ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 2, Issue 11, November 2013

Design and Implementation of Low Power Digital FIR Filter Based on Configurable Booth Multiplier

K.Prasanthi, G.V.K.S.Prasad, K.Swarajya Lakshmi

Abstract— **In this paper, an FIR filter using configurable booth multiplier (CBM) is proposed. Generally an FIR filter design requires many complex computations which effect the performance of the common digital signal processors (DSPs) in terms of power, speed, cost, area etc. Direct form fixed point FIR filter realization consists of different modules such as multiplier, adder etc. The low power consumption quality of configurable booth multiplier makes it a preferred choice in designing different circuits. It supports single 16-bit, single 8-bit or twin parallel 8-bit multiplication operations to be performed. To efficiently reduce the power consumption, a novel dynamic range detector is developed to dynamically detect the effective dynamic range of two operands. The detection results are used not only to pick the operand with small dynamic range for booth encoding to increase the probability of partial products becoming zero but also deactivate the redundant switching activities in ineffective ranges as much as possible. Moreover, the output product of the proposed configurable booth multiplier can be truncated to further decrease power consumption by sacrificing a bit of output precision. Dadda compression technique is used to compress the partial products and to produce final product. A carry look ahead adder is used to add the multiplier outputs to generate the FIR filter response. All these are implemented using VHDL.**

Index Terms— **Low power, booth multiplier, configurable multiplication, truncation, partially guarded computation.**

I. INTRODUCTION

The multimedia and portable communication devices have experienced explosive growth recently. Longer battery life is one of the crucial factors in the widespread success of these products. As such low power circuit design for multimedia and wireless communication application has become very important, Finite impulse response (FIR) filters are widely used in various DSP applications. In some applications, the FIR filter circuit must be able to operate at high sample rates, while in other applications, the FIR filter circuit must be a low-power circuit operating at moderate sample rates. The low-power or low-area techniques developed specifically for digital filters can be found in parallel processing. Parallel (or block) processing can be applied to digital FIR filters to either increase the effective throughput or reduce the power consumption of the original filter. The topology of the multiplier circuit also affects the resultant power consumption. Choosing multipliers with more hardware breadth rather than depth would not only reduce the delay, but also the total power consumption [1]. A lot of design methods of low power digital FIR filter are proposed, for example, in[2] implementing FIR filters using just registered adders and hardwired shifts. In [3] proposed a novel approach for a design method of a low power digital baseband processing. Their approach is to optimize the bit width of each filter coefficient. The later approach presents the method to reduce dynamic switching power of a FIR filter using data transition power diminution technique (DPDT). This technique is used on adders, booth multipliers. The booth multiplier has two stages. In the first stage, the partial products are generated by the booth encoder and the Partial Product Generator (PPG), and are summed by compressors. In the second stage, the two final products are added to form the final product through a final adder. In the proposed method the dynamic switching power is reduced by using configurable booth multiplier. In this multiplier configuration, partially guarded computation and the truncation techniques are combined. Applying these techniques by using dynamic range detector and some additional components like error vector (EV) generator, correcting-vector (CV) generator, an adjustor, a sign bit (SB) generator and a sign extension unit etc. The rest of the paper is structured as follows. Section2 gives a brief summary of FIR filter theory and Section3 presents the architecture of the booth multiplier in our implementation. Architecture of the Configurable booth multiplier in our implementation is given at section4 .Finally section5 provides the conclusion of this paper.

II. FIR FILTER THEORY

Digital filters are typically used to modify the attributes of a signal in the time or frequency domain. The most common digital filter is the linear time-invariant (LTI) filter. An LTI interacts with its input signal through a process called linear convolution, denoted by $y = f * x$ where f is the filter's impulse

.
K.Prasanthi,*Final MTech Student, Dept of Electronics & Communication Engineering,AKRG College of Engineering & Technology,Nallajerla, W.G.Dist., Prasanthi.konerru@gmail.com*
G.V.K.S.Prasad, *Asst.Professor. Dept of Electronics & Communication Engineering, AKRG College of Engineering & Technology, Nallajerla, W.G.Dist.,gvksp24@gmail.com*
K.Swarajya Lakshmi, *Asst.Professor. Dept of Electronics & Communication Engineering,VKR, VNB&AGK College of Engineering & Technology,Gudivada, KrishnaDist., Swarajlakshmi1983@gmail.com*

response, *x* is the input signal, and *y* is the convolved output. The linear convolution process is formally defined by:

$$Y[n] = x[n]*f[n] = \sum_{k=0}^{\infty} x[n]f\,[n\text{-}k] = \sum_{k=0}^{\infty} f[k]x\,[n\text{-}k]. \qquad (1)$$

An FIR filter consists of a finite number of sample values, reducing the above convolution sum to a finite sum per output sample instant.

An FIR with constant coefficients is an LTI digital filter. The output of an FIR of order or length L, to an input time sequence x[n], is given by a finite version of the convolution sum which is given by

$$Y[n] = x[n]*f[n] = \sum_{k=0}^{L-1} f[k]x\,[n\text{-}k], \qquad (2)$$

Where f [0] ≠ 0 through f [L -1] ≠ 0 are the filter's L coefficients. They also correspond to the FIR's impulse response. For LTI systems it is sometimes more convenient to express in the z-domain as

$$Y\,(z) = \sum F\,(z)\,X\,(z), \qquad (3)$$

Where F (z) is the FIR's transfer function defined in the z domain is

$$F\,(z) = \sum_{k=0}^{L-1} f[k]z^{-k} \qquad (4)$$

The direct form L^th^-order LTI FIR filter is graphically represented in Fig.1.

