International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.3 Issue No.4, pp : 440-443 1April 2014

Comparison Between FPGA Logic Resources And Embedded Resources Used By Discrete Arithmetic (DA) Architecture To Design FIR Filter

Irfan Ahmed Usmani[1], Dr. Arshad Aziz[2], Muzaffar Rao[3], Razia Zia[2]
[1]Department of Electronic Engineering, University of Tabuk, Saudia Arabia
[2]Department of Electrical & Electronic Engineering, NUST, Karachi Pakistan
[3]Department of Electronic & Computer Engineering, University of Limerick, Ireland
[1]iusmani@ut.edu.sa

Abstract — FIR Filter is a fundamental function in many applications, mainly in the field of Digital Signal Processing (DSP). The main problem of this function is its computational complexity, needed to process a signal. Currently, FPGAs are widely used for a variety of computationally complex applications. New generation of FPGAs contain not only basic configurable logic resources but also complex embedded resources. The FPGA implementation of FIR filters based on customary design & techniques utilizes extensive hardware resources, which does not comply circuit scale and system speed. In this paper we investigated the resource utilization and system speed of a new implementation of FIR Filter using DA architecture with and without use of embedded component like DSP units. The filter is designed and Co-simulated with the help of 'filter design HDL Coder' and then synthesized using Xilinx ISE 13.3 Design suit. Result shows that DA based FIR filter design that utilizes basic logic resources gives better performance and efficient resource utilization in comparison with the design utilizing DSP components.

Keywords — FPGA, Discrete Arithmetic, FIR, HDL Coder

I. Introduction

There are two types of digital filters: (i) Infinite Impulse Response (IIR) Filters (ii) Finite Impulse Response (FIR) Filters. FIR filters are widely applied in digital signal processing based applications because of its stability and linear phase; but the main dilemma of such filters is the amount of computation needed to process a signal. A FIR Filter process an incoming signal by using its conventional multiply and accumulate (MAC) based arithmetic. The conventional FIR computes the convolution (MAC based arithmetic) sum, according to equation 1.

$$y(n) = \sum_{k=0}^{N-1} a(k)\, x(n-k); n = 0,1,2,\cdots \qquad \text{Equation 1}$$

Where $a(n)$ and $x(n)$ are representing the filter coefficient and input sequence respectively. A block diagram of FIR filter representing the MAC operation is shown in Figure 1.

In general, FIR filters are implemented on Digital Signal Processor, but implementation on such processor, some time does not meet the high speed requirement. Currently, FPGAs are widely used for a variety of computationally complex applications. FPGA based solution can achieve high speed due to its parallel structure, configurable logic and embedded resources, which provides great flexibility and high reliability in the course of design and later maintenance [2]. Concerning hardware requirement to implement MAC operations on FPGA; one MAC unit will be enough but it will take multiple numbers of MAC cycles, before the next input sample can be processed. The process can speed up by using a parallel implementation with multiple numbers of MAC units. It is clear from FIR arithmetic and block diagram representation that it will consume 'N' MAC blocks of FPGA, which are expensive in high speed system [2].

To handle these operations, embedded resources like DSP units can be used. Improved hardware performance and good balance in terms of the overall FPGA utilization can be achieved with the use of these embedded elements [3]. An alternate approach is to use DA realization. This approach employs no explicit multipliers in the design, only look up table (LUTs), shift registers, and a scaling accumulator [1]. In this paper, we investigated both approaches in a way that first we implemented FIR filter based on DA realization and then we restructured the design implementation using embedded DSP units for the efficient utilization of hardware resources. We used 'filter design HDL coder' to design filter and to generate Verilog code. 'Filter design HDL coder' is also used to test and co-simulate the design. All the designs, in this paper, are synthesized using 'Xilinx ISE design suit'.

The rest of the paper is structured as follows: Section 2 describes DA realization. Section 3 is about filter designing using HDL coder and generation of Verilog Code. Section 4 describes the results and finally we concluded our work in Section 6.

II. Discrete Arithmetic (DA) Realization

As mentioned above, a DA realization employs no explicit multipliers in the design, only LUTs, shift registers and a scaling accumulator. Its realization says that LUTs are used to save the MAC values that can be called out according to the input data. Therefore, DA can save considerable hardware resources through using LUT by taking place of MAC units [4]. Another virtue of this method is that it can avoid system speed decrease with the increase of the input data bit width or the filter coefficient bit width, which can occur in traditional direct method and consume considerable hardware resources [5]. Using DA method, the filter can be implemented either in serial or fully parallel mode according to the requirement of speed and area utilization. Figure 2 represents the realization of DA used into the design of FIR filter. The mathematical description of this realization has been discussed many times in [6][7][8].

