International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 2, Issue 12 (August 2013) PP: 05-10

FPGA Implementation of High Speed FIR Filters and less power consumption structure

[1]Deepak Shankhala, [2]Anoop Kumawat
[1](Lecturer, Electronics & Communication Engineering Department
JECRC UDML College of Engineering, JECRC Foundation, Jaipur (Rajasthan)
[2](Lecturer, Electronics & Communication Engineering Department
JECRC UDML College of Engineering, JECRC Foundation, Jaipur (Rajasthan)

Abstract: *FPGAs are being increasingly used for a variety of computationally intensive applications, mainly in the realm of Digital Signal Processing (DSP) and communications. This paper describe the development of FIR filter on Field programmable gate array (FPGAs) using Xilinx. FIR filter has been designed and realized by FPGA for filtering the digital signal because the term digital filter arises because these filters operate on discrete-time signals. So to designing The FIR filter the coefficient are generate by using MAT Lab FDATOOL and making to delay with signal Xilinx XC3S400FPGA(ISE 13.1) is considered. Process is done by taking the 16 Bit signed data for input and 38 bit for output (6 Bit extra). To verify the designed with proper power consumption output simulation, compilation and synthesis have been done. Field-Programmable gate Array (FPGA) has become an extremely cost-effective means of off-loading computationally intensive digital signal processing algorithms to means of off-loading computationally intensive digital signal processing algorithms to the dedicated hardware resources can effectively achieve application-specific integrated circuit (ASIC)-like performance while reducing development time cost and risks.*
Keywords: *FIR filter, FPGA, VHDL code, MATLAB.*

I. INTRODUCTION

For Designing FIR filtre By Frequency Sampling Method On FPGA for filtering the digital signal .This technique can be used any FPGA Families. Now a day we can write any filter designing code can be implement this code on hardware by using the optical link for required processing. In FPGA which is used thousand of memory element and gate can be process easily and fast this code. The reality is that today digital systems are designed by writing software in the form of hardware description languages (HDLs). Computer-aided design tools are used to both simulate VHDL design and to synthesize the design to actual hardware.

Due to rapid increases in the technology, current generation of FPGAs contain a very high number of Configureurable Logic Blocks (CLBs), and are becoming more feasible for implementing a wide range of applications. The high non recurring engineering (NRE) costs and long development time for ASICs are making FPGAs more attractive for application specific DSP solutions. DSP functions such as FIR filters and transforms are used in a number of applications such as communication and multimedia. These functions are major determinants of the performance and power consumption of the whole system. Therefore it is important to have good tools for optimizing these functions.

The designing of an FIR filter (with less power and less taking time) in VHDL with MATLAB (For the generation of coefficient of filter)and programming it on to an FPGA is explained in this paper Implementation of code. VHDL and MATLAB and basic digital filter equation concept are used.

II. FPGA IMPLEMENTATION OF HIGH SPEED FIR FILTERS

A method for implementing high speed Finite Impulse Response (FIR) filters using just registered adders and hardwired shifts. We extensively use a modified common sub expression elimination algorithm to reduce the number of adders. We target our optimizations to Xilinx Virtex III devices where we compare our implementations with those produced by Xilinx ISE 3.1 using Distributed Arithmetic.

III. METHAMATICAL ANALYSIS OF FIR FILTER

DSP functions such as FIR filters and transforms are used in a number of applications such as communication and multimedia. These functions are major determinants of the performance and Power consumption of the whole system. Therefore it is important to have good tools for optimizing these functions.

Equation (I) represents the output of an L tap FIR filter, which is the convolution of the latest L input samples. List the number of coefficients h(k) of the filter, and x(n) represents the input time series.

$$y[n] = \sum h[k]\, x[n-k] \qquad k = 0, 1, ..., L-1 \qquad (I)$$

The conventional tapped delay line realization of this inner product is shown in Figure 5.1. This implementation ran slates to L multiplications and L-1 additions per sample to compute the result. This can be implemented using a single Multiply Accumulate (MAC) engine, but it would require L MAC cycles, before the next input sample can be processed. Using a parallel implementation with L MACs can speed up the performance L times.

