iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** V **Month of publication:** May 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

Volume 3 Issue V, May 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Multiplier Using Canonical Signed Digit Code

Vishwanath B.R.[1] Theerthesha.T.S [2]
[1,2]Assistant Professor, Dept. of ECE, RIT Hassan (India)

Abstract--Real-time implementation of many digital signal processing (DSP) algorithms and multimedia applications are limited by the available speed, energy efficiency, and area requirement of multiplication. This is the major drawback in handheld multimedia devices due to the limited battery lifetimes. A novel canonical signed digit (CSD) iterative multiplier structure in which the conversion from 2's complement to CSD representation is implicitly implemented in real-time.

I. INTRODUCTION

Digital signal and image processing applications require a large number of floating point multiplications. For such applications fast multiplication techniques are required to improve the overall system speed. Fast multiplication can be achieved by reducing number of partial products. Canonical signed digit is a recoding technique, which recodes a number with minimum number of non-zero digits. As the number of partial products depends on the number of non-zero digits, by using Canonical recoding, the number of non-zero digits will be reduced, thereby reducing the number of partial products. Here floating point multiplication using canonical signed digit is proposed and is compared with Conventional multiplication technique. The design will be implemented in Verilog and simulated using Xilinx 9.2 ISE. Array multipliers and parallel multipliers are used widely when high speed multiplication is required. However, in addition to requiring large area, array multipliers usually do not seek to optimize energy efficiency though exploitation of the specific data dependent patterns of digits that occur in the multiplier and multiplicand; typical array multipliers are inherently energy inefficient in this regard. CSD representations have proven to be useful in implementing multipliers with reduced complexity, because the cost of multiplication is a direct function of the number of nonzero bits in the multiplier. For an n-bit 2's complement multiplier the number of non-zero bits in its CSD representation never exceeds n/2 and can be reduced to n/3 on average, as the word length of the multiplier grows. Therefore,by incorporating the CSD number representation into multiplier, the number of nonzero partial products were reduced, which in turn increases the multiplier throughput and energy efficiency.

II. CSD MULTIPLIER IMPLEMENTATION

A. Canonic Sign Digit (CSD)

Canonical Signed Digit (CSD) is a type of number representation. The important characteristics of the CSD presentation are:

CSD presentation of a number consists of numbers 0, 1 and -1.
The CSD presentation of a number is unique.
The number of nonzero digits is minimal.
There cannot be two consecutive non-zero digits

B. Conversion from Binary to CSD

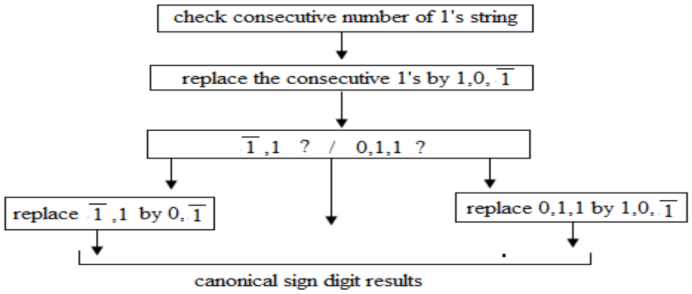


Figure 1:- Conversion from binary to CSD

Example 1
A number 287, which is 1 0001 1111 in binary representation. (256 + 16 + 8 + 4 + 2 + 1 = 287)=1 0001 1111

Starting from the right (LSB), if there are more than one non-zero elements (1 or -1) in a row, take all of them, plus the next zero. (if there is not zero at the left side of the MSB, create one there). We see that the first part of this number is 01 1111. Add 1 to the number i.e. changes the 0 to 1, and all the 1's to 0's, and force the rightmost digit to be -1.