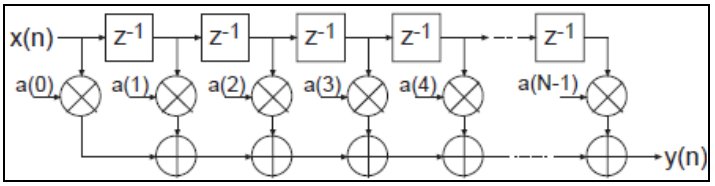

Figure 1: Block diagram of basic FIR Filter [1]

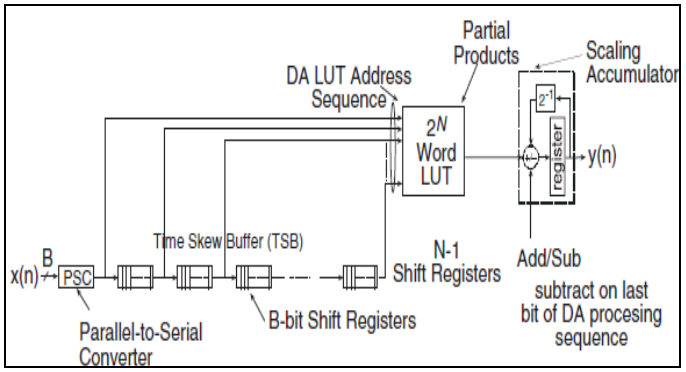

Figure 2: DA realization in designing FIR Filter [1]

Basically, DA based computations are bit serial in nature. The inherently bit serial nature of DA limits the throughput [9]. Throughput can be improved by updating the basic algorithm to compute more than one bit sum at one time. One approach to modify the algorithm is to factorize the input variable into single bit subwords. The resulting realization is known as fully parallel DA. This alternative approach computes a new output sample on each clock cycle. Parallel DA provides exponentially higher performance. Filters may be designed that process several bits in a clock period, through to a completely parallel architecture that processes all the bits of input data during a single clock period [1]. The higher degree of parallelism, the greater the FPGA logic resources required to implement the design. Specifying the number of clock cycles per output sample is an extremely powerful mechanism that allows the designer to trade off silicon area with filter output [1]. The basic functions required are shifts of input data sequence, a sequence of look up tables, and additions/subtractions. Figure 2 shows that the LUT is followed by a scaling accumulator (shift and Adder/Subtractor) that adds the in sequence values, obtained from the LUT. For each bit, starting from LSB, a table lookup is executed. On each clock cycle, the LUT result is added to the accumulated and shift result from the previous cycle. As DA is serial in nature, performs per bit operation at one time. If the input data sequence is B bits wide, then a DA-based asymmetric FIR structure consumes 'B' clock cycles to compute the output.

III. Filter Designing and HDL Code Generation

Traditionally, Hardware description language (HDL) is being used by the system designers and hardware developers to develop FPGA design. HDL code generation speed up the development of the designs and provide a channel between system level design and hardware development. Although HDL provides a proven method for hardware design, the task of coding filter designs, but in general, is labour intensive and the use of these languages for algorithm and system-level design is not optimal [9]. To design filters, this problem can be solved using 'Filter Design HDL Coder'. MATLAB based Filter design HDL Coder solution, used in this paper, is summarized in Figure 3. Observations tells us that if we use 'Filter Design HDL Coder' then we pay more time on updating algorithms and models through rapid prototyping and experimentation and less time on HDL coding. Using HDL Coder, We can efficiently design, analyze, simulate, and can hand over our designs to hardware developers for its implementation.

Figure 4 shows a flow chart, describing the process to design filter using 'Filter Design HDL Coder'.

