アルゴリズム

本節では，本報告で用いた開平アルゴリズムについて簡単に述べる[5]．変数を以下のように定義し，次に開平アルゴリズムを示す．

X : 入力の演算数 ($\frac{1}{4} \leq X < 1$)
Q : 出力の平方根 ($\frac{1}{2} \leq Q < 1$)
q_i : 平方根の小数点i桁目
Q_i : 平方根の小数点i桁目までの値
(すなわち, $[0.q_1q_2\cdots q_i]$)
$R^{(i)}$: i回目に得られる部分剰余
r_j : $R^{(i-1)}$のj桁目
N : 桁数

begin
<ステップ1>
$R^{(0)} \leftarrow X$ (冗長2進数に変換)
$Q_0 := 0$
<ステップ2>
$q_1 := 1$
$R^{(1)} := R^{(0)} - 2^{-2}$ (冗長2進数体系で計算)
$Q_1 := Q_0 + 2^{-1}$ ($Q_i := [0.1]_{SD2}$)
<ステップ3>
for $i := 2$ to N do
begin

$$q_i := \begin{cases} \bar{1} & \text{if } [r_0.r_1r_2]_{SD2} < 0 \\ 0 & \text{if } [r_0.r_1r_2]_{SD2} = 0 \\ 1 & \text{if } [r_0.r_1r_2]_{SD2} > 0 \end{cases}$$

$R^{(i)} := R^{(i-1)} - 2^{-i}q_i(2Q_{i-1} + 2^{-i}q_i)$
(冗長2進数体系で計算)
$Q_i := Q_{i-1} + 2^{-i}q_i$
($Q_i := [0.q_1q_2\cdots q_i]$)
end
end

4.3 開平器の構成

従来から提案されている開平用ハードウェアアルゴリズムを基に構成した開平器の構成は図6のようになる．なお，この図は演算対象となる数Xが符号ビットを含む純進数小数11ビット，そして，出力となる平方根Yが6桁の場合である．この開平器では，RセルとQセルの2つのタイプのセルで構成される．

Rセルは，部分剰余の各桁の生成を行うセルである(図7(a))．このセルでは，第4.1節で述べたIセルおよびIIセル，および，減数の符号変換を行う符号変換器を用いる(図7(a)の細線に囲まれた部分)．

Qセルは，部分剰余の上位桁の生成および平方根の各桁を求めるセルである．このセルでは，前

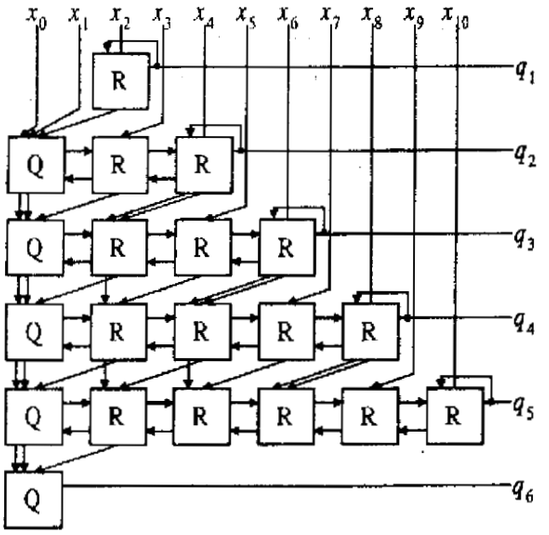

Fig. 6 Structure of square rooting circuit using redundant binary representation.

段の部分剰余の上位3桁から平方根の1桁を求め，さらに，部分剰余の上位桁用の計算規則に基づいて上位桁の部分剰余を生成する．上位桁の演算に専用の計算規則を用いる理由は，Rセルを用いて上位桁の部分剰余を生成すると，冗長2進数は冗長性を持っているために最上位桁が0とはならない場合が生じる．したがって，この状態でシフトを行うと，最上位桁の部分が切り捨てられて正しい部分剰余が得られなくなる．そこで，この演算部分を表を用いて生成することにより，演算結果の桁あふれを防ぐ．ここで，表4に示すように，従来の開平器では中間桁上げを$\{1,0\}$，そして，中間和を$\{0,\bar{1}\}$とする特殊な冗長2進加算器を用いているために上位桁専用の計算規則が複雑となる．

5. 本加減算器を適用した高速冗長2進開平器

5.1 本加減算器を適用した開平器の構成法

冗長2進表現を利用した開平器に本加減算器を適用するうえにおいて，図6に示す基本アルゴリズムは変わらないものの，その構成法自体は大きく異なる．それは，従来の開平器と本開平器では冗長2進数の表現方法および加減算器の構成法が全く異なるからである．本節では，本加減算器を適用した場合の具体的な構成法について述べる．

