DESIGN AND IMPLEMENTATION OF AN EDUCATIONAL FM RECEIVER WITH FPGA USING SDR TECHNIQUES

2022
MASTER THESIS
ELECTRICAL AND ELECTRONIC ENGINEERING

Abdalla M Mohamed ELSALHIN

Thesis Advisor
Assist. Prof. Dr. Bilgehan ERKAL

DESIGN AND IMPLEMENTATION OF AN EDUCATIONAL FM RECEIVER WITH FPGA USING SDR TECHNIQUES

Abdalla M Mohamed ELSALHIN

Assist. Prof. Dr. Bilgehan ERKAL

T.C.

Karabuk University

Institute of Graduate Programs

Department of Electrical and electronic engineering

Prepared as

Master Thesis

KARABUK

December 2022

I certify that in my opinion the thesis submitted by Abdalla M Mohamed ELSALHIN titled “DESIGN AND IMPLEMENTATION OF AN EDUCATIONAL FM RECEIVER WITH FPGA USING SDR TECHNIQUES” remains fully adequate in scope as well as based on the quality as a thesis for the degree of Master of Science**.**

Assist. Prof. Dr. Bilgehan ERKAL …………………
Thesis Advisor, Department of Electrical and Electronic Engineering

This thesis is accepted by the examining committee with a unanimous vote in the Department of Electrical and Electronic Engineering as a master thesis. December 19, 2022

<u>Examining Committee Members (Institutions)</u>	<u>Signature</u>
Chairman: Assoc. Prof. Dr. Salih GÖRGÜNOĞLU (KU)	…………………
Member : Assoc. Prof. Dr. Hüseyin DEMIREL (KBU)	…………………
Member : Assist. Prof. Dr. Bilgehan ERKAL (KBU)	…………………

The degree of Master of Science by the thesis submitted is approved by the Administrative Board of the Institute of Graduate Programs, Karabuk University.

Assoc. Prof. Dr. Müslüm KUZU …………………
Director of the Institute of Graduate Programs

"I declare that all the information within this thesis has been gathered and presented in accordance with academic regulations and ethical principles and I have according to the requirements of these regulations and principles cited all those which do not originate in this work as well."

Abdalla M Mohamed ELSALHIN

ABSTRACT

M. Sc. Thesis

DESIGN AND IMPLEMENTATION OF AN EDUCATIONAL FM RECEIVER WITH FPGA USING SDR TECHNIQUES

Abdalla M Mohamed ELSALHIN

Karabük University
Graduate School of Natural and Applied Sciences
Department of Electrical and Electronic Engineering

Thesis Advisor:
Assist. Prof. Dr. Bilgehan ERKAL
December 2022, 94 page

The complexity of hardware in traditional FM radio lead to huge radio size, noisy, difficulties in maintenance as well as higher production costs. In addition to limitation on protocols in addition, which is associated with frequency bands supported, which lead to utilize diverse radio devices for diverse networks also communication protocols. Software Defined Radio techy provides an efficient in addition, which is associated with comparatively inexpensive solution to these problems replacing a lot of cascaded electronics components via a single digital chip FPGA (Field Programmable Gate Array) resulting in low production cost, immunity to noise, ease of maintenance as well as enormous reduction of radio size. In this paper, digital FM receiver was designed in SIMULINK in addition, which is associated with implemented successfully on. The results of the study are obtained in two stages: simulation results in addition, which is associated with test results. Simulation results come from ideal simulation studies under MATLAB environment utilizing the codes listed on Appendix A.

Test results come from as recordings from real world tests of FMRX frameworks implemented on FPGA. Two frequency modulated test signals utilized in each stage are FM1 as well as FM2 which are derived utilizing two test recordings A1 also A2 respectively as the modulating signal. Each test signal lasts 10 seconds as well as sampled at a rate 48KSps. Results from simulation as well as tests, which are demodulated audio, are recorded as a separate wave file

**Key Words : ** Sdr, Fpga, Vhdl, Matlab, Fm.
Science Code : 90523

ÖZET

Yüksek Lisans Tezi

EGITIM AMAÇLI SDR TEKNIKLERINE DAYALI FPGA TABANLI FREKANS MODÜLELI RADYO ALICISI TASARIMI VE UYGULAMASI

Abdullah M Muhammed ELSALIHIN

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğr. Üyesi Bilgehan ERKAL
Aralık 2022, 94 sayfa

Geleneksel (FM) radyoda donanıma dayalı karmaşıklığın ana noktası, daha büyük radyo boyutları, gürültü, bakım zorlukları gibi bazı noktalara yönlendirilmiş ve buna ek olarak daha yüksek üretim maliyetleri ile ilişkilendirilmiştir. Ayrıca, frekans bantlarına bağlanan desteklenen protokollerdeki sınırlamalar, iletişim protokollerinin yanı sıra farklı ağlar için farklı radyo cihazlarının kullanılmasını zorunlu kılmaktadır. Aynı şekilde, Yazılım Tanımlanabilir Radyo teknolojisine, örneğin, Programlanabilir Kapı Dizisi Alanına dayalı tek bir dijital çip ((FPGA) ile birden fazla kademeli elektronik bileşenin değiştirilmesi gibi kaydedilen bu sorunlara karşı etkili ve nispeten ucuz bir çözüm sunulmuştur. Düşük üretim maliyeti, gürültü bağışıklığı, radyoya bağlı bakım kolaylığı ile bağlantılı Alan Programlanabilir Kapı Dizisi) boyutunda muazzam bir azalma sağlar. Bu çalışmada sayısal (FM) alıcı (MATLAB) kullanılarak tasarlanmış ve üzerinde başarıyla uygulanmıştır. Ayrıca, çalışmanın iki ana aşamada elde edilen bazı gerekli sonuçları vardır: simülasyon sonuçları ve test sonuçları. Ayrıca simülasyon sonuçları (MATLAB) ortamında bazı önemli kodları kullanarak simülasyon çalışmalarının idealinden gelmektedir. Bu nedenle, test sonuçları, (FPGA) üzerinde uygulanan (FMRX) çerçevesine dayalı gerçek dünya testlerinden kayıtlar olarak gelir. Her adımda, modülasyon sinyalleri olarak sırasıyla A1 ve A2 test kayıtları kullanılarak türetilen (FM1) ve (FM2) olan iki frekans modülasyonlu test sinyali kullanılmıştır. Ek olarak, her test sinyali on saniye sürer ve (48KSps)'de örneklenir. Ek olarak, demodüle edilmiş sesle bazı gerekli testler ile desteklenen simülasyonların sonuçları ayrı bir wave dosyası olarak kayıtlı kalır. Sonuç olarak, bu kazanımlar birbirleriyle karşılaştırılarak iyi sonuçlar elde edilmiştir. Ayrıca sonuçlar, (FMRX) çerçevesinin iletişim mühendisliği öğrencilerine dayalı çok yönlü bir eğitim ortamı olduğunu kanıtlamaktadır.

Anahtar Kelimeler **:** Sdr, Fpga, Vhdl, Matlab, Fm.
Bilim Kodu **:** 90523

ACKNOWLEDGMENT

I would like to give thanks to my advisor, Assist. Prof. Dr. Bilgehan ERKAL, for his great interest and assistance in preparation of this thesis.

CONTENTS

Page

Page

LIST OF FIGURES

LIST OF TABLES

SYMBOLS AS WELL AS ABBREVITIONS INDEX

ABBREVITIONS

FM	:Frequency Modulation
AM	:Amplitude based on Modulation
SDR	:Software as a Defined Radio
DSP	:Digital Signal Processors
FPGA	:Field Programmable regarding to a Gate Arrays
RF	:Radio Frequency
IF	:Intermediate Frequency
SNR	Signal-to-Noise Ratio
PSK	Phase Shift Keying
DAC	:Digital-towards-Analog Converter
PLL	:Phased Locked Loop
ADC	:Analog towards a Digital Converter
ASIC	:Application based on a Specific Integrated Circuit
DDC	:Digital Down Converter
HDL	:Hardware Description Language
LUT	:Look Up Table
RAM	:Random Access Memory
VHSIC	:Very High Speed Integrated Circuit
VHDL	:VHSIC Hardware Description Language
PCB	:printed circuit board
FF	:Flip-Flop
IC	:integrated circuit
PLD	:programmable logic device
CLB	:configurable logic blocks
RTL	:Register Transfer-Level

CHAPTER 1

INTRODUCTION

Wireless communication channels have become more commonplace since the advent of cellular communication in the past twenty years. Right now, hardware flexibility is necessary for the development of wireless applications as well as wireless techy. Making a new based on radios in response towards the evolution of wireless applications which is associated with the standards is time-consuming as well as extremely expensive [1]. Therefore, software based on the radio can address these issues via converting analog components to digital components. Programmable logic devices, for instance, Field Programmable Gate Arrays, can be utilized to implement radio functions (FPGAs).

Consequently, it was frequently challenging to update as well as modify these frameworks. Several of the encoders, modulators as well as filters, in addition, which is associated with decoders needed via some sorts based on the communication frameworks can now be implemented in software for the reason that to advancements in digital signal processor (DSP) techy [2].

While some frameworks implement all of their baseband functions in software, others just utilize it for minor tasks. Software defined radios (SDRs) are frameworks that the implementation based on all of their baseband functionality in software [3].

SDR are evolving into a preferable substitute for conventional radio frameworks thanks largely to their adaptability. Reprogramming SDRs makes them somewhat upgradeable for the reason that they have this capability. Though the term "future-proof" may be utilized too strongly, the idea is nonetheless well-made.

SDRs can accommodate a huge variety of waveforms as well as coding schemes, which makes them an extremely appealing alternative to traditional frameworks when waveforms evolve in addition, which is associated with require expensive replacement. In some circumstances, SDRs can take the place of a number of conventional radios [4].

Compared to, say, AM, the FM app has a higher signal-to-noise ratio (SNR). For audio communication channels, it is permitted to prepare from 5 towards 15 dB as compared to AM optimization the scheme. More improvement can be achieved via utilizing FM Wider deflection. For signal-to-noise (SNR) improvement in FM frequencies, additional methods, for instance, pre-emphasis on loudness Frequencies with the corresponding focus in the receiver, are typically applied. FM receivers typically incorporate selections that exclude AM noise for the reason that FM signals have a set amplitude, greatly enhancing SNR. [5] [6].

1.1. LITERATURE REVIEW

Since the discovery of the first wireless communication in the late 1980s, many advancements in radio communication techy have been made with the goal of ensuring that radio utilizers are always connected. Attributable to bandwidth restrictions at the time, triumphant radio, the first sort of broadcast, employed analog voice communication. In the 1950s, broadcast communication took over, in addition, which is associated with analog television transmission became the norm. This type of communication utilized a lot of bandwidth but provided exceptional customer service. The usage of computers, which could send data over great distances which is associated with make utilize of both wired in addition, which is associated with wireless connections, gained popularity in the 1960s. After the development of cell phones, wireless voice communication was later discovered, enabling transmission from any location. However, the mobile devices were difficult to utilize for the reason that they were not portable [4].

Ahmeda Gareane talked about utilizing the Softrock Ensemble RXTX SDR transceiver for FM signal transmission as well as receiving. Two input frequencies also a modulation index were utilized as inputs for the developed frameworks to generate in addition, which is associated with plot FM signals utilizing MATLAB. The FM modulation as well as FM demodulation stages of the frameworks were both present. To convey a recorded audio file, the carrier signal frequency was modified during the FM modulation stage dependent on the signal strength. The resulting modulated wave is then transferred to the receiver. A low pass filter was utilized in the FM demodulation stage to recover the audio signal from the modulated wave, removing the carrier frequency in addition, which is associated with allowing the audio signal to be transmitted to the speaker. Then, Audacity was utilized to examine in addition, which is associated with listen to the signal. Modulation as well as demodulation were performed utilizing Matlab software methods [7].

Ali Hander created an AM receiver in addition, which is associated with utilized SDR techniques to put it in an FPGA. His study's major goal was to develop a low-cost, straightforward FPGA-based platform for teaching SDR fundamentals. First, MATLAB scripts were utilized to create a simulation environment for the investigation. MATLAB scripts were utilized to generate an AM test signal from a collection of test signals. The signal was utilized in the testing as well as simulation of Implementation of FPGA. A foundation for that (VHDL) design based on the FPGA-based SDR framework which is also provided via simulation code. To compare in addition, which is associated with contrast the test also simulation findings, another MATLAB script was built. The results of those tests on the two signals showed that, since the greater SNR ratio is better, based on some sorts of some main tests which is associated with the signal A1 remain superior than the main tests by means of a signal A2. It has been found that the sort based on a real-world (SNR) findings remained marginally lower than the value of the main value of the simulations based on the (SNR) when has been compared with the main actual real-world values which is associated with the simulations for each test signal. SNR levels greater than the value 20dB, which were considered acceptable for an AM receiver, remained found in test results. Whereas test as well as simulation findings demonstrated that an effective candidate for AM demodulation in addition, which is associated with reception is an FPGA AM RX frameworks. The developed and implemented FPGA AM RX framework remained which is associated with helpful in teaching fundamental SDR principles, which served as the study's major focus [8].

In the Caner KREMTC investigation, Matlab codes were utilized to model the modeling of the main amplitude modulated based on the radio transmitter in (FPGA). The amplitude modulated transmitter remained then that has been implemented on the (FPGA) board after the VHDL code remained generated on (ISE) Design Suite (14.7). (Mimas Spartan 6). The Audacity program sends a sample sound recording towards the (FPGA) card via the sound card in order towards modulating it. ADC (LM4550) card sent an analog signal towards the FPGA card, which then demodulated in addition, which is associated with the recorded the transmitter signal utilizing the HDSDR application.

The DAC (LM4550) card on the FPGA card converted the analog transmitter signal into analog form, which it then passed towards the laptops based on the sound card's microphone input. Finally, utilizing Matlab code, the captured transmitter signal was offline demodulated, in addition, which is associated with the the output was saved to the hard drive. The data analysis revealed that, for the identical test signal, there was very little variance amongst the simulation findings as well as the actual test results. Given that the signal was subjected to outside noise during actual tests, this was thought to be a totally plausible circumstance. A similar average value of about 20 dB was seen when the acquired SNR values were evaluated; this value was suitable for AM modulation. It was determined that the (FPGA) based on the AM transmitter frameworks could successfully realize AM broadcasts. The Ali HANDER architecture has produced a solid foundation for the deployment in addition, which is associated with the training of SDR frameworks on FPGA [9].

The design process in addition, which is associated with the implementation outcomes of a software defined radio (SDR) employing an Altera Cyclone II family board were provided via Hikmat N. Abdullah. The Cyclone II development in addition, which is associated with the educational board, Embedded Matlab blocks, as well as Matlab/Simulink were utilized in the implementation. The Matlab/Simulink platform was utilized to implement the idea initially. The Simulink HDL coder was then utilized to convert it to VHDL level. The Quartus II 9.0 Web Edition® software was utilized to synthesize the design before it was downloaded to an Altera Cyclone II board. The findings demonstrated that employing programmable logic tools, SDR implementation is simple to create in addition, which is associated with the comprehend. A useful design flow of the process utilized to obtain VHDL netlists that can be downloaded to FPGA boards was also provided via Hikmat N. Abdullah [10].

CHAPTER 2

SOFTWARE DEFINED RADIO (SDR)

SDR has gained popularity as an alternative to conventional radio frameworks mostly for the reason that of its flexibility. The potential for SDR has been reprogrammed, so it can be upgraded towards some extent. Although proving the future might be a fairly bold statement, the idea is well made. A huge variety of waveforms as well as encoder charts are supported via SDR. This is a particularly appealing choice for the reason that conventional frameworks need to be replaced on a costly basis as techy develops in addition, which is associated with the waveforms change. SDRs might only need a small upgrade. SDRs can occasionally take the place of a large number of conventional radios [11]. The expense of the job task is lower as well as the radios are simple to utilize. There are various ways to enumerate what Software Defined Radio (SDR) is. A radio remains a wireless device that sends which is associated with receives frequencies as well as information. In order to access the SDR, a number of issues must be overcome, including modifying the frameworks specifications in accordance with the several applications we may return, including modification, extraction, in addition, which is associated with the encryption. As a conclusion, this information aids in the reception of these criteria [12].

