ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

FPGA Based Modified Distributed Arithmetic FIR Filter

Pogula Srivani
M.Tech in VLSI System Design,
Indur Institute of Engineering and Technology,
JNTU, Hyderabad.

Vudthyavath Srinu, M.Tech
Assistant Professor,
Indur Institute of Engineering and Technology,
JNTU, Hyderabad.

Abstract:

This paper presents an efficient implementation of Finite Impulse Response Filter (FIR) using Modified Distributed Arithmetic (DA) architecture using shared look-up table (LUT). It is based on distributed arithmetic (DA) combined with a look-up table (LUT) reduction technique which allows the direct mapping to reconfigurable LUTs of the latest Xilinx FPGAs. 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 design. Therefore, a shared-LUT implementation is proposed to realize the Distributed Arithmetic computation. The LUT is for higher order filter. Each LUT operates different set of filter taps. The proposed architecture provides less latency and less area compared with existing structure of FIR filter.

Index Terms:

Distributed Arithmetic, Finite Impulse Response, Field Programmable Gate Array, Look Up Table.

I. INTRODUCTION:

Digital Signal Processing (DSP) has been increasing in popularity due to the declining cost of general purpose computers and Application Specific integrated circuits. Since many data communications applications have been moving to digital, the need for digital filtering methods continues to grow[1][2]. Along with the advancement in Very Large Scale Integration (VLSI) technology and the DSP has become increasingly popular day by day, the high speed realization of FIR digital filters with less power consumption has become much more demanding.

Since the complexity of implementation grows with the filter order and the precision, real-time realization of these filters with desired level of accuracy is a challenging. Several attempts have, been made to develop dedicated and reconfigurable architectures for realization of FIR filters in Application Specific Integrated Circuits (ASIC) and Field-Programmable Gate Arrays (FPGA) platforms.

A distributed arithmetic (DA)-based technique has gained substantial popularity in recent years for high-throughput processing capability and increase in regularity, which results in cost and area-time efficient computing designs. The main operations required for DA-based designs are a sequence of lookup table (LUT) which followed by shift accumulation operations of the LUT output. The convention DA implementation used for implementing 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, has been exponentially increases with the order of filter. To eliminate the problem for such a large memory requirement, systolic decomposition techniques are implemented by Meher et al. for DA-based implementation of long-length convolutions and FIR filter of large orders For a reconfigurable DA-based FIR filter whose filter coefficients dynamically change, we need to use reconfigurable RAM based LUT instead of ROM-based LUT. Further approach is to store the coefficients in the analog domain by using serial digital-to-analog converters resulting in mixed-signal architecture [10]. We also find quite a few works on DA based implementation for adaptive filters [11], [12] where the coefficients change at every clock cycle. In this approach, we present efficient schemes for the

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

optimized shared-LUT implementation of reconfigurable FIR filters using shared DA technique, where LUTs are shared by the DA units for bit slices of different weightage. In addition, the filter coefficients are subdivided into a number of LUT to reduce the size of the LUT and dynamically changed in runtime with a very small reconfiguration latency.In the next section, we briefly discuss the mathematical background of DA-based implementation of FIR filter. In section III derives the FIR structures for the proposed algorithm. Section IV gives the proposed structure. We provide the synthesis results of proposed and existing designs in Section V. The conclusion and the scope of the future work are presented in section IV.

II. FORMULATION OF THE ALGORITHM:

The output of an FIR filter of length N can be computed as an inner product of the impulse response vector (h(k), for k = 0, 1, . . .,N – 1) and an input vector (x(n – k), for k = 0, 1, . . . , 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)] l for 0 ≤ k ≤ N – 1} can either be 0 or 1, the partial sum Cl for 0 ≤ l ≤ L – 1 can have 2N possible values. If all the 2N possible values of Cl are precomputed and stored in the LUT, the partial sums Cl can be read out from the LUT using the bit sequence {[s(k)]l for 0 ≤ k ≤ 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 straightforward DA-based implementation of FIR filters using the LUT containing 2N possible values of Cl. 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 andM may be any two positive integers), one can map the index k into (m + pM) for m = 0, 1, . . .,M – 1 and p = 0, 1, . . . , P – 1 to express (7) as

