# High Speed radix256 algorithm using parallel prefix adder

T. Madhusudhan[1], Dr.A. Balaji Nehru[2], Shanmugha Priya[3]

[1](*Department of Electronics and Communication Engineering, CMR Institute of Technology, JNT University, Hyderabad,)*

[2](*Professor, Department of Electronics and Communication Engineering, CMR Institute of Technology, JNT University, Hyderabad )*

[3](*Assistant Professor, Department of Electronics and Communication Engineering, CMR Institute of Technology, JNT University, Hyderabad,)*

***ABSTRACT:*** *A finite impulse response (FIR) filter computes its output using multiply and accumulate operations. In the present work, a FIR filter based on novel higher radix-256 and RB arithmetic is implemented. The use of radix-256 booth encoding reduces the number of partial product rows in any multiplication by 8 fold. In the present work inputs and coefficients are considered of 16-bit. Hence, only two partial product rows are obtained in RB form for each input and coefficient multiplications. These two partial product rows are added using carry free RB addition. Finally the RB output is converted back to natural binary (NB) form using RB to NB converter. The performance of proposed multiplier architecture for FIR filter is compared with computation sharing multiplier (CSHM)*

***Keywords*** *- About five key words in alphabetical order, separated by comma*

## I. Introduction

Communication systems lead to high performance and low power VLSI digital signal processing systems. Filters and other signal processing units used in these systems need to have high speed and low power realization. Finite impulse response (FIR) filters play a crucial role in many signal processing applications. Various tasks such as spectral shaping, matched filtering, interference cancellation, channel equalization, etc. can be performed with these filters due to its absolute stability and linear-phase property. Hence, various architectures and implementation methods have been proposed to improve the performance of filters in terms of speed and complexity. Since complexity reduction of FIR filter leads to high performance as well as low-power design, authors of [1]-[4] have focused on the implementation of FIR filters with discrete coefficient values selected from the powers-of-two coefficient space. Canonical sign digit codes, numbers can be represented as sums or differences of powers-of-two, are often used to represent the FIR filter coefficients. By using add and shift operations FIR filter structure can be simplified. Optimization algorithm is used by Mustafaet. All in [5] to find the coefficients with reduced number of signed-power- of-two. The Differential Coefficients Method is proposed by Sankarayya in [6] and by A. P. Vinod in [7]. This involves using various orders of differences between coefficients, along with stored intermediate results, rather than using the coefficients themselves directly for computing the partial products in the FIR equation. In [8] authors have tried to implement the FIR filter in cascade form. However the numerical design and optimization of such structure are of much more difficult than the single-stage. In the present work, simple and high performance FIR filter architecture is proposed. This is based on the Novel Partial product Generation method using higher radix-256 Booth encoding (NPGHB). Use of radix-256 encoding reduces the number of partial product rows by 8-fold. Hence for a 16-bit coefficient multiplication with input, only two rows of partial products will be obtained. Again RB number system has unique characteristic of carry propagation free addition. The use of RB addition to add two partial product rows makes multiplication operation much faster.

As the proposed work is based on CSHM algorithm [9], same is discussed in brief in section 2. The RB number system and the higher radix booth encoding is discussed in section 3 The proposed architecture and its implementation with comparison are discussed in section 4 and5. Finally the concluding remark is given in section 6

### 1. Computation Sharing Multiplication In Fir Filter

A simple filter computation using convolution is given as:

$$Y(n) = \sum_{k=0}^{M-1} C_k X(n-k) \qquad (1)$$

Where Ck’s are filter coefficients and X(n-k) are the input to the filter delayed by k units