2.1. SDR BASED ON SOME AADVANTAGES

- Point as well as Click Control
- Easy based on the Tuning
- A PC Is based on the Sharing of the Workload
- Cheaper for several situations
- Is Smaller
- Visual based on the look at a signal
- Open CV Platforms

- Custom based on the Filtering Utilizes modern techy

2.2. SDR AS WELL AS DISADVANTAGES

- Filtering based on the Traded for Space
- Hard towards run on old computers
- Sending remains more expensive
- The framework remains dependent on PC
- Software based on some Limits

2.3. IDEAL BASED ON THE SDR DESIGN

One of the most important modern technologies for aiding the main communication based on in military service insecurity, war, in addition, which is associated with the peace is the software-defined radio frameworks (SDR). Instead of utilizing a traditional radio, SDR converts wireless signals at the best radio frequencies to an analog IF that can be utilized with regular radios. The conversion of an analog based on the signal towards a digital value of frequency (ADC) in an IF as well as the signal's conversion from a digital to an analog FM frequency (DAC) in an IF are both depicted in Figure 2.1. The handling of hardware in the receiver in addition, which is associated with the transfer based on the signal routed via the main converter sampling rate via the interface (ADC). SDR may process baseband utilizing a variety of digital devices, including as field-programmable gate arrays as well as digital signal processors (DSP) as well as (the main value FPGA).

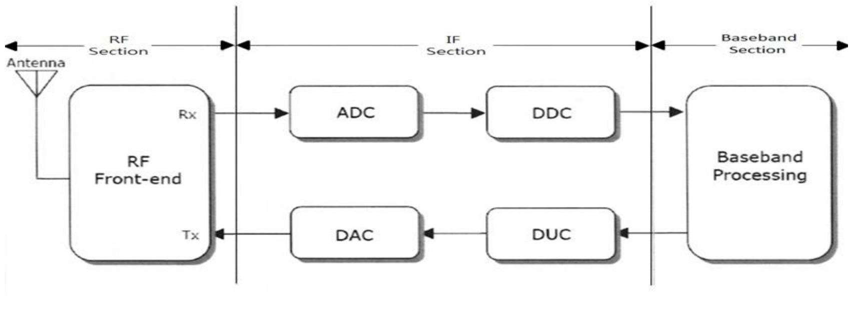


Figure 2.1. The Functional main block diagram based on the wireless as well as communication framework.

A Digital devices include benefits including reduced energy use, fast processors, in addition, which is associated with the flexibility. However, there is a contrast amongst the extreme degrees of flexibility as well as the rise in energy consumption for DSP to minimally limit the flexibility in addition, which is associated with the lower energy consumption than ASICs. FPGA offers more flexible, less expensive in addition, which is associated with the energy-efficient devices than DSP as well as ASICs. FPGA in addition, which is associated with the redesign have improved SDR performance [13].

2.4. THE RESEARCH MOTIVATION

SDR is mainly Looks like as a several techies in terms of development, goes via the name Speakeasy, in addition, which is associated with the has both military as well as non-military uses. Amongst (1991) as well as (1995), the United States' naval forces employed it. The fundamental laws, knowledge, radio program, wireless communication, in addition, which is associated with the programming were all greatly improved via this techy. Currently, all SDR software is inexpensive towards purchasing. [14].

2.5. THE SDR AS A HARDWARE

2.5.1. The Traditional Main Receiver

Towards determining the main associated sign in the main associated carrier frequency based on the setting of frequency of shifting, in addition, the candidate remains filtering as well as isolated from others utilizing the conventional receiver, the traditional demodulation, in addition, which is associated with the three processes. Enlargement via mass demodulation is utilized to insert compensation for enlargement-related transport losses. The majority of traditional receiving setups have been utilizing various strategies for about a century since they must transport the signal to the required level for circuit demodulation. Figure 2.2 illustrates how the fundamental design helps to distinguish amongest the conventional in addition, which is associated with modern reception SDR methods.

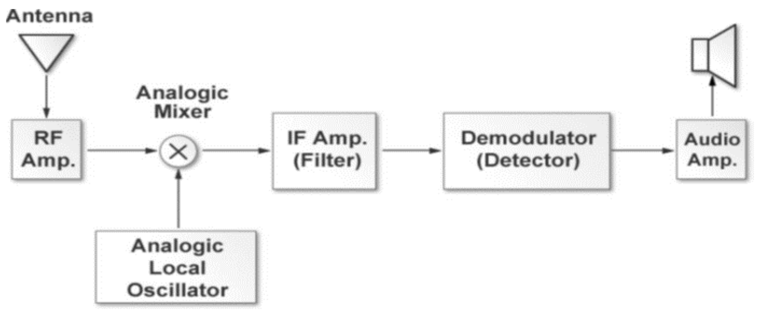


Figure 2.2. Internal actual blocks based on the super heterodyne of the receiver.

Based on the Figure. 2.2. Has represents that how the antenna based on the main interferes with signals. A transferring the main associated reference towards the mixer during the other input, which gets the contribution oscillator decoration in addition, which is associated with the designating local frequency oscillator, is the signal amplified during the duration of RF phase, which operates in the frequency range of benefit. The mixer is in charge of changing the frequency to the medium frequency via adjusting the radio, which translates the frequency signal mediator (IF) which is IF.

The main purpose based on the appointing a frequency oscillator is utilized to verify if the frequency signal's timing variance is equal to (IF). For instance, if the IF was supposed to be at 10.7 (MH)z in addition, which is associated with the the FM station's frequency remains 100.7 (MHz), the oscillator which has been required to be set to 90 MHz as an outcome of the low side of the main point of transformation. Therefore, the next phase based on the involvement of weakening all candidate wave signals, though undoubtedly a portion of the spectrum. The bandwidth prevents the band's received signal from being visible. If it only utilizes one substitution, the demodulator restores the original signal changed at the conclusion of the stage through the speakers. It depends on the main goal purpose via which it's intended towards boosting the signal processing recipient device. In addition, based on the cross of INFO learned towards a loudspeaker connected towards the speaker [15, 16].

2.6. THE SDR AS A RECEIVER

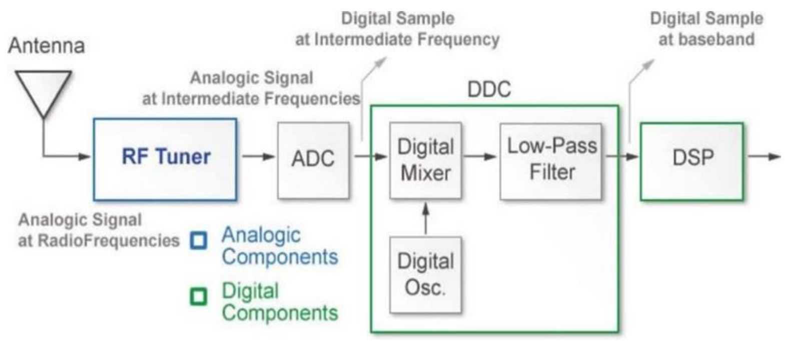

Antenna
Digital Sample at Intermediate Frequency
Digital Sample at baseband
Analogic Signal at Intermediate Frequencies
DDC
RF Tuner
ADC
Digital Mixer
Low-Pass Filter
DSP
Digital Osc.
Analogic Signal at RadioFrequencies
Analogic Components
Digital Components

Figure 2.3. The main Block Diagram based on the SDR AS a Receiver.

Based on the Figure. 2.3 to take in the SDR signal in addition, which is associated with be the first to pass it the first three boxes of the alternative reception device are where the procedure is carried out via the RF tuner, which converts the analog signal to IF to be identical. Frameworks [17].

At that moment, a cross-reference IF the frequency band is altered via the ADC, the following phase in addition, which is associated with the digital converter are fed new data. DDC is an important component of the SDR frameworks; it is inexpensive as well as has three basic components. Follows:

- The digital based on the local oscillator.
- The digital based on the mixer.
- The main Finite of pulsation based on the response (FIR) based on the low-scrolling IF filter.

Reference based on the transfers from the counter of phase elements via the analysis [18] towards the matching baseband in our digital mixer. Furthermore, the reference remains needed far away which is associated with up to 0Hz, in addition, which is associated with the variance with the bandwidth together with being a low-pass filter as well as detecting any receiver portion is a suitable signal. This is a modified digital local oscillator. Another strategy involves lowering the sampling frequency or the sampling ratio in order to create new samples from the baseband via splitting the original sample's frequency utilizing an N element. The decimation element is what it has been called. Via Nyquist, based on the main ratio of the final sample may be less than twice the higher-frequency components theory. The eventually, samples moving to baseband digital signal processing in a DSP box, for instance, decoding as well as demodulating [19].

2.7. SDR TRANSMITTER

As shown in In Figure. 2.4, the baseband of signal that the SDR will transmit is produced via the DSP. The first box, known as a DUC (Digital User Converter), converts the baseband signal into an IF signal via adjusting its pass band.

When (RF) remains moving based on towards the main high-frequency value of signal, the DAC sends the main associated samples towards the field analog. The signal is then enlarged as well as sent from the main associated main antenna. DUC Filter is in charge of the baseband signal's high sample ratio, which is necessary for the elements to operate properly in addition, which is associated with the causes the reverse process towards occurring at the main associated reception based on a frequency [18, 19].

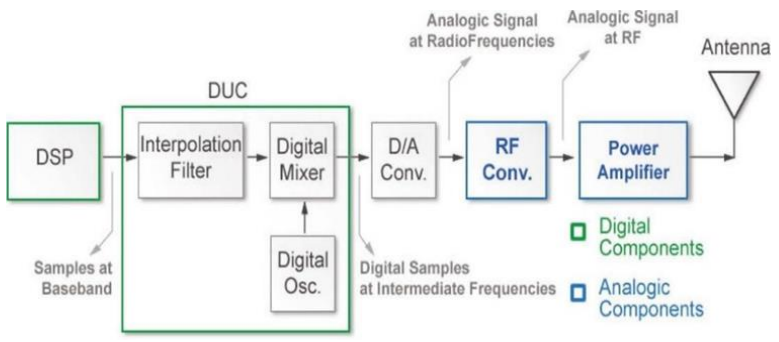


Figure 2.4. The block Diagram based on the SDR Transmitter.

CHAPTER 3

THE MAIN FIELD BASED ON THE PROGRAMMABLE GATE ARRAYS (FPGA)

The (FPGA) by means of a device with reconfigurable gate array logic circuits in a matrix. The internal circuitry of an FPGA is coupled in such a way during construction that in addition, the main associated hardware implementation based on the software application outcomes. FPGAs employ specialized hardware to process logic rather than an operating frameworks. Since FPGAs are parallel devices, various activities do not compete for the same resources. Consequently, one aspect of the application's performance changes when more processing is added is not influenced.

Additionally, several control of some loops can run at various ratios on a single FPGA chip. The operator-mandated I/O can be blocked via the main critical interlock based on the logic, which can be imposed via FPGA-based control frameworks. However, unlike hard-wired printed circuit board designs that have reliable hardware resources, FPGA-based frameworks have the potential to rearrange their internal circuitry to aid in reconfiguration when the control framework is deployed in the field.

FPGA provides the specialized hardware circuitry with dependability in addition, which is associated with the performance. Through the usage of FPGA, millions of logic gates can be combined into the main associated single based on an integrated circuit (IC) chip to replace thousands of discrete elements. The main internal based on several resources of the FPGA chip include a matrix of reconfigurable logic blocks surrounded via a peripheral of I/O blocks, as depicted in Figure 3.1.below. The several signals remain routed inside the FPGA matrix utilizing wire routes as well as programmable connections switches.

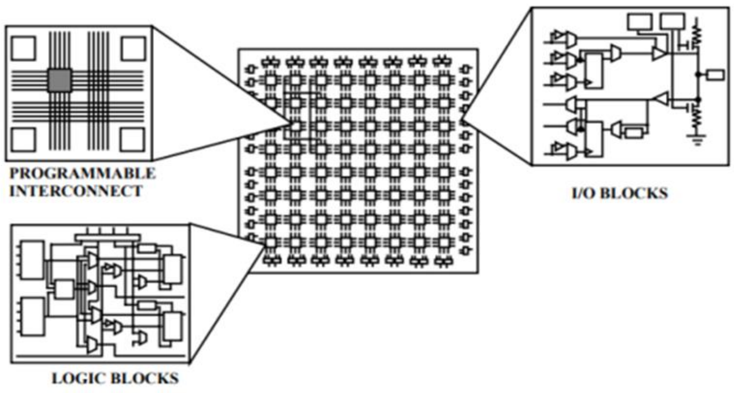


Figure 3.1. Internal of a Structure based on the (FPGA).

3.1. FPGA ARCHITECTURE

The FPGA structure is made up of various parts, including:

- The look-up table (LUT) is a component that performs a variety of logical operations.
- Flip-Flop (FF): The LUT result is stored in this register component.
- Wires: These parts link other parts together.
- Input as well as an output ((I) as well as (O)) pads: These physical ports remain utilized towards transfering data into in addition, which is associated with the out of the FPGA.

The collection of these parts combine to provide the fundamental framework of an FPGA, as shown in Figure 3.2. Although this structure is effective at implementing any algorithm, the finished implementation's proficiency remains limited based on some terms of computed output as well as realistic clock frequency, in addition, which is associated with the required resources.

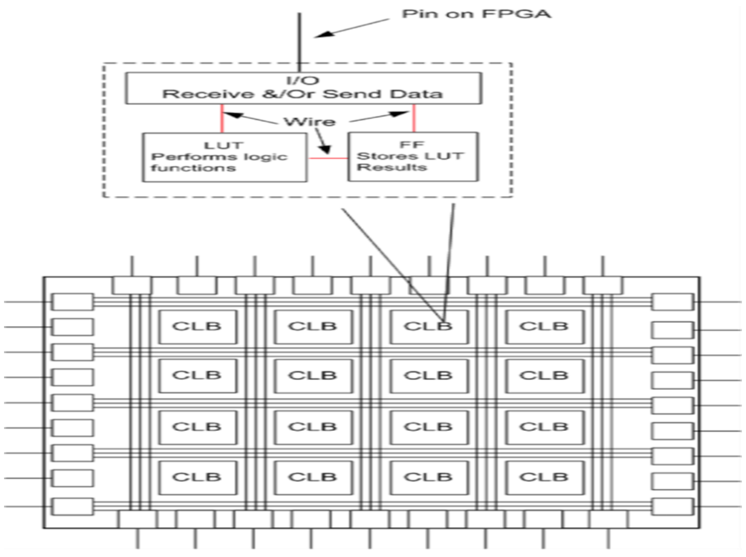


Figure 3.2. The main basic of (FPGA) the main Architecture.

The main modern FPGA design consists of a number of fundamental building blocks that are joined via additional computational in addition, which is associated with storage units to improve the efficiency in addition, which is associated with the computational density of the hardware. The following parts will address the other elements that remain as follows: follows:

- Embedded memory towards keeping the dispersed info.
- Phase-locked several loops (PLLs) for various clock ratios to drive the FPGA fabric.
- Devices for serial transmission in addition, which is associated with the reception at high speeds.
- Memory controller's off-chip

The combination of these elements enables the FPGA to implement any software algorithm based on some running on processors, creating the contemporary FPGA architecture displayed in the Figure. 3.3. Below.

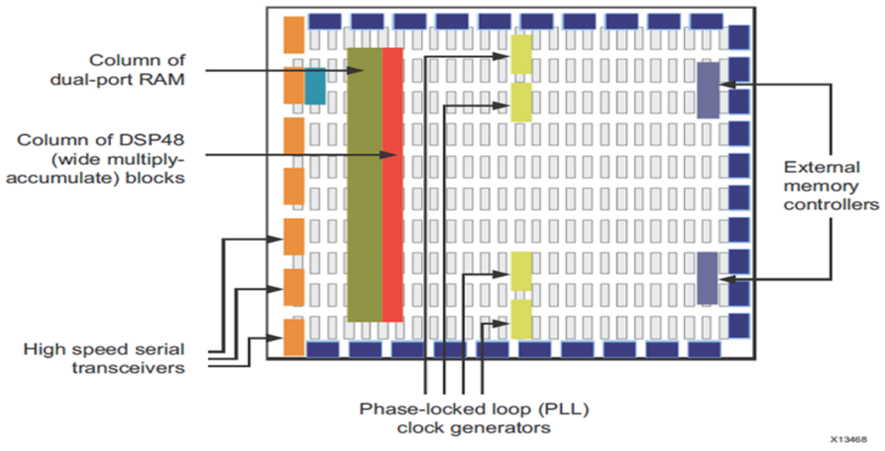


Figure 3.3. The main contemporary based on (FPGA) which associated with the Architecture.

3.1.1. Logic Cells

The simple each logical cell in an FPGA can be set towards performing as a wide range of operations. There is a fixed number of entries in addition, which is associated with the exits for each logic cell. The type of logic towards cells found in FPGAs is follows:

- Multiplexer logical based cells, for instance, Actel (FPGAs)
- Cells with memory-based logic, for instance, Xilinx (FPGAs)

A very broad view of the fundamental internal structure of an FPGA is presented. in the main Figure. Of the .3.4.below.

