 ISSN: 2321-8169
2027 – 2029

Design of FIR Filter using Distributed Arithmetic Architecture

Gurneet Kaur and Amandeep Singh Sappal
Department of Electronics and Communication Engineering, Punjabi University, Patiala

Abstract:--Digital filters can be realized in hardware, software. In general digital filters tend to be more expensive than analog filters.But digital filters can be used to implement complex applications that analog filters cannot. This paper presents the design of a double-precision lowpass direct form Finite Impulse Response (FIR) filter to meet the given specification. Simulation results have been presented.

Keywords---*Distributed Arithmetic, Double Precision, Finite Impulse Response, HDL, Quantized.*

*****

I. INTRODUCTION

Distributed Arithmetic (DA) is widely used method to save resources in multiply-and-accumulate structures implementing DSP functions [1-2]. DA trades memory for combinatory elements, resulting in ideal-to-implement custom DSPs in LUT-based FPGAs [3]. In addition to a DA implementation, the designer also can select from a bit-serial to a full-parallel implementation to trade bandwidth for resource utilization [4]. Cascade and lattice structures present several interesting properties such as low quantification error and high-stability in the filter coefficients. Moreover, you can expand lattice cells without a full redesign [5]. The paper presents the implementation of distributed arithmetic architecture implementation of FIR filter. The paper is organised as follows: section I is introduction, section II presents bit serial distributed arithmetic architecture, proposed filter implementation is presented in section III and section IV is conclusion.

II. BIT SERIAL DISTRIBUTED ARITHMETIC ARCHITECTURE

Equation 2 expresses FIR filter operation given by equation 1, using the 2's complement representation of the x[n] input samples of N bits.

$$y = \sum_{t=1}^{T} A_t x_t \tag{1}$$

$$y = \sum_{t=1}^{T} A_t(-x_{t0} + \sum_{n=1}^{N-1} x_{t,n} 2^{-n}) = \\ = -\sum_{t=1}^{T} A_t x_{t0} + \sum_{n=1}^{N-1}[\sum_{t=1}^{T} A_t x_{t,n} 2^{-n}] \tag{2}$$

You can pre-calculate the terms in brackets in Equation 2, save the results in memory, and address these terms by $x_{t,n}$ in Table 1. Considering that each x_{tn} can only take two values (0 or 1), each product term reaches one of the $2^{(N-1)}$ possible values.

Table 1: Distributed Arithmetic pre-calculated terms

Address					
$x_{t,N-1}$	...	$x_{t,2}$	$x_{t,1}$	$x_{t,0}$	Content
0	...	0	0	0	0
0	...	0	0	1	A_1
0	...	0	1	0	A_2
0	...	0	1	1	$A_2 + A_1$
1	...	1	1	1	$A_T... + A_2 + A_1$

A DA bit-serial implementation of a FIR filter addresses each product term once per bit (the MSB bit is the sign bit). When the last product-term is obtained, this term is added, with its appropriate shift, to the rest of the product term previously added.

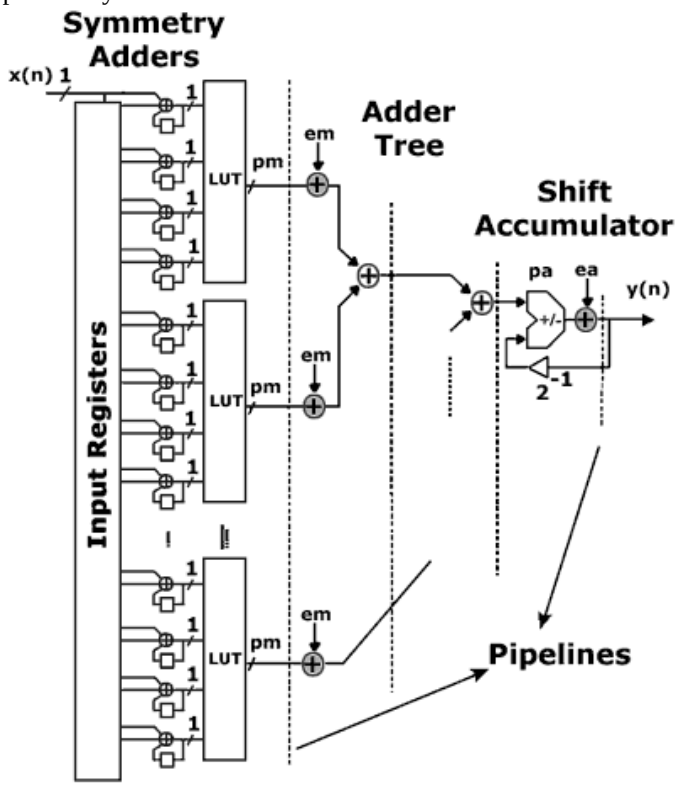

Figure 1: Bit-serial DA direct-form FIR filter

Figure 1 shows the structure representing the direct-form FIR filter. Now consider the FPGA, this paper discusses has four-inputs LUTs, the product-terms larger than four need to