01 1111=>10 000-1
The number is still the same: 16 + 8 + 4 + 2 + 1 = 31 = 32 + (-1). Now the number looks like this
1 0010 000-1
Since there are no more consecutive non-zero digits, the conversion is complete. Thus, the CSD presentation for the number 287 is 1 0010 000-1, which is 256 + 31 - 1.
Example 2
Number 345. In binary, it is
1 0101 1001
Find the first place (starting from right), where there are more than one non-zero numbers in a row. Take also the next zero. Add one to it, and force the rightmost digit to be -1.
1 0110 -1001
Take the 011, and add one to it (get 100), and force the last digit to be -1. (get 10-1). Now the number looks like this
1 10-10 -1001
Do the same thing again. This time, imagine a zero in the left side of the MSB.
10-10-10-1001

Table1:- Conversion of binary number to CSD digit representation

Yi+2	*Yi*+1	*Yi*	*Xi*+3	*Xi*+2	*Xi*+1	*Xi*
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	1	0
0	1	1	0	1	0	-1
1	0	0	0	1	0	0
1	0	1	0	1	0	1
1	1	0	1	0	-1	0
1	1	1	1	0	0	-1

Circuit module has been carried out via Table 1, and shown in below Figure. In this architecture *yi*, *yi*+1, *yi*+2 are the present state inputs, and the corresponding outputs are *xi*, *xi*+1, *xi*+2, *xi*+3. Si_*xi*,Si_*xi*+1, Si_*xi*+2,Si_*xi*+3 are representing the corresponding sign bits of the outputs.

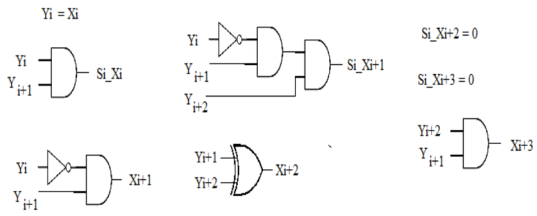


Figure.2:-CSD RECODER CIRCUIT

C. Multiplication Algorithm Using CSDC
Convert the multiplier into CSDC
Generate N X N bits partial products
Add the partial products
Convert addition results into 2's complement format

Volume 3 Issue V, May 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

TABLE 2:- CSD Multiplication Technique;

x_i	y_i	P_i
0	0	0
0	1	0
1	0	0
1	1	1
0	$\bar{1}$	0
$\bar{1}$	0	0
1	$\bar{1}$	$\bar{1}$
$\bar{1}$	1	$\bar{1}$
$\bar{1}$	$\bar{1}$	1

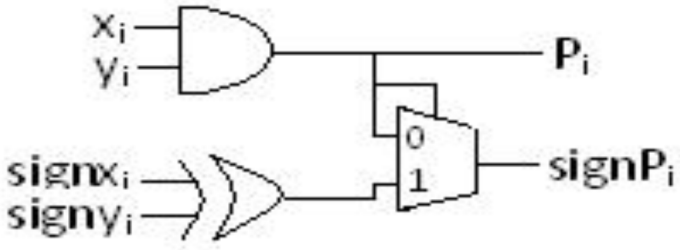


Figure.3:- CSD Multiplication logic

D. CSD Addition/Subtraction

The canonical sign digit adder/subtractor (CSD adder/subtractor) performs carry propagation free addition. Carry propagation free addition has been performed by determining the intermediate carry and intermediate sum digits. Carry propagation free addition has been performed in three steps:

Check the type of operation (addition). For addition, the sign of the individual bits remains unchanged. For subtraction, the signs of the individual nonzero bits are inverted.

Determine the intermediate carry, Ci (_1; 0; 1), and intermediate sum digits, Si (_1; 0; 1), satisfying the condition, xi+yi = zi+Ci-1, where xi+1 and yi+1 are the augends and addend digits, respectively.

Obtain the sum digits, Zi (-1; 0; 1), at each position by adding the intermediate sum digits, Si and Ci, from the next lower order positions. The truth table implementation from step (ii) is shown in Table 3.1. Boolean expressions have been formed from the above steps and shown in Eqs. (3.1)- (3.6). Here, 'zi' and 'ci-1' represent the intermediate sum and the intermediate carry. 'signxi' and 'signyi' represent the sign magnitude of 'xi' and 'yi', respectively. 'Signci-1' and 'signzi' are the sign magnitude of the intermediate carry and intermediate sum, respectively. 'sumi' and 'signsumi' are sum and its sign magnitude, respectively. A canonical sign digit adder circuit has been used here for both addition and subtraction. To implement the subtractor, a small hardware was added with the adder circuit. Hardware implementation of the adder/subtractor is shown in Figure