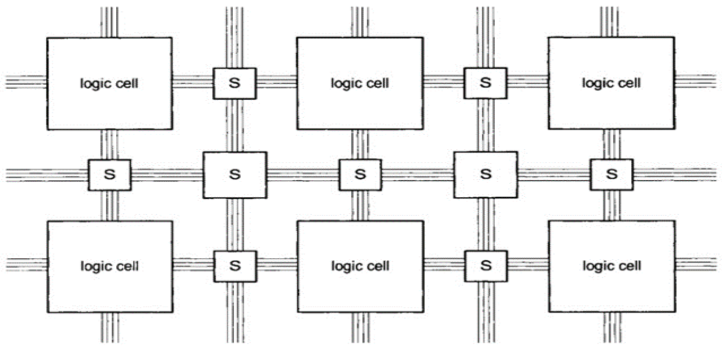


Figure 3.4. The Logic based on a Cells.

As presented in Figure. 3.4. a programmable connectivity in addition, which is associated with the adjustable logic make up the internal structure of an (FPGA) cells.

CHAPTER 4

FREQUENCY MODULATION (FM)

4.1. FM MODULATION AND DEMODULATION

4.1.1. Frequency Modulation

Frequency modulation refers to the process where via the carrier surge frequency is altered to match the signal density (FM) [20].

When a signal is modulated with frequency, only the carrier surge's frequency is altered to match the signal the carrier surge amplitude, or the amplitude of the modulated surge residue [21].

The immediate amplitude based on the main associated signal, as depicted in Fig. 1, determines how the frequency divergence of the transporter surge counts. The carrier frequency is 0 when the signal voltage is zero, as it is for A, C, E, as well as G is unchanged. The careful diverge period demonstrates that the transporter frequency is too excessive when the signal approaches its positive summit, as at B also F. However, as evidenced via the most spaced period [22], the carrier frequency is small to minimum due to the negative peak of the signal as at D.

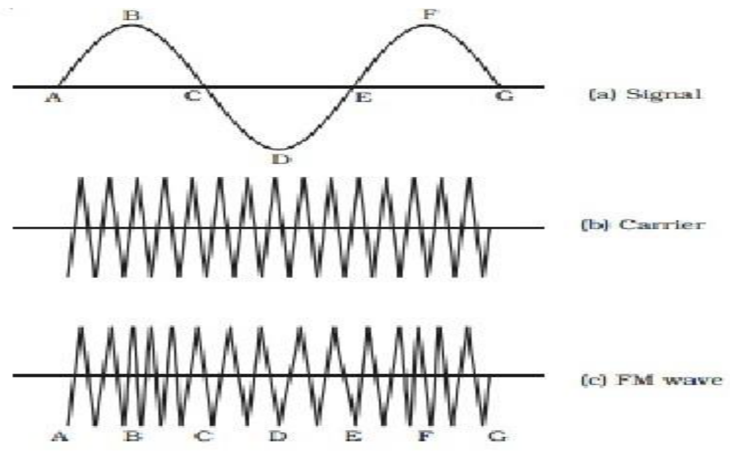


Figure 4.1. Frequency Modulation.

4.1.2. Frequency Demodulation

To correctly demodulate a modulation as well as restore the original signal, as with any modulation, is essential. The FM demodulator is also known as an FM detector, an FM discriminator, or any other number of names.

The frequency fluctuations of the entering signal can be transformed into amplitude variations on the output via a variety of diverse FM demodulators, all of which are capable of doing so. If data is being transmitted across the framework, these are typically fed into a digital interface or an audio amplifier.

In terms of practical application, frequency modulation (FM) is comparable to a straightforward balance where the baseband data signal's data or message signal changes the transporter wave's recurrence. Sound transmissions that are transmitted through FM radio channels are the most well-known FM signals. On the other hand, the FM can also communicate Radio Broadcast Data frameworks (RBDSs), which are sophisticated computerized data with modest transmission speeds. [23].

4.2. FREQUENCY MODULATION ADVANTAGES AS WELL AS DISADVANTAGES

There are a number of benefits also drawbacks to utilizing this type of modulation, as with any other. Before making any option or judgment regarding its utilize, they must be taken into account:

4.2.1. Advantages Based on the Frequency Modulation, FM

Frequency modulation has become widely used in a range of applications, including two-way radio communications as well as high-quality audio transmission, as a result of its many advantages. Resilience to noise: resistance to noise is one of its main benefits.

Frequency modulation has the robustness to changes in signal strength as a distinct advantage. Only frequency fluctuations are used to convey the modulation. This means that as long as the signal does not decrease to a level where the receiver cannot handle it, any changes in signal level won't have an impact on the audio output. For the reason that of this, FM is perfect for mobile radio communication uses like more broad two-way radio communication or portable uses where signal levels are expected to vary greatly. The resistance of FM to interference in addition, which is associated with noise is another benefit. For the reason that of this, FM is utilized to transmit high-quality broadcast signals.

Modulation is simple to implement at the transmitter's low power stage: The transmitters are connected to another benefit of frequency modulation. It is possible to apply the modulation to a low power stage of the transmitter, which is associated with a linear type of amplification is not required to enhance the ratio of the signal's initial power to its ultimate value.

Frequency modulated signals can be amplified effectively utilizing RF devices: In a transmitter, FM signals can be amplified utilizing non-linear RF amplifiers, which are more effective than the linear ones needed for signals with any amplitude changes, for instance, AM as well as SSB. The use of FM is thus more practical for portable two-way radio applications for the reason that less battery power is needed for a given power output.

4.2.2. Disadvantages Based on the Frequency Modulation, FM

The drawbacks of frequency modulation, like those of all other frameworks as well as methods, must be taken into account before deciding whether to utilize FM.

FM has a lower spectral efficiency than several other modulation formats: In some cases, frequency shift keying, a type of frequency modulation, is less effective at transmitting data than quadrature amplitude modulation in addition, which is associated with other phase modulation forms. So, the majority of data transmission framework utilize PSK as well as QAM.

Requires more complicated demodulator: One of the minor drawbacks of frequency modulation is that the demodulator is a little more complicated as well as hence a little more expensive than the extremely simple diode detectors used for AM. Due to the prevalence of radio integrated circuits with built-in frequency demodulators, this is a much less pressing concern today.

Around other modes have higher data spectral efficiency: In comparison to frequency shift keying, a type of frequency modulation, some quadrature amplitude modulation in addition, which is associated with phase modulation formats are more spectrally efficient for data transmission. As a result, PSK in addition, which is associated with QAM are used in the majority of data transmission frameworks.

Sidebands extend towards infinity   either side: The sidebands of an FM transmission might theoretically go on forever. Even though they are minimal for narrow band FM, they are often important for wideband frequency modulation transmissions. Filters are frequently used to reduce the transmission's bandwidth, but they distort the signal in some way. The majority of the time, this is not a big deal, but it is important to include these filters for wideband FM as well as make sure they are appropriately built.

4.3. MODULATION INDEX AS WELL AS DEVIATION RATIO

It is highly helpful to know how much modulation is actually there in a signal when utilizing a frequency modulated transmission.

This can be used to specify details like the signal's frequency modulation type, for example, whether it is wide-band or narrow-band. It is also highly helpful in making sure that every transmitter as well as receiver in a framework is configured to support a defined amount of modulation for the reason that it influences variables like the receiver bandwidth, channel spacing as well as the like.

Towards defining the level of modulation, figures known as the modulation index as well as deviation ratio are used.

4.3.1. Sinusoidal Baseband Signal

Mathematically, an FM-frequency sinusoidal continuous wave signal can be used to simulate a baseband modulating signal. Single-tone modulation is another name for this technique. An example signal's integral is:

$$\int_0^t \quad Xm(T)dt = \frac{sin\,sin\,(2\pi f_m t)}{2\pi f_m}$$

In this case, the expression for y(t) above simplifies to as below:

$$y_{(t)} = A_c 2\pi f_c t + \frac{f_\Delta}{f_m} sin\,sin\,(2\pi f_m\;\; t))$$

In addition, the amplitude Am of the modulating based on the sinusoid remains represented based on the peak deviation$f_\Delta = K_f A_m$.

A Bessel functions can be used to represent the harmonic distribution of a sine wave carrier modulated via such a sinusoidal signal, which forms the foundation for a mathematical knowledge of frequency modulation in the frequency domain.

4.3.2. Modulation Index

As In other modulation frameworks, the modulation index describes the degree of variation amongst the modulated variable in addition, which is associated with its unmodulated level. Variations in the carrier are relevant frequency.

$$h = \frac{\Delta f}{f_m}$$

Where fm is the highest frequency component found in the modulating signal xm(t), as well as f is the largest variance amongst the instantaneous frequency also carrier frequency. Therefore, the main ratio of the peak frequency based on the deviation of the carrier wave towards the frequency of the modulating sine wave is thought to represent the modulation index for the main of a sine wave modulation. If ḩ<< 1, the modulation remains called narrowband FM (NFM), in addition which is associated with its bandwidth is approximately 2Fm Sometimes modulation index ḩ<0.3 is considered as NFM, otherwise wideband FM (WFM as well as FM).(21)

4.4. FM BANDWIDTH

One of An FM signal's bandwidth remains one of its most important components. Any frequency modulated signal has sidebands that extend to either side. These actually go on forever, although their intensity diminishes as they do. Fortunately, it is possible to restrict an FM signal's bandwidth without significantly lowering the situation based on the quality.

Frequency modulation is widely employed in a variety of radio techy applications, including two-way radio communication as well as broadcasting. These applications can make effective use of its unique characteristics.

FM continues to provide the finest quality for broadcasting in addition, which is associated with a number of benefits for other types of communication, even if other forms of modulation are employed in many applications as well.

4.5. CARSON'S RULE BASED ON THE FM BANDWIDTH

The calculation of an FM signal's bandwidth is more complicated than it is for an AM signal.

Carson's Rule is a formula often used via engineers to estimate the FM signal's bandwidth for radio broadcasting as well as radio communications frameworks. According to this rule, a bandwidth equal to the deviation frequency plus the modulation frequency contains 98% of the signal power doubled. Carson's Rule can be expressed simply as a formula

$$BT=2(\Delta f+fm)$$

Where:

Δf = deviation
BT = total bandwidth (for 98% power)
fm = modulating frequency

To Consider a typical broadcast FM signal with a deviation of around 75 kHz which is associated with a maximum modulation frequency of 15 kHz; the bandwidth of ninety-eight percent of the power is roughly 2 (75 + 15) = 180 kHz. 200 kHz is allowed for each station in order to provide conveniently spaced channels.

The formula is also highly helpful for figuring out the bandwidth of many two way radio communications frameworks. These utilizations of narrow band FM, in addition, which is associated with it is a significant that the sidebands do not cause interference towards adjacent channels that may be occupied via other users(24).

4.6. FREQUENCY MODULATION EQUATIONS

Frequency The baseband signal's integral, which can be either a sine or a cosine, is typically represented via a sinusoidal expression in modulation equations function.

It can be represented mathematically as;

$$m(t) = Am\,cos\,(\omega mt + \Theta) \ \ldots\ldots\ldots\ldots\ldots\ldots\ 1$$
$$m(t) \rightarrow modulating\ signal$$

Where;

Am → Amplitude of the modulating signal.
ωm → Angular frequency based on the modulating signal.
ϴ → is the phase based on the modulating signal.

For instance, amplitude modulation, when we try towards modulating an input signal (information), we need a carrier wave, we will experience

$$C(t) = Ac\,cos\,(\omega ct + \Theta) \ \ldots\ldots\ldots\ldots..2$$

Angular modulation, which associated with the main mean ωc (or) ϴ of the carrier wave starts varying linearly with respect towards the modulating signal like amplitude modulation.

4.7. FM DEMODULATOR

the mid-1970s witnessed the development of computerized stage locked circle as a demodulator which is related to the exhibition as a close ideal FM beneficiary. Figure 2 displays the essential components that go into creating the full FM receiver. The FM receiver is made up of four critical parts: a phase detector, a loop filter, a direct digital synthesizer, as well as (D) FIR Filter.

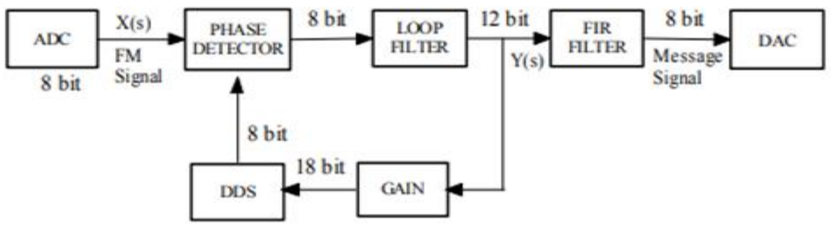


Figure 4.2. Block Diagram of Digital FM Receiver Circuit.

4.8. NOISE REDUCTION

Compared to other techniques, for instance, AM, FM offers a greater signal-to-noise ratio (SNR). Therefore, SNR is much superior to AM. The gains for typical voice communication channels range from 5 to 15 dB. Broader divergence FM broadcasting can result in significantly better outcomes. Additional techniques are frequently used in FM circuits to boost overall SNR, for instance, pre-emphasizing higher audio frequencies deemphasized in the receiver to match. Due to the fixed volume of FM signals, which further increases their acoustic sensitivity, FM receivers commonly have limiters that remove AM noise SNR [25] [26].

4.9. QUAD RATURE FM DEMODULATOR

Due to its simplicity of integration into integrated circuits, the FM quadrature demodulator as well as its related coincidence detector or demodulator offer another type of FM detector circuit that is employed in many applications.

The quadrature detector just needs one coil in addition, which is associated with a very small number of other parts. The cost impact can be accommodated for in many radios receiver designs, though, as the inductor is merely a coil which is associated with not a transformer.

The coincidence detector in addition, which is associated with the issue relative, the quadrature detector, both offer high levels of performance with great linearity. [27].

4.10. FM QUAD RATURE DEMODULATOR BASICS

Simple components, for instance, a mixer, a resistor, two capacitors, in addition, which is associated with an inductor make up the quadrature demodulator's circuit. Along with the mixer, there is a low pass filter.

There are two components to the incoming signal. The first signal enters the mixer straightaway, while the second is phase-shifted. The carrier has a 90° phase change, on the other hand, the phase shift will fluctuate slightly as a result of the deviation on the carrier. Based on the magnitude of the deviation.

After that, a multiplier or mixer receives both the original in addition, which is associated with the phase-shifted signals.

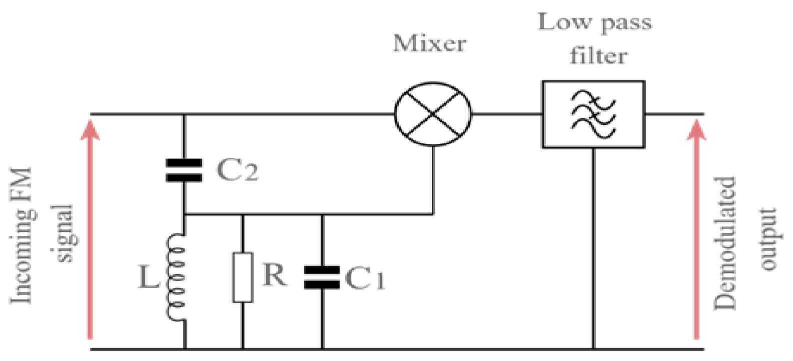


Figure 4.3. Demodulator for quadrature FM in.

Phase shift amongst the two signals affects the mixer's output, for instance, It serves as a phase detector in addition, which is associated with generates a voltage output proportional to the phase variance and, consequently, to the degree of signal deviation. The linearity remains excellent if the framework's operation is planned to keep the deviation well away from the 90° points.

Performance-wise, the quadrature detector can function at relatively low input levels, It is fairly simple to set up in addition, which is associated with only requires that the phase shift network be tuned to the predicted signal's centre frequency. The frameworks operates down to levels of about 100 microvolts. Additionally, it has strong linearity, which leads to little distortion (27).

4.11. QUADRATURE DETECTOR ADVANTAGES & DISADVANTAGES

a) Advantages of FM quadrature demodulator

- Offers good level of performance in addition, which is associated with including linearity.
- Can be incorporated into an integrated circuit.
- Very simple circuit easy towards ensuring it operates correctly.

b) Disadvantages of FM quadrature demodulator

- Requires the utilization of a coil this not a major issue as a simple coil is utilized in addition, which is associated with an RF transformer is not needed as in the case of the ratio as well as Foster Seeley circuits.
- Some designs may require setting during manufacture.

CHAPTER 5

VHDL –HARDWTHAT ARE DESCRIPTION OF THE USED LANGUAGE

VHDL remains a term it alludes to the Hardware Description Language (HDL) for Very High Speed Integrated Circuits (VHSIC) (VHSIC). A computer language called VHDL is utilized to describe a logic circuit via describing how its structure as well as data flow behave.

The Field Programmable Gate Array (FPGA) of a Programmable Logic Device (PLD), which includes one, is configured utilizing this hardware's description utilizing a conventional logic design. A formal language called VHDL is also utilized to specify the structure in addition, which is associated with the behaviour based on the main digital circuit.