A general purpose multiplier occupies a large area on FPGAs. Since all the multiplications are with constants, the full flexibility of a general purpose multiplier is not required, and the area can be vastly reduced using techniques develop for constant multiplication.Though most of the current generation FPGAs such as Virtex II have embedded multipliers to handle these multiplications, the number of these multipliers is typically limited.

Furthermore, the size of these multipliers is limited to only 18 bits, which limits the precision of the computations for high speed requirements. The ideal implementation would involve a sharing of the Combinational Logic Blocks (CLBs) and these multipliers. In this paper, we present a technique that is better than conventional techniques for implementation on the CLBs.

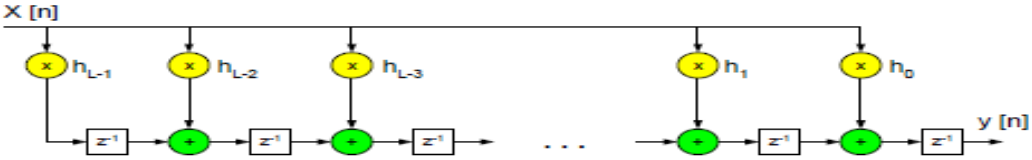


Figure5.1. A FIR filter block diagram

An alternative to the above approach is Distributed Arithmetic (DA) which is a well known method to save resources. Using DA method, the filter can be implemented either in bit serial or fully parallel mode to trade bandwidth for area utilization. Assuming coefficients c[n] are known constants, equation (I) can be rewritten as follows: $y[n] = \sum c[n] \cdot x[n]$ n = 0, 1, …, N-1

(II) Variable x[n] can be represented by:

$$x[n] = \sum x_b[n] \cdot 2^b \qquad b=0, 1, \ldots, B-1 \qquad \text{(III)}$$

$x_b[n] \in [0, 1]$

where $x_b[n]$ is the b^{th} bit of x[n] and B is the input width. Finally, the inner product can be rewritten as follows: $y = \sum c[n] \sum x_b[k] \cdot 2^b$

$y = c[0](x_{B-1}[0]2^{B-1} + x_{B-2}[0]2^{B-2} + \ldots + x_0[0]2^0) + c[1](x_{B-1}[1]2^{B-1} + x_{B-2}[1]2^{B-2} + \ldots + x_0[1]2^0) + \ldots\ldots + c[N-1](x_{B-1}[N-1]2^{B-1} + x_{B-2}[0]2^{B-2} + \ldots + x_0[N-1]2^0)$

$y = (c[0]x_{B-1}[0] + c[1]x_{B-1}[1] + \ldots + c[N-1]x_{B-1}[N-1])2^{B-1} + (c[0]x_{B-2}[0] + c[1]x_{B-2}[1] + \ldots + c[N-1]x_{B-2}[N-1])2^{B-2} + \ldots.. + (c[0]x_0[0] + c[1]x_0[1] + \ldots + c[N-1]x_0[N-1])2^0$

$$y = \sum 2^b \sum c[n] \cdot x_b[k] \qquad \text{(IV)}$$

where n=0, 1, …, N-1 and b=0, 1, …, B-1

The coefficients in most of DSP applications for the multiply accumulate operation are constants. The partial products are obtained by multiplying the coefficients c_i by multiplying one bit of data x_i at a time in AND operation. These partial products should be added and the result depend only on the outputs of the input shift registers.

The AND functions and adders can be replaced by Look Up Tables (LUTs) that gives the partial product. This is shown in Figure 5.2. Input sequence is fed into the shift register at the input sample rate. The serial output is presented to the RAM based shift registers (registers are not shown in Figure for simplicity) at the bit clock rate which is n+1 times (n is number of bits in a data input sample) the sample rate.

The RAM based shift register 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 over the time.

