OPTIMIZATION OF LOW POWER USING FIR FILTER

S. Prem Kumar

Lecturer/ ECE Department
Narasu’s Sarathy Institute of Technology
Salem, Tamil Nadu, India

S. Sivaprakasam

Lecturer/ ECE Department
Narasu’s Sarathy Institute of Technology
Salem, Tamil Nadu, India

G. Damodharan

Lecturer/ ECE Department
Mahendra Engineering College
Mallasamudram
Namakkal, Tamilnadu, India

V. Ellappan

Lecturer/ ECE Department
Mahendra Engineering College
Mallasamudram
Namakkal, Tamilnadu, India

***Abstract:* In this paper we proposed a three stage pipelined finite-impulse response (FIR) filter, this FIR filter contains multipliers such as Hybrid multiplier, Booth multiplier algorithm and Array multiplier. In general, multiplication process consists of two parts as multiplicand and multiplier. According to the array multiplier, the numbers of partial products (PP) are equal to the number of bits in multiplier. Booth multiplier is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. Booth's algorithm can be reduced by half using booth recoding. But in the hybrid multiplication technique, the partial products can still be reduced which in turn reduces the switching activity and power consumption. Multiplication is a very important operation in many digital signal processing (DSP) applications. In our proposed system, the performance of our hybrid multiplier is compared with an array multiplier and booth multiplier. The comparison is based on synthesis results obtained by synthesizing the multiplier architectures targeting a Xilinx FPGA.**

Keywords: FIR, hybrid Multipler and Booth Multiplier.

I. Introduction

Multiplication is a very important operation in many digital signal processing (DSP) applications. In our proposed system, the performance of our hybrid multiplier is compared with an array multiplier and booth multiplier. The comparison is based on synthesis results obtained by synthesizing the multiplier architectures targeting a Xilinx FPGA. The comparison is done by the help of VHDL synthesize. The number of logic gates occupied by each multiplier which determines the efficiency of the multiplier. According to the Spartan-3 FPGA kit, the result shows the number of gates occupied by 8x8 array multiplier is number of slices is 79 out of 1920, that is nearly 4% of the original slices available in the kit. FIR filters are one of two primary types of digital filters used in Digital Signal Processing (DSP) applications. Digital filters are a very important part of DSP. Because their extraordinary performance is one of the key reasons in DSP applications. Filters have two uses as signal separation and signal restoration. Signal separation is needed when a signal has been contaminated with interference, noise, or other signals. Signal restoration is used when a signal has been distorted in some way. A finite impulse response (FIR) filter is a type of a digital filter. The impulse response, the filter's response to a

Kronecker delta input, is finite because it settles to zero in a finite number of sample intervals. A finite impulse response (FIR) filter design the main thing is to calculate the power with different multipliers by using FIR filter this is in contrast to infinite impulse response (IIR) filters, which have internal feedback and may continue to respond indefinitely. The impulse response of an Nth-order FIR filter lasts for N+ 1 sample, and then dies to zero. In our finite impulse filter

II. Problem Domain

Multiplication is a very important operation in many digital signal processing (DSP) applications. In our proposed system, the performance of our hybrid multiplier is compared with an array multiplier and booth multiplier. The comparison is based on synthesis results obtained by synthesizing the multiplier architectures targeting a Xilinx FPGA. The comparison is done by the help of VHDL synthesize. The number of logic gates occupied by each multiplier which determines the efficiency of the multiplier. According to the Spartan-3 FPGA kit, the result shows the number of gates occupied by 8x8 array multiplier is number of slices is 79 out of 1920, that is nearly 4% of the original slices available in the kit

A. Methodology

This work presents the performance of our hybrid multiplier is compared with an array multiplier and booth multiplier. The comparison is based on synthesis results obtained by synthesizing the multiplier architectures targeting a Xilinx FPGA. This work presents the comparison is done by the help of VHDL synthesize. The number of logic gates occupied by each multiplier which determines the efficiency of the multiplier. According to the Spartan-3 FPGA kit, the result shows the number of gates occupied by 8x8 array multiplier is number of slices is 79 out of 1920, that is nearly 4% of the original slices available in the kit.