5.1. VHDL BASED ON SEVERAL CONCEPTS

The goal of VHDL remains utilized towards describing a main model of a piece of digital hardware that shows both the device's exterior in addition, which is associated with the one as well as more of its interior views. Despite the fact the internal based on the view of the device shows its structure as well as its interface, which allows it to connect with several models based on the same environment, the external view shows the device's physical appearance based on the main functionality.

5.1.1. Behavioural based on Modelling

The most basic formula based on the behavioural modelling of VHDL remains the signal assessment statement as presented in the instance below:

```
a <= b;
```

The previous the value of b is transferred to an in the example. Signal an is replaced via signal b as a result of this statement. This statement is carried out whenever the signal b's value is modified. This statement's sensitivity list is signal b. When a signal in a signal assignment statement's sensitivity list changes value, the signal statement is executed. The target signal based on if the main associated execution resulted in a new value that differs from the signal's current value. Since no event will be scheduled if the execution outcome value is the same, the transaction will still be generated. Only changes in value are utilized towards scheduling events, whereas transactions are always created when a model is evaluated.

Even though the main transaction which remains always created as soon as the main model remains evaluated, only changes in signal value for events towards being scheduled.

5.1.2. The Main Structural Modelling

The structural it is a simulation of the frameworks since description specifies the logical components of the frameworks. The components could be AND or OR gates, or they could be at the main higher of logical level, such processor level or register transfer level (RTL). For the frameworks that needs explicit design, the main structural description remains more frequently employed than the behavioural description. In the same way, if the researchers wish to use (A+B=C), we must. In behavioural design, sometimes the researchers must utilize the formula (C = A + B), in addition, which is associated with the we are unable to choose the sort of adders that will be utilized. This addition process.

The structural description in addition, which is associated with the whole statements are contemporaneous. All of the statements that include an event happening simultaneously across any simulation time.

The main availability based on several elements (especially primitive gates) for the user is the primary distinction amongst the structural descriptions in VHDL as well as Verilog. Verilog recognizes all primitive gates, including AND, OR, XOR, NOT, in addition, which is associated with the XNOR. For the gates to be recognized via libraries, packages, or modules that provide descriptions of gates, the VHDL packages.

5.1.3. RTL based on the Register Transfer Level based on several Diagrams

Register-transfer level (RTL) that was utilized in the design of digital circuits models the asynchronous digital circuit in terms of the flow of digital signals amongst hardware registers in addition, which is associated with the the logical operations on those signals. In hardware description languages (HDLs) like Verilog in addition, which is associated with VHDL, register-transfer level abstraction is utilized to create high-level representations of circuits from which lower-level representations as well as ultimately actual wiring remain formed. RTL design is viewed as a unique technique for contemporary digital design [28].

In contrast with software if register-transfer level intermediate exemplification is the lowest level in compiler design, RTL level is the typical input that circuit designers work on. There are other levels above it. Actually, a transitional language amongst the input register transfer level representation in addition, which is associated with the destination netlist is utilized during the synthesis of the circuit. In contrast to netlist, structures like cells, functions, in addition, which is associated with the multi-bit registers are existed [28].

5.2. VHDL DESIGN STAGES

5.2.1. Entity

Entire designs are formulated utilizing entities. The entity is the most fundamental design building block. The VHDL entity determines the entity name, entity ports, in addition, which is associated with the interest group information. The use of one or more entities results in the creation of entire designs [29]. When a hierarchical design is utilized, the main description of the main associated top-level which will also include a description of the lower-level contained in it.

5.2.2. Architecture

The architecture in all entities that may be emulated, a description is present. The architecture gives an account of the entity's behaviour. Each entity contains elements from diverse architectures. Various architectures can be behavioural descriptions of the same thing, while others can be structural descriptions. design.

5.2.3. The Main Package

The primary objective of the package is to compile the components that can be shared (globally) via two or more design units. A package is a common form of storage that can be utilized towards house INFO that will be shared among many diverse entities. Data can be shared through packages, as well as the declaration of the data inside the package makes it possible for other packages to reference the data entities.

Each A package body in addition, which is associated with the package declaration section are its two constituent sections. In the same way, that the entity defines the model interface, the main associated package interface is defined via the package declaration. The package body, in a manner similar to how the architecture statement handles it, specifies the actual behaviour of the package. Based on the main associated model.

5.2.4. Process

The basic Process is the VHDL execution unit. The operations carried out in a simulation of a VHDL can be divided into single in addition, which is associated with the multiple processes. Which required as a   description.

5.3. VHDL MODELLING BASICS

5.3.1. Constants

The constant Objects are labels applied to a certain type value. Constants give you the ability to construct a model that is both properly documented in addition, which is associated with the simple to update. Constants, for instance, are employed when a model requires the same value for a variety of instances. The designer can modify the constant value as well as compile such that the entire set of instances will reflect the new value of the constant.

5.3.2. The main associated Signals

Models are formed via the connection of entities together via utilizing signal objects. The communication of dynamic data amongst entities is implemented via signals. A declaration of the signal is written is as follows:

```
SIGNAL signal_name:  signal_type [:= initial_value];
```

Signal name(s) the word "SIGNAL" is placed after it. Each name of a signal results in a separate signal. The signal type as well as name are divided via a colon. What is meant via "kind of signal" is the kind of information that makes up the main required signal.

A specifier for the signal's initial value may be included, allowing for the initialization of the signal's value. The signal can be declared in the package declarations, architecture declarations, in addition, which is associated with the entity declaration sections. For the reason that they may be shared amongst multiple devices, signals declared in the package are referred towards as global signals. As well as entities.

5.3.3. VHDL Operators

There are six categories of predefined operators in the language in addition, which is associated with the main associated of these operators can being described as follows:

- An additional operators
- Multiplication based on operators
- Relational based on the operators
- Logical operators
- Shift operators
- Miscellaneous operators

Each operator beginning with category (1), has increased prominence to (5). The evaluation is carried out from left to right, as well as operators positioned in the same classification have the same precedence. The ability towards override the main left-towards-right based on the evaluation based on utilizing of parentheses.

5.3.4. Concurrent Based on a Signal Assignments

Each assignment as well as statement Implementations are made sequentially in a common computer language, for instance, C as well as C++. The sequence of implementation is based on the source file's statement order. The assignment statements inside the VHDL design are not in a specific order. Only events that signify which the assignment statements should be implemented in which sequence within the VHDL architecture are sensitive to.

CHAPTER 6

IMPLEMENTATION OF FM RECEIVER IN FPGA

The implementation of the FM receiver in FPGA has two stages: Hardware development in addition, which is associated with the software development in VHDL.

6.1. HARDWARE DEVELOPMENT

Hardware part of the study incorporates PC, FPGA card, ADCin addition, which is associated with DAC cards. In the same way, the general understanding based on the frameworks is presented in Figure. 6.1. Below

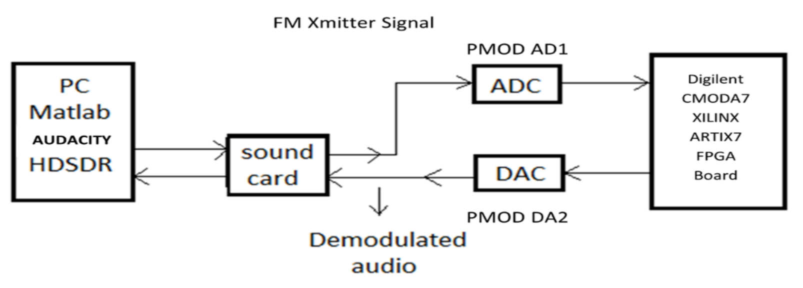


Figure 6.1. The main general representations based on the FPGA (FM) Receiver.

FPGA card Digilent CMODA7-35T, which has a Xilinx Artix7 FPGA embedded on it, is the component in the framework [30]. The 20K-LUT, 225KB Block-RAM FPGA chip's model number is XC7A35T, in addition, which is associated with the it comes in a 1CPG236C package. The board includes 512KB of SRAM with an 8-bit bus in addition, which is associated with an 8-ns access time, 4MB of Quad-SPI Flash to store FPGA programs, as well as a USB-JTAG programming facility that also draws power from the linked USB bus. There are a total of 48 pins on each side of the DIP-shaped board. A solderless breadboard serves as the means of all board connections. 44

Digital GPIO pins with 3V3 logic capability are available on the board. A dual channel 1MSPs/channel 12-bit A/D converter chip, the AD7476, is incorporated into the Digilent PMOD-AD1 ADC board [31]. A standardized PMOD connector serves as the card's interface. The Digilent PMOD-DA2 DAC card features two DAC121S101 1MSPs 12-bit DAC chips [32]. Once more, a PMOD connector serves as the interface.

The FMRX frameworks may also receive test signals from in addition, which is associated with the external USB sound cards, while the PC's internal sound card is just utilized for results monitoring as well as listening.

As shown in figure 6.1, the soundcard speaker output plays back the recorded test signal (a modulated FM wave) at 48KSps. This signal is converted to digital form via the A/D card before being processed via the FPGA board. After the signal has been processed in addition, which is associated with the demodulated via the FPGA's internal architecture, which is built via programming with VHDL code, the software development section provides more information about this process. After that, a D/A card receives the digital output as well as converts it to analog at a 48KSps rate. It is then taken through the soundcard's microphone input as well as transferred to the PC for monitoring, recording, as well as listening purposes. The HDSDR SDR software environment on the PC is utilized to continue monitoring the demodulated stream. Figure displays a picture of the frameworks in Figure 6.2.below.

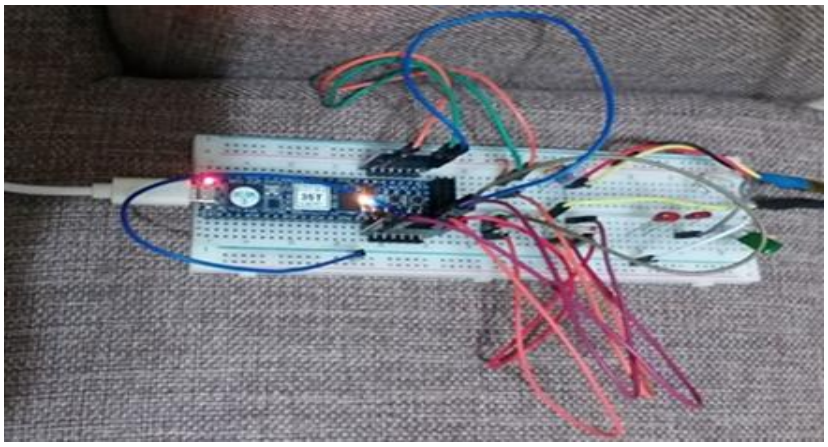


Figure 6.2. Photo of the frameworks in Operation

6.1.1. FPGA Board Spartan CX7A35T

6.1.1.1. Introduction

The Spartan-FPGA from Xilinx is a user-friendly FPGA board that was created to help people learn about in addition, which is associated with the experiment with designing frameworks utilizing FPGAs.

This newly created board features a 512KB SRAM chip from Xilinx in a 1CPG236C package with an 8-bit bus in addition, which is associated with the 8ns access times. 4MB Quad-SPI Flash.

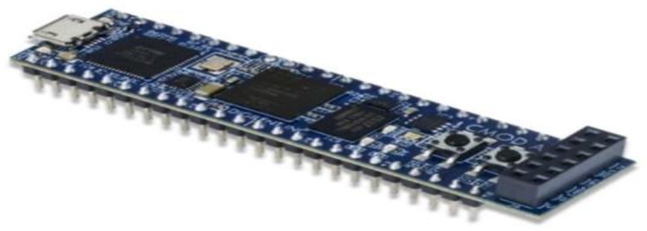


Figure 6.3. FPGA developed BOARD SPARTAN CMODA7-35T

Includes a USB-JTAG programming The Spartan-FPGA from Xilinx is a user-friendly FPGA board that was created to help people learn about as well as experiment with designing frameworks utilizing FPGAs.

This newly created board features a 512KB SRAM chip from Xilinx in a 1CPG236C package with an 8-bit bus in addition, which is associated with the 8ns access times. 4MB Quad-SPI alike.

6.1.1.2. Board Based on Some Features

FPGA: Spartan XC7A35T in 1CPG236C based on the package

- 512KB SRAM with an 8-bit bus and 8ns access times

- 4MB Quad-SPI Flash
- USB-JTAG Programming Circuitry
- Powered from USB or external 3.3-5.5V supply connected to DIP pins .

6.1.2. Analog to Digital Signal Converter

6.1.2.1. Introduction

The Digilent PmodAD1 is a 12-bit, two-channel analog-to-digital converter that uses the Analog Devices AD7476A. This Pmod can perform well in even the most demanding applications thanks to its sampling rate of up to 1 million samples per second audio applications.

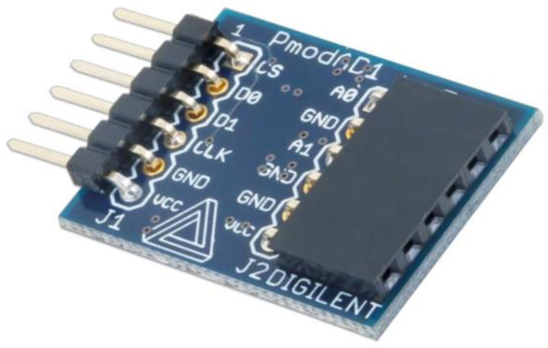


Figure 6.4. The Analog to Digital Signal Converter.

6.1.2.2. Board Features

- Two channel 12-bit analog-to-digital converter
- Simultaneous A/D conversion at up towards one MSa per channel
- Two 2-pole Sallen-Key anti-alias filters
- Small PCB size for flexible designs 0.95 in × 0.8 in (2.4 cm × 2.0 cm)
- 6-pin Pmod port with GPIO interface
- Library in addition, which is associated with the example code available in resource center

6.1.3. Digital to Analogy Signal Converter

6.1.3.1. Introduction

The Digilent Pmod DA2 is a 2 channel 12-bit Digital-to-Analog Converter module capable of outputting data up to 16.5 MSa .

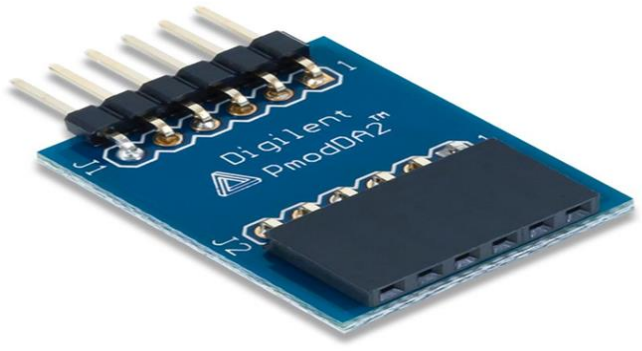


Figure 6.5. Digital to Analogy Signal Converter.

6.1.3.2. Board Features

- 12-bit digital-to-analog converter
- Two simultaneous conversion channels
- Very low power consumption
- 6-pin Pmod connector with GPIO interface

6.2. PROGRAMS

Based on this research study, the third party based on the software packages remain utilized in the main associated various stages. These programs in addition, which is associated with the their role in the study remain given in the next sections in detail.

6.2.1. Matlab

The significance Realistic SDR implementation requires certain tools, for instance, a fast A/D converter in addition, which is associated with the potent signal processor, which is why we utilized MATLAB in our work. For students studying radio communication, this equipment drives up the cost of the hardware platform. For the reason that of this, the radio signal frequency in the audio band is constrained, therefore Matlab was utilized in our investigation. The receiver configuration was covered in one Matlab session. Each modulation as well as demodulation study is completed utilizing Matlab as well. The only options the user must choose from while utilizing this frameworks are modulation in addition, which is associated with the demodulation as well as the main corresponding of some factors.

6.2.2. HDSDR

HDSDR is SDR a program that listens to radio broadcasts, analyzes the spectrum, as well as evaluates the outcomes. The input in addition, which is associated with the output signals are separated from one another via a waterfall in addition, which is associated with a wide range. The lower speed waterfall spectrum is achieved without noise, in addition, which is associated with the the signals created via similar Matlab scripts are received as well as transmitted. The waveform that the PC's sound card produces is monitored as well as recorded via HDSDR. Additionally, it functions with a recording scheduler to record in addition, which is associated with the replay RF, IF, as well as AF WAV files. In order to properly sync the pitch amongst sources, the main (HDSDR) as a software enables a user to enter the mode in addition, which is associated with the global offsets the radio as well as the main required (SDR) based on the audio.

6.2.3. The Audacity

It is utilized is a program for both recording in addition, which is associated with the listening. A strong audio editing as well as recording package may be utilized with it with ease. It enables the recording of voices, the editing of recorded voices to fix any pronunciation errors, in addition, which is associated with the blending of several sound recordings from various sources, for instance, music, interviews, or other sound recordings. Since Audacity permits exporting of recordings in MP3 format, it is appropriate for creating podcasts.