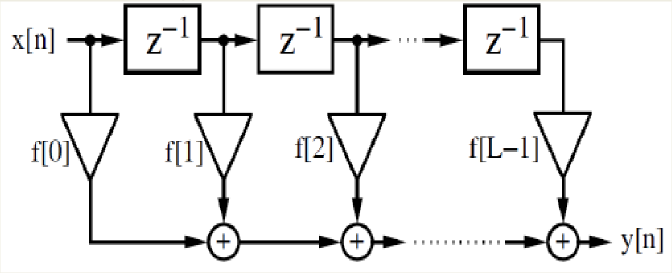


Fig. 1 FIR filter direct form structure.

It consist of a collection of a "tapped delay line," adders, and multipliers. One of the operands present at each multiplier is an FIR coefficient, often referred to as a "tap weight".

III. IMPLEMENTATION OF FIR FILTER

A. EXISTING METHOD: MAC FIR Filter Based on Booth Multiplier

In the MAC FIR filter design based on booth multiplier the multiplier has two stages. In the first stage, the partial products are generated by the booth encoder and the partial product generator (PPG), and are summed by compressors. In the second stage, the two final products are added to form the final product through a final adder.

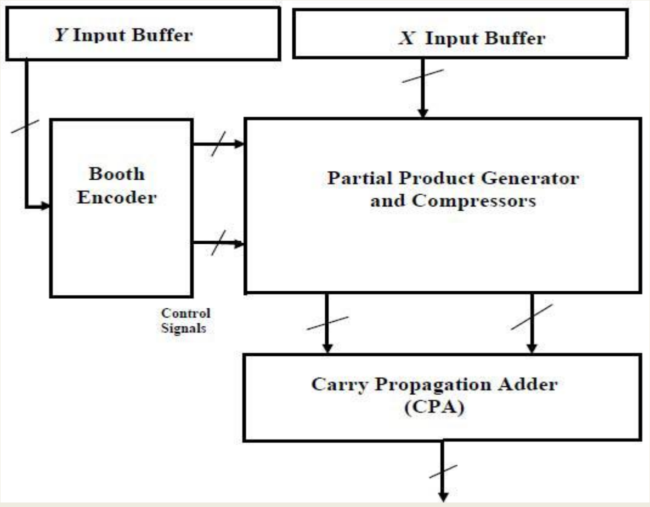
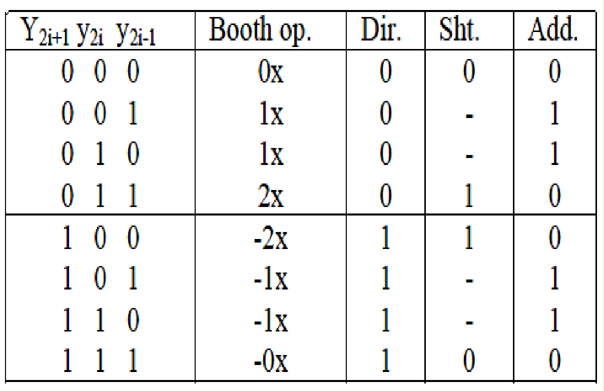

Y Input Buffer
X Input Buffer
Booth Encoder
Partial Product Generator and Compressors
Control Signals
Carry Propagation Adder (CPA)

Fig. 2 Block Diagram of Multiplier Architecture

It employs a booth encoder block, compression blocks, and an adder block. X and Y are the input buffers. Y is the multiplier which is recorded by the booth encoder and X is the multiplicand. PPG module and compressor form the major part of the multiplier. Carry propagation adder (CPA) is the final adder used to merge the sum and carry vector from the Compressor module. For radix-4 recoding, the popular algorithm is parallel recoding or Modified Booth recoding. In parallel radix-4 recoding, Y becomes:

$$Y = \sum_{i=0}^{n/2-1} vi4i = \sum_{i=0}^{n/2-1} \left(-2y_{2i+1} + y_{2i} + y_{2i-1}\right)4^{i} \qquad (5)$$

Table I Truth Table for Booth Encoding



Y_{2i+1} y_{2i} y_{2i-1}	Booth op.	Dir.	Sht.	Add.
0 0 0	0x	0	0	0
0 0 1	1x	0	-	1
0 1 0	1x	0	-	1
0 1 1	2x	0	1	0
1 0 0	-2x	1	1	0
1 0 1	-1x	1	-	1
1 1 0	-1x	1	-	1
1 1 1	-0x	1	0	0

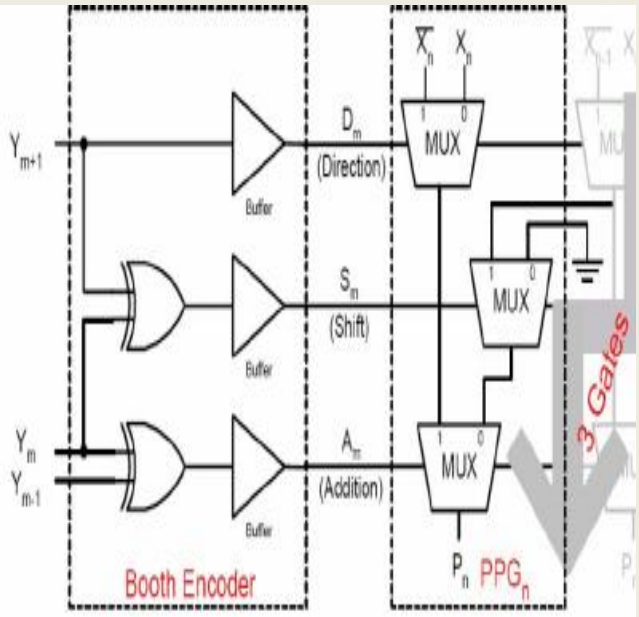


Fig. 3 Booth encoder and PP (m=2i)

The table I described Booth function as three basic operations, which they called ‘direction’, ‘shift’, and ‘addition’ operation.

