Redundant Binary Booth Recoding*

Chung Nan Lyu & David W. Matula

Dept. of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275

Abstract

We investigate the efficiencies attainable pursuing Booth recoding directly from redundant binary input with limited carry propagation. As a digit conversion problem we extend the important result that each radix 4 Booth recoded digit can be determined from 5 consecutive input signed bits to obtain that each radix 2^k Booth recoded digit can be determined from $2k+1$ consecutive input signed bits and prove this to be the minimum possible for any $k \geq 2$. Analysis of alternative bit pair encodings of signed bits yields the improved result that each radix 2^k Booth recoded digit can be determined from only $2k$ encoded bit pairs employing sign and magnitude bit encoding, a result which does not extend to conventional borrow-save or carry-save redundant binary digit encodings.

Radices 4 and 8 gate level designs are illustrated for alternative encodings, with our signed bit design shown to yield smaller depth and fewer gates than existing redundant binary Booth recoding circuits from the literature.

I. Introduction and Summary

Multiplier recoding by various extensions of the original Booth recoding have been developed over some 40 years in the literature [2, 8, 10, 12, 13, 14, 15]. Multipliers employing Booth recoding from conventional binary to radices 4 and 8 are common in implementations, and even higher radices have been argued to be competitive [13].

Numerous current floating-point unit designs incorporating an expensive fast multiplier make iterative use of the multiplier for implementing fast algorithms for division, square root, and/or transcendental function computation by extended polynomial approximation [3, 5, 6, 12, 14, 15]. When multipliers are to be used iteratively in hardware or under microcode control, it is advantageous for the multiplier to accept redundant binary coded input for at least one of the arguments. This similarly pertains to the direct input of redundant binary seed reciprocals as described in our companion paper in this proceedings [7]. Direct use of redundant binary input avoids the carry completion delay attendant to intermediate conversion to conventional binary representation for each of an iterative chain of dependent multiply or multiply-add operations. Fast multipliers where the multiplicand (hence the input partial products to be summed) is redundant can be found in [3, 6]. Implementations where the input to the multiplier recoding unit is redundant are described in [5, 8, 12, 14].

Radix 4 Booth recoding directly from redundant binary input has been investigated in [5, 8, 12, 14, 15], but higher radices appear not to have been pursued in the literature. It is useful to recognize that redundant binary Booth recoding is essentially a digit set conversion from maximally redundant to balanced minimally redundant digit sets. Digit set conversions have been investigated more generally in [1, 9, 11]. Our purpose here is to explore the efficiencies attainable for redundant binary Booth recoding for any higher radix 2^k, $k \geq 2$, with particular regard to minimizing the number of encoded input bits of the redundant binary input on which each output Booth recoded digit is dependent.

In Section II we investigate Booth recoding as a problem in digit set conversion. It has been shown for both signed bit and carry-save redundant binary input [8, 12, 14, 15] that each radix 4 Booth recoded digit can be determined from just 5 redundant binary

*Supported in part by grants from Cyrix Corporation and the Texas Advanced Technology Program Grant 003613013.

input digits. Our principal result in Section II is to extend this result proving for any $k \geq 2$, that each radix 2^k Booth recoded digit can be determined as a function of at most $2k+1$ consecutive redundant binary digits (with signed bit or carry-save input), and furthermore that this is the minimum possible in that an output digit can depend on no less than $2k+1$ signed bits.

In Section III, multiplier recoding is then considered at the more atomic level of the conversion from the encoded redundant binary digits to the Booth recoded output format. Note that the digit set $\{\bar{1}, 0, 1\}$ has two popular alternative encodings, sign and magnitude (sign-mag) and borrow-save. For sign-mag, each digit has a sign bit and a magnitude bit, and for borrow-save, each digit has a positive (sum) bit and a negative (borrow) bit as shown in Fig. 1. Our principal result in Section III is that the $2n$ bits of an n signed bit encoding may be partitioned into $\lfloor n/k \rfloor+1$ disjoint $2k$ bit windows such that each $2k$ bit window generates a unique "residue digit" in the range $\{-2^{k-1}, -2^{k-1}+1, \ldots, 2^{k-1}-1\}$ and a carry bit in $\{0, 1\}$ where the sum of residue and next lower carry is then in the desired minimally redundant (Booth) digit set $\{-2^{k-1}, -2^{k-1}+1, \ldots, 2^{k-1}\}$. It is further noted that this result is a particular advantage of sign-mag encoding as neither borrow-save nor carry-save encodings yield such reduced dependence.

In Section IV we employ the results of Section III as a foundation for efficient logic design of redundant binary Booth recoding circuits. The designs are illustrated at the gate level employing negative logic and exclusive-or (nor) gates which are preferred for VLSI implementation. Radix 4 Booth output recoding has a four gate delay and octal Booth output is obtained with a five gate delay. Comparison with other designs from the literature is included [8, 12].

