
GAZİ ÜNİVERSİTESİ
Gazi
1926

# FPGA BASED DIGITAL FILTER DESIGN

**A THESIS SUBMITTED TO**
**THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**
**OF**
**GAZİ UNIVERSITY**

**BY**
**Cihan AYDIN**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS**
**FOR**
**THE DEGREE OF MASTER OF SCIENCE**
**IN**
**SMART GRIDS**

**JULY 2019**

The thesis study titled "FPGA BASED DIGITAL FILTER DESIGN" is submitted by Cihan AYDIN in partial fulfillment of the requirements for the degree of Master of Science in the Department of Smart Grids, Gazi University by the following committee.

**Supervisor**: Prof. Dr. İbrahim SEFA

Electrical and Electronics Engineering Department, Gazi University

I certify that this thesis is a graduate thesis in terms of quality and content

.....

**Chairman**: Prof. Dr. Şevki DEMİRBAŞ

Electrical and Electronics Engineering Department, Gazi University

I certify that this thesis is a graduate thesis in terms of quality and content

.....

**Member**: Dr. Lect. Selami BALCI

Electrical and Electronics Engineering Department, Karamanoğlu Mehmet Bey University

I certify that this thesis is a graduate thesis in terms of quality and content

.....

Date: 25/07/2019

I certify that this thesis, accepted by the committee, meets the requirements for being a Master of Science Thesis.

.....

Prof. Dr. Sena YAŞYERLİ

Dean of Graduate School of Natural and Applied Sciences

## ETHICAL STATEMENT

I hereby declare that in this thesis study I prepared in accordance with thesis writing rules of Gazi University Graduate School of Natural and Applied Sciences;

- All data, information and documents presented in this thesis have been obtained within the scope of academic rules and ethical conduct,
- All information, documents, assessments and results have been presented in accordance with scientific ethical conduct and moral rules,
- All material used in this thesis that are not original to this work have been fully cited and referenced,
- No change has been made in the data used,
- The work presented in this thesis is original,

or else, I admit all loss of rights to be incurred against me.

Cihan AYDIN

25/07/2019

FPGA BASED DIGITAL FILTER DESIGN

(M. Sc. Thesis)

Cihan AYDIN

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

July 2019

## ABSTRACT

The goal of the thesis was to design and make use of a FIR filter with low pass by using the latest architecture of Xilinx 7-series FPGA. The newest architectures and designs such as Microblaze soft core processor and IP blocks in this thesis were used for the design. Also, System Generator from Simulink extension was utilized to simulate the FIR filter using Xilinx blocksets. Hence, the parallel implementation, various taps of filters and simulations may be used for these purposes. In this context, detailed explanation of resources usage, performance comparisons between Xilinx 7-series FPGAs and latency were discussed. In addition, detailed design processes were shown. It was accomplished that the filter that is used in the design carry out the demands, timing considerations and resource usages.

Science Code : 90520
Key Words : FPGA, Xilinx, FIR filter, multichannel, IP blocks, Microblaze
Page Number : 64
Supervisor : Prof. Dr. İbrahim SEFA

FPGA TABANLI SAYISAL FİLTRE TASARIMI

(Yüksek Lisans Tezi)

Cihan AYDIN

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Temmuz 2019

ÖZET

Son mimari yapıya sahip Xilinx 7-serisi FPGA kullanarak, alçak geçiren FIR filtreden yararlanarak yapılan tasarım bu tezin amacını oluşturmaktadır. Microblaze soft-core işlemci ve IP bloklar gibi en yeni tasarımlar ve mimari yapılar tezde yapılan tasarımlarda kullanılmıştır. Ayrıca, Xilinx blok setleri kullanarak System Generator uygulamasında FIR filtre simülasyonları yapılmıştır. Dolayısıyla, paralel uygulama, çeşitli filtre dereceleri ve simülasyonlar bu amaçlar için kullanılabilmektedir. Bu kapsamda, kaynakların kullanımı, Xilinx 7-serisi FPGAler arasındaki performans karşılaştırmaları ve gecikme ile ilgili konuların detaylı açıklamaları bu tezde belirtilmektedir. Ek olarak, detaylı tasarım işlemleri gösterilmektedir. Bu tezde kullanılan filtre ile ilgili zamanlama konuları ve kaynak kullanımları, talepleri karşıladığı görülmüştür.

Bilim Kodu : 90520
Anahtar Kelimeler : FPGA, Xilinx, FIR filtre, çoklu kanal, IP blokları, Microblaze
Sayfa Adedi : 64
Danışman : Prof. Dr. İbrahim SEFA

## ACKNOWLEDGEMENT

I want to mention my deepest gratitude and regard to my advisor Prof. Dr. İbrahim SEFA, Deparment of Electrical and Electronics Engineering for his valuable guidance, patience, and encouragement throughout of this thesis. I also sincerely thank to my advisor for providing the neccessary tools that aid me to finish my thesis.

**CONTENTS**

**Page**

**Page**

# LIST OF TABLES

**Table** **Page**

# LIST OF FIGURES

**Figure** **Page**

**Figure** **Page**

**Figure** **Page**

# LIST OF ABBREVIATIONS AND UNITS

Symbols and abbrevations in which are used in this study are submitted below with their explanations.

| Abbreviations | Explanations |
|---|---|
| **ADC** | Analog-to-digital converter |
| **BRAM** | Block random access memory |
| **CAN** | Controller are network |
| **CE** | Clock enable |
| **CLB** | Configurable logic block |
| **CPU** | Central processing unit |
| **DAC** | Digital-to-analog converter |
| **DMA** | Direct memory access |
| **DRAM** | Distributed ramdom access memory |
| **DRP** | Dynamic reconfiguration port |
| **DSP** | Digital signal processing |
| **FIFO** | First-in-first-out |
| **FIR** | Finite impulse response |
| **FPGA** | Field programmable gate array |
| **GPIO** | General purpose input output |
| **HDL** | Hardware description language |
| **HDMI** | High definition multimedia interface |
| **HLS** | High level synthesis |
| **IIR** | Infinite impulse response |
| **ILA** | Integrated logic analyzer |
| **IP** | Intellectual property |
| **LC** | Logic cell |
| **LTI** | Linear time invariant |
| **LUT** | Look-up-table |
| **MAC** | Multiply accumulate |
| **PWM** | Pulse width modulation |
| **RAM** | Random access memory |

| Abbreviations | Explanations |
|---|---|
| **RISC** | Reduced instruction set computer |
| **RTL** | Register transfer level |
| **SoC** | System-on-chip |
| **SQNR** | Signal-to-quantization noise level |
| **SPI** | Serial peripheral interface |
| **UART** | Universal asynchronous receiver-transmitter |
| **USB** | Universal serial bus |
| **WE** | Write enable |
| **XADC** | Xilinx analog-to-digital converter |

# 1. INTRODUCTION

Digital filtering is one of the important aspect in digital signal processing world. Digital filters are essentially used to filter unwanted portions of the signal for various applications such as power electronics and control systems. Application of digital filters utilizes adders, multipliers and shift register blocks. Architecture of the digital filters manipulates these blocks and determines the speed, complexity and power [1].

The signals with noise that are considered in control cycle without filtering cause some problems with respect to unstability or misdetection. Yet another problem is that the systems that are processed with microprocessors cause some timing problems on account of long time of filtering the signals. That is why with minimum latency, there is a possibility that the performance of the control signals can be enhanced by filtering multiple analog channels.

Analog filters can be cost friendly, quicker and vast dynamic range compared to digital filters. However, as far as the performance is concerned, digital filters are superior to the analog filters.

Digital signal processing is composed of few steps such as digitizing, mathematical calculations and converting digital signals to analog signals. In this operation, *an analog-to-digital converter* (ADC) receives analog signals with noise and digitizes the noisy signal. After digitizing, *Digital signal processor / Field programmable gate array* (DSP/FPGA) receives the samples and executes the suitable mathematical operations according to the type of the filter and finally, a *digital-to-analog converter* (DAC) receives the filtered signal to convert the signal from digital to analog form.

Nowadays, the latest trend for DSP is gradually changing into FPGAs. FPGAs are being gradually used in digital signal processing world. New generation FPGAs comprise of wide range of CLBs (configurable logic blocks). This structure provides speed, flexibility, cost and performance compared to conventional DSP (Digital signal processing) processors [2].

In this thesis, the newest architectures and designs such as IP blocks and Microblaze are executed. In particular, efficient implementation of *finite impulse response* (FIR) filters using Xilinx Artix-7 series (XC7A100T-1CSG324C) is implemented. This implementation uses optimum resources regarding to Xilinx FPGA. Parallel implementation, various taps of filters and simulations are used for this purpose. Detailed explanation of resource and latency is discussed in this thesis. And besides, detailed design processes are shown in this thesis.

# 2. DIGITAL SIGNAL PROCESSING

Digital signal processing is related with fulfilling operations upon digital signals such as filtering, convolution, correlation and Fourier transforms. Those operations demand summation and multiplication and thus, a *multiply-accumulate* (MAC) unit is included in the DSP processors. DSP applications were utilized by making use of DSP processors. The summation and multiplication are carried out *sequentially* in the DSP processors. As applying filtering process in digital filters, a DSP processor may only fulfill single summation-multiplication process in single-cycle. Thus, output of every tap of the filter is carried out in single cycle, yet in a higher order filter more clock cycles are needed to operate the filter specifications.

FPGAs are famous with their *concurrency or parallelism*. The summation and multiplication processes are fulfilled concurrently in FPGAs. Unlike a DSP processor, an FPGA operates multiple summation and multiplication processes in one cycle. Hence, the FPGAs requires lower clock cycle compared to the DSP processors.

Algorithmic complexity increases as application demand increases. In order to implement these new algorithms, higher performance signal processing hardwares are required. Classic fixed architecture DSP processors cannot keep pace on their own. Because of that, performance gap increases as algorithmic complexity increases. Figure 2.1 shows the difference between the conventional DSP processor implementation and an FPGA implementation [3].

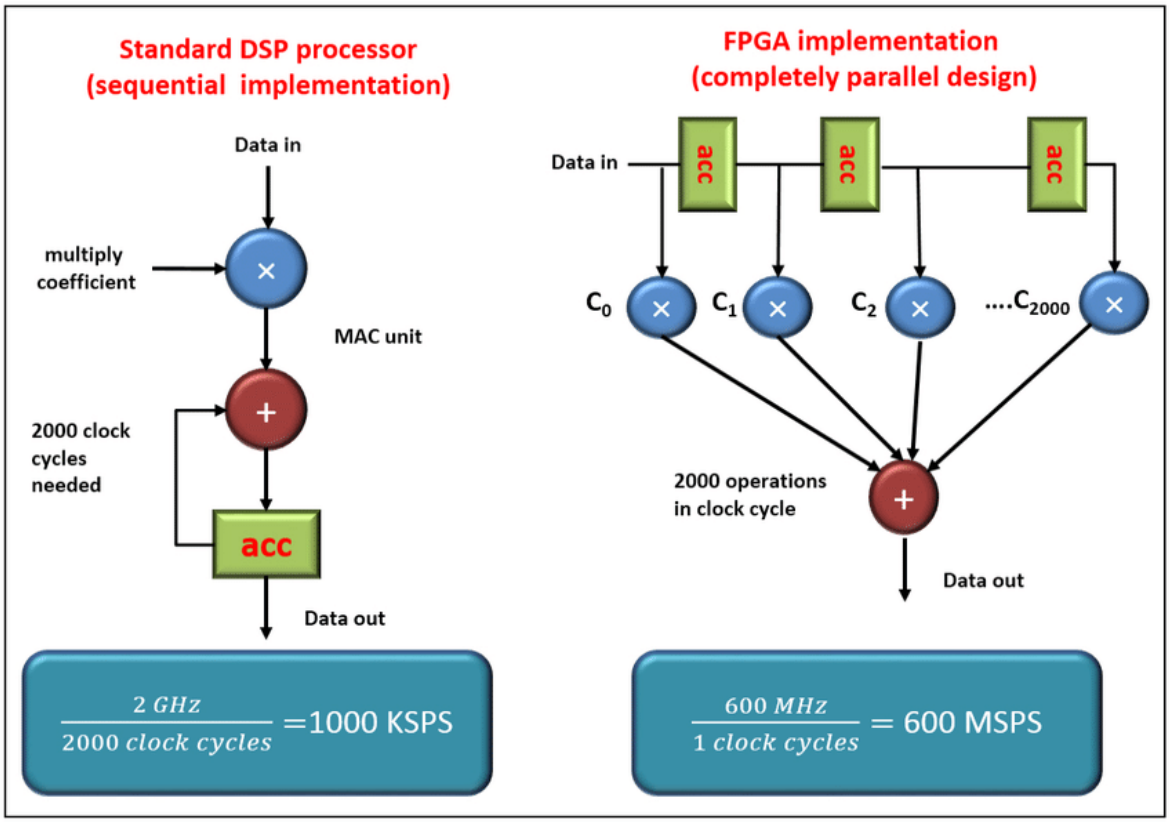


Figure 2.1. Conventional DSP and an FPGA

FPGAs are very well favorable to replenish this performance gap for the various reasons;

- They suggest extremely high-performance signal processing capability via concurrency.
- They ensure very low risk owing to the flexible architecture.
- They permit design migration to handle changing standards.
- Developers can utilize them to form a specialized and differentiated solution.
- Their price is relatively low.
- They provide very low power per function [3].

A DSP or an FPGA manipulates mathematical algorithms according to a digital filter used in a system. A digital filter receives digital inputs and gives digital outputs. Typically, in a filtering, a digital signal processor or an FPGA reads sample from an analog to digital converter, manipulates mathematical processes according to the filter type and extracts the result to a digital to analog converter. Figure 2.2 supports above statements.

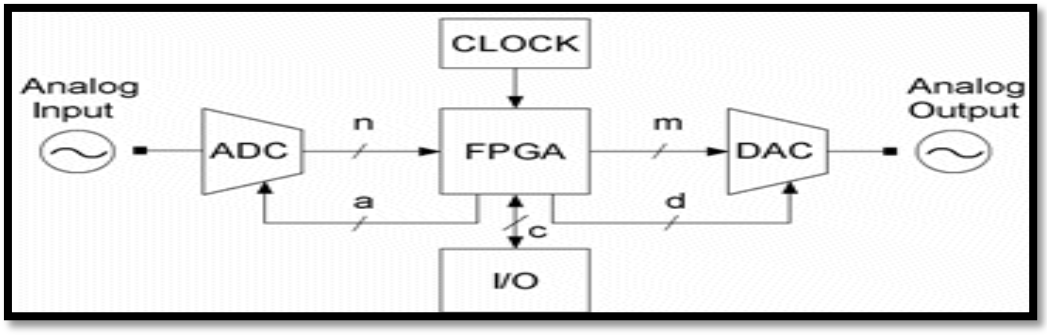


Figure 2.2. FPGA DSP flow

## 2.1. Sampling Process

Analog-to-digital transformation or digitization of analog signals is composed of the sampling (or digitization in time) and quantization (or digitization in magnitude). The sampling part represents an analog signal in a sequence of values. This process is implemented with "sample-and-hold" circuit that retains the sampled level up to the subsequent sample [4]. Figure 2.3 shows an basic DSP operation.

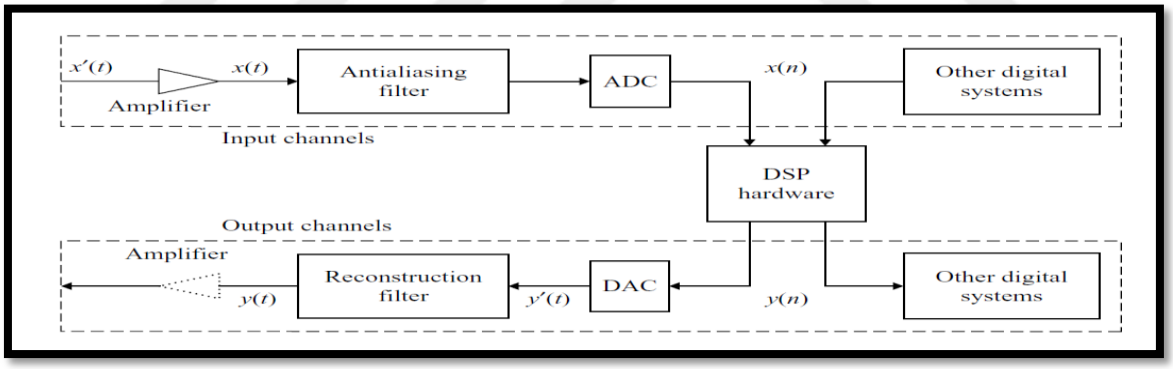


Figure 2.3. Basic DSP operation

The analog-to-digital process is carried out as follows;

- The bandlimited $x(t)$ is sampled with uniformly gap. This process transforms an analog signal in to a discrete time signal.

$$x[n] = x(nT) \qquad -\infty < n < \infty \tag{2.1}$$

where T is the sampling period and n is a positive integer.

- The magnitude of each sample is quantized in to one $2^k$ levels, where k is the number of bits which the ADC has. And these samples are coded into binary representation.

An traditional ADC may be debated as a switch which opens and closes in every T seconds. And therefore the sampling frequency is depicted as;

$$f_s = 1/T \tag{2.2}$$

where $f_s$ is in Hz. The discrete-time signal $x(nT)$ can be represented with a positive integer at discrete time nT, n = 0,1,2,…,∞ [3]. This can be represented in Figure 2.4.

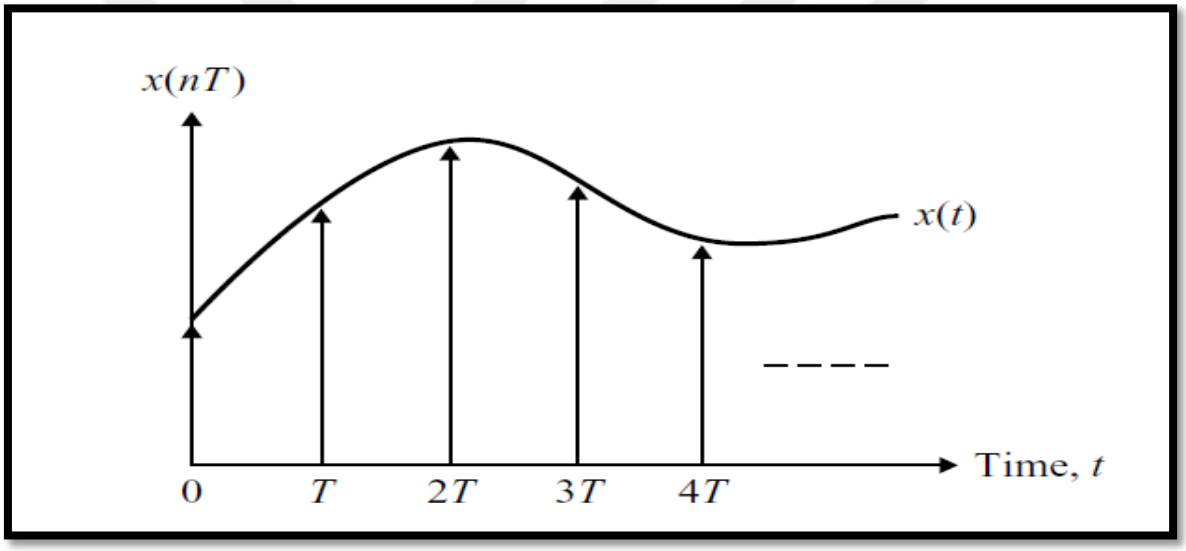


Figure 2.4. $x(t)\ and\ x(nT)$

If an analog signal $x(t)$ is represented by a discrete-time signal $x(nT)$ precisely, the sampling frequency $f_s$ need to have at least double the maximum frequency portion of the analog signal, $f_M$. It is expressed as;

$$f_s \geq 2f_M \tag{2.3}$$

where $f_M$ is commonly depicted as the "*Nyquist frequency*" and $2f_M$ which has to be overpassed by the sampling frequency is defined as "*Nyquist rate*" [4].

## 2.2. Signal Resolution

Similarity of a digital signal, in other words, how analogous a digital signal is based on, how many bits are utilized to exemplify the amplitude steps. These steps are named as *quantization steps*, displayed with;

$$\Delta = \frac{x_{max} - x_{min}}{2^b - 1} \tag{2.4}$$

where *x* is the input signal and *b* exemplify the number of bits that exemplifies the signal. Δ identifies the *quantizer step size* that are designated by an ADC [5].

Equation 2.4 indicates that a great enhancement in signal resolution in consequence of enhancing the bit width. That situation may be displayed by an analog amplitude of 1 V. Provided that a 4 bit digital signal is considered to be represented this signal, there are only 16 steps to exemplify that 1 V. Yet, provided that a 24 bit digital signal is utilized to represent that analog signal, $2^{24}$ steps are fitted. Provided that the quantization is uniform, then each quantization step has matched size, and one step would be equal to;

$$\frac{1V}{2^{24} - 1} = 0{,}0596\ \mu V \tag{2.5}$$

Provided that 4 bit digital is utilized, then each quantization step would be equal to;

$$\frac{1V}{2^{4} - 1} = 125\ mV \tag{2.6}$$

## 2.3. Quantization and Quantization Error

The quantization and encoding processes are the processes that the discrete-time signal $x(nT)$ is represented in a binary number with finite number of bits. If an ADC has k bits, then there are $2^k$ different levels which may be utilized to depict a sample. If $x[n]$ is among two quantization values, then one of the process called rounding or truncation needs to be applied. Rounding means the value of the nearest quantization level. Trancation means that the value of the level below it [5].

The difference among the quantized value and the original value is called the quantization error. The quantization error places in noise in the converted output. This can be also defined as quantization noise. If an ADC has k-bits, then *signal-to-quantization-noise-error* (SQNR) can be approached as;

$$SQNR \approx 6 * k\ dB \tag{2.7}$$

Practically, this 6k dB value is less than 6k dB because of the faultiness of the converters. For example, an 8-bits ADC has 48 dB SQNR and an 16-bit ADC has 96 dB SQNR [5]. Figure 2.5 shows two-bits quantization and errors.

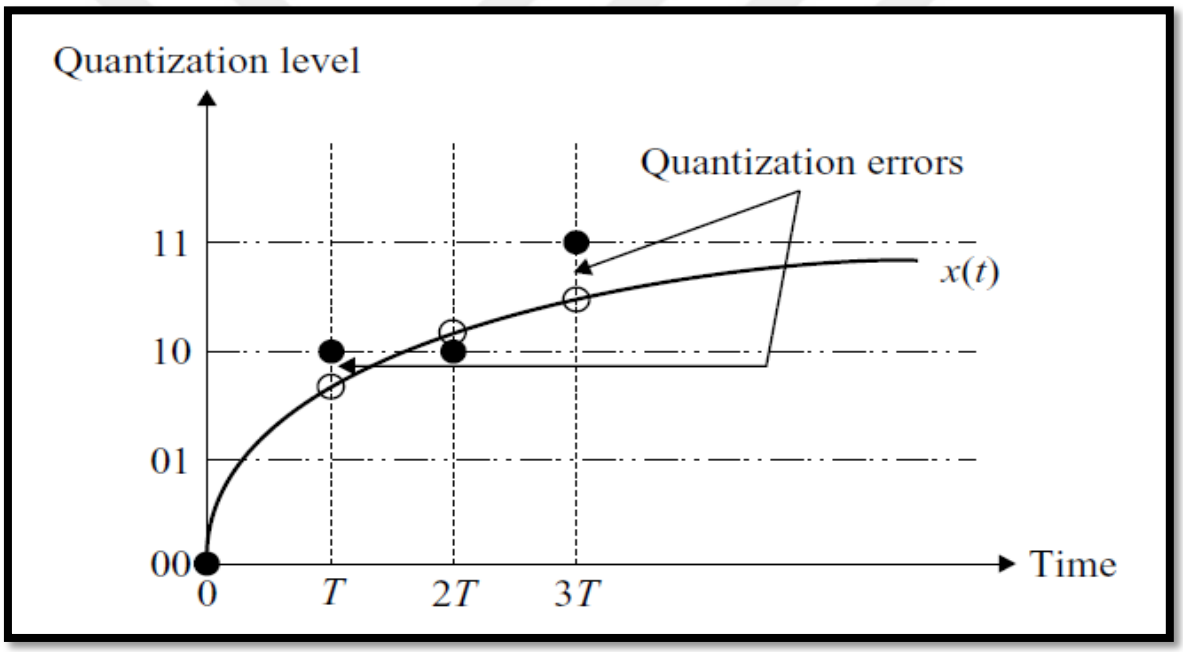


Figure 2.5. 2-bits quantization and error representation

**2.4. Digital Filtering**

Digital filtering is succeeded implementing mathematical operations on the sampled data (digitized data). In the analog filtering, filtering operations are implemented thanks to the electronic circuits. Digital filtering operation is performed on the sampled signals with the help of coefficients [6]. There are a whole range of filters, however in this thesis, one of digital and *linear time invariant* (LTI) filters named the FIR filter is utilized. These LTI filters may be categorized as: FIR and *infinite impulse response* (IIR) filters.