ISSN: 2321-8169
2027 – 2029

be divided into r parts such that 4<T/r, where T is the number of taps of the filter. In other words, the adders in the tree structure add the r LUT outputs. Eventually, you need a shift accumulator to add and shift each product term. A bit-serial implementation of a filter with samples of 8 bits is represented in figure 1. The output of the filter occurs each eight clock cycles. A subtraction instead of an addition in the shift-accumulator is done after arriving the sign bit By using carry save adders before the LUT, you can implement a symmetrical filter"detailed information of this operation is found in [1,4].Additionally, you can extend the range of processing speed by pipelining the structure. Equation3 expresses the operation frequency (fs), where L is the latency and n the number of the bits of each input sample.

$$f_s = f_{clk} / n + L \quad (3)$$

Despite the increment of registers in the DA pipeline version, the final area resources increase slightly, due to the FPGA structure.

III. DA IMPLEMENTATION OF FIR FILTER

We have designed a FIR filter at sampling rate of 48 kHz, passband edge frequency of 9.6 kHz and stop frequency of 12k. The allowable peak-to-peak passband ripple and the stopband attenuation are set at 1dB and -90 dB respectively. Since DA implements the FIR filter by serializing the input data bits, it requires a quantized filterdue to fixed data path requirements or input ADC/output DAC widths. We have assumed 12 bit input and output word lengths.

To generate HDL Code with DA architecture, we have used the generatehdl command, passing in a valid value to the 'DALUTPartition' property. The 'DALUTPartition' property directs the code generator to use DA architecture, and divides the LUT into a specified number of partitions. The 'DALUTPartition' property specifies the number of LUT partitions, and the number of the taps associated with each partition. For a filter with many taps it is best to divide the taps into a number of LUTs, with each LUT storing the sum of coefficients for only the taps associated with it. The sum of the LUT outputs is computed in a tree structure of adders. To calculate the value of the DALUTPartition property, we have assumed 8-input LUTs.

A symmetrical filter structure offers advantages in hardware, as it halves the number of coefficients to work with. As a result the hardware complexity is reduced substantially. So, we have created a new filter object 'Hdsym' by converting the filter structure to 'dfsymfir' from the previously created filter, 'Hd'. Quantizing this new filter we have again generated HDL code on the new filter It has been concluded that a symmetrical filter takes one additional clock cycle before the output is obtained. This is due to the carry bit that is added to the input word length as the input data from the symmetrical taps are summed together. The clock rate for 'Hdsym' is 13 times the input sample rate, whereas for 'Hd' the clock rate was 12 times the input sample rate.

The default DA architecture is a Radix 2 implementation, which operates on one bit of input data on each clock cycle. Before an output is obtained, the number of clock cycles elapsed is equal to the number of bits in the input data. Thus DA can potentially limit the throughput. To improve the throughput of DA, we have configured DA to process multiple bits in parallel. The 'DARadix' property is used for this purpose. In selecting different 'DARadix' values, we can trade off speed vs. area within the DA architecture. The number of bits operated in parallel determines the factor by which the clock rate needs to be increased. This is known as folding factor. It has been concluded that he default 'DARadix of 2^1, implying 1 bit at a time, results in a clock rate 12 times the input sample rate or a folding factor of 12. A 'DARadix' of 2^3 results in a clock rate only 4 times the input sample rate, but requires 3 identical sets of LUTs, one for each bit being processed in parallel.

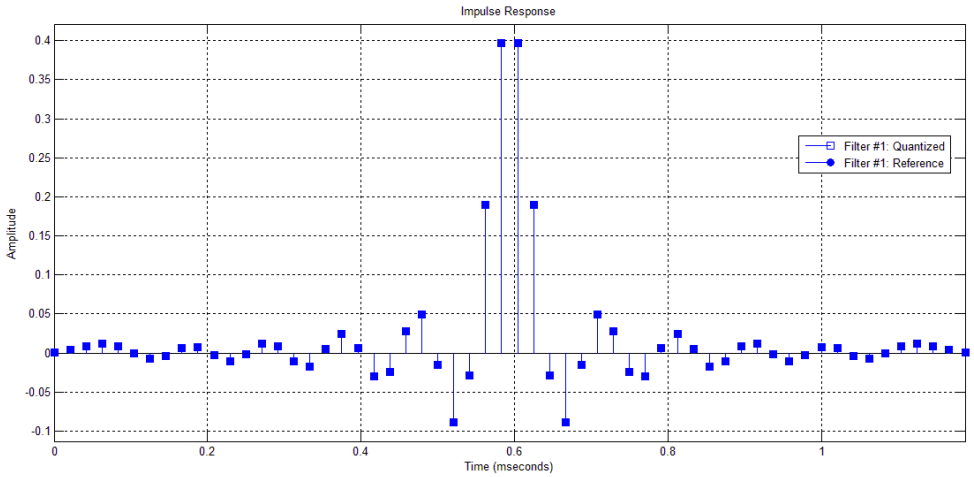

Figure 2: Impulse Response of the Designed Filter

 ISSN: 2321-8169
2027 – 2029

Figure 2 shows the impulse response of the designed FIR filter. The magnitude and phase responses of quantized and reference filters are shown in figures 3 and 4 respectively.