Direction determined whether the multiplicand was positive or negative, shift explained whether the multiplication operation involved shifting or not and addition meant whether the multiplicand was added to partial products.

The Booth encoder was implemented using two XOR gates and the selector using 3MUXes and an inverter. Careful optimization of the partial-product generation can lead to some substantial delay and hardware reduction. In the normal 8*8 multiplication 8 partial products need to be generated and accumulated. For accumulation seven adders are required but to reduce power in the case of booth multiplier only 4 partial products are required to be generated and for accumulation three adders, reduced delay required to compute partial sum which then reduces the power consumption.

B. PROPOSED METHOD: MAC FIR Filter Based on Configurable Booth Multiplier

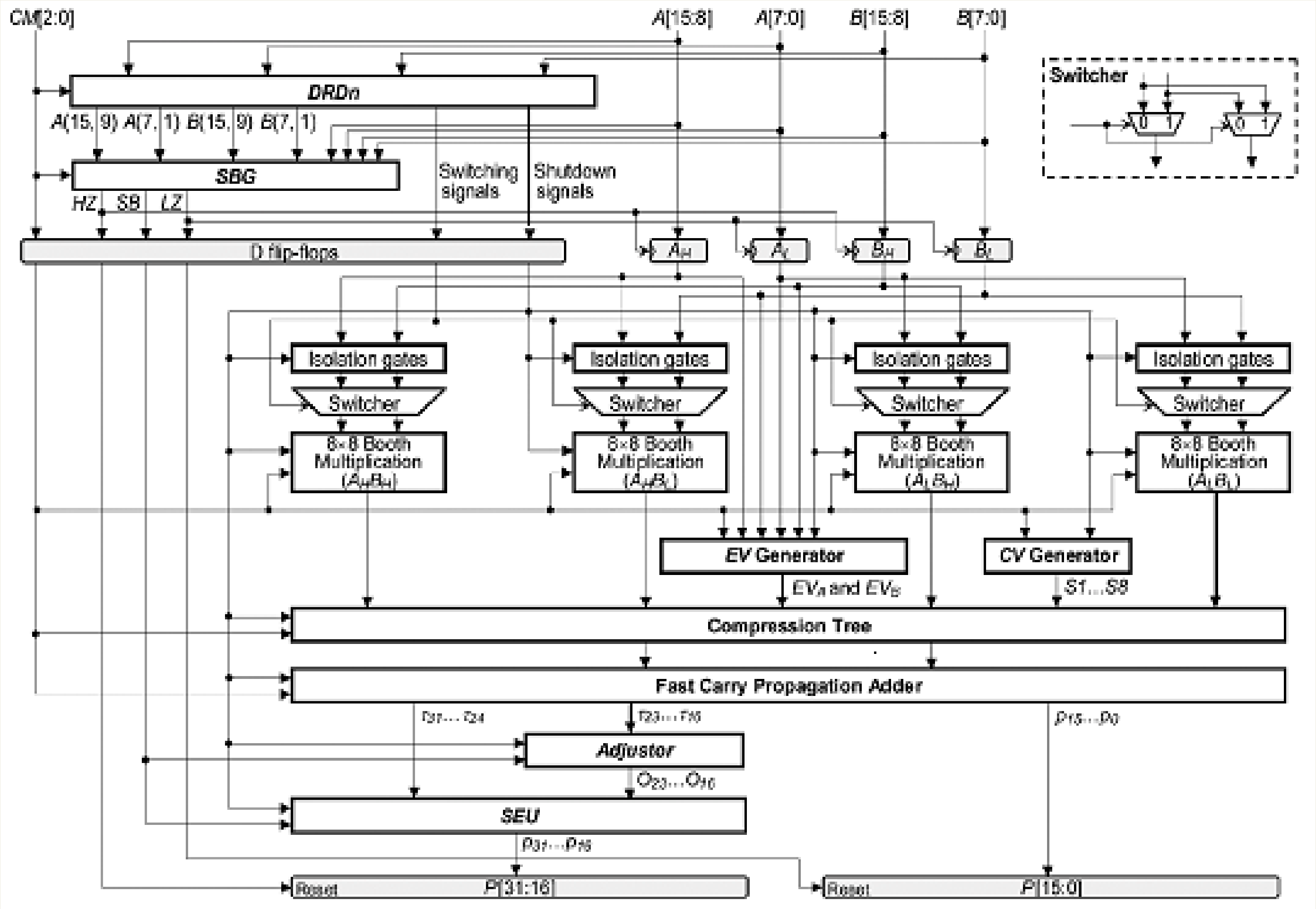


Fig. 4 Block Diagram of the Configurable Booth Multiplier (CBM)

In the MAC FIR filter design based on configurable booth multiplier, the multiplier is combined with partially guarded computation and truncation technique. The main concerns are speed, power efficiency and structural flexibility. The proposed multiplier not only performs single 16-bit , single 8-bit or twin parallel 8-bit multiplication operations but also offer a flexible tradeoff between output accuracy and power consumption to achieve more power savings. The configuration signals CM [2:0] in fig 4 are utilized to configure the operation of the proposed multiplier into 6 modes as shown in table II.

