FIR Filter Design based on FPGA

1.1 Base Paper Abstract:

FIR (Finite Impulse Response) Filters: the finite impulse response filter is the most basic components in digital signal processing systems are widely used in communications, image processing, and pattern recognition. Based on FPGA(editable logic device) to achieve FIR filter, not only take into account the fixed -function DSP-specific chip real-time, but also has the DSP processor flexibility. The combination of FPGA and DSP technology can further improve integration, increase work speed and expand system capabilities.

1.2 Enhancement of this project:

- To Design the FIR Filter with the five blocks of FIR8, FIR8FirstI, FIR8FirstQ, FIR8Last and IntCompMux based on the FPGA concept using Truncation Multiplier.

1.3 Proposed Title:

- **FPGA concept of FIR Filter Design using Truncation Multiplier.**

1.4 Proposed Abstract:

With the widespread use of the high-speed digital signal processing(DSP) technology, FPGA has also increased speed and integration, which provides a new way for digital signal processing. FIR(Finite Impulse Response) Filter is the basic component in the digital signal processing system, whose impulse response is of finite duration, because it settles to zero in finite time. This paper proposes the new design concept of FIR Filter based on FPGA. This includes five blocks of operations are FIR8, FIR8FirstI, FIR8FirstQ, FIR8Last, IntCompMux. Thus, the FIR Filter Design based on FPGA has the greater advantage. Because, FPGA architecture of LUT's is be applicable to implement real-time, high speed and reliable FIR filter. Therefore, this paper focused on the Fir filter which is based on concept of FPGA. Finally, the proposed FIR filter based on FPGA is implemented in the VHDL and synthesized in the XILINX and compared in terms of area, power and delay reports.

1.5 Existing System:

With the widespread use of the high speed digital signal processing (DSP) technology, PLD has also increased speed and integration, which provides a new way for digital signal processing. In practice, the signal processing must be timely and accurate, otherwise it will lose its meaning. Therefore, the real time digital signal processing, rapidity and reliability become an important indicator of signal processing. DIR digital filter not only ensures the accurate strict linear phase characteristic, simple structure and stable in the design and application of digital filter, FPGA is a ones of usual PLD, its logical block arrangement is regular and the connection resource is rich, simultaneously its architecture of LUT is be applicable to implement real time, high speed and reliable FIR filter. Therefore FIR filter module based on FPGA chip design has great advantages.

One of the key technologies of EDA is to design digital hardware system with hardware description language(HDL). Currently verilog HDL is the most widely used hardware description languages. Its hardware description ability is very powerful from the logic gate level, circuit level to system level and other levels can be described and modeled, such as a counter, a storage system, a microprocessor, becoming the industry standard hardware description language. In addition it can carry on the simulation, the synthesis and the debugging are advantageous to the circuit function and the improvement of the structure, can shorten the design cycle greatly, reduces the expense. Most importantly, the design of the verilog HDL language has nothing to do with the specific hardware, thereby reducing the difficultly of designing the hardware circuit, allowing independent design and flexibility description, Therefore, under the support of EDA technology, reconfiguration the internal hardware structure of EDA technology, reconfiguration the internal hardware structure and working mode of the FPGA chip is more timesaving, cost saving, good flexibility, and good transplant ability. Using verilog HDL as a description method, a method of implementation an FIR filter with an FPGA is studied, which results in higher performance, lower scale, and lower cost.

1.5.1 FIR Filter Design

As the word indicates, a filter separates a desired signal from unwanted disturbances. For example, when we want to remove a disturbance such as noise from an audio signal, we design an appropriate filter that passes only the desires signal. But only in a few cases can we remove the disturbance completely and recover the desired signal; most of the time we have to settler for a compromise, most of the disturbance is rejected, most of the signal is recovered. The first candidate in filter is a linear filter. The main reason for this choice is that we have a good understanding of how a linear system operates. It

is only when a linear design fails or it yields unsatisfactory results that we look for other solutions, such as nonlinear or, adaptive techniques, for example.

Digital filters include infinite impulse response (UR) digital filter and finite impulse response (FIR) digital filter. As the FIR system have a lot of good features, such as only zeros, the system stability, operation speed quickly, linear phase characteristics and design flexibility, so that FIR has been widely used in the digital audio, image processing, data transmission, biomedical and other areas. FIR filter has a variety of ways to achieve, with the processing of modem electronic technology, taking use of field programmable gate array FPGA for digital signal processing technology has made rapid development, FPGA with high integration, high speed and reliability advantages, FIR filter implementation using FPGA is becoming a trend.

1.5.2 Basic Principle And Structure Of Fir Filter

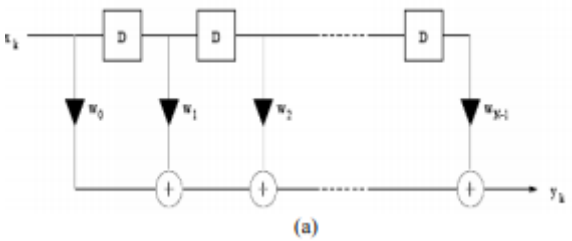
Set the unit impulse response h (n) of finite unit impulse response filter as the sequence of length N, the transfer function usually has the following form:

$$H(Z) = \sum_{n=0}^{N-1} h(n)Z^{-n} \text{.....(1)}$$

Differential equation can be described as

$$y(n) = \sum_{i=0}^{N-1} h(i)x(n-i) \text{.....(2)}$$

A FIR filter is a filter structure that can be used to implement almost any sort of frequency response digitally. It is usually implemented by using a series of delays, multipliers, and adders to create the filter's output.