Sign-mag encoding				
Sign bits:	0	0	1	1
Mag. bits:	0	1	1	0
Digit values	0	1	-1	X
Borrow-save encoding				
Borrow bits:	0	1	0	1
Sum bits:	0	1	1	0
Digit values:	0	0	1	-1
Carry-save encoding				
Carry bits:	0	0	1	1
Sum bits:	0	1	0	1
Digit values	0	1	1	2

Fig. 1 Redundant binary encodings

II. Redundant Digit Set Conversions

The digit values $\{d \mid -2^k+1 \leq d \leq 2^k-1\}$ comprise the *maximally redundant digit set radix* 2^k. A digit $d \in \{\bar{1}, 0, 1\}$ is termed a *signed bit* and a signed bit sting $\bar{1}011100\bar{1}110\bar{1}$ denotes a signed bit number. Conversion from signed bit to a maximally redundant higher radix digit set is simply accomplished by partitioning the signed bit number into k-digit windows and evaluating each window as a higher radix digit. For example, $\bar{1}01\ 110\ 0\bar{1}1\ 10\bar{1}$ becomes $\bar{3}6\bar{1}3$ in octal. Booth recoding employs conversion to the smaller digit set $\{d \mid -2^{k-1} \leq d \leq 2^{k-1}\}$ comprising the *balanced minimally redundant (Booth) digit set radix* 2^k, e. g. for our preceding example $\bar{3}6\bar{1}3_8 = \bar{2}\,\bar{2}\,\bar{1}3_8$. Conversion from signed bit to a redundant radix 2^k digit set is termed *digit parallel* if each converted digit is a function of some constant number of consecutive signed bits independent of the length of the number. Thus digit parallel conversion to a maximally redundant digit set radix 2^k has each output digit being a function of just k consecutive input signed bits, which is clearly minimum. Digit parallel conversion from signed bit to Booth radix 4 has been shown [5, 8, 12, 14, 15] to be attainable with each Booth recoded digit a function of just 5 signed bits. The following extends this result to higher radices and confirms the minimum number of signed bits needed to determine a Booth recoded output digit.

Theorem 1: Digit parallel conversion from signed bit to minimally redundant (Booth) radix 2^k digit representation for any $k \geq 2$ can be obtained with each radix 2^k recoded digit a function of at most $2k+1$ consecutive signed bits and $2k+1$ is the minimum possible.

Proof: Partition an n signed bit number into $\lfloor n/k \rfloor+1$ disjoint k consecutive signed bit strings (windows) starting up from the lowest signed bit, where $k \geq 2$. For any given k-digit string let $-2^k+1 \leq d \leq 2^k-1$ be the value of that string and let $-1 \leq b \leq 1$ be the value of the leading signed bit of the adjacent lower string. Determine a residue digit value $-2^{k-1} \leq r \leq 2^{k-1}$ and signed carry bit $-1 \leq c \leq 1$ such that $-2^k+1 \leq d = c2^k+r \leq 2^k-1$ according to Table 1.

Now for each k-digit string, add in the carry from the adjacent lower string to the residue digit for that string to obtain the converted digit d'. Note that the leading string of the $\lfloor n/k \rfloor+1$ disjoint k-signed bit strings has a leading zero and generates a carry of zero by Table 1 so the resulting converted radix 2^k

Maximally redundant digit range	$2^{k-1}+1 \le d$	$-2^{k-1}+1 \le d \le 2^{k-1}-1$	$d \le -2^{k-1}-1$	$\lvert d \rvert = 2^{k-1}$ $db \le 0$	$\lvert d \rvert = 2^{k-1}$ $db > 0$
Residue digit	$d-2^k$	d	$d+2^k$	d	$d-2^k b$
Carry digit	1	0	-1	0	b

Table 1 Residue and carry digit generation for conversion from maximally redundant digits to balanced minimally redundant (Booth) digits radix 2^k

Extended digit value	$0 \le d \le 2^{k-1}-2$	$2^{k-1} \le d \le 3 \times 2^{k-1}-2$	$3 \times 2^{k-1} \le d$	$d = 2^{k-1}-1$ $b = 2$	$d = 2^{k-1}-1$ $b = 0,1$	$d = 3 \times 2^{k-1}-1$ $b = 2$	$d = 3 \times 2^{k-1}-1$ $b = 0,1$
Residue digit	d	$d-2^k$	$d-2^{k+1}$	$-2^{k-1}-1$	d	$-2^{k-1}-1$	$2^{k-1}-1$
Carry digit	0	1	2	1	0	2	1