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

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

where Sl,p is the sum of partial product of M samples represented as

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

for l = 0, 1, . . . , L − 1 and p = 0, 1, . . . , P − 1.
For any given sequence of impulse response {h(k)}, the 2M possible values of Sl,p corresponding to the 2M permutations of M-point bit sequence {(s(m + pM))l}, for m = 0, 1, . . . , M − 1 and l = 0, 1, . . . , L − 1, may be stored in the LUT of 2M words. These values of Sl,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

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

where $F(\mathbf{b}l,p) = Sl,p$, and

$$\mathbf{b}_{l,p} = \{[s(pM)]_l\,[s(1+pM)]_l, \ldots, [s(M-1+pM)]_l\}$$

for $0 \le l \le L-1$ and $0 \le p \le P-1$. The bit vector $\mathbf{b}l,p$ is used as address word for the LUT, and $F(\cdot)$ is the memory-read operation.

III. FIR FILTER REALIZATION USING DA:

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–parallelout shift register (SIPOSR) of sizeN. The SIPOSR decomposes the N recent most samples to P vectors bp of length M for p = 0, 1, . . . , 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 MUXes.

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 + _log2M_) bits, where W is the wordlength of the filter coefficient. The input of the MUXes are 0, h(2p), h(2p + 1), and h(2p) + h(2p + 1); and the two-bit digit bl,p is fed to MUX l for 0 ≤ l ≤ L − 1 as a control word. We can find that MUX l provides the partial product Sl,p for 0 ≤ l ≤ L − 1 given by (8b).

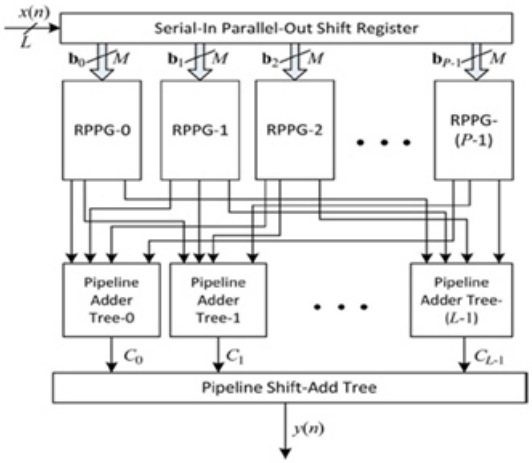

Fig.1. Structure of the high-throughput da-based fir filter.

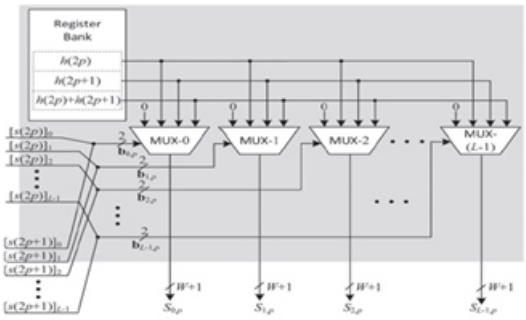

Fig.2. pth RPPG forM = 2

IV. PROPOSED DA TECHNIQUE FOR 3RD ORDER FIR FILTER:

Coefficients = 4
No. of inputs = 4
LUT size = 24 = 16 memory location

In this method possible outputs are pre computed and stored in LUT.LUT addressed through input of the filter. For 4 tap filter,4 tap represents the no. of co-efficient of the filter as well as it represents the no. of inputs to the filter and address bit for the LUT.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Each location has different output for the corresponding inputs .The possible inputs for this filter is 0(0000) - 15(1111).For each input the computation of output is easy by using this technique.

- Input = 1011 means

Output = 1.ho + 0. h1 + 1. h2 + 1.h3
= ho+h2+h3

- Input = 1111 means

Output = ho+ h1+h2+h3

- Input = 0101 means

Output = h1+h3

- Input = 1010 means

Output = ho+h2

It represents the addition of high level input co-efficient. We can easily find 16 output for corresponding input with¬out any mathematical calculation. Table 1 shows the content of the LUT for 3rd order filter

For Example:

Input = X0, X1 ,X2 ,X3

X0 = 1011=11

X1 = 1101=13

X2 = 1010=10

X3 = 1001=9

h0 = h1 = h2 = h3 = 1

Address	Data
0000	0
0001	h_3
0010	h_2
0011	$h_2 + h_3$
0100	h_1
0101	$h_1 + h_3$
0110	$h_1 + h_2$
0111	$h_1 + h_2 + h_3$
1000	h_0
1001	$h_0 + h_3$
1010	$h_0 + h_2$
1011	$h_0 + h_2 + h_3$
1100	$h_0 + h_1$
1101	$h_0 + h_1 + h_3$
1110	$h_0 + h_1 + h_2$
1111	$h_0 + h_1 + h_2 + h_3$

Table I LUT Table for 3rd Order Filter

Step 1:

Store the values in input buffer.