### 2.4.1. FIR filters

The output of LTI systems is the digitized input convoluted with the impulse response coefficients of the system. Considering that the filter is causal, the output at time *n* is given as;

$$y[n] = \sum_{l=0}^{\infty} h[l] * x[n-l] \tag{2.8}$$

The process that implement the linear convolution requires four levels;

1. Fold $x[l]$ about $l = 0$ to obtain $x[-l]$
2. Shift $x[-l]$ with "n" samples to the right to get $x[n-l]$
3. Multiply $h[l]$ with $x[n-l]$ to get the products of $h[l] * x[n-l]$ for all $l$
4. Add all the products to get the output at time n [4].

The FIR filters are chosen by users over IIR filters due to some disadvantages of IIR filters. Some the of major benefits of FIR filters is stated as following;

1. The FIR filters are steady since no feedback is present from output and the absence of poles.
2. The layout of linear-phase filters may be assured.
3. The finite precision faults are less powerful in FIR filters than IIR filters.
4. The FIR filters can be applied to the DSP processors and FPGAs efficiently [4].

In Figure 2.6 displays that the basic and the direct of the FIR filter. As seen in the Figure 2.6, $x[n]$ is the input, $y[n]$ is the output, $h[l]$ is the coefficients and $Z^{-1}$ is the delay.

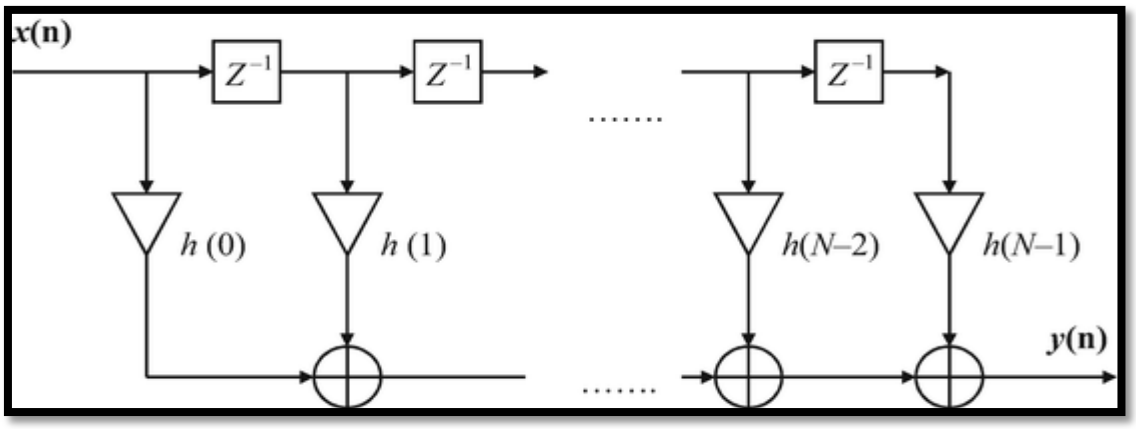


Figure 2.6. Direct form of the FIR filter

Alternatively, the form can be transformed into the transposed structure as shown in Figure 2.7. As it can be seen that the transposed structure of the FIR filter benefits from the advantage of a substantially shorter path than the conventional direct form. This advantage results in an increase in efficiency and throughput.

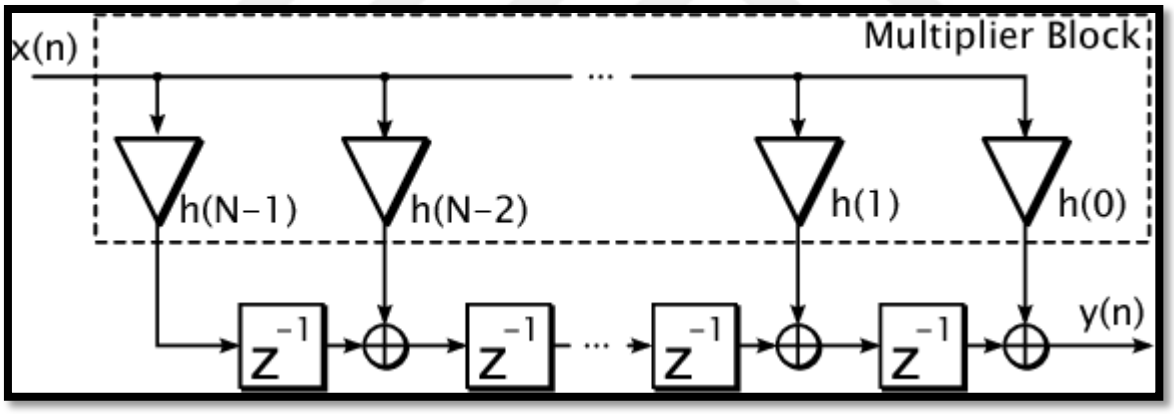


Figure 2.7. Transposed structure of the FIR filter

### 2.4.2. Low pass FIR filters

The magnitude response of an ideal low pass filter is shown in Figure 2.8. The areas $0 \leq w \leq w_c$ and $w \geq w_c$ are called as the *passband* and the *stopband* respectively. The frequency $w_c$ that divides the passband and stopband is called the *cutoff* frequency [4].

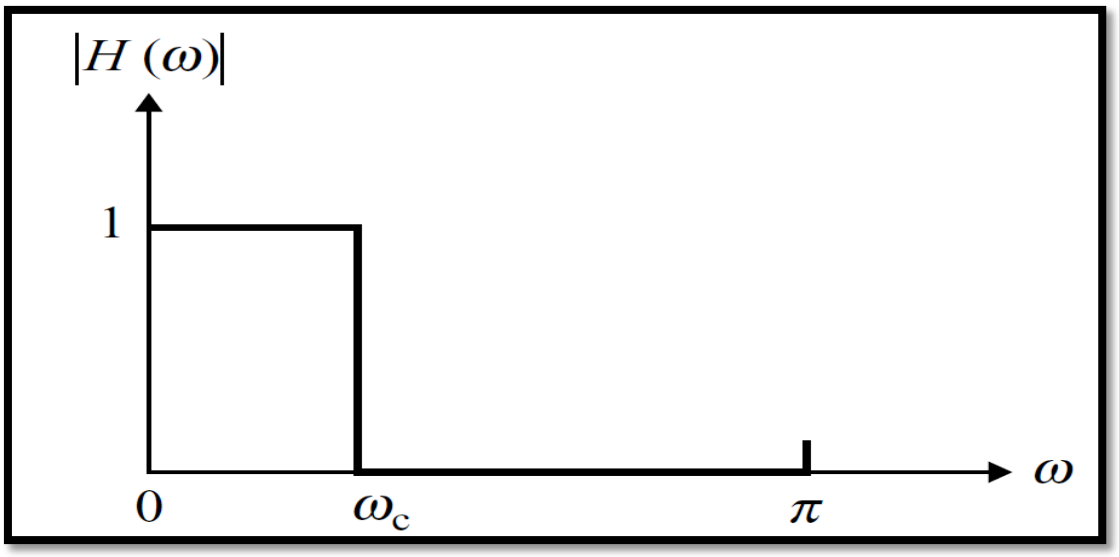


Figure 2.8. Magnitude response of the low pass filter

The magnitude of an ideal low pass which in the range of frequency has $|H(w)| = 1$ and $|H(w)| = 0$ in the range of $w \geq w_c$. Hence, a low pass filter allows the low frequency elements under the cutoff frequency and rejects the high frequency components over the $w_c$ [4].

Practically, the infinitely sharp cutoff shape as it can be seen in Figure 2.8. As expected, more graded cutoff in a transition band among the passband and the stopband needs to be considered. The features can be given in *tolerance* (ripple), schemes and the transition band is determined to allow the straight magnitude roll-off [4]. An expected and conventional magnitude response of the low pass filter is shown in Figure 2.9.

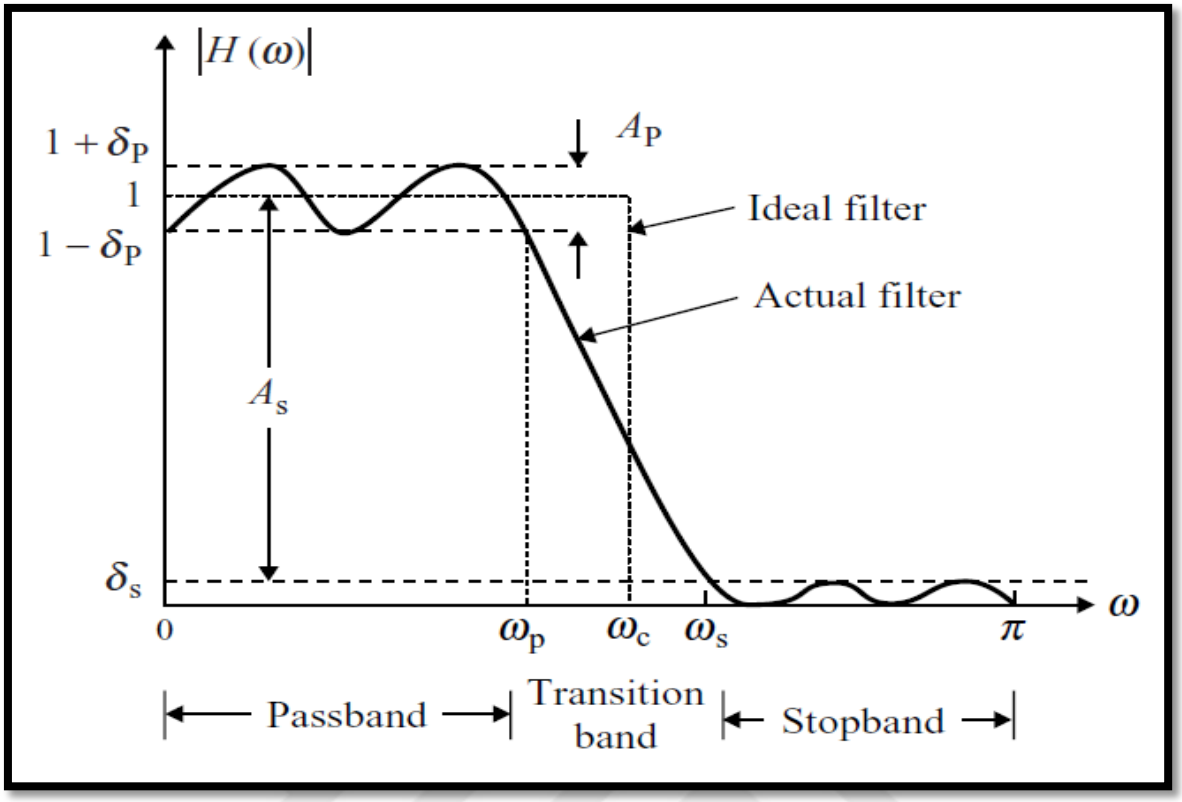


Figure 2.9. Actual magnitude response of a low pass filter

As it can be seen from the Figure 2.9 that $\delta_p$ is a peak deviate in the passband and a maximum deviate $\delta_s$ in the stopband. The magnitude of passband $(0 \leq w \leq w_c)$ approximates unity in the error of $+\delta_p$ and $-\delta_p$;

$$1 - \delta_p \leq |H(w)| \leq 1 + \delta_p, \quad 0 \leq w \leq w_p \tag{2.9}$$

The passband ripple $\delta_p$ in the passband is permitted in the magnitude reply. The gain of the magnitude reply is normalized to *1* (0 dB). Likewise, in the stopband, the magnitude response closes to zero with an error $\delta_s$;

$$|H(w)| \leq \delta_s, \quad w_s \leq w \leq \pi \tag{2.10}$$

The stopband ripple $\delta_s$ above the $w_s$ is described as the minimum attenuation for signal components. The peak passband attenuation and the minimum stopband ripple in decibels are considered as follows:

$$A_p = 20\log_{10}\frac{1+\delta_p}{1-\delta_p} \text{ in dB} \quad (2.11)$$

$$A_s = 20\log_{10}\delta_s \text{ in dB} \quad (2.12)$$

### 2.4.3. The systolic MAC filter architecture

In this thesis, the systolic *multiply accumulate* (MAC) architecture is used for optimum performance, efficiency and storage. Figure 2.10 displays the simplified illustration of a MAC FIR using a single MAC engine.

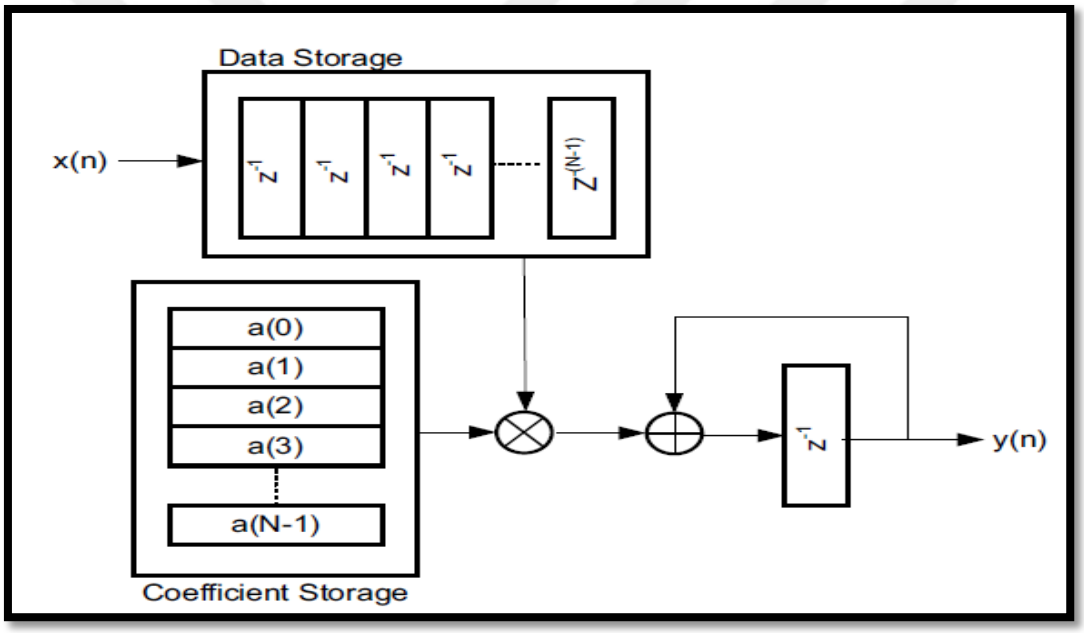


Figure 2.10. Single MAC engine

This single application can be expandable to multi-MAC applications for usage with managing superior performance filter specifications (greater number of coeficients, superior sample rate and more channels) [7].

A range of multipliers needed to apply and the filter is configured by calculating a range of multiplies needed to implement the calculation and dividing by a range of clocks present to manipulate each input sample. The present clock cycles rate is all the time rounded down and a range of multipliers rounded up to the nearest value [7].

The output samples represent the padding of the coefficient vector. Therefore, a reply to an implemented impulse includes a range of zeros prior to the first coefficient of the designated impulse response occurs at the output. The FIR compiler automatically forms the application which includes the user-defined performance needs with respect to the system clock rate, the sampler rate, the number of taps and channels [7].

This architecture is practically asisted by the DSP-slice and makes possible are productive and performance filter implementations. Moreover, this architecture expands to utilize coefficient symmetry, and hence including farther resource savings [7]. Figure 2.11 illustrates multi-MAC implementation.

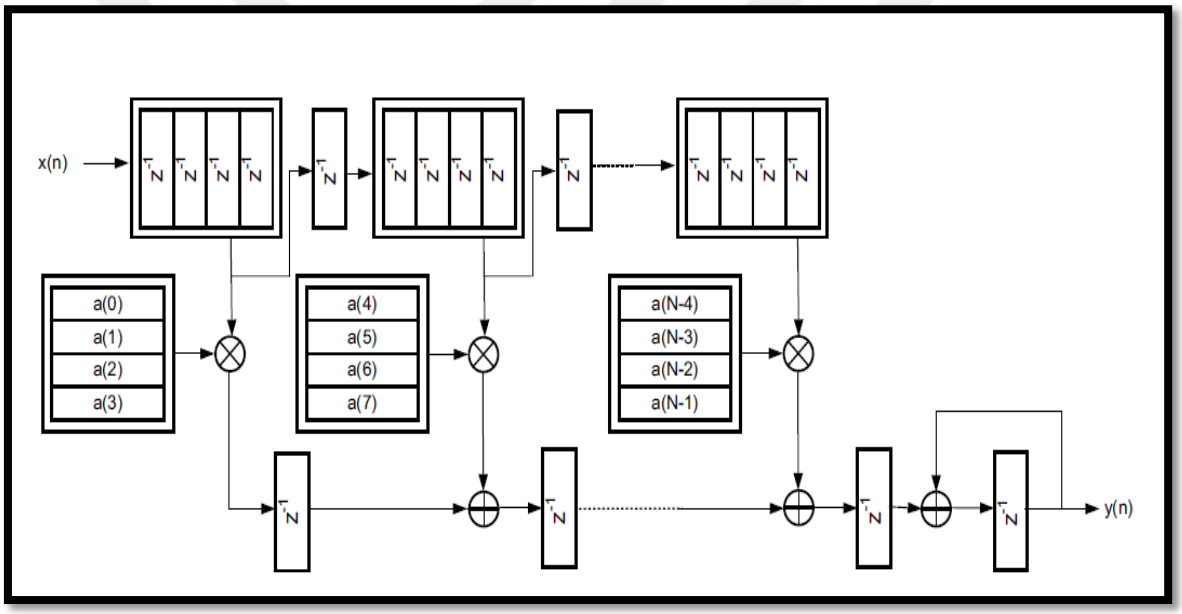


Figure 2.11. Multi-MAC engine

# 3. HARDWARE

## 3.1. Fixed Point Representation

A commonly utilized format for symbolizing and storing binary data is the fixed-point format, where a value "*a*" symbolized by $a_{m+n-1}, a_{m+n-2}, \dots, a_0$ is mapped as $a_{m+n-1}, a_{m+n-2}, \dots, a_n$ symbolizes the integer part of the data and the statement $a_{n-1}, a_{n-2}, \dots, a_0$ symbolizes the fractional part of the data [6].

The major point while deciding on a fixed-point representation is the usage of dynamic range in calculation. Scaling may be implemented to overcome the worst situation, however that cause low dynamic range. Altering to have the topnotch implementation of the dynamic range shows that *overflow* will occur in some situations and extra circuitry need to be applied to get rid of that issue. This issue usually occurs in *two's complement* that causes entirely different sign. This issue may be fixed by applying saturation circuitry to protect the worst situation negative or positive overflow, however that has a nonlinear effect on performance [6]. The effect of overflow in two's complimentary can be shown in Figure 3.1.

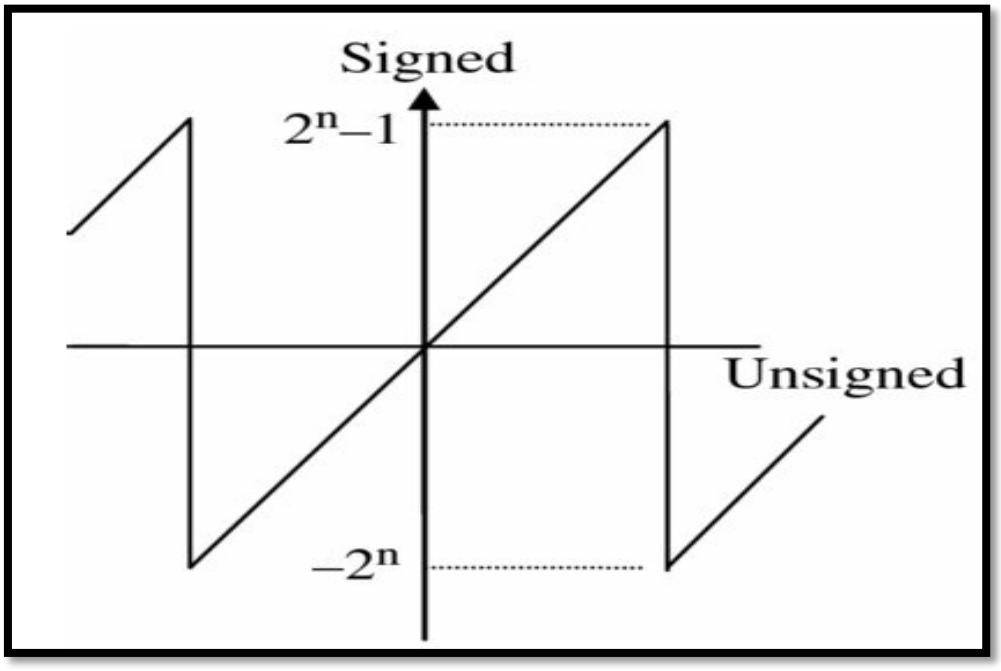


Figure 3.1. Effect of overflow in two's complimentary

If the addition of 7 (0111) and 1 (0001) is taken into consederation, it can be seen that 8 (1000) is the result in unsigned binary, however this value symbolizes -8 in two's complimentary that symbolizes the worst situation. A solution can be implemented in circuitry that saturates at the output data which is 7 (0111) [6]. Figure 3.2 can display saturation operation.

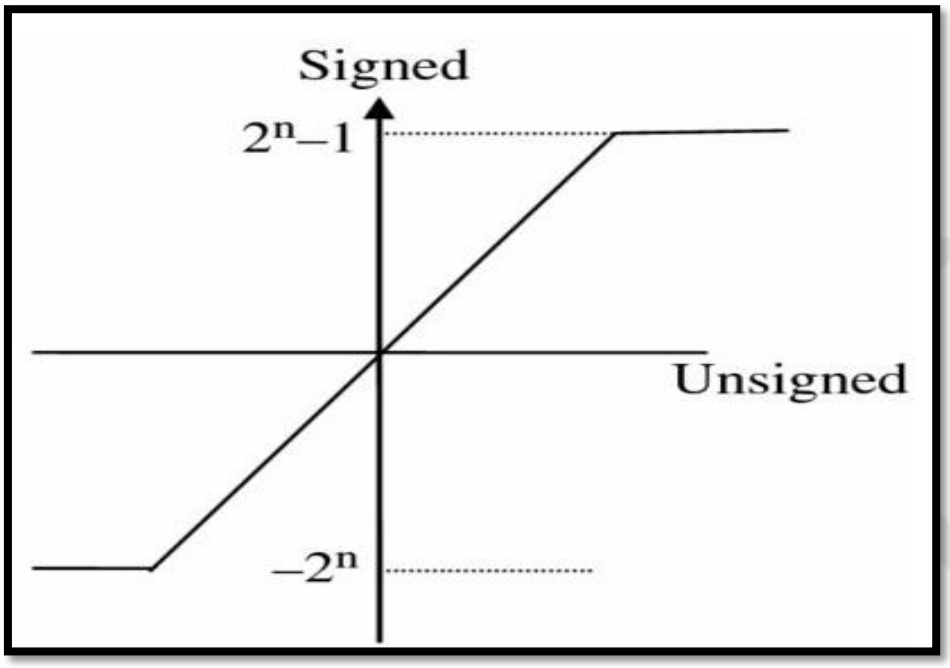


Figure 3.2. Saturation operation

### 3.2. Structure of an FPGA

The principal implementation of the FPGAs used by clients is to build customized system using programmable logics. FPGAs are made of programmable logic blocks which are interconnected in the system. At the edges, customizable I/O blocks are designated to communicate with outside the world [8]. Figure 3.3 illustrates a basic structure of a conventional FPGA.

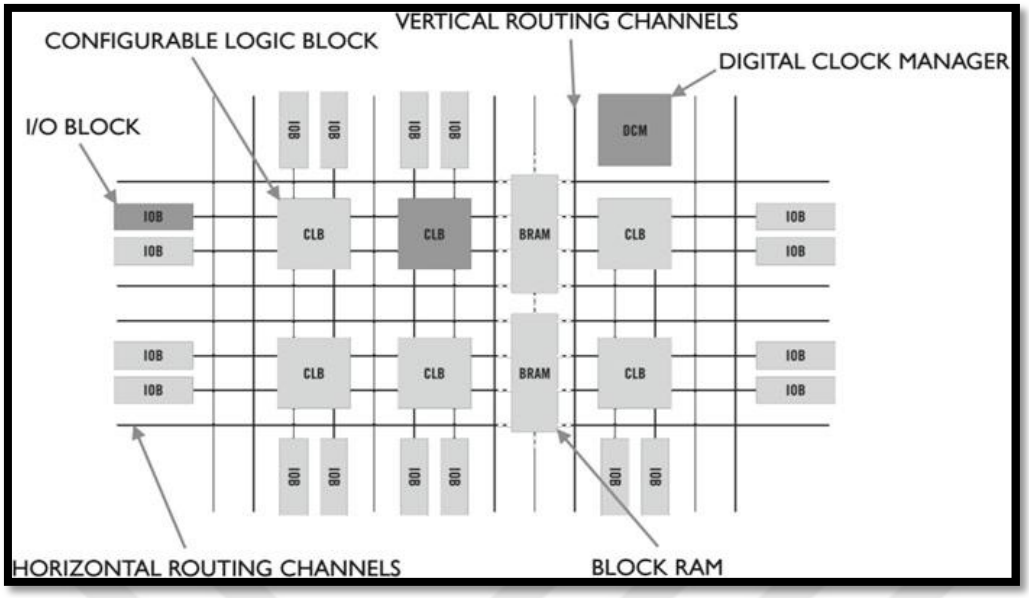


Figure 3.3. Basic structure of a conventional FPGA

Carry chains, block *random access memories* (RAMs) and DSP blocks may be executed in configurable logics. And, all of the FPGAs contain these specifications. Clients take advantage of these specifications and execute them in *look-up-tables* (LUTs). Since the FPGAs are being improved day by day, these specifications are being more cost friendly and more efficient than obsolete FPGAs [8].

