Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

Carry-Propagation-Free Adder Based on an Asymmetric High-Radix Signed-Digit Number System

Shao-Hui Shieh, Bin-Hong Lin, and Cheng-Wen Wu
Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan
cww@ee.nthu.edu.tw

Abstract

We propose an asymmetric high-radix signed-digit (AHSD) number system for fast binary addition and array multiplication. Practical implementation of a carry-propagation-free adder and an array multipler is discussed and a radix-4 example is designed. AHSD has the advantage of simple conversion such that a practical carry-propagation-free adder and fast multiplier can be achieved. Besides, multiple-valued current-mode circuits are especially suited to our designs which result in very high-performance practical arithmetic circuits.

1 Introduction

Addition is the most important and frequently used operation in computer arithmetic. Generally, two methods can be used to boost this operation. One is explicitly shortening the carry-propagation chain by circuit design techniques or detecting the completion of the carry-propagation chain as soon as possible, so that no time is wasted [1]. Another is by converting the binary number into a redundant number system, e.g., the signed-digit number system and the residue number system, to operate the addition in a carry-propagation-free (CPF) manner. This implicitly eliminates the carry-propagation chain so that a fast addition can be done. Though these number systems are carry free, they have the difficulty in conversion from/to the binary number system. Recently, the CPF adder is investigated based on redundant positive-digit numbers [2] and symmetrical radix-4 signed-digit numbers [3] for high-speed area-effective multipliers.

In this paper, an asymmetric high-radix signed-digit (AHSD) number system is proposed. By the AHSD representation, an inherent CPF addition is developed, which is the basis for our high-performance sequential addition and array multiplication circuits. The detailed adder and multiplier designs on AHSD(4), the radix-4 AHSD which are suitable for VLSI implementation by using multiple-valued current-mode (MVCM) logic circuits, are presented. Our approach is shown to be practical due to simple conversion interface and fast MVCM circuits.

2 The AHSD Number System

An n-bit unsigned binary number X is represented in our AHSD number system as

$$X = \sum_{j=0}^{q-1} X_j r^j, \tag{1}$$

where $-1 \leq X_j < r$ and $r = 2^m$ for some integer m such that n is divisible by m. The number X is denoted as

$$X = (X_{q-1}, X_{q-2}, \cdots, X_0)_r. \tag{2}$$

In the context, if not otherwise specified, $(x_{n-1}, \cdots, x_1, x_0)$ is used as the binary representation of X. Also, a radix-r AHSD is denoted as AHSD(r). Arithmetic operations based on our AHSD number system can be performed if the operands are all available in the AHSD representation.

Proposition 1 (AHSD Representation) *An unsigned number $X = (x_{n-1}, \cdots, x_1, x_0)$ has an AHSD(r) representation as follows:*

$$X = \sum_{j=0}^{q-1} X_j r^j = \sum_{j=0}^{q-1} (\sum_{k=0}^{m-1} x_{k+mj} 2^k) r^j, \quad (3)$$

where $X_j \in \{-1, 0, 1, \cdots, r-1\}$, $r = 2^m$, and $q = n/m$.

The digit -1 is introduced into the system for flexibly applying our AHSD number system to some arithmetic operation. The arithmetic based on AHSD is developed next.

Table 1: Conversion of Z_j in AHSD(r).

Z_j	Z_{j-1}	C_j	μ_j
-1	x	0	-1
0	x	0	0
1	x	0	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$r-2$	x	0	$r-2$
$r-1$	$< r-1$	0	$r-1$
	$\geq r-1$	1	-1
r	x	1	0
$r+1$	x	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2(r-1)$	x	1	$r-2$

2.1 Carry-Propagation-Free Addition

By the AHSD representation, fast CPF addition can be done. Let $X = (X_{q-1}, \cdots, X_1, X_0)_r$ and $Y = (Y_{q-1}, \cdots, Y_1, Y_0)_r$ be two nonnegative AHSD(r) numbers. For consistency, $X_j \geq 0$ for each j. The addition of $S = X + Y$ can be realized as follows:

1. Internal Individual Summation (IIS): Simply sum up the individual digits of the two operands X and Y such that

$$Z_j = X_j + Y_j, \ 0 \leq j \leq q-1, \quad (4)$$

where Z_j's are the digit sums. Apparently, $Z_j \in \{-1, 0, \cdots, 2(r-1)\}$ if $X_j \geq 0$, which may not fall in the field of our AHSD, so conversion is required.