III. Filter and Methodology

A. Filters

Digital filters are very important part of DSP. Infact their extraordinary performance is one of the key reasons that DSP has become so popular. Filters have two uses: signal separation and signal restoration. Signal separation is needed when the signal has been contaminated with interference, noise or other signals. For example imagine a device for measuring the electrical activity of a baby’s heart (EKG) while in the womb. The raw signal will be likely to be corrupted by the breathing and the heartbeat of the mother. A filter must be used to separate these signals so that they can be individually analyzed. Signal restoration is used when the signal has been distorted in some way. For example, an audio recording made with poor requirement may be filtered to better represent the sound as it actually occurred. Another example is of debluring of an image acquired with an improper focused lens, or a shaky camera. These problems can be attacked with either digital or analog filters. Which is better? Analog filters are cheap, fast and have a large dynamic range both in amplitude and frequency. Digital filters in comparison are vastly superior in the level of performance that can be achieved. Digital filters can achieve thousands of times better performance than an analog filter.

This makes a dramatic difference in how filtering problems are approached. With analog filters, the emphasis is on handling limitations of the electronics such as the accuracy and st ability of the resistors and capacitors. In comparison digital filters are so good that the performance of the filter is frequently ignored. The emphasis shifts to the limitations of the signals and the theoretical issues regarding their processing. It is common in DSP to say that a filter input and output signals are in time domain. This is because signals are usually created by sampling at regular intervals of time. But this is not the only way sampling can take place. The second most common way of sampling is at equal intervals in space. For example imagine taking simultaneous readings from an array of strain sensors mounted at one centimeter increments along the length of an aircraft wing. Many other domains are possible; however, time and space are by far the most common. When you see the term time domain in DSP, remember that it may actually refer to samples taken over time, or it may be a general reference to any domain that the samples are taken in. Every linear filter has an impulse response, a step response and a frequency response. Each of these responses contains complete information about the filter, but in a different form. If one of three is specified, the other two are fixed and can be directly calculated. All three of these representations are important, because they describe how the filter will react under different circumstances. The most straightforward way to implement a digital filter is by convolving the input signal with the digital filter’s impulse response. All possible linear filters can be made in this manner. When the impulse response is used in this way, filters designers give it aspecial name: the filter kernel. There is also another way to make digital filters, called recursion. When a filter is implemented by a convolution, each sample in the output is calculated by weighting the samples in the input, and adding then together. Recursive filters are an extension of this, using previously calculated values from the output, besides points from the input. Instead of using a filter kernel, recursive filters are defined by a set of recursion coefficients. For now the important point is that all linear filters have an impulse response, even if you don't use it to implement the filter. To find the impulse response of a recursive filter, simply feed in the impulse and see what comes out. The impulse responses of recursive filters are composed of sinusoids that exponentially decay in amplitude.

In principle, this makes their impulse responses infinitely long. However the amplitude eventually drops below the round off noise of the system, and the remaining samples can be ignored. Because of these characteristics, recursive filters are also called Infinite impulse response or IIR filters. In comparison, filters carried out by convolution are called Finite impulse response or FIR filters. The impulse response is the output of a system when the input is an impulse. In this same manner, the step response is the output when the input is a step. Since the step is the integral multiple of the impulse response.

B. FIR Filters

Digital filters can be divided into two categories: finite impulse response (FIR) filters; and infinite impulse response (IIR) filters. Although FIR filters, in general, require higher taps than IIR filters to obtain similar frequency characteristics, FIR filters are widely used because they have linear phase characteristics, guarantee stability and are easy to implement with multipliers, adders and delay elements the number of taps in digital filters varies according to applications. In commercial filter chips with the fixed number of taps, zero coefficients are loaded to registers for unused taps and unnecessary calculations have to be performed. To alleviate this problem, the FIR filter chips providing variable-length taps have been widely used in many application fields. However, these FIR filter chips use memory, an address generation unit, and a modulo unit to access memory in a circular manner. The paper proposes two special features called a data reuse structure and a recurrent-coefficient scheme to provide variable-length taps efficiently. Since the proposed architecture only requires several MUXs, registers, and a feedback-loop, the number of gates can be reduced over 20 % than existing chips. In, general, FIR filtering is described by a simple convolution operation

An N-Tap transversal was assumed as the basis for this adaptive filter. The value of N is determined by practical considerations, An FIR filter was chosen because of its stability. The use of the transversal structure allows relatively straight forward construction of the filter.