Even if all FPGAs are common in structure, their configurable logic amount differ from each other due to the vendors. This situation permits the clients to choose the right FPGA device for their needs. And also, the FPGAs differ from each other in speed, tempereture and voltage perspective. The fastest FPGAs are roughly 25% faster than the slowest FPGAs. Cost and performance wise specifications are choosable for clients [8].

Clock speed of the latest FPGAs are in the range of 100-500 MHz. Generally, conventional designs require low-cost FPGAs and these FPGAs can operate at low frequency range. More complex FPGAs can operate at mid frequency range. DSP targeted designs require the highest frequency range and they make use of intensely DSP blocks and *block RAM* (BRAM) [8].

### 3.2.1. Interconnection

The structure of a conventional FPGA is composed of links that is linked together to link any two blocks in an FPGA. This feature permit the clients take advantage of random logic networks. If the client wants to observe the interconnected links, vendor tool can come to aid [8].

### 3.2.2. Configurable logic blocks

The Xilinx *configurable logic blocks* (CLBs) are composed of 4-6 bit LUTs, flip-flops, a configurable carry chain and configurable hardware to enable customized configurations to be created [8]. Figure 3.4 gives an idea about an configurable logic block of an FPGA.

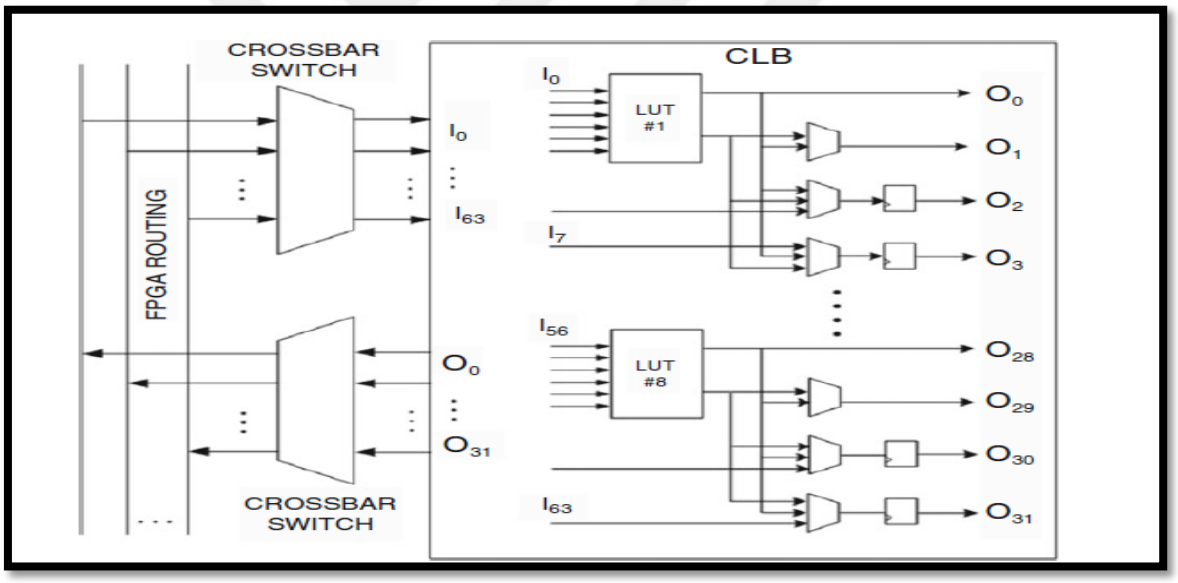


Figure 3.4. Configurable logic block

The combined form of the LUT, carry chain, flip-flop and registers is defined as a *logic cell* (LC). The capable of an FPGA is generally obtained with logic cells. Xilinx Virtex series FPGA is for example composed of 4 million LCs, as the slowest Spartan is composed of 2000 logic cells [8].

A elevated performance carry chain permits the FPGA to execute very superior performance adders. Carry chains of the latest FPGAs may execute 64 bit adder at 500

MHz. Registers in the FPGA permit *pipelined* designs, that are play important role in exploiting faster speed [8].

### 3.2.3. Memory

Xilinx shows us various memory sizes and customizations. The memory is configured as columns displayed in Figure 3.5.

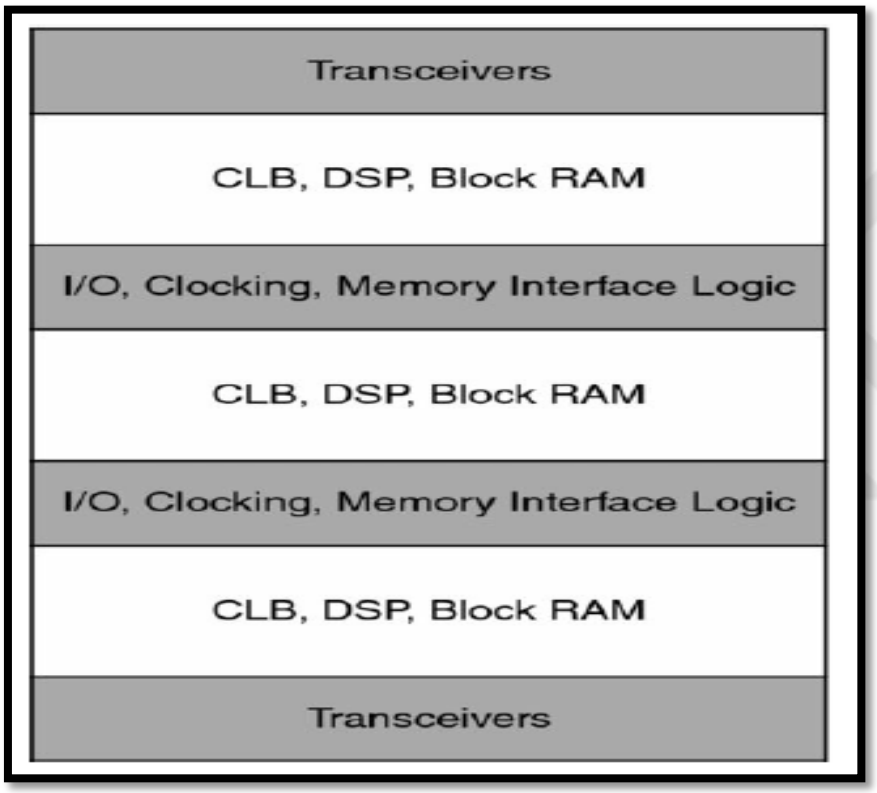


Figure 3.5. Organized memory blocks

Memory accessability is sychronized to the system clock and inputs, data address, *clock eables* (CE) and *write enables* (WE) are filed with an option to turn off address latching An another output data pipeline register permits higher clock rates however, it costs additional cycle of latency [6].

Block RAMs may be customized vertically to build extended, fast memory arrays. Also *first in-first out* (FIFOs) with widely decreased power consumption take important place in FPGAs. BRAM blocks which stand unconsumed in the design are automatically turned off.

BRAMs may be configured as 32K * 1-bit, 16K * 2-bit, 8K * 4-bit, 2K * 16-bit and 1K * 32-bit [6].

Registers or flip-flops are utilized to status and control registers, pipelining and shallow FIFOs. Shift registers are generally utilized to signal delay and pipeline stability in DSP. *Distributed RAMs* (DRAMs) are utilized to shallow memories until 64-bit deep. BRAMs are also utilized for buffers and deeper memories [8].

### 3.2.4 DSP blocks

The latest FPGAs are composed of discrete multipliers to allow efficient DSP processing. Conventional DSP implementations exploit pipelines or flow graphs and data streams [8]. Figure 3.6 illustrates a basic DSP flowgraph.

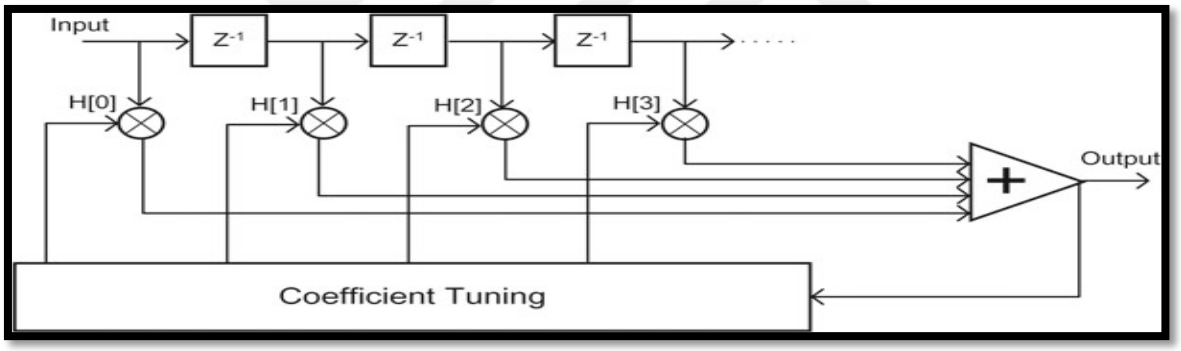


Figure 3.6. Basic DSP flowhgraph

In order to execute digital filtering and DSP implementations, multipliers and adders are utilized to apply the flow graph. Xilinx FPGAs include a DSP block named as a DSP48 that include an 18-bit * 27-bit multiplier, a 48-bit accumulator and a 27-bit preadder. Four levels of pipelining at 500 MHz may be implemented in high-end Xilinx devices [8]. Figure 3.7 can give an idea of a DSP48 block.

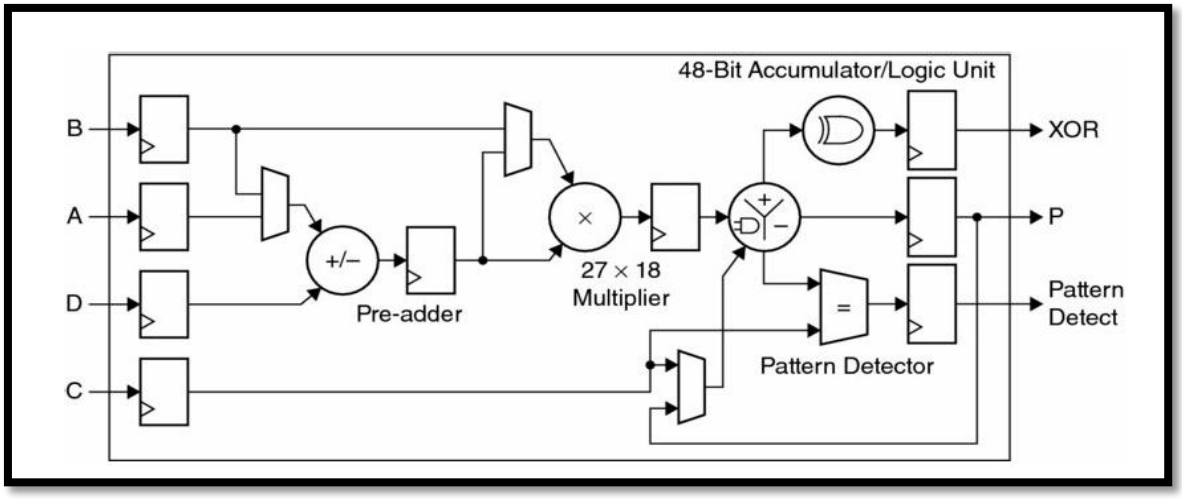


Figure 3.7. DSP48 block diagram

### 3.2.5. I/O blocks

One of the major feature of an FPGA can be that it accepts external input and output signals. In order to make possible this feature, the latest FPGAs include blocks named I/O blocks. I/O blocks include high-powered buffers to support external signals and include input receivers with registers to I/O signals. Figure 3.8 can display configurable I/O blocks.

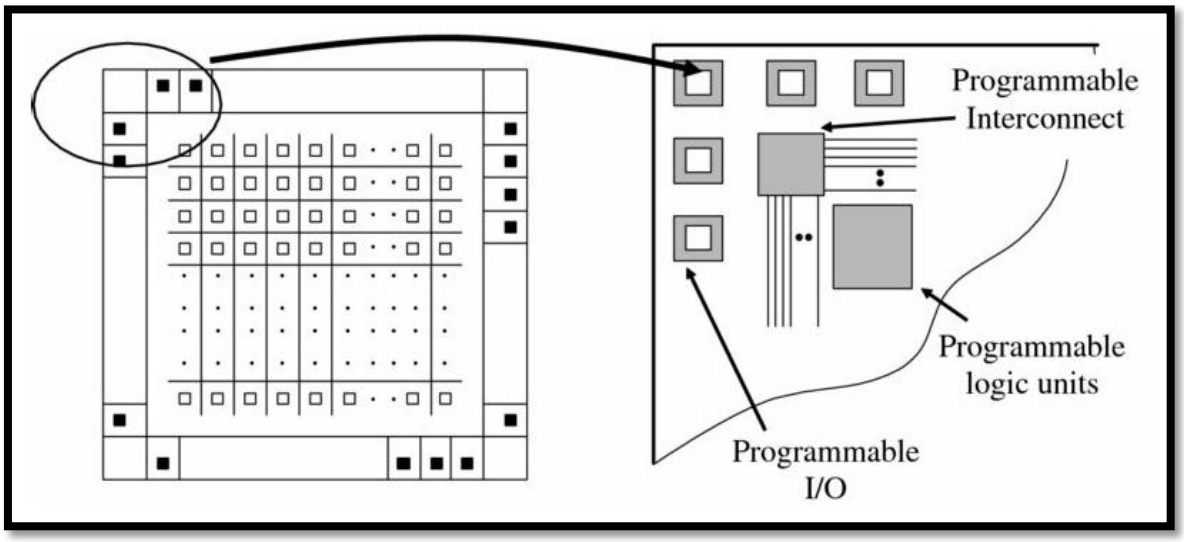


Figure 3.8. I/O blocks representation

I/O blocks can be driven a range of 1.2 – 3.3 V CMOS and LVDS as well. These I/O blocks can be different among the FPGAs. If a client wants low – cost FPGA, then there

will be low amount of I/O blocks. For high – cost FPGAs, there will be tremendous amount of I/O blocks [8].

### 3.2.6. Clocking in an FPGA

Xilinx FPGAs can support up to 24 clocks. Generally, the customized designs need roughly 12 clocks. The untouched clock lines may be divided into smaller clock lines. That may provide hundreds of smaller clock lines [8]. Figure 3.9 can illutrate this system.

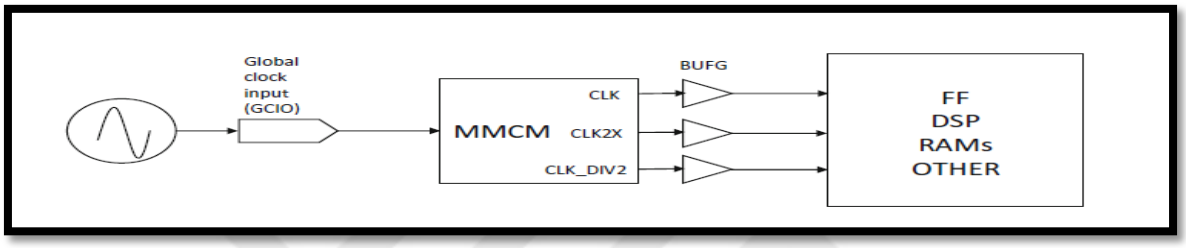


Figure 3.9. Clock network

Vivado can handle until 24 clocks with no problem. Also, there is a possibility that design of a user can have more than clock line. But, this possibility can not pass 24 clocks [8].

The FPGAs can support powerful clock management an sythesis to clock management of the particular device. Besides, the user should remember that the clocking of the customized design need to suffice to support low – latency synchronous connections utilized in DSP systems [6]. Figure 3.10 shows high level clock networks based on design.

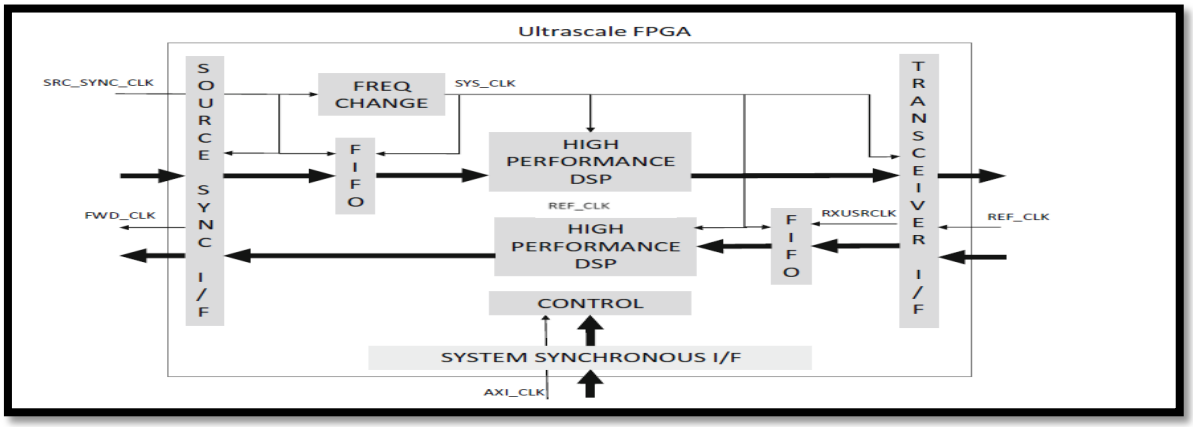


Figure 3.10. High level clock network

### 3.3. Microblaze Soft Core Processor

The Microblaze *central processing unit* (CPU) is defined as a 32-bit *reduced instruction set computer* (RISC) microprocessor, optimized to application in Xilinx FPGAs and *system on chip* (SoC). Xilinx integrated the Microblaze to the Vivado design environment. And thus, the users can utilize the Microblaze with their peripherals, memory and interface features. It shows that the Microblaze processors can be utilized as an alternative to ARM Cortex CPUs [9]. Figure 3.11 illustrates the block diagram of the Microblaze processor.

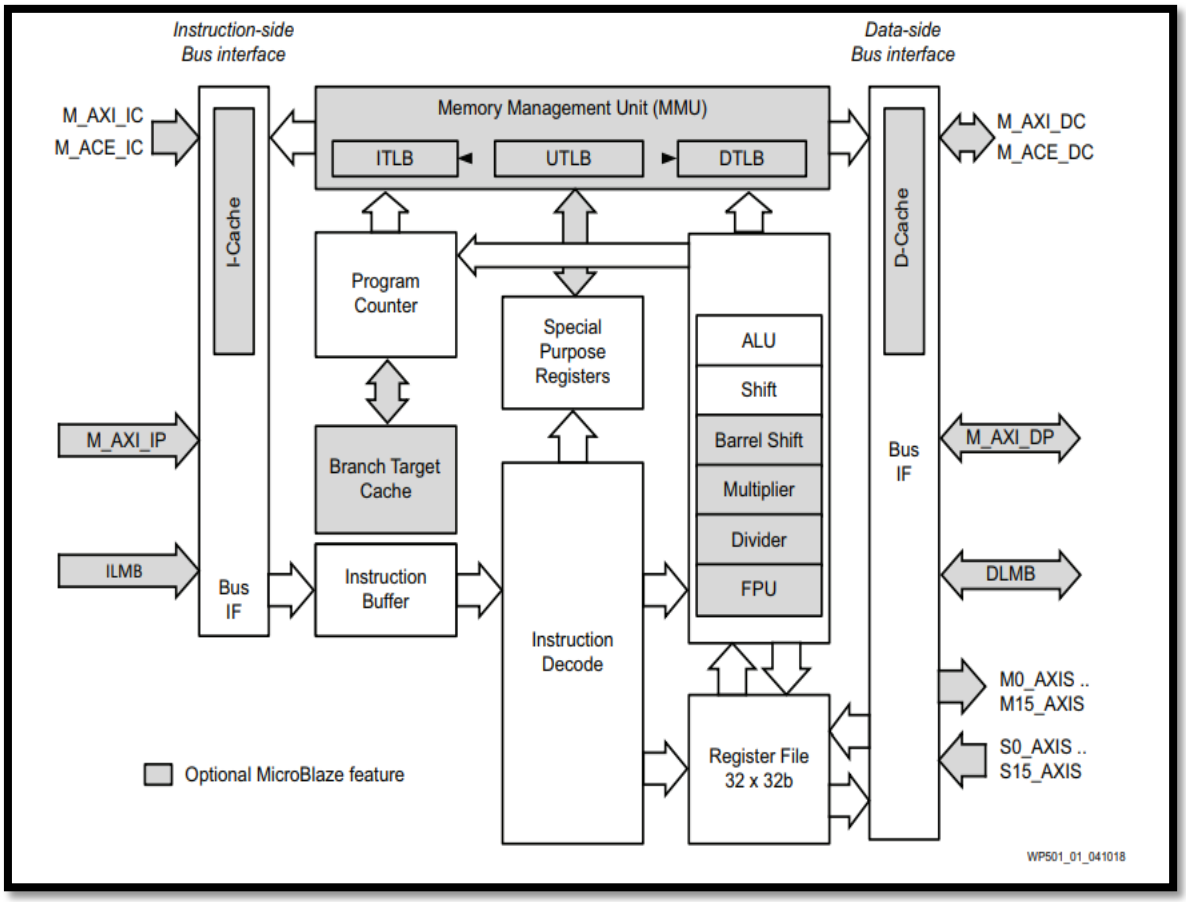


Figure 3.11. Block diagram of the Microblaze processor

The Microblaze processor can be used with lots of *intellectual property* (IP) blocks. These blocks are as follows: Multichannel *direct memory access* (DMA), *ethernet subsystem*, *controller area network* (CAN), streaming FIFO, *high definition multimedia interface* (HDMI) camera/display interface, video DMA, timer/watchdog, *universal asynchronous receiver transmitter* (UART), *universal serial bus* (USB), *quad serial peripheral interface* (SPI), *general purpose input output* (GPIO) *and pulse width modulation* (PWM) [9].

In case of situations at which a user need more process power, the Microblaze processors can come aid with either one or two ARM Cortex – A9 processors. They may run independently or in conjunction with applications which use the whole cores. In addition, users also create their processors and they may obtain from IP vendors and open source [9]. Figure 3.12 can give idea of combined the Microblaze and ARM processors.

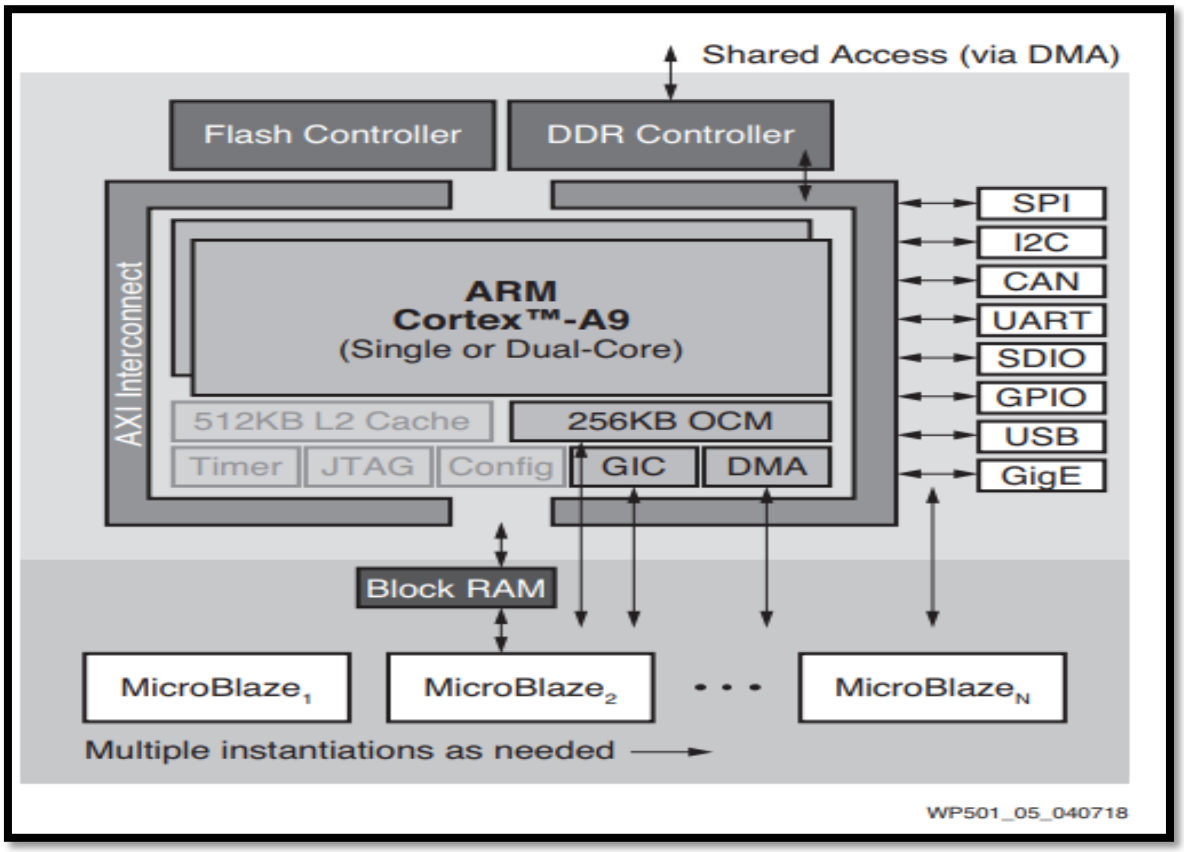


Figure 3.12. Combined form of the Microblaze and ARM processors

### 3.4. AXI Bus

AXI is the category of ARM AMBA, a part of microcontroller buses first released in 1996. There are there types of AXI interfaces;

1. AXI4, for high performance memory-mapped applications.
2. AXI4-Lite, for simple, low throughput memory-mapped communications
3. AXI4-Stream, for high speed streaming data [10]

The AXI interfaces define an interface among a single AXI master and a single AXI slave. And, reflect IP cores which change information with each other. AXI4 and AXI4-Lite interfaces are composed of five different channels as follows;

- Read Address Channel
- Write Address Channel
- Read Data Channel
- Write Data Channel
- Write Response Channel

Data may transmit in either directions among the master and slave simultaneously and amount of data transfer may differ. The limitation for AXI4 is a burst transaction and up to 256 words of data transfers. However, AXI4-Lite is 32 words of data transfer per transaction [10]. Figure 3.13 a and b shows read and write transactions respectively.