2. Internal Consistency Conversion (ICC): For consistency in our AHSD operation, conversion must be made to Z_j.

   (a) Self-Adjustment (SA): For each Z_j, an intermediate carry digit C_j and an intermediate sum digit μ_j is chosen such that

$$Z_j = r \times C_j + \mu_j \quad (5)$$

   subject to

$$(C_j, \mu_j) = \begin{cases} (0, Z_j), & \text{if } (Z_j < r-1) \vee \\ & ((Z_j = (r-1) \wedge \\ & (Z_{j-1} < r-1))) \\ (1, Z_j - r), & \text{otherwise.} \end{cases} \quad (6)$$

   Table 1 lists the details of our conversion rule. It is seen that the conversion results in two digits in AHSD for each Z_j. In the table, "x" stands for any value in $\{-1, 0, \cdots, 2(r-1)\}$.

   (b) Adjacent Modification (AM): The final sum digit S_j is then modified by

$$S_j = \mu_j + C_{j-1}, \ 0 \leq j < q. \quad (7)$$

   where $C_{-1} = 0$. By doing so, each digit of the final result $S = (C_{q-1}, S_{q-1}, \cdots, S_0)$ falls in our AHSD representation.

The requirement that $X_j \geq 0$ preserves the consistency of our addition as described by Table 1. With the negative digit -1 introduced, the conversion table is simpler than those of other redundant number systems. Besides, an AHSD number can be easily converted to/from a binary number. This makes our AHSD a realistic candidate for fast arithmetic computation.

This procedure can be illustrated by Fig. 1. In the first step, each intermediate summation digit is obtained concurrently. In the second step, only

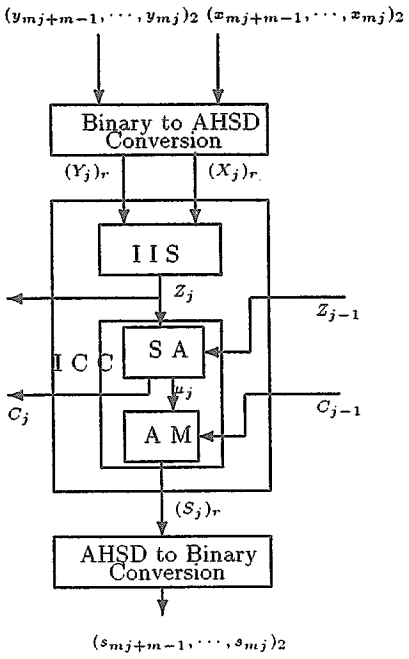

Figure 1: The addition procedure.

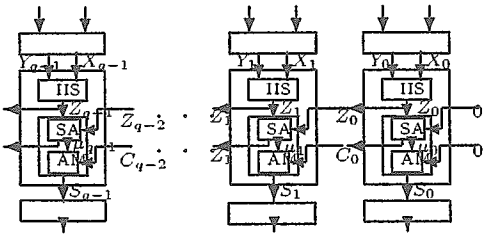

Figure 2: Structure of our AHSD CPF adder.

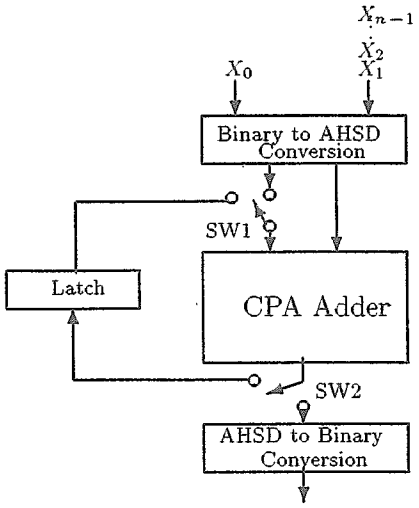

Figure 3: Construction of the sequential adder.

adjacent information are required for the consistency conversion. Apparently, the carries are localized to their adjacent digits, and every μ_j and C_j are generated concurrently. Therefore, the proposed AHSD addition can be performed in a constant time independent of the length of the operands (q), i.e., our AHSD adder is carry-propagation-free. The adder structure is shown in Fig. 2.