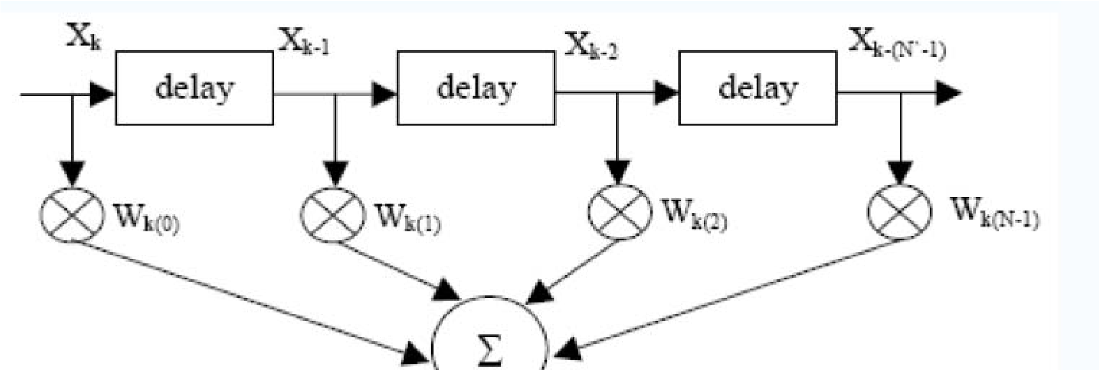


Figure 1 N-Tap Transversal filter

C. Adders

An adder is a digital circuit that performs addition of numbers. In modern computers adders reside in the arithmetic logic unit (ALU) where other operations are performed. Although adders can be constructed for many numerical representations, such as Binary-coded decimal or excess-3, the most common adders operate on binary numbers. In cases where two's complement is being used to represent negative numbers it is trivial to modify an adder into an adder-subtractor.

D. Binary Multiplier

A Binary multiplier is an electronic hardware device used in digital electronics or a computer or other electronic device to perform rapid multiplication of two numbers in binary representation. It is built using binary adders. The rules for binary multiplication can be stated as follows

1. If the multiplier digit is a 1, the multiplicand is simply copied down and represents the product.

2. If the multiplier digit is a 0 the product is also 0.

For designing a multiplier circuit we should have circuitry to provide or do the following three things:

1. It should be capable identifying whether a bit 0 or 1

2. It should be capable of shifting left partial products.

3. It should be able to add all the partial products to give the products as sum of Partial products

4. It should examine the sign bits. If they are alike, the sign of the product will be a Positive, if the sign bits are opposite product will be negative. The sign bit of the product stored with above criteria should be displayed along with the product. From the above discussion we observe that it is not necessary to wait until all the partial products have been formed before summing them. In fact the addition of partial product can be carried out as soon as the partial product is formed.

IV Architecture of radix 2^n Multiplier

The architecture of a radix 2^n multiplier is given in the Figure 2. This block diagram shows the multiplication of two numbers with four digits each. These numbers are denoted as V and U while the digit size was chosen as four bits. The reason for this will become apparent in the following sections. Each circle in the figure corresponds to a radix cell which is the heart of the design. Every radix cell has four digit inputs and two digit outputs. The input digits are also fed through the corresponding cells. The dots in the figure represent latches for pipelining. Every dot consists of four latches. The ellipses represent adders which are included to calculate the higher order bits. They do not fit the regularity of the design as they are used to “terminate” the design at the boundary. The outputs are again in terms of four bit digits and are shown by W’s. The 1’s denote the clock period at which the data appear.

