International Journal of Scientific Research Engineering & Technology (IJSRET)
Volume 1 Issue4 pp 052-056 July 2012 www.ijsret.org ISSN 2278 – 0882

FPGA Implementation analysis of FIR Filter Over MAC, DA FIR Cores

Ms. Mausmi Verma
Shriramswaroop Memorial professional Colleges ,Lucknow
Mausmi.7nov@gmail.com

ABSTRACT

FIR filter realization based on Distributed Arithmetic approach are presented and compared with a conventional design based N tab (Multiply And Accumulate) MAC unit Finite Impulse Response (FIR) filters are of great importance in the digital signal processing (DSP) world. Their characteristics of linear phase and feed forward implementation make it very useful for building high performance filters.

. Field programmable gate array (FPGA) is the best solution for implementation of filters. Digital filters are approximation of actual analog filters .For the DA based realization the arithmetic operations are implemented by means of look-up tables, shift register and scalable accumulate and MAC based realization arithmetic operation implemented by means of multiplier, adder, delay unit.

Devices and implementation methods are having limitations. Here we have taken search for best results with different filter designing approaches. Distributed Arithmetic filter core is faster and gives best results with moderate use of device resource.

1. FIR FILTER

FIRs have the advantage of being much more realizable in hardware [12] because they avoid division and feedback paths. FIRs are dependent on the data coming through the filter and on past values Eq.(1) represents relationship between FIR filter coefficient and input sequence, which is finite in nature. e.q. (2) gives FIR filter presentation in Z transform. An FIR filter is completely specified by the following input-output relationship:

$$y(n) = \sum_{i=0}^{K} b_i x(n-i)$$
$$= b_0\, x(n) + b_1 x(n-1) + b_2 x(n-2) + \cdots + b_K x(n-K)$$

……(1)

Applying the z-transform on both sides

$$Y(z) = b_0 X(z) + b_1 z^{-1} X(z) + \cdots + b_K z^{-K} X(z).$$

……..(2)

We have the transfer function, which depicts the FIR filter,as

$$H(z) = \frac{Y(z)}{X(z)} = b_0 + b_1 z^{-1} + \cdots + b_K z^{-K}.$$

…….(3)

2. FIR FILTER DESIGN METHODS

The major advantages of using window method are their relative simplicity as compared to other methods and ease of use. The fact that well-defined equations are often available for calculating the window coefficients has made this method successful. Hence Kaiser window expressed in Eq.(4) is used in for calculating FIR filter coefficient.

$$W(n) = \frac{I_o(\beta\sqrt{1-(2(n+1)/(N+1))^2})}{I_o(\beta)} \qquad n = 0,1,\ldots,N\text{-}1$$
$$= 0 \qquad \text{otherwise}$$

……(4)

3. FIR FILTER REALIZATION

Given the transfer function of the FIR filter in Eq. (5), which is used for final realization of filter using DA Architecture.

$$H(z) = b_0 + b_1 z^{-1} + \ldots + b_K z^{-K},$$

……(5)

We obtain the difference equation as which consists of addition of partial product. Partial product obtained through multiplication process. This is an expected architecture to be implemented in FPGA.

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + \ldots + b_K x(n-K).$$

…….(6)

Realization of such a transfer function is direct form I form, displayed in Fig.1. Realization of FIR filter consists of order of the filter as 31 to optimize between transition band and main lobe width. Input sequence is 12 bit wide as well as output sequence is also 12 bit wide.

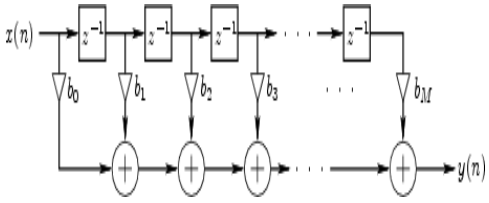

Fig. 1: Direct form I

4. DISTRIBUTED ARITHMETIC

A completely different FIR architecture is based on the distributed arithmetic concept. A distributed arithmetic (DA) realization [26, 27] is employed. This approach employs no explicit multipliers in the design, only look-up tables (LUTs), shift registers, and a scaling accumulator.