Table 2 Residue and carry digit generation for conversion from k-extended-bit string values to balanced minimally redundant (Booth) digits radix 2^k

number has $\lfloor n/k \rfloor+1$ digits. It remains to show that $-2^{k-1} \le d' \le 2^{k-1}$ holds for each converted digit after the carry absorbing addition. Note that if originally $|d| \ne 2^{k-1}$, then the resulting residue digit of Table 1 is in $\{-2^{k-1}+1, -2^{k-1}+2, \ldots, 2^{k-1}-1\}$, and so adding in a carry from $\{-1, 0, 1\}$ yields $-2^{k-1} \le d' \le -2^{k-1}$. If $|d| = 2^{k-1}$, then our choice of residue digit is opposite in sign to the sign of any non-zero carry digit that could be generated in the adjacent lower order string, so d' in this case satisfies $|d'| = 2^{k-1}$ or $|d'| = 2^{k-1}-1$. Hence a balanced minimally redundant (Booth) digit radix 2^k number is obtained where each output digit depends on its input k-signed bit string generating the residue digit and the next lower (k+1) signed bits determining the adjacent lower string carry, or in total at most 2k+1 consecutive input signed bits.

To show the necessity for dependence of an output digit on at least 2k+1 consecutive input signed bits, we show a particular case for octal conversion (k = 3). Examples for all other values of k ≥ 2 can be similarly generated. Consider that the three following signed bit numbers, x, y, and z have unique balanced minimally redundant octal representations as shown:

$$x = 110\ \bar{1}00\ 111 = 1\bar{2}\bar{3}\bar{1}_8$$
$$y = 010\ \bar{1}00\ 111 = 2\bar{3}\bar{1}_8$$
$$z = 010\ \bar{1}00\ \bar{1}\bar{1}\bar{1} = 131_8$$

Consider the determination of the third lowest octal digit d from a string $b_8b_7b_6b_5b_4b_3b_2b_1b_0$ of consecutive signed bit positions of x, y, and z. This digit must be $d = \bar{2}$ for x, d = 2 for y, and d = 1 for z. Since x and y differ only in the leading signed bit position b_8, determination of d must include this position. Since y and z are the same for the six positions $b_8b_7b_6b_5b_4b_3$, determination of d must include inspection of at least some signed bit of $b_2b_1b_0$. Hence the consecutive signed bit sub-string $b_8b_7 \ldots b_2$ must at least be employed to determine d. Hence 2k+1 is the minimum number of consecutive signed bits for which digit parallel conversion is possible. ∎

To visualize the use of Table 1, we can convert the 9 signed bit string for x from the proof of Theorem 1, employing evaluation first to maximally redundant octal digits.

Example 1:

	110	$\bar{1}00$	111	signed bit
	6	$\bar{4}$	7	max redundant octal
	$\bar{2}$	$\bar{4}$	$\bar{1}$	residue digit
1	0	1		carry digit
1	$\bar{2}$	$\bar{3}$	$\bar{1}$	carry absorbing sum yielding Booth recoded octal

An alternative redundant binary digit set is the *extended bit* set {0, 1, 2} corresponding to carry-save encoded digit values. A result similar to Theorem 1 extending the radix 4 result from [9, 15] can be obtained using the conversion residue and carry digits shown in Table 2, where the input of k-extended-bit strings are denoted as values of d in the range $\{0, 1, \ldots, 2(2^k-1)\}$, and where b denotes the leading extended bit in the next lower order string.

Use of Table 2 is illustrated in the following.

Example 2:

	211	020	212	extended bit
	B	4	C	extended octal (in hexadecimal)
	3	$\bar{4}$	$\bar{4}$	residue digit
1	1	2		carry digit
1	4	$\bar{2}$	$\bar{4}$	carry absorbing sum yielding Booth recoded octal

III. Converting Redundant Binary Encodings to Booth Encodings

For purposes of implementing redundant binary to higher radix Booth 2^k multiplier recoding, it is relevant to consider the detailed encodings of the redundant binary inputs and of the desired Booth recoder output. Encoding of the signed bits {-1, 0, 1} requires two bits for each digit, and may be realized in either borrow-save or sign-mag encoding formats. Another popular redundant binary encoding is carry-save which represents the redundant extended set {0, 1, 2}. These encodings were given in Fig. 1. Each digit of the representations is encoded by two bits. Sign-mag has a unique encoding for each digit 0, 1 and $\bar{1}$. For carry-save and borrow-save, the bit encodings introduce a second level of redundancy – that is, redundant encodings of the same digit value from a redundant set of digit values.