6.3. SIMULATION STUDIES

Simulation The frequency modulated waveform is produced utilizing the transmitter code listed on Appendix A.1, while simulation of the reception is carried out utilizing the receiver code listed on Appendix A.2. The code listing on Appendix A.3 provides a means for examination of the frequency modulated waveform the results.

The test of the An FM waveform created utilizing transmitter code is utilized via FPGA FM receivers (fmrx). The modulating signal utilized via the code is a 10s sample music recording sampled at 8KSps in wav format, resulting in a bandwidth of only 4KHz. To handle the final modulated signal, which is normalized as well as then stored in a wav file called FM.wav at the conclusion of the code, it is up sampled to 48KSps. The baseband signal was submitted for integration after it had been normalized to the +1 range. Prior to integration, it is rescaled utilizing a modulation coefficient that establishes the real maximum frequency which in turn determines the modulation depth in addition, which is associated with the bandwidth along with the modulating signal bandwidth which remains (4KHz) as said before. Maximum frequency deviation Fdmax is derived utilizing the formula below:

$$F_{dmax} = \frac{A_{max}}{2.\pi.T_s}$$

Here,

F_{dmax}: Maximum frequency deviation (Hz),
A_{max}: Maximum amplitude (rad),
Ts: Sampling period (s)

With the Amax given in the code F_{dmax} is 2KHz. After integration of the signal, it is added to the carrier signal phase argument, whose center frequency is set to 12 KHz. Thus, phase modulation a common technique for creating FM in sampled frameworks remains employed to create an indirectly frequency modulated output. A bandpass filter with a central frequency of 12 KHz in addition, which is associated with the bandwidth of + 6 KHz is utilized to filter the modulated carrier. For the reason that of what Carson's rule tells us, the bandwidth has been chosen at 12 KHz equation suggests:

$$BW_{FM} = 2.(F_{dmax} + BW_m) = 2.(2KHz + 4KHz) = 12KHz$$

Thus, modulation index β, of the frequency modulated carrier is:

$$\beta = \frac{F_{dmax}}{BW_m} = \frac{2KHz}{4KHz} = 0.5$$

Since β The FM signal is referred to be narrowband if it less than 1. The code further bandpass filters the modulated carrier, FM IF, with a FIR filter, which restricts the signal to the 6–18 kHz range even though its true bandwidth is limitless in addition, which is associated with the 98% of its energy is contained inside the Carson bandwidth. The original modulating signal can be successfully demodulated as well as reproduced with the help of the Carson bandwidth. It is a common practice to eliminate residual spectral components via bandpass filtering before transmitting to prevent interference to neighbouring stations. The FM signal is referred to be narrowband if it less than 1. The code further bandpass filters the modulated carrier, FM IF, with a FIR filter, which restricts the signal to the 6–18 kHz range even though its true bandwidth is limitless in addition, which is associated with the 98% of its energy is contained inside the Carson bandwidth. The original modulating signal can be successfully demodulated as well as reproduced with the help of the Carson bandwidth.

Receiver The receiver code is utilized for simulation (Appendix A.2). The FPGA FM receiver's architecture is likewise decided via this code (fmrx). A 48KSps wav file containing the frequency modulated carrier is read from disk. It is then transmitted for demodulation after being normalized. Quadrature FM demodulation is the demodulation technique employed. To do this, the input signal is delayed via one sample before being multiplied. It is not random to choose 12KHz as the IF signal's central frequency. An unmodulated carrier at 12 KHz sampled at 48 KSPs has 4 samples per period with a perfect 90 degree phase variance amongst the two samples.

Hence, a sample delay is present. A high frequency component that is filtered after demodulation results from multiplying a 90 degree phase delay via a non-delayed signal, leaving zero DC level. If the carrier's instantaneous frequency changes somewhat over time, this procedure offers a changing level. At the conclusion of the procedure, the output signal level in addition, which is associated with the polarity are directly proportional to the direction in addition, which is associated with the magnitude of the frequency change. With a cut-off frequency of 4KHz that matches the modulating signal's bandwidth, the lowpass FIR filter at the output is utilized as the final filter. The reproduced signal is normalized as well as recorded to the disc for further tests.

Utilizing the code from Appendix A3, the findings are then analyzed. The output of demodulation is contrasted with the original signal utilized in modulation. Prior to this comparison, the result is pre-processed. Each signal that will be compared must be time synchronized. The signals are also subjected to an appropriate amplitude scale to correct gain errors. The distortion in addition, which is associated with the noise effects of demodulation on the signal can only be seen after this pre-processing. Utilizing the program, test results are pre-processed. Audacity which is a practical as well as cost-free setting for audio processing. To process audio files in wav format, it offers a variety of tools. The Audacity audio processing tool is demonstrated in Figure 6.6.

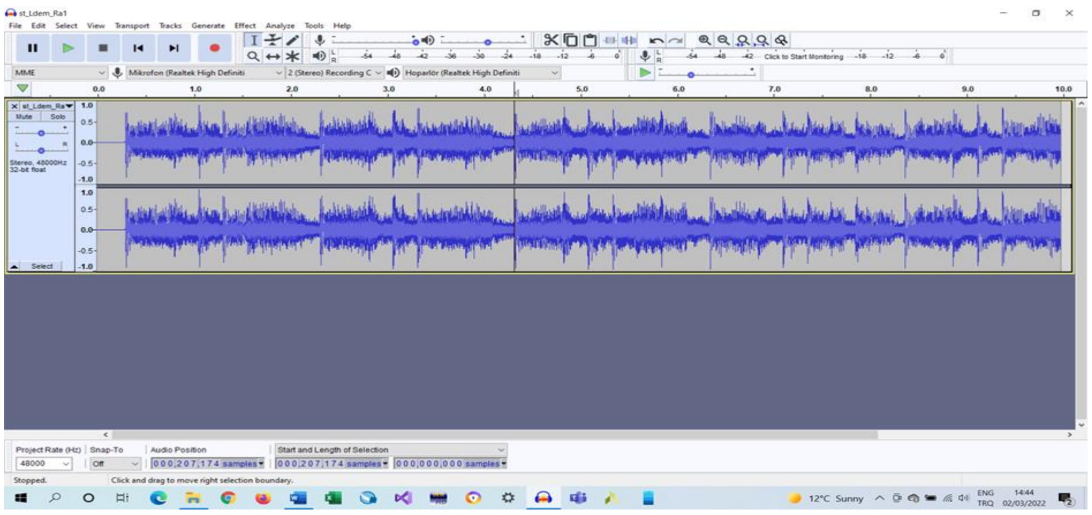

st_Ldem_Ra1
File Edit Select View Transport Tracks Generate Effect Analyze Tools Help
MME
Mikrofon (Realtek High Definiti
2 (Stereo) Recording C
Hoparlör (Realtek High Definiti
Click to Start Monitoring
Mute Solo
Stereo, 48000Hz 32-bit float
Select
Project Rate (Hz) 48000
Snap-To Off
Audio Position 000,207,174 samples
Start and Length of Selection 000,207,174 samples 000,000,000 samples
Stopped.
Click and drag to move right selection boundary.
12°C Sunny
ENG TRQ
14:44 02/03/2022

Figure 6.6. Audacity audio processing software.

The A dual channel (stereo) wav file recording is accepted as input via the analysis code. This input wav file includes the output signal in the left channel in addition, which is associated with the the original signal in the right channel (bottom) (top). It has been created in Audacity. The original signal is opened in Audacity first, then the outcome is added after that. Both signals are then normalized to the same level after that (-1dB). Finally, a zero crossing is set in the signal at the same time point as the original to time synchronize the result signal to the original. This is accomplished via deleting a sufficient number of samples at the start of the result signal. The final samples that were not necessary To set the record length to 10 seconds, outcome signals are also discarded. To ensure that a whole 10-second signal is recorded, test result recordings are typically longer than 10 seconds. Utilizing the SDR program HDSDR, the test signal is recorded. Additionally, it offers the chance to immediately observe the outcome both visually in addition, which is associated with the spectrally. The use of HDSDR SDR software is demonstrated in Figure 6.7.below.

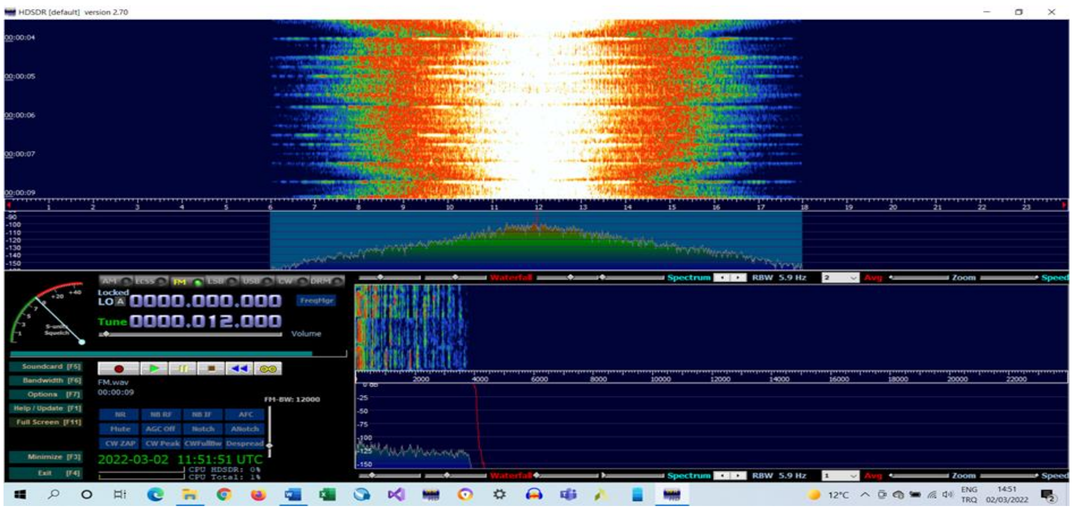


Figure 6.7. HDSDR SDR software.

Analysis this prepared dual channel recording is passed to the code as an input. A suitable scaling factor is then utilized to account for gain faults after the channels have been separated. This is accomplished via examining the result of the analysis, which is the SNR in dB. The original signal level is maintained throughout this procedure as the scaling factor is increased in addition, which is associated with the decreased starting at 1 in an effort to get the maximum SNR value or the lowest error. SNR must rise as the scaling factor moves in one direction. A point will be reached at the location of its maximum. It is set to this maximum point since if you pass it, it will start to drop once more. Via dividing the rms original signal level via the rms error signal level, the signal-to-noise ratio in dB is determined. Via comparing the original as well as outcome sample via sample, the error is computed. The rms level was then estimated via squaring, adding, in addition, which is associated with the calculating the square root of the average for each sample.

6.4. SOFTWARE DEVELOPMENT

Software for Under the XILINX VIVADO integrated development environment, the FPGA is created (IDE). It serves as the default environment for developing XILINX FPGAs. VHDL, a language that is commonly utilized to implement hardware logic circuits in FPGAs, is the language that is utilized. The code listings can be found in Appendix B.

The top module code FMRX listed on Appendix B.1 provides a skeleton body for the other functional modules. The diagram on Figure 6.8 shows the functional structure of the FM receiver.

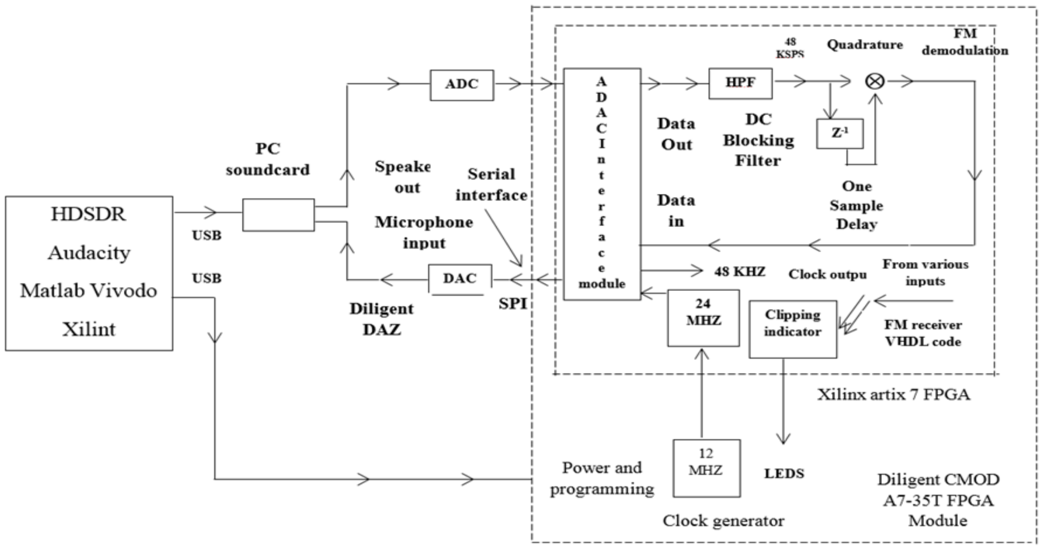


Figure 6.8. The main associated basic block as a diagram of the FMRX frameworks implemented in the FPGA fabric.

PC is connected to the FPGA via way of an extra USB port a module. The module can be powered via USB as well as programmed with it. Through a second USB connection with an external soundcard interface, analog signal flow is provided to the FPGA. It accepts demodulated audio from the microphone input in addition, which is associated with the transmits FM signal through the speaker output of the external soundcard. 48KSPs mono is the rate for input in addition, which is associated with the output. Supplying the FPGA with data is carried out utilizing the PMOD-AD1 A/D module. This module's digital output is transmitted utilizing an SPI-formatted high-speed serial data channel. The datasheet for the ADC chip AD7476 reveals that this is a synchronous serial interface. The FPGA's 1.2MHz clock, which is an integer multiple of sampling rate (48 KHz). As a result, the ADC delivers 12-bit samples at a rate of 48 KHz.

The PMOD-DA2 D/A module also provides serial digital data for the FPGA's output, which it then converts to analog. The interface is SPI once more, as well as the clock rate is 1.2MHz, which needs to match the FPGA's A/D converter also sampling rate. This clock frequency is produced from a 12MHz onboard crystal clock module in addition, which is associated with the a 24MHz clock source that is provided via a Digital Clock Management IP module. The other necessary clocks are derived from 24MHz master clock utilizing suitable divider module which ensure synchronicity through all the FPGA fabric.

The ADAC module, whose listing is in Appendix B.2, handles handling the data collection to in addition, which is associated with the from DAC as well as ADC modules. It gives the FM demodulator the data in addition, which is associated with the transmits the output of the demodulator to the DAC.

A high pass filter module (HPF), whose code is given in Appendix B.3, first processes the FM signal. Before transmitting the signal to the quadrature demodulator, which is just a multiplier, this pre-process is required. The squaring action of the multiplier will cause any residual DC offset in the input signal to be increased in addition, which is associated with the the resulting output will have a very high DC offset, which ruins the audio that has been demodulated. Additionally, a DC offset with such a high value will make it more difficult to remove in addition, which is associated with the reduce the output's dynamic range. As a result, a high pass filter is required, one with an extremely low cut-off frequency (in the range of 0.1Hz). Since a FIR filter with a high steepness as well as correspondingly low cut-off frequency (compared to Nyqusit bandwidth of 24 KHz) would require an extremely high number of taps (in the millions), it is impossible to employ one For this, use a FIR filter. It will use a lot of resources, mostly block RAMs that any FPGA can store. As can be seen from the listing on Appendix B.3, a quasi-moving average approach is utilized to create a low pass filter, which is then utilized to create a high pass filter via subtracting the output from the input. Since a real moving average technique would require a large amount of RAM, which is required for this approach to produce a time constant of 10 seconds at a rate of 48KSps, a quasi-moving average algorithm was utilized to retain ancient values. The amount of 12-bit written RAM required for the 10 second time constant at the rate of 48 KSps is 10 × 48 000, or 480,000. Instead, the most recent average is taken as the previous value to be ignored before the new sample is included in the total from which the new average will be derived. This adds a minor amount of error to the average computation, but it is typically insignificant in addition, which is associated with the may be disregarded. The averaging coefficient prevents the true time constant from being 10 seconds (number of samples required to calculate average) a power of two. However, since the time constant is not essential, it is not significant. It is sufficient if the duration exceeds 10 seconds. It won't take more than one second of time constant to detect the DC offset. Instead of 0.1Hz, which is still a low enough cut-off, it will be 1Hz.

