Title: Design Algorithm For Low Complexity Reconfigurable Fir Filter Based On Edbns.

Volume 07, Issue 02, Page No: 334 – 339.

Paper Authors

***MORTHA VARAPRASAD, P NAGA RAJU.**

* Dept of ECE, Kakinada Institute of Engineering &Technology.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

DESIGN ALGORITHM FOR LOW COMPLEXITY RECONFIGURABLE FIR FILTER BASED ON EDBNS.

*MORTHA VARAPRASAD,** P NAGA RAJU
*M.Tech Scholar, Dept of ECE. Kakinada Institute of Engineering &Technology, East Godavari Dist., AP,India.
**Assoc.Prof, Dept of ECE, Kakinada Institute of Engineering &Technology, East Godavari Dist., AP,India.

ABSTRACT:
Coefficient multipliers are the stumbling blocks present in programmable finite impulse response (FIR) digital filters. As the filter coefficients change either dynamically or periodically, the search for common sub expressions for multiplier less implementation needs to be performed over the entire gamut of integers of the desired precision, and the amount of shifts associated with each identified common sub expression needs to be memorized. The complexity of a quality search is thus beyond the existing design algorithms based on conventional binary and signed digit representations. A new design paradigm for the programmable FIR filters by exploiting the extended double base number system (EDBNS). Due to its sparsity and innate abstraction of the sum of binary shifted partial products, the sharing of adders in the time-multiplexed multiple constant multiplication blocks of the programmable FIR filters can be maximized by a direct mapping from the quasi-minimum EDBNS. The multiplexing cost can be further reduced by merging double base terms. In this, power is reduced by using modified booth encoding algorithm. Partial products generation stage is optimized by using Radix8 modified booth encoding algorithm.
Keywords: Extended Double Base Number System (EDBNS), Product of b generation (POBG), Finite Impulse Response (FIR).

I. INTRODUCTION

Filter is a frequency selective network. It passes a band of frequencies while attenuating the others. Filters are classified as analog and digital depending on nature of inputs and outputs. Filters are further classified as finite impulse response and infinite impulse response filters depending on impulse response. This chapter gives a brief about the types of filters. Digital filters are used extensively in all areas of electronic industry. This is because digital filters have the potential to attain much better signal to noise ratios than analog filters and at each intermediate stage the analog filter adds more noise to the signal, the digital filter performs noiseless mathematical operations at each intermediate step in the transform. The digital filters have emerged as a strong option for removing noise, shaping spectrum, and minimizing inter-symbol interference in communication architectures. These filters have become popular because their precise reproducibility allows design engineers to achieve performance levels that are difficult to obtain with analog Filters Digital Filters can be constructed from 3 fundamental mathematical operations.

- Addition (or subtraction)
- Multiplication (normally of a signal by a constant)
- Time Delay i.e. delaying a digital signal by one or more sample periods

Multipliers are key components of many high performance systems such as FIR filters, microprocessors, digital signal processors, etc. A system’s performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest clement in the system. Furthermore, it is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. As a result, whole spectrums of multipliers with different area-speed constraints are designed with fully parallel processing. In between are digit serial multipliers where single digits

consisting of several bits are operated on. These multipliers have moderate performance in both speed and area. However, existing digit serial multipliers have been plagued by complicated switching systems and/or irregularities in design. Radix 2^n multipliers which operate on digits in a parallel fashion instead of bits bring the pipelining to the digit level and avoid most of the above problems. They were introduced by M. K. Ibrahim in 1993.
These structures are iterative and modular. The pipelining done at the digit level brings the benefit of constant operation speed irrespective of the size of' the multiplier. The clock speed is only determined by the digit size which is already fixed before the design is implemented.

$$Y[n]=b0x[n]+b1x[n-1]+b2x[n-2]+b3x[n-3]$$

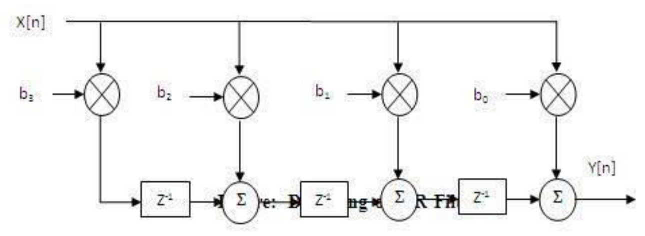
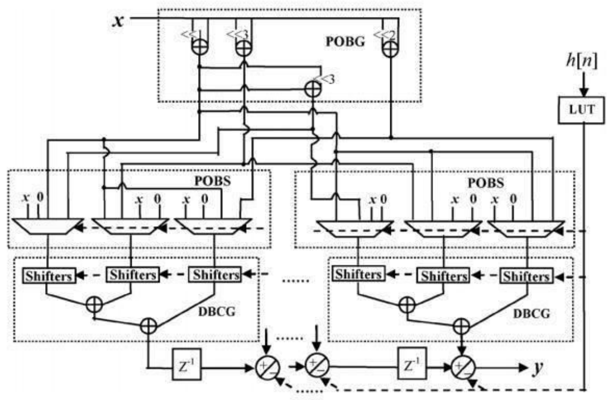


