

Reconfigurable Fir Digital Filter Realization on FPGA

Atmakuri Vasavi¹

Sita Madhuri Bondila²

¹PG Student (M.Tech), Dept. of ECE, Gandhiji Institute of Science & Tech., Jaggaiahpetta, AP, India

²Assistant professor, Dept. of ECE, Gandhiji Institute of Science & Tech., Jaggaiahpetta, AP, India

Abstract— This paper presents efficient distributed arithmetic (DA)-based approaches for high-throughput reconfigurable implementation of finite-impulse response (FIR) filters whose filter coefficients change during runtime. Conventionally, for reconfigurable DA-based implementation of FIR filter, the lookup tables (LUTs) are required to be implemented in RAM and the RAM-based LUT is found to be costly for ASIC implementation. Therefore, a shared-LUT design is proposed to realize the DA computation. Instead of using separate registers to store the possible results of partial inner products for DA processing of different bit positions, registers are shared by the DA units for bit slices of different weightage. The proposed design has nearly 68% and 58% less area-delay product and 78% and 59% less energy per sample than the DA-based systolic structure and the carry save adder (CSA)-based structure, respectively, for the ASIC implementation. A LUT, which stands for LookUp Table, in general terms is basically a table that determines what the output is for any given input(s). In the context of combinational logic, it is the truth table. This truth table effectively defines how your combinatorial logic behaves. In other words, whatever behaviour you get by interconnecting any number of gates (like AND, NOR, etc.), without feedback paths (to ensure it is state-less), can be implemented by a LUT.

Keywords—FIR, Digital Filter, Reconfigurable, FPGA

I. INTRODUCTION

In the majority of digital signal processing (DSP) applications the critical operations are the multiplication and accumulation. Real-time signal processing requires high speed and high throughput Multiplier-Accumulator (FIR) unit that consumes low power, which is always a key to achieve a high performance digital signal processing system. The purpose of this work is to design and implementation of a low power FIR unit with block enabling technique to save power. Firstly, a 1-bit FIR unit is designed, with appropriate geometries that give optimized power, area and delay. The delay in the pipeline stages in the FIR unit is estimated based on which a control unit is designed to control the data flow between the FIR blocks for low power. Similarly, the N-bit FIR unit is designed and controlled for low power using a control logic that enables the pipelined stages at appropriate time. The adder cell designed has advantage of high operational speed, small Gate count and low power.

In general, a multiplier uses Booth's algorithm and array of full adders (FAs), or Wallace tree instead of the array of FA's, i.e., this multiplier mainly consists of the three parts: Booth encoder, a tree to compress the partial products such as Wallace tree, and final adder. Because Wallace tree is to add the partial products from encoder as parallel as possible, its operation time is proportional to, where is the number of inputs. It uses the fact that counting the number of 1's among the inputs reduces the number of outputs into. In real implementation, many (3:2) or (7:3) counters are used to reduce the number of outputs in each pipeline step. The most effective way to increase the speed of a multiplier is to reduce the

number of the partial products because multiplication precedes a series of additions for the partial products. To reduce the number of calculation steps for the partial products, MBA algorithm has been applied mostly where Wallace tree has taken the role of increasing the speed to add the partial products. To increase the speed of the MBA algorithm, many parallel multiplication architectures have been researched. Among them, the architectures based on the Baugh-Wooley algorithm (BWA) have been developed and they have been applied to various digital filtering calculations.

One of the most advanced types of FIR for general-purpose digital signal processing has been proposed by Elguibaly. It is an architecture in which accumulation has been combined with the carry save adder (CSA) tree that compresses partial products. In the architecture proposed in , the critical path was reduced by eliminating the adder for accumulation and decreasing the number of input bits in the final adder. While it has a better performance because of the reduced critical path compared to the previous FIR architectures, there is a need to improve the output rate due to the use of the final adder results for accumulation. Architecture to merge the adder block to the accumulator register in the FIR operator was proposed in to provide the possibility of using two separate /2-bit adders instead of one -bit adder to accumulate the -bitFIR results. Recently, Zicari proposed an architecture that took a merging technique to fully utilize the 4-2 compressor. It also took this compressor as the basic building blocks for the multiplication circuit.

