International Journal of Modern Trends in Engineering and Research
www.ijmter.com
e-ISSN No.:2349-9745, Date: 2-4 July, 2015

Implementation of DA based FIR 16 Order LPF using IP Core Generator

Anisha Unnikrishnan[1], Manisha Ghortale[2], Ankita Aher[3], Mayuri Joshi[4]

[1]*Electronics and Telecommunication Engineering, KKWIEER,* anishau26@gmail.com
[2]*Electronics and Telecommunication Engineering, KKWIEER,* mghortale@gmail.com
[3]*Electronics and Telecommunication Engineering, KKWIEER,* ankitaaher16@gmail.com
[4]*Electronics and Telecommunication Engineering, KKWIEER,* mpjoshi@kkwagh.edu.in

Abstract- This paper provides the principle of Distributed Arithmetic and introduces it into a FIR 16 order low-pass filter using IP core generator. The implementation of FIR filters on FPGA based on traditional method costs considerable hardware resources, which goes against the decrease of circuit scale and the increase of system speed. A new design and implementation of FIR filters using Distributed Arithmetic is provided to solve this problem. We can make a Look-Up-Table (LUT) to conserve the MAC values and callout the values according to the input data. The simulation results indicate that FIR filters using Distributed Arithmetic can work stable and can save almost less than 50 percent hardware recourses, and can be applied to a variety of areas for its great flexibility and high reliability. IP core generator methodology will reduce the time for design, generation, integration and testing of system.

Keywords— Distributed Arithmetic (DA), Field Programmable Gate Arrays (FPGA), Finite Impulse Response (FIR), Look-Up-Table (LUT), Intellectual Property (IP) core.

I. INTRODUCTION

Filters are a basic component of all signal processing and telecommunication systems. Filters are widely employed in signal processing and communication systems in applications such as channel equalization, noise reduction, radar, audio processing, video processing, and analysis of economic data. Digital filters are divided into two categories, including Finite Impulse Response (FIR) and Infinite Impulse Response (IIR).FIR filters are widely applied to a variety of digital signal processing areas for the virtues of providing linear phase and system stability.

The distributed algorithm is the way of computation to realize the multiply-accumulate operations. Compared with the traditional algorithm of multiply-accumulate operations, DA is different implementation of the order of the product operation. Multiply-accumulate function of distributed algorithm is that part in which product of each corresponding position is formed by adding the corresponding part of the plot by the input data, and then each part to accumulate, and form the final results. But the traditional method is to wait until the product has been produced, all come together, to complete the multiply-accumulate operations. Compared with the traditional algorithm, the distributed algorithm can greatly reduce the hardware circuit scale and improve the execution speed of circuit.

@IJMTER-2015, All rights Reserved

Core is a pre-defined and pre-verified complex functional block that is integrated into the designer's logic to save development time while focusing engineering time and energy on those parts of the design that add value to product.

II. FILTER DESIGN

In the course of FIR filters design, ringing can be generated at the edge of transition band for the reason that finite series Fourier transform cannot produce sharp edges. So windows are often used to produce suitable transition band, and Kaiser Window is widely used for providing good performance.

A 16-order FIR low-pass filter is designed using Kaiser Window. We can obtain the filter coefficients using Matlab. In Matlab data are described in the floating-point form while in FPGA system it is required in fixed point form. After quantizing the filter coefficients we can obtain the final coefficients as follows:

Coefficient Data
h(0) = h(15) = 09de ;
h(1) = h(14) = fc6d ;
h(2) = h(13) = f4cc ;
h(3) = h(12) = 0dd7;
h(4) = h(11) = 06db;
h(5) = h(10) = e0bb;
h(6) = h(9) = 102c;
h(7) = h(8) = 7f4e;