Figure 2: Designing of FIR Filter

In electronics, an adder or summer is a digital circuit that performs addition of numbers. In many computers and other kinds of processors, adders are used not only in the arithmetic logic unit(s), but also in other parts of the processor, where they are used to calculate addresses, table indices, and similar operations. Although adders can be constructed for many numerical representations, such as binary-coded decimal orexcess-3, the most common adders operate on binary numbers. In cases where two's complement or ones' complement is being used to represent negative numbers. We can view a full adder as a 3:2 loss compressor: it sums three one-bit inputs, and returns the result as a single two-bit number; that is, it maps 8 input values to 4 output values. Thus, for example, a binary input of 101results in an output of 1+0+1=10 (decimal number '2'). The carry-out represents bit one of the results, while the sum represents bit zero. Likewise, a half adder can be used as a 2:2 loss compressor, compressing four possible inputs into three possible outputs. Such compressors can be used to speed up the summation of three or more addends. If the addends are exactly three, the layout is known as the carry-save adder. If the addends are four or more, more than one layer of compressors is necessary and there is various possible designs for the circuit. The most common are Wallace trees. This kind of circuit is most notably used in multipliers, which is why these circuits are also known as Wallace multipliers. Minimization of POBS by EDBNS Reduction Properties and Exponential Diophantine Equation (EDE): Each of the POBS blocks consists of a bank of multiplexers with inputs feeding from the POBG. The products of every power-of-integers and the input signal from the POBG are routed to the inputs of these multiplexers according to their frequencies of occurrences in the EDBNS representation of the coefficient. The same partial product may appear in the inputs of several multiplexers of the same POBS block if it occurs more than once in the EDBNS representation of the same coefficient. The implementation cost of a POBS block is determined by the number of multiplexers and the complexity of each multiplexer. The complexity of a multiplexer is approximately proportional to its number of inputs and the bit width of the inputs. Therefore, one way to reduce the hardware cost of a POBS block is to minimize the total number of input lines to the multiplexers. Although has been constrained as a whole in the generation of the EDBNS representations for all –bit coefficients, the EDBNS representations of some coefficients may have fewer than terms. Therefore, some elements in do not have to be connected to the inputs of all multiplexers. Further reduction in the total number of inputs to the multiplexers of a POBS block can be achieved with the help of the following two reduction rules [35] for DBNS. The second property is also valid for of EDBNS.



8-bit programmable FIR filter designed by proposed algorithm

Step 1) Compute and for all -bit coefficients. Generate the EDBNS array for all the -bit coeffi- cients using the search algorithm.

Step 2) Implement the POBG block by producing all power-of- integers in using the method

Step 3) Design the POBS block with multiplexers. Each power-of- integer from the POBG block is first connected to an input of different multiplexers, where is the maximum number of times that power-of- value can appear in the EDBNS representation of any coefficient. Then, minimize the number of input lines to the multiplexers of POBS block by the algorithm presented in above figure.

Step 4) Design programmable shifters for the DBCG block. Extract the amount of shifts for each power-of- integer and store it in the LUT addressable by the fundamentals.

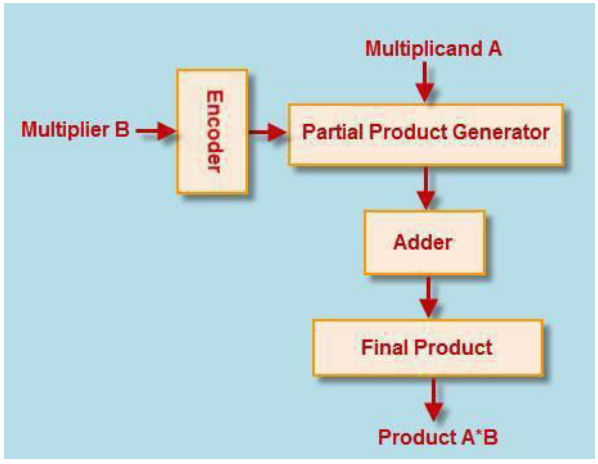
Step 5) Sum the double base terms in DBCG by a carry save adder (CSA) tree to reduce the delay.