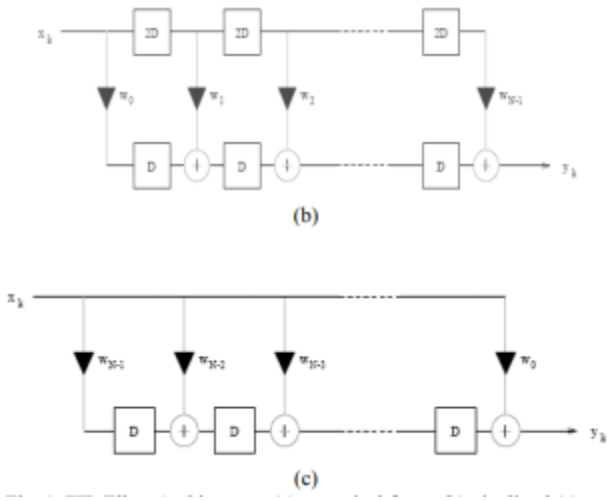


Figure 1: FIR Filter Architecture, (a) canonical form, (b) pipelined,(c) inverted form.

From this structure the transfer function of canonic form of the filter can be easily described in Z-domain as:

H(Z)=w0+w1z -1+w2z -2+------+wM-1Z L-1 .....(3)

In addition the advantages of FIR filter generally we use raised cosine pulse. The rectangular pulse occupies a large bandwidth so an alternative to rectangular pulse is a sinc pulse, which reduces the bandwidth and Inter Symbol Interference. The rectangular pulse is passed through the Root Cosine Filter a set of FIR filters to pulse shape the pulses to sinc . If very high sampling rates are required, full parallel hardware must be used. Such filters can be implemented on FPGAS using combinations of the general purpose logic fabric, on-board RAM and embedded arithmetic hardware. Full-parallel filters cannot share hardware over multiple clock cycles and so tend to occupy large amounts of resource. Hence, efficient implementation of such filters is important to minimize hardware requirement.

1.5.3 Fir Filter Design Techniques

FIR filters are particularly useful for applications where exact linear phase response is required. The FIR filter is generally implemented in a non-recursive way, which guarantees a stable filter. FIR filter design essentially consists of two parts:-

1.5.3.1 Approximation Problem

The approximation stage takes the specification and gives a transfer function through four steps. They are as follows:

- A desired or ideal response is chosen, usually in the frequency domain.
- An allowed class of filters is chosen (e.g. the length N for a FIR filters).
- A measure of the quality of approximation is chosen.
- A method or algorithm is selected to find the best filter transfer function.

1.5.3.2 Realization problem

The realization part deals with choosing the structure to implement the transfer function which may be in the form of circuit diagram or in the form of a program. There are essentially three well-known methods for FIR filter design namely:

- The window method
- The frequency sampling technique
- Optimal filter design methods

1.5.4 Fir Filter Add/Shift Implementation

In binary arithmetic, multiplication by a power-of-two is simply a shift operation. Implementation of systems with multiplications may be simplified by using only a limited number of power-of-two terms, so that only a small number of shift and add operations are required . An FIR filter tap can be implemented in two array columns of Xilinx series FPGAs. Because of the high degree of spatial and temporal locality, most signal routing delays are not critical, as they are with typical high performance FPGA designs. Each of the bit slices for the tap requires two combinational logic blocks (CLBs) in the array for implementation. The extensive local routing capability of typical FPGAs can be used for the majority of signals within and between taps. The implementation is based on a Xilinx board . This board has the following features, relevant to the presented implementation:

- Spartan3 FPGA with 500000 equivalent gates (XC3S5500E),
- 50MHz crystal oscillator,
- Asynchronous serial port, with RS232 drivers,
- Expansion connector with 100 I/O pins,
- Flash memory for bit stream storage,
- USB port for FPGA configuration and memory programming.

The scheme for the implementation of FIR filter to FPGAs. The main components of the implemented circuit are as follows.

1.5.4.1 Memory

Prepared for storage of past position data of bunches. z-1 means 1-turn delay .

1.5.4.2 Adder

Adder is made to reduce the number of stages and is a key for stable operation of the FPGA used in the board. This reduction of the stage is effective to avoid errors by clock skew in the FPGA and to reduce the power consumption and a circuit area(or number of gates) on FPGA.

1.5.4.3 Multiplier

Build-in multipliers are used to fulfill the requirements of high-speed operation; therefore this number of build-in multipliers is one of the constraints to the number of taps of FIR filter. D. Shift-Register It is used for additional delay for adjust latency to one or two revolution period.

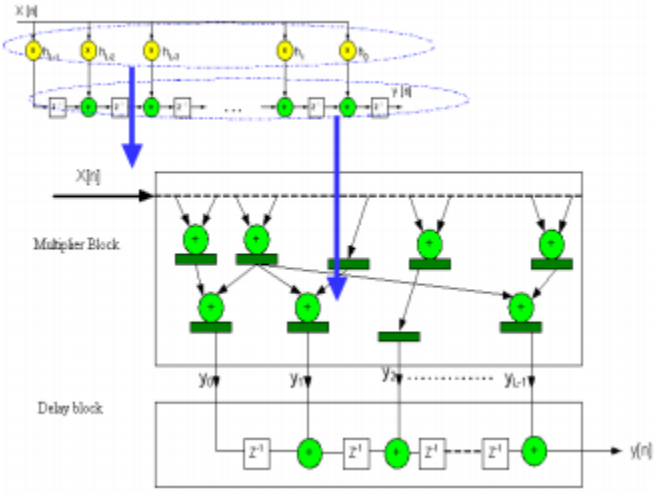


Figure 2: Implementation of FIR filter.