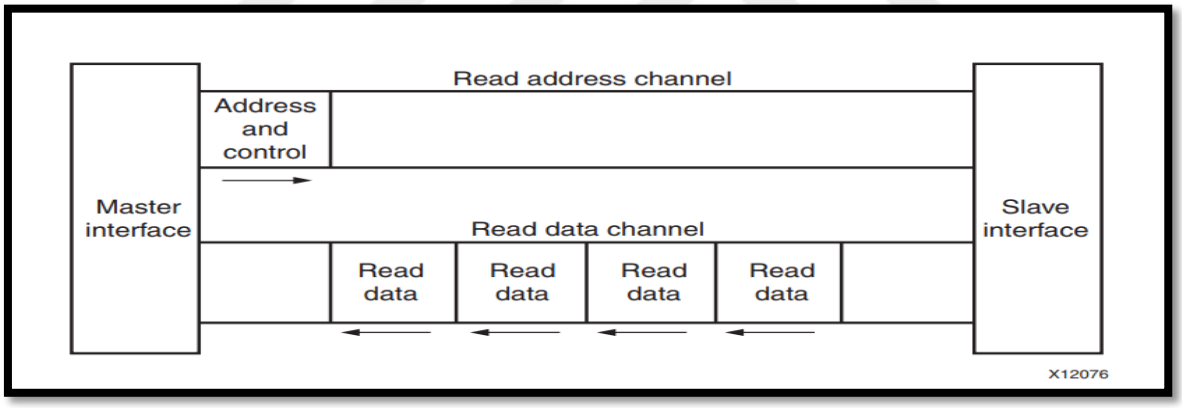


Figure 3.13. a) Read transaction

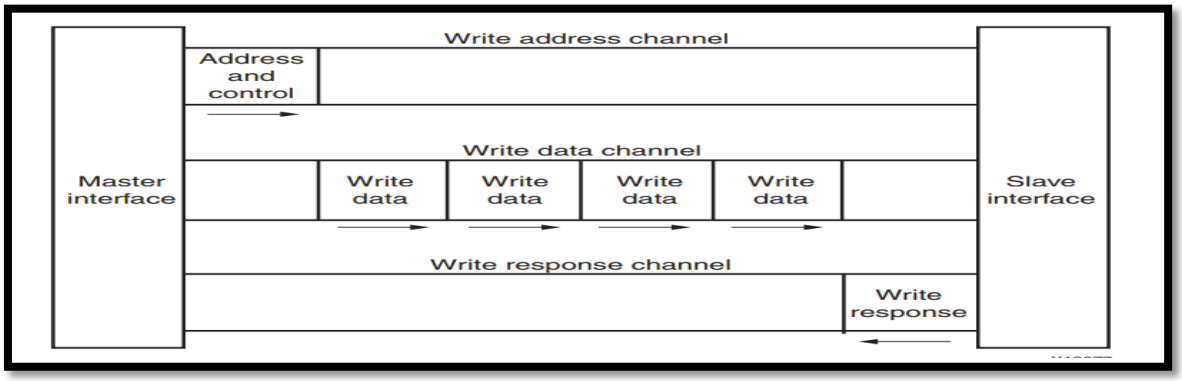


Figure 3.13. b) Write transaction

The AXI4-Stream protocol can be utilized in to implementations which generally include a data-centric and data-flow paradigm that the address is not available or not needed. The AXI4-Stream is a single unidirectional channel for a handshake flow [10].

### 3.4.1. AXI4-Stream

The AXI4-Stream interface allows the user to operate IP blocks each other with this standard. IP blocks with AXI interface compatible consists of *tvalid*, *tready*, *tdata*, *tlast* and *tuser*. tvalid and tready apply handshake to transmit the message, where the mission load is tuser and tlast. IP blocks manages on the data with the input data channel named *s_axis_data_tdata* and the output data channel *m_axis_data_tdata*. The IP blocks also allow to transmit a user port in tuser and tlast signal from input data channel to the output data channel with the same latency for the data [11].

### 3.4.2. Handshake protocol

In this handshake protocol, tvalid is driven by the master side and tready is driven by the slave side. tvalid shows that the data in tdata, tuser and tlast is effective. tready shows that the slave side is waiting to get data. As tready and tvalid are both correct in the cycle, then the data transfer happens [11].

tlast may electively be utilized to show the last sample in a cycle of interleaved data. Likewise, tuser may optionally be utilized to transmit the user field and/or channel ID field [11]. Figure 3.14 display the illustration of the basic handshake protocol.

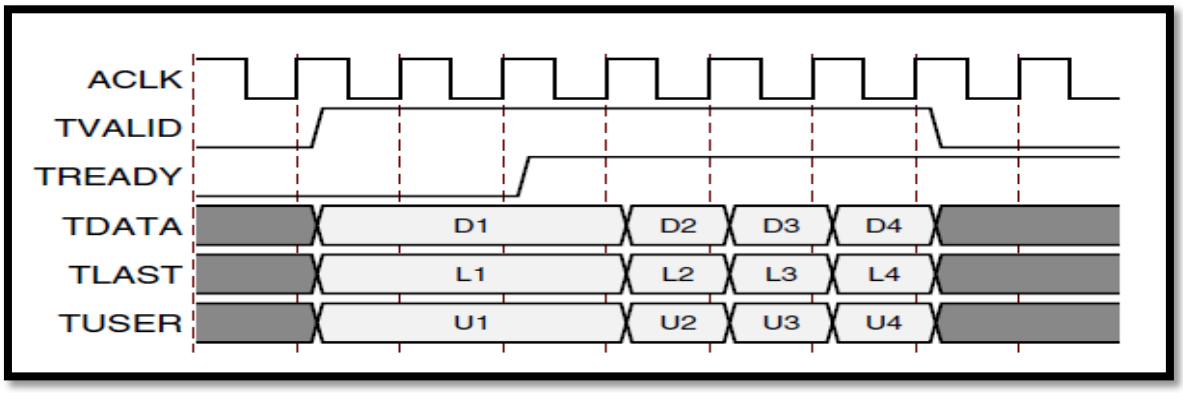


Figure 3.14. Basic handshake protocol

### 3.5. Integrated Logic Analyzer

The customizable *integrated logic analyzer* (ILA) IP core is a logic analyzer that may be used as a monitoring device to observe the signals at the system. Signals in the design are attached to ILA core clock and probes. Connected inputs are sampled at design speed and stocked up using BRAM [12]. Figure 3.15 figures the ILA IP core.

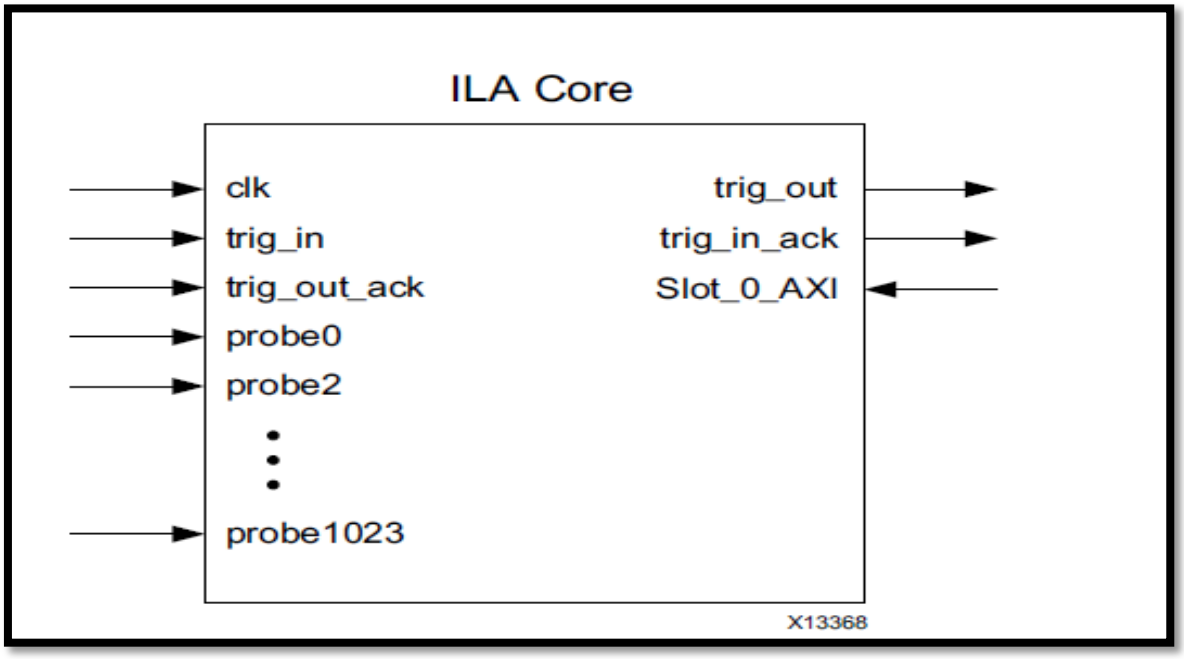


Figure 3.15. ILA IP core

### 3.6. Xilinx Analog to Digital Converter

The *Xilinx analog-to-digital converter* (XADC) provides a dual 12-bit, 1 *Mega sample per second* (MSPS) ADC and on-chip sensors. This ADCs include a general purpose, high precision features for variable applications [13]. Figure 3.16 displays the block diagram of the XADC.

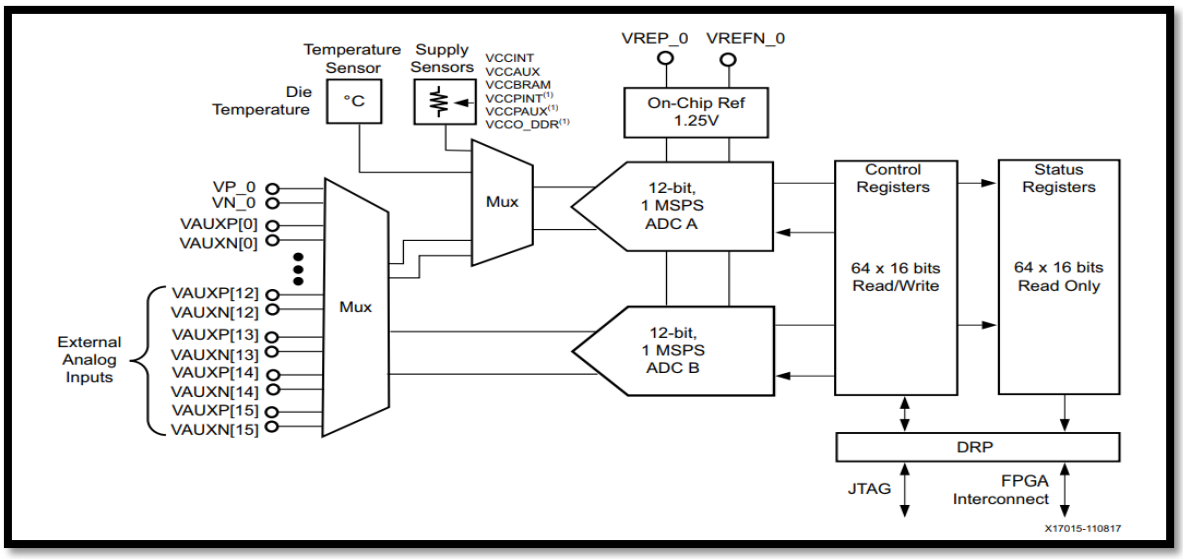


Figure 3.16. XADC block diagram

### 3.6.1. Unipolar mode

The analog input range is 0-1 V in this mode. The XADC designates “000h” when 0 V is applied on the XADC and “FFFh” when 1 V is applied onto the input [13]. Figure 3.17 displays transfer function for unipolar mode and Figure 3.18 displays range of unipolar input.

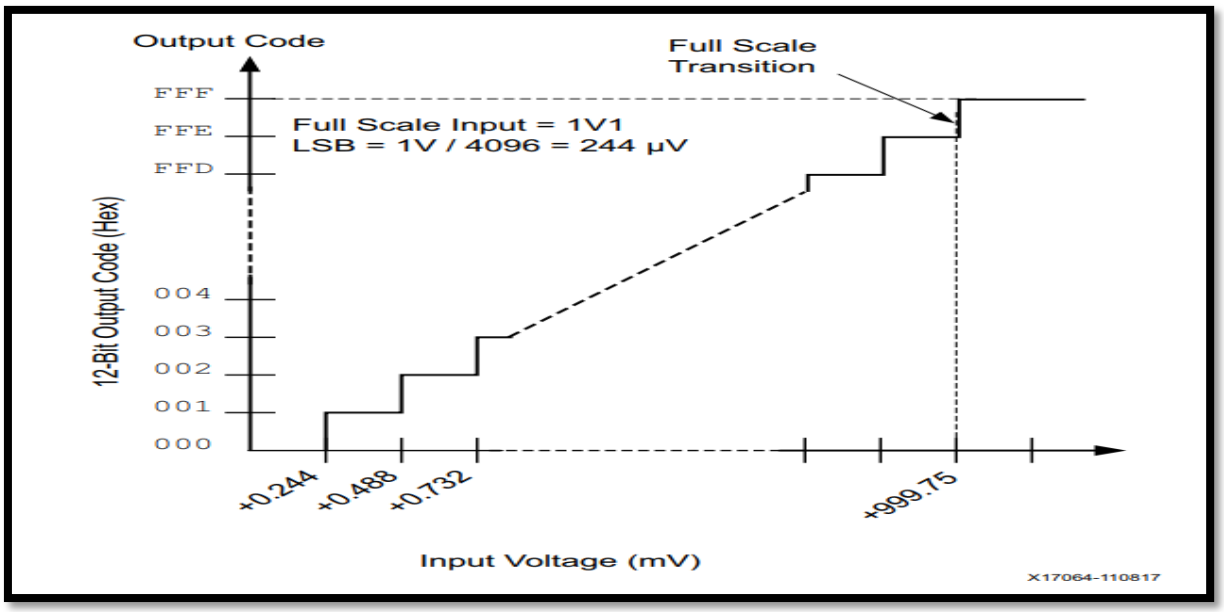


Figure 3.17. Transfer function for unipolar mode

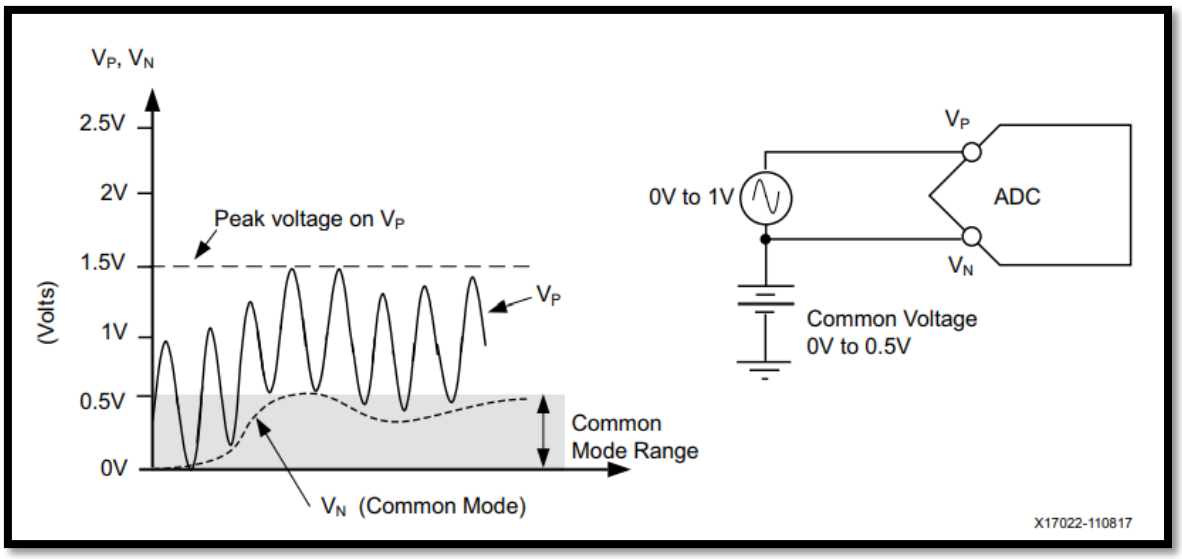


Figure 3.18. Unipolar input range

### 3.6.2. Bipolar mode

The analog input range is -0.5-0.5 V in this mode. In observing differential signal type, this mode may be useful to get information about the analog input signal. “000h” is designated to 0 V. When -0.5 V is applied to the input, “800h” is designated. When 0.5 V is applied to the input, “7FFh” is designated. As it can be seen that two’s complimentary is used to show that relative Vn and Vp [13]. Figure 3.19 displays the transfer function for bipolar mode and Figure 3.20 displays the bipolar input range.

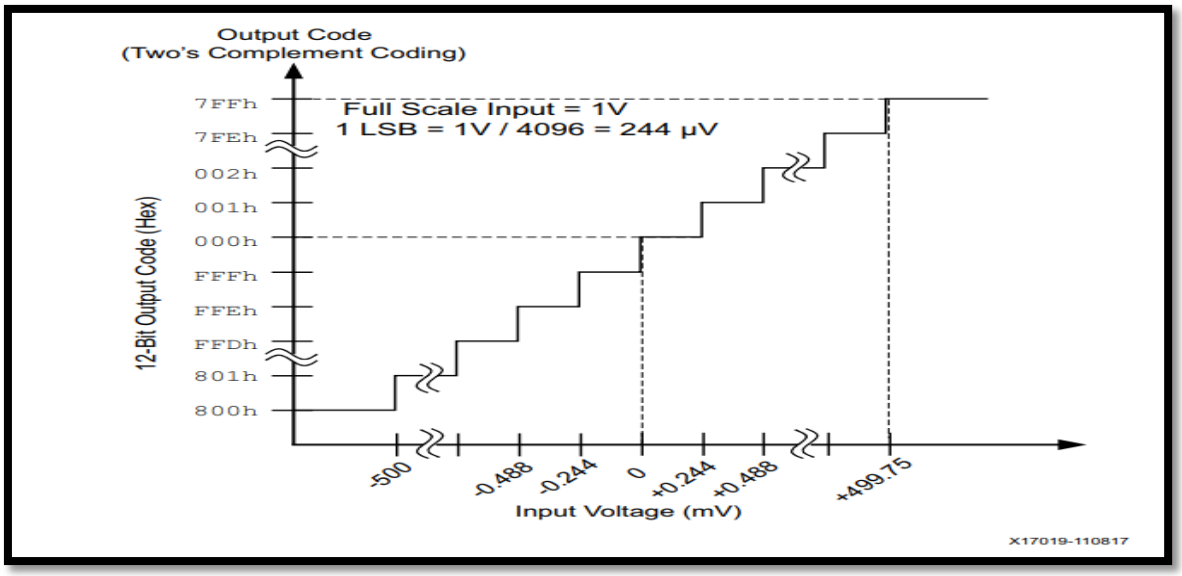


Figure 3.19. Transfer function for bipolar mode

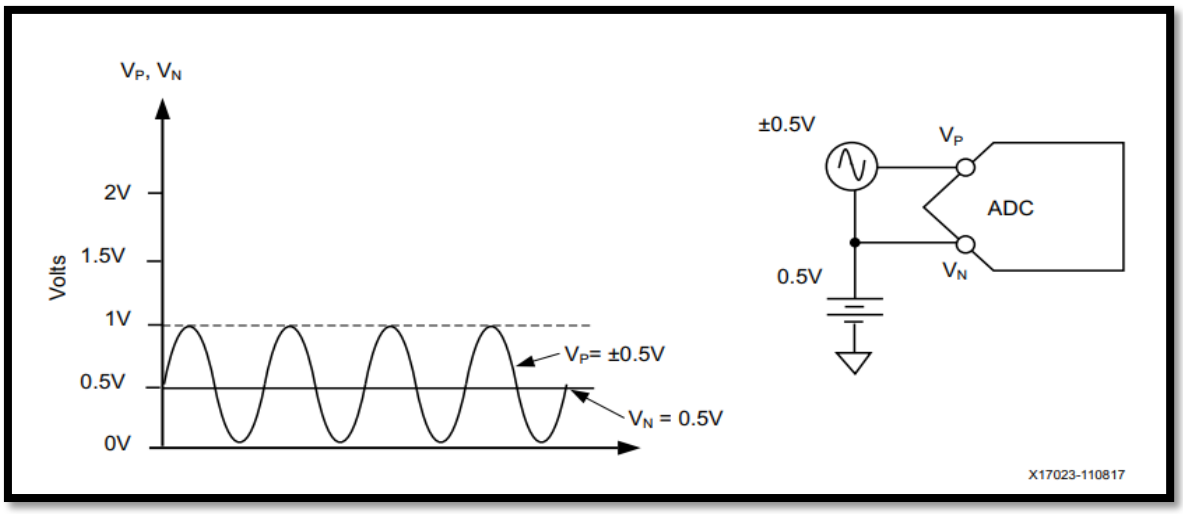


Figure 3.20. Bipolar input range

### 3.7. Digital to Analog Converter

A DAC is a device that converts a digital signal to analog signal. In this thesis, 12 bit output, 12 bit parallel input and 20.4 MSPS DAC that is manufactured by Texas Instruments model of 7821. This DAC utilizes from parallel input capability that brings up top notch performance. In Figure 3.21, the schematic of the DAC can be seen.

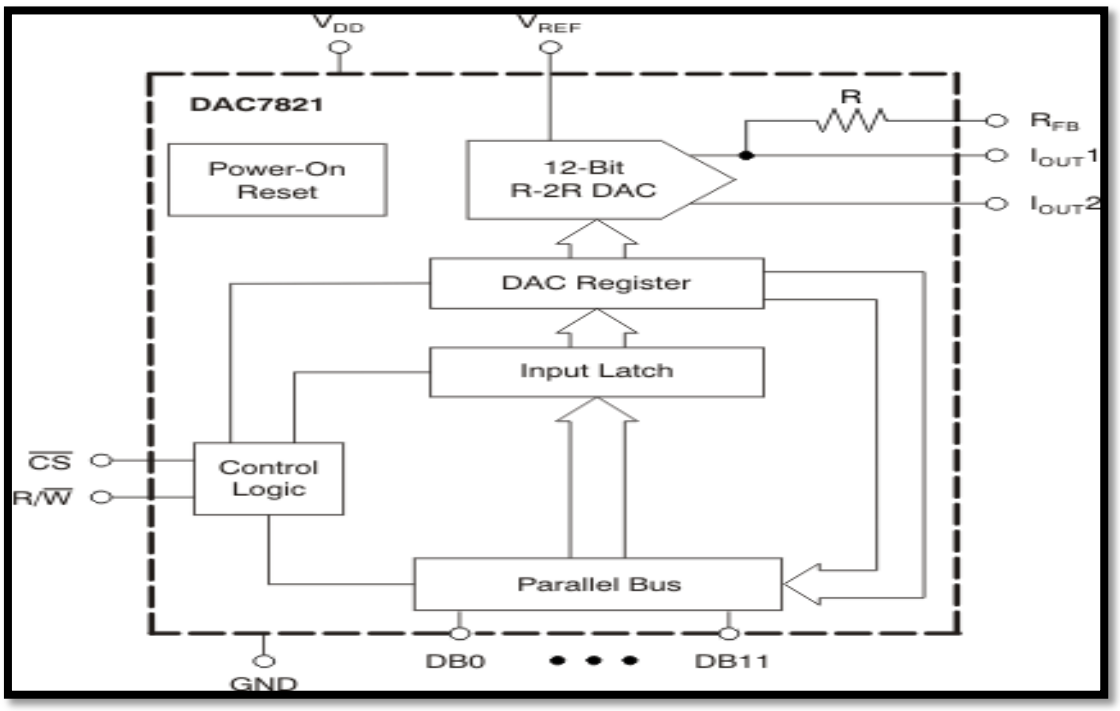


Figure 3.21. Digital to analog converter schematic

# 4. SOFTWARE

## 4.1. IP Blocks

IP blocks are essential benchmark as choosing which FPGA provider to select to a specific design. These IP blocks allows the user to include mixed design in their designs from high speed GHz transceivers to DSPs, from Microblaze soft core processor to ARM Cortex A9. Xilinx based IP blocks are optimized for users to facilitate from DSP, image processing, high speed transceivers, etc [8]. Figure 4.1 displays the design flow of the IP blocks.