DA can be used to compute sum of products. Many DSP algorithms like convolution and correlation are formulated in a sum of products (SOP) fashion; a general purpose multiplication is not required. In case of filter implementation, if filter coefficients are constant in time, then the partial product term x[n]c[n] becomes multiplication with a constant. Consider the following sum of products:

$$y = \langle c, x \rangle = \sum_{n=0}^{N-1} c[n]x[n] = c[0]x[0] + c[1]x[1] + \ldots + c[N-1]x[N-1]$$

……..(7)

Further assume that the coefficients c[n] is known values and that the variable x[n] can be represented by

$$x[n] = \sum_{b=0}^{B-1} x_b[n] \times 2^b \quad \text{With } x_b[n], \in [0,1]$$

……(8)

Where $x_b[n]$ represents the bth bit position of the number's binary representation. The SOP can be represented as:

$$y = \langle c, x \rangle = \sum_{n=0}^{N-1} c[n] \sum_{b=0}^{B-1} x_b[n] \times 2^b$$

……..(9)

Expanding the summations yields to:

$$\begin{aligned} y = \langle c, x \rangle &= c[0] \times \left(x_{B-1}[0]2^{B-1} + x_{B-2}[0]2^{B-2} + \ldots x_0[0]2^0 \right) \\ &+ c[1] \times \left(x_{B-1}[1]2^{B-1} + x_{B-2}[1]2^{B-2} + \ldots x_0[1]2^0 \right) \\ &\vdots \\ &+ c[N-1] \times \left(x_{B-1}[N-1]2^{B-1} + x_{B-2}[N-1]2^{B-2} + \ldots x_0[N-1]2^0 \right) \end{aligned}$$

……...(10)

Redistributing the terms we have:

$$\begin{aligned} y = \langle c, x \rangle &= \left(c[0]x_{B-1}[0] + c[1]x_{B-1}[1] + \ldots c[N-1]x_{B-1}[N-1] \right) \times 2^{B-1} \\ &+ \left(c[0]x_{B-2}[0] + c[1]x_{B-2}[1] + \ldots c[N-1]x_{B-2}[N-1] \right) \times 2^{B-2} \\ &\vdots \\ &+ \left(c[0]x_0[0] + c[1]x_0[1] + \ldots c[N-1]x_0[N-1] \right) \times 2^0 \end{aligned}$$

…….…(11)

In more compact form:

$$y = \langle c, x \rangle = \sum_{b=0}^{B-1} 2^b \sum_{n=0}^{N-1} c[n] \times x_b[n]$$

……(12)

The key is to realize that the second summation can be mapped to a Look Up Table (LUT). The coefficients c[n] are known and the $x_b[n]$ values are either 1 or 0 then each SOP is just a combination of the c[n]'s for which a true table can be constructed.
Suppose we have:

$$(c[0]x_{B-2}[0] + c[1]x_{B-2}[1] + \ldots c[N-1]x_{B-2}[N-1]) \times 2^{B-2}$$

Where each xB-2 digit belongs to a different x[n] variable; nevertheless we can form an N bit word that can take 2N values.

Multiplication by a power of 2 is no more that a bit shift, so what need to do is to slice and concatenate the bits of the different x[n] in order to build a table given that the c[n] are all known.

What is left is to show how we can deal with signed implementations of DA. A minor modification needs to be introduced when working with signed two's complement numbers. In two's complement, the MSB is used to determine the sign of the number. We use, therefore, the following B-bit representation $x[n] = -2^{B-1} \times x_{B-1}[n] + \sum_{b=0}^{B-2} x_b[n] \times 2^b$

Then, the output y[n] is defined by:

$$y[n] = -2^{B-1} \times \sum_{n=0}^{N-1} c[n] \times x_{B-1}[n] + \sum_{b=0}^{B-2} 2^b \times \sum_{n=0}^{N-1} c[n] \times x_b[n] \ldots$$

…….(13)

Finally, a block diagram for the DA implementation of a FIR filter is shown in Fig.2

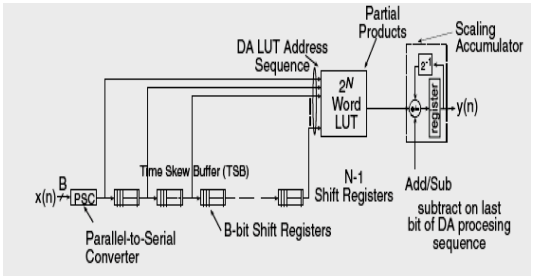

Fig. 2: DA architecture with lookup table.