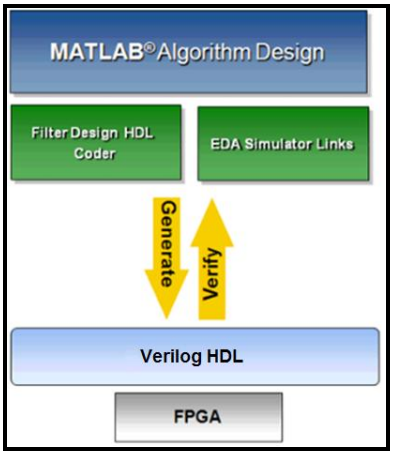

Figure 3: The Algorithm Design Solution

A brief description about how to use this tool is provided by the MathWorks, Inc., in Filter Design HDL Coder™ User's Guide [9]. The designed filters, in this paper, are summarized in table 1. The table sums up the following specifications. 'Design method' describes the mathematical technique used to generate the coefficients, 'Type' is the characteristics of the filter, F_S, F_{Pass} and F_{Stop} represent the sampling frequency, passband frequency and stopband frequency respectively, and lastly Order is the number of taps minus one. Also, table 1 shows different quantization parameter of the filters. Where, input is the size of the input signal and width is the coefficient word size. After designing filters, we used HDL coder to restructure the basic filter architecture using DA, performed the Cosimulation using Simulink and ModelSim and generated Verilog code to synthesize it on Xilinx ISE. To restructure the filter using DA, the two parameters, DA Radix/Folding Factor and Address width/LUT Partition has to be chosen carefully. These two parameters are important to increase the system speed by using pipeline structure and decrease the required memory units by using the divided LUT method. All the filters, summarized in table 1, are tested and Cosimulated using Simulink and ModelSim.

A unit impulse is applied at the input of the filter. Because of the linear time invariant (LTI) systems, Unit impulse will not make

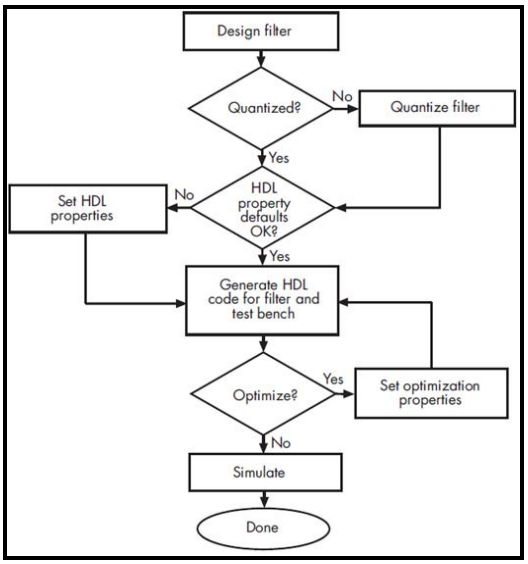

Figure 4: Flow Chart used by Filter Design HDL Coder

Table 1: Benchmark FIR filter specifications and parameters [10]

FIL-TER	FILTER SPECIFICATION						FILTER PARAMETERS	
	DESIGN METHOD	TYPE	F_s	F_{pass}	F_{stop}	ORDER	INPUT	WIDTH
1	EQUIRIPPLE	LOWPASS	48000	9600	12000	50	16	16
2	EQUIRIPPLE	LOWPASS	48000	7200	9600	50	16	20
3	EQUIRIPPLE	LOWPASS	48000	4800	6400	76	16	14
4	WINDOW (KAISER)	LOWPASS	48000	N/A	10800 (F_c)	151	16	16
5	EQUIRIPPLE	HIGHPASS	48000	12000	9600	56	16	18
6	EQUIRIPPLE	HIGHPASS	44100	12000	9600	50	20	18
7	LEAST-SQUARE	HIGHPASS	44100	6615	4410	50	16	12
8	WINDOW (HAMMING)	HIGHPASS	48000	N/A	10800 (F_c)	79	16	13

any effect on the impulse response of the system. The output or system response will be exactly equal to the impulse response of the filter. Figure 5 shows the impulse response of the filter in simulink environment. Figure 6 shows the Cosimulation result. Simulink behavioural model and ModelSim both generated the same output and accordingly zero error.

IV. Filter Designing and HDL Code Generation

To investigate the effect of DA radix on system speed, throughput and area utilization, we simulated Filter 1 from table 1.The simulations have been performed on different radix values at an address width of 2. Result shows, summarized in table 2, lower clock frequencies as the number of bits for parallel computing are increased; while an exponential increase in throughput with an increase in utilized area. With the parameters (DA radix, Address width) = $(2^1, 2)$, because of bit-serial nature, DA architecture produced the highest clock speed and the lowest utilized area at the cost of limited throughput. Through is calculated by using the formula:

$$\text{Throughput} = \frac{\text{Clock Frequency} \times \text{Number of simultaneous processed input bits}}{\text{Clock Cycles per input sample}}$$

For our further investigations, we used DA radix = 2^1 and Address width = 2 according to its resulted system speed and utilized area. We used Xilinx 13.3 ISE for targeted FPGA Vertex 5 device 'xc5vsx50t-3ff1136'to synthesize all HDL coder generated Verilog code, related to each filter design, mentioned in table 1. These filters are first investigated for utilizing basic configurable logic resources and then for embedded (DSP) resources. Results, associated to speed and area utilization for all investigations are summarized in table 3.