Booth encoding is comprised of a sign bit and a set of magnitude select bits, one for each possible digit magnitude. Thus for radix 4, the Booth encoding is a sign bit and 3 magnitude select bits corresponding to selecting magnitude 0, 1, 2. Radix 8 Booth encoding has a sign bit and 5 magnitude select bits for the values of 0, 1, 2, 3, and 4. The total number of input bits and output bits has a direct relation on the number of input and output terminals of a Booth recoder.

Observation 1: Booth recoding of an n digit sign-mag, borrow-save, or carry-save encoded input to a Booth encoded output has an input size of 2n bits in any of these encodings and an output size for radix 4 of either 2n+2 or 2n+4 bits, and an output size for radix 8 between 2n+2 and 2n+6 bits. ▮

Implementation of a digit level Booth recoder radix 4 from redundant binary input suggests by Theorem 1 that overlapping windows of 5 redundant binary digits or a total of 10 encoded bits are required. Closer inspection of the alternative sign-mag, borrow-save, and carry-save encoding reveals a further simplification particular to the signed bit encoding.

Observation 2: Writing the sign bit and magnitude bit pairs for sign-mag as two bit numbers, we have 00 = 0, 01 = 1, 11 = -1, 10 = undefined. For the three defined pairs, it is possible to interpret these bit pairs as 2 bit 2's complement representations, which allows that the sign bit may be given a weight of -2 and the magnitude bit a weight of 1. ▮

Observation 2 allows that in forming k-signed-bit windows for interpretations of signed bit as radix 2^k digit values, we may simply skew our windows including the sign bit of a leading encoded digit of each k digit window with the next higher window, obtaining conversion to the "semi-max redundant" digit set $\{-2^{k-1}, -2^{k-1}+1, ..., 2^k-1\}$ at essentially the same cost as conversion to a maximally redundant digit set. For radix 8, the skewed windows yield digits in $\{\bar{4}, \bar{3}, ..., 6, 7\}$ as illustrated in Fig. 2.

A digit d of the proposed semi-max digit set $\{-2^{k-1}, -2^{k-1}+1, ..., 2^k-1\}$ may be written in carry bit c and residue r form $d = c2^k+r$ with c=1 for $d \geq 2^{k-1}$ and c = 0 otherwise, yielding the residue digit $r \in \{-2^{k-1}, -2^{k-1}+1, ..., 2^{k-1}-1\}$. We thus obtain unique residue and carry digits independently from each window as shown further in Fig. 2, where the carry absorbing sum then simply yields our desired Booth octal digits. Generalization of the preceding argument provides the proof of the following theorem, the details of which are omitted for brevity.

Theorem 2: The 2n bit sign-mag encoding of an n-signed bit number can be partitioned into $\lfloor n/k \rfloor+1$ disjoint 2k bit windows where each window determines a unique residue digit $r \in \{-2^{k-1}, -2^{k-1}+1, ..., 2^{k-1}-1\}$ and single carry bit $c \in \{0, 1\}$ such that the carry absorbing sum of each residue digit and the carry from the next lower window yields an output sting of $\lfloor n/k \rfloor+1$ Booth radix 2^k digits.

Corollary: It is possible to convert the 2n bits comprising an n signed bit encoded number to an $\lfloor n/k \rfloor+1$ Booth radix 2^k number where each output digit depends on at most 4k of the 2n input bits for any $k \geq 2$. In particular on 8 encoded input bits for Booth recoded radix 4 output and 12 encoded input bits for Booth recoded octal output.