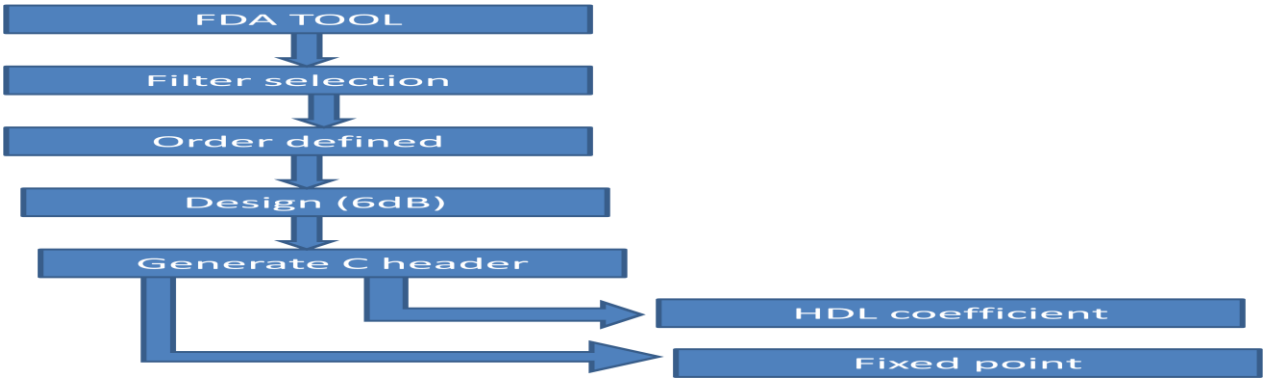
For an n bit input, n+1 clock cycles are needed for a symmetrical filter to generate the output

IV. STEP OF CODING DESIGNING AND PARAMETER OF FIR FILTER

Step 1: Coefficient generate through the MATLAB
Step 2: Decide the level (Till 36), product and sum 0 to 31
Step 3: Port mapping (Input and output)
filter input 16 bit
filter output 32 bit
Step 4: Coefficient declaration
Step 5: Signal declaration (for the product)
Step 6: Addition and subtraction
Step 7: Delay pipe line process 0 to 80 coefficient
Step 8: Multiply output save as product
Step 9: Resize (match with previous value)

IV.1 Coefficient selection step by using MATLAB

The window, optimal and frequency sampling method are the most commonly used for designing filter coefficient. In this paper the frequency sampling method is used for designing the FIR filter.. Flow chart in the Figure 4.1 shows the generation of the coefficient of the FIR filter generated through MATLAB software. As seen in the flow chart, coefficients generation requires three inputs, first is the starting and ends of the band, second is the input sampling frequency and finally the order of the filter. The coefficient generated for the eight stages FIR filter so the order of the filter is sixteen



4.1 Filter Coefficient Design

IV.1.1 Frequency Sampling Method: The frequency sampling method allows us to design nonrecursive FIR filter for both standard frequency filters (low pass, high pass & band pass filter) and filter with arbitrary y frequency response. A unique attraction of the frequency sampling method is that it also allows recursive implementation of FIR filters.

IV.1.2 No recursive frequency sampling:To obtain the FIR coefficients of the filter whose frequency response is depicted in Figure 2.3 By taking N samples the frequency response at intervals of Kfs/N, k=0,1,……N-1.
The filter coefficients can be obtained as inverse DFT of frequency samples.

$$h(n) = 1/N \sum_{k=0}^{n-1} H(k) e^{j(2\pi/N)nk}$$

Where *H(k)* k = 0, 1, 2,""""".., N-1, are sample of the ideal frequency response. The impulse response coefficients of linear phase FIR filter with positive symmetry, for N even, can be expressed as:

$$h(n) = 1/N \sum_{k=1}^{\frac{N}{2}-1} 2|H(k)|\cos[2\pi k(n-\alpha)/N] + H(0)$$

Where " = (N-1)/2, and H(k) are the samples of the frequency response of the filter taken at intervals of k Fs/N. For N odd, the upper limit in the summation is (N-1)/2.The resulting filter will have exactly the same frequency response as the original response at the sampling instants. To obtain a good

approximation to the desired frequency response, a sufficient number of frequency samples must be taken.
An alternative frequency sampling filter, know as type 2, results if frequency
Sample taken at intervals of
f_k = (k+1/2)Fs/N. k = 0,1,2,3……..,N-1

To improve the amplitude response of frequency samples in the wider transition, introducing frequency samples in the transition band. For a low pass filter the stop band attenuation increases, approximately, by 20 dB for each transition band frequency sample, with a corresponding increase in the transition width:
Approximate stopband attenuation (25+20M) dB
Approximate transition width (M+1)Fs/N
Where M is the number of transition band frequency samples and N is the filter length.