X0[0] X1[0] X2[0] X3[0] =1101

X0[1] X1[1] X2[1] X3[1] =1010

X0[2] X1[2] X2[2] X3[2] =0100

X0[3] X1[3] X2[3] X3[3] =1111

Step 2:

Read the values from LUT for corresponding values in buffer.

Output of LUT:

O1 = 0011 = 3

O2 = 0010 = 2

O3 = 0001 = 1

O4 = 0100 = 4

Step 3:

If the value is multiplied by 2, it implies left shift.

Output = O1 + Shift the value of O2 one time + Shift the value of O3 2 times + Shift value of O4 3 times.

Output = 3 + 4 + 4 + 32 = 43.

A.LUT Partitioning:

The above technique holds good only when we go for lower order filters. For higher order filters, the size of the LUT also increases exponentially with the order of the filter. For a filter with N coefficients, the LUT have 2N values. This in turn reduces the performance.Therefore, for higher order filters, LUT size to be reduced to reasonable levels. To reduce the size, the LUT can be subdivided into a number of LUTs, called LUT partitions. Each LUT partition operates on a different set of filter taps. The results obtained from the partitions are summed.Suppose the length LK inner product, then Eq.5 becomes

$$y = \sum_{k=1}^{LK} A_k x_k \qquad (10)$$

Then the sum can be partitioned into L independent Kth parallel DA LUTs resulting in

$$y = \sum_{l=0}^{L-1} \left[\sum_{n=0}^{N-1} X_{LI} + A_{LI+n} \right] \qquad (11)$$

For 3rd order filter

- Number of partition = 2
- 2 LUT tables are used .Each has 2 inputs
- Memory location = no .of partition * 2n = 2*22 = 8 location
- n = number of inputs of LUT

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

B.3rd Order FIR Filter with Partition Method:

LUT is divided into LUT 1&LUT 2.Each LUT has 2 inputs and 4 memory location. It shown in figure 3.

» Input = 1011 means
» First 2 bits are address bit of LUT 1, output becomes 10 = ho
» Remaining 2 bits are address bit of LUT 2, output becomes 11 = h2 + h3
» Output = output of LUT1 + output of LUT 2 = ho+ h2 + h3

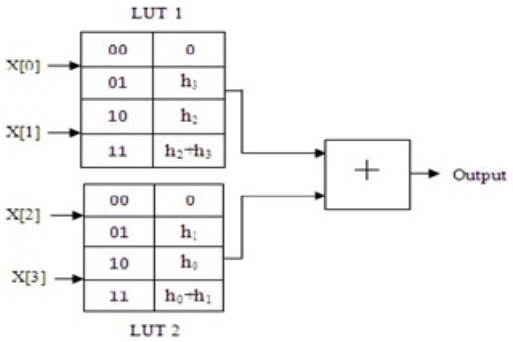

Fig.2. 3rd order FIR filter with partition method

In this method the memory location reduced to 8 loca¬tions. Previous method 16 memory location required to produce the same output.

C.3rd order FIR filter Summary:

» Normal method use : 4 multiplier 3 adders
» In DA technique : Memory location = 24 =16 location
» LUT partition method: Memory location = 2*22 = 8 location

V. RESULT ANALYSIS:

For the DA-based designs, the duration of minimum clock periods depends on the delay of the final adder stage in the PSAT. However, the critical path of the structure in Fig. 1 depends on the parameter M. Specifically, if M is large, the longest path might be the delay of the RPPG plus the delay of the adder in the first stage of the PAT. otherwise, the delay of the final stage of the PSAT becomes the critical path of the filter.It is found that the proposed structure shown in Fig. 1 (using RPPG) produces the most number of output per cycle (NOC) with the systolic structure among all the listed structures.