A new architecture for a high-speed FIR is proposed. In this FIR, the computations of multiplication and accumulation are combined and a

hybrid-type CSA structure is proposed to reduce the critical path and improve the output rate. It uses MBA algorithm based on 1's complement number system. A modified array structure for the sign bits is used to increase the density of the operands. A carry look-ahead adder (CLA) is inserted in the CSA tree to reduce the number of bits in the final adder. In addition, in order to increase the output rate by optimizing the pipeline efficiency, intermediate calculation results are accumulated in the form of sum and carry instead of the final adder outputs.

II. FIR FILTER USING DA

A four tap adaptive digital filter architecture using DA is shown in Fig.1, which is suitable only for small order filters. For this, the filter output is computed by (3) that use LUT to store and update the filter contents according to (4). DA FIR filters perform the filtering operation as per bit precision of the input signal sample irrespective of filter length. To reduce the memory and hardware requirements, updating of filter weights can be done without using any other memory element. The DA-F-LUT contains all possible combination sums of the filter weights which recalculated and updated according to the input signal sample and error signal. The filter architecture shown in Fig. 2 can be used for higher order filters by increasing the number of input taps.

The LMS algorithm is used to update the weights to minimize the error between filter output $y(n)$ and desired signal $d(n)$. To compute the new filter weights LMS algorithm uses the error signal in every iteration cycle. Thus, each recursion shifts the filter weights closer to their optimum value.

If $W(n) = [w_0(n) w_1(n) \dots w_{K-1}(n)]^T$ is the tap weight vector,

$X(n) = [x(n) x(n-1) \dots x(n-K+1)]^T$ the tap input vector during the n th iteration and k is the filter order then the weight updating equation for k th filter tap is given by

$$w_k(n+1) = w_k(n) + \mu e(n)x(n-k) \quad (4)$$

where $e(n)=d(n)-y(n)$ is the error value and μ is the step size

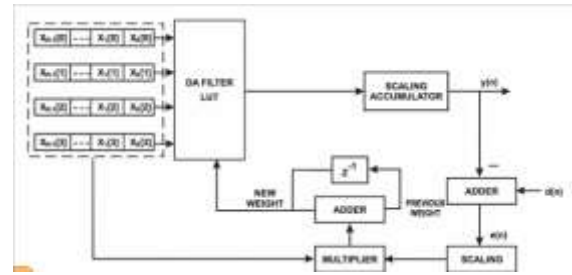


Figure 1 DA LMS adaptive filter using single LUT with one SA for 4 tap.

The weight computed at time n becomes the weight value at time $n+1$ by using the LMS weight adapting algorithm. Therefore, the value of filter weight at time $n+1$ is stored in LUT to perform the filtering operation according to the input data sample at that time instant. The DA-F-LUT $n+1$ is updated by reading the memory location DA-F-LUT[n] and by multiplying the input data sample by $ue[n]$ as

$$DA - F - LUT[n + 1] = DA - F - LUT[n] + \mu e[n]x[n - (k)]$$

III. RECONFIGURABLE FIR

A Reconfigurable finite-impulse response (FIR) filter whose filter coefficients dynamically change during runtime plays an important role in the software defined radio systems, multichannel filters, and digital up/down converters. However, the well-known multiple-constant multiplication-based technique, which is widely used for the implementation of FIR filters, cannot be used when the filter coefficients dynamically change. On the other hand, a general multiplier-based structure requires a large chip area and consequently enforces a limitation on the maximum possible order of the filter that can be realized for high-throughput applications.