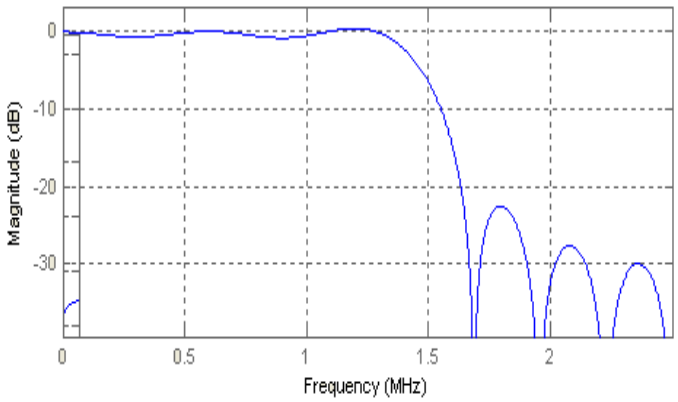

Figure1. Magnitude Response of Kaiser Filter

III. IP CORE GENERATOR

The IP-Core generator tool has been developed for automatic creation of an EDK-compatible core starting from a generic VHDL functionality.

The IP-Core generator tool can be described as constituted by three main steps:

a) *Input phase*: The tool needs the name and the VHDL description of the core.

b) *Reader process*: In this it computes the first part of the IP-Core generator tool:

– Parse the VHDL core description;

– Builds the signals list: A signal is recognized, analyzed and its information is stored in the signals list, this action is repeated until the end of the core is reached.

– Save the core name in a variable used by the following process.

@IJMTER-2015, All rights Reserved

c) *Writer:* It is characterized by the following actions:

– Accept the signals list, generated by the Reader, and the core name, provided by the user as its input.

– Create a stub file between the Core and the IP Core description VHDL files.

– Design the top architecture which is the final.

IP-Core according to the bus architecture chosen, OPB or PLB, the correct communication infrastructure is created to complete the generated stub file to realize the final IP-Core design.

IV. MAC CORE

A basic function of calculation of partial products and accumulation of it is done by the MAC unit. Every input sample is to be multiplied by every filter coefficient to obtain the output of fir filter described as:

$y = h_1 * x_1 + h_2 * x_2 + \ldots + h_k * x_k$

Where $h = [h_1, h_2, \ldots h_k]$ are filter coefficients which are constant

And $x = [x_1, x_2 \ldots x_k]$ is the input data stream which is variable.

So the minimum number of MAC unit required would be equal to the number of the filter coefficients which will put a severe restriction on filter order. With increase in MAC units there will be increase in memory requirement and decrease in speed of operation.

The MAC FIR core uses more MAC functional units to service the N sum-of-product calculations in the filter. The core automatically determines the minimum number of MAC engines required to meet the user specified throughput.

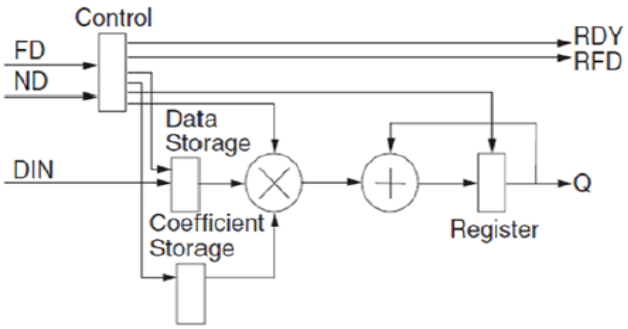

Figure2. Single MAC Block Diagram

Here, DIN is input data port. ND is New Data port enables loading of samples into MAC. RFD is ready for Data port used to receive new data.RDY is ready port which makes new output available. Registers are used to present final summation at output. Control input manages the flow of data and reading of coefficients. Coefficient Storage is flexible ROM for coefficient storage. Data Storage is the block memory to store data. Q is the filter output.