Generally, a binary result is preferred for most applications. This final conversion can be done by Prop. 1.

2.2 Sequential Addition

The CPF adder can be used effectively in sequential addition and array multiplication. A sequential adder is used to evaluate sequentially the sum of q numbers, X_i, $0 \le i \le q-1$, which can be expressed as

$$S = \sum_{i=0}^{q-1} X_i = (\cdots((X_0 + X_1) + X_2) + \cdots) + X_{q-1}. \tag{8}$$

This sequential adder can be realized by using a CPF adder and a latch L as shown in Fig. 3.

The time required for a n-operand sequential

Table 2: Conversion rule for AHSD(4).

Z_j	Z_{j-1}	C_j	μ_j
−1	x	0	−1
0	x	0	0
1	x	0	1
2	x	0	2
3	< 3	0	3
	≥ 3	1	−1
4	x	1	0
5	x	1	1
6	x	1	2

addition is

$$T_{add} = t_{BA}+(n-2)[max(t_{BA}, t_L)]+(n-1)t_{CPF}+t_{AB}, \quad (9)$$

where t_L, t_{BA}, t_{CPF}, and t_{AB} denote the processing time for latching, binary-to-AHSD(4) conversion, CPF addition, and AHSD(4)-to-binary conversion, respectively. Speedup results from adding a large number of operands concurrently. The conversion time is small and can be neglected. The processing time is therefore approximately

$$T_{add} \simeq n[max(t_{BA}, t_L) + t_{CPF}], \quad (10)$$

regardless of the operand length.

3 AHSD(4)

The radix $r = 2^m$ is often used in practical applications. We use AHSD(4) to illustrate design examples in this section.

3.1 Implementation of the AHSD(4) CPF Adder

Consider the two binary numbers X = (11111111) and Y = (00000001) to be added. Apparently the longest carry propagation chain occurs in this case if ordinary ripple-carry adder is used. By using the proposed addition algorithm, we can generate the sum without any carry propagation as shown in Fig. 4.

	11	11	11	11	X	Binary
+	00	00	00	01	Y	
	3	3	3	3	X	Binary to AHSD(4)
	0	0	0	1	Y	
	3	3	3	4	Z	IIS
1	1	1	1		C	SA
	-1	-1	-1	0	μ	
1	0	0	0	0	S	AM
1	00	00	00	00	S	AHSD(4) to binary

Figure 4: The proposed addition algorithm.

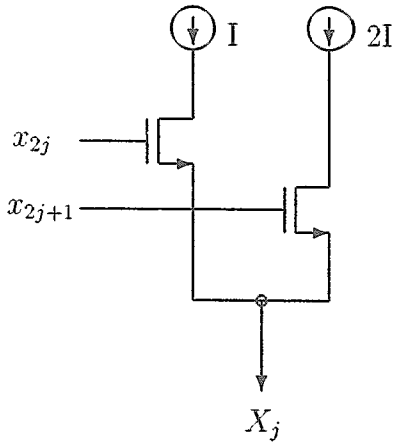

Figure 5: Binary to AHSD(4) converter.

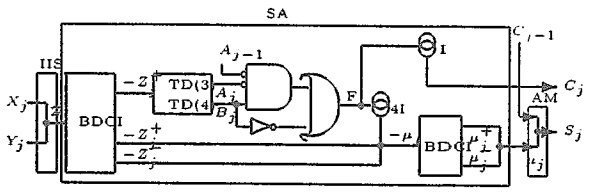

Figure 6: Bidirectional current mode symbolic representation of the CPF adder.