1.6 Disadvantages:

- Power consumption is high.
- Consumes large area size for FIR filter by using the concept of conventional Multiplier Design.

1.7 Proposed System:

With the widespread use of the high-speed digital signal processing(DSP) technology, FPGA has also increased speed and integration, which provides a new way for digital signal processing. FIR(Finite Impulse Response) Filter is the basic component in the digital signal processing system, whose impulse response is of finite duration, because it settles to zero in finite time. This paper proposes the new design concept of FIR Filter based on FPGA. This includes five blocks of operations are FIR8, FIR8FirstI, FIR8FirstQ, FIR8Last, IntCompMux. Thus, the FIR Filter Design based on FPGA has the greater advantage. Because, FPGA architecture of LUT's is be applicable to implement real-time, high speed and reliable FIR filter. Therefore, this paper focused on the Fir filter which is based on concept of FPGA. Finally, the proposed FIR filter based on FPGA is implemented in the VHDL and synthesized in the XILINX and compared in terms of area, power and delay reports.

1.7.1 FPGA Implementation

Advances in field programmable gate array technology have enabled FPGAs to be applied to a variety of problems. In particular, FPGAs prove particularly useful in data path designs, where the regular structure of the array can be utilized effectively. The programmability of FPGAs adds flexibility not available in custom approaches, while retaining relatively high system clock rates. The disadvantages of FPGAs are primarily related to the limited number of logic operations that can be implemented on a particular device, and the limited signal routing options that are available for connecting logical operators on the array.

The hardware description language VHDL is very popular among designers. One reason is good support of integrated circuit design by offering many integrated circuit related function and Data types. There is also large number of libraries available. It supports the behavioral modeling of hardware necessary when implementing a Fir filter generation program. Reducing flip-flop count through minimizing multiplier logic depth has instead been shown to yield the lowest area solutions. The results presented establish a clear low area. Total memory usage is 147920 kilobytes and Minimum period is 4.255ns (Maximum Frequency: 235.026MHz)

With the widespread use of the high-speed digital signal processing(DSP) technology, FPGA has also increased speed and integration, which provides a new way for digital signal processing. FIR(Finite Impulse Response) Filter is the basic component in the digital signal processing system, whose impulse

response is of finite duration, because it settles to zero in finite time. This paper proposes the new design concept of FIR Filter based on FPGA. This includes five blocks of operations are FIR8, FIR8FirstI, FIR8FirstQ, FIR8Last, IntCompMux. Thus, the FIR Filter Design based on FPGA has the greater advantage. Because, FPGA architecture of LUT's is be applicable to implement real-time, high speed and reliable FIR filter. Therefore, this paper focused on the Fir filter which is based on concept of FPGA. Finally, the proposed FIR filter based on FPGA is implemented in the VHDL and synthesized in the XILINX and compared in terms of area, power and delay reports.

1.7.2 Reduction of Parallel Tree Multiplier

A parallel tree multiplier design consists of three steps, i.e., PP generation, PP reduction, and final carry propagate addition. PP generation produces PP bits from the multiplicand and the multiplier. The goal of PP reduction is to compress the number of PPs to two, which is to be summed up by the final addition. The two most famous reduction methods are Wallace tree and Dadda tree reductions. Wallace tree reduction manages to compress the PPs as early as possible, whereas Dadda reduction only performs compression whenever necessary without increasing the number of carry-save addition (CSA) levels.

Here we achieve around 33% area optimization and 20% Fmax enhancement with the same performance. The two most famous reduction methods are Wallace tree and Dadda tree reductions. Wallace tree reduction manages to compress the PPs as early as possible, where as Dadda reduction only performs compression whenever necessary without increasing the number of carry-save addition (CSA) levels. Here we design a multiplier based on Wallace tree only.