Table II Modes of the Proposed Configurable Multiplier

MODE	CM[2:0]	FUNCTION
1	111	Single 16-bit multiplication without truncation
2	110	Single 16-bit multiplication with truncation
3	101	Single 8-bit multiplication without truncation
4	100	Single 8-bit multiplication with truncation
5	001	Two parallel 8-bit multiplication without truncation
6	000	Two parallel 8-bit multiplication with truncation

When CM [2:1] =11 or 10, the single 16 bit or single 8 bit multiplication operation is performed. On the other hand two parallel 8 bit multiplication operations that satisfy the high throughput requirement is carried out if CM[2:1]=00.

More power saving is obtained for n bit multiplication by disabling the computations of the n-1 least significant product bits if CM [0] = 0. In this case the partial products which produce the n-1 least significant product bits are set to zero and error compensation values are added to the n most significant product bits to reduce the product error. As mentioned earlier in section 1FIR Filter using CBM consists of Dynamic range detector, Adjustor, etc., are discussed below.

1) *DYNAMIC RANGE DETECTOR (DRD)*

Dynamic range detector is one of the building blocks and which plays a major role in configurable booth multiplier. The dynamic range detector is made up of switching logic and shutdown logic to significantly decrease the incorrect judgment and the power consumption by properly exchanging the input operands and shutting down the unused functional blocks based on the multiplication mode and the effective range of the input operands.

2) *SWITCHING LOGIC*

The configuration signals CM[2:0] and the input operands A[15:0] and B[15:0] are applied to the switching logic circuit as shown in fig 5 which then generates the switching signals SW_{HH}, SW_{HL}, SW_{LH}, SW_{LL} .These signals are used for each 8 bit booth multiplication to pick the operands that leads more partial products to zero for booth encoding.

The proposed switching logic for four 8-bit booth multiplication operations whose input operands are A [15:8], B [15:8], A [7:0], B [7:0]. The input operands are portioned first into 3 bit groups and are then fed to the 3-input comparators to obtain more fine grained comparison results. If the output of a comparator is one it indicates that the input 3-bit group is successive 0’s or 1’s so that its booth encoded product will be zero. The proposed switching logic can aid in detecting the length of the sign extension bits of the input operands and is used to determine which operand is a multiplier.

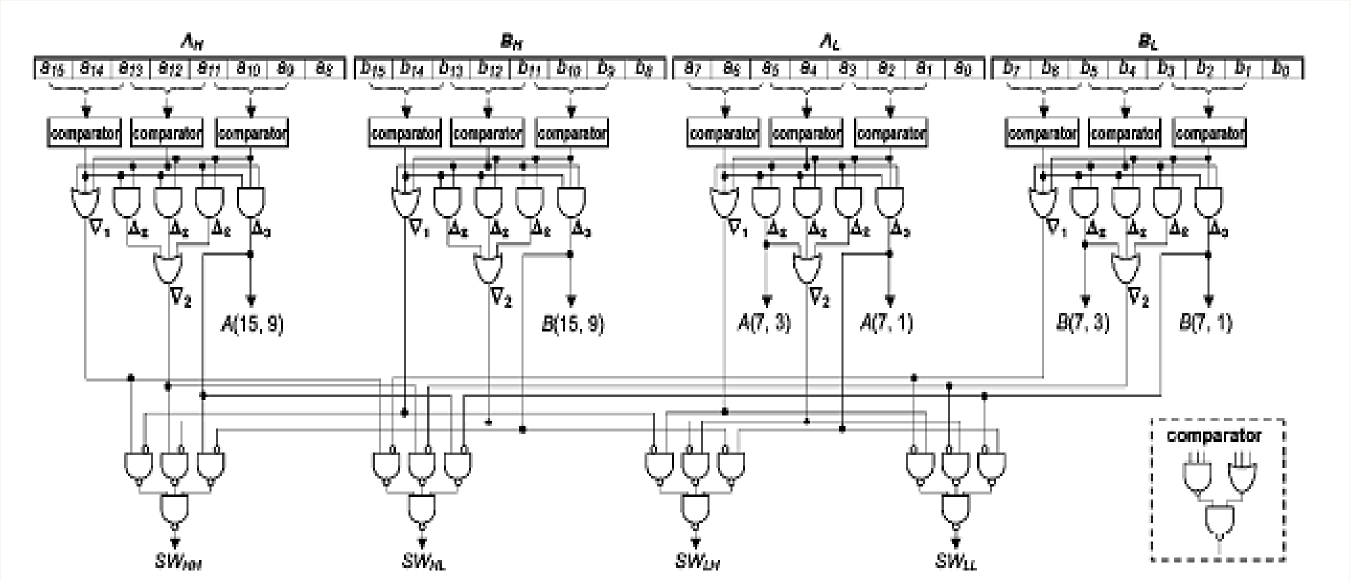

A_H
B_H
A_L
B_L
a_{15} a_{14} a_{13} a_{12} a_{11} a_{10} a_9 a_8
b_{15} b_{14} b_{13} b_{12} b_{11} b_{10} b_9 b_8
a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0
b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0
comparator
∇_1 Δ_2 Δ_2 Δ_2 Δ_3
∇_2
A(15, 9)
B(15, 9)
A(7, 3)
A(7, 1)
B(7, 3)
B(7, 1)
SW_{HH}
SW_{HL}
SW_{LH}
SW_{LL}
comparator

Fig 5 Switching Logic

3) SHUTDOWN LOGIC

Given the multiplication mode and the effective range of the input operands produces signals SD_{HH}, SD_{HL}, SDLH, SD_{LL} to individually shutdown A_HB_H, AHBL.ALBH and A_LB_L multiplications as shown in fig 6. When CM[2:1]=00, only the multiplication operations A_HB_H and A_LB_L need to be performed and others can be shut down by forcing their input operands and sign bit Δ to 0. If CM [2:0] =110 the computation of A_LB_L multiplication should be disabled to achieve truncation.