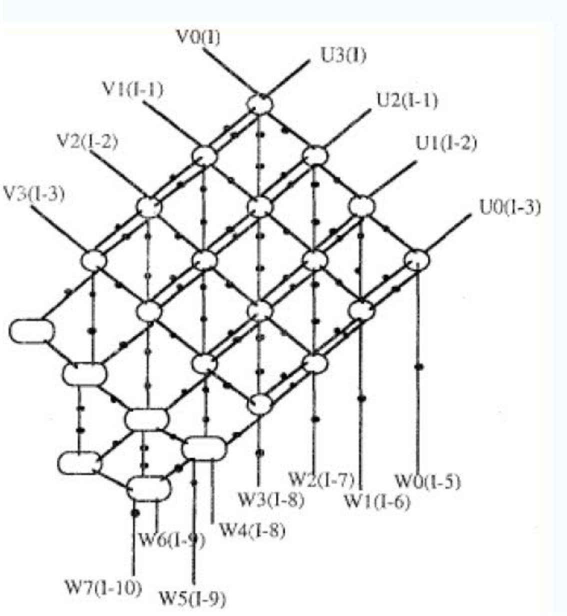


Figure 2 Architecture of radix 2^n Multiplier

V Booth Multiplier

The decision to use a Radix-4 modified Booth algorithm rather than Radix-2 Booth algorithm is that in Radix-4, the number of partial products is reduced to *n/2.* Though Wallace Tree structure multipliers could be used but in this format, the multiplier array becomes very large and requires large numbers of logic gates and interconnecting wires which makes the chip design large and slows down the operating speed.

VI Proposed Array Multiplier Architecture

Figure 3 shows the proposed low power multiplier architecture using SMT. This algorithm is applied to a 1-D FIR filter module in Fig. .The architecture contains multiplexers for choosing between the current multiplication results and previously calculated and stored results in a cache. If the higher bits of input coincide with the tags in a cache, the multiplexers select the stored value. If not, the multiplexers choose the new result of the multiplier instead and it is stored back to one of the entries of a cache. The registers located in back of the multipliers are prepared in order to prevent transferring the multiplication result directly to the next adder and to give time to load a cache data on bus. The adder is necessary in order to add the results of the higher and the lower bits multiplication. The final result is the same as the output of the no separated multiplication processors. Its functionality is verified by Verilog–HDL simulation. A cache typically lies between the processor and the main memory. However, in this architecture, the only link of a cache is to the internal interface-multipliers.

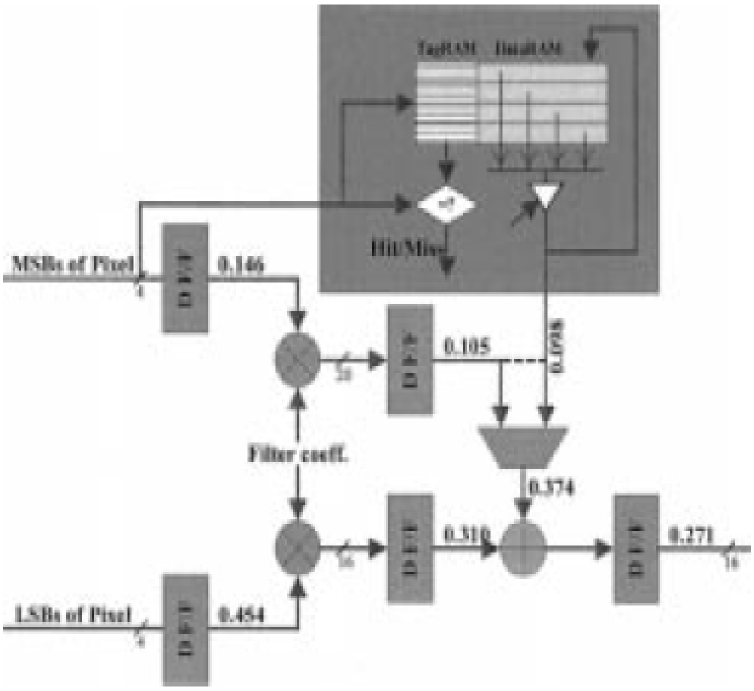


Figure 3 Array Multiplier Architecture

VI Power Performance

Excessive power is expensive in many ways. It creates the need for special design and operational considerations – requiring everything from heat sinks to fans to sophisticated heat exchangers. Even the cost of building larger power supplies must be taken into consideration. Overall, increased power requires more of everything, including: more area on the PCB, a larger chassis, more floor space, and larger air conditioning systems. The costs continue to compound. Perhaps the most critical issue is the effect excessive power can have on reliability. As the junction temperatures rise, transistors consume more power, thereby further increasing the temperature of the device. Continuously operating systems with junction temperatures running from 85°C to over 100°C increases reliability issues. Fortunately, Xilinx encountered the first evidence of this 90 nm inflection point in the early development stages of Spartan™-3 FPGAs, the first Xilinx FPGA family for the 90 nm processes. Xilinx began immediately developing new ways to cope with the inherent power issues posed by the 90 nm process. Consequently, when the higher performance Virtex-4 family was introduced in September 2004, Xilinx was confident that the new family would simultaneously deliver the best of both worlds – the highest performance and lowest power consumption in a 90 nm FPGA.