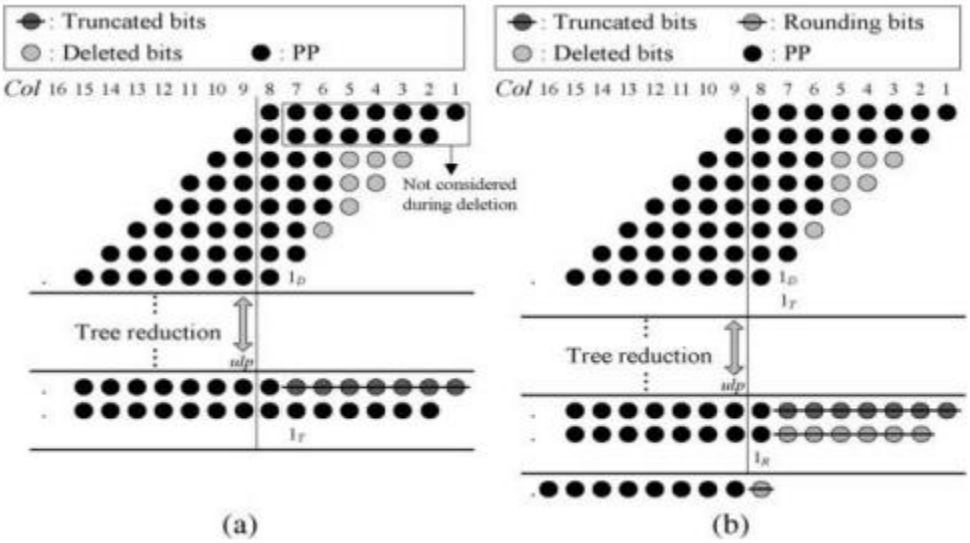


Figure 3 : proposed truncation multiplier

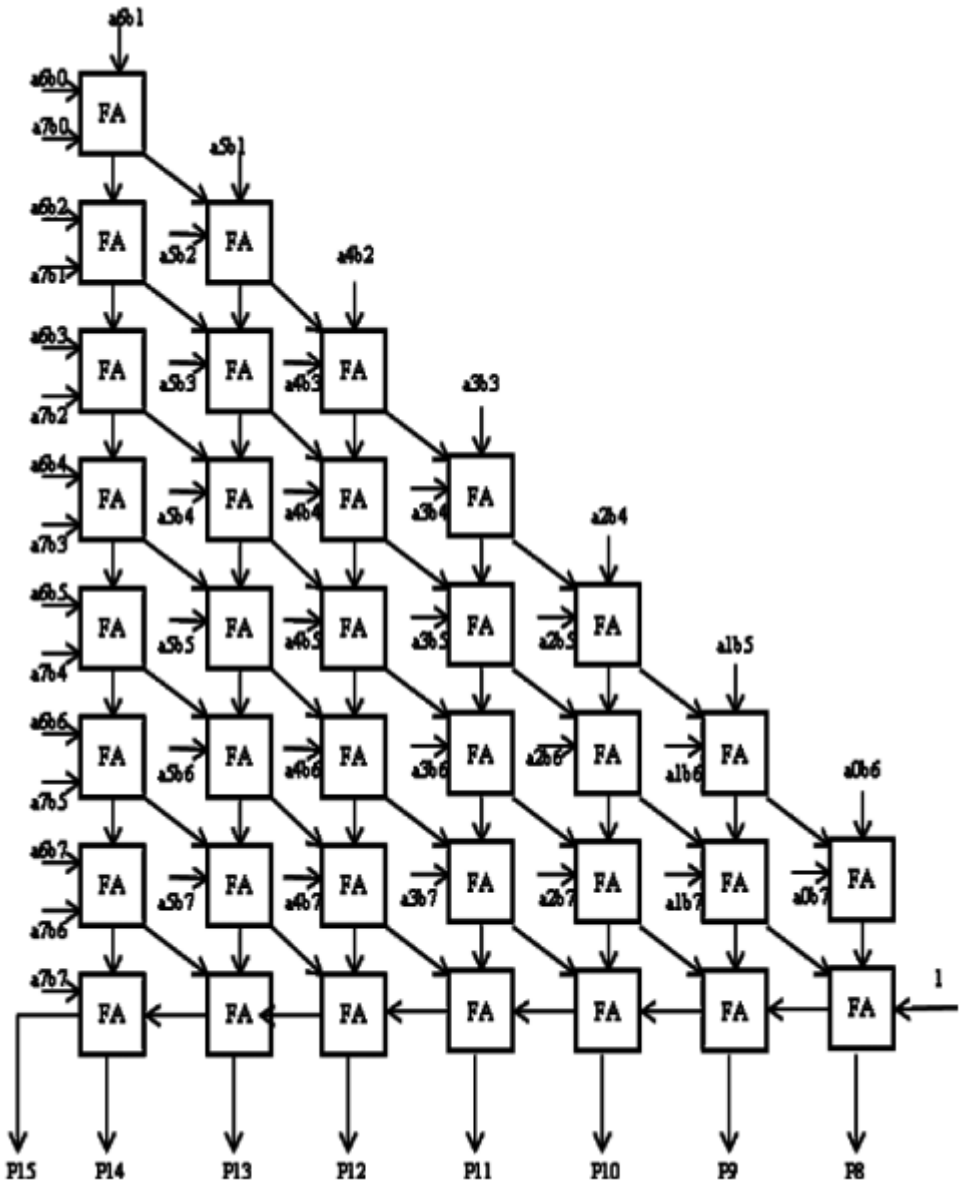


Figure 4 : proposed truncation multiplier

1.7.3 Proposed Truncated Multiplier Design

1.7.3.1 PP truncation and compression

The objective of the truncated multiplier design is to compute P MSBs of the product with a maximum truncation error of no more than 1 ulp, where 1 ulp = 2−P.The FIR filter design in this brief adopts the direct form is shown in figure. where the MCMA module sums up all the products aˆi × x[n − i]. Instead of accumulating individual multiplication for each product, it is more efficient to collect all the PPs into a single PPB matrix with carry-save addition to reduce the height

of the matrix to two, followed by a final carry propagation adder. In order to avoid the sign extension bits, we complement the sign bit of each PP row and add some bias constant using the property $s^{-} = 1 - s$, where s is the sign bit of a PP row, as shown in Figure. All the bias constants are collected into the last row in the PPB matrix. The complements of PPBs are denoted by white circles with over bars.

In the proposed truncated multiplier design in FIR filter implementation, it is required that the total error introduced during the arithmetic operations is no larger than one ulp. compares the two approaches. In the removal of unnecessary PPBs is composed of three processes: deletion, truncation, and rounding. Two rows of PPBs are set undeletable because they will be removed at the subsequent truncation and rounding .

1.7.4 Proposed FIR Filter:

The overall block diagram of proposed BsTxFir Filter is shown in fig.5.

