International Journal of Engineering Research and General Science Volume 2, Issue 4, June-July, 2014
ISSN 2091-2730

Design of a High Speed FIR Filter on FPGA by Using DA-OBC Algorithm

Vijay Kumar Ch[1], Leelakrishna Muthyala[1], Chitra E[2]

[1]Research Scholar, VLSI, SRM University, Tamilnadu, India

[2]Assistant Professor, ECE, SRM University, Tamilnadu, India

E-mail- leelakrishna424@gmail.com

Abstract: The main objective of the project is to implement FIR filter on FPGA using Distributed Arithmetic-Offset Binary Coding (DA-OBC) reduction technique. Digital filtering algorithms are most commonly implemented using general purpose digital signal processing chips for audio applications, or special purpose digital filtering chips and application- specific integrated circuits (ASICs) for higher rates. This paper describes an approach to the implementation of digital filter algorithms based on field programmable gate arrays (FPGAs). Implementing hardware design in Field Programmable Gate Arrays (FPGAs) is a formidable task. There is more than one way to implement the digital FIR filter. Based on the design specification, careful choice of implementation method and tools can save a lot of time and work. Mat Lab is an excellent tool to design filters. There are toolboxes available to generate VHDL descriptions of the filters which reduce dramatically the time required to generate a solution. Time can be spent evaluating different implementation alternatives. Computation algorithms are required that exploit the FPGA architecture to make it efficient in terms of speed and/or area. By using this algorithm, memory size can be reduced and also the speed of the operation can be increased. The FIR filter can be simulated on FPGA device by VHDL language in MODEL SIM and simulation on MATLAB is in this project

Key Words: DA-OBC ALGORITHIM

1. INTRODUCTION

The most common approaches to the implementation of digital filtering algorithms are general purpose digital signal processing chips for audio applications, or special purpose digital filtering chips and application-specific integrated circuits (ASICs) for higher rates[1] . This project describes an approach to the implementation of digital filter algorithms on field programmable gate arrays (FPGAs).Digital filters are basic units in many digital signals processing system. Finite-impulse-response (FIR) filters are basic processing elements in applications such as video signal processing and audio signal processing. There are two kinds of the realization method of FIR Filter at present. One is hardware implementation used the chips such as digital signal processor(DSP) Application-Special Integrated Circuit (ASIC) and field programmable gate arrays (FPGA),The other is software implementation used advanced language such as C/C++, MATLAB. Implementing hardware design in Field Programmable Gate Arrays (FPGAs) is a formidable task [2]. There is more than one way to implement the digital FIR filter. Based on the design specification, careful choice of implementation method and tools can save a lot of time and work. Mat Lab is an excellent tool to design filters. There are toolboxes available to generate VHDL descriptions of the filters which reduce dramatically the time required to generate a solution [3]. Time can be spent evaluating different implementation alternatives. Proper choice of the computation algorithms can help the FPGA architecture to make it efficient in terms of speed and/or area. The design method of multiplication and accumulation (MAC) operation is the core of FIR filter implementation. The method can be classified in two categories generally. One is direct- multiply structure, which are expensive in hardware because of logic complexity and area usage. The other is Distributed Arithmetic (DA) [1], which convert calculation of MAC to a serious of look-up table accesses and summations. This solution especially suited for LUT-based FPGA architectures and can greatly improve the speed of operation. Distributed arithmetic (DA) is commonly used for signal processing algorithms where computing the inner product of two vectors comprises most of the computational workload. This type of computing profile describes a large portion of signal processing algorithms, so the potential usage of distributed arithmetic is tremendous.

The inner product is commonly computed using multipliers and adders. When computed sequentially, the multiplication of two B-bit numbers requires from B=2 to B additions, and is time intensive. Alternatively, the multiplication can be computed in parallel using B=2 to B adders, but is area intensive. Whether a K-tap filter is computed serially or in parallel, it requires at least B=2 additions per multiplication plus the (K – 1) additions for summing the products together. In the best case scenario, K - (B + 2) =2 -1 additions are needed for a K-tap filter using multipliers and adders. A competitive alternative to using a multiplier is distributed arithmetic [4]. It compresses the computation of a K-tap filter from K multiplications and K-1 additions into a memory table and generates a result in B-bit time using B-1 additions. DA significantly reduces the number of additions needed for Filtering [1]. This reduction is particularly noticeable for filters with high bit precision. This reduction in the computational workload is a result of storing the pre computed partial sums of the filter coefficients in the memory table. When compared with other alternatives, distributed arithmetic requires fewer arithmetic computing resources and no multipliers. This aspect of distributed arithmetic is a favorable one for computing environments with limited computational resources, especially multipliers [3]. This type of computing environments can be found on older field-programmable gate arrays (FPGAs) and low-end, low-cost FPGAs. By using distributed arithmetic, these types of devices can be used for low latency, area constrained, high-order filters. Implementing such a filter using a multiplier based approach would be difficult.