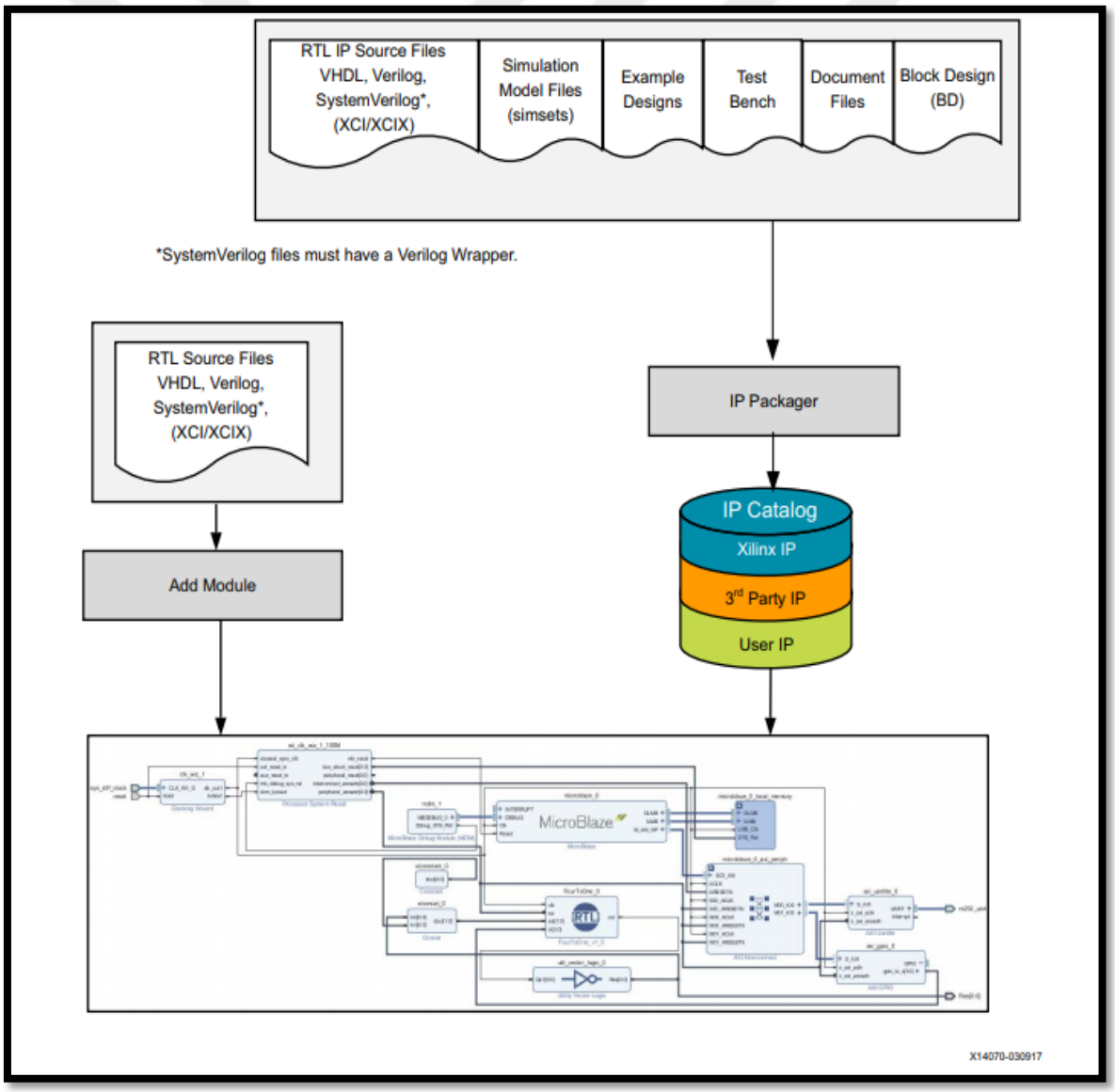


Figure 4.1. IP block design flow

Other than using Xilinx-based IP blocks for the designs, Xilinx allows the clients to expand or to recreate IP blocks for their designs and reuse the IP blocks created by the clients. These IP blocks are creared as follows;

- Using with Vivado *High Level Synthesis* (HLS) tool (users can use C/C++ programming languages to create and compile IP blocks).
- Using with System Generator for DSP designs (users can use Simulink to create and simulate their designs).
- Using with third-party IP vendors.
- Using with Vivado IP packager tool (users can use this tool to create reusable IP blocks and add the blocks their IP catalog).

### 4.2. XADC Wizard

The XADC wizard creates an *hardware description language* (HDL) wrapper to customize XADC for users to configure channels, operations and alarms. Besides, converting analog to digital, it monitors the tempereture and voltage of the FPGA, generates alarms to user-specification, is suitable for AXI interface and calculates all the parameters and register values [14]. Figure 4.2 illustrate the detailed block diagram of the XADC IP block.

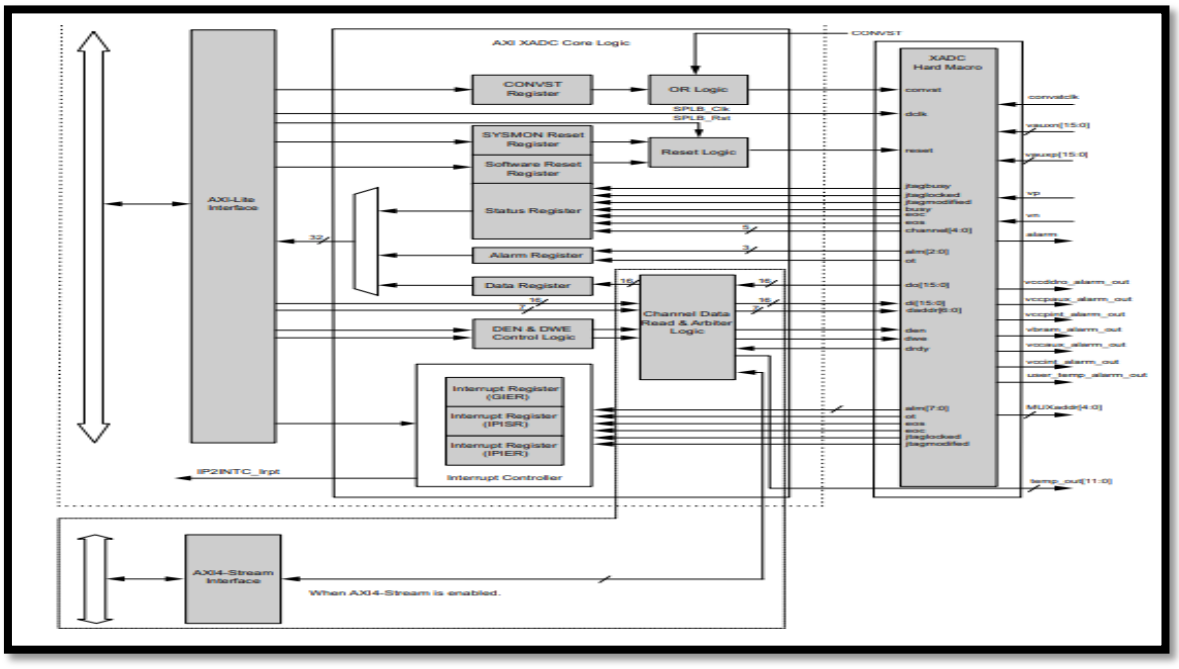


Figure 4.2. Detailed block diagram of XADC IP block

### 4.3. FIR Compiler

The Xilinx FIR compiler IP core procures an interface to implement extremely parameterizable, area-effective and superior performance FIR filters. Other than filtering operation, it features AXI interface. It supports high performance FIR, polyphase decimator, polyphase interpolator, half-band, half-band decimator, half-band interpolator, Hilbert transform and interpolated filter implementations. Also, it supports up to 256 sets of coefficients and input data in 49-bit precision (also for coefficients). By supporting for multiple parallel data channels, it allows the user to implement multichannel signal processing. User selectable output rounding and architecture enables more efficient and optimized results [7]. Figure 4.3 gives an idea about FIR compiler IP core.

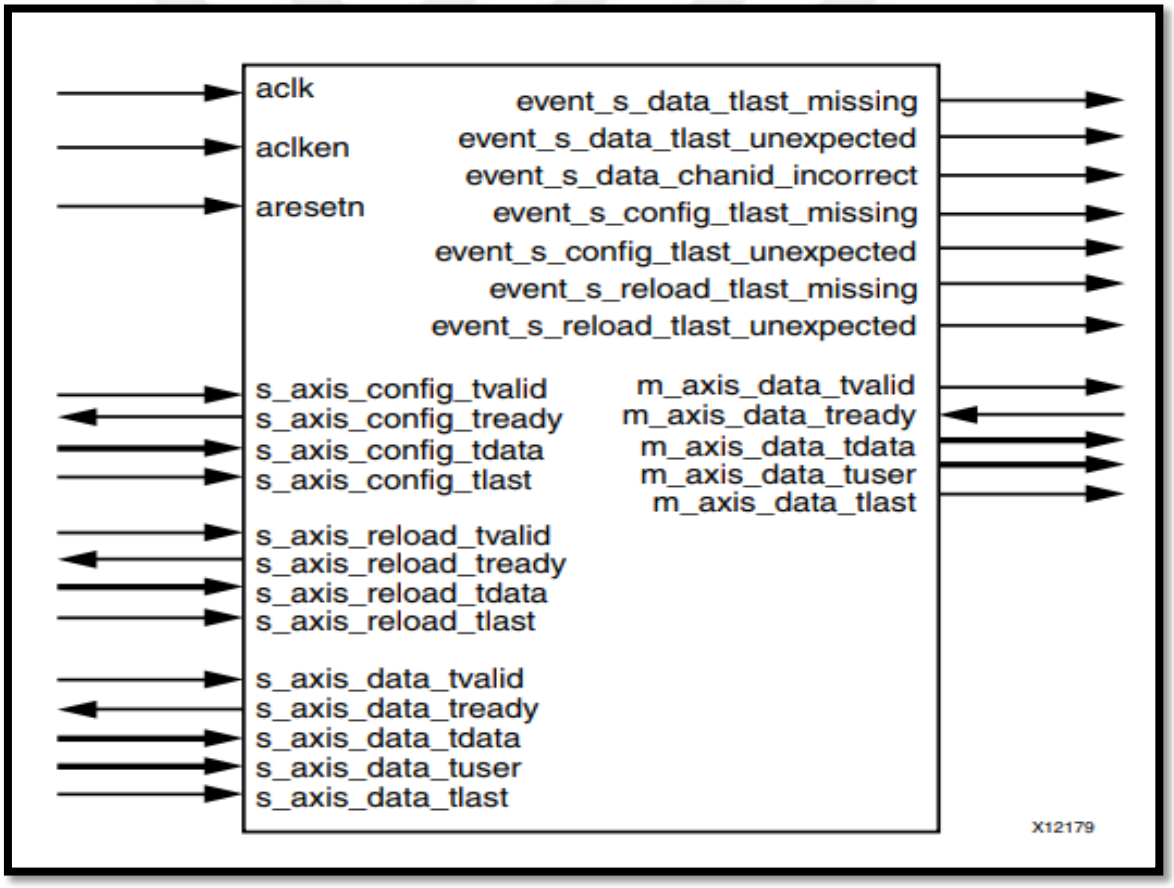


Figure 4.3. FIR compiler IP core

#### 4.3.1 The coefficient quantization

The FIR compiler IP block demonstrates three quantization settings as follows;

1. Integer coefficient
2. Quantize only
3. Maximize dynamic range

If the coefficients are designated in binary or hexadecimal, the integer coefficient option is allowed. Conversely, if the coefficients are specified in non-integer decimal values, then the quantize only and maximize dynamic range options are allowed [7].

Since the coefficients are generated in non-integer values using *FDA Tool* from one of the Simuling blocks, quantize only option is used for tests and simulation purposes. In quantize only option, the coefficients are quantized to the designated bit width. The coefficients are rounded to nearest quantum utilizing "*simple round towards zero algorithm*". The coefficients are divided into integer and fractional bits. If the designated bit width is fewer than the required bit width, coefficients are rounded [7].

Because the whole coefficients are among -1 and 1, only one integer bit is required (to retain the sign identification), the rest bits are designated to fractional bits [7]. Figure 4.4 shows an implementation of coefficient quantization regarding to simple round towards zero algorithm.

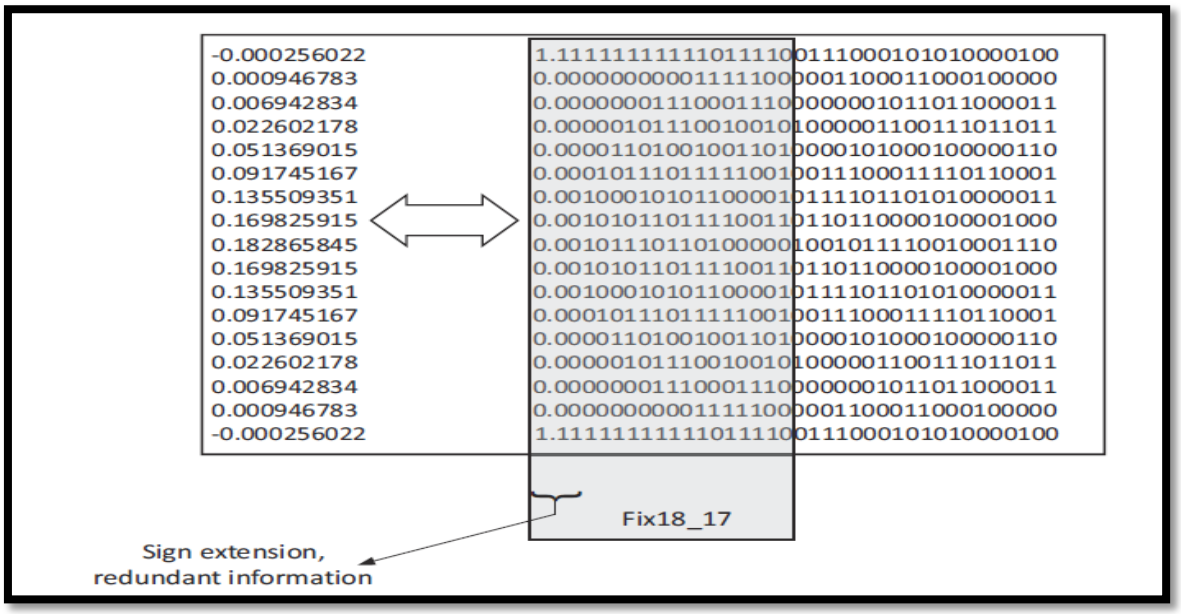


Figure 4.4. Coefficient quantization instance

### 4.3.2. Output rounding

The users can determine the output width of the filter in terms of their needs. In MAC implementations, the FIR compiler IP block allows the users to determine the output bit-width and round the outcome to the nearest number in this bit-width [7].

Non symmetric rounding up (or to positive) rounding mode is used for the testing and the simulation purposes. In this mode, 0.5 is added to the accumulator outcome and the most *least significant bit* (LSBs) are removed. This can be done with fewer or no resource cost in specific hardware. Figure 4.5 shows an representation for this mode.

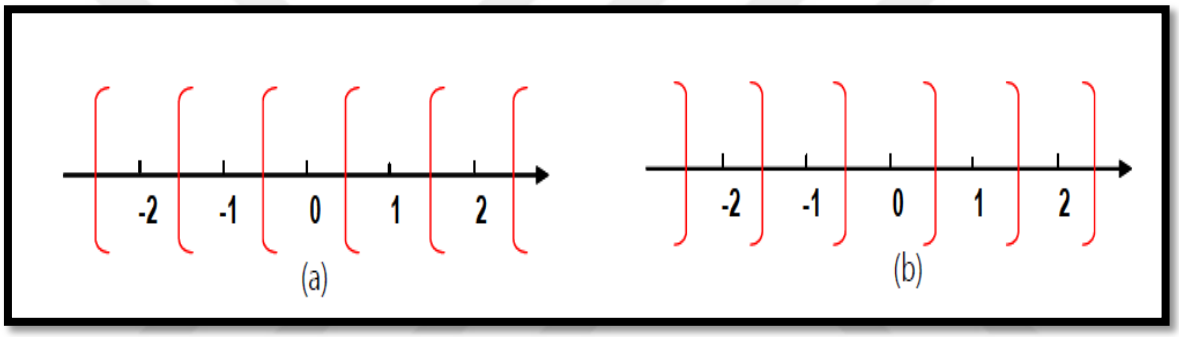


Figure 4.5. Output rounding

## 4.4. System Generator for DSP

System Generator is a DSP design support, which is from Xilinx and enable the user to utilize Simulink from MathWorks for a FPGA design. Xilinx FPGA or *register transfer level* (RTL) are not needed as utilizing System Generator tool. Users design their systems using Xilinx based blockset in Simulink [15]. Also, users by using System Generator can create IP blocks and then import their IP blocks in VHDL or Verilog into Vivado or using hardware co-simulation feature, they can simulate an implement the design both on Simulink and specific Xilinx FPGA. Figure 4.6 shows some blocksets associated with Xilinx block sets.

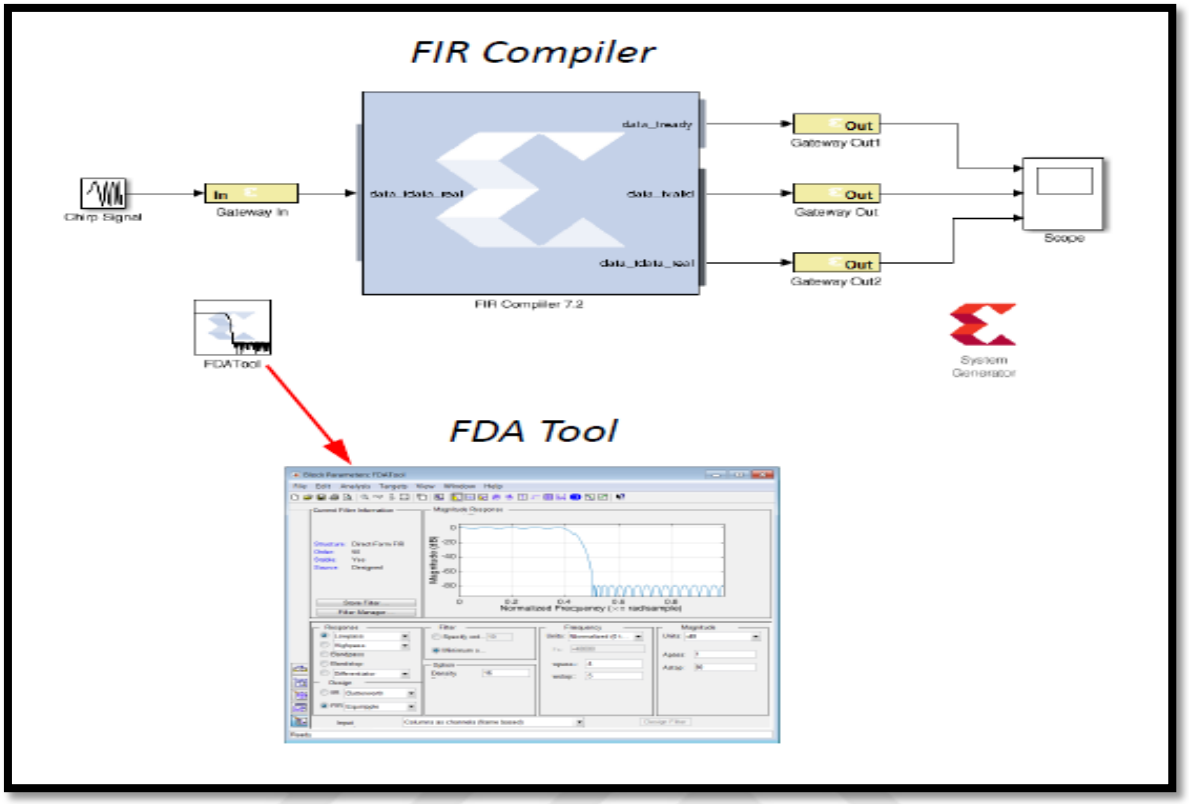


Figure 4.6. Some blocks sets regarding to Xilinx block sets

Design blocks of System Generator differ from conventional Simulink blocks. They can be found in the library of Matlab/Simulink. However, System Generator blocks can not interface directly with the Simulink blocks. In order to overcome this issue, Xilinx provided "*Gateway In*" and "*Gateway Out*" blocks must be used for these circumstances.

If the users want to implemenet targets named *IP catalog* and hardware co-simulation, they need to facilitate a token named *System Generator token*. In this token, the users can adjust simulation time, their boards, type of implementations etc.

# 5. SIMULATION AND VERIFICATION

In this thesis, for the testing and simulation studies, Xilinx Artix-7 (XC7A100-1CSG324C) series FPGA, Vivado 2017.4 version design tool and System Generator have been utilized in order to implement simulation and test. Other than using the FPGA itself, the Microblaze soft-core processor, which are composed of the FPGA fabric, and IP blocks are utilized for the design itself. The Vivado design tool is very powerful and versatile tool for users who have zero knowledge about VHDL. Thanks to the Vivado, the IP blocks used for designs are converted to the VHDL language via *HDL wrapper* operation. This operation turns IP blocks into specific VHDL code and a user can observe this VHDL code to analyze the specific design.

In Figure 5.1, the overall system can be observed. In this system, Nexys 4 DDR board, which contains Xilinx Artix-7 (XC7A100-1CSG324C) is utilized for this thesis. Also, there is a signal generator device which is called Analog Discovery 2. This device is developed by Digilent and utilizes a Xilinx FPGA and it can create every one of the signal that a user wants. On the left hand side, there is a DAC circuit that accepts 12 bit parallel signals coming from Nexys 4 DDR board to observe the filtered analog signal.

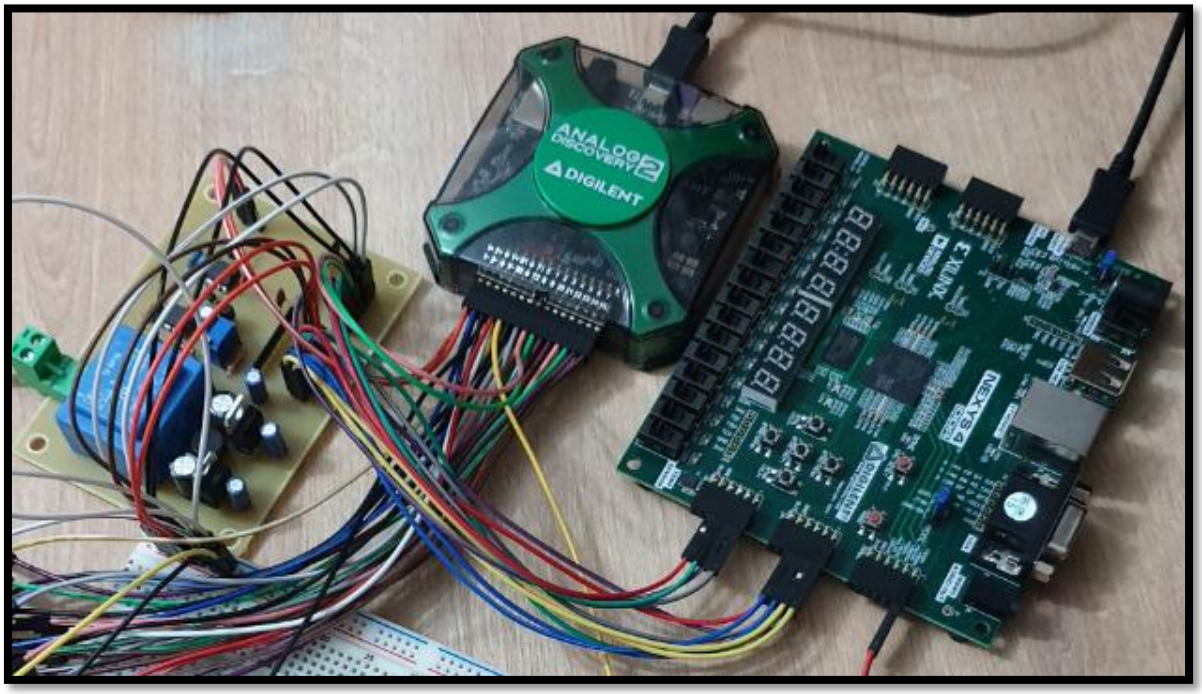


Figure 5.1. Overall system overlook

In Figure 5.2, the noised signal that is used for filtering is displayed. The noised signal has been generated through Analog Discovery 2 device. Base voltage of 0.5, offset voltage of 0.5 and frequency of 10 kHz are set to observe the digital filtering for a high frequency noised signal.