IV.1.3 Recursive frequency sampling: Recursive forms of the frequency sampling offer significant computational advantages over the no recursive forms if a large number of frequency samples are zero valued. The transfer function of an FIR filter, *H (z)* can be expressed in a recursive form:

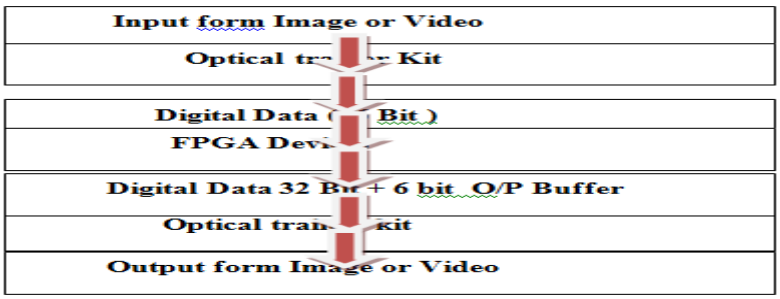
$$\mathrm{H(z)} = 1 - Z^{-N}/N \sum_{k=0}^{n-1} \mathrm{H(k)}/\, Z^{-1}\, e^{\frac{j2\pi k}{N}}$$

Thus in recursive form, H(z) can be viewed as a cascade of two filters: a comb filter, *H(z)* zeros uniformly distributed around the unit circle, and a sum of N which has N single all-pole filters,H(z) The zero of comb filter and the poles of the single pole filters are coincide on the unit circle at points .Thus the zero cancel the pole, making H(z) an FIR as it effectively has no poles.
In practice, due to finite word length effects the poles of H_2 (z) not to be located exactly on unit circle so that they are not cancelled by the zeros, making H(z) IIR and potentially unstable. Stability problems can be avoided by sampling H(z) at a radius, r, slightly less than unity. Thus the transfer function in this case becomes

$$\mathrm{H(z)} = 1 - r^N Z^{-N}/N \sum_{k=0}^{n-1} \mathrm{H(k)}/\; 1 - \; rZ^{-1}\, e^{\frac{j2\pi k}{N}}$$

In general, the frequency samples, H(k) are complex. Thus direct implementation requires complex arithmetic. To avoid this, the symmetry inherent use in frequency response of any FIR filters with real impulse, . So above equation can expressed as

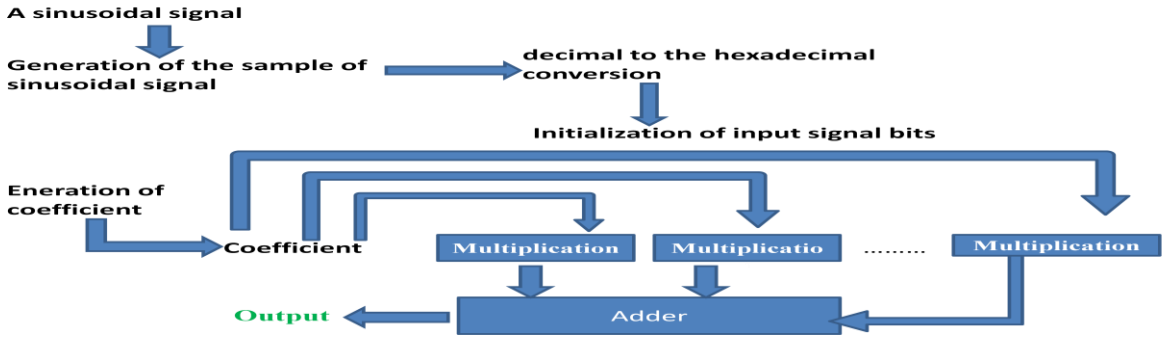
V. FPGA IMPLEMENTATION STEP



The coefficient and input sinusoidal signal is generated in the MATLAB and it converted into hexadecimal values by digitization. The hexadecimal values of coefficient and signal are proceeded to obtain the desire output through the multiplication and addition is done with IP cores which is inbuilt in Xilinx 10.1 software. Figure 7 shows the design flow of the entire process of FIR filter implementation on FPGA through VHDL coding done in Xilinx ISE design suit 10.1 versions.