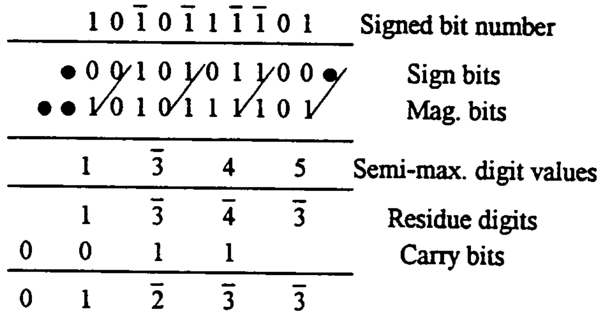


Fig. 2 Radix 8 Booth recoding with skewed windows

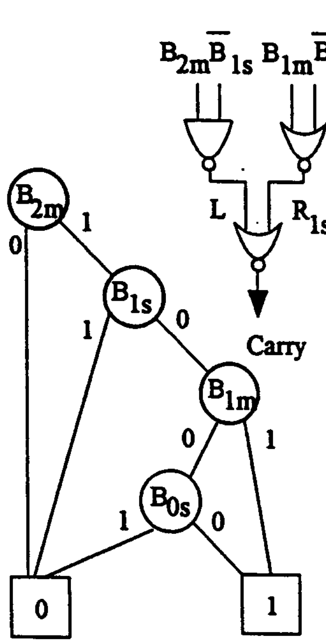


Fig. 3 An OBDD for radix 4 carry generation

Window i		Window (i-1)	Booth recoder outputs				
Value	$R_{2s}R_{2m}R_{1s}R_{1m}$	Carry	Sign	M_2	M_1	M_0	P.P.
-3	1111	0	x	x	x	x	-
		1	x	x	x	x	-
-2	1100	0	1	1	0	0	-2X
		1	1	0	1	0	-X
-1	1101	0	1	0	1	0	-X
		1	x	0	0	1	0X
	0011	0	1	0	1	0	-X
		1	x	0	0	1	0X
0	0000	0	x	0	0	1	0X
		1	0	0	1	0	+X
1	0111	0	0	0	1	0	+X
		1	0	1	0	0	2X
	0001	0	0	0	1	0	+X
		1	0	1	0	0	2X
2	0100	0	1	1	0	0	-2X
		1	1	0	1	0	-X
3	0101	0	1	0	1	0	-X
		1	x	0	0	1	0X

Table 3 Truth table of a signed bit radix 4 Booth recoder

The results of Theorem 2 and the corollary do not extend similarly to either conventional borrow-save or carry-save encodings. It still does not extend even if a unique form of carry or borrow-save is employed, such as by excluding the 11 pair as an alternative encoding (see [12]). To see the difficulty one may use borrow-save to encode the signed bit digits of x, y, and z from the proof of Theorem 1 along with $w = \bar{1}10\,\bar{1}00\,111 = \bar{2}\,3\,\bar{1}_8$.

IV. Logic Design of Redundant Binary Booth Recoders

Our approach will be to derive an efficient gate level design for transforming sign-mag encoding to Booth encoding for both radix 4 and 8 using distinct preprocessing and core logic sections. Input of borrow-save and carry-save encodings will then be handled by revising only the preprocessing section of the resulting design.

Radix 4 Sign-mag Booth recoder

Implementation of a sign-mag to radix 4 Booth recoder using Theorem 2 has two steps. The preprocessing step starts by aligning the weight of the sign and magnitude bits in a skewed style as shown in Fig. 2. The magnitude bit B_{im} (in *ith* position) and the sign bit $B_{(i-1)s}$ (in (i-1)*th* position) have the same weight and cancel out each other in place if they both are 1's. A new sign-mag number with encoded digits designated $R_{is}R_{im}$, is formed. This new signed bit number has the property that the first non-zero signed bit after a $\bar{1}$ must be 1. There is no substring $\bar{1}00\cdots0\bar{1}$. The possible digit values of a radix 4 window are within [-2, 3] and therefore no carry of $\bar{1}$ to the higher window is needed. A leading bit $L = OR(R_{2s}, \bar{R}_{2m}) = NAND(B_{2m}, \bar{B}_{1s})$ replaces leading R_{is} in each window to simplify the core logic section computation of the carry as apparent from Fig. 3. The core section then accepts $LR_{2m}R_{1s}R_{1m}$ and needs only to generate a single carry bit $\in \{0, 1\}$ to the higher window, along with the residue digit of the current window, and absorb the carry from the lower window to form the final Booth value. The *Ordered Binary Decision Diagram* [1](OBDD) [4] in Fig. 3, where i = 2, explains the logic design of the carry generation process.

Table 3 lists the radix 4 recoding process for the core Booth digit select logic . From this table, several observations can be made. 1) If we make the sign of

[1]A OBDD is directed acyclic graph in which each Boolean variable is represented by a vertex. A outgoing arc from a vertex is labeled as Boolean value 0 or 1 that corresponds to the assigned value of the variable in the vertex. Any Boolean function can be represented as an OBDD. The function equals to 1 if the terminal label equals T, or 0 if it terminates in F. The order of the variables in an OBDD directly affects the efficiency of the function evaluation. Taking Fig. 3 as an example, variable B_{2m} should be inspected first. If it is 0, the function terminates in 0.

M_0 a don't-care, the final sign digit is independent of carry. 2) M_1 is 1 only if the exclusive-or of (R_{1m}, carry) is 1. 3) If M_1 is 0, either M_2 or M_0 is 1. 4) Exclusive-or of (R_{2m}, R_{1s}) if 1 represents a weight of -2 and if 0, 0. Together with R_{1m} and Carry, both having a weight of 1, form the final Booth value. 5) R_{2s} is a don't-care. It is not needed in the process. Fig. 4 shows the gate level implementation. The design has a total gate count of 10 2-input gates and a depth of 4 gates and inverters. The gate count can be further decreased by one if we allow M_0 to be represented by the condition that both M_2 and M_1 are 0's. This is the form of Booth digit selection encoding in [8] and [12]. We have restricted our design to use only negative-logic and exclusive-or (nor) gates which are preferred for CMOS technology.