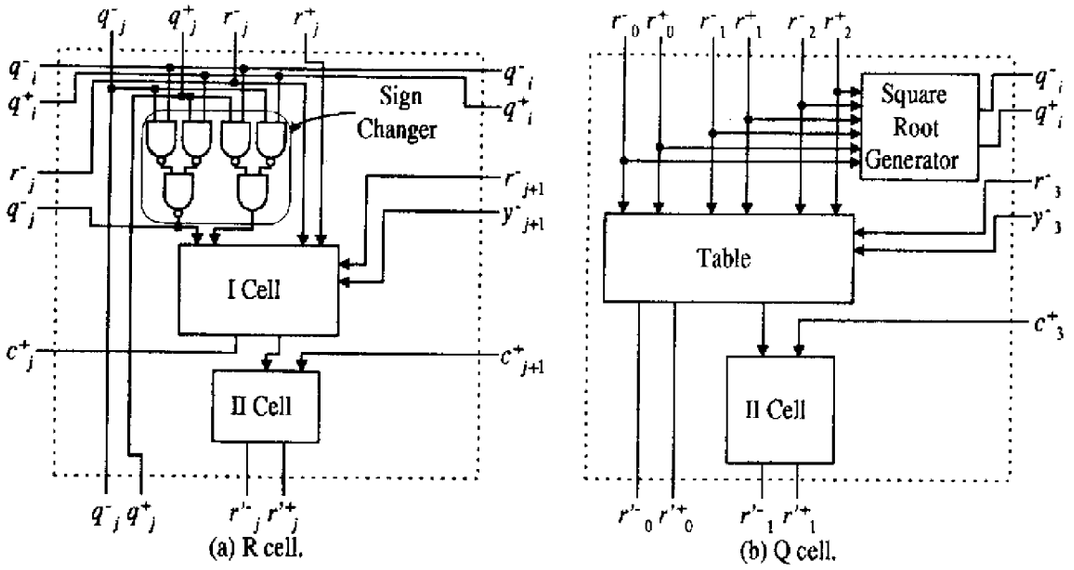

Fig. 7 Structure of each cell used in conventional square rooting circuit.

Table 4 Computation rule for Q cell of conventional square rooting circuit.

r_0	r_1	r_2	t_3	r'_0	s_1
1	0	0	—	1	0
1	0	$\bar{1}$	0	1	0
1	$\bar{1}$	1			
0	1	1	$\bar{1}$	1	$\bar{1}$
1	$\bar{1}$	0	0	1	$\bar{1}$
0	1	0	$\bar{1}$	0	0
1	$\bar{1}$	$\bar{1}$			
0	1	$\bar{1}$	0	0	0
0	0	1			
0	0	0			
0	0	$\bar{1}$			
0	$\bar{1}$	1	$\bar{1}$	0	$\bar{1}$
$\bar{1}$	1	1			
0	$\bar{1}$	0	0	0	$\bar{1}$
$\bar{1}$	1	0	$\bar{1}$	$\bar{1}$	0
0	$\bar{1}$	$\bar{1}$	0	$\bar{1}$	0
$\bar{1}$	1	$\bar{1}$			
$\bar{1}$	0	1	$\bar{1}$	$\bar{1}$	$\bar{1}$
$\bar{1}$	0	0	—	$\bar{1}$	$\bar{1}$

最初に，部分剰余の各桁の演算を行うRセルの構成法について述べる．本報告で用いる開平アルゴリズムは，部分剰余の生成において加算および減算のほかにどちらの演算も行わずにシフトのみ行うという操作が必要とされる．それに対し，本加減算器では加算または減算のどちらか一方の演算を必ず行う構成法を用いている．そのため，制御信号によって演算結果または被加数または被減数のどちらかを選択するのに，本加減算器をそのまま適用すると図8に示すようなセレクタが必要とされる．しかし，そのセレクタの部分が従来の減数を加数に変換する符号変換器と遅延時間が変わらないため，図8に示す構成法では本加減算器の高速性をいかせない．そこで，われわれはセレクタの機能をIIセルに含ませる．そのIIセルの構成を図9に示す．この図を見てわかるように，これまでの加算器で用いてきたIIセルの構成と比べると，演算を行う前の被加数または被減数の入力，および，それを選択するかどうかを決定する制御信号が付加されていることがわかる．このIIセルの構成法を用いることにより，加減算を行うかまたは行わないかの制御をIIセルの演算と同時に行うことが可能となる．したがって，本加減算器の高速

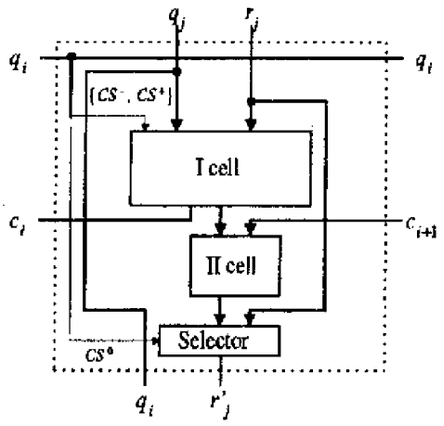

Fig. 8 Structure of R cell using selector.