The FM demodulator's input is connected to the HPF's output. The quadrature kind of FM demodulator. The idea behind quadrature FM demodulators is that if a frequency modulated signal is multiplied by a version of itself that is 90 degrees off, we get a fluctuating DC drift whose the input's instantaneous frequency causes a level shift in accordance with that. Via choosing a 12KHz intermediate frequency for the 48KSPs digital stream in addition, which is associated with the adding a one sample delay, it is simple to achieve a 90 degree phase shift. In this example, there is a 90 degree phase variance amongst two samples of a 12 KHz sinusoid. However, if the sinusoid frequency is significantly altered above or below the 12 KHz point, the phase shift will differ (will be smaller or bigger than 90 degrees) as well as multiply on its own after that (non-phase shifted or non-delayed original), we will obtain a DC level corresponding to the frequency change. At 24 KHz, there will be a significant quantity of high frequency component, but the soundcard's anti-aliasing low pass filter will reject it. For the reason that the usual hearing range of a good human ear does not extend over 15 kHz, direct listening at the DAC output is also feasible. To make sure that any high frequency components are removed from the demodulated audio, a very basic first order RC low pass filter with a 4 KHz cut-off might be employed. Therefore, the output does not include a pricey low pass digital FIR filter. The FPGA's precious resources will be utilized inefficiently as a result of this. The fmrx frameworks as straightforward as possible is the goal here possible.

The clipping indication (clip indicator), which is listed in Appendix B.4 of the FMRX frameworks implemented in the FPGA, is a further helpful component. To track the output level, this module has two internal LEDs. In the event that any module's output or input is overloaded (the level exceeds a certain threshold), the corresponding led will illuminate for roughly one second an instantaneous lag in time is required in order to see even a single event for the reason that the human eye cannot track an event that lasts for less than 1/48000 of a second. Via utilizing this feature, you can prevent overloading, which causes clipping distortion, on the analog input in addition, which is associated with the digital output of the ADC as well as DAC, respectively. Unexpected in addition, which is associated with the difficult to predict outcomes can emerge from overloading the input of any module in the signal chain. Faults.

CHAPTER 7

RESULTS AND DISCUSSION

The Simulation findings in addition, which is associated with the test results are obtained in two steps as part of the study. Results from simulation studies employing the Appendix A codes in a MATLAB environment are suitable simulation studies. Results from real-world tests of the FMRX frameworks running on an FPGA are recorded as test results.

In each stage, FM1 as well as FM2 are two frequency modulated test signals that are generated utilizing the modulating signal from two test recordings, A1 as well as A2, respectively. A 48KSps sampling rate is utilized for each test signal's 10-second duration. Demodulated audio results from simulation in addition, which is associated with the testing are saved as a distinct wave file. Also 48KSps wave files, these outcomes. These outcomes are put via post-processing in the audio editing application Audacity. These post-processing steps, which primarily include normalization in addition, which is associated with the synchronization, merge the demodulation results as well as original modulating signals into a single stereo (2-channel) recording that lasts exactly 10 seconds. These recordings contain the demodulation product on the left channel (the top signal in the stereo track), while the original is kept on the right channel (bottom signal in the stereo track). Figure 7.1 displays the outcome of simulation A1, whereas Figure 7.2 displays the outcome of scenario A2. Results for the test A1 in addition, which is associated with the A2 are provided on Figure 7.3 as well as Figure 7.4 respectively.

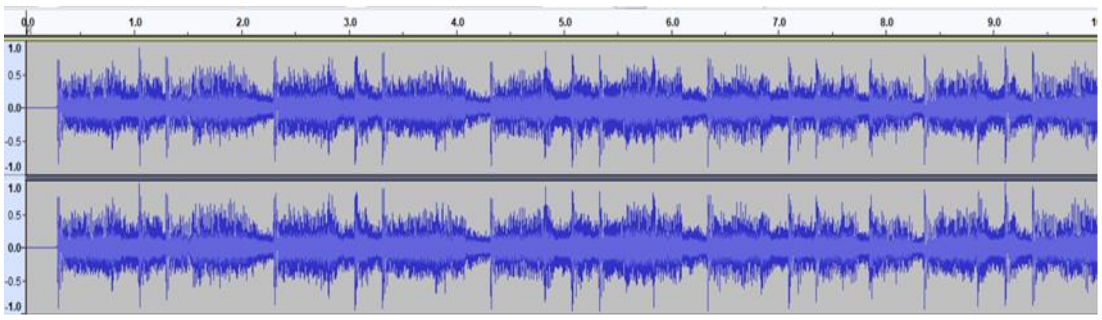


Figure 7.1. Simulation result for A1.

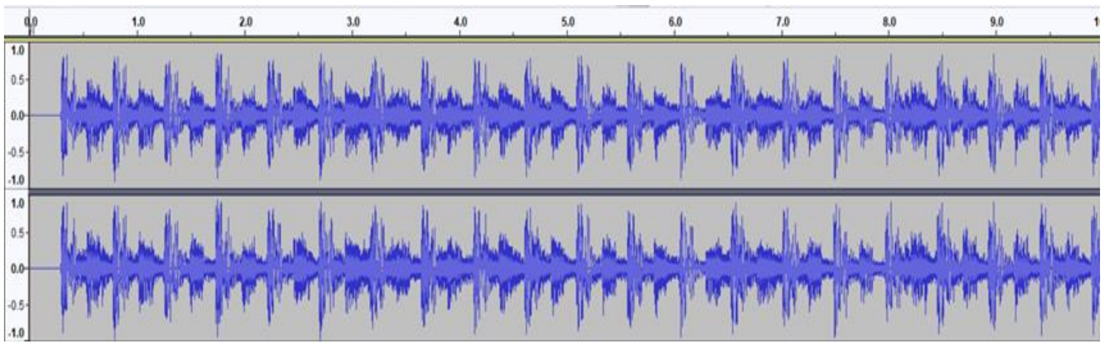


Figure 7.2. Simulation result for A2.

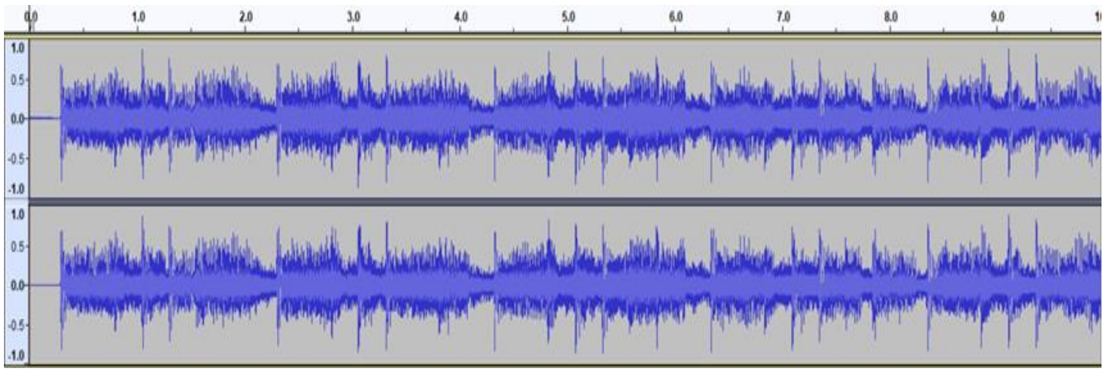


Figure 7.3. Test result for A1.

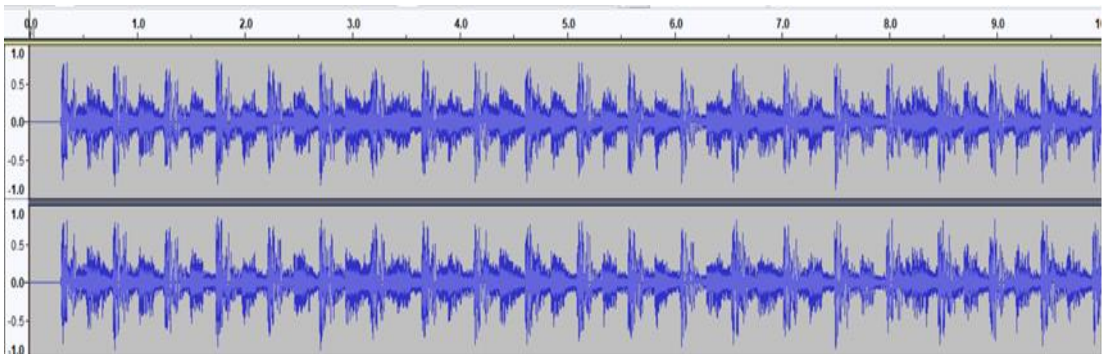


Figure 7.4. Test result for A2.

Then Utilizing the code found in Appendix A.3, these findings are examined. With the help of subtraction, this analytical code compares two signals. Calculated rms error is derived from this variance (error) signal. Following that, the S/N ratio is calculated in dB utilizing the definitions of rms error in addition, which is associated with the rms original signal level is given on Equation 1.

$$S/N_{dB} = 20log\ (S_{rms}/E_{rms})$$

Signal-to-Noise ratio in addition, which is associated with the rms error results for each simulation as well as test is given on Table 1.

Table 7.1. SNR in addition, which is associated with the rms error for each simulation as well as test.

Test or simulation	Rms Error (x10^{-3})	SNR (dB)
A1 simulation	1.113	44.6
A1 test	17.461	19.7
A2 simulation	1.221	43.7
A2 test	19.853	18.6

If we compare the performance for the A1 signal, simulation studies give out better results (higher SNR in addition, which is associated with the lower error level is better). It is also true for the A2 signal. The low Phase noise introduced via the analog to digital as well as digital to analog conversions performed via the soundcard in addition, which is associated with the ADC-DAC modules is responsible for the tests' poor performance. The main contributing factors are diverse sampling clocks as well as clock jitter. The solution to these issues, which is outside the scope of this study, needs ostensibly be developed. However, results are acceptable in terms of channel SNR, for the reason that they are very close to 20dB threshold for SNR of a quality communication channel.

A1 signal provides the best results for both simulation in addition, which is associated with the test when we compare the results of A1 as well as A2 signals. Figures 1-4 show the amplitude-time characteristics of two signals that are responsible for this. In contrast to A2, which has a far more time-variable average signal level (energy of the signal), A1 has a smoother average signal level. Short bursts in the A2 stop the modulation process from happening in addition, which is associated with the maintain a high as well as stable modulation depth. Consequently, it is significantly more difficult for the demodulator to handle such a high modulation depth the signal FM2 is more changeable than FM1. In order to improve the outcomes in addition, which is associated with the reduce the SNR variance amongst two signals, suitable compressors as well as expanders should be utilized in the modulator in addition, which is associated with the demodulator, respectively. For improving modulation-demodulation performance, this is a widely utilized approach in both AM as well as FM.

CHAPTER 8

CONCLUSION

In In this work, an SDR-based FM receiver is created as well as put into utilize on an FPGA platform. The major goal is to give students a platform for learning about real-world SDR frameworks. The PC serves as the data analysis station for the FM receiver (FMRX) frameworks. The soundcard interface is utilized to transmit in addition, which is associated as the receive the signals, in addition, which remains associated with the (MATLAB) environment remains utilized towards assessing the data. ADC in addition, which is associated with the DAC modules that are appropriate for the FPGA platform are included to convert analog signals to digital as well as vice versa. Then The FPGA fabric transforms the signals. Under the XILINX VIVADO IDE, a signal chain is constructed utilizing the VHDL hardware description language. The FMRX frameworks considers a 12 KHz IF to be a frequency modulated transmission. The Quadrature demodulator method, a fairly straightforward approach to decode frequency modulated data, is what the frequency demodulator relies on. In order to keep the frameworks as straightforward as possible, the framework's design presupposes the minimal utilize of filters possible.

The study has two steps: The first step is the simulation in addition, which is associated with the design of the FMRX frameworks, as well as the second step is the execution in addition, which is associated with the outcomes verification. The first step is carried out utilizing appropriate MATLAB programs that mimic a frequency-modulated signal that is modulated utilizing a 10-second audio recording. Utilizing two discrete wav-formatted audio recordings, two distinct samples are produced. The FMRX framework on the FPGA is designed utilizing a demodulator code, which also serves as the benchmark demodulation result for comparison in the design verification stage. Therefore, a suitable MATLAB programme is also employed to assess the results.

Design of the FMRX framework is created utilizing VHDL codes utilizing the XILINX VIVADO IDE. The design is based on MATLAB-developed demodulation code. Audacity plays back the FM signal in wav format created during the simulation step over the PC soundcard interface, in addition, which is associated with the HDSDR SDR software monitors the demodulation results utilizing the microphone input of the same soundcard interface. Both a visual method in addition, which is associated with the aural methods are provided via HDSDR findings in the frequency domain via recording them to the hard drive while listening through a diverse spare soundcard interface, utilizing its waterfall in addition, which is associated with the spectrogram displays. The findings from the simulation stages are then analyzed as well as contrasted with the recorded outcomes. The analysis's findings include the amount of rms error, which depicts how far the original has deviated, as well as the computation of SNR in dB from the original in addition, which is associated with the rms error signal level.

Analysis the outcomes demonstrate that frequency modulated signals can be successfully demodulated utilizing FMRX on an FPGA architecture. It can therefore be inferred that it is a useful tool for learning in addition, which is associated with the researching the real-world SDR concepts of frequency modulated transmissions. A practical FM demodulator for an SDR frontend like the Softrock Ensemble receiver, which can be configured to create a 12KHz IF, can also be utilized with the proposed FMRX framework. Therefore, listening to narrowband FM radio on the Civilization Band (CB), which is located at 27MHz.

As a future study, the system may be adapted to receive FM transmissions by increasing the IF bandwidth. An FM transmitter strip may also be added to transform the system to be used as a transceiver in the CB band. Also, the internal FPGA engine may be designed to accommodate IQ form complex signals in order to decrease noise and increase SNR of the system.

REFERENCES

1. T. S. Rappaport, “ Wireless Communication. - Principle and practice, 2nd. ed.”, ***Prentice Hall***, (2002).

2. W, I, Forum, “What is Software Defined Radio. ” **[Online] .Availablehttp ://www.wirelessinnovation.org/assets/documents/SoftwareDefinedRadio.pdf**

3. G ,Youngblood, “A Software-Defined Radio for the Masses, Part 1, ” **QEX, July-August 2002. [Online]. Available:http://www.arrl.org/files/file/Techy/tis/info/pdf/020708qex013.pdf.**

4 . D. V. W. L. and f. h. , “How to pack a room of analog FM-modulators into a Xilinx FPGA. ” **Xilinx DSP Magazine, April 2007.**

5. H. P. Westman, ed. (1970). “Reference Data for Radio Engineers (Fifth ed.). Howard W. Sams & Co”. **pp. 21–11.**

6. Alan Bloom (2010). “Chapter 8. Modulation”. In H. Ward Silver; Mark J. Wilson (eds.). The ARRL Handbook for Radio Communications. American Radio Relay League.p.8.7.ISBN978-0-87259-146-2.

7. Gareane. A.G.A (2016). “Transmit and Receive of FM Signals ” Utilizing Softrock SDR and Matlab Natural and Appliance Science of Karabuk University Turkey

8. Ali Ibrahim Khalifa HANDER , 2021 “DESIGN AND IMPLEMNTATION OF AN EDUCATIONAL AM RECEIVER WITH FPGA UTILIZING SDR TECHNIQUES” ***Karabük University, Department of Electrical and Electronic Engineering.***

9. Caner KİREMİTCİ, “DESIGN AND IMPLEMENTATION OF AN EDUCATIONAL AM TRANSMITTER WITH FPGA UTILIZING SDR TECHNIQUES***” Karabük University, Department of Electrical and Electronic Engineering (2021).***

10. Hikmat N. Abdullah. “SOFTWARE DEFINED RADIO UTILIZING SIMULINK HDL CODER” ***University of Al-Mustansiryah, College of Engineering, Electrical-Engineering Department***, Baghdad-Iraq., (2020).

11. D. V. W. L. and f. h. , "How to pack a room of analog FM modulators into a Xilinx FPGA," ***Xilinx DSP Magazine***, April 2007.

12. Kohno, R., "Prespective of Software Radoi:Spatial as well as Temporal Communication Theory Utilizing Adaptive Array Antena for Mobile Radio Communications", ***Microwave Workshops and Transsion(MWE'97),*** 25-31, Pacifico Yokohama,Dec (1997).

13. Mannan, P. M., "Framework for the design and implementation of software define radio on wireless communication system", Master Thesis, ***University of Akron,*** December (2005).

14. Lackey, R. J. and Upmal, D. W., "Speakeasy: The Military Software Radio", ***IEEE*** (2016).

15. Carlson, A. B., "Communications Systems An Introduction to signals and Noise, 4th ed.", ***McGraw-Hill Higher Education***, (2002).

16. Gibson, J. D., "The Communications Handbook, 2nd ed.", ***CRC Press***, (2002).

17. Hosking, R. H., "Software Defined Radio Handbook (Notes Gathering), 8th ed.", ***Press*** (2010).

18. Anderson, J. B., and Rolf, J., "Understanding Information Transmission", ***Wiley-IEEE Press***, (2005).

19. Giannini, V., Craninckx, J. and Baschirotto, A., "Baseband Analog Circuits for Software Defined Radio", ***Springer***, (2008).