3.6. The architecture for the CSD adder/subtractor can be decomposed into two sections, viz, addition/ subtraction, through CSDC and CSD, to binary conversion.

$$Zi = xi \oplus yi \quad \text{.....3.1}$$

$$ci\text{-}1 = (signxi+1 \oplus signyi+1)(xi \oplus yi) \oplus (signxi \oplus signyi) \quad \text{.....3.2}$$

$$signzi = zi\,(\overline{signxi+1 + signyi+1}) \quad \text{.....3.3}$$

$$signci\text{-}1 = (\overline{xi \oplus yi})(signxi.signyi) + (xi \oplus yi)(signxi+1 + signyi+1) \quad \text{...3.4}$$

$$sumi = (zi \oplus ci\text{-}1) \quad \text{.....3.5}$$

$$signsumi = (signzi \oplus signci\text{-}1)(zi \oplus ci\text{-}1) \quad \text{.....3.6}$$

Table 3. Truth table for determining the intermediate sum and intermediate carry.

Volume 3 Issue V, May 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Augend digits (xi)	Addend digits (yi)	Digits of the previous higher order posions (x1+1; yi+1)	Intermediate carry (Ci-1)	Intermediate sum (zi)
0	0	----------	0	0
0	1	both are non-negative	1	-1
1	0	Otherwise	1	0
1	1	----------	0	1
0	-1	both are non-negative	0	-1
-1	0	Otherwise	-1	1
1	-1	----------	0	0
-1	1	----------	0	0
-1	-1	----------	-1	0

Sign digit adder circuit has been used here for both addition and subtraction. To implement the subtractor, a small hardware was added with the adder circuit. Hardware implementation of the adder/subtractor is shown in Figure 3.6. The architecture for the CSD adder/subtractor can be decomposed into two sections, viz, addition/ subtraction, through CSDC and CSD, to binary conversion. Boolean expressions are to be formed from the above steps, as shown in Eqs. (3.1)-(3.6). Here, 'zi' and 'ci-1' represent the intermediate sum and intermediate carry. 'signxi' and 'signyi' represent the sign magnitude of 'xi' and 'yi' respectively. 'signci-1' and 'signzi' is the sign magnitudes of intermediate carry and intermediate sum, respectively, and 'sumi' and 'signsumi' are the sum and its sign magnitude, respectively. The second segment, consisting of a half adder and a sub-tractor, is used for the conversion of CSD to a binary number system. A Boolean expression is shown here only for addition.

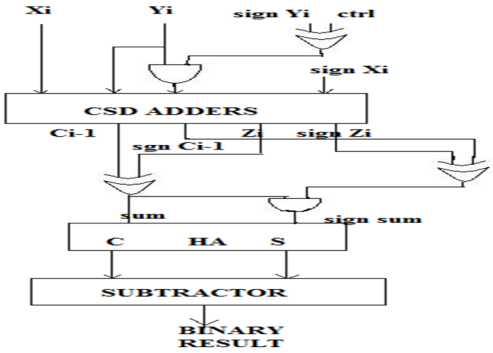


Figure 4:-CSD adder/subtractor.

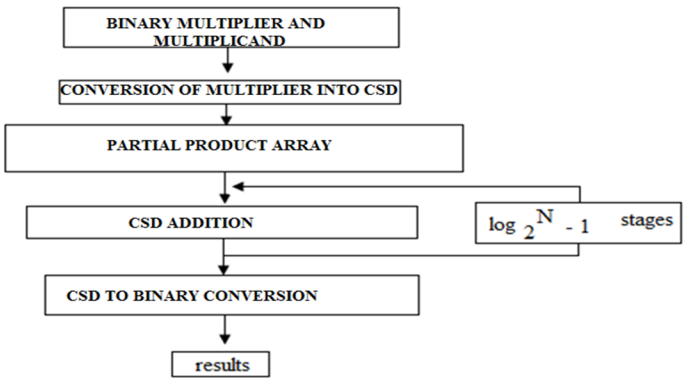


Figure 5:-Flowchart diagram of improved multiplication algorithm using CSD.