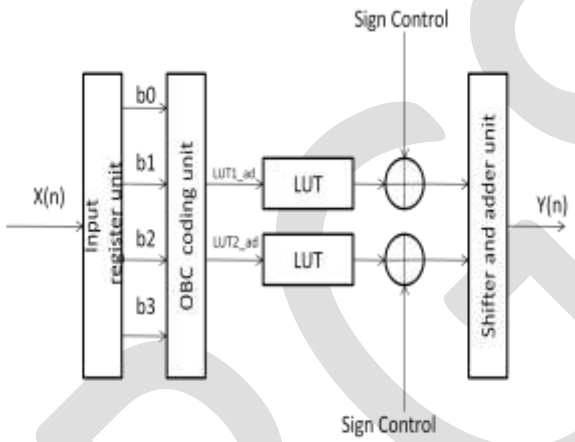
Digital Filters

Digital filters are used extensively in all areas of electronic industry. This is because Digital filters have the potential to attain much better signal to noise ratios than analog filters and at each intermediate stage the analog filter adds more noise to the signal, the digital filter performs noiseless mathematical operations at each intermediate step in the transform[5]. As the digital filters have emerged as a strong option for removing noise, shaping spectrum, and minimizing inter-symbol interference in communication architectures. These filters have become popular because their precise reproducibility allows design engineers to achieve performance levels that are difficult to obtain with analog filters. FIR and IIR filters are the two common filter forms. A drawback of IIR filters is that the closed-form IIR designs are preliminary limited to low pass, band pass, and high pass filters, etc. Furthermore, these designs generally disregard the phase response of the filter. For example, with a relatively simple computational procedure we may obtain excellent amplitude response characteristics with an elliptic low pass filter while the phase response will be very nonlinear. In designing filters and other signal-processing system that pass some portion of the frequency band undistorted, it is desirable to have approximately constant frequency response magnitude and zero phases in that band [2]. For casual systems, zero phases are not attainable, and consequently, some phase distortion must be allowed. As the effect of linear phase with integer slope is a simple time shift. A nonlinear phase, on the other hand, can have a major effect on the shape of a signal, even when the frequency-response magnitude is constant. Thus, in many situations it is particularly desirable to design systems to have exactly or approximately linear phase. Compare to IIR filers, FIR filters can have precise linear phase. Also, in the case of FIR filters, closed-form design equations do not exist. While the window Method can be applied in a straightforward manner, some iteration may be necessary to meet a prescribed specification. The window method and most algorithmic methods afford the possibility of approximating more arbitrary frequency response Characteristics with little more difficulty than is encountered in the design of low pass filters [4]. Also, it appears that the design problem for FIR filters is much more under control than the IIR design problem because there is an optimality theorem for FIR filters that is meaningful in a wide range of practical situations. The magnitude and phase plots provide an estimate of how the filter will perform; however, to determine the true response, the filter must be simulated in a system model using either calculated or recorded input data. The creation and analysis of representative data can be a complex task. Most of the filter algorithms require multiplication and addition in real-time. The unit carrying out this function is called MAC (multiply accumulate). Depends on how good the MAC is, the better MAC the better performance can be obtained. Once a correct filter response has been determined and a coefficient table has been generated, the second step is to design the hardware architecture. The hardware designer must choose between area, performance, quantization, architecture, and response.