A distributed arithmetic (DA)-based technique has gained substantial popularity in recent years for its high-throughput processing capability and increased regularity, which result in cost-effective and area-time efficient computing structures. The main operations required for DA-based computation are a sequence of lookup table (LUT) accesses followed by shift accumulation operations of the LUT output. The conventional DA implementation used for the implementation of an FIR filter assumes that impulse response coefficients are fixed, and this behavior makes it possible to use ROM-based LUTs. The memory requirement for DA-based implementation of FIR filters, however, exponentially increases with the filter order. To eliminate the problem of such a large memory requirement, systolic decomposition techniques are suggested by Meher et al. for DA-based implementation of long-length convolutions and FIR filter of large orders [7], [8]. For a reconfigurable DA-based FIR filter whose filter coefficients dynamically change, we need to use rewritable RAM based LUT

instead of ROM-based LUT. Another approach is to store the coefficients in the analog domain by using serial digital-to-analog converters resulting in mixed-signal architecture. We also find quite a few works on DA based implementation of adaptive filters [11], [12] where the coefficients change at every cycle. In this brief, we present efficient schemes for the optimized shared-LUT implementation of reconfigurable FIR filters using DA technique, where LUTs are shared by the DA units for bit slices of different weightage. In addition, the filter coefficients can be dynamically changed in runtime with a very small reconfiguration latency.

The output of an FIR filter of length N can be computed as an inner product of the impulse response vector $(h(k), \text{for } k = 0, 1, \dots, N - 1)$ and an input vector $(x(n - k), \text{for } k = 0, 1, \dots, N - 1)$, which is given by

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n - k). \tag{1}$$

For simplification of subsequent derivation, let us remove time index n as

$$y = \sum_{k=0}^{N-1} h(k)s(k) \tag{2}$$

where $s(k) = x(n - k)$. Assuming L to be the word length, the input sample $s(k)$ may be expressed in two's complement representation, i.e.,

$$s(k) = -[s(k)]_0 + \sum_{l=1}^{L-1} [s(k)]_l 2^{-l} \tag{3}$$

where $[s(k)]_l$ denotes the l th bit of $s(k)$. Substituting (3), we can write (2) in an expanded form, i.e.,

$$y = - \sum_{k=0}^{N-1} h(k) [s(k)]_0 + \sum_{k=0}^{N-1} h(k) \left\{ \sum_{l=1}^{L-1} [s(k)]_l 2^{-l} \right\} \tag{4}$$

To convert the sum-of-products form of inner product of (2) into a distributed form, the order of summations over the indices k and l in (4) can be interchanged to have

$$y = - \sum_{k=0}^{N-1} h(k) [s(k)]_0 + \sum_{l=1}^{L-1} 2^{-l} \left\{ \sum_{k=0}^{N-1} h(k) [s(k)]_l \right\} \tag{5}$$

and the inner product given by (5) can be computed as

$$y = \sum_{l=1}^{L-1} 2^{-l} C_l - C_0 \tag{6a}$$

Where

$$C_l = \sum_{k=0}^{N-1} h(k) [s(k)]_l. \tag{6b}$$

Since any element of the N -point bit sequence $\{s(k)\}$ for $0 \leq k \leq N - 1$ can either be 0 or 1, the partial sum C_l for $0 \leq l \leq L - 1$ can have $2N$ possible values. If all the $2N$ possible values of C_l are precomputed and stored in the LUT, the partial sums C_l can be read out from the LUT using the bit sequence $\{[s(k)]_l \text{ for } 0 \leq k \leq N - 1\}$ as address bits for computing the inner product.

Without a loss of generality, and for simplicity of discussion, we may assume the signal samples to be unsigned words of size L , although the proposed algorithm can be used for two's complement coding and offset binary coding also. We can always obtain unsigned input signal by adding fixed offset when the original input signal is signed. The inner product given by (6a) then can be expressed in a simpler form, i.e.,

$$y = \sum_{l=0}^{L-1} 2^{-l} C_l \tag{7}$$

so that no sign reversal of LUT output is required. We can use (7) directly for straight forward DA-based implementation of FIR filter using the LUT containing $2N$ possible values of C_l . For large values of N , however, the LUT size becomes too large, and the LUT access time also becomes large. The straightforward DA-based implementation is, therefore, not suitable for large filter orders. When N is a composite number given by $N = PM$ (P and M may be any two positive integers), one can map the index k into $(m + pM)$ for $m = 0, 1, \dots, M - 1$ and $p = 0, 1, \dots, P - 1$ to express (7) as