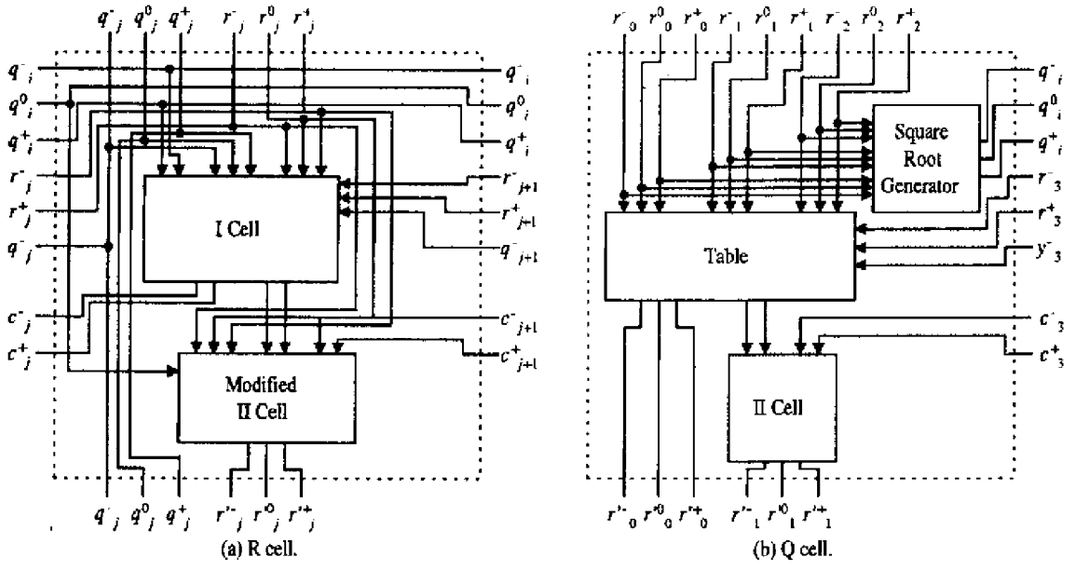

Fig. 10 Structure of each cell used in proposed square rooting circuit.

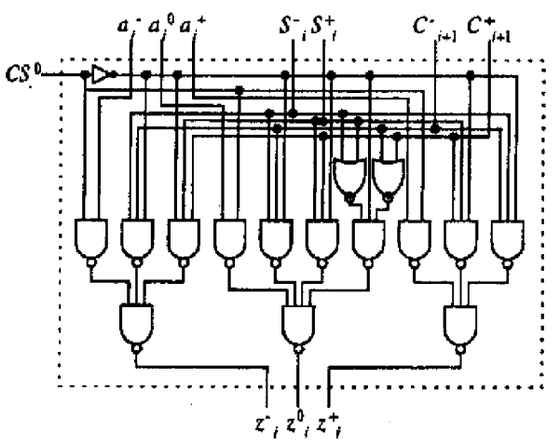

Fig. 9 Structure of modified II cell.

性を失われずに，われわれがこれまで提案してきた1桁2ビット/3ビット表現を用いた高速加算器と全く同等の処理時間で演算することが可能となる．なお，このIIセルを用いた場合のRセルの構成を図10(a)に示す．

次に，Qセルの構成法について述べる．本開平器も従来と同様に部分剰余の上位桁を求める計算規則を用いる．ただし，従来の開平器で用いた冗長2進数の加算規則は本加減算器で用いたものとは異なるため，新たに計算規則を作成する．その計算規則を表5に示す．表4と表5を比較するとわかるように，従来の開平器で用いた計算規則の方が部分剰余を出力する際，下位桁の条件が必要となる部分が多いことに注目していただきたい．これは，前節でも述べたように，従来の開平器で用いた加算器がハードウェア量の効率化を目的としたものであるため，特別な加算規則を用いているからである．したがって，本開平器で用いる上位桁生成用の表の方がシンプルなものとなる．また，部分剰余の上位3桁を検出する部分においても，本構成法では1桁3ビット表現を用いていることから各桁が0であることを容易に検出できるため，高速に演算を行うことが可能となる．

Table 5 Computation rule for Q cell of proposed square rooting circuit.