The throughput rate is the amount of output produced during a duration amounting to the critical path. Therefore, if M is small, the proposed structure in Fig. 1 offers the highest throughput, and it involves nearly L times less LUT resource than that of the systolic structure since N/M register arrays are shared to realize the LUT function corresponding to all the bit slices of different weights.The proposed structure shown in Fig. 2 guarantees a higher throughput than the structures of [6] and [12] for R < L and fewer adders and smaller LUT than the systolic structure. The structure shown in Fig.2 involves fewer adders and registers, but marginally larger LUT, than the structure shown in Fig. 1.The proposed DA-based structure (see Fig. 1) is written in Verilog hardware description language and synthesized by Xilinx Compiler. The proposed structure has less area and shorter MSP compared with the DA-based structure and involves 68% less ADP, respectively.

A.Design Summary Report:

FIR_DA_TOP_4 Project Status (06/07/2015 - 23:37:43)			
Project File:	DESIGN_FIR..xise	Parser Errors:	No Errors
Module Name:	MOD_FIR_DA_TOP_4	Implementation State:	Synthesized
Target Device:	xc3s1600e-4fg320	• Errors:	No Errors
Product Version:	ISE 13.2	• Warnings:	No Warnings
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	46	14752	0%
Number of Slice Flip Flops	52	29504	0%
Number of 4 input LUTs	77	29504	0%
Number of bonded IOBs	20	250	8%
Number of GCLKs	1	24	4%

Fig.3. Summary report for da-based modifier fir filter

B.RTL Schematic:

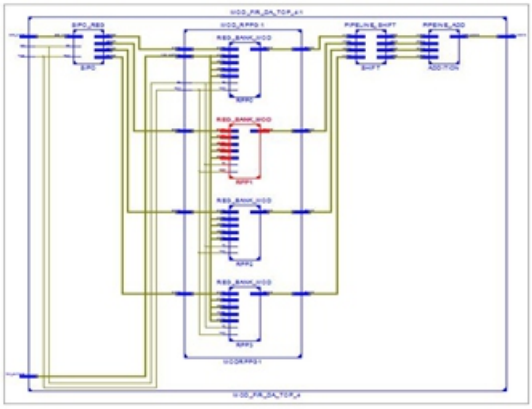

Fig.4.RTL Schematic for da-based modified fir filter

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

C.Simulation Results:

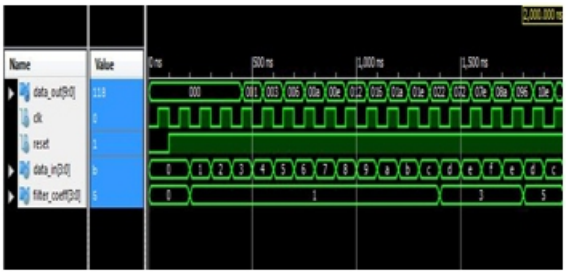

Fig.5. Simulation result for da-based modified fir filter

VI. SUMMARY AND CONCLUSION:

Finite Impulse Response filter plays an important role in many Digital Signal Processing applications. In this method, the multiplier less FIR filter is implemented using Distributed Arithmetic which consists of Look Up Table and then partitioning is involved. Memory access time is less than multiplication time. LUT partition reduces memory requirements. This technique reduces the delay, area, power consumption. This technique reduces the delay by 15%, area by 34% and LUT by 75%. The performance can be further improved by pipelining all the partial tables and design suitable for large filter orders.

REFERENCES:

[1]Kyung-Saeng K, Lee K (2003). Low-power and area efficient FIR filter implementation suitable for multiple tape, Very Large Scale Integration (VLSI) Systems, vol 11, No 1.

[2]Meyer-Base U (2004). Digital Signal Processing with Field Programmable Gate Arrays, 2nd Edn., Chapter 2, 60–66.

[3]Meher P K (2006). Hardware efficient systolization of DA-based calculation of finite digital convolution of finite digital convolution, IEEE Transactions on Circuit and Systems II: Express Briefs, vol 53(8), 707–711.

[4]Meher P K, Chandrasekaran S et al. (2008). FPGA realization of FIR filters by efficient and flexible systolization using dis¬tributed arithmetic, IEEE Transactions on Signal Processing, vol 56(7), 3009–3017.

[5]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.

[6]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.

[7]LogiCORE IP FIR Compiler v5.0, Xilinx, Inc., San Jose, CA, USA, 2010.