DA-Based Reconfigurable FIR Digital Filter With Efficient FPGA and ASIC Realizations

Madhuri Pabolu¹, K. Bala²

¹M.Tech, Dept of ECE, Srinivasa Institute of Technology and Science, Ukkayapalli, kadapa, India
²Associate Professor, Dept of ECE, Srinivasa Institute of Technology and Science, Ukkayapalli, kadapa, India

ABSTRACT

This brief 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 weight age. 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 distributed-RAM-based design is also proposed for the field-programmable gate array (FPGA) implementation of the reconfigurable FIR filter, which supports up to 91 MHz input sampling frequency and offers 54% and 29% less the number of slices than the systolic structure and the CSA-based structure, respectively, when implemented in the Xilinx Virtex-5 FPGA device (XC5VSX95T-1FF1136)

Keywords: Efficient Distributed Arithmetic (DA), Finite Impulse Response (FIR), Look-up-table (LUT).

1. INTRODUCTION

Finite Impulse Response (FIR) filters are one of the most common components of Digital Signal Processing (DSP) systems. FIR filtering is achieved by convolving the input data samples with the desired unit response of the filter. 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 FPGA platforms. DA provides an approach for multiplier-less implementation of FIR filters where the filter coefficients are programmable. In other words, the same filter structure can be used for a different set of coefficients.

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, 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. and further developed by Peled and Lui 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-power-efficient implementation of FIR filter in FPGA platform.

In general, filtering is the reduction of some unwanted input spectral components, that is, filters will allow certain frequencies to pass while attenuating other frequencies. The filter design process can be described as an optimization problem where each requirement contributes with a term to an error function which should be minimized. The major drawback is that filters with sharp transitions have very high implementation complexity. Certain parts of the design process can be automated, but normally it is necessary to get a good result. One of the techniques to design optimal FIR filters is Equiripple FIR Filter. The coefficients of filter can be generated by using FDA tool and are stored in finite length registers. After designing process over, effect of finite word length that includes the quantization of coefficients on filter design performed. A finite impulse response (FIR) filter is a type of a discrete-time filter. The impulse response, the filter's response to a Kronecker delta input, is finite because it settles to zero in a finite number of sample intervals. This is in contrast to infinite impulse response (IIR) filters, which have internal feedback and may continue to respond indefinitely. The impulse response of an Nth-order FIR filter lasts for N+1.

II. DESIGN OF A FIR FILTER USING DISTRIBUTED ARITHMETIC ALGORITHM

The realization FIR filter requires more number of additions and multiplications. For the hardware implementation we need to consider the parameters like area power and timing. If in the case of FPGA implementation we cannot consider that much trade off in area why because the area specification of the FPGA is fixed. So we need to study the efficient implementation algorithm. Here we presented the
Distributed Arithmetic algorithm to reduce the number of addition and multiplication by using this method we can make sure that the area of the FPGA is reduced.

FILTER DESIGN

Digital filters with finite-duration impulse response (all-zero, or FIR filters) have both advantages and disadvantages compared to infinite-duration impulse response (IIR) filters. FIR filters have the following primary advantages:

- They can have exactly linear phase.
- They are always stable.
- The design methods are generally linear.
- They can be realized efficiently in hardware.
- The filter startup transients have finite duration.

The primary disadvantage of FIR filters is that they often require a much higher filter order than IIR filters to achieve a given level of performance. Correspondingly, the delay of these filters is often much greater than for equal performances IIR filter.

STEPS IN DESIGNING FIR FILTER

1. Specifications include passband and stopband frequency, sampling frequency, transition band, passband and stopband attenuation.
2. Realization includes the type of filter structure and its filter length or order.
3. Quantization of the filter coefficients for a given word length in Q format. The quantization error is introduced by multiply-add, round off, and coefficient representations. The quality of the filter designed is in reference to its infinite precision equivalent.
4. Develop the VHDL code for FIR filter.
5. Synthesis the design for Spartan 3E FPGA

DESIGN SPECIFICATIONS

The design of low pass, FIR equiripple filter has following specifications.