In Fig. 4 we can observe the carry and residue generation portion of the conversion as comprising six gates of the first two levels generating the three outputs termed Carry, Sign, and R_{1m}. The four gates at levels three and four comprise the carry absorption and magnitude select recoding portion. The following can be verified employing Table 3 along with the carry output of window i (not shown).

Observation 3: Given that the input string $B_{2m}B_{1s}B_{1m}B_{0s}$ has respective weights 2, -2, 1, -1, and digit value $d = 2B_{2m}-2B_{1s}+B_{1m}-B_{0s}$ with encoding guaranteeing $-2 \le d \le 3$, the three outputs Carry, Sign, and R_{1m} available after two logic levels form a unique encoding of d in the mixed radix system with weights 4, -2, and 1. That is $d = 4\text{Carry}-2\text{Sign}+R_{1m}$, with $\text{Sign}R_{1m}$ being the 2's complement representation of the residue digit $-2 \le r \le 1$. ∎

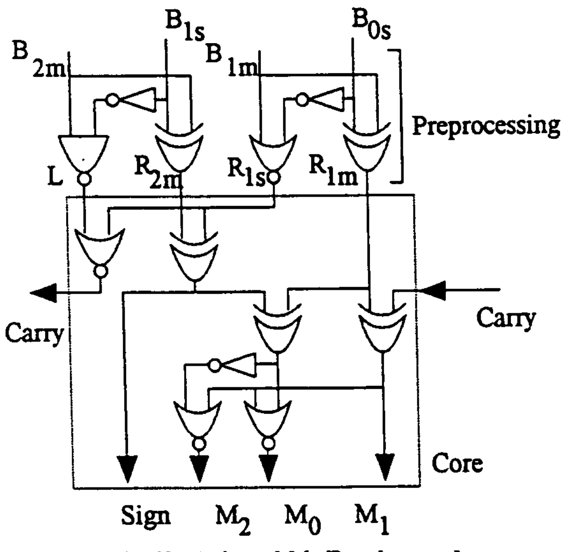


Fig. 4 Radix 4 signed bit Booth recoder

Radix 4 carry-save and borrow-save Booth recoders

Booth recoding from carry-save or borrow-save can be done in a similar way. The only change is an expansion of the preprocessing step. For carry-save encoding, the additional preprocessing step would be pass a 1 to the left bit position when both digits are 1's or pass a 1 to the left bit position and compensate a $\bar{1}$ in the current bit position when either digit is a 1. For borrow-save encoding, the additional step is essentially just conversion from a borrow-save encoding with weights, 1, -1 to a sign-mag encoding with weights -2, 1. After this step, the number has the same properties as in the radix 4 signed bit approach. Fig. 5 shows gate level implementations with the common core logic.