II. DOUBLE BASE NUMBER SYSTEM:

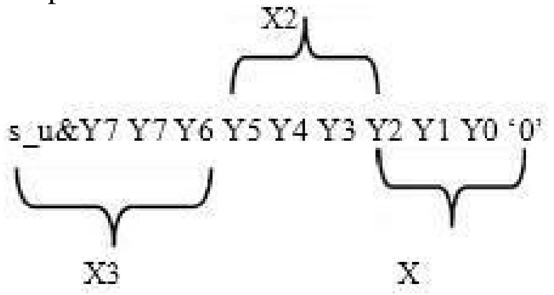
Double base number system (DBNS) is an alternative number system besides the binary system. Its representation is similar to the radix number system together with two bases, usually be two and three. DBNS preserves the two important properties: redundancy and sparseness. The redundancy is the property accommodating with the parallelism. In this research, we are interested in parallel addition algorithm on DBNS. Our theoretical result shows that parallel addition in DBNS can be performed

RADIX-8 MODIFIED BOOTH ALGORITHM:

Booth's algorithm involves repeatedly adding one of two predetermined values to a product P, and then performing a rightward arithmetic shift on P.



Multiplier architecture comprise of two architectures, i.e., Modified Booth. Based on the study of various multiplier architectures, we find that Modified Booth increases the speed because it reduces partial products to half. Further, the delay in multiplier can be reduced by using Wallace tree. Power consumption of Wallace tree multiplier is also less as compared to booth and array. Features of both multipliers can be combined to produce high speed and low power multiplier. Modified Booth multiplier consists of Modified Booth Recorder (MBR). MBR have two parts, i.e., Booth Encoder (BE) and Booth Selector (BS). The basic operation of BE is to decode the multiplier signal and output will be used by BS to generate the partial product. The partial products are then, added with the Wallace tree adders, similar to the carry save adder approach. The last row of carry and sum output is added together by carry look- ahead adder with the carry skewed to the left by position. Radix-8 Booth encoding is most often used to avoid variable size partial product arrays. Before designing Radix-8 BE, the multiplier has to be converted into a Radix-8 number by dividing them into four digits respectively according to Booth Encoder Table given after wards. Prior to convert the multiplier, a zero is appended into the Least Significant Bit (LSB) of the multiplier.

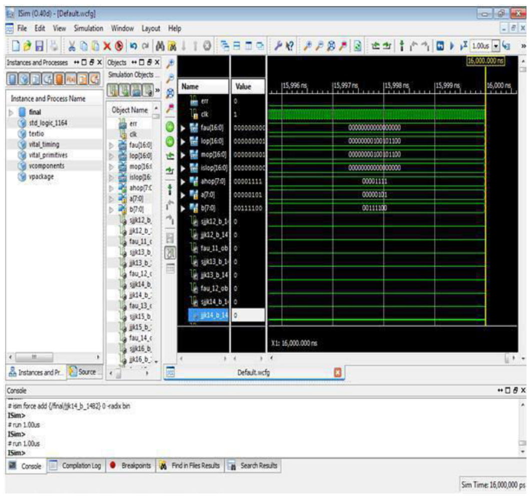


Radix 8 Booth recoding applies the same algorithm as that of Radix 4, but now we take quartets of bits instead of triplets. Each quartet is codified as a signed digit using below Table. Radix 8 algorithm reduces the number of partial products to n/3, where n is the number of multiplier bit s. Thus it allows a ti me gain in the partial products summation Radix-8 recoding applies the same algorithm as radix-4, but now we take quartets of bits instead of triplets. Each quartet is codified as a signeddigit using the table

Quartet value	*Signed-digit value*
0000	0
0001	+1
0010	+1
0011	+2
0100	+2
0101	+3
0110	+3
0111	+4
1000	-4
1001	-3
1010	-3
1011	-2
1100	-2
1101	-1
1110	-1
1111	0

Here we have an odd multiple of the multiplicand, 3Y, which is not immediately available. To: generate it we need to perform this previous add:2Y+Y=3Y. But we are designing a multiplier for specific purpose and thereby the multiplicand belongs to a previously known set of numbers which are stored in a memory chip. We have tried to take advantage of this fact, to ease the bottleneck of the radix-8 architecture, that is, the generation of 3Y. In this manner we try to attain a better overall multiplication time, or at least comparable to the time we could obtain using a radix-4 architecture (with the additional advantage of using a less number of transistors). To generate 3Y with 21-bit words we only have to add 2Y+Y, that is, to add the number with the same number shifted one position to the left. A product formed by multiplying the multiplicand by one digit of the multiplier when the multiplier has more than one digit. Partial products are used as intermediate steps in calculating larger products. Partial product generator is designed to produce the product by multiplying the multiplicand A by 0, 1, -1, 2, - 2,-3,-4, 3, 4. For product generator, multiply by zero means the multiplicand is multiplied by "0".Multiply by "1" means the product still remains the same as the multiplicand value. Multiply by "-1" means that the product is the two's complement form of the number. Multiply by "-2" is to shift left one bit the two's complement of the multiplicand value and multiply by "2" means just shift left the multiplicand by one place. Multiply by "-4" is to shift left two bit the two's complement of the multiplicand value and multiply by "2" means just shift left the multiplicand by two place. Here we have an odd multiple of the multiplicand, 3Y, which is not immediately available. To generate it we need to perform this previous add: 2Y+Y=3Y. But we are designing a multiplier for specific purpose and thereby the multiplicand belongs to a previously known set of numbers which are stored in a memory chip. We have tried to take advantage of this fact, to ease the bottleneck of the radix-8 architecture, that is, the generation of 3Y. In this manner we try to attain a better overall multiplication time, or at least comparable to the time we could obtain using radix-4 architecture (with the additional advantage of using a less number of transistors). To generate 3Y with 8-bit words we only have to add 2Y+Y, that is, to add the number with the same number shifted one position to the left.