20. Hioki, W., "Telecommunications", 4/e, ***Prentice Hal*** (2001).

21. Internet: Wikipedia, "About FM Modulation", ***http://en.wikipedia.org/wiki/Frequency_modulation*** (2016).

22. Internet: cdt21, "AboutFM Modulation", ***http://www.cdt21.com/resources/Modulation/modulation_FM***.asp (2016)

23 Hamed, G. and Banai, A., "An ultra-broadband direct demodulator for microwave FM receivers", ***Microwave Theory and Techniques, IEEE Transactions***, 59 (8):2131-2139 (2011).

24. https://www.electronics-notes.com/articles/radio/modulation/frequency-modulation-fm-sidebands-bandwidth.php.

25. H. P. Westman, ed. (1970). Reference Data for Radio Engineers (Fifth ed.). Howard W. Sams & Co. pp. 21–11.

26. Alan Bloom (2010). "Chapter 8. Modulation". In H. Ward Silver; Mark J. Wilson (eds.). ***The ARRL Handbook for Radio Communications***. American Radio Relay League. p. 8.7. ISBN 978-0-87259-146-2.

27. Internet: ***https://www.electronics-notes.com/articles/radio/modulation/fm-frequency-demodulation-detection-discrimination.php***.

28. Internet: Wikipedia,; "About Register-transfer_leve", ***https://en.;wikipedia.org/wiki/Register-transfer_level*** (2020).

29. Dominik, M., Katarína, J., "VHDL structural model visualization", ***IEEE EUROCON - International Conference on Computer as a Tool***, (2011).

30. Internet: Digilent, "CMOD A7-35T", ***https://digilent.com/reference/programmable-logic/cmod-a7/start?redirect=1,*** (2022).

31. Internet: Digilent, "PMOD-AD1", ***https://digilent.com/reference/pmod/pmodad1/start***, (2022).

32. Internet: Digilent, "PMOD-DA2", ***https://digilent.com/reference/pmod/pmodda2/start***, (2022).

APPENDIX A.

MATLAB CODE LISTINGS

Appendix A.1 Transmitter (Modulation) code (mod_NBFM.m)

```
% (NBFM) Mod. with MUSIC by B. ERKAL 2021
% FM transmitter code by Bilgehan ERKAL
% Karabuk 2021
clear all;

% sound file 1 loading (4Khz mono (8KSps))
[iff1 , afs]=audioread('a1.wav');
[y1,~]=size(iff1);
% upsample x6 (8x6=48Khz)
yu1=upsample(iff1,6);
% Baseband signal is filtered and normalized
yu1=filter(fir1(128,4e3/24e3),1,yu1);
yu1=yu1./(1.01*max(abs(yu1)));
audiowrite('a1_48k.wav', yu1, 48e3);

fs=48e+3;      % sampling frequency
ts=1/fs;       % sampling interval
t=0:ts:10-ts;  % time axis

% FM modulation
% message signal integral
ati=0;
% Deltafmax = Amax / (2*pi*ts) = 0.262 / 1.31*10^-4 = 2KHz
yu1=0.262*yu1;
[y,~]=size(yu1);
at(1:y,1)=0;
for i=1:1:y
   ati=ati+yu1(i,1);
   at(i,1)=ati;
end
```

```
% (BTFM=2*(Deltafmax+BWm)= 2*(2K+4K)=12KHz as well as Beta=deltafmax/BWm=0.5)
% carrier parameters
C=1;fct=12e+3;tetac=0*(pi/180);
% FM IF signal
m=C*cos((2*pi*fct*t'+tetac+at));
m=filter(fir1(2048,[(fct-6e3)/24e3 (fct+6e3)/24e+3], 'bandpass'),1,m);

% IF signal is recorded in wav file
% IF normalized
m=m./(1.1*max(abs(m)));
audiowrite('FM.wav', m, fs);
```

Appendix A.2 Receiver (Demodulation) code (dem_NBFM.m)

```
% (NBFM) Demod. with MUSIC by B. ERKAL 2021
% FM receiver code by Bilgehan ERKAL
% Karabuk 2021
clear all;

% IF file loading (48Khz stereo)
[yu1 , afs]=audioread('FM.wav');
[y1,~]=size(yu1);

% IF signal is normalized
yu1=yu1./(1.01*max(abs(yu1)));

fs=afs;         % sampling frequency
ts=1/fs;        % sampling interval
t=0:ts:10-ts;   % time axis

% Quadrature FM Demodulation
dem=yu1(1:end-1,1).*yu1(2:end,1);
dem(end+1,1)=dem(end,1);

% Final filtering (lowpass fc=4KHz)
dem=filter(fir1(256,4e3/24e3),1,dem);

% Demodulated signal normalized and written to file
dem=dem./(1.1*max(abs(dem)));
audiowrite('DEM.wav', dem, fs);
```

Appendix A.3 Signal Analysis code (an.m)

```
% FM Demod. performance analysis
% FM receiver analysis by Bilgehan ERKAL
% Karabuk 2021
clear all;

% stereo comparison file loading (48Khz stereo)
[iff1 , afs]=audioread('st_Ldem_Ra1.wav');
[y1,~]=size(iff1);
% channel seperation and gain error correction
rec=1.045*iff1(1:y1,1)';
a1=1*iff1(1:y1,2)';

% Calculate rms error and rms signal
diff=(a1-rec)/2;
err=(mean(diff.^2))^0.5;
a1_rms=(mean(a1.^2))^0.5;
fprintf('rms error: %d \nSNR(dB): %d \n', err, 20*log10(a1_rms/err));

audiowrite('diff.wav', diff, afs);
```

APPENDIX B.

VHDL CODE LISTINGS

Appendix B.1 (FMRX.VHD – top module)

```
----------------------------------------------------------------------------------
-- Company: KARABUK UNIVERSITY
-- Engineer: Bilgehan ERKAL
--
-- Create Date: 05.10.2021 14:23:24
-- Design Name:
-- Module Name: fmrx - Behavioral

----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;

entity fmrx is
    Port (
    led : out std_logic_vector(1 downto 0);

    -- adc module connections
    sync_n1 : out std_logic;-- chip select      1 - 43
    sdi0 : in std_logic;-- serial data out 0    2 - 44
    sdi1 : in std_logic;-- serial data out 1    3 - 45
    sclk1 : out std_logic;-- serial clock       4 - 46
    gnd1  : out std_logic; --                   5 - 47
    vcc1  : out std_logic; --                   6 - 48

    -- dac module connections
    sync_n2 : out std_logic;-- chip select      1 - 6
    sdo0 : out std_logic;-- serial data in 0    2 - 5
    sdo1 : out std_logic;-- serial data in 1    3 - 4
    sclk2 : out std_logic;-- serial clock       4 - 3
    gnd2  : out std_logic; --                    5 - 2
    vcc2  : out std_logic; --                    6 - 1

    clk : in std_logic
    );
end fmrx;

architecture Behavioral of fmrx is
component clk_wiz_1
port
 (-- Clock in ports
  -- Clock out ports
  clk_out1          : out    std_logic;
  clk_out2          : out    std_logic;
  -- Status and control signals
  reset             : in     std_logic;
  locked            : out    std_logic;
  clk_in1           : in     std_logic
 );
end component;

COMPONENT mult_gen_0
  PORT (
    CLK : IN STD_LOGIC;
    A : IN STD_LOGIC_VECTOR(11 DOWNTO 0);
    B : IN STD_LOGIC_VECTOR(11 DOWNTO 0);
    P : OUT STD_LOGIC_VECTOR(23 DOWNTO 0)
  );
END COMPONENT;

COMPONENT hpf
    Port (
    filt_in : in std_logic_vector(11 downto 0);
    filt_lp_out : out std_logic_vector(11 downto 0);
    filt_hp_out : out std_logic_vector(11 downto 0);
    reset_n_in : in STD_LOGIC;
    clk_48KHz : in STD_LOGIC
    );

END COMPONENT;

COMPONENT clip_indicator
  PORT (
            clip_in                : in std_logic_vector(11 downto 0);
            clip_out               : out STD_LOGIC;
            clk                    : in  STD_LOGIC --48khZ

  );
END COMPONENT;

component adac
Port (     clk_1200KHz    : in STD_LOGIC;
           reset_n      : in STD_LOGIC;
            -- adc module connections
            sync_n1     : out std_logic;-- chip select          1 - 43
            sdi0        : in std_logic;-- serial data out 0     2 - 44
            sdi1        : in std_logic;-- serial data out 1     3 - 45
            -- dac module connections
            sync_n2     : out std_logic;-- chip select          1 - 6
            sdo0        : out std_logic;-- serial data in 0     2 - 5
            sdo1        : out std_logic;-- serial data in 1     3 - 4

            -- dac signals
            dac_input1 : in std_logic_vector(15 downto 0);
            dac_input2 : in std_logic_vector(15 downto 0);
            -- adc signals
            adc_reg1 : out std_logic_vector(15 downto 0);
            adc_reg2 : out std_logic_vector(15 downto 0);

           clk_48KHz_out : out STD_LOGIC
           );
end component;

signal counter : std_logic_vector(24 downto 0) := (others => '0');
signal res_count : std_logic_vector(24 downto 0) := (others => '0');
signal clk_count : std_logic_vector(3 downto 0) := (others => '0');
signal reset_n : std_logic := '0';
signal clk_48MHz : std_logic;
signal clk_24MHz : std_logic;
signal clk_48KHz : std_logic;
signal clk_1200KHz : std_logic := '0';

signal adc_out1 : std_logic_vector(15 downto 0) := (others => '0');
signal dac_input1 : std_logic_vector(15 downto 0) := (others => '0');

signal adc_out2 : std_logic_vector(15 downto 0) := (others => '0');
signal dac_input2 : std_logic_vector(15 downto 0) := (others => '0');

signal demod_in : std_logic_vector(11 downto 0) := (others => '0');
signal demod_in_del : std_logic_vector(11 downto 0) := (others => '0');
signal demod_in_del2 : std_logic_vector(11 downto 0) := (others => '0');
signal demod_out : std_logic_vector(23 downto 0) := (others => '0');

signal clip_in1 : std_logic_vector(11 downto 0) := (others => '0');
signal clip_in2 : std_logic_vector(11 downto 0) := (others => '0');

-- dc blocking filter (high-pass)
signal filt_lp_out : std_logic_vector(11 downto 0) := (others => '0');
signal filt_hp_out : std_logic_vector(11 downto 0) := (others => '0');
signal filt_in : std_logic_vector(11 downto 0) := (others => '0');

begin
-- module connectors

dac_input1 <= X"0" & not demod_out(23) & demod_out(19 downto 9);--demod data
--dac_input1 <= X"0" & not filt_hp_out(11) & filt_hp_out(10 downto 0);--filter data
--dac_input1 <= adc_out1;--loopback data
--dac_input1 <= X"0FFF";--fixed data
dac_input2 <= X"0000";--fixed data
--dac_input2 <= adc_out2;--loopback data

--filt_in <= not adc_out1(11) & adc_out1(7 downto 0) & "000";
filt_in <= not adc_out1(11) & adc_out1(10 downto 0);
demod_in <= filt_hp_out;
--demod_in <= not adc_out1(11) & adc_out1(10 downto 0);

sclk1 <= clk_1200KHz;
sclk2 <= clk_1200KHz;
gnd1 <= '0';
gnd2 <= '0';
vcc1 <= '1';
vcc2 <= '1';

--led(0) <= counter(22);
--led(1) <= '0';

-- clipping indicators
--clip_in1 <= not adc_out1(11) & adc_out1(10 downto 0);
--clip_in1 <= filt_in;
clip_in1 <= demod_in;
clip_in2 <= not dac_input1(11) & dac_input1(10 downto 0);
```

```
inst_c_ind1 : clip_indicator
    port map (
    clip_in => clip_in1,
    clip_out => led(0),
    clk => clk_48KHz

    );

inst_c_ind2 : clip_indicator
    port map (
    clip_in => clip_in2,
    clip_out => led(1),
    clk => clk_48KHz

    );

-- Clock module
pll1 : clk_wiz_1
   port map (
  -- Clock out ports
   clk_out1 => clk_48MHz,
   clk_out2 => clk_24MHz,
  -- Status and control signals
   reset => '0',
   locked => open,
   -- Clock in ports
   clk_in1 => clk
 );
 -- clk_1200KHz process
            clk_process: process(clk_24MHz)
                        begin
                                    if rising_edge(clk_24MHz) then
                                                if clk_count = 9 then
```

```
                    clk_1200KHz <= not(clk_1200KHz);
                    clk_count <= (others => '0');
                else
                    clk_1200KHz <= clk_1200KHz;
                    clk_count <= clk_count + 1;
                end if;
            end if;
        end process;

-- CLK alive indicator
alive_proc:process(clk_24MHz)
    begin
        if rising_edge(clk_24MHz) then
            counter <= counter + 1;
        end if;
    end process;

-- Master reset of FPGA fabric
    reset_proc: process(clk_24MHz)
        begin
            if rising_edge(clk_24MHz) then
                if res_count(24) = '1' then
                    res_count <= res_count;
                else
                    res_count <= res_count + 1;
                end if;
            end if;
        end process;

    reset_n <= res_count(24);

-- DC blocking filter
inst_hpf : hpf
```

```
  PORT MAP (
    filt_in => filt_in,
    filt_lp_out => filt_lp_out,
    filt_hp_out => filt_hp_out,
    reset_n_in => reset_n,
    clk_48KHz => clk_48KHz
  );

-- demod process
            demod_process: process(clk_48KHz)
                        begin
                                    if rising_edge(clk_48KHz) then
                                                demod_in_del <= demod_in;
                                                demod_in_del2 <= demod_in_del;
                                    end if;
                        end process;

-- QUADRATURE DEMODULATOR
inst_quad_fm_demod : mult_gen_0
  PORT MAP (
    CLK => clk_1200KHz,
    A => demod_in_del,
    B => demod_in_del2,
    P => demod_out
  );

-- adac module
inst_adac : adac
    port map (
    clk_1200KHz => clk_1200KHz,
    reset_n => reset_n,
    -- adc module connections
     sync_n1  => sync_n1,
     sdi0 => sdi0,
     sdi1 => sdi1,
     -- dac module connections
     sync_n2 => sync_n2,
     sdo0  => sdo0,
     sdo1 => sdo1,

     -- dac signals
     dac_input1 => dac_input1,
     dac_input2  => dac_input2,

     -- adc signals
     adc_reg1 => adc_out1,
     adc_reg2 => adc_out2,

     clk_48KHz_out => clk_48KHz
         );

end Behavioral;
```