VII Reducing power in FPGA

There are two major components to power consumption in FPGAs: static power and dynamic power. Inrush current is another factor that can occur when the FPGA is powered on. Each component poses a unique challenge. For the 90 nm FPGA, the most challenging component is static power

VIII Results and conclusion

Before any hardware implementation and testing is performed, all the design modules are tested for correct functionality using the FPGA Advantage MODELSIM functional simulation tool. The MODELSIM tool provided the necessary environment for complete functional simulation of the target hardware. The MODELSIM tool features high speed and target hardware platform adaptability. It also allows for modification and verification of the design simultaneously. For this purpose a test bench facility is available in the EDA tool which is the most suitable method to run a complete simulation for the design. It was described with the VHDL code. The test bench provides access to text file which contains the data of the tested input signal. The input data is generated by MATLAB program. The designed test bench has been run and Our completed design for the wavelet transform achieved a throughput of 19.2 K Samples per second, which means that it is capable of taking in a new sample of the input signal approximately every 52.1 μs, and producing a reconstructed signal at the output at the same rate. First, the type of signal is defined and analyzed with the Daubechies wavelet (Fatma H 2008). The simulation is done extensively with discrete signal then wave file that quantized and converted to a binary code by the MATLAB program. The Bit Error Rate between the input and the reconstructed audio signal gives quantitative evidence for the performance of the wavelet implementations. The BER is simply defined to be BER = Errors/Total Number of Bits. For the Daubechies wavelet transforms, the BER is zero, which prove that the implementations execute the operation of the wavelet transform correctly and verifying the perfect reconstruction conditions.