The hardware description language (HDL) specification of the lookup table can be easily obtained for filter described by its c[i] coefficients. Since the size of lookup tables grows exponentially with the number of inputs the efficient implementation of these blocks becomes crucial to final resource utilization of filter implementation.

5. DSP FPGA'S

FPGA's are devices with highest gate count and pool of flip flops, on chip memory support and special purpose logic blocks. The distributed interconnection logic is advantageous in implementing complex logic blocks. Hence FPGA's are most suitable devices for implementing digital signal processing algorithms.

Xilinx and Altera are providing wide range of devices and development tools for implementation of DSP algorithm. They give specialized DSP supportive logic blocks on chip. Implementation of digital filters on programmable logic chips will need parallel multipliers and adders for faster results. It is also called as multiply and Accumulate unit i.e. MAC unit. If filters will implement on FPGA which is not having MAC units, it will take lot of glue logic on FPGA for implementation of multipliers and adders. In Xilinx, Spatan3 devices provide on chip 18 X 18 multipliers.

The Spartan-3A DSP family builds on the success of the Spartan-3A FPGA family by increasing the amount of memory per logic and adding XtremeDSP™ DSP48A slices. New features improve system performance and reduce the cost of configuration. These Spartan-3A DSP FPGA enhancements, combined with proven 90 nm process technology, deliver more functionality and bandwidth per dollar than ever before, setting the new

standard in the programmable logic and DSP processing industries. The Spartan-3A DSP FPGAs extend and enhance the Spartan-3A FPGA family. The XC3SD1800A and the XC3SD3400A devices are tailored for DSP applications and have additional block RAM and XtremeDSP DSP48A slices. The XtremeDSP DSP48A slices replace the 18x18 multipliers found in the Spartan-3A devices and are based on the DSP48 blocks found in the Virtex™-4 devices.The block RAMs are also enhanced to run faster by adding an output register. Similarly Virtex -4 SX and Virtex-5SX devices also deliver high performance for performance centric or multi-channel applications for DSP designs that demand a high ratio of DSP to logic. They offer more DSP algorithms and higher levels of DSP integration than were available in prior programmable devices, while delivering low power consumption and efficient silicon utilization. Up to 512 XtremeDSP DSP48 slices operating at 500 MHz help you solve complex challenges such as implementing hundreds of IF-to-baseband down-conversion channels.

6. FILTER IMPLEMENTATION:

Considering above constraints the FIR filter is implemented using Matlab and Xilinx ISE tool chain. First the FIR Filter is designed in multiply and Accumulate form using Filter design and analysis tool in Matlab.

MAC Filter:
For the given parameters the filter coefficients are obtained. The digital filter designed is coded in hardware description language using HDL coder. It generates VHDL code for the given filter. The coefficients are used in the fixed point binary format. After this the filter program is opened in the Xilinx ISE software project and synthesized for Sparten3 FPGA obtaining hardware details and operational speed and frequency details. It is observed that it takes considerable part of logic provided and routing resources for implementation of multipliers and adders.

N tap FIR filter:
The same filter is implemented using Matlab Simulink and Xilinx System generator environment. Here we have used Xilinx n-tap FIR filter core. The Filter coefficients are obtained from the Filter design and analysis tool and imported in the simulink n tap filter core. The project is simulated and generated using Xilinx System generator. The generated project is opened and synthesized in Xilinx ISE environment. The device utilization and operational speed details are obtained. We got that it takes optimum hardware than the previous implementation. The operational speed is also getting improved. This method gives improved results over the previous method.

DA FIR Filter:
Now the same filter problem is implemented using DA FIR filter. Here the filter is designed using Matlab Simuink and Xilinx system generator. The filter coefficients are obtained using Filter Design And Analysis Tool and imported in DA FIR core. Simulation is performed and Xilinx project is generated using Xilinx System generator. This project is opened in Xilinx ISE tool and synthesis is performed. The device utilization is more and operational speed get improved. The results obtained are mentioned in the following table.