r_0	r_1	r_2	nn_2, ns_2	r_0'	s_1
1	0	0	—	1	1
1	0	$\bar{1}$			
1	$\bar{1}$	1	—	1	0
0	1	1			
1	$\bar{1}$	0	$ns_2 = 1$	0	1
0	1	0	$ns_2 = 0$	$\bar{1}$	$\bar{1}$
1	$\bar{1}$	$\bar{1}$			
0	1	$\bar{1}$			
0	0	1			
0	0	0	—	0	0
0	0	$\bar{1}$			
0	$\bar{1}$	1			
$\bar{1}$	1	1			
0	$\bar{1}$	0	$nn_2 = 1$	0	$\bar{1}$
$\bar{1}$	1	0	$nn_2 = 0$	$\bar{1}$	1
0	$\bar{1}$	$\bar{1}$			
$\bar{1}$	1	$\bar{1}$	—	$\bar{1}$	0
$\bar{1}$	0	1			
$\bar{1}$	0	0	—	$\bar{1}$	$\bar{1}$

Table 6 VLSI evaluation of redundant binary square rooting circuits.

(a) Power. (mW/MHz)

Digit width	Conventional square rooting circuit	Proposed square rooting circuit
4	0.11	0.10
8	0.51	0.77
16	1.96	4.01
32	7.83	17.33

(b) Area. (mm^2)

Digit width	Conventional square rooting circuit	Proposed square rooting circuit
4	0.02	0.02
8	0.10	0.14
16	0.40	0.69
32	1.56	3.02

(c) Gates. (gates)

Digit width	Conventional square rooting circuit	Proposed square rooting circuit
4	181	161
8	839	1238
16	3233	6152
32	12535	26607

(d) Delay time. (ns)

Digit width	Conventional square rooting circuit	Proposed square rooting circuit	Ratio of speed
4	44	28	1.71
8	141	86	1.65
16	331	207	1.60
32	723	447	1.62

5.2 性能評価

本節では，前節で述べた開平器をPARTHENONによってVLSI設計および評価を行った[6]．評価対象を消費電力，面積，ゲート数，最大遅延時間とし，桁数を8桁から64桁まで変えて評価した．その結果を表6に示す．なお，実部品として用いたセル・ライブラリの設計ルールは0.6μmCMOSスタンダードセル(VLSIテクノロジ社)であり，電源電圧は5.0[V]とした．また，表6(c)のゲート数とは2入力NAND換算値であり，表6(d)の高速化とは本開平器に対する従来の構成の演算速度の比率を表す．

この評価からわかるように，本開平器は従来の開平器と比べ約1.6倍の高速化が得られていることがわかる．これは，本開平器で用いた本加減算器と従来の加減算器のほぼ同等な結果が得られている．このことから，本提案の加減算器の有用性が明らかである．

6. おわりに

本報告では，われわれがすでに提案してきた高速冗長2進加減算器の応用として，冗長2進表現を利用した開平アルゴリズムに基づく開平器に適用し，その高速化に関する考察を行った．最初に，従来の冗長2進表現を利用した開平アルゴリズムに基づいた構成について述べた．そして，開平器に本加減算器を適用する上での検討を行い，その結果による構成法を示した．最後に，VLSI設計システムPARTHENONを用いて本加減算器を適用した開平器をVLSI設計および評価を行った．その結果，本加減算器を適用した高速開平器は従来の開平器に対し本加減算器の高速化と同等の約1.6倍の高速化が得られることを示した．これより，開平器に対する本加減算器の有効性が明らかになった．

参考文献

1) 齋藤正人，日野杉充希，恒川佳隆，三浦守: 1桁2ビット／3ビット混合表現を用いた高速冗長2進加減算器の構成法，信学技報，CAS99-41，99-105，83/90 (1999)

2) 恒川佳隆，日野杉充希，齋藤正人，虻川勝己，三浦守: 1桁2ビット／3ビット混合型高性能冗長2進加算器とその乗算器への応用，電気学会論文誌，119-C-5，644/653 (1999)

3) A. Avizienis: Signed-Digit Number Representations for Fast Parallel Arithmetic, IEEE Trans. Elec. Comput., EC-10-9, 389/400 (1961).

4) N. Takagi, H. Yasuura and S. Yajima: High-Speed VLSI Multiplication Algorithm with a Redundant Binary Addition Tree, IEEE Trans. Comput., C-34-9, 789/796 (1985).

5) 高木直史，矢島脩三: 冗長2進表現を利用した開平用ハードウェアアルゴリズム，電子通信学会論文誌，J69-D-1，1/10 (1986)

6) NTTデータ通信株式会社: PARTHENON User's Manual (1990)