Passband FP: Frequency range of input signal that pass without attenuation. It is also called the bandwidth of the filter.

Figure 1: Low Pass Filter Specifications

III. PROPOSED RECONFIGURABLE DA-BASED FIR FILTER FOR ASIC IMPLEMENTATION

The proposed structure of the DA-based FIR filter for ASIC implementation is shown in Fig. 3.1.

Figure 2: Proposed structure of the high-throughput DA-based FIR filter for ASIC implementation. RPPG stands for reconfigurable partial product generator.
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 \( p=0,1,...,P-1 \) 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 \((W+[\log_2 M])\) 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 \( Sl,p \) for \( 0 \leq l \leq L-1 \).

Figure 3: pth RPPG for \( M=2 \).

The \((W+1)\)-bit partial products generated by the PRPPG blocks are added by \( L \) separate pipeline adder trees (PATs) according to the inner summation in (8a). 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 in (8a). The PAT requires \( P-1 \) adders in \([\log_2 P]\) stages and the PSAT requires \( L-1 \) adders in \([\log_2 L]\) stages. 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. 3.3(b). 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, PDRPPGs in the \( q \)th section generate \( P \) partial inner products \( Sr+qR,p \) for \( p=0,1,...,P-1 \) to be added by the PAT. The outputs 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. 3.3(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 off \( f_{clk}/R \).
IV. LOOK UP TABLE FOR 4 TAP FILTER

The number of filter inputs is 4. So, the number of memory locations needed will be 16(24 ) words. The following table shows the contents which are stored in look up Table (LUT). whenever there is change in coefficient values through buffer. The serial output is presented to the Read Only Memory (ROM) based shift registers. It stores the data in a particular address. The outputs of registered LUTs are added and loaded to the scaling accumulator from LSB to MSB and the result which is the filter output will be accumulated on to the output register over the time. For an n bit input, n+1 clock cycles are needed for a symmetrical filter to generate the output. If there is any change in h[n], it will be updated and the resultant content is stored in the LUTs. This is shown in Figure 2.

FIGURE 4.1: WAVEFORMS OF DA-FIR FILTER

Figure 4: Proposed structure of the DA-based FIR filter for FPGA implementation. (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
5. CONCLUDING REMARKS

We have suggested efficient schemes for high-throughput reconfigurable DA-based implementation of FIR digital filters. It is shown that the hardware cost could be substantially reduced by sharing the same registers by the DA units for different bit slices. The proposed design has nearly 68% and 58% less ADP and 78% and 59% less EPS than the DA-based systolic structure and the DA-based structure using CSA, respectively, for the ASIC implementation. The proposed structure of reconfigurable FIR filter for FPGA implementation supports up to 91 MHz input sampling frequency. It is found to offer 54% and 29% less NOS than the systolic structure and the CSA-based structure, respectively. In feature work more reconfigurable FIR filter may also be implemented as part for the complete system on FPGA. Therefore, implementing of the reconfigurable DA-based FIR filter for FPGA sections. To get rid of the problem of such large memory requirement, systolic decomposition techniques for DA-based implementation of long-length convolutions and adding FIR filter of large orders.

6. REFERENCES


**Author’s Biography**

Madhuri Pabolu received her B.Tech degree from Annamacharya Institute of Technology & Sciences, Rajampet, Kadapa, India (Affiliated to JNTUA Ananthapur) Department of ECE. She is pursuing M.tech in Srinivasa Institute of Technology and Science, Ukkayapalli, Kadapa, AP.

Mr. K. Bala is currently working as an associate professor in ECE Department, Srinivasa Institute of Technology and Science, Ukkayapalli, Kadapa, India. He received his M.Tech from Sri Kottam Tulasi Reddy Memorial college of Engineering Kondair, Mahaboobnagar, A.P., India