A. Results of different multipliers

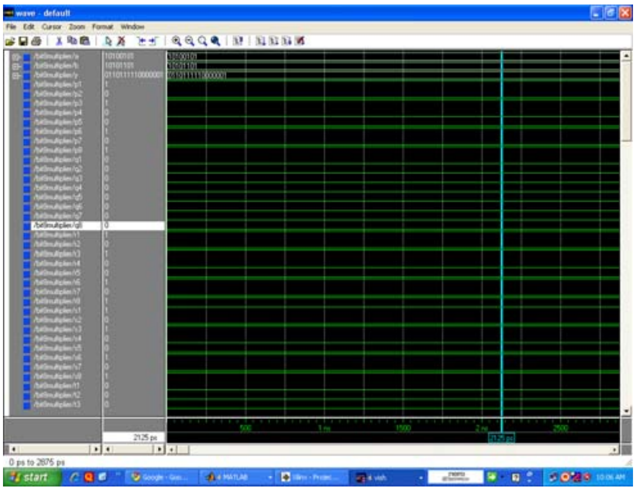


Figure 4 simulations results of normal array multiplier

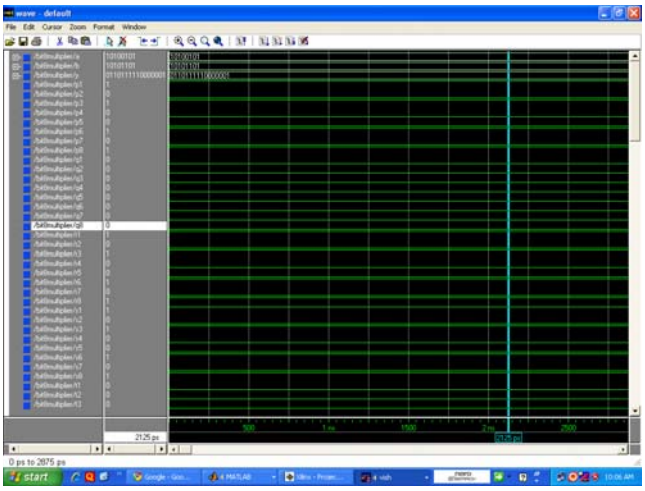


Figure 5 simulation results of normal booth multiplier

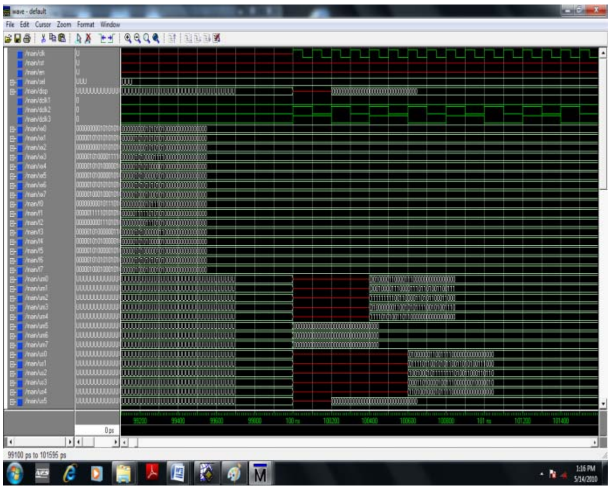
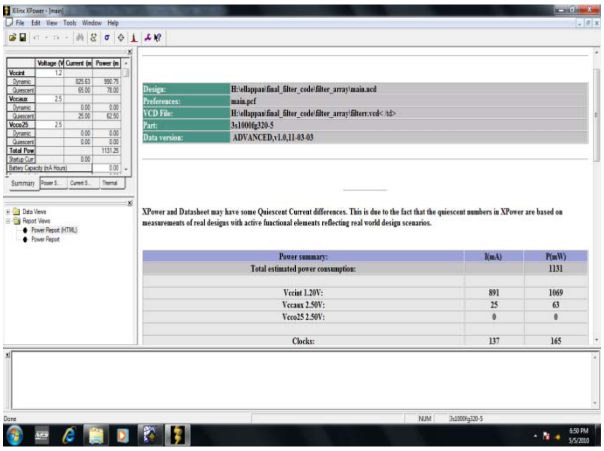


Figure 6 simulations result of array multiplier with filter

B. Power consumption Details of Filters



Xilinx XPower - [main]

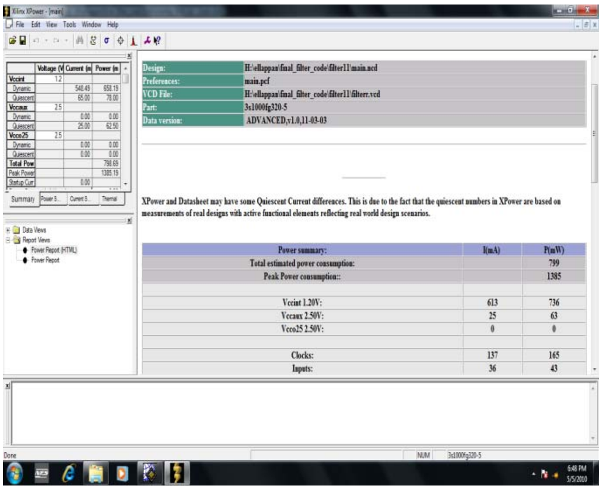
	Voltage (V	Current (m	Power (m
Vccint	1.2		
Dynamic		825.63	990.75
Quiescent		65.00	78.00
Vccaux	2.5		
Dynamic		0.00	0.00
Quiescent		25.00	62.50
Vcco25	2.5		
Dynamic		0.00	0.00
Quiescent		0.00	0.00
Total Pow			1131.25
Startup Cur		0.00	
Battery Capacity (mA Hours)			0.00

Design:	H:\ellappan\final_filter_code\filter_array\main.ncd
Preferences:	main.pcf
VCD File:	H:\ellappan\final_filter_code\filter_array\filterr.vcd< /td>
Part:	3s1000fg320-5
Data version:	ADVANCED,v1.0,11-03-03

XPower and Datasheet may have some Quiescent Current differences. This is due to the fact that the quiescent numbers in XPower are based on measurements of real designs with active functional elements reflecting real world design scenarios.

Power summary:	I(mA)	P(mW)
Total estimated power consumption:		1131
Vccint 1.20V:	891	1069
Vccaux 2.50V:	25	63
Vcco25 2.50V:	0	0
Clocks:	137	165

Figure7 shows power details of array multiplier with filter



Xilinx XPower - [main]