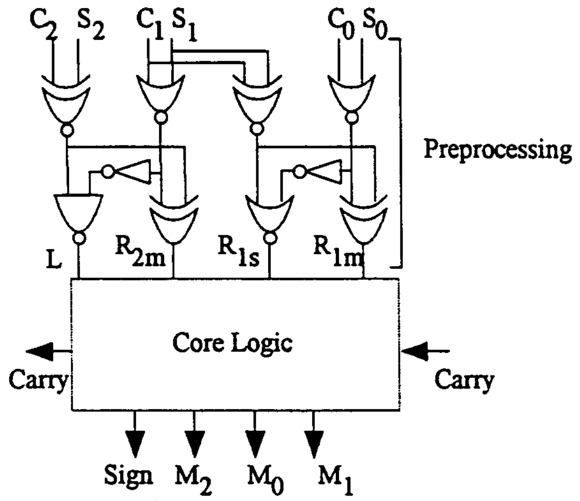


Fig. 5a Radix 4 carry-save Booth recoder

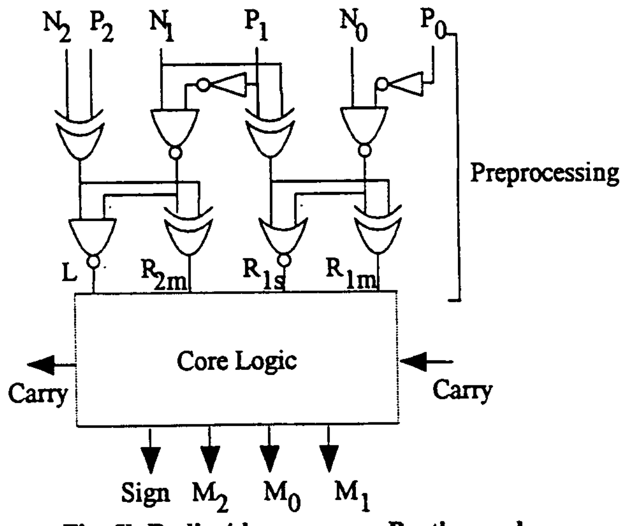


Fig. 5b Radix 4 borrow-save Booth recoder

	Radix 4 Signed bit Booth recoder		Radix 4 Carry-save Booth recoder	
Depth (gate delay)	5 (2-input gates)	4 (2-input gates)	4 (1@4, 2@3, and 1@2-input gates)	5 (2-input gates)
Size (gate count)	10 (2-input gates)	10(-1) (2-input gates)	16 (3@4-, 5@3-, 8@2-input gates)	14(-1) (2-input gates)
Reference	[9] Kabuo, et. al.	Fig. 4	[12] Quek, et. al.	Fig 5a

Table 4 Comparison with other existing designs

Two recent publications/patents [8], [12] provide gate level designs with which we can compare our results. In [8] a form of borrow-save encoding (excluding the 1, 1 pair) is used, and provides for a convenient comparison with our sign-mag encoding as both employ non redundant encodings of the three signed bit values. In [12] carry-save input is used and provides for comparison with our carry-save encoding illustrated in Fig. 5a. Both [8] and [12] have radix 4 output of sign, M_2, and M_1 and exclude generation of M_0, and our gate count should be reduced by one for comparison. In both cases, Table 4 shows our design is smaller in size and also faster after compensating in [12] for the 3- and 4-input gate delay effects.

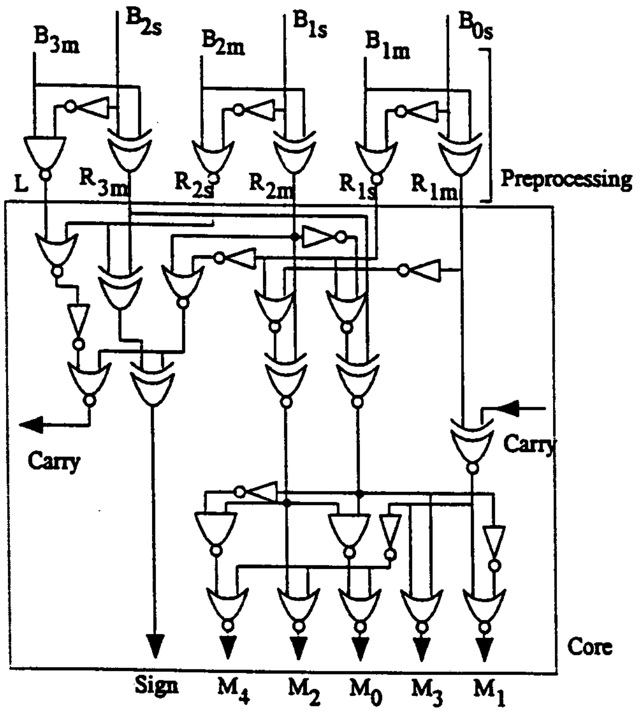


Fig. 6 Radix 8 signed bit Booth recoder

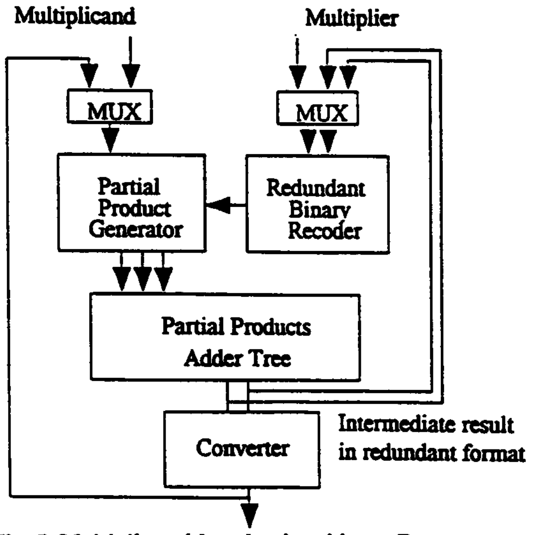
Radix 8 Booth recoding

A sign-mag to radix 8 Booth recoder can be designed using the same methodology. Fig. 6 shows a time efficient gate level implementation. The whole Booth recoding process finishes in a depth of 5 gates. This is only one level more than that for radix 4. The outlined block in Fig. 6 can be used after appropriate preprocessing for carry-save and borrow-save encoded input.

Redundant binary Booth recoding can be designed accordingly for a radix higher than 8 as argued for conventional binary Booth recoders in [13].