$$y = \sum_{l=0}^{L-1} 2^{-l} \left(\sum_{p=0}^{P-1} S_{l,p} \right) \tag{8a}$$

Where $S_{l,p}$ is the sum of partial product of M samples represented as

$$S_{l,p} = \sum_{m=0}^{M-1} h(m + pM) [s(m + pM)]_l \tag{8b}$$

for $l = 0, 1, \dots, L - 1$ and $p = 0, 1, \dots, P - 1$. For any given sequence of impulse response $\{h(k)\}$, the $2M$ possible values of $S_{l,p}$ corresponding to the $2M$ permutations of M -point bit sequence $\{[s(m + pM)]_l\}$, for $m = 0, 1, \dots, M - 1$ and $l = 0, 1, \dots, L - 1$, may be stored in the LUT of $2M$ words. These values of $S_{l,p}$ can be read out when the bit sequence is fed to the LUT as address. Equation (8) may, thus, be written in terms of memory-read operation as

$$y = \sum_{l=0}^{L-1} 2^{-l} \left[\sum_{p=0}^{P-1} \mathcal{F}(b_{l,p}) \right] \tag{9}$$

where $\mathcal{F}(b_{l,p}) = S_{l,p}$, and

$$b_{l,p} = \{[s(pM)]_l, [s(1 + pM)]_l, \dots, [s(M - 1 + pM)]_l\} \tag{10}$$

The proposed structure of the DA-based FIR filter for ASIC implementation is shown in Fig. 1. The input samples $\{x(n)\}$ arriving at every sampling instant are fed to a serial-in-parallel out shift register (SIPOSR) of size N . The SIPOSR decomposes the N recent most samples to P vectors b_p of length M for $p = 0, 1, \dots, P - 1$ and feeds them to P reconfigurable partial product generators (RPPGs) to calculate the partial products according to (8b). The structure of the proposed RPPG is depicted in Fig. 2 for $M = 2$. For high-throughput implementation, the RPPG generates L partial products corresponding to L bit slices in parallel using the LUT composed of a single register bank of $2M - 1$ registers and L number of $2M : 1$ MUX es. In the proposed structure, we reduce the storage consumption by sharing each LUT across L bit slices. The register array is preferred for this purpose rather than memory-based LUT in order to access the LUT contents simultaneously. In addition, the contents in the register-based LUT can be updated in parallel in fewer cycles than the memory-based LUT to implement desired FIR filter. The width of each register in the LUT is $(W + \lceil \log_2 M \rceil)$ bits, where W is the word length of the filter coefficient. The input of the MUX es are $0, h(2p), h(2p + 1),$ and $h(2p) + h(2p + 1)$; and the two-bit digit $b_{l,p}$ is fed to MUX l for $0 \leq l \leq L - 1$ as a control word. We can find that MUX l provides the partial product $S_{l,p}$ for $0 \leq l \leq L - 1$ given by (8b).

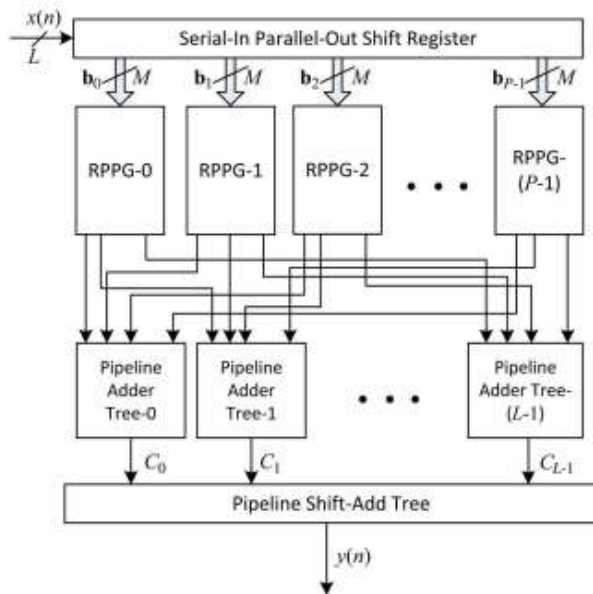


Figure 2: Proposed structure of the high-throughput DA-based FIR filter for ASIC implementation. RPPG stands for reconfigurable partial product generator.