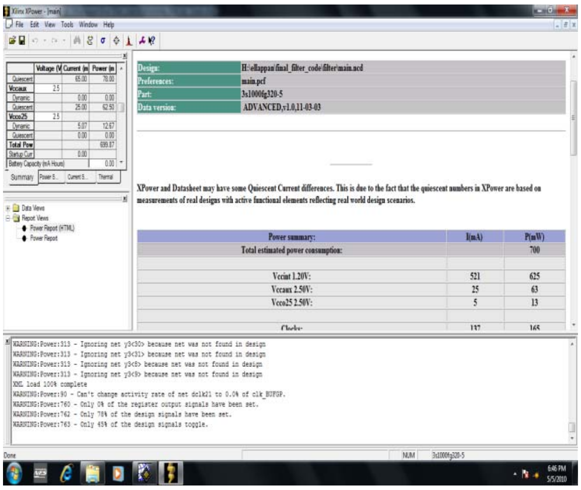
	Voltage (V	Current (m	Power (m
Vccint	1.2		
Dynamic		548.49	658.19
Quiescent		65.00	78.00
Vccaux	2.5		
Dynamic		0.00	0.00
Quiescent		25.00	62.50
Vcco25	2.5		
Dynamic		0.00	0.00
Quiescent		0.00	0.00
Total Pow			798.69
Peak Power			1385.19
Startup Cur		0.00	

Design:	H:\ellappan\final_filter_code\filter11\main.ncd
Preferences:	main.pcf
VCD File:	H:\ellappan\final_filter_code\filter11\filterr.vcd
Part:	3s1000fg320-5
Data version:	ADVANCED,v1.0,11-03-03

XPower and Datasheet may have some Quiescent Current differences. This is due to the fact that the quiescent numbers in XPower are based on measurements of real designs with active functional elements reflecting real world design scenarios.

Power summary:	I(mA)	P(mW)
Total estimated power consumption:		799
Peak Power consumption::		1385
Vccint 1.20V:	613	736
Vccaux 2.50V:	25	63
Vcco25 2.50V:	0	0
Clocks:	137	165
Inputs:	36	43

Figure 8 shows power details of booth multiplier with filter



	Voltage (V	Current (m	Power (m
Quiescent		65.00	78.00
Vccaux	2.5		
Dynamic		0.00	0.00
Quiescent		25.00	62.50
Vcco25	2.5		
Dynamic		5.07	12.67
Quiescent		0.00	0.00
Total Pow			699.87
Startup Cur		0.00	
Battery Capacity (mA Hours)			0.00

Design:	H:\ellappan\final_filter_code\filter\main.ncd
Preferences:	main.pcf
Part:	3s1000fg320-5
Data version:	ADVANCED,v1.0,11-03-03

XPower and Datasheet may have some Quiescent Current differences. This is due to the fact that the quiescent numbers in XPower are based on measurements of real designs with active functional elements reflecting real world design scenarios.

Power summary:	I(mA)	P(mW)
Total estimated power consumption:		700
Vccint 1.20V:	521	625
Vccaux 2.50V:	25	63
Vcco25 2.50V:	5	13
Clocks:	137	165

```
WARNING:Power:313 - Ignoring net y3<30> because net was not found in design
WARNING:Power:313 - Ignoring net y3<31> because net was not found in design
WARNING:Power:313 - Ignoring net y3<8> because net was not found in design
WARNING:Power:313 - Ignoring net y3<9> because net was not found in design
XML load 100% complete
WARNING:Power:90 - Can't change activity rate of net dclk21 to 0.0% of clk_BUFGP.
WARNING:Power:760 - Only 0% of the register output signals have been set.
WARNING:Power:762 - Only 78% of the design signals have been set.
WARNING:Power:763 - Only 45% of the design signals toggle.
```