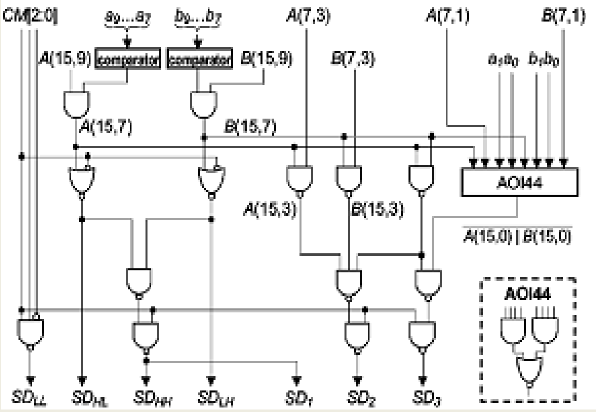


Fig 6 Shutdown Logic

To avoid unnecessary sign extension computation in single 16-bit multiplication, 3-predetermined guarded boundary positions GP_1=8, GP_2=12 and GP_3=16 are selected. The shutdown logic circuits also generate SD_1, SD_2 and SD_3. SD_1 is identical with SD_{HH}. In addition, SD1 must be zero when SD_2=0, and SD_2 must be zero when SD_3=0. Therefore, when SD_1=0, A_HB_H multiplication is disabled and the output product bits (guarded boundary position GP_1) P [31:24] are replaced by an SB directly. When SD_2=0 the partial product bits from bit position 23 to 20 are forced to zero and the output product bits (guarded boundary position GP_2) P [23:20] are replaced by an SB directly. When SD_3=0, the partial product bits from bit position 19 to 16 are set to zero and the output product bits (guarded boundary position GP_3) P [19:16] must be replaced by an SB directly.

4) EV and CV generator

EV (Error Vector) and CV (Correcting Vector) must be generated and fed into the compression tree to ensure that the final product P is corrected. EV is obtained by summing up EV_A and EV_B which are generated from A and B as shown in fig 7. Table III lists the corresponding EV_A and EV_B for different operation situations of the proposed multiplier, where X denotes "don't care".

The required CV for the proposed multiplier also relies on the multiplication mode and the effective range of the input operands. Table IV lists all possible correcting vectors for different operation situation of the proposed multiplier. Note that CV for CM [2:1] =00 is equal to $AB00AB00_{16}$ because two independent 8 bit multiplication operations are performed in parallel. To simplify the CV generator all possible correcting vectors in table IV are represented as 2's complement binary numbers as shown in table V, and the bit positions that have the same values for each situation are identified and classified into 8 groups denoted as S1,S2,. . . .S8. All bit positions in the same group can be generated by the same circuit and the final CV generator is shown in fig 8.