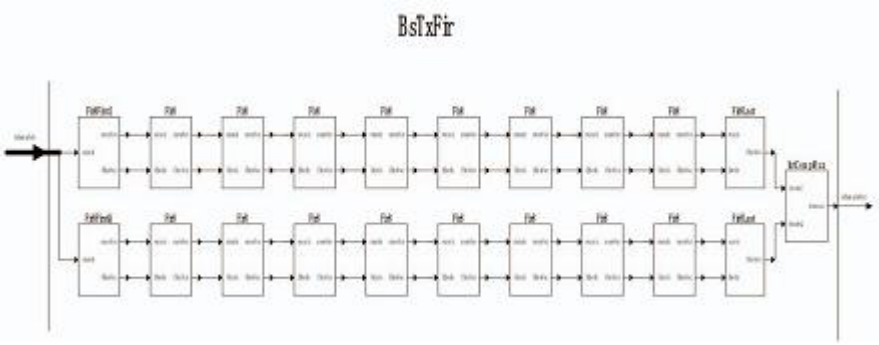


Figure 5: BsTxFir Filter Block Diagram

The proposed filter module consists of five blocks of small modules includes Fir8FirtsI, Fir8FirstQ, Fir8, Fir8Last and IntCompMux. Which Fir8 module is called 16 times Fir8Last modules is called 2 times. Fir8FirstI module and Fir8FirstQ module receives data form txsampleIn port, the difference between the two is the Fir8FirstI module process the data of the lower 16 bit and Fir8FirstI module process the data of the upper 16 bits. Fir8FirstI module and Fir8FirstQ module process the data through the 8 stage fir module respectively. Then, the output to the intcompmux module through the Fir8Last output respectively. The final output is the lower 16 bits of the output processed by the series of fir8firstI modules will be taken as the lower 16 bits of the final output, this form a 32 bit data is eventually exported from the txSampleOut.

Table 1: Module Interface and Parameters

Signal name	Signal width	Explanation
txSampleIn	32bit	Used to pass in data to be processed
txSampleOut	32bit	Output filter processed data

1.7.5 Module Algorithm Analysis

Fir8FirstI, Fir8FirstQ, Fir8, Fir8Last and IntCompMux these five modules of the algorithm code are basically the same. They all use the simplest direct network structure, with the difference being that the details of the parameters and the handling of the differences are different. The following call to the most modules Fir8 as an example to introduce the specific implementation of the algorithm, and then analyze the differences of the other four modules, you can understand the algorithm of theses five implemented is:

$$acc0 = \sum_{i=0}^{7}(x[n-i] * cffs[i]) \text{ .....()}$$

Cyclic shift x7→x8, x6→x7, x5→x6, x4→x5, x3→x4, x2→x3, x1→x2, x0→x1. This process gets data in the statein port, saved to [datahi: datalo] (the actual data low only deal with the data), then read acc acc0, mul2, tmp\ copy.0 datalo, x0\copy1 x8, datalo this sentence is to achieve datalo→x0, x8→datalo. completed a complete cycle shift. The statein port of each fir8 module accepts data from the output of the stateout port of the previous module, which is then passed to x0 of this module. The data of the stateout port of each fir8 module is output to the statein port of the next module as the x0 of the next module, continuing to participate multiply-accumulate cyclic shift operation.

1.7.5.1 Fir8 module flow chart

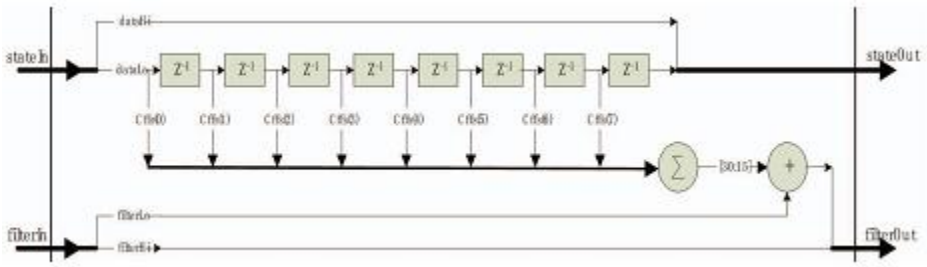


Figure 6: Fir8 Module Flow Chart

Input from state input 32bit data, divided into high 16 bit datahi and low 16 bit data as datalo is shifted to right by the unit delay, x0 becomes x1, and x1 multiplies by cffs(0), x1 is multiplied by cffs(1) and behind datalo is the initial value x0 is shifted respectively with cffs(i) by cumulative operation. Filter in is also a 32bit input data is also divided into 16 bit datahi and 16 bit datalo and temp signal that it passes by the multiply cumulative in the high 16 bit, and then and filterhi splicing, and finally get 32 bit filter out.

Table 2: Fir8 Module Interface and Parameters

Signal name	Signal width	Explanation
stateIn	32bit	Used to pass in data to be processed
filterIn	32bit	Used to pass in data to be processed
stateOut	32bit	Output module processed data
filterOut	32bit	Output module processed data

1.7.5.2 Fir8FirstI module Flow chart

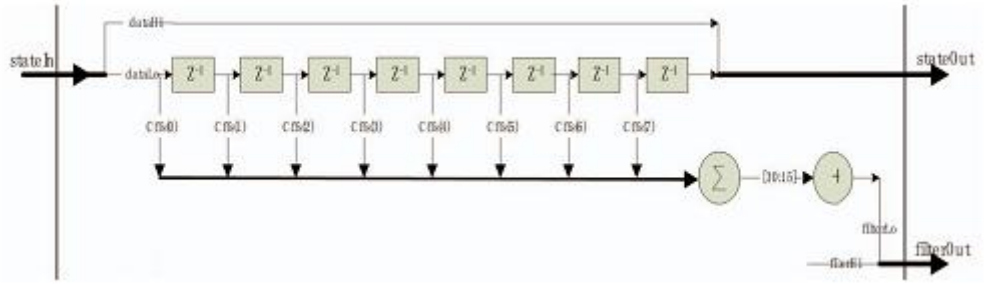


Figure 7: Fir8firstI Module Flow Chart

Similar to Fir8 module, but the filter out is different. The upper 16 bit of the multiply accumulator tmp signal on this module are directly used as the lower 16 bits of filterout. The below table shows the interface and parameters of the Fir8FirstI module.