Design and implementation of digital FIR filter

MATLAB combines the high-level, mathematical language with an extensive set of pre-defined functions to assist in the creation and analysis of filter data. Toolbox are available for designing filter response and generating coefficient tables, each with varying levels of sophistication. Graphical filter design tools provide selections for specifying pass band, filter order, and design methods, as well as provide plots of the response of the filter to various standard forms of inputs[5]. FDA tool from The Math Works, which can generate a behavioral model and coefficient tables. Once a correct filter response has been determined and hardware architecture has been defined, the implementation can be carried out. Three choices of technology exist for the implementation of filter algorithms. These are: Programmable DSP chips, ASICs and FPGAs. At the heart of the filter algorithm is the multiply-accumulate operation; Programmable DSP chips typically have only one MAC unit that can perform one MAC in less than a clock cycle. DSP processors or programmable DSP chips are flexible, but they might not be fast enough. The reason is that the DSP processor is general purpose and has architecture that constantly requires instructions to be fetched, decoded and executed. ASICs can have multiple dedicated MACs that perform DSP functions in parallel. But, they have high cost for low volume production and the inability to make design modifications after production makes them less attractive. FPGAs have been praised for their ability to implement filters since the introduction of DSP savvy architectures . Which can be efficiently realized using dedicated DSP resources on these devices. More than 500 dedicated multiply-accumulate blocks are now available, making them exceptionally well suited for high-performance, high-order filtering applications that benefit from a parallel, non-resource shared hardware architecture. In this particular project, FPGA has been chosen as the implementation tool. To program FPGA, hardware description language is needed. VHDL synthesis offers an easy way to target a model towards different implementation.

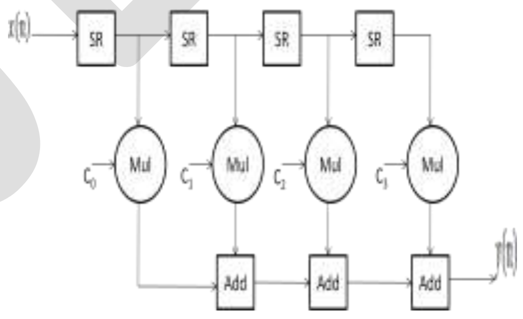
Architecture of FIR Filter by Using DA-OBC coding unit



FIR Filter Using shift register

One of the most fundamental elements for a DSP system is an FIR Filter Impulse Response – A set of FIR coefficients, which represent all possible frequencies.

Tap - A coefficient/delay pair. The number of FIR taps is an indication of the amount of memory required to implement the filter. DUE to the intensive use of FIR filters in video and communication systems[3], high performance in speed, area and power consumption is demanded. Basically, digital filters are used to modify the characteristic of signals in time and frequency domain and have been recognized as primary digital signal processing operations. They are typically implemented as multiply and accumulate (MAC) algorithms with the use of special DSP devices and how MAC is implemented with N multiplications and (N-1) additions per sample to compute the result



FIR Filter Using shift register

FIR Filter Using Distributed Arithmetic

Distributed Arithmetic (DA) is a different approach for implementing digital filters[2]. The basic idea is to replace all multiplications and additions by a table and a shifter accumulator. DA relies on the fact that the filter coefficients in are known, so multiplying c[n]x[n] becomes a multiplication with a constant Distributed Arithmetic (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.

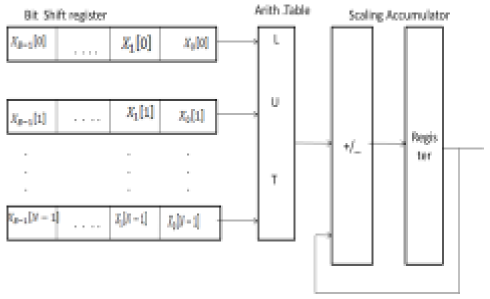
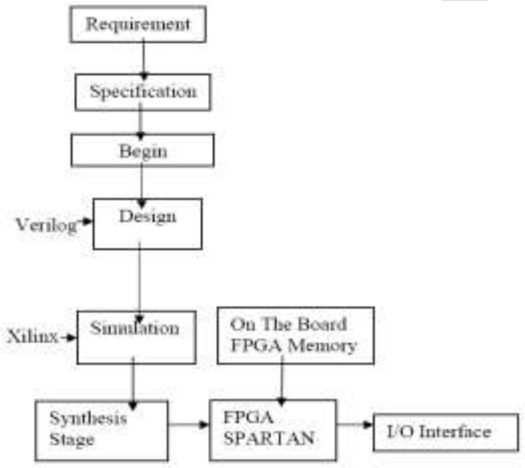