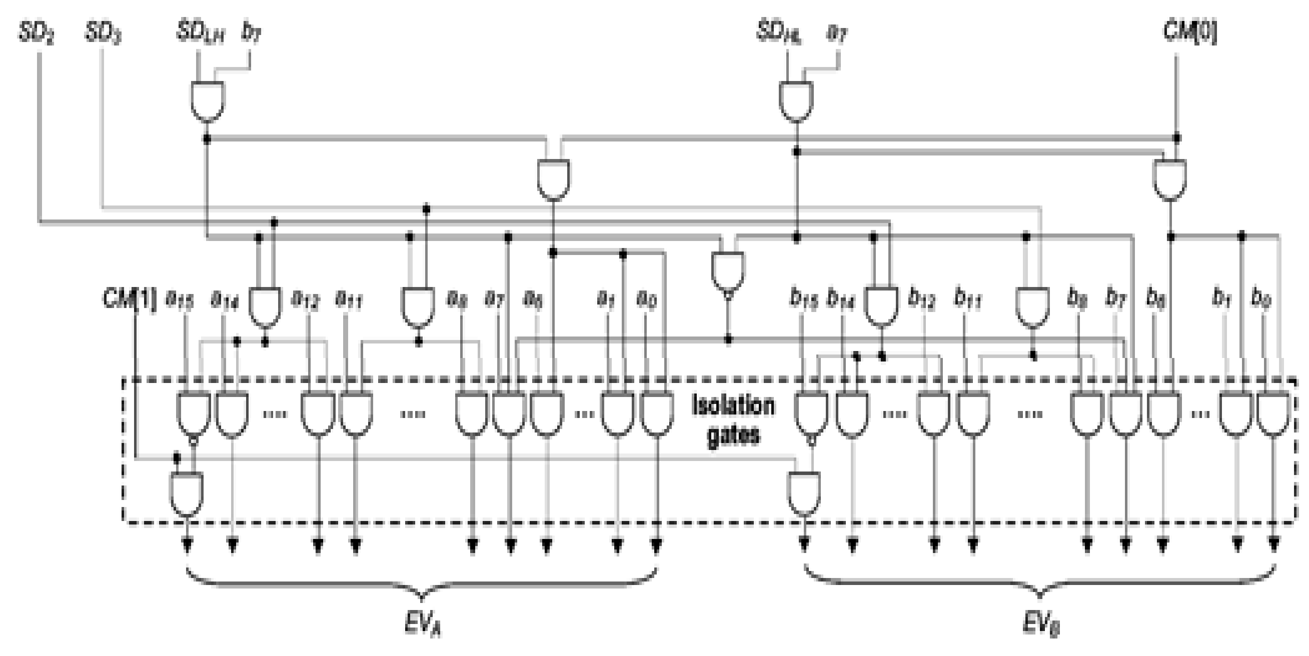


Fig 7 EV Generator

Table III All Proposed Values of the EV Multiplier

CM[1]	CM[0]	a_7	b_7	SD_2	SD_3	SD_{HL}	SD_{LH}	EV_A	EV_B
0	X	0	0	X	X	X	X	00…..0	00…..0
1	X	0	0	X	X	X	X	10…..0	10…..0
0	X	X	0	X	X	0	X	00…..0	00…..0
1	X	X	0	X	X	0	X	10…..0	10…..0
0	X	0	X	X	X	X	0	00…..0	00…..0
1	X	0	X	X	X	X	0	10…..0	10…..0
0	X	X	X	X	X	0	0	00…..0	00…..0
1	X	X	X	X	X	0	0	10…..0	10…..0
1	1	X	1	0	0	0	1	10…0a_7…a_0	10…..0
1	1	1	X	0	0	1	0	10…..0	10…0b_7…b_0
1	1	X	1	0	1	0	1	10...a_{11}…. a_0	10…..0
1	1	1	X	0	1	1	0	10…..0	10...b_{11}…. b_0
1	1	X	1	1	1	0	1	$\overline{a_{15}}$ a_{14}…. a_0	10…..0
1	1	1	X	1	1	1	0	10…..0	$\overline{b_{15}}$ b_{14}…. b_0
1	1	0	1	1	1	1	1	$\overline{a_{15}}$ a_{14}…. a_0	10…..0
1	1	1	0	1	1	1	1	10…..0	$\overline{b_{15}}$ b_{14}…. b_0
1	1	1	1	1	1	1	1	$\overline{a_{15}}$ a_{14}…. a_0	$\overline{b_{15}}$ b_{14}…. b_0

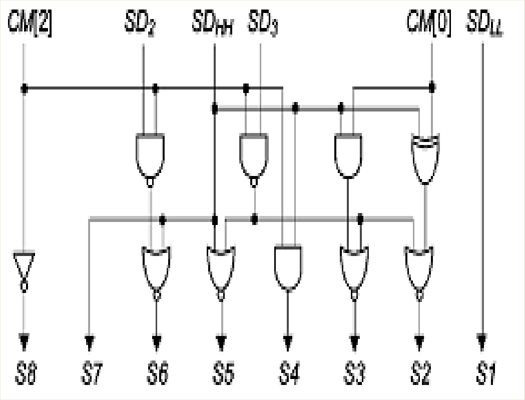
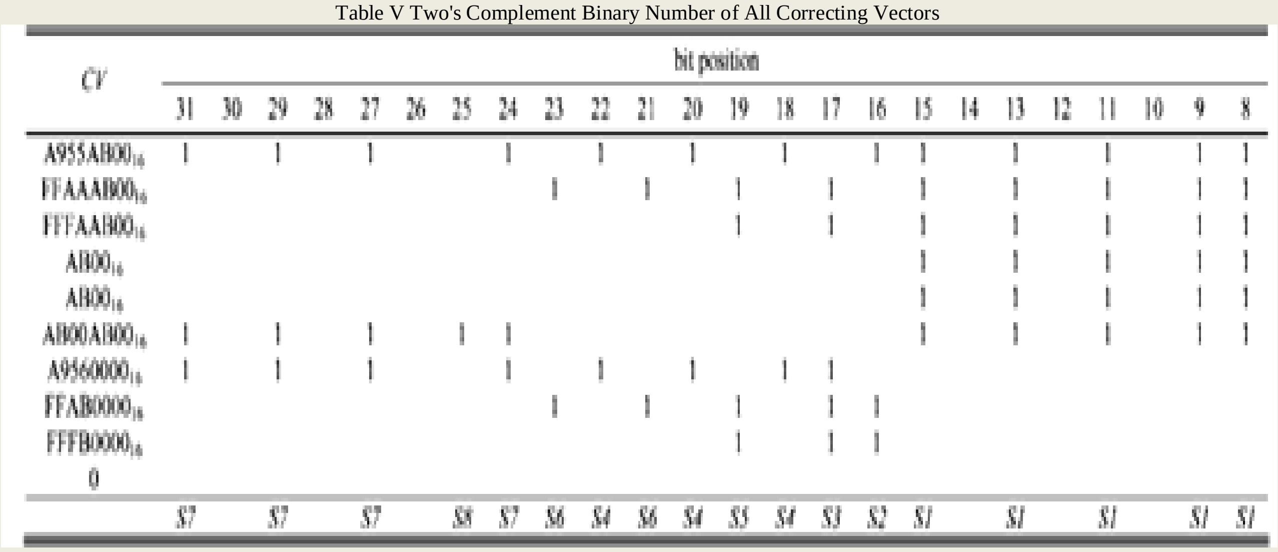