Result shows that use of DSP units reduced the clock frequency as well as the throughput. It is possible because of the use of cascaded DSP architecture. DSP operations could delay the critical path due to the possible routing delays from basic logic resources to DSP slices. Regarding area utilization, result shows that use of embedded (DSP) units save almost 50% of logic resources by moving it into embedded resources. This saving is not significant because one DSP unit is equivalent to over 500 CLB Slices [11]. Table 3 clearly shows that the moved CLB

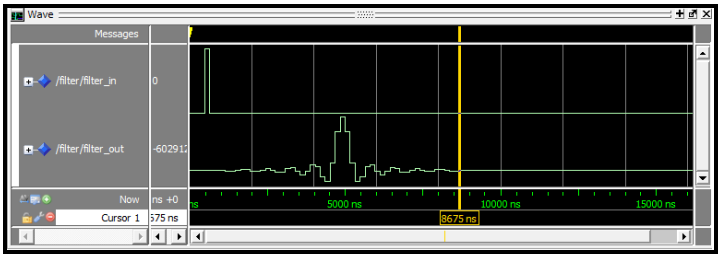

Figure 5: Impulse response of the filter – simulated by ModelSim

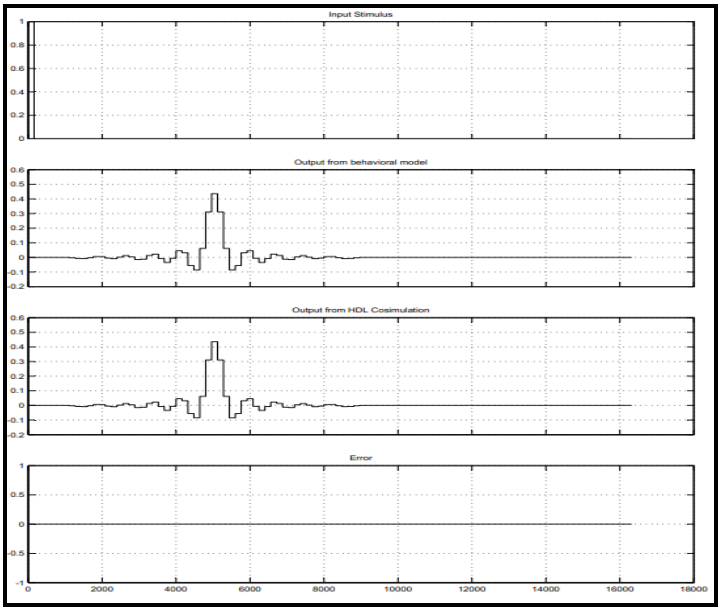

Figure 6: Simulink Cosimulation result

Slices portion into DSP is not very large compared to over 500 CLB Slices/DSP.

V. Conclusion

Eight bench mark FIR filters were implemented using DA architecture on Xilinx FPGA. All implemented filters using DA architecture in its direct form showed the utilization of only basic reconfigurable logic resources (LUTs and Shift Registers). Xilinx tool is used to optimize/ modify the basic design to use embedded DSP units; but utilization of embedded DSP units did not show any significant improvement specially in increasing system speed. Due to the computational complexity of FIR filters, their implementation demands high speed to process a signal mainly in the field of Digital Signal Processing. Therefore, we should use DA architecture directly to design FIR filter without modifying it to utilize the embedded DSP units.