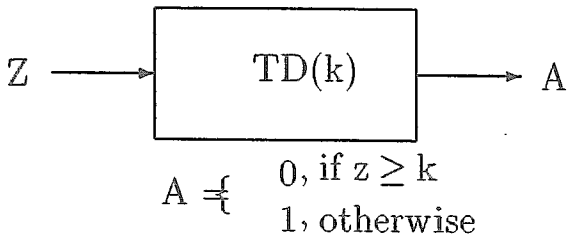

Figure 7: The threshold detector.

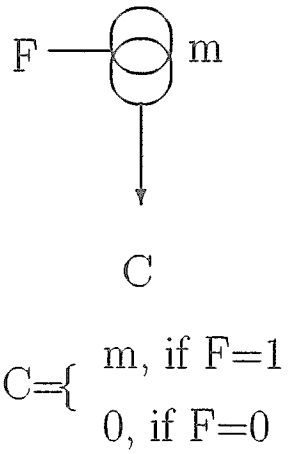

Figure 8: The voltage controlled current source.

Using the MVCM circuit, the AHSD(4) CPF adder can be implemented according to the following procedure.

1. Binary to AHSD(4) Conversion:
   A binary number can be partitioned, from the LSB to the MSB, into 2-bit blocks, which can then be converted directly into their corresponding AHSD(4) digits. A current-mode CMOS binary-to-quaternary encoder proposed by Current [4] can be used as the binary to AHSD(4) converter as shown in Fig. 5, which is simply a current adder constructed with two voltage controlled current source circuits and linear summation operation.

2. CPF Adder Based On AHSD(4):
   The CPF adder is also suitable for VLSI implementation using MVCM circuits. The most important advantage in using MVCM is that both IIS and AM steps in AHSD(4) addition can be performed by bidirectional wired summation as introduced by Kawahito *et al.* [3]. Fig. 6 shows the bidirectional current mode symbolic representation of the AHSD(4) CPF adder. In the figure, symbols A_j, A_{j-1}, and B_j represent the corresponding conditions $Z_j < 3$, $Z_{j-1} < 3$, and $Z_j \geq 3$. The polarity of Z_j can be detected by a bidirectional current input circuit (BDCI) [3]; hence, Z_j^+ and Z_j^- represent positive and negative Z_j values, respectively. The threshold detector is defined in Fig. 7. The function of voltage controlled current source is shown in Fig. 8. Details on the MVCM circuit elements in our design can be found in [3,5].

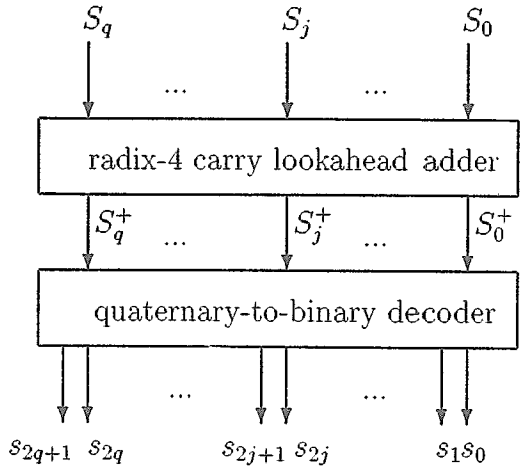

Figure 9: AHSD(4) to binary converter.

3. AHSD(4) to Binary Conversion:
   To convert the sum digit S_j to the equivalent final binary bits, we may first convert $S_j \in \{-1, 0, 1, 2, 3\}$ to $S_j^+ \in \{0, 1, 2, 3\}$ by a radix-4 carry-lookahead adder. This step is the most time-consuming in the procedure, and its computation time is proportional to $\log_2 \frac{n}{2}$. We can then convert S_j^+ to the equivalent binary bits $(s_{2j+1}, s_{2j})_2$ by the current mode CMOS quaternary-to-binary decoder [4]. The structure of the AHSD(4) to binary converter is shown in Fig. 9.