Table 3: Fir8FirstI Module Interface and Parameters

Signal name	Signal width	Explanation
stateIn	32bit	Used to pass in data to be processed
stateOut	32bit	Output module processed data
filterOut	32bit	Output module processed data

1.7.5.3 Fir8FirstQ module Flow chart

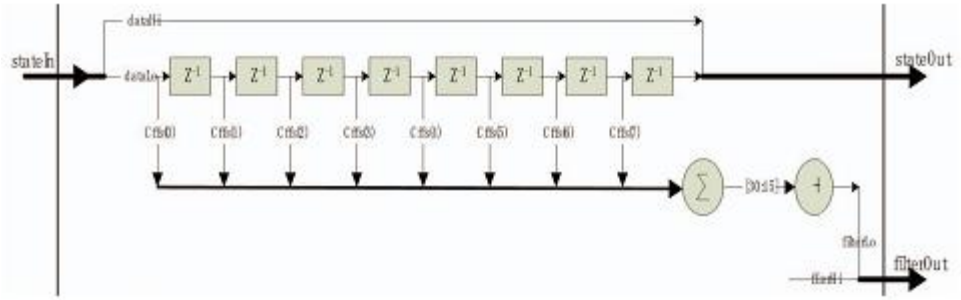


Figure 8: Fir8FirstQ Module Flow Chart

This module is basically the same as the Fir8FirstI module. The below table shows the interface and parameters of the Fir8FirstQ module.

Table 4: Fir8FirstQ Module Interface and Parameters

Signal name	Signal width	Explanation
stateIn	32bit	Used to pass in data to be processed
stateOut	32bit	Output module processed data
filterOut	32bit	Output module processed data

The algorithm of fir8last module is similar to that of the fir8 module, The difference is that the datahi data and the datalo data obtained after completing 8 shifts from the input 32 bit statein data are discarded, so there is no state out signal in the fir8 module, Fir8last module flow chart is as follows.

1.7.5.4 Fir8Last module Flow chart

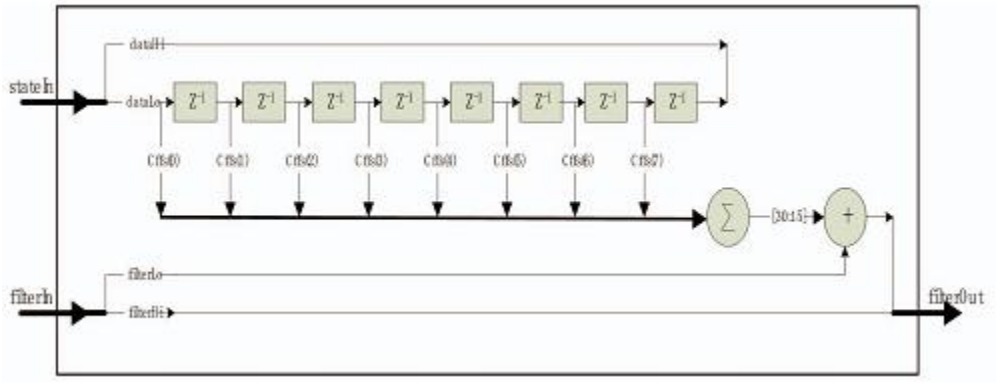


Figure 9: Fir8Last Module Flow Chart

The interface and parameters of the Fir8Last module is shown below.

Table 5: Fir8Last Module Interface and Parameters

Signal name	Signal width	Explanation
stateIn	32bit	Used to pass in data to be processed
filterIn	32bit	Used to pass in data to be processed
stateOut	32bit	Output module processed data

1.7.5.5 IntCompMux module Flow chart

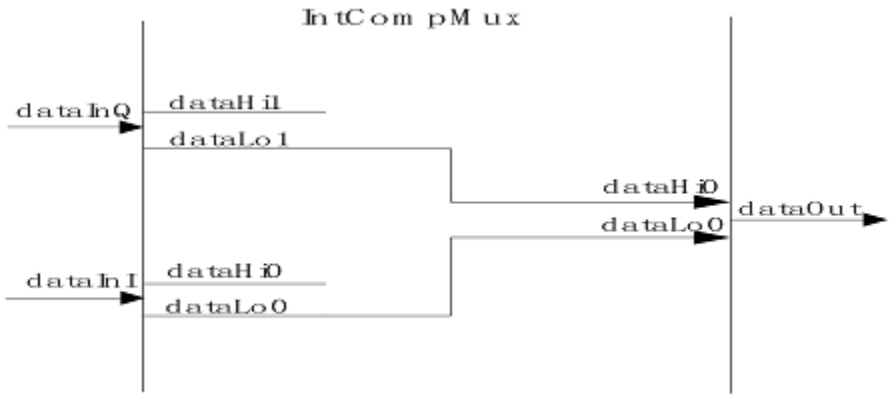


Figure 10: IntCompMux Module Flow Chart

IntCompMux module can realize signal splitting function and splice function. DatainI and the dataInQ are respectively connected by the filter out output port of the fir8last. First, datainQ divides the datahi of the upper 16 bit and the datalo1 of the lower 16 bit and then the datainI divides the dataHi0 of the upper 16 bits and the datalo0 of the lower 16 bits. Finally, put datalo1 to the upper 16 bits, datalo0 to the lower 16 bits, and concatenate them into the final 32 bit output signal.