Fig 4.7 DA Block Diagram



Flow Chart Diagram

Multiplication in FPGA

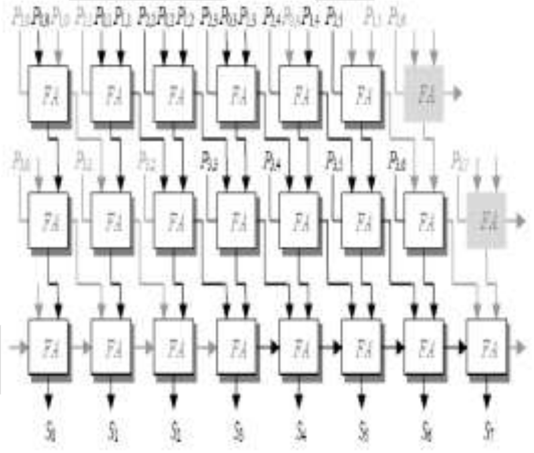
Multiplication is basically a shift add operation. There are, however, many variations on how to do it. Some are more suitable for FPGA use than others. Bit parallel multiplier are of two main types, array and tree multipliers. Because of speed and power consideration, the selection, here, is a tree multiplier structure. There are many tree structures one of them is Wallace tree. A Wallace tree is an implementation of an adder tree designed for minimum propagation delay. Rather than completely adding the partial products in pairs like the ripple adder tree does, the Wallace tree sums up all the bits of the same weights in a merged tree. Usually full adders are used, so that 3 equally weighted bits are combined to produce two bits: one (the carry) with weight of n+1 and the other (the sum) with weight n. Each layer of the tree, therefore, reduces the number of vectors by a factor of 3:2. The tree has as many layers as is necessary to reduce the number of vectors to two (a carry and a sum).The structure for this type of multiplier Wallace tree is a tree of carry-save adders. A carry save adder consists of full adders like the more familiar ripple adders, but the carry output from each bit is brought out to form second result vector rather than being wired to the next most significant bit. The carry vector is 'saved' to be combined with the sum later, A Wallace tree multiplier is one that uses a Wallace tree to combine the partial products from a field of 1x n multipliers (made of AND gates). If the Wallace tree combined with a fast adder can offer a significant advantage there. As Wallace tree is optimal in speed but it has a complicated routing, which in makes it impractical to implement since the cells in the tree has different loads and must be individually optimized, so a modification for fast parallel multiplier using both Wallace tree and Booth algorithms the overturned-Stairs adder is one of the modification of the Wallace tree which has the same speed of Wallace tree and is sufficient for most DSP and communication application.

BOOTH MULTIPLIER

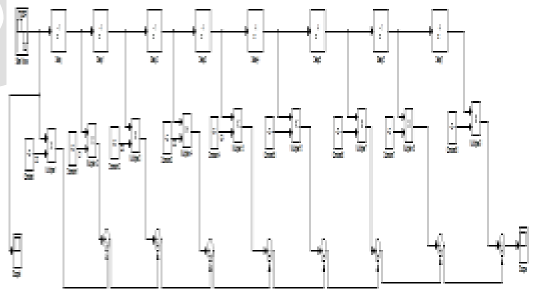
A power-efficient twin-precision modified-Booth multiplier is presented. For full-precision operation (16-bit inputs), the twin-precision modified-Booth multiplier shows an insignificant increase of critical-path delay (0.4% or 32 PHS) compared to a conventional multiplier. To cancel the power overhead of extra logic and to achieve overall power reduction, the implemented 16-bit twin-precision multiplier needs to run in dual 8-bit mode for more than 4.4% of its computations. Recent development of embedded systems indicates an increased interest in reconfigurable functional units that dynamically can make the data path adapt to varying computational needs. A system may need to switch between, for example, one application for speech encoding that requires functional units operating at 8-bit precision and another application that is based on 16-bit functional units to perform audio decoding. The twin-precision multiplier can switch between N bit and N/2-bit precision multiplications without significant performance or area overhead. Previous work introduced a twin-precision technique for radix-2 tree multipliers, but when higher performance is needed, multipliers with higher radix than two may be an option. In this paper we therefore explore the possibility of implementing the twin-precision concept on modified-Booth multipliers.