5.b Flow chart for coding

This flow show you can implement the logic in coding



VI. COMMENT ON FLOW CHART OF CODING

A multiplier less technique, based on the add and shift method and common sub expression elimination for low area, low power and high speed implementations of FIR filters. We validated our techniques on Virtex IITM
devices where we observed significant area and power reductions over traditional Distributed Arithmetic based techniques. In future, we would like to modify our algorithm to make use of the limited number of embedded multipliers available on the FPGA devices.

VII. HARDWARE PERFORMANCE

The system above was implemented on a Xilinx Virtex II Pro FPGA (part number XCVP50) on a Cray XD1. The FPGA synthesized system ran at 160 MHz. FPGAs contain a certain amount of logic (AND, OR, etc.) and memory (block RAM) on chip. Any design is converted to a circuit that is programmed on the FPGA. Our circuit used 69% of the logic and 75% of the memory on the FPGA. The algorithm was also implemented in Matlab and in C.

VIII. RESULTS

VIII.1 Device

Table 8.1

Name	Type
Family	Virtex4
| Part	xc4vsx35
| Package	ff668
Grade	Commercial
Process	Typical
Speed Grade	-10

VIII.2 Default Activity Rates

Table 8.2

FF Toggle Rate (%)	12.5
| I/O Toggle Rate (%)	12.5
| Output Load (pF) (%)	5
| I/O Enable Rate (%)	100
| BRAM Write Rate (%)	50
| BRAM Enable Rate (%)	25
| DSP Toggle Rate (%)	12.5

VIII.3 Summary

8 .3.1 An On-Chip Power Summary

Table 8.4

On chip	Power(mw)	used	available	Utilization
Clocks	36.54	1	---	---
Logic	0	87	2506	30720
Signals	0.00	11309	---	---
IOs	0.00	| 124	448	28
DSPs	0.00	118	192	61
Quiescent	439.52	-	-	-
Total	476.06	-	-	-

VIII.3.2 B Thermal summary

Table 8.5

Thermal	NEEDED
Effective TJA (C/W)	8.31.0
Junction Temp (C	54
Max Ambient (C)	83

VIII .3.3 C Power Supply Summary

Power Supply Summary

Table 8.6

	Total	Dynamic	Quiescent
Total power (mw)	476.06	36.54	439.52

VIII.3.4 D Power Supply Currents

Table 8.7

Supply source	Supply voltage	Total current (mA)	Dynamic Current(mA)	Quiescent Current
Vcc int	1.200	248.28	30.45	217.83
Vcc aux	2.500	70.00	0.00	70.00
Vcco 25	2.500	1.25	0.00	1.25

IX. CONCLUSION

This paper has discussed an effective method for designing FIR filter of isolated less power consume and less delay time. It presents a parallel designing of filter for image recognition recent years there has been a steady movement towards the development of image recognigation technologies to replace or enhance text input called as have Mobile, video Search Applications. Recently NASA is working search applications. Future work can include improving the recognition filter design of the individual image reorganization by combining the multiple classifiers. Matched filters are designed to extract the maximum SNR of a signal that is buried in noise.

REFERENCES

[1]. Burrus C S, #Digital Filters Structures described by Distributed arithmetic$, IEEE Transactions on Circuits and Systems, vol. CAS-24, page: 12, December 1977.

[2]. Parhi K K., #A Systematic Approach for Design of Digit-serial Signal Processing Architectures$, Circuits and Systems, 1991.

[3]. Chapman S. J., # Matlab Programming for Engineers$, 3rd Edition, Thomson learning 2005.

[4]. #An Introduction to Digital Filters$ by INTERSIL, Application Note, January 1999.

[5]. Ifeachor E.C., Jervis B.W., #Digital Signal Processing$, 2nd Edition, Low Price Edition 2007.

[6]. http://www.Xilinx.com/bvdocs/whitepapers/wp116.pdf

[7]. Lee H., Sobelman G E. #Performance Evaluation and Optimal design for FPGA- based Digit-serial DSP Functions$. Computers and Electrical Engineering 29, 2003.

[8]. #FPGA Architect - XilinxXC4000/Spartan$ by ELANIX Inc.

[9]. Prokis J. G., Manolakis D. G., #Digital Signal Processing$, 3rd Edition, PHI publication 2004.

[10]. Deepak Sankhala , International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 www.ijert.org Vol. 2 Issue 6, June – 2013.