Fig 8 CV Generator

Table IV Correcting Vectors of The Proposed Multiplier

CM[2:0]	SD_{HH}	SD_{LL}	SD_2	SD_3	CV
111	1	1	1	1	$A955AB00_{16}$
111	0	1	1	1	$\theta_2+\theta_4$(or $\theta_3+\theta_4$)
111	0	1	0	1	$-2^{18}-2^{16}+\theta_4$
111	0	1	0	0	θ_4
10X	0	1	0	0	θ_4
00X	1	1	1	1	$AB00AB00_{16}$
110	1	0	1	1	$\theta_1+\theta_2+\theta_3+\theta_4$
110	0	0	1	1	θ_2 (or θ_3)
110	0	0	0	1	$-2^{18}-2^{16}$
110	0	0	0	0	0

Table V Two's Complement Binary Number of All Correcting Vectors



CV	bit position																							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
$A955AB00_{16}$	1		1		1			1		1		1		1		1	1		1		1		1	1
$FFAAAB00_{16}$									1		1		1		1		1		1		1		1	1
$FFFAAB00_{16}$													1		1		1		1		1		1	1
$AB00_{16}$																	1		1		1		1	1
$AB00_{16}$																	1		1		1		1	1
$AB00AB00_{16}$	1		1		1		1	1									1		1		1		1	1
$A9560000_{16}$	1		1		1			1		1		1		1	1									
$FFAB0000_{16}$									1		1		1		1	1								
$FFFB0000_{16}$													1		1	1								
0																								
	S7		S7		S7		S8	S7	S6	S4	S6	S4	S5	S4	S3	S2	S1		S1		S1		S1	S1

5) ADJUSTOR

If truncation and partially guarded computations are performed a large product errors occur. If sign extension part of product P is replaced by the SB generated from SBG the adjustor shown in fig 9 is developed to overcome the problem of large product error. The large product error is avoided by forcing the output bits from O23 to O16 to be 1, if SB =1, SD_1=0 and (τ23| τ22|….| τ16) =0. Similarly the output bits from O19 to O16 are forced to be 1 when SB=1, SD_2=0 and (τ19| τ18|….| τ16) =0.

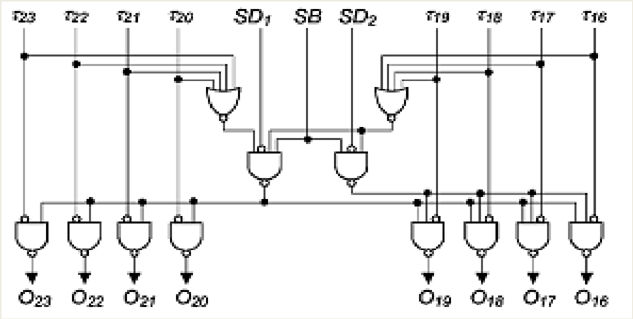


Fig 9 Adjustor

6) SBG And SEU

In partially guarded computation, the sign extension bit of product P is replaced by an SB to avoid unnecessary sign extension computations. Moreover, if one of the input operand is zero(clock gating [7] and[8]) the entire operation of the configurable multiplier can be shut down to obtain more power savings by preventing input registers from loading new data and directly resetting the output registers to zero. Therefore we develop an SBG to generate an SB and shutdown the entire multiplier when one of the input operands is zero. Fig 10 shows the proposed SBG, where HPZ =0 indicates that at least one input operand of A_HB_H multiplication is equal to zero. Where LPZ=0 indicates that at least one input operand of A_LB_L multiplication is equal to zero, and PZ=0 indicates that at least one input operand A and B is equal to zero. HZ and LZ are generated based on the following principles where HZ=0 will disable the clock signals of input registers A_H and B_H and reset the output register P[31:16], where LZ=0 will disable the clock signals of input registers A_L and B_L and reset the output register P[15:0].

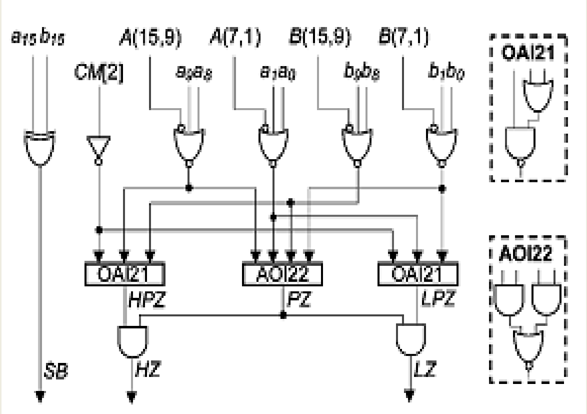


Fig 10 SBG

The SEU consists of 16 multiplexers as shown in fig11 is utilized to directly assign sign extension bits to the output product P to avoid the redundant sign extension computation on the left side of GP_1,GP_2 or $GP_{3.}$ Through using shutdown signals SD_1, SD_2 and SD_3, the SB is selected through multiplexers.

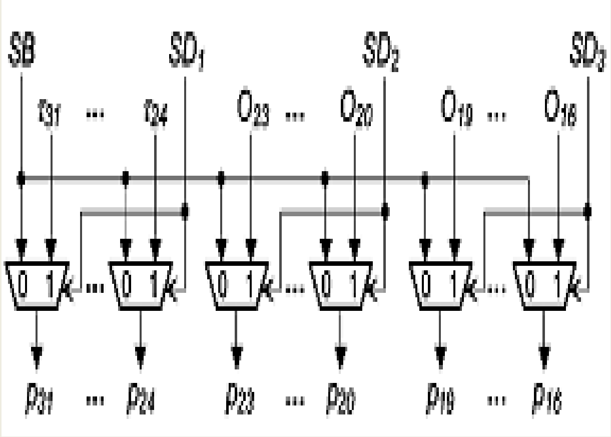


Fig 11 SEU

7) 16 Bit Multiplication Matrix

The total multiplication expression is divided into four sub expressions by using divide and conquer method. Then the expression is modified as A_HB_H,A_LB_L,A_HB_L,A_LB_H,EV_A,EV_B and CV as shown in fig 12 , where A_H means A[15:8] and A_L means A[7:0] and similarly for operand B. Four independent partial product arrays are produced by using radix booth-encoding approach. The partial products generated from all the individual blocks are grouped as shown in fig 12 to obtain the final product using adders and compressors.