Figure 9 shows power details of modified booth multiplier with filter

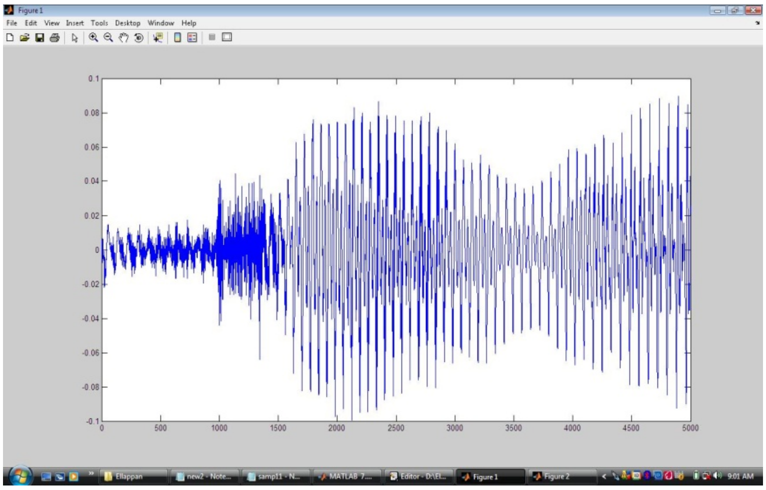


Figure 9 shows input of filters

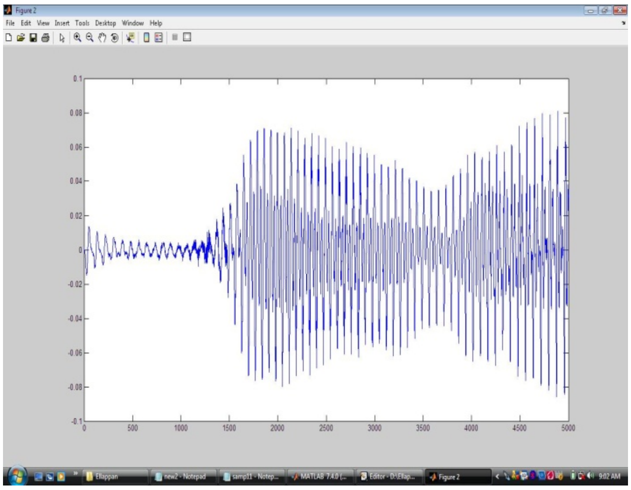


Figure 10 shows output of filters

Table 1 Array Multiplier

Number of Slices	229
Numb er of 4 input LUTs	302
Number of bonded INPUT	16
Number of bonded OUTPUT	16
CLB Logic Power	1131mW

Table 2Booth Multiplier

Number of Slices	130
Number of 4 input LUTs	249
Number of bonded INPUT	16
Number of bonded OUTPUT	17
CLB Logic Power	799mW

Table 3 Modified Booth Multiplier

Number of Slices	229
Number of 4 input LUTs	302
Number of bonded INPUT	16
Number of bonded OUTPUT	16
CLB Logic Power	699mW

IX Conclusion

This paper gives a clear concept of different multiplier and their implementation in tap delay FIR filter. We found that the parallel multipliers are much option than the serial multiplier. We concluded this from the result of power consumption and the total area. In case of parallel multipliers, the total area is much less than that of serial multipliers. Hence the power consumption is also less. This is clearly depicted in our results. This speeds up the calculation and makes the system faster. While comparing the radix and the radix modified booth multipliers we found that radix consumes lesser power than that of radix. This is because it uses almost half number of iteration and adders when compared to radix. When all the three multipliers were compared we found that array multipliers are most power consuming and have the maximum area. This is because it uses a large number of adders. As a result it slows down the system because now the system has to do a lot of calculation. Multipliers are one the most important component of many systems. So we always need to find a better solution in case of multipliers. Our multipliers consume less power and cover less power. We try to determine which of the three algorithms works the best. In the end we determine that modified booths Algorithm work the best.

XII References

[1] L. Benini et al, Glitch power minimization by selective gate freezing, IEEE Trans. VLSI Systems, 8(3): 287-298, 2000

[2] Y. L. Hsu and S. J. Wang, Retiming-based logic synthesis for low power, Proc.Int. Symp. Low Power Electronics and Design, ACM Press, 2002, pp. 275-278.

[3] M. Pandemic et al, Influence of compiler optimizations on system power, IEEE Trans. VLSI, 9(6):801- 804, 2001G. Sutter et al, Logic depth, power, and pipeline granularity: updated results on XC4K and Virtex FPGAs, Computacion Reconfigurable & FPGAs, Publications Digitales S.A., 2003, pp. 201-207.

[4] W. Tsu, et al, HSRA: High-speed, hierarchical synchronous reconfigurable array,

[5] ACM Seventh International Symposium on Field-Programmable Gate Arrays, Feb. 1999

[6] J. Park, H. Choo, K. Muhammad, K. Roy, “ Non adaptive and Adaptive filter implementation based on sharing multiplication,” ICASSP, June 2000.

[7] Jan M. Rabaey, “Digital Integrated Circuits : A Design Perspective,” Prentice Hall, New Jersey, 1996.

[8] Samueli, “An improved Search Algorithm for the Design of Multiplier less FIR filter with Powers-of-Two Coefficients,” IEEE Trans. On circuits and systems, Vol.36, No. 7, pp. 1044-1047, Jul. 1989.