The below table shows the interface and parameters of the IntCompMux module.

Table 6: IntCompMux Module Interface and Parameters

Signal name	Signal width	Explanation
dataInQ	32bit	Used to pass in the output signal
dataInI	32bit	Used to pass in the output signal
dataOut	32bit	Output module processed data

DataInI and dataInQ are respectively connected to by filterout output port of fir8last. As shown in below figures.

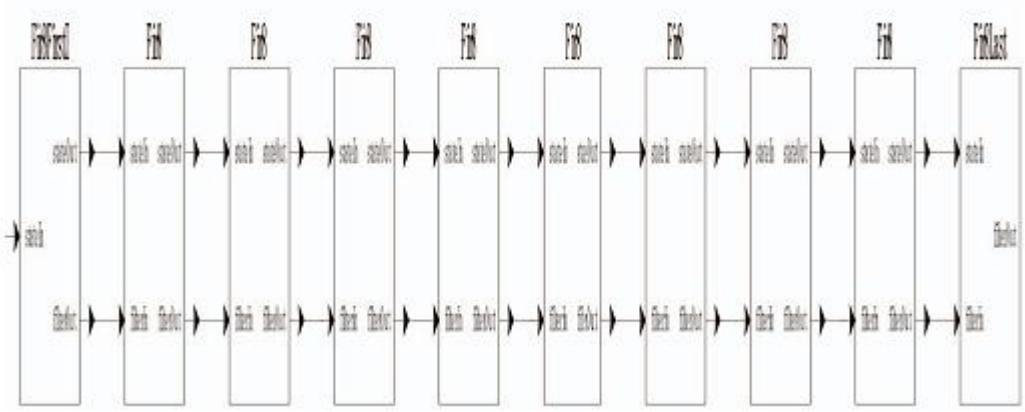


Figure 11: Input to Data In I interface

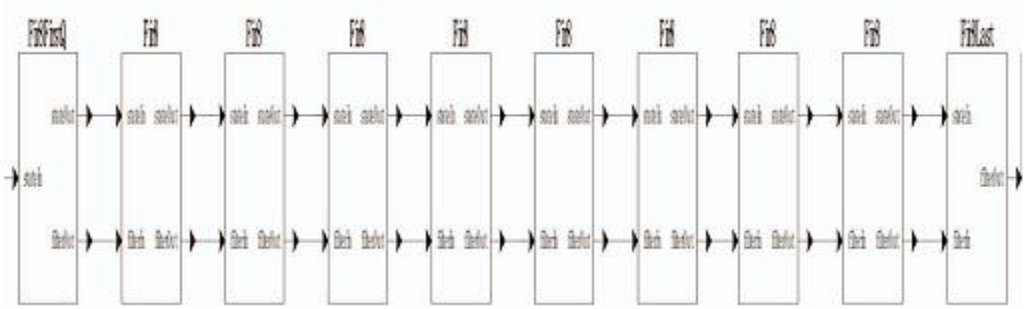


Figure 12: Input to Data In Q interface

The parameters of upper and the lower corresponding positions in these two strings of modules are the same.

1.8 Advantages:

- FPGA architecture of LUT's is be applicable to implement real-time, high speed and reliable FIR filter.
- Reduces the area and high efficient FIR filter by using the concept of Truncation Multiplier.

1.9 Literature Survey:

- S. Mirzaei, A. Hosangadi, and R. Kastner, FPGA Implementation of High Speed FIR Filters Using Add and Shift Method, 2006 -We present 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 II devices where we compare our implementations with those produced by Xilinx CoregenTM using Distributed Arithmetic. We observe up to 50% reduction in the number of slices and up to 75% reduction in the number of LUTs for fully parallel implementations. We also observed up to 50% reduction in the total dynamic power consumption of the filters. Our designs perform significantly faster than the MAC filters, which use embedded multipliers.

- K. N. Macpherson, "Low FPGA area multiplier blocks for full parallel FIR filters," in Proc. IEEE International Conference on Field-Programmable Technology, pp. 247 – 254, Issue Date: 6-8 Dec. 2004 -A new algorithm is presented that synthesizes multiplier blocks with the goal of minimizing FPGA hardware cost. Comparisons with existing algorithms are made via implementing synthesized blocks as the multiplication hardware of fully-pipelined. &U parallel, transposed form FIR fitters. Results establish that the classic optimization goal of minimizing adders does not minimize FPGA hardware. Instead, minimizing multiplier block logic depth is shown to be the primary factor for low area FPGA implementation. Filters generated using the new algorithm are also shown to consume less FPGA area than equivalents implemented using !he distributed arithmetic technique.