The $(W + 1)$ -bit partial products generated by the P RPPG blocks are added by L separate pipeline adder trees (PATs) according to the inner summation in The output of PATs are appropriately shifted and added to obtain the filter output $y(n)$ by a pipeline shift-add tree (PSAT) as the outer summation . The PAT requires $P - 1$ adders in $\lceil \log_2 P \rceil$ stages and the PSAT requires $L - 1$ adders in $\lceil \log_2 L \rceil$ stages.

FPGA technology has tremendously grown from a dedicated hardware to a heterogeneous system, which is considered to be a popular choice in communication base stations instead of being just a prototype platform. The proposed reconfigurable FIR filter may be also implemented as part for the complete system on FPGA. Therefore, here we propose a reconfigurable DA based FIR filter for FPGA implementation. The architecture suggested in Section III for high-throughput implementation of DA-based FIR filter is not suitable for FPGA implementation. The structure in Fig. 1 involves $N(2M - 1)/M$ number of registers for the implementation of LUTs for FIR filter of length N . However, registers are scarce resource in FPGA since each LUT in many FPGA devices contains only two bits of registers. Therefore, the LUTs are required to be implemented by distributed RAM (DRAM) for FPGA implementation. However, unlike the case of the RPPG in Fig. 2, the multiple number of partial inner products $S_{l,p}$ cannot be retrieved from the DRAM simultaneously since only one LUT value can be read from the DRAM per cycle. Moreover, if L is the bit width of input, the duration of the sample period of the design is L times the operating clock period, which may not be suitable for the application requiring high throughput. Using a DRAM to implement LUT for each bit slice will lead to very high resource consumption.

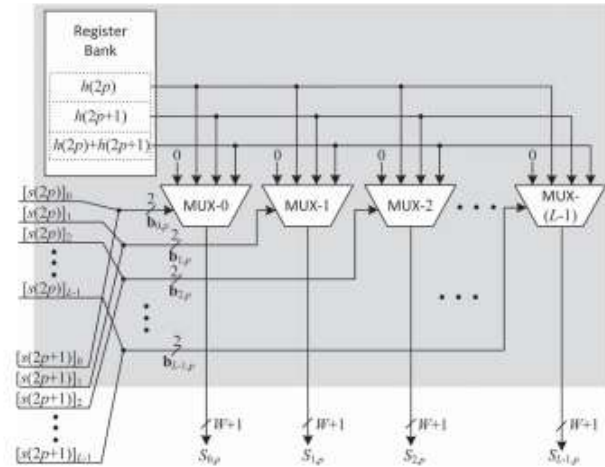


Figure 3: p th RPPG for $M = 2$.

Thus, we decompose the partial inner-product generator into Q parallel sections and each section has R time-multiplexed operations corresponding to R bit slices. When L is a composite number given by $L = RQ$ (R and Q are two positive integers), the index l in (8a) can be mapped into $(r + qR)$ for $r = 0, 1, \dots, R - 1$ and $q = 0, 1, \dots, Q - 1$ to modify (8a) as

$$y = \sum_{q=0}^{Q-1} 2^{-Rq} \left[\sum_{r=0}^{R-1} 2^{-r} \left(\sum_{p=0}^{P-1} S_{r+qR,p} \right) \right] \quad (11)$$