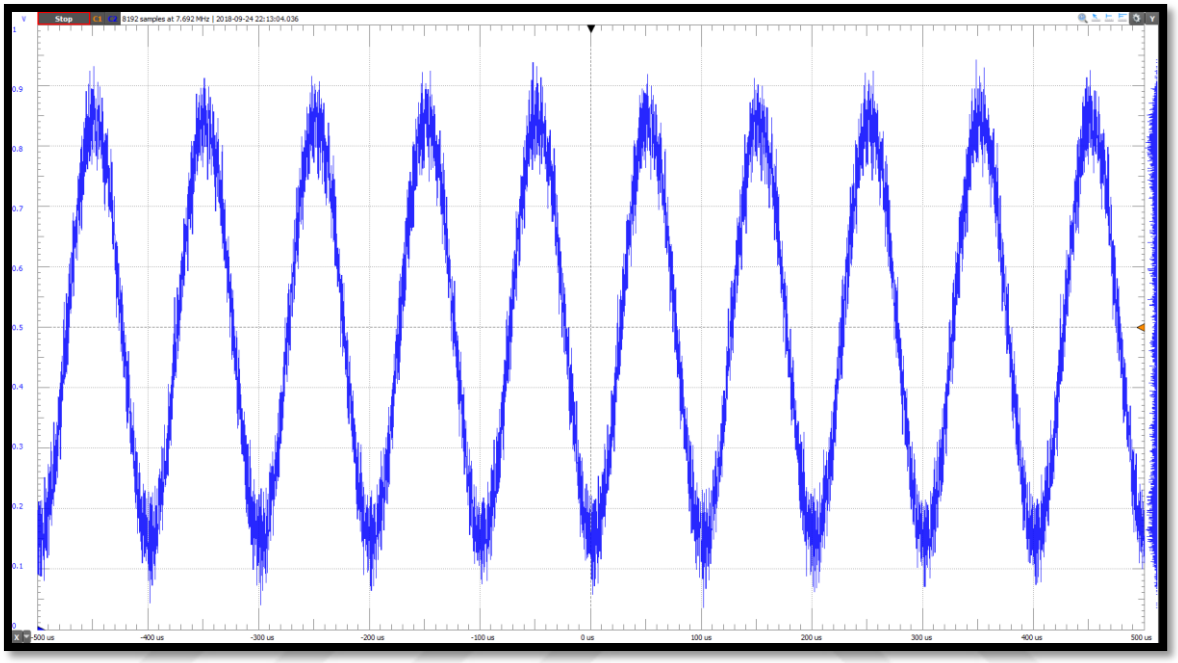

Figure 5.2. Noised signal

The FIR compiler developed by Xilinx is used in order to observe the digital filtering. The coefficients implemented on the FIR compiler are generated via FDA tool from Simulink block. The sampling frequency, cut-off frequency and stopband frequency are set to 20 kHz, 4 kHz and 5 kHz respectively. Passband magnitude and stopband magnitudes are set to 1 dB and 80 dB respectively.

Elaborated design perspective gives an complete vision of the specific design. In Figure 5.3, the elaborated design for the two channels implementation used in thesis and, in Figure 5.4, single channel elaborated design can be seen. As it can be seen from the Figure 5.3 and 5.4 that all of the inputs and outputs can be observed.

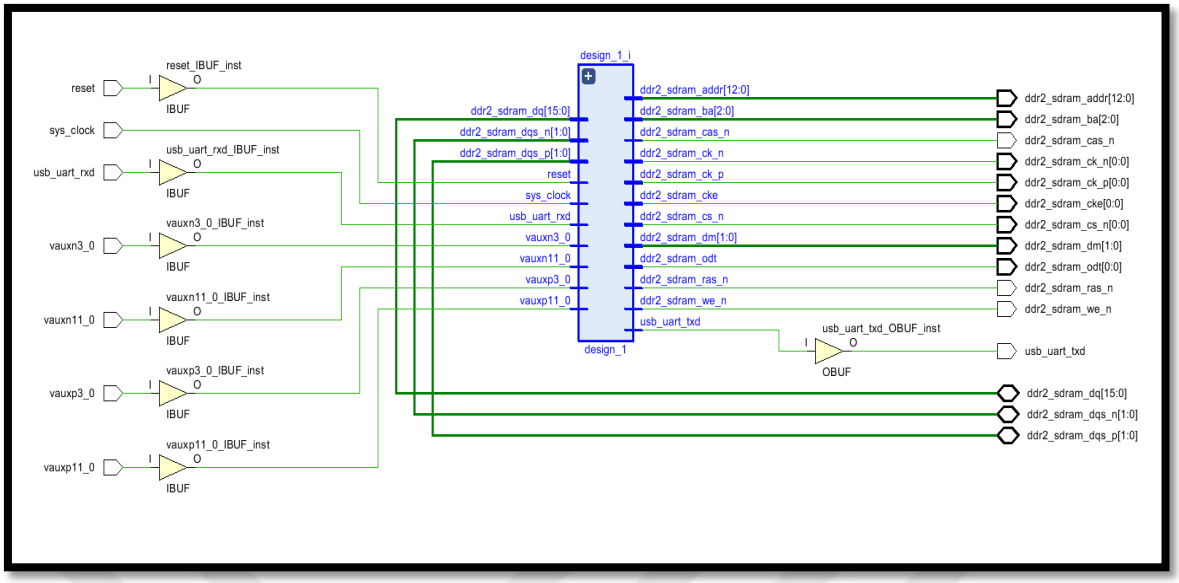


Figure 5.3. Elaborated design for two channels implementation

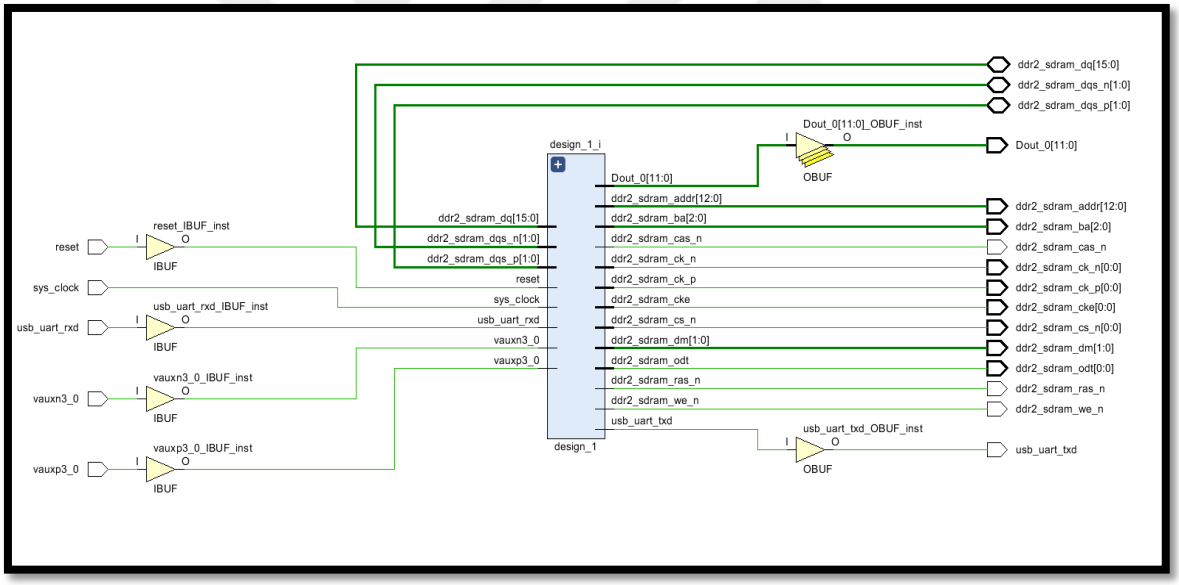


Figure 5.4. Elaborated design for single channel implementation

In Figure 5.5, the whole design implemented for two channels application can be seen. All of the components including the XADC wizard, the FIR compiler, ILA, clock componentsand all of the AXI interface components can be observed. Likewise, in Figure 5.6, for single channel implementation can be observed. In Figure 5.7 and 5.8, the closer look of the two channels implementation and single channel implementation can be seen.

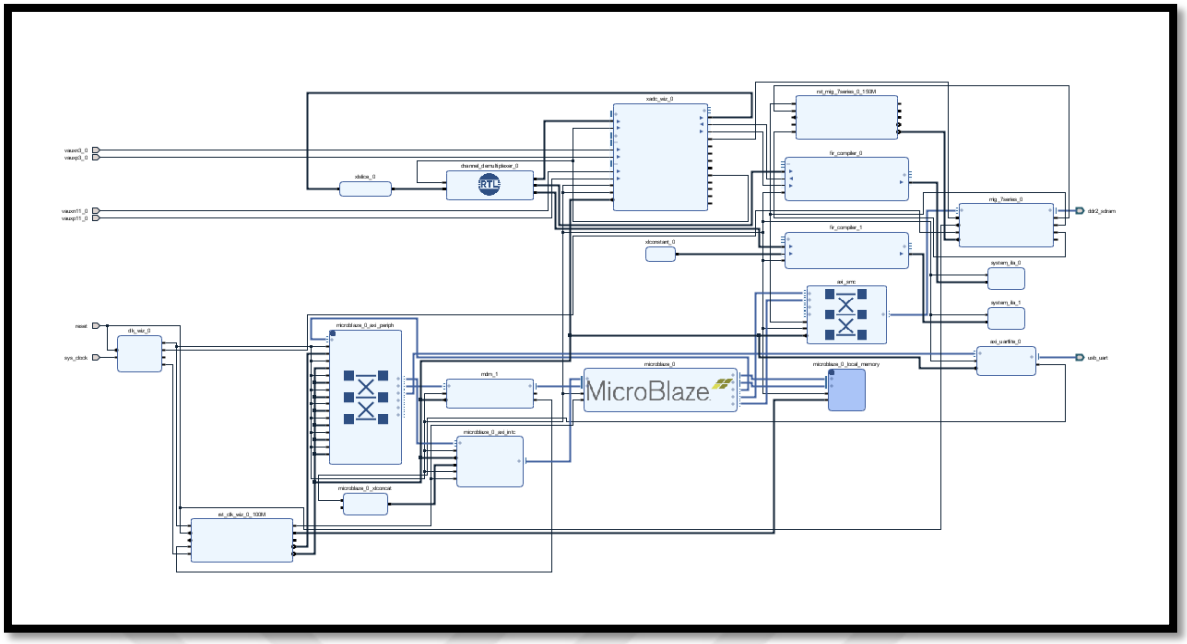


Figure 5.5. The whole design of the two channel implementation

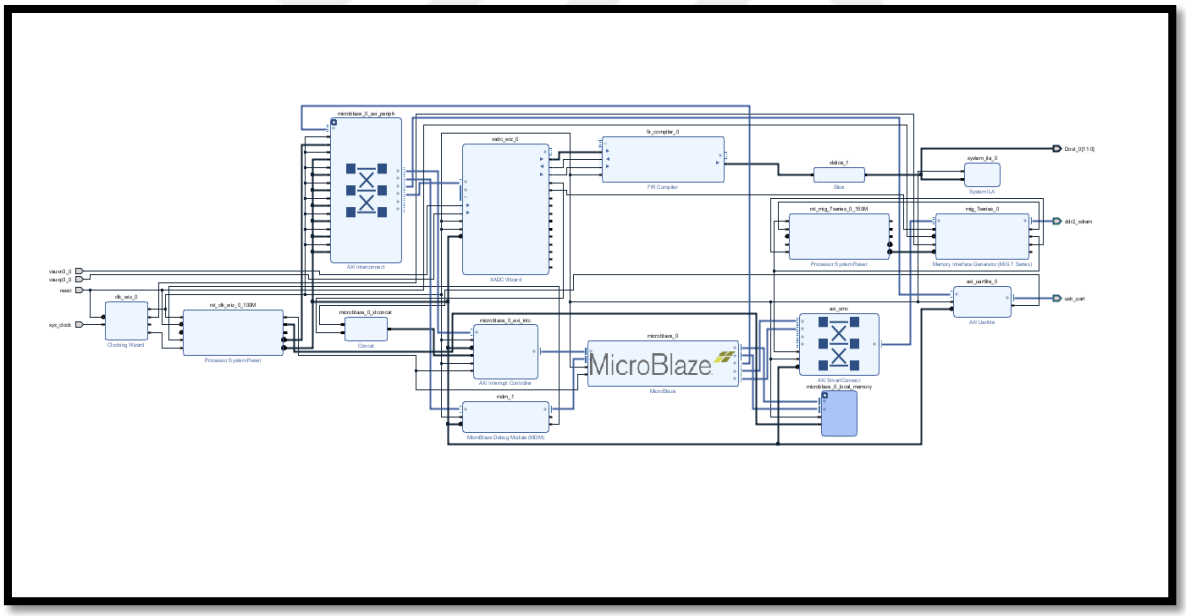


Figure 5.6. The whole design of the single channel

In closer look of the designs, components that are being used for these designs can be displayed. The data buses are connected together via AXI smart-connect and AXI interconnect. These two AXI IP blocks connect the IP blocks used in the design in conjuction with the Microblaze soft-core processor, clocks, DDR2 RAM etc.

The channel multiplexer is utilized to seperate the bus used for two interleaved signals. The XADC wizard is set to run as *dynamic reconfiguration port* (DRP) mode. With this

mode, a user can reach the signals via using the address of the related signals. Xslice IP blocks are used to extract the 12-bit from the most significant bit of 16-bit. System ILAs are being utilized as probes to observe the filtered signals. AXI UART-lite is being utilized to run serial communication between the FPGA and the PC.

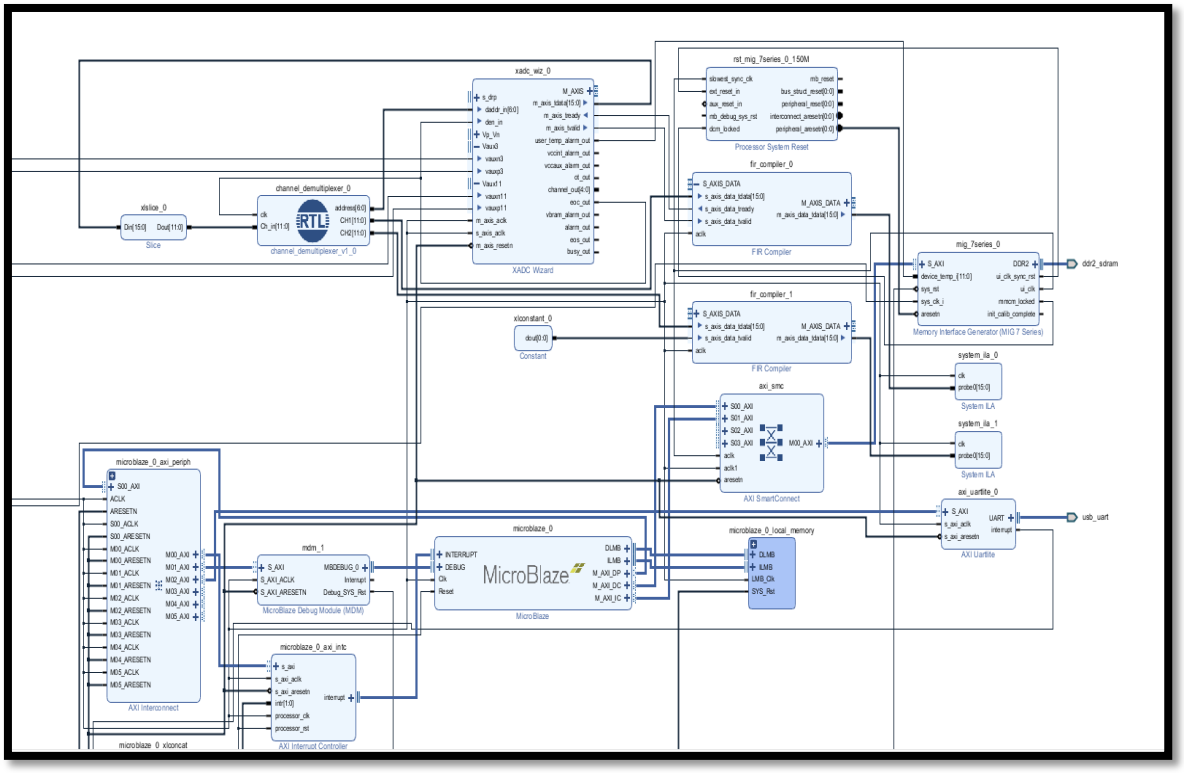


Figure 5.7. The closer look for two channels implementation

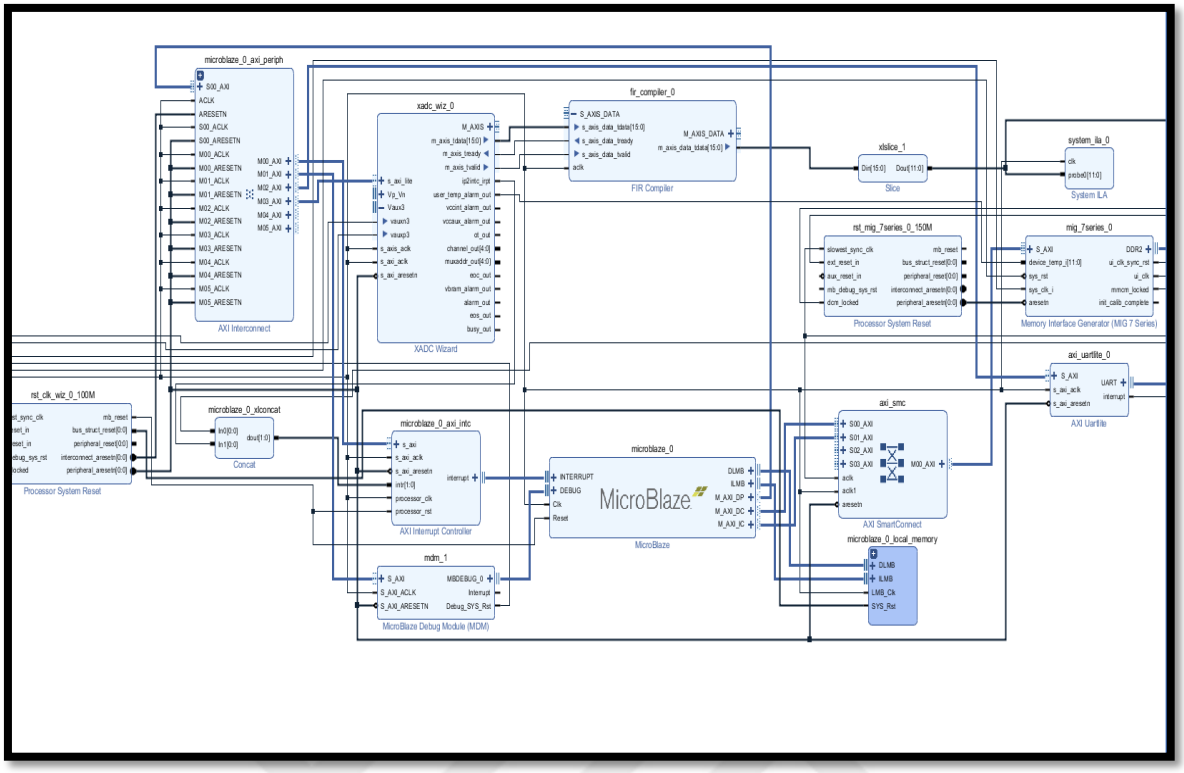


Figure 5.8. The closer look for single channel implementation

As it mentioned before, System Generator can be utilized for simulation purposes using Xilinx block sets. For this purpose, the noised signal is generated using *band-limited white noise* to observe the digital filtering. In Figure 5.9, the whole design can be seen for simulation purposes. The token named *System Generator* is utilized to set the timing, board, producing IP block in order to implement in Vivado design tool, hardware co-simulation, generating VHDL netlists etc. The *FIR compiler v7.2* is the counterpart of the Vivado design tool. The blocks called *Gateway In* and *Gateway Out* are blocks that are used to interact with the Simulink blocks. Normally, the Xilinx block sets and Simulink blocks are not connected without the Gateway In and Gateway Out. These two blocks can be thought as inputs and outputs of the design. FFT block and registers are used for their purposes. Scopes and spectrum analyzers are being used for observing the signals.

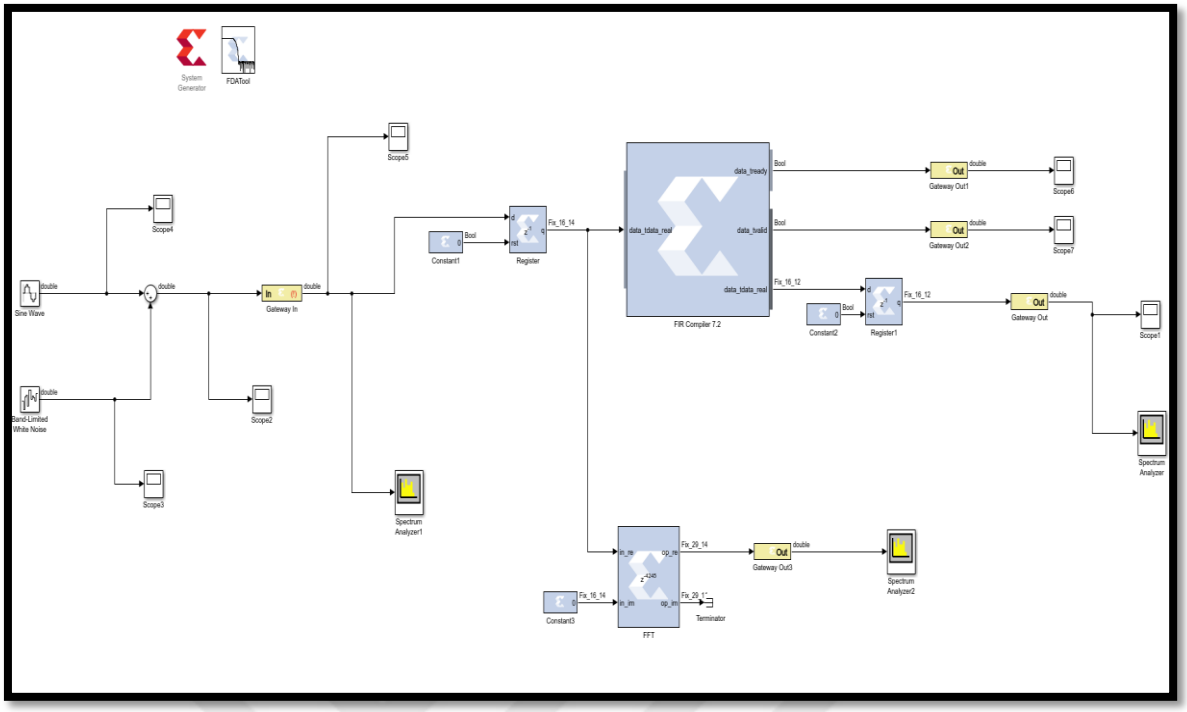


Figure 5.9. System Generator design

In Figure 5.10, the generated band-limited white noise with sine wave can be seen. The sine wave is set to 1V and 150 Hz to observe the low frequency filtering.

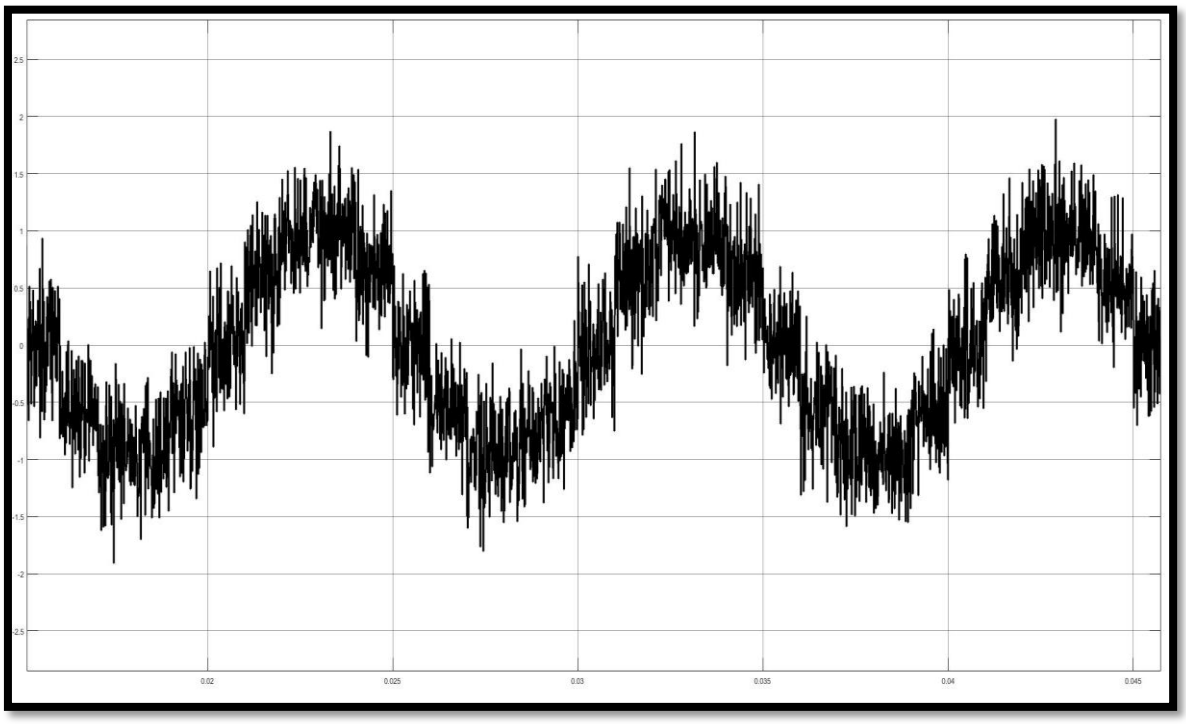


Figure 5.10. Noised sine wave for System Generator

# 6. RESULTS

All of the filtering results, simulation outcomes, resource outcomes are explained in detailed. 3-taps, 4-taps, 5-taps and 8-taps low-pass FIR filters are used for single channel filtering. For these taps, 4, 5, 6 and 9 coefficients are used via FDA tool. Figure 6.1 illustrates 3-taps, Figure 6.2 illustrates 4-taps, Figure 6.3 illustrates 5-taps and Figure 6.4 illustarates 8-taps filtered signals using noisy signal showed in Figure 5.1.