III. RESULTS



```
# isim force add {/final/jjk14_b_1482} 0 -radix bin
ISim>
# run 1.00us
ISim>
# run 1.00us
ISim>
```

Figure 6: Experimental Result

IV. CONCLUSION

Design methodologies aiming at minimizing their implementation cost have been intensively studied by many researchers. This paper presents a radically different approach to this problem for the minimization of the TMMCM block of programmable FIR digital filters based on the extended form of DBNS. An algorithm for the generation of quasi-minimum EDBNS has been proposed. The obtained EDBNS can be directly mapped to an efficient TM-MCM architecture. Further, this is enhanced by using radix-8 modified booth encoding algorithm for improvement architecture.

V. REFERENCES

[1] A. G. Dempster and M. D. Macleod, "Use of minimumadder multiplier blocks in FIR digital filters," IEEE Trans. Circuits Syst. II, AnalogDigit. Signal Process., vol. 42, no. 9, pp. 569–577, Sep. 1995.
[2] Y. Wang and K. Roy, "CSDC: A new complexity reduction technique for multiplierless implementation of digital FIR filters," IEEE Trans.Circuits Syst. I, Reg Papers, vol. 52, no. 9, pp. 1845–1853, Sep. 2005.
[3] C. Y. Yao, W. C. Hsia, and Y. H. Ho, "Designing hardware-efficient fixed-point FIR filters in an expanding subexpression space," IEEETrans. Circuits Syst. I, Reg Papers, vol. 61, no. 1, pp. 202–212, Jan. 2014.
[4] Y. Pan and P. K. Meher, "Bit-level optimization of addertrees for multiple constant multiplications for efficient FIR filter implementation," IEEE Trans. Circuits Syst. I, Reg Papers, vol. 61, no. 2, pp. 455–462, Feb. 2014.
[5] J. Park, W. Jeong, H. M. Meimand, Y. Wang, H. Choo, and K. Roy, "Computation sharing programmable FIR filter for low-power and high-performance applications," IEEE J. Solid-State Circuits, vol. 39, no. 2, pp. 348–357, Feb. 2004.
[6] E. Grayver and B. Daneshrad, "Low power, area efficient programmable filter and variable rate decimator," in Proc. IEEE Int.Symp. Circuits Syst., Geneva, Switzerland, May 2000, pp. 341–344.
[7] S. S. Demirsoy, R. Beck, A. G. Dempster, and I. Kale, "Reconfigurable implementation of recursive DCT kernels for reduced quantization noise," in Proc. IEEE Int. Symp. Circuits Syst., Bangkok, decimator," in Proc. IEEE Int.Symp. Circuits Syst., Geneva, Switzerland, May 2000, pp. 341–344.
[8] J. Chen and C. H. Chang, "High-level synthesis algorithm for the design of reconfigurable constant multiplier," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 28, no. 12, pp. 1844–1856, Dec. 2009.
[9] R. Mahesh and A. P. Vinod, "Reconfigurable low area complexity filter bank architecture based on frequency response masking for nonuniform channelization in software radio receivers," IEEE Trans. Aerosp. Electron. Syst., vol. 47, no. 2, pp. 1241–1255, Apr. 2011.
[10] T. Chen, Y. V. Zakharow, and C. Liu, "Low-complexity channel-estimate based adaptive linear equalizer," IEEE Signal Process. Lett., vol. 18, no. 7, pp. 427–430, Jul. 2011.
[11] I. Hautala, J. Boutellier, and J. Hannuksela, "Programmable lowpower implementation of the HEVC adaptive loop filter," in IEEE Int. Conf. Acous., Speech,

Signal Process., Vancouver, BC, Canada, May 2013, pp. 2664–2668.
[12] E. J. Norberg, R. S. Guzzon, J. S. Parker, L. A. Johansson, and L. A. Coldren, "Programmable photonic microwave filters monolithically integerated in InP-InGaAsP," J. Lightwave Technol., vol. 29, no. 11, pp. 1611–1619, Jun. 2011.