Table 2: Effect of DA Radix on FIR filter implementation

DA RADIX	CLOCK FREQUENCY MHZ	THROUGHPUT MBITS/S	UTILIZED AREA [LUT(LOGIC, SRL)]
$2(2^1)$	409.861	25.61	[611 (559, 50)]
$4(2^2)$	229.988	28.74	[1146 (1046, 100)]
$16(2^4)$	293.010	73.25	[2177 (1981, 196)]
$256(2^8)$	289.155	144.57	[3470 (3470, 0)]
$65536(2^{16})$	334.475	334.475	[6767 (6765, 2)]

Table 3: Comparison between Basic Logic resources and Embedded (DSP) resources utilized by FIR Filter design using DA architecture

FIL-TER	RESOURCE TYPE	CLOCK FREQUENCY (MHZ)	THROUGHPUT (MBITS/S)	RESOURCE UTILIZATION [CLB SLICES, LUT(LOGIC, SRL), DSP]	RESOURCE REPLACEMENT RATIO
1	LOGIC RESOURCES	409.861	25.61	[201, 611(559,50),0]	2.14 CLB SLICES/DSP
	EMBEDDED RESOURCES	300.004	18.75	[141, 292(241,50),28]	
2	LOGIC RESOURCES	395.124	24.69	[240, 713(659,50),0]	4.6 CLB SLICES/DSP
	EMBEDDED RESOURCES	298.497	18.65	[111, 298(247,50),28]	
3	LOGIC RESOURCES	413.719	25.85	[259, 767(680,76),0]	2.68 CLB SLICES/DSP
	EMBEDDED RESOURCES	298.728	18.67	[149, 365(288,76),41]	
4	LOGIC RESOURCES	406.075	25.37	[456,1462(1300,151),0]	2.74 CLB SLICES/DSP
	EMBEDDED RESOURCES	299.359	18.70	[242, 690(538,151),78]	
5	LOGIC RESOURCES	402.358	25.14	[214, 676(618,56),0]	3.5 CLB SLICES/DSP
	EMBEDDED RESOURCES	298.405	18.65	[105, 304(248,56),31]	
6	LOGIC RESOURCES	402.358	20.11	[226, 648(595,50),0]	3.35 CLB SLICES/DSP
	EMBEDDED RESOURCES	305.850	15.29	[132, 296(245,50),28]	
7	LOGIC RESOURCES	425.74	26.60	[155, 444(389,48),0]	0.48 CLB SLICES/DSP
	EMBEDDED RESOURCES	300.03	18.75	[142, 252(203,48),27]	
8	LOGIC RESOURCES	417.650	26.103	[236, 676(597,79),0]	1.1 CLB SLICES/DSP
	EMBEDDED RESOURCES	247.892	15.49	[189, 359(279,79),42]	

References

i. XilinxInc, "Product Specification: Xilinx LogiCore, Distributed Arithmetic FIR Filter v9.0", April 2005, http://www.xilinx.com/ipcenter/catalog/logicore/docs/da_fir.pdf

ii. Yajun Zhou, Pingzheng Shi, "Distributed Arithmetic for FIR Filter implementation on FPGA", International Conference on Multimedia Technology (ICMT)/IEEE, 2011: 294-297.

iii. Malik Umar Sharif, RabiaShahid, MarcinRogawski, Kris Gaj, "Use of Embedded FPGA Resources in Implementation of Five Round Three SHA-3 Candidates", in CRYPT II Hash Workshop 2011, May 2011.

iv. Tsao Y C, Choi K, "Area-Efficient Parallel FIR Digital Filter Structures for Symmetric Convolutions Based on Fast FIR Algorithm [J]", IEEE Transactions on Very Large Scale Integration (VLSI) System, 2010, PP(99): 1-5.

v. Chao Cheng, Keshab K Parhi, "Low Cost Parallel FIR Filter Structures with 2 stage Parallelism[J]", IEEE Transactions on Circuit and Systems I: Regular, 2007, 54(2): 280-290

vi. Peled, B. Liu, "A New Hardware Realization of Digital Filters", IEEE Transaction on Acoust., Speech. Signal Processing, Dec. 1974: 456-462

vii. S. A. White, "Applications of Distributed Arithmetic to Digital Signal Processing", IEEE ASSP Magazine, July 1989: 4-19

viii. ShahnamMirzaei,AnupHosangadi, PyanKastner, "FPGA Implementation of High Speed FIR Filters Using Add and Shift Method", ACM/SIGDA 14th International Symposium on Field programmable gate arrays, 2006

ix. The MathWorks, Inc., "Filter Design HDL Coder™: User's Guide (R2012b)", Retrieved September 2012 from http://www.mathworks.com/help/pdf_doc/hdlfilter/hdlfilter.pdf

x. Fabio Fabian Daitx, Vagner S. Rosa, Eduardo Costa, "VHDL Generation of Optimized FIR FIlters", International Conference on Signals, Circuits and Systems, 2008: 1-5

xi. National Instruments Corporation, "LabVIEW 2010 FPGA module: Introduction to DSP48E Slices (FPGA Module)", 2010