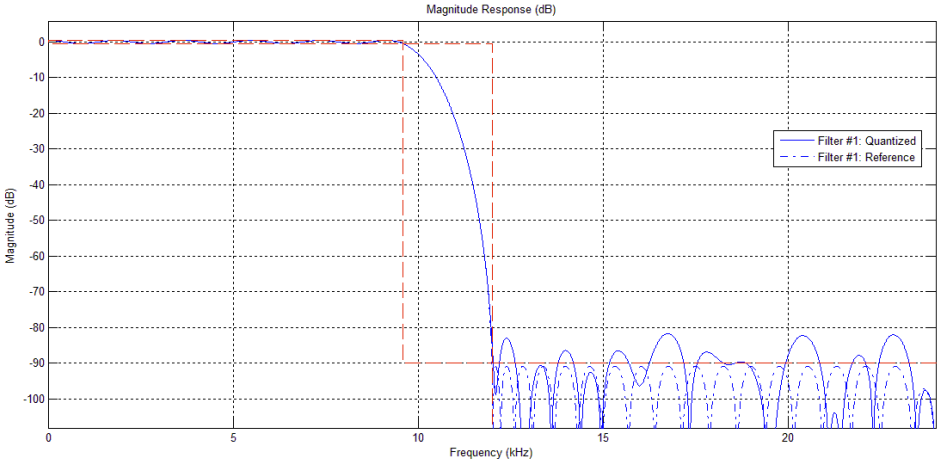

Figure 3: Magnitude of the Quantized and Reference Filter

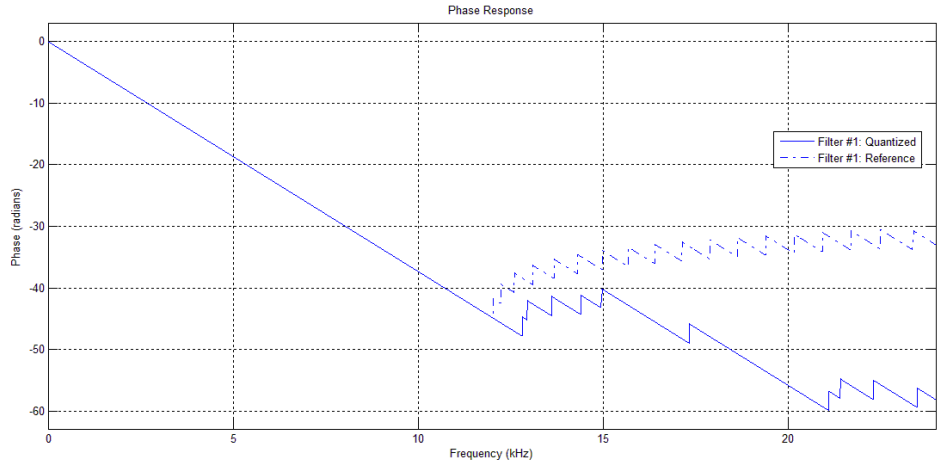

Figure 4: Phase Response of the Quantized and Reference Filter

From figures 3 and 4, it has been concluded that the quantized design has almost same characteristics as that of reference design. The inaccuracy of response during stop band can be increased by assuming more number of quantized bits, which will lead to increase in implementation cost.

IV. CONCLUSION

We have designed a double-precision lowpass direct form FIR filter to meet the given specification. We have generated VHDL code for DA with various radices and explored speed vs. area trade-offs within DA by replicating LUTs and operating on multiple bits in parallel. A test bench has also been generated to verify the generated HDL code for DA architectures.

REFERENCES

[1] S. Mitra, “Digital Signal Processing: A Computer-Based Approach”, 2nd ed. Boston: McGraw-Hill Irwin, 2001.

[2] R. Mersereau& M. Smith, “Digital Filtering: A Computer Laboratory Textbook,”John Wiley & Sons, Inc, 1994.

[3] D. Goldberg, ”Genetic Algorithm in Search, Optimization, and Machine Learning”. Reading, MA: Addison Wesley Pub. Co., 1989.

[4] J. Holland, “Adaptation in Natural and Artificial Systems. Ann Arbor”, MI: The Univeristy of Michigan Press, 1975.

[5] C. Darwin, “The Origin of Species”, ser. The Harvard Classics. New York: P F Collier & on, 1909, vol. 11.

[6] W. Edmonson et al., “A global least mean square algorithm for adaptive iir filtering," IEEE Trans. on Circuits and Systems, vol. 45, no. 3, pp. 379-384, Mar 1998.

[7] D. Talla, S. Rao, & L. John, “An evolutionary computation embedded iirlms algorithm," in Proceedings of International Conference on Signal Processing Applications and Technology, Orlando, FL, Nov 1-4, 1999.

[8] L. Wang, W. Li, & D. Zheng, “A class of hybrid strategy for adaptive iir filter design." Shanghai, China: 8th International Conference on Neuaral Information Processing, Nov 14-18, 2001.