Figure 6.1. 3-taps filtered signal displayed in ILA

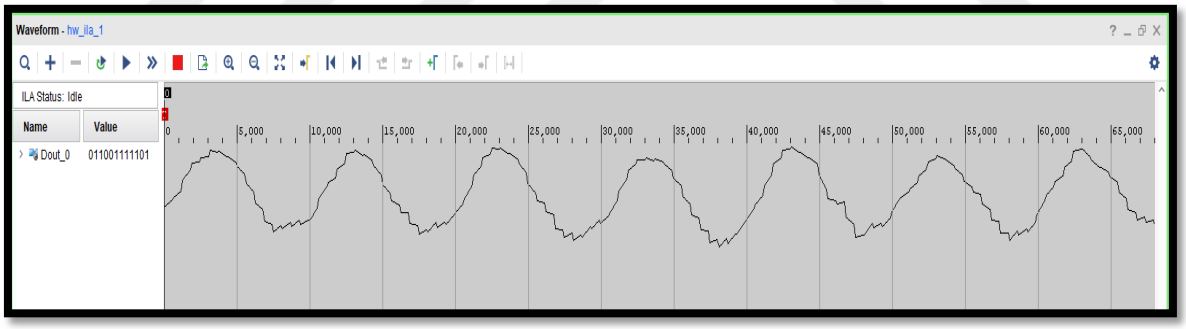


Figure 6.2. 4-taps filtered signal displayed in ILA

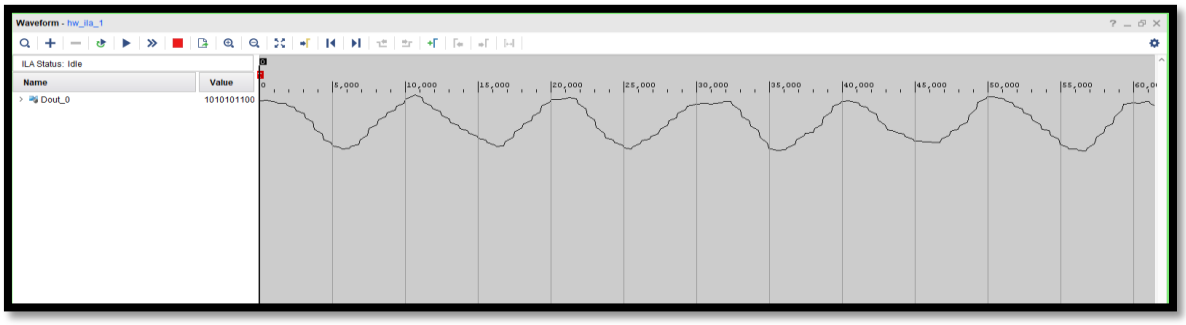


Figure 6.3. 5-taps filtered signal displayed in ILA

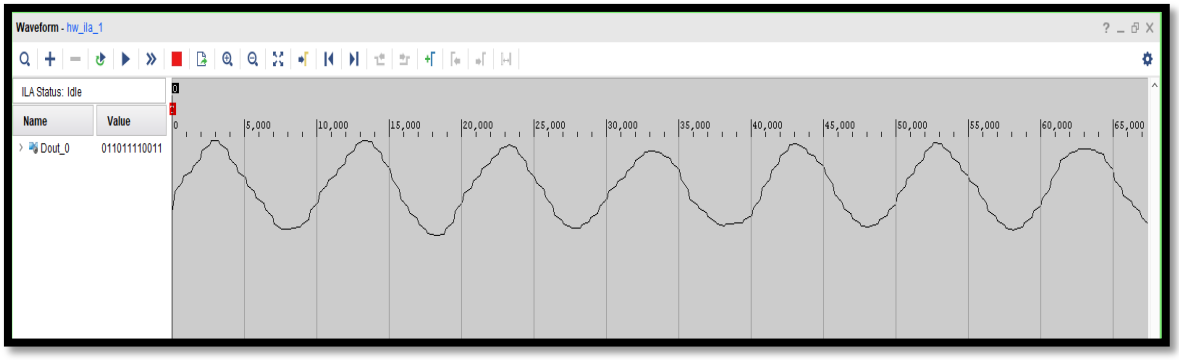


Figure 6.4. 8-taps filtered signal displayed in ILA

In addition to the hardware ILA to observe the filtered signal, a 12 bit DAC is used to compare and observe the filtered signals in adjacent with the results obtained with ILA. Figure 6.5 displays 3-taps filtered, Figure 6.6 displays 4-taps filtered signal, Figure 6.7 displays 5-taps filtered signal and finally, Figure 6.8 displays 8-taps filtered signal obtained from the DAC.

In comparison with DAC, the resutls obtained from the hardware ILA are consistent with results obtained from DAC. This also employs that hardware ILA can be trustful when there is no DAC that can be used for these situations.

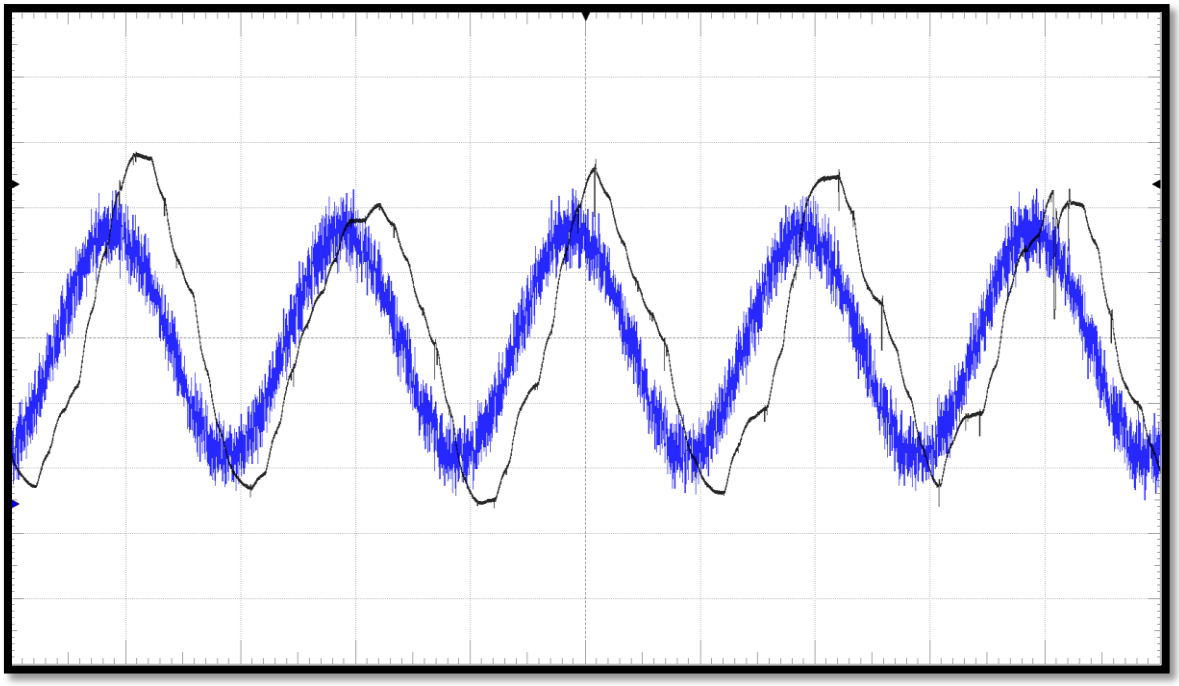

Figure 6.5. 3-taps filtered signal output of DAC

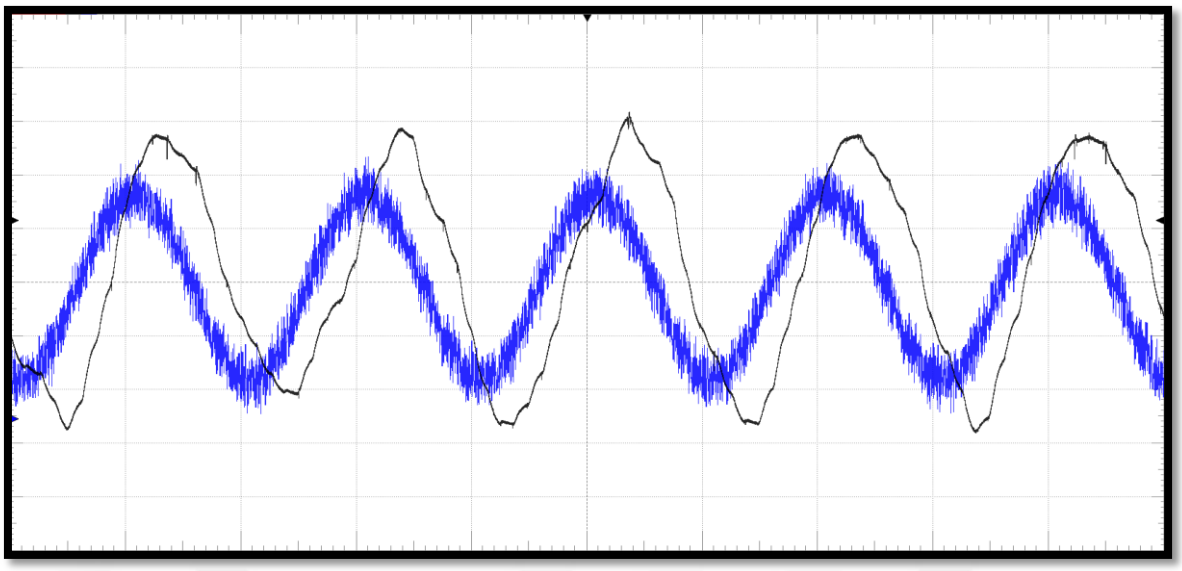

Figure 6.6. 4-taps filtered signal output of DAC

As it can be seen from the Figure 6.5, Figure 6.6, Figure 6.7 and Figure 6.8 that even if the test is implemented in low taps, the results is very promising as far as a low-cost FPGA is concerned. Thanks to the architecture and optimization Xilinx, there is a greeat amount of resource for further taps.

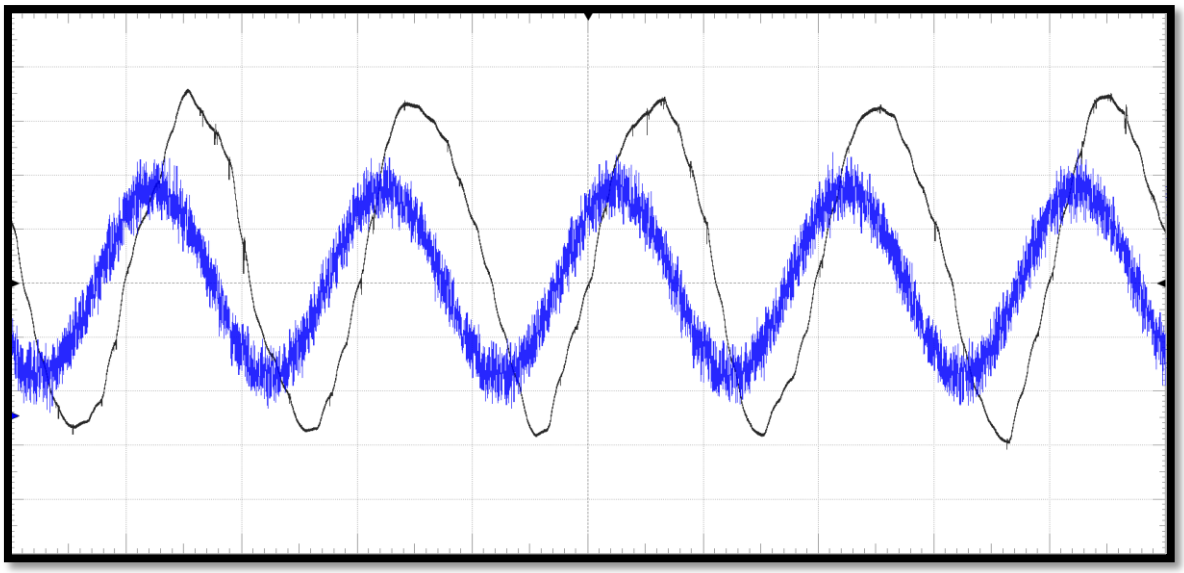

Figure 6.7. 5-taps filtered signal output of DAC

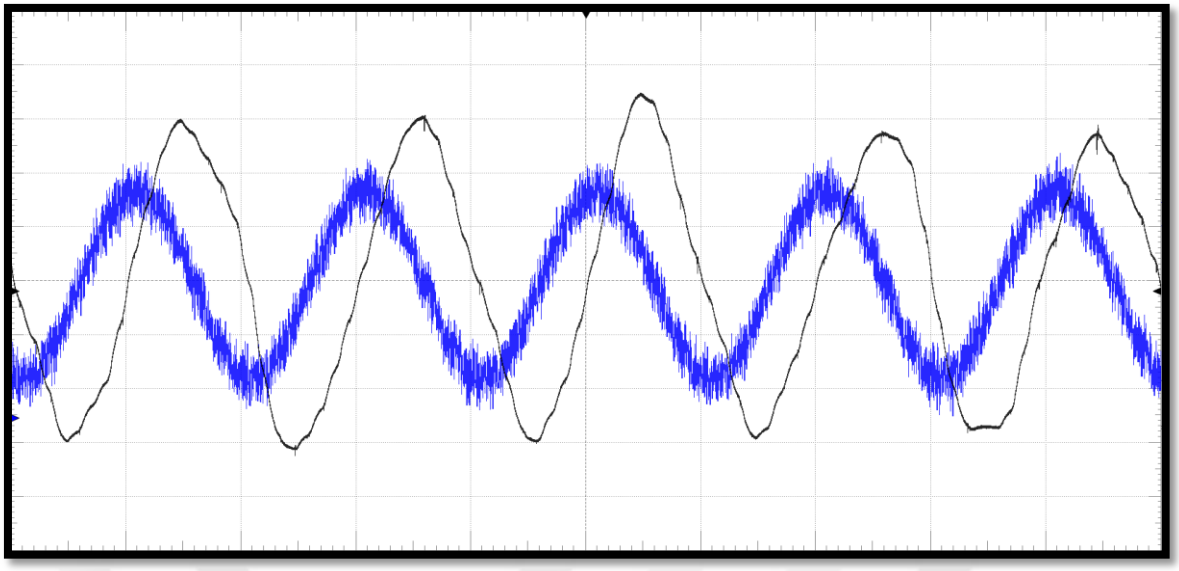

Figure 6.8. 8-taps filtered signal output of DAC

As it can be seen from the Figure 6.1, 6.2 and 6.3 that digital filtering process is done and as the greater the taps, the smoother the signal is. Input and output widths are set to 12-bit. 12-bit output are sent to the ILA to get the results displayed in Figure 6.1, 6.2 and 6.3. For single channel DSP slice resource usage, for example, for 8-taps we need 5 DSP slices. Thanks to the systolic MAC architecture, resource usage for DSP slice is decreased. That being said, maximum of 478 taps FIR filter can be used for single channel. Since the used board has maximum of 240 DSP slices.

Two channels implementation is designed and applied in this thesis. Figure 6.9 and Figure 6.10 can display the parallel two channel implementation. In this implementation, latency is not changed and DSP slice usage is two times the single channel. 8-taps, two channel implementation results in no latency.

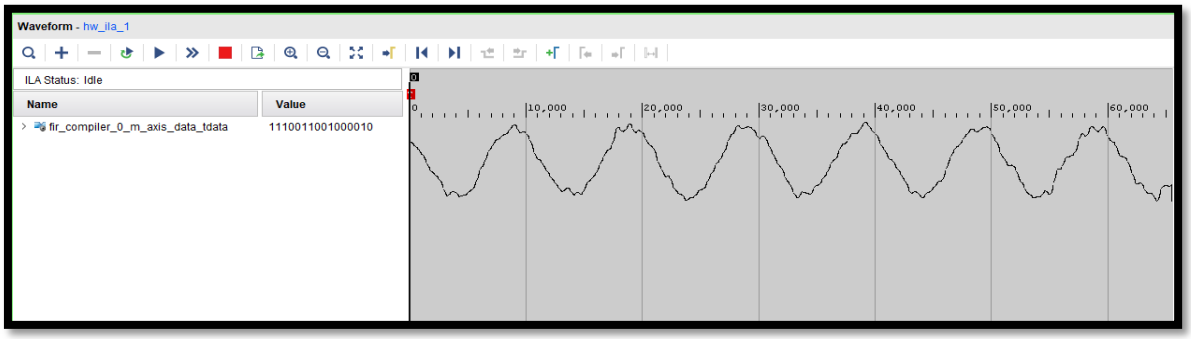


Figure 6.9. Filtered signal for third channel

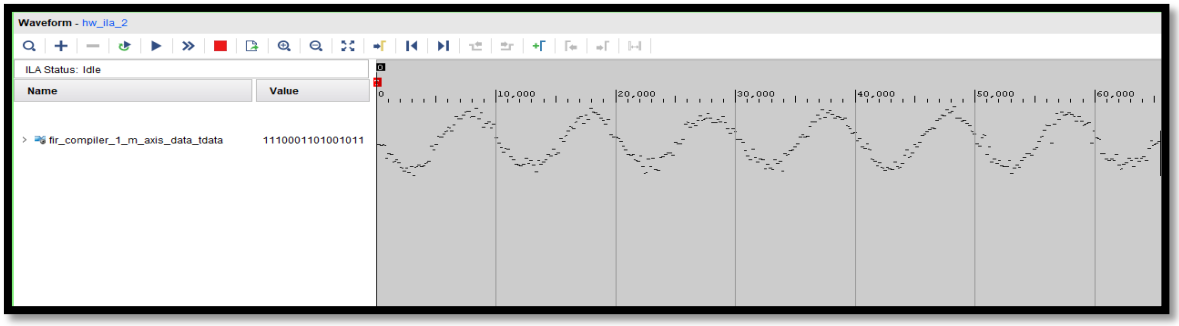


Figure 6.10. Filtered signal for eleventh channel

In resource purpose, 8 parallel channels configuration to set to see the latency and resource usage. Table 6.1-6.7 outlines the latency and LUTs and flip-flops other than DSP slice. As it can be seen from the tables that increase of the parallel channel (in this case maximum of 8) does not change dramatically in respect to flip-flops and LUTs. As mentioned above maximum of 478-taps can be used for single channel. it can be seen from the results that up to 64-taps, there are more resources if we want to increase the number of channel.

Table 6.1. DSP slice usage (for single channel)

| Taps | DSP SLICE | | |
|---|---|---|---|
| | *Utilization* | *Available* | *Utilization %* |
| 3 | 2 | 240 | 0.83 |
| 4 | 3 | 240 | 1.25 |
| 8 | 5 | 240 | 2.08 |
| 16 | 9 | 240 | 3.75 |
| 32 | 17 | 240 | 7.08 |
| 64 | 33 | 240 | 13.75 |
| 128 | 65 | 240 | 27.08 |
| 256 | 129 | 240 | 53.75 |

In Table 6.1, as it can be seen that utization DSP slice usage does not increase dramatically. As for Table 6.2, until 64 taps, it is implemented 8 parallel channel and 8 times more than DSP slice usage seen on Table 6.1. It can be said that DSP slice usage

again does not increasing dramatically. It also shows that how the DSP slice usage can be optimized by Xilinx itself.

Table 6.2. DSP slice usage (for 8 channels maximum)

| Taps | DSP SLICE | | |
|---|---|---|---|
| | *Utilization* | *Available* | *Utilization %* |
| 3 | 16 | 240 | 6.67 |
| 4 | 24 | 240 | 10 |
| 8 | 40 | 240 | 16.67 |
| 16 | 72 | 240 | 30 |
| 32 | 136 | 240 | 56.67 |
| 64(max 7-Channel) | 231 | 240 | 96.25 |
| 128(max 3-Channel) | 195 | 240 | 81.25 |
| 256(max 1-Channel) | 129 | 240 | 53.75 |

Table 6.3. LUT usage (for single channel)

| Taps | LUT | | |
|---|---|---|---|
| | *Utilization* | *Available* | *Utilization %* |
| 3 | 13812 | 63400 | 21.79 |
| 4 | 13844 | 63400 | 21.84 |
| 8 | 13908 | 63400 | 21.94 |
| 16 | 14036 | 63400 | 22.14 |
| 32 | 14292 | 63400 | 22.54 |
| 64 | 14805 | 63400 | 23.35 |
| 128 | 16326 | 63400 | 25.75 |
| 256 | 21080 | 63400 | 33.25 |

The same situation continues here. For single channel usage, LUT resource usage does not change greatly. This can be said that for the multiple channel appllication. Again, it is the same as the before and until 64 taps it is implemented with 8 channels. Thanks to the

Xilinx's good optimization features, the LUT usage for multiple channel implementation, it does not change dramatically.

Table 6.4. LUT usage (for 8 channels maximum)

| Taps | DSP SLICE | | |
|---|---|---|---|
| | *Utilization* | *Available* | *Utilization %* |
| 3 | 14036 | 63400 | 22.14 |
| 4 | 14292 | 63400 | 22.54 |
| 8 | 14804 | 63400 | 23.35 |
| 16 | 15828 | 63400 | 24.97 |
| 32 | 17876 | 63400 | 28.20 |
| 64(max 7-Channel) | 22629 | 63400 | 35.69 |
| 128(max 3-Channel) | 21414 | 63400 | 33.78 |
| 256(max 1-Channel) | 21080 | 63400 | 33.25 |

As it can be seen from the Table 6.5 that flip-flop usage for single channel does not change dramatically thanks to the optimization Xilinx.

Table 6.5. Flip-flop usage (for single channel)

| Taps | FLIP-FLOP | | |
|---|---|---|---|
| | *Utilization* | *Available* | *Utilization %* |
| 3 | 14769 | 126800 | 11.65 |
| 4 | 14802 | 126800 | 11.67 |
| 8 | 14900 | 126800 | 11.75 |
| 16 | 15096 | 126800 | 11.91 |
| 32 | 15488 | 126800 | 12.21 |
| 64 | 16272 | 126800 | 12.83 |
| 128 | 17840 | 126800 | 14.07 |
| 256 | 21044 | 126800 | 16.60 |

As it mentioned above, the same situation goes on here. In comparison with one channel implementation, flip-flop usage for multiple channel implementation, it increases slightly thanks to the Xilinx's optimization.

Table 6.6. Flip-flop usage (for 8 channels maximum)

| Taps | FLIP-FLOP | | |
|---|---|---|---|
| | *Utilization* | *Available* | *Utilization %* |
| 3 | 15665 | 126800 | 12.35 |
| 4 | 15922 | 126800 | 12.56 |
| 8 | 16692 | 126800 | 13.16 |
| 16 | 18232 | 126800 | 14.38 |
| 32 | 21312 | 126800 | 16.81 |
| 64(max 7-Channel) | 26776 | 126800 | 21.12 |
| 128(max 3-Channel) | 24112 | 126800 | 19.02 |
| 256(max 1-Channel) | 21044 | 126800 | 16.60 |

In the Table 6.7, it can be seen that latency between the single channel implementation and multiple channel implementation remains the same. The reason why the multiple channel implementation is fulfilled is that the latency remains the same even if 8 channels is implemented. This feature can be useful when there is more than one signal and in addition, it does it fast compared to DSP processors.

Ability of multichannel implementations may be required in power electronics circuits, control circuits or communication. Based on importance of time and parallel implementation, as it can be viewed from the Table 6.1-6.7 that FPGAs can handle this situation successfully and give promising results even if low-cost Artix-7 model FPGA is utilized.

Table 6.7. Latency (single channel vs 8 channels)

| Taps | LATENCY (CYCLES) | | |
|---|---|---|---|
| | *Single Channel* | *Multichannel* | *Difference* |
| 3 | 9 | 9 | - |
| 4 | 10 | 10 | - |
| 8 | 12 | 12 | - |
| 16 | 16 | 16 | - |
| 32 | 24 | 24 | - |
| 64(max 7-Channel) | 40 | 40 | - |
| 128(max 3-Channel) | 72 | 72 | - |
| 256(max 1-Channel) | 140 | - | - |

As far as the power usage is concerned Vivado design tool comes to the aid. Figure 6.11 and Figure 6.12 shows the power usage for single channel and two channels implementations. As it can be seen from the figures that the power usage difference between the single and two channel implementation has slight difference. This shows how this system efficient and suitable for this application.