V. DA CORE

An FIR filter can be described by the following equation

$$y[n] = \sum_{k=0}^{K-1} h[k]x[n-k] \quad (1)$$

$$y = \sum_{k=0}^{K-1} h[k].x1[k] \quad (2)$$

Then we use B-bit two's complement binary number top represent the input data

@IJMTER-2015, All rights Reserved

$$x1[k] = -2^{B}.x_B[k] + \sum_{b=0}^{B-1} x_b[k].2^b \qquad (3)$$

Substitution of (3) into (2) yields:

$$\begin{aligned} y &= \sum_{k=0}^{K-1} h[k].(-2^{B}.x_B[k] + \sum_{b=0}^{B-1} x_b[k].2^b) \\ &= -2^{B}.\sum_{k=0}^{K-1} h[k].x_B[k] + \sum_{b=0}^{B-1} 2^b.\sum_{k=0}^{K-1} h[k].x_b[k] \\ &= -2^{B}.f\ (h[k],(x_B[k]) + \sum_{b=0}^{B-1} 2^b.f(h[k],x_b[k]) \qquad (4) \end{aligned}$$

In equation (4), we observe that the filter coefficients can be pre-stored in LUT, and addressed. The coefficient of FIR filter is h(n) which is been calculated by MATLAB. Using a lookup table structure of FPGA device, this constant multiplication can be stored and combining multiply-add, the convolution computation is realized. In this paper this is done using IP core logic Xilinx.

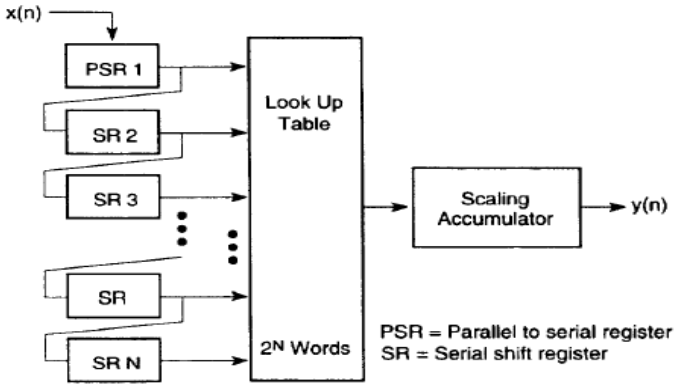

Figure 3. Distributed Arithmetic FIR filter

A simplified view of a DA FIR is shown in Figure 3. Input samples are presented to the input parallel-to-serial shift register. As the new sample is serialized, the bit-wide output is presented to a serial shift register which stores the input sample history in a bit-serial format and is used in forming the required inner-product computation. The nodes in the cascade connection of serial shift registers are used as address inputs to a look-up table. This LUT stores all possible partial products.

VI. SIMULATIONS AND RESULTS

A. MAC simulation

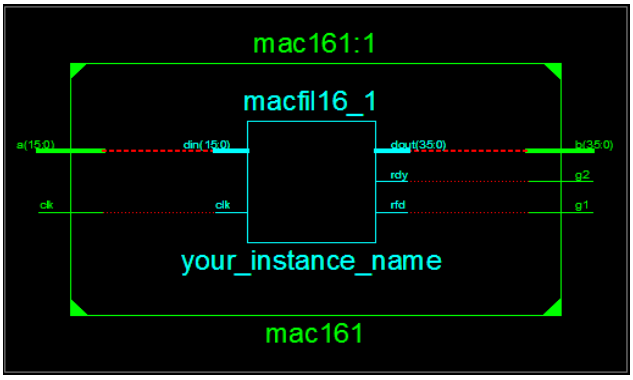

Figure4. RTL schematic of MAC

@IJMTER-2015, All rights Reserved

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	138	768	17%
Number of Slice Flip Flops	224	1536	14%
Number of 4 input LUTs	241	1536	15%
Number of bonded IOBs	55	97	56%
Number of MULT18X18s	1	4	25%
Number of GCLKs	1	8	12%