We have referred to the indices q and r in (11) as section index and time index, respectively. We have R time slots of the same duration as the operating clock period so that we can have one filter output every R cycles. Fig. 3(a) shows the structure of the proposed time-multiplexed DA-based FIR filter using DRAM. To implement (11), the proposed structure has Q sections, and each section consists of P DRAM-based DRPPGs (DRPPGs) and the PAT to calculate the rightmost summation, followed by shift-accumulator that performs over R cycles according to the second summation. However, we can use dual-port DRAM to reduce the total size of LUTs by half since two DRPPGs from two different sections can share the single DRAM. The structure of a DRPPG is shown in Fig. 3. The proposed structure can produce QP partial inner products in a single cycle, whereas the structure in Fig. 1 can generate LP inner products. In the r th cycle, P DRPPGs in the q th section generate P partial inner products $S_{r+qR,p}$ for $p = 0, 1, \dots, P - 1$ to be added by the PAT. The output of the PAT are accumulated by a shift-accumulator [see Fig. 3] over R cycles. Finally, the PSAT produces the filter output using the output from each section every R cycles. The accumulated value is reset every R cycles by the control signal [acc_rst in Fig. 3] to keep the accumulator register ready to be used for calculation of the next filter output. If the maximum operating clock period is f_{clk} , the proposed structure can support the input sample rate of f_{clk}/R .

References

1. T. Hentschel, M. Henker, and G. Fettweis, "The digital front-end of software radio terminals," *IEEE Pers. Commun. Mag.*, vol. 6, no. 4, pp. 40–46, Aug. 1999.
2. K.-H. Chen and T.-D. Chiueh, "A low-power digit-based reconfigurable FIR filter," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 8, pp. 617–621, Aug. 2006.
3. L. Ming and Y. Chao, "The multiplexed structure of multi-channel FIR filter and its resources evaluation," in *Proc. Int. Conf. CDCIEM*, Mar. 2012, pp. 764–768.
4. I. Hatai, I. Chakrabarti, and S. Banerjee, "Reconfigurable architecture of a RRC FIR interpolator for multi-standard digital up converter," in *Proc. IEEE 27th IPDPSW*, May 2013, pp. 247–251.
5. A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 42, no. 9, pp. 569–577, Sep. 1995.
6. S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Mag.*, vol. 6, no. 3, pp. 4–19, Jul. 1989.
7. P. K. Meher, "Hardware-efficient systolization of DA-based calculation of finite digital convolution," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 8, pp. 707–711, Aug. 2006.
8. P. K. Meher, S. Chandrasekaran, and A. Amira, "FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3009–3017, Jul. 2008.
9. M. Kumm, K. Moller, and P. Zipf, "Dynamically reconfigurable FIR filter architectures with fast reconfiguration," in *Proc. 8th Int. Workshop ReCoSoC*, Jul. 2013, pp. 1–8.
10. E. Ozalevli, W. Huang, P. E. Hasler, and D. V. Anderson, "A reconfigurable mixed-signal VLSI implementation of distributed arithmetic used for finite-

impulse response filtering," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 2, pp. 510–521, Mar. 2008.

11. D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, "LMS adaptive filters using distributed arithmetic for high throughput," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 7, pp. 1327–1337, Jul. 2005.
12. P. K. Meher and S. Y. Park, "High-throughput pipelined realization of adaptive FIR filter based on distributed arithmetic," in *Proc. IEEE/IFIP 19th Int. Conf. VLSI-SOC*, Oct. 2011, pp. 428–433.
13. *DesignWare Building Block IP User Guide*, Synposys, Inc., Mountain View, CA, USA, 2012, 06-SP2. *LogiCORE IP FIR Compiler v5.0*, Xilinx, Inc., San Jose, CA, USA, 2010.

Authors Profile:

ATMAKURI VASAVI is pursuing her M. Tech in Department of Electronics and Communication Engineering at Gandhiji Institute of Science & Technology, Jaggaiahpetta.

Her specialization is VLSID.

Mrs. SITA MADHURI BONDILA is an Assistant professor in the Department of Electronics and Communication Engineering at Gandhiji Institute of Science & Technology, Jaggaiahpetta. She has 8 years of teaching experience.