Volume 3 Issue V, May 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

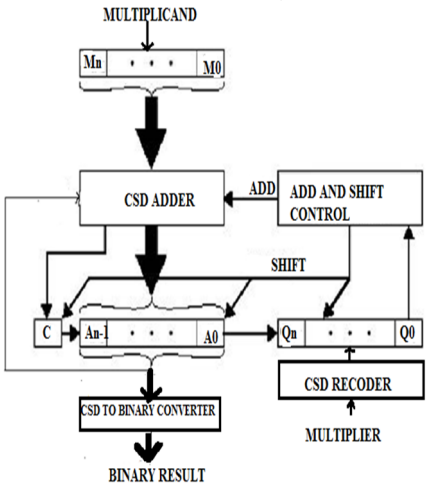


Figure 6:-CSD multiplier logic diagram.

E. Conventional Multiplication Example

Example: 14 multiplied with 15 should give 210.

Multiplicand => 1110

Multiplier=> 1111

1 1 1 0 X 1 1 1 1

1 1 1 0

1 1 1 0

1 1 1 0

1 1 1 0

11010010

F. Multiplication using CSDC

Example: - 14 X 15 = 210

Multiplicand=> 14=1110

Multiplier=> 15=1111

Convert multiplier into CSD format i,e 1111=>1000-1 and multiply it with multiplicand.