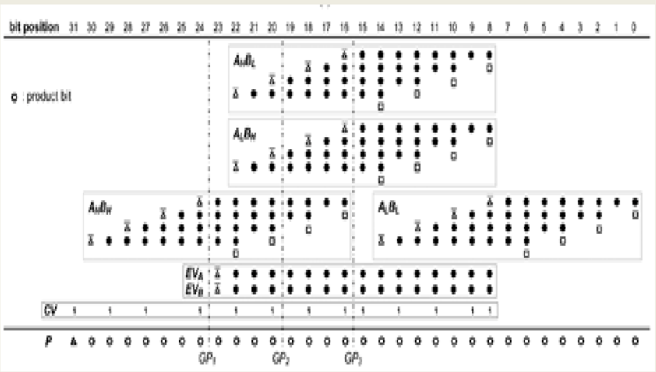

bit position 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
○ : product bit
AHBL
ALBH
AHBH
ALBL
EVA
EVB
CV
P
GP1
GP2
GP3

Fig 12 16-bit multiplication matrix

8) Dadda Compression Technique

The partial products can be effectively reduced using Dadda tree compression techniques. In the compression algorithm each and every partial product is combined in groups of three and compressed in groups of 2 using full adder which is the 3:2 compressor. This process will be continued until all the partial products along with their carries are compressed. Thus the number of stages and the delay in those stages are reduced effectively using Dadda tree [9] compression technique. The Dadda multiplier is usually faster and smaller than Wallace tree multiplier and hence it is preferred.

9) Carry Lookahead Adder

The carry look ahead adder (CLA) solves the carry delay problem by calculating the carry signals in advance, based on the input signals. In FIR filter multiple bit carry look ahead adders are used to add two 32-bit numbers which accumulates the multiplication of the present product with the previous product.

IV. RESULTS

The FIR filter using ordinary Radix4 booth encoding and the proposed FIR filter using CBM for n=16 are designed in VHDL and synthesized by using Xilinx ISE 8.2i.The implementation results in table VI shows the comparison between proposed FIR filter using CBM and the FIR filter using ordinary Radix4 booth encoding in terms of power consumption, no of slices, no of LUT's and latches.

Table VI. Comparison between Both FIR Filters

Device Utilization & Power Optimization	PREVIOUS	PRESENT
Power Consumption(mW)	1021.40	1006.08
Slices Used	3255	2870
LUT's Used	2540	2117
Latches Used	3054	2047

V.CONCLUSION

An FIR filter using configurable booth multiplier has been designed which provides a flexible arithmetic capacity and a tradeoff between output precision and power consumption. Moreover, the ineffective circuitry can be efficiently deactivated, thereby reducing power consumption and increasing speed of operation.

The experimental results have shown that the proposed FIR filter using CBM outperforms the conventional FIR filter using ordinary RADIX-4 booth multiplier in terms of power and speed of operation with enough accuracy at the expense of extra area.

REFERENCES

[1]AHMED F. SHALASH, KESHAB K. PARHI ”Power Efficient Folding of Pipelined LMS Adaptive Filters with Applications” Journal of VLSI Signal Processing, pp. 199–213, 2000.

[2]Shahnam Mirzaei, Anup Hosangadi, Ryan Kastner, “FPGA Implementation of High Speed FIR Filters Using Add and Shift Method”, IEEE, 2006.

[3]Kousuke TARUMI, Akihiko HYODO, Masanori Muroyama, Hiroto YASUURA, “A design method for a low power digital FIR filter”in digital wireless communication systems,” 2004.

[4] N.-Y. Shen and O. T.-C. Chen, “Low-power multipliers by minimizing switching activities of partial products,” in Proc. IEEE Int.Symp. Circuits Syst., May 2002, vol. 4, pp. 93–96.

[5] M. J. Schulte and E. E. Swartzlander Jr., “Truncated multiplication with correction constant,” in Proc. Workshop VLSI Signal Process., Oct.1993, pp. 388–396.

[6] Shiann-Rong Kuang and Jiun-Ping Wang “Design of power efficient configurable booth multiplier” IEEE Trans. Circuits Syst. IRegular Papers vol. 57, no.3, pp. 568-580, March 2010.

[7] T. Kitahara, F. Minami, T. Ueda, K. Usami, S. Nishio, M. Murakata and T. Mitsuhashi, “A clock -gating method for low power LSI design,” in Proc. Int. Symp. Low Power Electron. Des, Feb. 1998, pp. 07–312.

[8] X. Chang, M. Zhang, G. Zhang, Z. Zhang, and J. Wang, “Adaptive clock gating technique for low power IP core in SOC design,” inProc. IEEE Int. Symp. Circuits Syst., May 2007, pp. 2120–2123.

[9] L. Dadda, “Some schemes for parallel multipliers,” Alta Freq., vol. 34, pp. 349–356, May 1965.

[10]J.Park, S.Kim and Y.S.Lee” A low power booth multiplier using novel data partition method”in proc.IEEE Asia-pacific conf.Adv.Syst.Integr.Circuits, Aug.2004, pp 54-57.