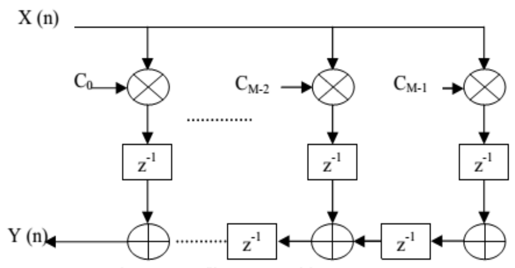


Figure 1. FIR filter transposed form structure

This can be implemented using the direct or the using the transposed direct form realization as given in Figure 1. In this FIR filter the coefficient multiplication C0, C1,---,CM-1with input X(n) using conventional multiplier is hardware expensive and time consuming. For high performance a computation sharing multiplication (CSHM) approach for FIR filter implementation as shown in Figure. 2 is proposed in [5]. In this the coefficient is divided in to group of small bit sequence named as alphabet. Multiplication of input with each alphabet is obtained by selecting from the outputs of a precomputer and then shifting as required. Finally a multiplication result of a coefficient with input is obtained by adding the alphabet multiplication results with required shifting.

For illustration of use of CSHM algorithm, a 16-bit coefficient C0= 1010 1101 1011 0110 can be divided into four alphabets alpha1=1010, alpha 2=1101, alpha 3=1011 and alpha 4= 0110. An alphabet of 4-bit, can have possible decimal values of 0, 1, ----, 14 or 15. All these alphabet multiplications can be obtained from few fundamental alphabet multiplications i.e. 1X, 3X, 5X, 7X, 9X, 11X, 13X and 15X, where X is the input. Hence these fundamental alphabet multiplications are obtained using pre-computer stage as shown in Figure 2. Then based on the alphabets of each coefficient, outputs of pre-computer are selected and finally added to get the coefficient multiplication using select and add units(S &A). For the present illustration, the multiplication of coefficient with X is:

$$C_0 * X = 2^{12}(\alpha_3 * X) + 2^8(\alpha_2 * X) + 2^4(\alpha_1 * X) + (\alpha_0 * X)$$

Here alpha*x= (1010)2 = (10)10*X can be obtained by selecting the fundamental alphabet multiplication 5X from precomputer output and shifting by one bit. Similarly all other alphabet multiplications can be obtained by selecting the proper fundamental alphabet multiplication and then shifting as required using the S & A unit. Then alphabet multiplications are aligned properly and then added to get coefficient multiplication of C0with X.

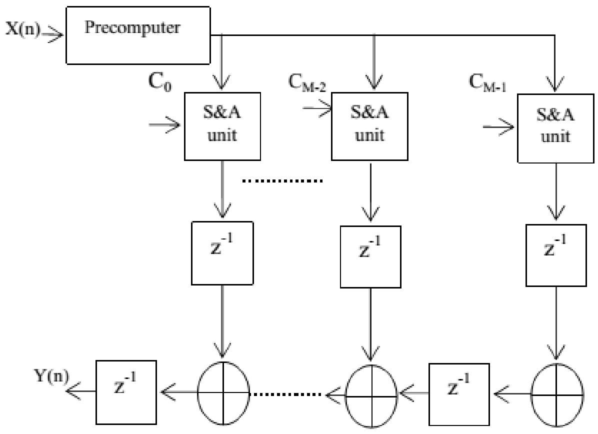


Figure 2. FIR filter using CSHM algorithm

## II. RADIX256

In the present work a Novel Partial product Generation method using higherradix-256 Booth encoding (NPGHB) is used. Use of this new scheme reduces the number of partial product rows to only two, with small additional complexity. For generating the partial product rows, addition of S-groups and T-groups, is used. Hence the four partial product rows generated using radix-256 can be added using RB adders in three stages. This reduces both delay and hardware requirement. In radix-256 encoding, the binary number in two's complement form is partitioned to overlapping group of nine bits to form digit Di. This Dican have one of the

257 values from –128, -127,-1, 0, 1, 127, 128. The partial product generation for digit D in the form of 2m(where m=1, 2, 3, 4, 5, 6, 7) is easy, as can be obtained just by shifting. Generation of other partial products for digits, like 3, 5, 7, 9, 11, ---, 127 will be difficult. In the present work all partial products corresponding to different digits Diin RB form are generated by only adding a S-group digit multiplied partial product(SGDMPP) with T-group digit multiplied partial product(TGDMPP). The SGDMPP are ZERO, X, 2X, 3X, 4X, 5X, 6X, 7X, 8X and the TGDMPP are ZERO, 16X, 32X, 48X, 64X, 80X, 96X, 112X, 128X. All SGDMPP and TGDMPP coefficients are obtained from the nine bits of the digit Di using the algorithm given below

Algorithm:

The following is the algorithm to find the fundamental Sgroup and T-group digits to represent a digit Di. Assume the digit D0 is made of lower eight bits of coefficient and an overlapping bit C7C6C5C4C3C2C1C0C-1Here S_group and T_group are temporary variables to store intermediate value. The S_digit and T_digit are the final digit values

$$S.group = 4 \times C_2 + 2 \times C_1 + 1 \times C_0 + 1 \times C_{-1}$$
$$T.group = 16 \times (4 \times C_6 + 2 \times C_5 + 1 \times C_4 + 1 \times C_3)$$

```
If C7= 0 and C3= 0
Then S_digit = S_group and T_digit = T_group
else if C7= 0 and C3= 1
Then S_digit = S_group-8 and T_digit = T_group
else if C7= 1 and C3= 0
Then S_digit = S_group and T_ digit = T_group-128
else if C7= 1 and C3= 1
Then S_digit = S_group-8 and T_ digit = T_group-128
End
```

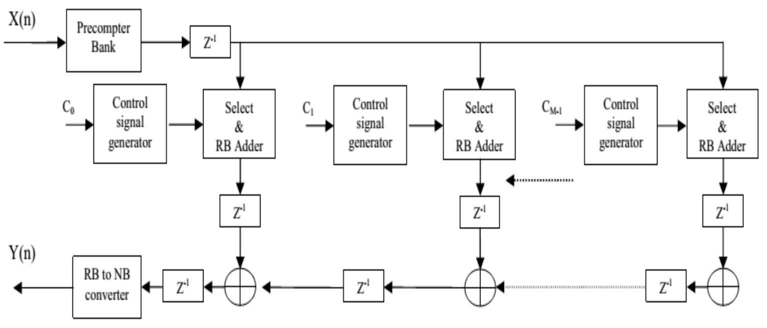
An example of multiplication using radix-256 encoding is given below.
Here inputs are X and Y, where
Y = 206 =00000000000000000000000011001110.
This coefficient can be grouped into four digits with D0= -50,
D1= 1, D2= 0, D3= 0.

Hence $Y = 2^{32}.D_3+2^{16}.D_2+2^{8}.\ D_1 + D_0$ and

$$Y.X = 2^{32}.\ (D_3.X)+2^{16}.\ (D_2.X)+2^{8}.\ (D_1.X) + (D_0.X)$$

Here D0.X can be obtained by adding -2X (the SGDMPP) with - 48X (the TGDMPP). This GDMPP can be obtained by shifting -X, 4-bit position. Similarly the TGDMPP can be obtained by shifting -3X by 3-bit position. Similarly D1.X also can be obtained by selecting proper SGDMPP and TGDMPP and grouping them for addition. Finally D0.X, and D1.X shifted by 8-bit positions are added using RB adders.

Figure captions appear below the figure, are flush left, and are in lower case letters. When referring to a figure in the body of the text, the abbreviation "Fig." is used. Figures should be numbered in the order they appear in the text.



The architecture of improved FIR filter using CSHM multiplier with RB arithmetic and Radix-256 encoding is given in Figure. The input to this filter is X and the coefficients at each tap point are C0, C1, …,CM-1. Both input and the coefficient are assumed to be of 16-bits. Each 16-bit coefficient is grouped in to two digits D0 and D1. To generate a DMPP, D0.X or D1.X, a SGDMPP need to be subtracted from a TGDMPP. The SGDMPP and TGDMPP are obtained from FDMPP. The FDMPP are generated using a precomputer stage. To select proper SGDMPP and TGDMPP from all FDMPPs, control signals are generated using control signal generator from the bits of digit Di. The select and add unit consists of a select unit and a RB adder. The selector selects proper SGDMPP and TGDMPP from FDMPPs, which forms the DMPP in RB form. The two DMPPs, D0X and D1X shifted by 8-bit positions are added using the RB adder in select and add unit to get coefficient multiplication with input. The coefficient multiplication results from each tap point are added using RB adders as shown. As external interface understands NB number, finally the FIR filter output in RB form can be converted into NB form using RB to NB converter. The detailed circuit and architecture of different units are discussed in this section.

Kogge-Stone prex tree is among the type of prex trees that use the fewest logic levels. A 16-bit example is shown in Figure 3.8. In fact, Kogge-Stone is a member of Knowles prex tree 3.10. The 16-bit prex tree can be viewed as Knowels [1,1,1,1]. The numbers in the brackets represent the maximum branch fan-out at each logic level. The maximum fan-out is 2 in all logic levels for all width Kogge-Stone prex trees. The key of building a prex tree is how to implement Equation (3.2) according to the specific features of that type of prex tree and apply the rules described in the previous section. Gray cells are inserted similar to black cells except that the gray cells nal output carry outs instead of intermediate G=P group. The reason of starting with Kogge-Stone prex tree is that it is the easiest to build in terms of using a program concept. The example in Figure 3.8 is 16-bit (a power of 2) prex tree. It is not difcult to extend the structure to any width if the basics are strictly followed.

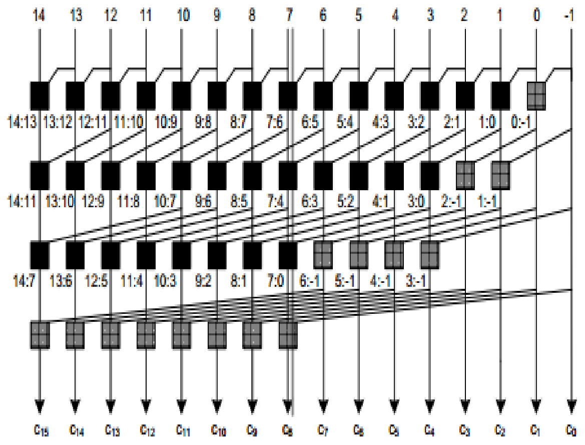
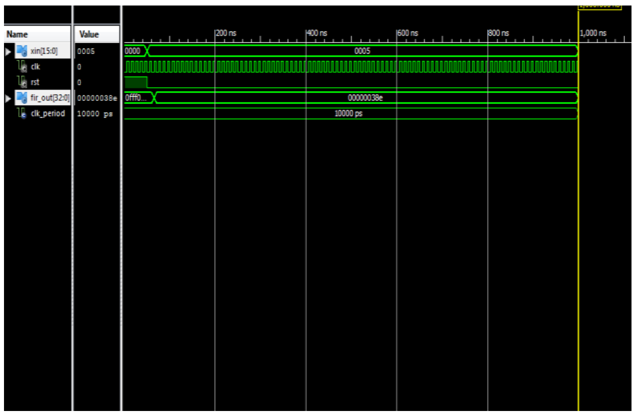


Figure 3.8: 16-bit Kogge-Stone Prefix Tree.

## III. Result



| Name | Value |
|---|---|
| xin[15:0] | 0005 |
| clk | 0 |
| rst | 0 |
| fir_out[32:0] | 0000038e |
| clk_period | 10000 ps |

## IV. Conclusion

An FIR filter based on NPGHB and prefixadder is proposed. The use of radix-256 reduces the number of partial product rows in any multiplication by 8 fold. Thus a multiplication of 16-bit input results only two partial product rows in prefixadder form. The obtained partial products in prefixadder form are added using prefixadder adder, which is much faster as no carry propagation occurs. The proposed multiplication method based on NPGHRB for FIR filter is found to be faster

## REFERENCES

[1] Y. C. Lim, S. R. Parker, and A. G. Constantinides, "Finite word length FIR filter design using integer programming over a discrete coefficient space," IEEE Trans. Acoustics, Speech Signal Processing,vol. ASSP-30, pp. 661–664, Aug. 1982.

[2] Y. C. Lim and S. R. Parker, "FIR filter design over a discrete power-oftwo coefficient space," IEEE Trans. Acoustics, Speech Signal Processing, vol. ASSP-31, pp. 583–591, June 1983

[3] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filter with powers-of-two coefficients," IEEE Trans. Circuits Syst., vol. 36, pp. 1044–1047, July 1989

[4] Hai Huyen Dam, Cantoni A., Kok Lay Teo, Nordholm S.,"FIR Variable Digital Filter With Signed Power-of-Two Coefficients," IEEE Transactions on Circuits and Systems I, Volume: 54 , Issue: 6, pp. 1348 - 1357, June 2007.

[5] Mustafa Aktan, Arda Yurdakul, and Günhan Dündar,"An Algorithm for the Design of Low-Power Hardware-Efficient FIR Filters," IEEE Transactions on Circuits and Systems I, VOL. 55, NO. 6, pp. 1536-1545, JULY 2008

[6] N. Sankarayya, K. Roy, and D. Bhattacharya, "Algorithms for low power and high speed FIR filter realization using differential coefficients," IEEE Trans. Circuits Syst., vol. 44, pp. 488–497, June 1997.

[7] Vinod A.P., Singla A., Chang C.H., "Low-power differential coefficients-based FIR filters using hardware-optimised multipliers, " IET, Circuits, Devices & Systems, Volume: 1 Issue:1, pp.13-20, February 2007.

[8] Dong Shi, Ya Jun Yu, "Design of Discrete-Valued Linear Phase FIR Filters in Cascade Form," IEEE Transactions on Circuits and Systems I, Volume: 58 , No. 7, pp. 1627 - 1636, July 2011.

[9] Jongsun Park, Woopyo Jeong, Hamid Mahmoodi-Meimand, Yongtao Wang, Hunsoo Choo, Kaushik Roy., Computation sharingprogrammable fir filter for low-power and high-performance applications. IEEE Journal of Solid-State Circuits. 2004, 39: 348-357.

[10] Avizienis, "Signed-digit number representation for fast parallel arithmetic," IRE Trans. Electron. Computer., vol.EC-10, pp. 389-400, Sept.1961.

[11] Hiroshi Makino,Yasunobu Nakase, Hiroaki Suzuki, Hiroyuki Morinaka, Hirofumi Shinohara, and Koichiro Mashiko, "An 8.8-ns 54 x 54-Bit multiplier with high speed redundant binary architecture," IEEE J. Of Solid State Circuits, Vol.31, No.6, pp. 773-783, June 1996.

[12] N. Takagi, H. Yasura and S. Yajima., "High-speed VLSI multiplication algorithm with a redundant binary addition tree," IEEE Transactions on Computers, C-34(9 ): 217-220, Sept.1985.

[13] S. Kuninobu, T. Nishiyama, T. Edamatsu, T. Taniguchi, and N. Takagi, "Design of high speed MOS multiplier and divider using redundant binary representation," In Proc. 8th Symp. Computer Arithmetic, Italy, pp. 80-86, May 1987.

[14] Yum Kim, Bang-Sup Song, John Grosspietsch, and Steven F. Gilling , " A carry-free 54b x 54b multiplier using equivalent bit conversion algorithm, " IEEE J. Of Solid State Circuits, Vol. 36, No.10, pp. 1538-1544, Oct. 2001.

[15] A.D. Booth, "A signed binary multiplication technique," Quarterly J. Mech. Appl. Math, Vol. 4, Part 2, pp. 236-240, 1951.

[16] J. M. Rabaey, Digital Integrated Circuits: A Design Perspective. Englewood Cliffs, NJ: Prentice-Hall, 1996..

[17] Y. Harata, Y. Nakamura, H. Nagese, M. Takigawa, and N. Takagi, "A high-speed multiplier using a redundant binary adder tree," IEEE J. Solid-State Circuits, Vol. 22, pp. 28-34, Feb. 1987.

[18] 90-nano meter silicon process technology library, UMC