1110 X 1000-1

-1-1-10

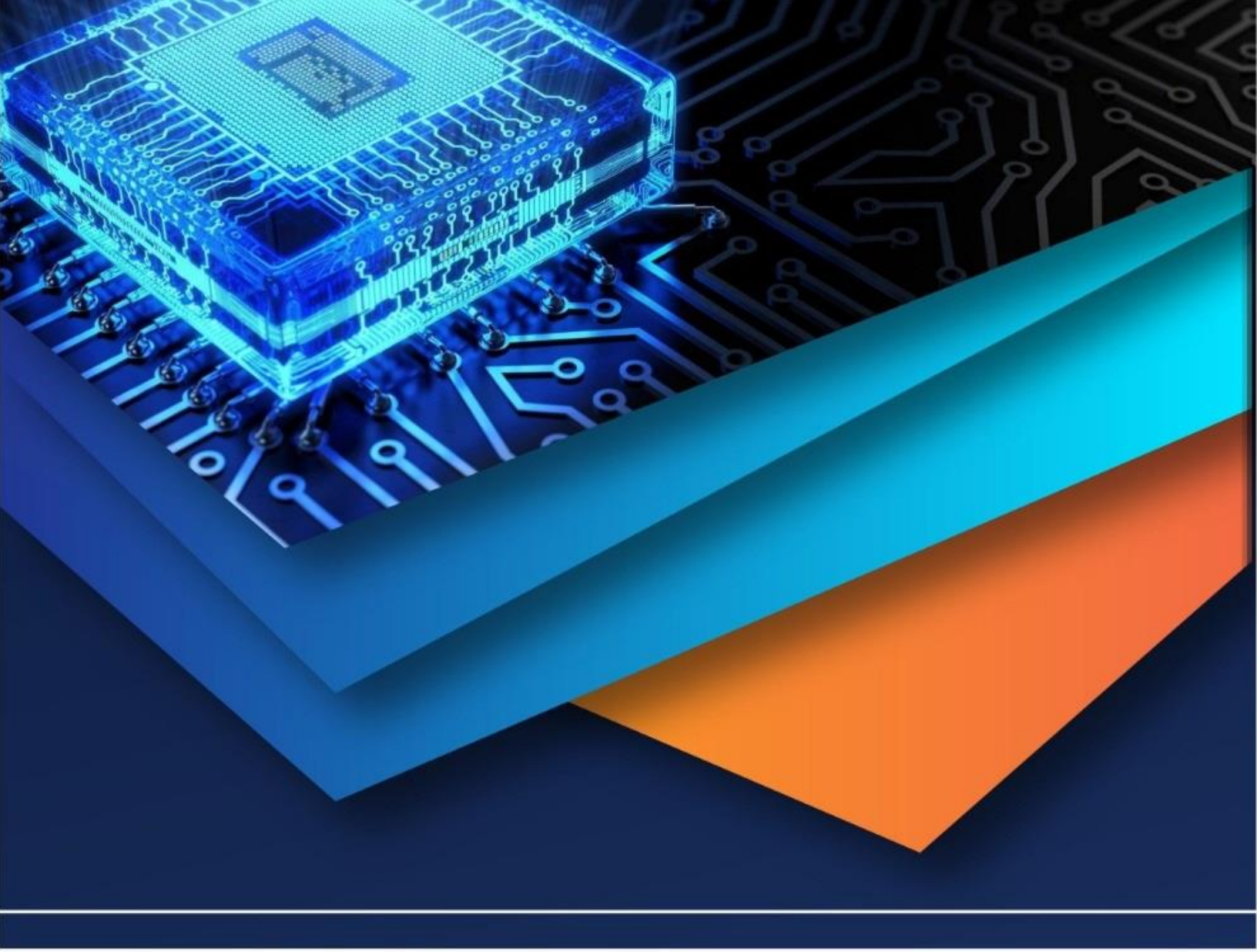
1 1 1 1

III. RESULTS AND CONCLUSION

The implementation methodology ensures stage reduction, leading to substantial reduction of the propagation delay and power. A high speed multiplier, using a high accuracy CSD recoder conversion methodology, was designed for practical digital signal processors. Multiplication of higher order bits requires a large number of hardware components, due to the generation and processing of huge partial products. In these schemes, partial product handling was avoided by using CSDC, where multiplication reduces to direct addition. The improvement in speed, by avoidance of carry propagation, was achieved through Canonical Signed Digit Code (CSDC) implementation. The corresponding improvement, in terms of power, was found to be _ 54%, _ 66%, and _ 61%, respectively, with reference to the above conventional methodologies.

REFERENCES

[1] J. Kang and J. Gaudiot, "A simple high-speed multiplier design," IEEE Trans. Comput., vol. 55, No. 10, pp. 1253-1258, Oct. 2006.
[2] A. Efthymiou, W. Suntiamorntut, J. Garside, and L.E.M. Brackenbury, "An asynchronous, iterative implementation of the original Booth multiplication algorithm," in Proc. 10th IEEE Int'l. Symp. Asynchronous Circuits, and Syst., Crete, Greece, Apr. 19-23, 2004, pp. 207-2 15.
[3] J. Hensley, A. Lastra, and M. Singh, "An area- and energy-efficient asynchronous Booth multiplier for mobile devices," in Proc. IEEE Int'l. Conf Comput. Design, San Jose, CA, Oct. 11-13, 2004, pp. 18-25.
[4] M.A. Soderstrand, "CSD multipliers for FPGA DSP applications," in Proc. IEEE Int'l. Symp. C
ircuits, Syst., vol. 5, Bangkok, Thailand, May 25-28, 2003, pp. V-469 - V-472.
[5] C.-L. Chen, K.-Y. Khoo, and A.N. Willson, Jr., "A simplified signed powers-of-two conversion for multiplierless adaptive filters," in Proc. IEEE Int'l. Symp. Circuits, Sys., vol. 2, Atlanta, GA, May 12-15, 1996, pp. 364-367.
[6] G.K. Ma and F.J. Taylor, "Multiplier policies for digital signal processing," IEEEASSP Mag., vol. 7, no. 1, pp. 6-20, Jan. 1990.
[7] G.A. Ruiz and M.A. Manzano, "Self-timed multiplier based on canonical signed-digit recoding," IEE Proc.: Circuits, Devices, Syst., vol. 148, no. 5, pp. 235-241, Oct. 2001.
[8] S.-M. Kim, J.-G. Chung, and K.K. Parhi, "Design of low error CSD fixed-width multiplier," in Proc. 2002 IEEE Int'l. Symp. Circuits, Syst., vol. 1, Scottsdale, AZ, May 26-29, 2002, pp. 1-69 - 1-72.
[9] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, Oxford Press, London, 1999.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY