Design and Implementation of Distributed Arithmetic-Based Reconfigurable FIR Digital Filter

(1) V. Sowmya, PG Scholar in VLSI, VITS, Proddatur.
(2) Mr. C. Md. Aslam, M.Tech (PhD), Associate Professor, VITS, Proddatur.
(1) vundelasowmya9@gmail.com, (2) aslamhodece@gmail.com.

ABSTRACT:
A distributed arithmetic (DA)-based methodology for high-throughput reconfigurable usage of finite impulse response (FIR) channels whose channel coefficients change during runtime. Expectedly, for reconfigurable DA-based usage of FIR filters, the look-up-tables (LUTs) are obliged to be executed in RAM and the RAM-based LUT is discovered to be immoderate for ASIC execution. Along these lines, a mutual LUT outline is proposed to understand the DA reckoning. As opposed to utilizing separate registers to store the conceivable consequences of fractional internal items for DA preparation of diverse bit-positions, registers are shared by the DA units for bit cuts of distinctive weightage. The proposed design has almost and less delay and less area than the DA-based systolic structure and the carry slave adder (CSA)-based structure, separately, for the ASIC usage. A disseminated RAM-based outline is additionally proposed for the field programmable gate array (FPGA) execution of reconfigurable FIR filter, which backings up to 91 MHz in testing recurrence and offers less quantity of cuts than the systolic structure and the CSA-based structure, individually, when executed in the Xilinx Virtex-5 FPGA gadget (XC5VSX95T-1FF1136).

Keywords: Finite impulse response (FIR) filter, systolic array, field programmable gate arrays (FPGA), application specific integrated chip (ASIC), distributed arithmetic, reconfigurable implementation.

I. INTRODUCTION
Finite impulse response (FIR) digital filters are extensively used due to their key role in various digital signal processing (DSP) applications [1], [2]. Along with the advancement in very large scale integration (VLSI) technology as the DSP has become increasingly popular over the years. Since the complexity of implementation grows with the filter order and the precision of computation, real-time realization of these filters with desired level of accuracy is a challenging task. Several attempts have, therefore, 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. Systolic designs represent an attractive architectural paradigm for efficient hardware implementation of computation-intensive DSP applications, being supported by the features like simplicity, regularity and modularity of structure. Additionally, they also possess significant potential to yield high-throughput rate by exploiting high-level of concurrency using pipelining or parallel processing or both [3]. To utilize the advantages of systolic processing, several algorithms and architectures have been suggested for systolization of FIR filters [4]–[7]. However, the multipliers in these structures require a large portion of the chip-area, and consequently enforce limitation on the maximum possible number of processing elements (PEs) that can be accommodated and the highest order of the filter that can be realized. Multiplierless distributed arithmetic (DA)-based...
technique, has gained substantial popularity, in recent years, for their high-throughput processing capability, and increased regularity which results in cost-effective and area-time efficient computing structures. The main operations required for DA-based computation of inner-product are a sequence of look-up-table (LUT)-accesses followed by shift accumulation operations of the LUT output. DA-based computation is well-suited for FPGA realization, because the LUT as well as the shift-add operations can be efficiently mapped to the LUT-based FPGA logic structures.

In FIR filtering, one of the convolving sequences is derived from the input samples while the other sequence is derived from the fixed impulse response coefficients of the filter. This behavior of FIR filter makes it possible to use DA-based technique for memory-based realization. It yields faster output compared with the multiplier-accumulator-based designs because it stores the pre-computed partial results in the memory elements [8], which can be read out and accumulated to obtain the desired result. The memory requirement of DA-based implementation for FIR filters, however, increases exponentially with the filter order. DA was first introduced by Croisier et al [9]; and further developed by Peled and Lui [10] for efficient implementation of digital filters. Attempts are made to use offset-binary coding to reduce the ROM size by a factor of 2. An LUT-less adder-based DA approach has been suggested by Yoo and Anderson, where memory-space is reduced at the cost of additional adders. Memory-partitioning and multiple memory-bank approach along with flexible multi-bit data-access mechanisms are suggested for FIR filtering and inner-product computation in order to reduce the memory size of DA-based implementation. Allred et al have suggested an efficient DA-based implementation of least mean square (LMS) adaptive filter using a decomposition of DA based FIR computation and subsequent memory decomposition. All these structures, however, are not suitable for implementation of the FIR filters in systolic hardware since the partial products available from the partitioned memory modules are summed together by a network of output adders. A new tool for the automatic generation of highly parallelized FIR filters based on PARO design methodology is presented, where the authors have performed hierarchical partitioning in order to balance the amount of local memory with external communication, and they have achieved higher throughput and smaller latencies by partial localization. A systolic decomposition technique is suggested in a recent paper for memory-efficient DA-based implementation of linear and circular convolutions. In this paper we have extended further the work to obtain an area-delay efficient implementation of FIR filter in FPGA platform.

PROBLEM STATEMENT

Due to the high performance requirements and increasing complexity of DSP and multimedia communication applications, filters with large number of taps are required to increase the performance in terms of high sampling rate. As a result the filtering operations are computationally intensive and more complex in terms of hardware requirements. The FIR filters perform the weighted summations of input sequences with constant coefficients in most of the signal processing and multimedia applications. The decrease in computational complexity causes the increase in the performance, in terms of speed, area and power. High speed and low area conscious design techniques in SoC include efforts at all level of abstraction. One way to efficiently incorporate high performance design technique is to implement IP cores.
These cores have following major advantages.

- Reusability across designs
- Reduction of the design effort
- Shorter time to market.

The disadvantage of FIR filters is that they require high order. The high order demands more hardware and area consumption. To minimize these parameters, our goal is to implement an efficient high order filter in digital systems. By the reduction of arithmetic in terms of multipliers, our goal is to reduce the parameters namely, hardware, area and delay. This is ultimate goal of the implementation of an efficient FIR filter and hence DA algorithm is used for implementation of high order FIR filter. FIR filter is incorporated with a MAC unit. The purpose of MAC unit is to multiply the input with constant coefficients, to shift and then to add them. This process is repeated until all partial products produce the output after accumulation. It increases the hardware complexity because a simple multiplier circuitry is used. The idea is to somehow bypass or replace the multiply and shift operations with less complex operations. Distributed Arithmetic (DA) Algorithm can be used to replace MAC unit. The DA Algorithm actually uses lookup table for storing constant coefficients. So the use of lookup tables reduces the hardware complexity and hence the new design is more efficient in terms of less area and more speed. FIR filter reference core uses a simple MAC unit. We have replaced MAC unit in FIR filter reference core with DA Algorithm. In this study, performance of Reference Core with Simple MAC and reference core with DA is compared.

II. INTRODUCTION OF DISTRIBUTED ALGORITHM

Distributed arithmetic is a bit level rearrangement of a multiply accumulate to hide the multiplications. It is a powerful technique for reducing the size of a parallel hardware multiply-accumulate that is well suited to FPGA designs. It can also be extended to other sum functions such as complex multiplies, Fourier transforms and so on. In most of the multiply accumulate applications in signal processing, one of the multiplicands for each product is a constant. The DA targets the products of sums which cover all filtering application and frequency transfer functions. DA uses Look-Up Table (LUT) which stores the constant coefficients of FIR Filter. The size of Look-Up Table (LUT) in DA algorithm is 2, where k is the number of filter taps. When number of taps increases, LUT grows exponentially. By using offset Binary Code (OBC), the size of the LUT can be reduced. This is very efficient in terms of less hardware and more speed. Many DSP applications required FIR which having MAC (Unit multiplier and add accumulator), replacing MAC with LUT-Based DA algorithm having efficiency and less area usage. Proposed DA algorithm is hardware efficient for VLSI and FPGA, but LUT-Less OBC is efficient only for custom VLSI [5]. We have used DA for multiplier less architecture in FPGA. For DA based on look-up table having constant coefficient and changing variable, one needs to design a highly efficient FIR in digital signal processing.

DA can be used for high order filter. There are two techniques used in DA algorithm, one of which is parallel distributed and the other is serial distributed. [6]. The DSP FIR filter functions are used in telecommunications (e.g. Telecomm in Biomedical Signal Processing Communication, Wireless satellite and Image processing) which are performed efficiently. The multipliers in MAC unit of many DSP functions have more area requirements. There are two techniques in this respect which are multiplier less.
One of them is Conversion based, in which coefficients of filters are converted into numeric representation. The second is based on LUT which stores pre-computed coefficients values of FIR filters. The LUT in DA algorithm uses more memory. [7].

III. IMPLEMENTED DA-BASED FIR FILTER FOR ASIC

The implemented structure of the DA-based FIR filter for ASIC implementation is shown in Fig. 1.

![Fig. 1. Implemented structure of DA-based FIR filters for ASIC.](image)

The input samples \( x(n) \) arriving at every sampling instant are fed to a serial-in-parallel-out shift register (SIPOSOR) of size \( N \). The SIPOSOR decomposes the \( N \) recent most samples to \( P \) vectors \( h_p \) of length \( M \) for \( p = 0, 1, \ldots, P - 1 \) and feeds them to \( P \) reconfigurable partial product generators (RPPGs) to calculate the partial products. The structure of the implemented RPPG is depicted in Fig. 2 for \( M = 2 \).

![Fig. 2. pth RPPG for \( M = 2 \).](image)

For high-throughput implementation, the RPPG generates \( L \) partial products corresponding to \( L \) bitslices in parallel using the LUT composed of a single register bank of \( 2M - 1 \) registers and \( L \) number of 2M:1MUXes. In the implemented 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 inorder 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 \((W + \log_2 2M)\) bits, where \( W \) is the word length 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 \( b_l,p \) 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 \( 0 \leq l \leq L \).

IV. IMPLEMENTED DA-BASED FIR FILTER FOR FPGA

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 implemented reconfigurable FIR filter may be also implemented as part for the complete system on FPGA. Therefore, here we implement a reconfigurable DA-based FIR filter for FPGA implementation. The architectures suggested in Section IV 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 resources 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. 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 $l$ can be mapped into $(r + qR)$ for $r = 0, 1, \ldots, R - 1$ and $q = 0, 1, \ldots, Q - 1$

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

Fig. 3(a) shows the structure of the implemented time-multiplexed DA-based FIR filter using DRAM. To implement above equation, the structure has $Q$ sections, and each section consists of $P$ DRAM-based RRPGs (DRPPGs) and the PAT to calculate the rightmost summation, followed by shift-accumulator that performs over $R$ cycles according to the second summation.

Fig. 3. Implemented structure of the DA-based FIR filter for FPGA. (a) Structure of the DA-based FIR filter. (b) Structure of the DRPPG for $M=2$ and $R=2$. (c) Structure of the shift-accumulator.

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(b). The implemented 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 $Sr+qR, p$ for $p = 0, 1, \ldots, P–1$ to be added by the PAT. The output of the PAT are accumulated by a shift-accumulator [see Fig. 3(c)] 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(c)] 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 $f_{clk}/R$.

V. SIMULATION AND RESULTS

Implementation of FIR filter cores has been observed and we can see that FIR filter cores have been implemented with both reference and DA structure. Results have been taken in terms of area utilized and speed performance for 16 bits-20 taps and 8 bits-20 taps. FIR filter cores have been designed in Verilog HDL and implemented using Xilinx 13.2 tool. Simulations were performed using Modelsim 6.4b.

Table 1 show that the area of Conv. UDF FIR Filter is less as compared
with same core implemented with DA algorithm

<table>
<thead>
<tr>
<th>Filter Cores</th>
<th>Conv. UDF FIR Filter Core</th>
<th>Conv. UDF FIR Filter Core with DA Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Slices</td>
<td>70%</td>
<td>74%</td>
</tr>
<tr>
<td>Slice Flip Flips</td>
<td>32%</td>
<td>37%</td>
</tr>
<tr>
<td>Input LUTs</td>
<td>38%</td>
<td>37%</td>
</tr>
<tr>
<td>Bonded IOBs</td>
<td>60%</td>
<td>60%</td>
</tr>
<tr>
<td>Total Eq Gate Count</td>
<td>13610</td>
<td>13573</td>
</tr>
</tbody>
</table>

**Speed comparison (16 bit 16-taps)**

Table 2 and Fig. 5 show results comparison of conv. UDF FIR Filter Core and conv. UDF FIR Core with DA Algorithm for speed. About 47% increase in speed has been found when using DA algorithm.

<table>
<thead>
<tr>
<th>Filter Cores</th>
<th>Conv. UDF FIR Filter Core</th>
<th>Conv. UDF FIR Filter Core with DA Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min Period</td>
<td>26.798 nS</td>
<td>18.161 nS</td>
</tr>
<tr>
<td>Input Arrival time</td>
<td>9.574 nS</td>
<td>9.574 nS</td>
</tr>
<tr>
<td>Output Req Time</td>
<td>15.842 nS</td>
<td>16.526 nS</td>
</tr>
<tr>
<td>Max Freq</td>
<td>37.316 Hz</td>
<td>55.063 Hz</td>
</tr>
<tr>
<td>Speed Improvement</td>
<td>——</td>
<td>47.56%</td>
</tr>
</tbody>
</table>

**EXTENSION:**

In this paper, we present an efficient architecture for the implementation of a delayed DA-based FIR filter. For achieving lower adaptation-delay and area-delay efficient implementation, we use a novel partial product generator and propose a strategy for optimized balanced pipelining across the time-consuming combinational blocks of the structure, to reduce the adaptation delay as well.

**Conclusion and Future Work**

The results show that distributed arithmetic algorithm is better for FIR filters implementation on FPGAs. The efficiency in terms of area, speed has been analyzed. Comparison of results clearly shows that efficiency in terms of speed has been increased having almost same area consumption. The DA has two techniques, one of which is the serial DA and other one is the parallel DA. In this thesis, the serial distributed arithmetic is used to make the FIR Filter more efficient. In future, the parallel DA can be used to increase the efficiency of FIR Filter in terms of data rates. The implementation of DA based algorithm, serial distributed arithmetic algorithm and parallel distributed arithmetic use the look up table. The size of the look up table increases when the number of filter taps is increased. For better performance of FIR Filter, LUT less DA implementation uses MUX. Every shift register uses MUX that select 0 or filter coefficients, and this technique can be used to increase the efficiency of the FIR Filter in future.

**REFERENCES**