- M. Mehendale, S. D. Sherlekar, and G. Venkatesh, "Techniques for Low Power Realization of FIR _flters," ASP-DAC 1995.-In this paper we propose techniques for low power realization of FIR Alters on programmable DSPs. We Arst analyse the FIR implemendation to arrive at useful measures to reduce power and present techniques that exploit these measures. We then identify limitations of the existing DSP architectures in implementing these techniques and propose simple architectural extensions to overcome these limitations. Finally we present experimental results on real FIR filter examples that show upto 88% reduction in coefficient memory data bus power, upto 49% reduction in coefficient memory address bus power.

- H. Samueli, "An improved search algorithm for the design of multiplier less FIR filters with powers-of-two coefficients," IEEE Transactions on Circuits and Systems, pp. 1044-1047, July 1989,-An improved algorithm is presented for the discrete optimization of FIR digital filter coefficients which are represented by a canonic signed-digit (CSD) code, i.e., numbers representable as sums or differences of powers-of-two. The proposed search algorithm

allocates an extra nonzero digit in the CSD code to the larger coefficients to compensate for the very non uniform nature of the CSD coefficient distribution. This results in a small increase in the filter complexity however the improvement in the frequency response is substantial. The coefficient optimization is performed in two stages. The first stage searches for an optimum scale factor and the second stage consists of a local bivariate search in the neighborhood of the scaled and rounded coefficients.

- P. V. Rao, Cyril Raj Prasanna, and S. Ravi, “Design and ASIC Implementation of Root Raised Cosine Filter,” European Journal of Scientific Research, ISSN 1450-216X, vol. 31, no. 3, pp. 319-328, 2009, Xilinx Inc.,-Raised Cosine filter is a FIR filter that can be used to counter many problems of communication such as the Inter Symbol Interference. The rectangular pulse occupies a large bandwidth so an alternative to rectangular pulse is a sinc pulse, which reduces the bandwidth and Inter Symbol Interference. The rectangular pulse is passed through the Root Cosine Filter a set of FIR filters to pulse shape the pulses to sinc. The specification of the Root Raised Cosine FIR filter considered in this design is sampling frequency of 15.36 MHz and the cutoff frequency of 1.92 MHz, which is suitable to 3G-WCDMA. In the current project the Root-Raised-Cosine Filter is implemented using the quantized filter coefficients, which make it more, appropriate to be implemented using the integer format of representation. The filter coefficients were generated in Matlab and the response for the same is plotted and the Verilog model of the design is achieved using the MAC based structure, which uses the Booth Multiplier and adder that's not generic. The ASIC implementation of the Filter module is carried out and GDS-II of the filter module is obtained. The Filter Design was carried out in MATLAB and the coefficients were generated from MATLAB the coefficients were quantized and then used for the Implementation of the filter as Integer coefficients where the coefficients with 6 fixed points were multiplied by 8192. The Filter model was developed using Verilog HDL where the MAC based structure was developed and then the multiplier used was a booths multiplier the adder is also an adder which doesn't infer a design ware or a generic adder. The Filter developed was verified in Modelsim using the simulation and then the code is synthesized and physical design process of the netlist is done to get a design that works at 100MHz.

- Zhou YaFeng, Li Yuehua and Zhu Hao, "Design of 16 order FIR filter based on FPGA," Journal of Nanjing University of Technology, Vol. 27, Jan. 2005, pp. 46-50. -At first, this paper analyzes the

basic structure and hardware characteristics of the FIR digital filter, and then a design method of the FIR filter is discussed on the basis of the FIR filter structure. It is a method that is based on FPGA, draws the coefficient by Matlab and adopts the pipeline to implete the FIR digital filter. The article focuses on the introduction of the overall framework of the FIR digital filter adopting the finite-state machine as well as the principle of each module of the design. The design is impleted by use of the Verilog hardware description language and each module is verified and simulated by Quartus 8.0 and Modelsim-Altera.

1.10 References:

[1]S. Mirzaei, A. Hosangadi, and R. Kastner, FPGA Implementation of High Speed FIR Filters Using Add and Shift Method, 2006

[2] K. N. Macpherson, “Low FPGA area multiplier blocks for full parallel FIR filters,” in Proc. IEEE International Conference on Field-Programmable Technology, pp. 247 – 254, Issue Date: 6-8 Dec. 2004

[3] M. Mehendale, S. D. Sherlekar, and G. Venkatesh, “Techniques for Low Power Realization of FIR _flters,” ASP-DAC 1995.

[4] H. Samueli, “An improved search algorithm for the design of multiplier less FIR filters with powers-of-two coefficients,” IEEE Transactions on Circuits and Systems, pp. 1044-1047, July 1989,

[5] P. V. Rao, Cyril Raj Prasanna, and S. Ravi, “Design and ASIC Implementation of Root Raised Cosine Filter,” European Journal of Scientific Research, ISSN 1450-216X, vol. 31, no. 3, pp. 319-328, 2009, Xilinx Inc.,

[6]Jiang Xiaoyan, Digital Signal Processing and Applications, Nanjing: Southeast University Press, 2008

[7] Xue Nianxi, Application of MAT LAB in digital signal processing (2nd ed), Beijing: Tsinghua University Press, 2008: 397 - 400.

[8] Zhou YaFeng, Li Yuehua and Zhu Hao, "Design of 16 order FIR filter based on FPGA," Journal of Nanjing University of Technology, Vol. 27, Jan. 2005, pp. 46-50.

[9] UweMeyer - Baese, Digital signal processing with field programmable gate arrays, 2nd ed. Beijing: Tsinghua University Press, 2006 102 - 105.

[10] Zhang Haijun, "Design and implementation of 16 order FIR filter based on FPGA," Journal of Anhui University Natural Science Edition, Vol. 33, Jan. 2009, pp. 62-65.