3.2 Array Multiplication on AHSD(4)

High performance array multiplier can be designed by using the CPF adders. Assume the two inputs are 8-bit numbers $X = (x_7, x_6, \cdots, x_0)$ and $Y = (y_7, x_6, \cdots, y_0)$. The 16-bit product $P = (p_{15}, p_{14}, \cdots, p_0)$ is then obtained as follows:

1. Generate partial products p_{ij}:

$$p_{ij} = x_i \times y_j,\ p_{ij} \in \{0, 1\};\ 0 \leq i, j \leq 7. \tag{11}$$

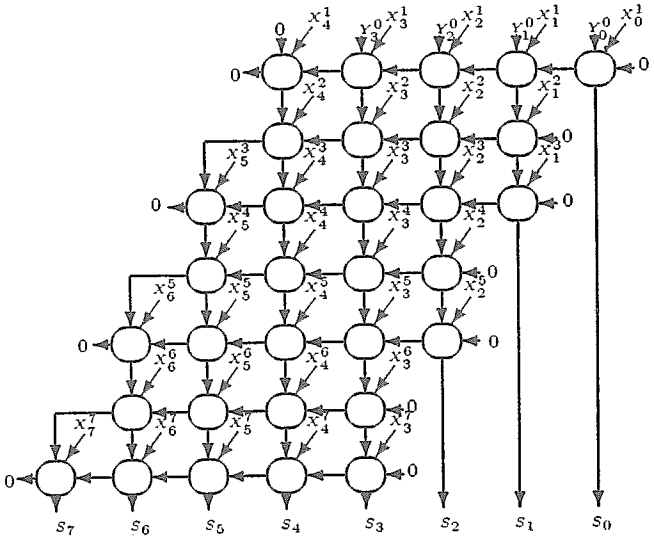

Figure 10: The array multiplier based on AHSD.

Thus, partial products $P_j = (p_{7j}, p_{6j}, \cdots, p_{0j})$, $0 \leq j \leq 7$, can be generated.

2. Convert each P_j into AHSD(4) according to

$$\begin{cases} Y_k^0 = & 2p_{(i+1,0)} + p_{(i,0)}, \ k = \frac{i}{2}; \\ X_k^j = & 2p_{(i+1,j)} + p_{(i,j)}, \ k = \frac{i+j}{2}; \\ & 0 \leq i \leq 7, \ 1 \leq j \leq 7. \end{cases} \quad (12)$$

3. Add up the partial products Y_k^0 and X_k^j from $j = 1$ to 7. All additions are performed in parallel at each level in the array multiplier. Finally, the sum $S = (S_7, S_6, \cdots, S_0)_4$ is generated. If the binary result is preferred, it can be generated by an AHSD-to-binary conversion circuit.

The array multiplier is shown in Fig. 10. The computation time for $n \times n$-bit array multiplication is

$$T_{MUL} = t_{AND} + t_{BG} + (n-1)t_{CPFA} + t_{GB}, \quad (13)$$

where t_{AND} denotes the processing time for an AND operation. The total number of CPF adders used is $N = \frac{n^2}{2}$.

4 Conclusion

In this paper, a novel redundant number system referred to as AHSD is proposed, which is effective for CPF addition, sequential addition, and array multiplication. Its conversion interface with the binary number system is shown to be simple. AHSD(4) is used as an example and shown to be suitable for VLSI MVCM circuit implementation.

References

[1] I. Koren, *Computer Arithmetic Algorithms.* Englewood Cliffs, New Jersey, 07632: Prentice-Hall, 1993.

[2] S. Kawahito, M. Ishida, M. Kameyama, and T. Higuchi, "High-speed area-efficient multiplier design using multiple-valued current-mode circuits," *IEEE Trans. Computers*, vol. 43, no. 1, pp. 34–42, Jan. 1994.

[3] S. Kawahito, M. Kameyama, T. Higuchi, and H. Yamada, "A 32×32-bit multiplier using multiple-valued MOS current-mode circuits," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 1, pp. 124–132, Feb. 1988.

[4] K. W. Current, "Current-mode CMOS multiple-valued logic circuits," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 2, pp. 95–107, Aug. 1994.

[5] A. K. Jain, R. Bolton, and M. Abd-El-Barr, "CMOS multiple-valued logic design—Part I: Circuit implementation," *IEEE Trans. Circuits and Systems*, vol. 40, no. 8, pp. 503–514, Aug. 1993.