CARRY-SAVE-ADDER

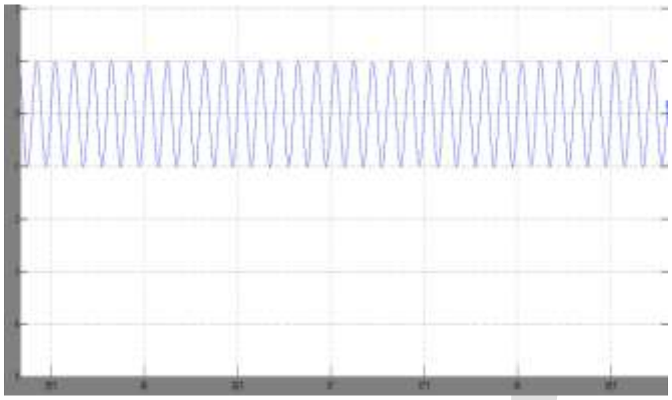
Carry-save-adder(CSA) is the most often used type of operation in implementing a fast computation of arithmetic's of register-transfer level design in industry. This paper establishes a relationship between the properties of arithmetic computations and several op- timing transformations using CSAs to derive consistently better qualities of results than those of manual implementations. In particular, we introduce two important concepts, operation-duplication and operation-split, which are the main driving techniques of our algorithm for achieving an extensive utilization of CSAs. Experimental results from a set of typical arithmetic computations found in industry designs indicate that automating CSA optimization with our algorithm produces designs with significantly faster timing and less



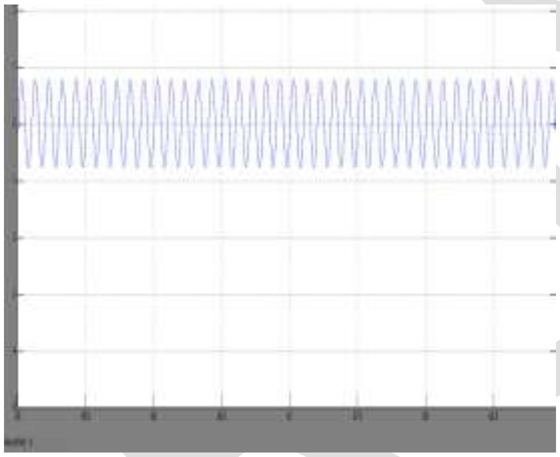
Circuit Design of 8-tap FIR Filter in Matlab



Mat lab Results:



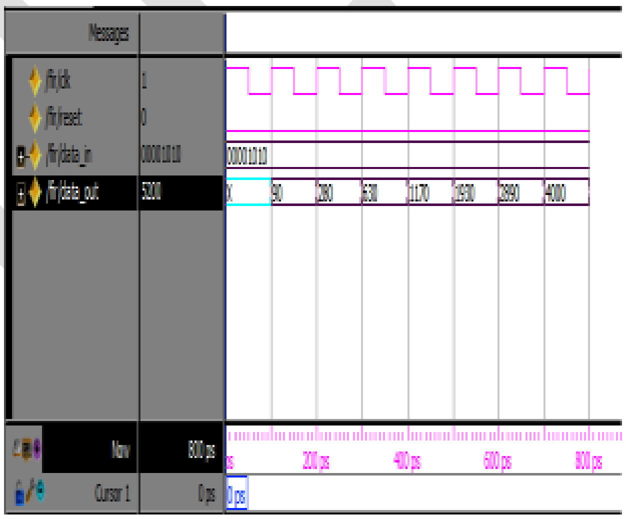
8-tap FIR Filter input signal



8-tap FIR Filter output signal

Here the Low Pass FIR Filter the present input signal and the output signal is to be same there is no loss of noise at the output signal without any loss of data. In this X-axis is for time period in micro seconds and Y-axis is amplitude the band width is 10 kHz.