Applications and Conclusion

Higher radix redundant binary Booth recoding based on digit set conversions and encodings for signed bit, carry-save, and borrow-save representations are introduced. Multipliers based on signed bit and carry-save Booth recoders are useful for speeding up iteratively dependent multiplication. Fig. 7 illustrates a typical speed-up mechanism. The



Multiplicand
Multiplier
MUX
MUX
Partial Product Generator
Redundant Binary Recoder
Partial Products Adder Tree
Converter
Intermediate result in redundant format

Fig. 7 Multiplier with redundant binary Booth recoder

adder tree adds the partial products into a carry-save or signed bit number that being the input of the multiplier and hence the Booth recoder for successive dependent multiplication. The borrow-save Booth recoder also finds its application in processing a seed reciprocal for division and square root algorithms [7]. A step requires X (B_1-B_2) where B_1 and B_2 are from a ROM table. A multiplier with a borrow-save Booth recoder can combine the subtraction-multiplication process by treating B_1 and B_2 as positive and negative digits of a borrow-save number.

Booth recodings of three different redundant binary representations have been shown that are very regular and suitable for high speed VLSI implementation. Multipliers based on redundant binary Booth recoders have flexibility and can obtain improved performance with no major change in the numerical algorithms.

References

[1] A. Avizienis: Signed-Digit Number Representations for Fast Parallel Arithmetic, *IRE Trans. Electronic Computers, Vol. EC-10, pp. 389~400*, 1961

[2] A. D. Booth: A Signed Binary Multiplication Technique, *Quart. J. Mech. Appl. Math., Vol. 4, Pt. 2, pp. 236-240, 1951*

[3] W. S. Briggs and D. W. Matula, A 17 × 69 Bit Multiply and Add Unit with Redundant Binary Feedback and Single Cycle Latency, *Proc. of IEEE 11th Symp. Comput. Arith., pp. 163~170*, 1993

[4] E. Bryant: Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Trans. on Comput., Vol. C-35, No. 8, pp. 677~691*, Aug. 1986

[5] P. Chai, T. Chuk, Y. H. Fong, L. Hu, Ken Ng, J. Prabhu, A. Quek, A. Samuels, and J. Yeun: A 120 MFLOPS CMOS Floating-Point Processor, *Proc. of IEEE Custom Integrated Circuits Conference, pp. 15.1.1~15.1.4*, 1991

[6] H. M. Darley, M. C. Gill, D. C. Earl, D. T. Ngo, P. C. Wang, M. B. L. Hipona and J. Dodrill: Floating-Point/Integer Processor with Divide and Square Root Functions, U. S. Patent No. 4878190, Oct. 1989

[7] D. Das. Sarma and D. W. Matula: Faithful Bipartite ROM Reciprocal Tables, *Proc. of IEEE 12th Symp. Comput. Arith.*, 1995

[8] H. Kabuo, T. Taniguchi, A. Miyoshi, H. Yamashita, M. Urano, H. Edamatsu, and S. Kuninobu: Accurate Rounding Scheme for the Newton-Raphson Method Using Redundant Binary Representation, *IEEE Trans. Comput., Vol. 43, No. 1, pp. 43~51*, Jan. 1994

[9] P. Kornerup: Digit-Set Conversion: Generalizations and Applications, *IEEE Trans. Comput., Vol. 43, No. 5, pp. 622~629*, May 1994

[10] O. L. MacSorley: High-Speed Arithmetic in Binary Computers, *Proc. of IRE, Vol. 49, pp. 69~71*, Jan. 1961

[11] D. W. Matula: Radix Arithmetic: Digital Algorithms for Computer Architecture in Raymond T. Yeh (editor) *Applied Computation Theory: Analysis, Design, Modeling, pp. 375~448*, Prentice-Hall, Englewood Cliff, New Jersey, 1976

[12] S. M. Quek, L. Hu, J. P. Prabhu, and F. A. Ware: Apparatus For Determining Booth Recoder Input Control Signals, U. S. Patent No. 5280439, Jan. 1994

[13] H. Sam and A. Gupta: A Generalized Multibit Recoding of Two's Complement Binary Numbers and Its Proof with Application in Multiplier Implementations, *IEEE Trans. Comput., Vol. 39, No. 8, pp. 1006~1015*, Aug. 1990

[14] N. Takagi: Arithmetic Unit Based on a High-Speed Multiplier with a Redundant Binary Addition Tree, *Proc. of SPIE, Vol. 1566, "Advanced Signal Processing Algorithms, Architectures, and Implementations II"*, pp. 244~251, Jul. 1991

[15] N. Takagi and S. Yajima: On A Fast Iterative Multiplication Method by Recoding Intermediate Product, *Proc. of 36th National Convention of Information Science*, Kyoto University, Aug. 1987