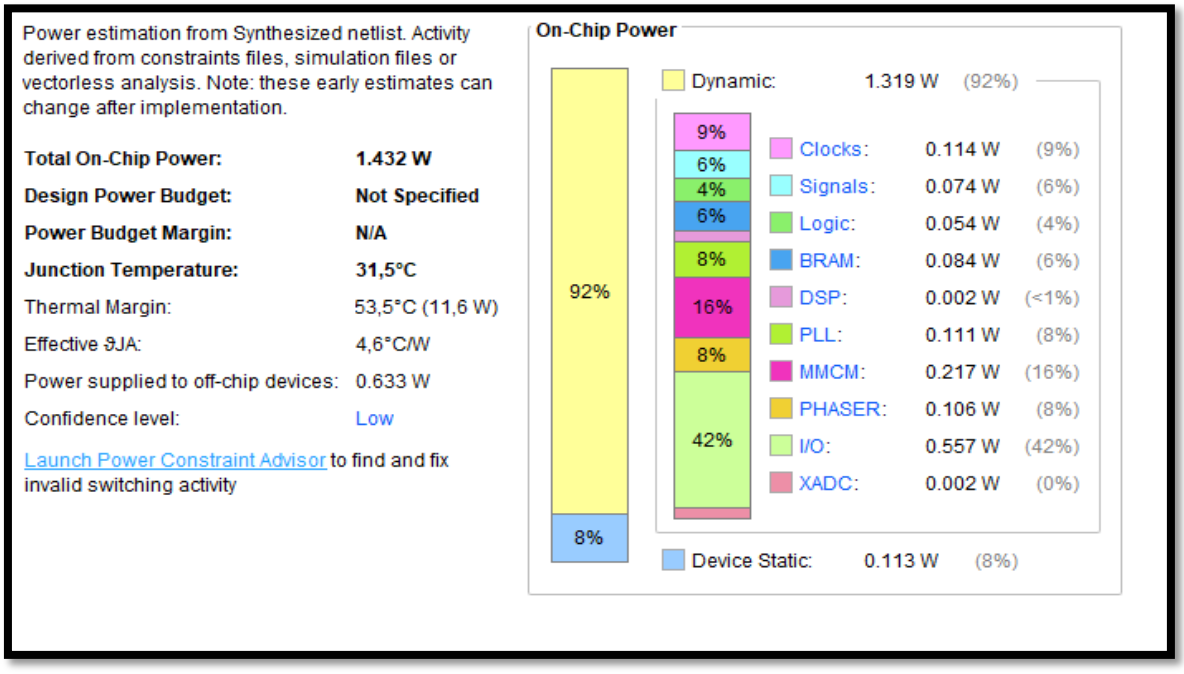


Figure 6.11. Single channel power usage

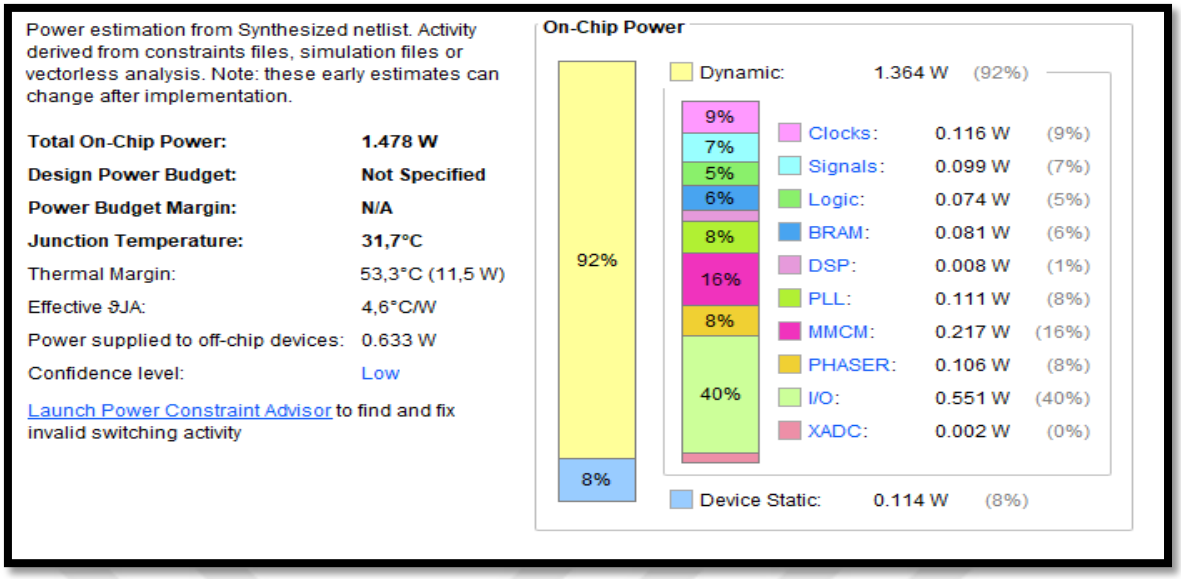


Figure 6.12. Two channels power usage

In System Generator side, Figure 6.9 illustrates the filtered signal. For this simulation, 32-taps lowpass FIR filter is used. Besides, it has 500 Hz sampling frequency, 150 Hz cut-off frequency and 200 Hz stopband frequency.

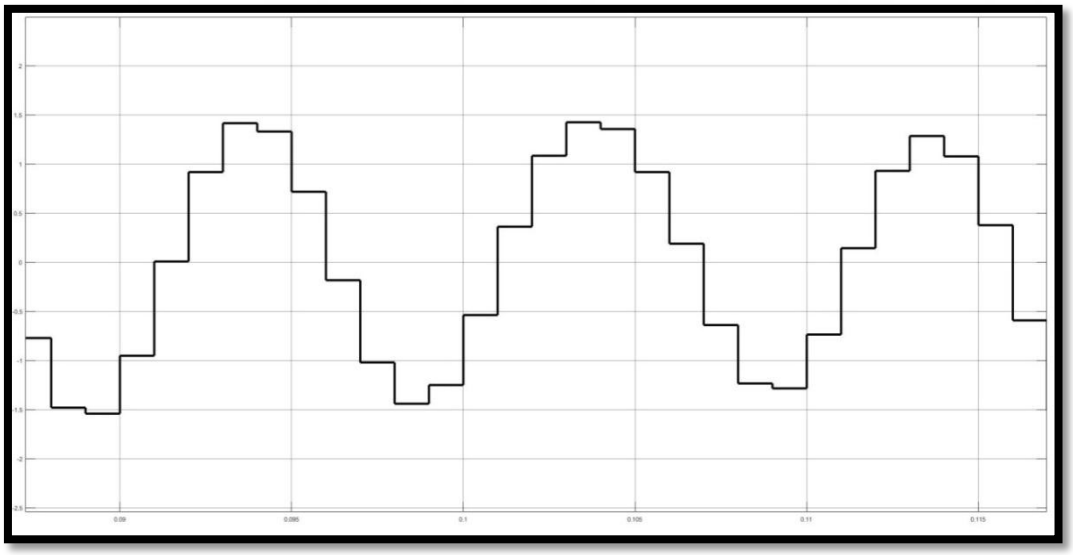

Figure 6.13. Filtered signal simulated in System Generator

Since the Simulink blocks are *double* formatted (i.e. 64-bit) and since the output signal fixed point and is 12-bit, the output signal is displayed in Figure 6.9. Drastic decrease in output width results in output signal like in Figure 6.9.

Spectrum of the noised signal and filtered signal illustrate in Figure 6.10 and Figure 6.11 respectively. As it can be seen from the spectrum of the filtered signal, the filter does not pass freqency above 150 Hz.

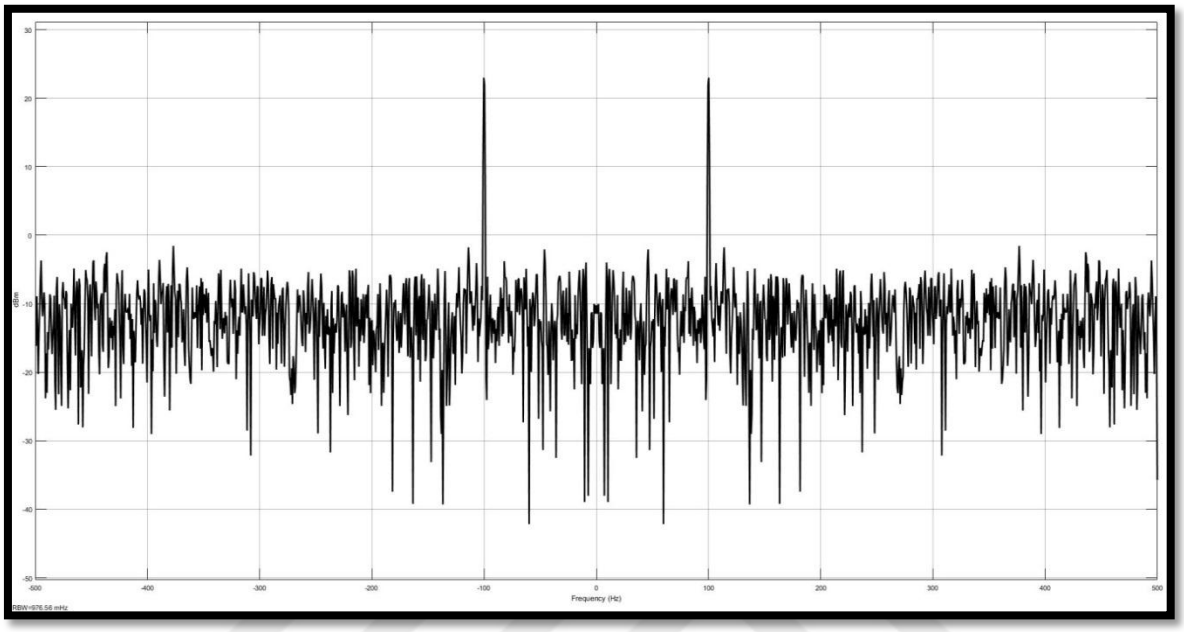


Figure 6.14. Spectrum of the noised signal

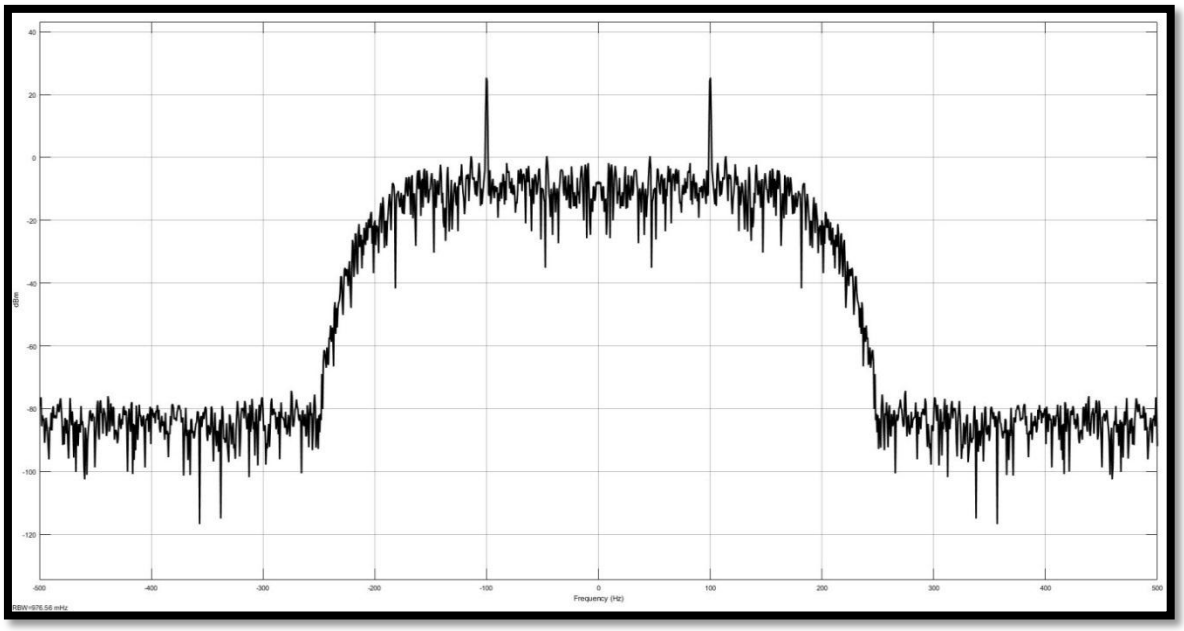


Figure 6.15. Spectrum of the filtered signal

# 7. COMPARISON OF PERFORMANCES OF XILINX FPGAs

Xilinx's 7-Series FPGA family consists of four elements regarding cost and performance. This family ranges from short cost, little form factor, cost sensitive, superior performance practices to superior connectivity bandwidth, logic capability and most importantly signal processing capacity for the various applications. These family members are named as;

- Spartan-7 Series: Short cost, least power, superior I/O performance and small form-factor.
- Artix-7 Series: For little power practices related to serial transceivers and superior DSP and logic outcome. This makes this series lowest material cost for superior outcome and cost sensitive practices.
- Kintex-7 Series: Topnotch price productivity with a X2 improvement by comparison with the prior generations.
- Virtex-7 Series: High-end system performance and capacity compared to other series of family.

Figure 7.1 can display the summary of the phrases mentioned above. The higher the performance they give, the lower the form-factor get smaller.

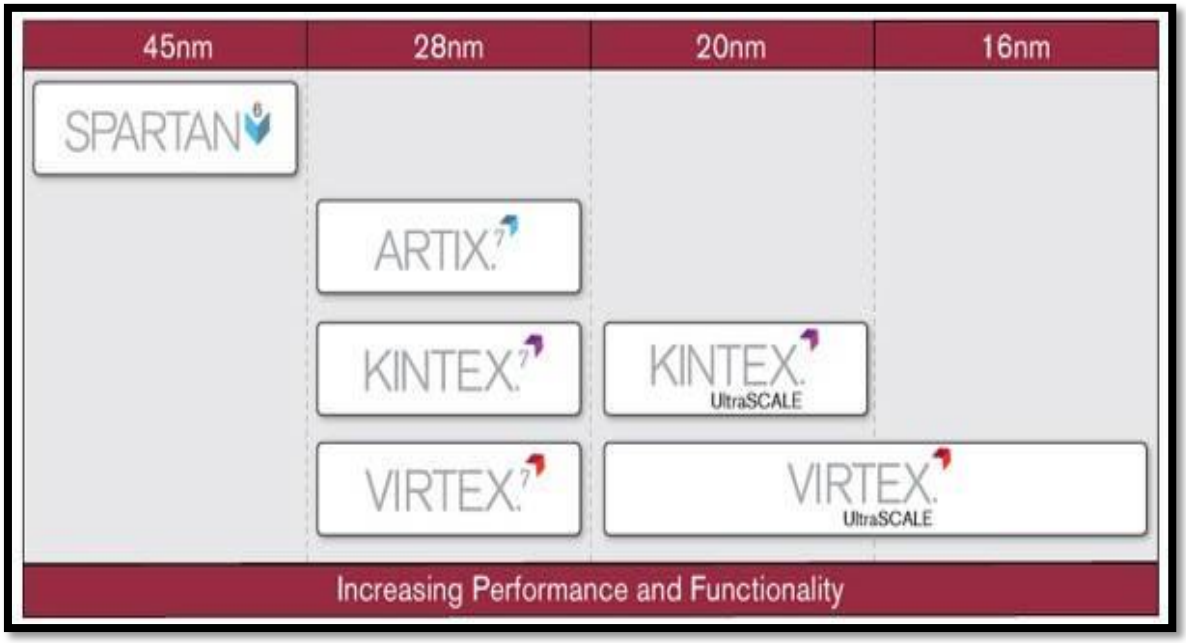


Figure 7.1. Xilinx 7-Series FPGA family

As for the DSP capability, Xilinx 7-Series FPGAs can really handle what users want from performance wise. DSP slices with 25 x 18 multiplier, 48-bit accumulator, and pre-adder for superior-performance filtering, includes optimised symmetric coefficient filtering.

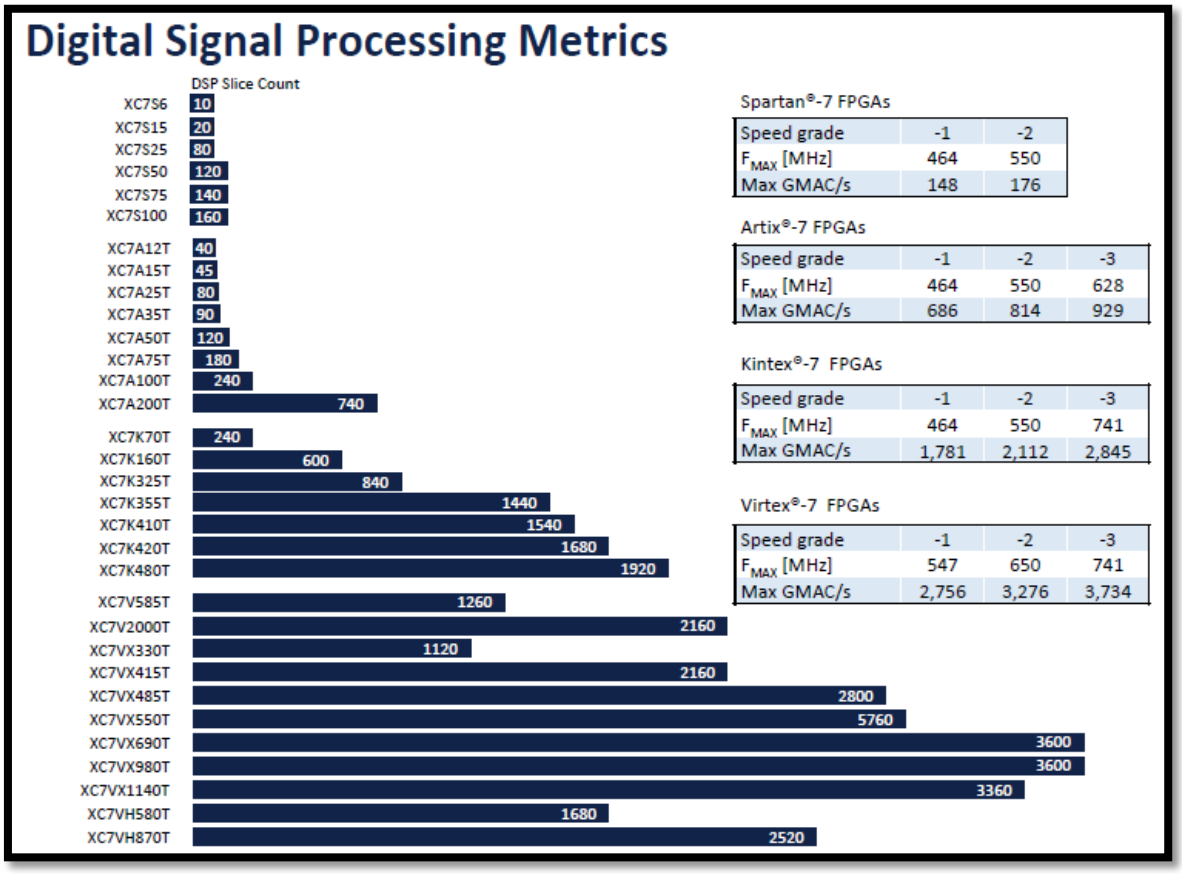


Figure 7.2. Xilinx 7-Series FPGAs DSP performance

As Figure 7.2 depicts that, whilst the DSP performance and slices increases, family members gets powerful. XC7S, XC7A, XC7K and XC7V are named as Spartan, Artix, Kintex and Virtex respectively.

# 8. CONCLUSION

Various taps and low pass FIR filters are implemented in FPGA. Systolic Multiply-Accumulator architecture is used to preserve resource. As it can be seen from the results that parallel multichannel implementation is possible as long as the DSP slice is available. For one channel, 478 taps can be executed to fully use the DSP slice. 8-channel is executed to show parallel implementation, however, it can be seen from the results that up to 64 taps, there are more resources if we want to increase the number of channel. The Flip-flops and LUTs change slightly with respect to the number of taps and parallel multichannel. As long as DSP slice is available for multichannel implementation, latency does not change. Filtered signals are captured through ILA and DAC. Phase difference between the original noisy signal and the filtered signal is small. In this implementation, it is seen that hardwares that are based on FPGA parallel data processing capabilities can get rid of the unwanted portions of signals with FIR filters in minimum time considerations. The low cost Artix 7-series FPGA with low-taps filter the noisy signal as intended. Considering the cost of the conventional DSP processors, as it can be seen that an FPGA with DSP handling capabilities displays better results.

Noisy signals which are involved to control cycle without filtering cause some troubles related to instability and misdetection. Yet another important issue is that the systems that are processed only microprocessors have performance and response time loss due to long time of filtering the signals. In power electronics applications, a large number of signals mostly included switching noises need to read and evaluate regarding protection or control. That is why the need of filtering the signals with minimum latency is a must to enhance the performance of the systems.

Minimum latency and parallel multichannel processing features enable the users to real-time process with no latency for critical systems such as power electronic devices used for smart grids for the future scope and also the unwanted portions of the signals (noise) that are produced from switching devices etc. can be eliminated. Thanks to the parallel processing implementation feature of the FPGAs, the users can use these systems required high speed applications.

Finally, Artix-7 FPGA used in this implementation is up to date, low-cost and shows high performance. Time considerations acquired from the results could be used as benchmark timings for further implementations and applications.

## REFERENCES


1. Mirzaei S., Hosangadi A. and Kastner R. (2007). FPGA implementation of high speed FIR filters using add and shift methods. *2006 International Conference on Computer Design*, 10(1), 308-313.

2. Qasim S. M., BenSaleh M. S. and Obeid A. M. (2012). Efficient FPGA implementation of microprogram control unit based FIR filter using Xilinx and Synopsys tools. *Proc. of Synopsys User Group Conference (SNUG)*, 3(1), 1-14.

3. Internet: From character to personality. (2005, March). *DSP: Designing for Optimal Results High Performance DSP Using Virtex-4 FPGAs,* URL: https://www.xilinx.com/publications/archives/books/dsp.pdf, Accessed: 20.04.2019.

4. Kuo, S., M., Lee, B., H., Tian, W. (2006). *Real Time Digital Signal Processing* (Second Edition). West Sussex: Wiley, 666.

5. Proakis, J., G., Manolakis D., G. (2007). *Digital Signal Processing: Principles, Algorithms and Applications*. (Fourth Edition). New Jersey: Pearson Education, 1033.

6. Woods, R., Mcallister, J., Lightbody, G., Yi, Y. (2017). *FPGA-based Implementation of Signal Processing Systems* (Second Edition). United Kingdom: Wiley, 488.

7. Internet: From character to personality. (2015, Nov). *FIR Compiler v7.2 LogiCORE IP Product Guide,* URL: https://www.xilinx.com/support/documentation/ip_documentation/fir_compiler/v7_2/pg149-fir-compiler.pdf, Accessed: 20.04.2019.

8. Churiwala, S. (2017). *Designing with Xilinx FPGAs*, Switzerland: Springer International, Publishing, 257.

9. Seely, J., Erusalagandi, S., Bethurem, J. (2017, April). *The Microblaze Soft Processor: Flexibility and Performance for Cost-Sensitive Embedded Designs*, 14.

10. Internet: From character to personality. (2012, Jan). *AXI Reference Guide,* URL: https://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/v13_4/ug761_axi_reference_guide.pdf, Accessed: 20.04.2019.

11. Internet: From character to personality. (2017, July). *Vivado Design Suite AXI Refence Guide,* URL: https://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/latest/ug1037-vivado-axi-reference-guide.pdf, Accessed: 20.04.2019.

12. Internet: From character to personality. (2016, Oct). *Integrated Logic Analyzer v6.2 LogiCORE IP Product Guide,* URL: https://www.xilinx.com/support/documentation/ip_documentation/ila/v6_2/pg172-ila.pdf, Accessed: 20.04.2019.

13. Internet: From character to personality. (2018, July). *7Series FPGAs and Zynq-7000 Soc XADC Dual 12-Bit 1MSPS Analog-to-Digital Converter,* URL:

https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf, Accessed: 20.04.2019.

14. Internet: From character to personality. (2016, Oct). *XADC Wizard v3.3 LogiCORE IP Product Guide,* URL: https://www.xilinx.com/support/documentation/ip_documentation/xadc_wiz/v3_3/pg091-xadc-wiz.pdf, Accessed: 20.04.2019.

15. Internet: From character to psersonality. (2018, April). *Vivado Design Suite User Guide Model-Based DSP Design Using System Generator,* URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug897-vivado-sysgen-user.pdf, Accessed: 20.04.2019.

# CURRICULUM VITAE

**Personal Information**


Surname, Name : AYDIN, Cihan

Nationality : T.C.

Date and Place of Birth : 03.11.1991, Kayseri

Marital status : Single

Phone number : 0 (537) 469 58 55

e-mail : cihanaydin1991@gmail.com


**Education**

| Degree | School/ Program | Graduation Date |
|---|---|---|
| MSc | Gazi University / Smart Grids | 2019 |
| Undergraduate | Çukurova University / Electrical-Electronic Engineering | 2014 |
| High School | Mehmet Emin Resulzade Anatolian High School | 2009 |

**Professional Experience**

| Year | Place of Work | Position |
|---|---|---|
| January 2019-Ongoing | TÜBİTAK SAGE | Researcher |
| September 2017-January 2019 | Aselsan-Havelsan Teknoloji Radar | SystemEngineer |
| September 2015-January 2017 | Best Elektrik Taahhüt Ticaret AŞ. | ElectricalEngineer |

**Foreign Language**

English

**Publications**


1. Aydin, C., Sefa, I. (27 June – 29 June 2019). FPGA Implementations of Multichannel FIR Filters. *2019 11th International Conference on Electronics Computers and Artificial Intelligence (ECAI)*, Pitesti – Romania.

**Hobbies**

Football, technology, cars, swimming, reading

*GAZİ GELECEKTİR...*