7. RESULT ANALYSIS:

The low pass filter is designed with three different methods such as filter using multiply and accumulate method, N tap fir filter core and using Distributed arithmetic core. The filters are synthesized for the Sparte3 FPGA device It has been observed that filters designed with multiply and accumulate unit utilizes more logic from the FPGA. N tap filter core uses less logic than both methods. But the delay in the DAFIR filter is less as compared to other filter methods. It means the DAFIR filter is faster as compared to other filters.

8. CONCLUSION

All above filters are implemented for the Xilinx Sparten3 FPGA. Sophisticated devices which are providing on chip DSP supporting hardware will give optimized results The Filter are implemented on FPGA board. It has the facilities for audio and external input. It also has facility for audio output and external output connection. The audio input is given to each filter and

for variable number of taps. The quality of sound gets improved for higher number of taps.

For sharper cutoff frequencies and for fine results number of taps should be higher. For above particular design methods we got following results. For less number of taps the MAC method is best method. It generates less hardware and gives good operational speed. For moderate number of taps for example 10 and 20 taps N tap Fir method is suitable. In such cases the DA FIR filter will be suitable. The filter design with DAFIR core is faster than filter designed with other methods.

9. REFERENCES

[1] Shantanu Das, "Functional fractional calculus for system identification and controls", Springer, Berlin, 2008.

[2] B.T. Krishna, "Studies on fractional order differentiators and integrators: a survey", Signal Processing, vol. 91, no. 3, pp. 386-426, March 2011.

[3] Guillermo E. Santamaria, Jose V. Valverde, Raquel Perez-Aloe and Blas M. Vinagre, "Microelectronic implementations of fractional-order integrodifferential operators", ASME Journal of Computational and Nonlinear Dynamics, vol. 3, no. 2, pp. 021301, April 2008.

[4] Xilinx Inc., http://www.xilinx.com

[5] Bull, D.R., and Horrocks, D.H.: 'Primitive operator digital filters', IEE Proc. G, Circuits Devices Syst., 1991, 138, (3),pp. 401–412

[6] Dempster, A.G., and Macleod, M.D.: 'Use of minimumaddermultiplier blocks in FIR digital filters', IEEE Trans.Circuits Syst. II, Analog Digit. Signal Process., 1995, 42, (9),pp. 569–577

[7] Meyer-Baese, U.: 'Digital signal processing with field programmablegate arrays' (Springer-Verlag, Berlin, Heidelberg, 2001)

[8] Wirthlin, M.J., and McMurtrey, B.: 'Efficient constant coefficient multiplication using advanced FPGA architectures'. Proc. 11th Int.Workshop on Field-Programmable Logic and Applications, 2001,pp. 555–564

[9] Xilinx Inc.: 'Distributed arithmetic FIR filter v8.0', http://www.xilinx.com

[10] Synplicity Inc., http://www.synplicity.com

[11] Actel Corp., ProASIC Plus Family Flash FPGAs, v3.5, April 2004

[12] Altera Corp., Stratix II Device Handbook, February 2004

[13] Lattice Semiconductor Corp, ispXPGA Family, January 2004

[14] Xilinx, Inc., Virtex II Datasheet, June 2004

[15] G. Goslin, "A Guide to Using {FPGAs for Application-Specific Digital Signal Processing Performance," Xilinx Application Notes,1995.

[16] S. Hauck, S. Burns, G. Borriello, and C. Ebeling, "An FPGA for

[17] Implementing Asynchronous Circuits," IEEE Design and Test of Computers, vol. 11, no. 3, pp. 60-69, 1994.

[18] B. Von Herzen, "Signal Processing at 250 Mhz Using High-Performance FPGAs,"

[19] K.N. Macpherson , R.W. Stewart "Area efficient FIR filters for high speed FPGA Implementation" IEE Proc.-Vis. Image Signal Process., Vol. 153, No. 6, December 2006

[20] John Teifel, Rajit Manohar, IEEE Transaction on Computers, VOL. 53, NO. 11, NOVEMBER 2004