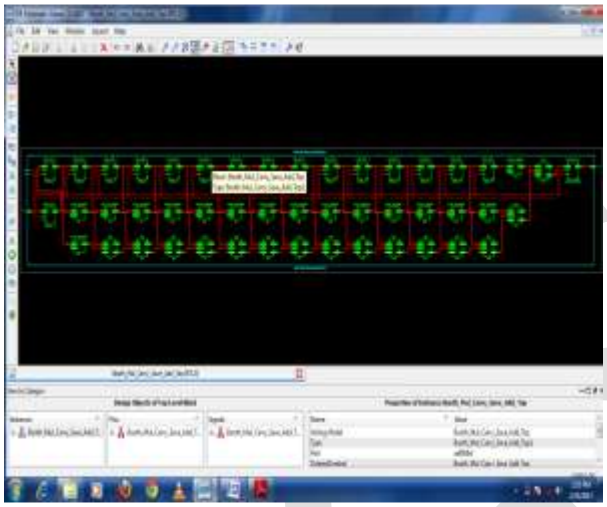
5.2. ModelSim results:

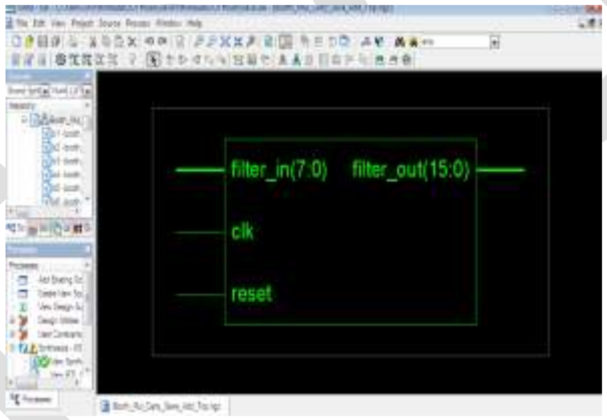


Simulation result of fir filter

The simulated result of FIR Filter by DA_OBC algorithm for 8 order LUT. In this we are taking the input as 10 it will multiply with each bit value and added simultaneously for all 8 ordered inputs, we get the outputs as 90,280,630

5.3. Xilinx simulated results





The RTL schematic report of Booth multiplier with Carry-Save-Adder in DA-OBC coding unit working for the FIR Filter. The above simulated results explain how the booth multiplier and carry-save-adder operates the FIR filter in DA-OBC coding unit in this Booth Multiplier multiplies the operands and automatically add it to the carry-save-adder .

ANALYSIS REPORT OF MULTIPLIER AND ADDERS BY DA-OBC

NORMAL MULTIPLIER

Power

Power summary: I(mA) P(mW)

Total estimated power consumption: 42

Timing and memory report

Total 7.165ns (6.364ns logic, 0.801ns route)

(88.8% logic, 11.2% route)

Total memory usage is 214620 kilobytes

Power leakage is 0.081

BOOTH MULTIPLIER

Power

```
Power summary:                I(mA)    P(mW)
-------------------------------------------------------------
Total estimated power consumption:          27
```

Timing and memory report

```
Total            7.165ns (6.364ns logic, 0.801ns route)
                         (88.8% logic, 11.2% route)
```

Total memory usage is 194064 kilobytes

Power leakage is 0.027

CONCLUSION

My project is mainly for the high speed of fir filter. For increasing the speed the DA-OBC algorithm is best method in that we are using the multipliers and adders for filtering operation my enhanced system is DA-OBC coding unit by using the Booth-Multiplier and Carry-Save-Adder replacing of normal multiplier and normal adder. The power and memory usage is very less for the Booth-Multiplier with Carry-Save-Adder compared with normal multiplier and normal Adder the power leakage is also less, based on all these factors speed of the FIR filter is increased.

REFERENCES:

[1] [1]. Xiumin Wang, “Implementation of FIR Filter on FPGA Using DAOBC Algorithm“ , in IEEE,2010.
[2] Shunwen Xiao,Yajun chen, "The design of FIR filter based on improved DA algorithm and its FPGA implementation ",IEEE International conference on computer and automation engineering(ICCAE’10), 2010,Vol.2,PP.589-591.
[3] New Approach to Look-up-Table Design and Memory-Based Realization of FIR Digital Filter Pramod Kumar Meher, Senior Member, IEEE.
[4] Design & implementation of FPGA based digital filters Ankit Jairath Department of Electronics & Communication, Gyan Ganga Institute of Technology and Sciences, Jabalpur (M.P) Issue 7, September 2012 199 All Rights Reserved © 2012 IJARCE