Figure5. Design summary of MAC

B. DA simulation

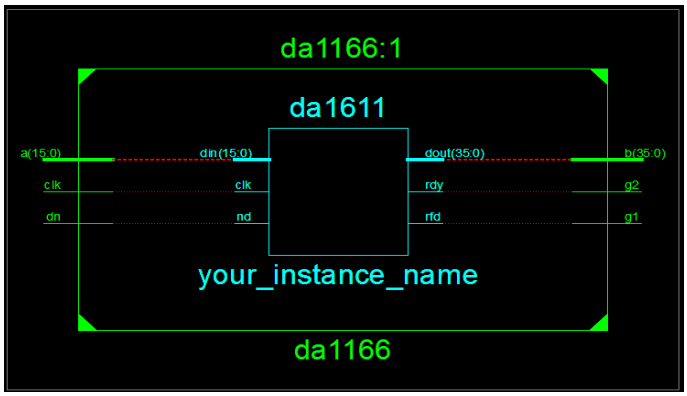

Figure6. RTL schematic of DA

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	102	768	13%
Number of Slice Flip Flops	191	1536	12%
Number of 4 input LUTs	136	1536	8%
Number of bonded IOBs	56	97	57%
Number of GCLKs	1	8	12%

Figure7. Design summary of DA

Results have been discussed in the design summary and explained design meets all the objectives.

VII. CONCLUSION

Thus the DA architectures reduce the memory usage. We have successfully implemented an efficient 16-tap DA filter, using both MAC and DA architecture using IP core generator in Xilinx12.2. The test results indicate that the designed filter using Distributed Arithmetic can work stable with high speed and can save almost less than 50 percent hardware recourses. Meanwhile, it is very easy to transplant the filter to other applications through modifying the order parameter or bit width and other parameters, and therefore have great practical applications in digit signal processing.

@IJMTER-2015, All rights Reserved

Table 1. Device utilization summary of MAC (estimated value)

Logic utilization	Used	Available	Utilization
No. of slices	138	768	17%
No. of slice flip-flops	224	1536	14%
No. of 4-input LUTs	241	1536	15%

Table 2. Device utilization summary of DA (estimated value)

Logic utilization	Used	Available	Utilization
No. of slices	102	768	13%
No. of slice flip-flops	191	1536	12%
No. of 4-input LUTs	136	1536	8%

After the comparison of both device utilization summary of MAC and DA method it is found that number of 4-input LUTs are reduced by 43.56%, number of slice flip-flops by 14.73%, number of slices by 26%.

VIII. FUTURE SCOPE

In this paper, we have designed a mid order filter so the LUT access and memory requirement is at ease. However as the filter order increases the LUT size increases exponentially. So for higher order filters the LUT memory requirement, accessing the values from it and the time for access pose a problem. The higher order FIR Filters are essential in various DSP Applications as their characteristic are closer to ideal filter which makes it more efficient. So the development here is the use of modified DA which uses pipeline register to reduce the access time and instead of using one LUT we make use of multiple smaller LUTS. The smaller LUTs along with combinational circuit of full adder, multiplexers and pipe register are known as Modified DA. Thus implementation of more efficient higher order FIR Filters using Modified DA involves the future work.

REFERENCES

[1] Baha Ali Nasir, "Efficient Programmable Finite Impulse Response Filter Using Xilinx MAC FIR Filter for Software Defined Radi"; International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-2, Issue-6, January 2014.
[2] Cui Guo-wei, Wang Feng-ying "The Implementation of FIR Low-pass Filter Based on FPGA and DA"; 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP)
[3] Fabrizio Ferrandi,Giovanna Ferrara, Roberto Palazzo, Vincenzo Rana, Marco D. Santambrogio, Marco D. Santambrogio, "VHDL to FPGA automatic IP-Core generation: A case study on Xilinx design flow".
[4] Les Mintzer, **"**Digital Filtering in FPGAs", Signals, Systems and Computers, 1994. 1994 Conference Record of the Twenty-Eighth Asilomar Conference (Volume:2).
[5]Xilinx.Inc.,LogiCore DS240 April 28, 2005.
[6]Xilinx.Inc.,IP LogiCore FIR Compiler v5.0,DS534 March 1, 2011.

@IJMTER-2015, All rights Reserved