Appendix B.2 (ADAC.VHD)

```
----------------------------------------------------------------------------------
-- Company: KARABUK UNIVERSITY
-- Engineer: BILGEHAN ERKAL
--
-- Create Date: 09.10.2021 12:39:05
-- Design Name:
-- Module Name: adac - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------

library IEEE;
utilize IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
utilize IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--utilize UNISIM.VComponents.all;
utilize IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity adac is
    Port ( clk_1200KHz    : in STD_LOGIC;
           reset_n      : in STD_LOGIC;
            -- adc module connections
            sync_n1     : out std_logic;-- chip select          1 - 43
            sdi0        : in std_logic;-- serial data out 0     2 - 44
            sdi1        : in std_logic;-- serial data out 1     3 - 45
            -- dac module connections
            sync_n2     : out std_logic;-- chip select          1 - 6
            sdo0        : out std_logic;-- serial data in 0     2 - 5
            sdo1        : out std_logic;-- serial data in 1     3 - 4

            -- dac signals
            dac_input1 : in std_logic_vector(15 downto 0);
            dac_input2 : in std_logic_vector(15 downto 0);

            -- adc signals
            adc_reg1 : out std_logic_vector(15 downto 0);
            adc_reg2 : out std_logic_vector(15 downto 0);

           clk_48KHz_out : out STD_LOGIC);
end adac;

architecture Behavioral of adac is

-- ADAC process vars
signal adac_state : std_logic_vector(4 downto 0) := (others => '0');
signal cs_n : std_logic;
signal clk_48KHz : std_logic := '0';
signal sd_reg1 : std_logic_vector(15 downto 0) := (others => '0');
signal adc_out1 : std_logic_vector(15 downto 0) := (others => '0');
signal dac_in1 : std_logic_vector(15 downto 0) := (others => '0');
signal sd_reg2 : std_logic_vector(15 downto 0) := (others => '0');
signal adc_out2 : std_logic_vector(15 downto 0) := (others => '0');
signal dac_in2 : std_logic_vector(15 downto 0) := (others => '0');

begin

-- module connectors
sync_n1 <= cs_n;
sync_n2 <= cs_n;
sdo0 <= dac_in1(15);
sdo1 <= dac_in2(15);
clk_48KHz_out <= clk_48KHz;
adc_reg1 <= adc_out1;
adc_reg2 <= adc_out2;

-- ADAC process
adac_proc: process(clk_1200KHz)
        begin
                if rising_edge(clk_1200KHz) then
                 if reset_n = '1' then
                        CASE adac_state(4 downto 0) IS
                                WHEN X"00" => --
                                        adac_state <= adac_state + 1;
                                        cs_n <= '0';
                                        sd_reg1 <= (others => '0');
                                        adc_out1 <= adc_out1;
                                        dac_in1 <= dac_in1;
                                        sd_reg2 <= (others => '0');
                                        adc_out2 <= adc_out2;
                                        dac_in2 <= dac_in2;
                                        clk_48KHz <= '0';

                                WHEN X"01" => --
```

```
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"02" => --
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"03" => --
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"04" => --
```

```
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"05" => --
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"06" => --
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"07" => --
```

```
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"08" => --
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"09" => --
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"0A" => --
```

```
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"0B" => --
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"0C" => --
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"0D" => --
```

```
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"0E" => --
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"0F" => --
          adac_state <= adac_state + 1;
          cs_n <= '0';
          sd_reg1 <= sd_reg1(14 downto 0) & sdi0;
          adc_out1 <= adc_out1;
          dac_in1 <= dac_in1(14 downto 0) & '0';
          sd_reg2 <= sd_reg2(14 downto 0) & sdi1;
          adc_out2 <= adc_out2;
          dac_in2 <= dac_in2(14 downto 0) & '0';
          clk_48KHz <= '0';

WHEN X"10" => --
```

```
                adac_state <= adac_state + 1;
                cs_n <= '1';
sd_reg1 <= sd_reg1;--(14 downto 0) & sdi0;
                adac_out1 <= adc_out1;
                dac_in1 <= dac_in1;
      sd_reg2 <= sd_reg2;--(14 downto 0) & sdi1;
                adc_out2 <= adc_out2;
                dac_in2 <= dac_in2;
                clk_48KHz <= '0';

   WHEN X"11" => --
                adac_state <= adac_state + 1;
                cs_n <= '1';
                sd_reg1 <= sd_reg1;
                adc_out1 <= sd_reg1;
                dac_in1 <= dac_in1;
                sd_reg2 <= sd_reg2;
                adc_out2 <= sd_reg2;
                dac_in2 <= dac_in2;
                clk_48KHz <= '0';

   WHEN X"12" => --
                adac_state <= adac_state + 1;
                cs_n <= '1';
                sd_reg1 <= sd_reg1;
                adc_out1 <= adc_out1;
                dac_in1 <= dac_input1;
                sd_reg2 <= sd_reg2;
                adc_out2 <= adc_out2;
                dac_in2 <= dac_input2;
                clk_48KHz <= '0';

   WHEN X"13" => --
```

```
                adac_state <= adac_state + 1;
                cs_n <= '1';
                sd_reg1 <= sd_reg1;
                adc_out1 <= adc_out1;
                dac_in1 <= dac_in1;
                sd_reg2 <= sd_reg2;
                adc_out2 <= adc_out2;
                dac_in2 <= dac_in2;
                clk_48KHz <= '0';

        WHEN X"14" => --
                adac_state <= adac_state + 1;
                cs_n <= '1';
                sd_reg1 <= sd_reg1;
                adc_out1 <= adc_out1;
                dac_in1 <= dac_in1;
                sd_reg2 <= sd_reg2;
                adc_out2 <= adc_out2;
                dac_in2 <= dac_in2;
                clk_48KHz <= '0';

        WHEN X"15" => --
                adac_state <= adac_state + 1;
                cs_n <= '1';
                sd_reg1 <= sd_reg1;
                adc_out1 <= adc_out1;
                dac_in1 <= dac_in1;
                sd_reg2 <= sd_reg2;
                adc_out2 <= adc_out2;
                dac_in2 <= dac_in2;
                clk_48KHz <= '0';

        WHEN X"16" => --
        adac_state <= adac_state + 1;
        cs_n <= '1';
        sd_reg1 <= sd_reg1;
        adc_out1 <= adc_out1;
        dac_in1 <= dac_in1;
        sd_reg2 <= sd_reg2;
        adc_out2 <= adc_out2;
        dac_in2 <= dac_in2;
        clk_48KHz <= '0';

WHEN X"17" => --
        adac_state <= adac_state + 1;
        cs_n <= '1';
        sd_reg1 <= sd_reg1;
        adc_out1 <= adc_out1;
        dac_in1 <= dac_in1;
        sd_reg2 <= sd_reg2;
        adc_out2 <= adc_out2;
        dac_in2 <= dac_in2;
        clk_48KHz <= '1';

WHEN X"18" => --
        adac_state <= (others => '0');
        cs_n <= '1';
        sd_reg1 <= sd_reg1;
        adc_out1 <= adc_out1;
        dac_in1 <= dac_in1;
        sd_reg2 <= sd_reg2;
        adc_out2 <= adc_out2;
        dac_in2 <= dac_in2;
        clk_48KHz <= '0';

WHEN OTHERS =>--
```

```
                                        adac_state <= (others => '0');
                                        cs_n <= '1';
                                        sd_reg1 <= sd_reg1;
                                        adc_out1 <= adc_out1;
                                        dac_in1 <= dac_in1;
                                        sd_reg2 <= sd_reg2;
                                        adc_out2 <= adc_out2;
                                        dac_in2 <= dac_in2;
                                        clk_48KHz <= '0';

                              END CASE;
                    else -- reset in order
                              adac_state <= (others => '0');
          cs_n <= '1';
          sd_reg1 <= (others => '0');
          adc_out1 <= (others => '0');
          dac_in1 <= (others => '0');
          sd_reg2 <= (others => '0');
          adc_out2 <= (others => '0');
          dac_in2 <= (others => '0');
                              clk_48KHz <= '0';

                    end if;
                  end if;
          end process;

end Behavioral;
```

Appendix B.3 (HPF.VHD)

```
--------------------------------------------------------------------------------
-- Company: KARABUK UNIVERSITY
-- Engineer: BILGEHAN ERKAL
--
-- Create Date: 22.12.2021 17:59:53
-- Design Name:
-- Module Name: hpf - Behavioral

library IEEE;
utilize IEEE.STD_LOGIC_1164.ALL;

entity hpf is
    Port (
    filt_in : in std_logic_vector(11 downto 0);
    filt_lp_out : out std_logic_vector(11 downto 0);
    filt_hp_out : out std_logic_vector(11 downto 0);
    reset_n_in : in STD_LOGIC;
    clk_48KHz : in STD_LOGIC
    );
end hpf;
architecture Behavioral of hpf is

COMPONENT sub
  PORT (
    A : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    B : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    S : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;
COMPONENT add1
  PORT (
    A : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    B : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    S : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
```

```
END COMPONENT;

COMPONENT c_addsub_0
  PORT (
    A : IN STD_LOGIC_VECTOR(11 DOWNTO 0);
    B : IN STD_LOGIC_VECTOR(11 DOWNTO 0);
    S : OUT STD_LOGIC_VECTOR(11 DOWNTO 0)
  );
END COMPONENT;

signal acc : std_logic_vector(31 downto 0) := (others => '0');
signal s : std_logic_vector(31 downto 0) := (others => '0');
signal add : std_logic_vector(31 downto 0) := (others => '0');
signal hpf_out : std_logic_vector(11 downto 0) := (others => '0');
signal fin : std_logic_vector(11 downto 0) := (others => '0');
signal avg : std_logic_vector(11 downto 0) := (others => '0');
signal ss : std_logic_vector(19 downto 0) := (others => '0');
signal ff : std_logic_vector(19 downto 0) := (others => '0');

begin

avg <= acc(31 downto 20);
filt_lp_out <= avg;

ss <= (others => avg(11));
inst_sub1 : sub
  PORT MAP (
    A => acc,
    B => ss & avg,
    S => s
  );

ff <= (others => fin(11));
```

```
inst_add1 : add1
  PORT MAP (
    A => s,
    B => ff & fin,
    S => add
  );

inst_sub2 : c_addsub_0
  PORT MAP (
    A => fin,
    B => avg,
    S => hpf_out
  );
-- filter process
            filter_process: process(clk_48KHz)
                        begin
                                    if rising_edge(clk_48KHz) then
                                     if reset_n_in <= '1' then
                                         acc <= add;
                                         fin <= filt_in;
                                                 filt_hp_out <= hpf_out;
                                     else
                                         acc <= (others => '0');
                                         fin <= (others => '0');
                                                 filt_hp_out <= (others => '0');
                                     end if;
                                    end if;
                        end process;

end Behavioral;
```

Appendix B.4 (CLIP_INDICATOR.VHD)

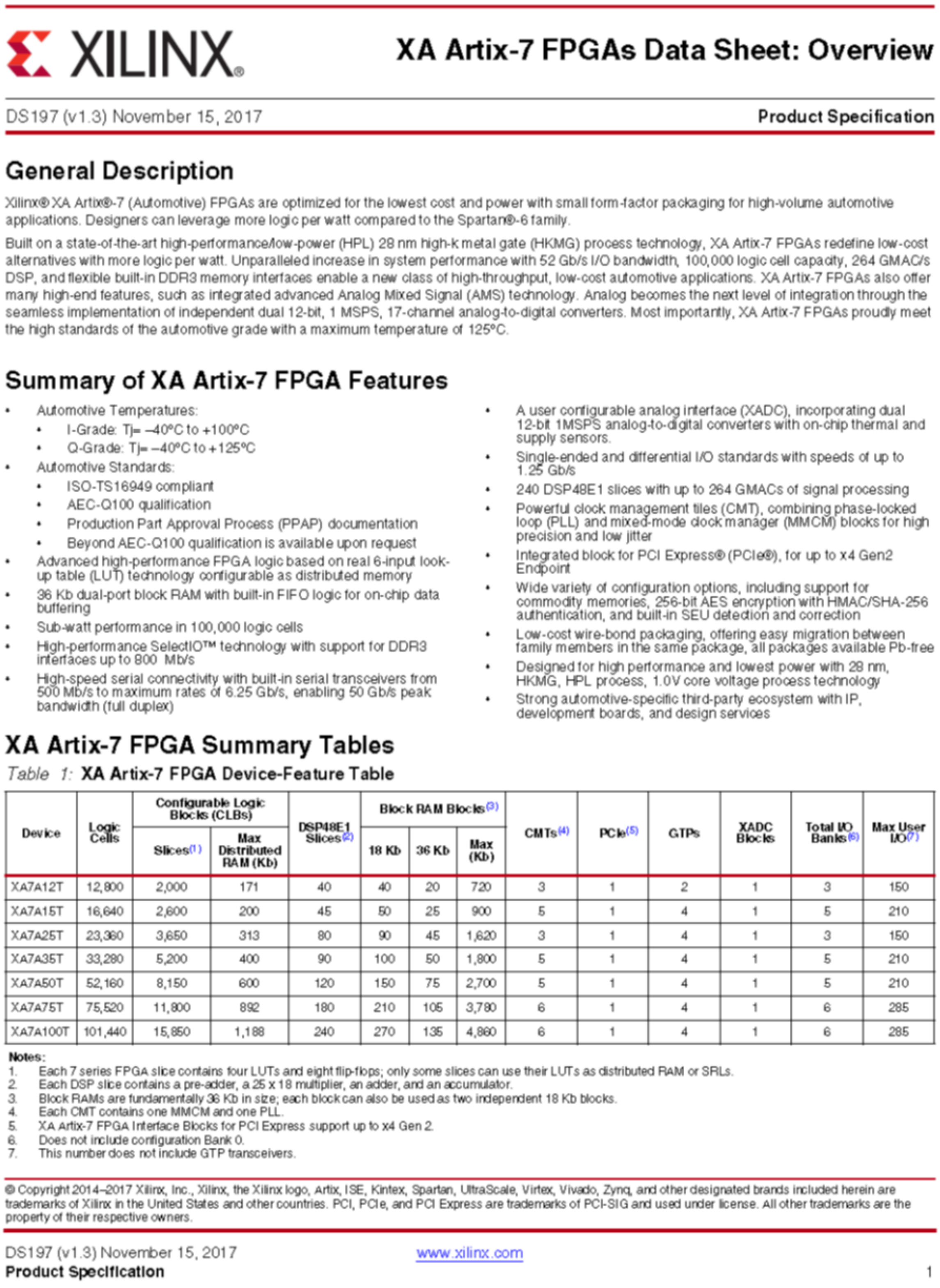
```
--------------------------------------------------------------------------------
-- Company:  KARABUK UNIVERSITY
-- Engineer: BILGEHAN ERKAL
--
-- Create Date: 12.10.2021 22:06:00
-- Design Name:
-- Module Name: clip_indicator - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------

library IEEE;
utilize IEEE.STD_LOGIC_1164.ALL;
utilize IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
utilize IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--utilize UNISIM.VComponents.all;
entity clip_indicator is
   Port (
          clip_in                 : in std_logic_vector(11 downto 0);
```

```
        clip_out                : out STD_LOGIC;
        clk                     : in  STD_LOGIC --48khZ

        );

end clip_indicator;

architecture Behavioral of clip_indicator is

signal c_in_s : std_logic_vector(11 downto 0) := (others => '0');
signal timex : std_logic_vector(15 downto 0):= (others => '0');

begin

c_in_s <= not clip_in(11) & clip_in(10 downto 0);

clip_proc:process(clk)
        begin
                if rising_edge(clk) then
                        if ((c_in_s < x"266") or (c_in_s > x"D9A")) then
                                clip_out <= '1';
                                timex <= x"0001";
                        else
                                if (timex = 0) then
                                        clip_out <= '0';
                                else
                                        clip_out <= '1';
                                        timex <= timex + 1;
                                end if;
                        end if;
                end if;
        end process;
end Behavioral;
```

APPENDIX C.

DATASHEETS OF CHIPS UTLIZED IN THE PROJECT

Appendix C.1 XILINX Artix-7 FPGA (CX7A35T-1CPG236C) Datasheet



XILINX

XA Artix-7 FPGAs Data Sheet: Overview

DS197 (v1.3) November 15, 2017 **Product Specification**

General Description

Xilinx® XA Artix®-7 (Automotive) FPGAs are optimized for the lowest cost and power with small form-factor packaging for high-volume automotive applications. Designers can leverage more logic per watt compared to the Spartan®-6 family.

Built on a state-of-the-art high-performance/low-power (HPL) 28 nm high-k metal gate (HKMG) process technology, XA Artix-7 FPGAs redefine low-cost alternatives with more logic per watt. Unparalleled increase in system performance with 52 Gb/s I/O bandwidth, 100,000 logic cell capacity, 264 GMAC/s DSP, and flexible built-in DDR3 memory interfaces enable a new class of high-throughput, low-cost automotive applications. XA Artix-7 FPGAs also offer many high-end features, such as integrated advanced Analog Mixed Signal (AMS) technology. Analog becomes the next level of integration through the seamless implementation of independent dual 12-bit, 1 MSPS, 17-channel analog-to-digital converters. Most importantly, XA Artix-7 FPGAs proudly meet the high standards of the automotive grade with a maximum temperature of 125°C.

Summary of XA Artix-7 FPGA Features

- Automotive Temperatures:
  - I-Grade: Tj= –40°C to +100°C
  - Q-Grade: Tj= –40°C to +125°C
- Automotive Standards:
  - ISO-TS16949 compliant
  - AEC-Q100 qualification
  - Production Part Approval Process (PPAP) documentation
  - Beyond AEC-Q100 qualification is available upon request
- Advanced high-performance FPGA logic based on real 6-input look-up table (LUT) technology configurable as distributed memory
- 36 Kb dual-port block RAM with built-in FIFO logic for on-chip data buffering
- Sub-watt performance in 100,000 logic cells
- High-performance SelectIO™ technology with support for DDR3 interfaces up to 800 Mb/s
- High-speed serial connectivity with built-in serial transceivers from 500 Mb/s to maximum rates of 6.25 Gb/s, enabling 50 Gb/s peak bandwidth (full duplex)
- A user configurable analog interface (XADC), incorporating dual 12-bit 1MSPS analog-to-digital converters with on-chip thermal and supply sensors.
- Single-ended and differential I/O standards with speeds of up to 1.25 Gb/s
- 240 DSP48E1 slices with up to 264 GMACs of signal processing
- Powerful clock management tiles (CMT), combining phase-locked loop (PLL) and mixed-mode clock manager (MMCM) blocks for high precision and low jitter
- Integrated block for PCI Express® (PCIe®), for up to x4 Gen2 Endpoint
- Wide variety of configuration options, including support for commodity memories, 256-bit AES encryption with HMAC/SHA-256 authentication, and built-in SEU detection and correction
- Low-cost wire-bond packaging, offering easy migration between family members in the same package, all packages available Pb-free
- Designed for high performance and lowest power with 28 nm, HKMG, HPL process, 1.0V core voltage process technology
- Strong automotive-specific third-party ecosystem with IP, development boards, and design services

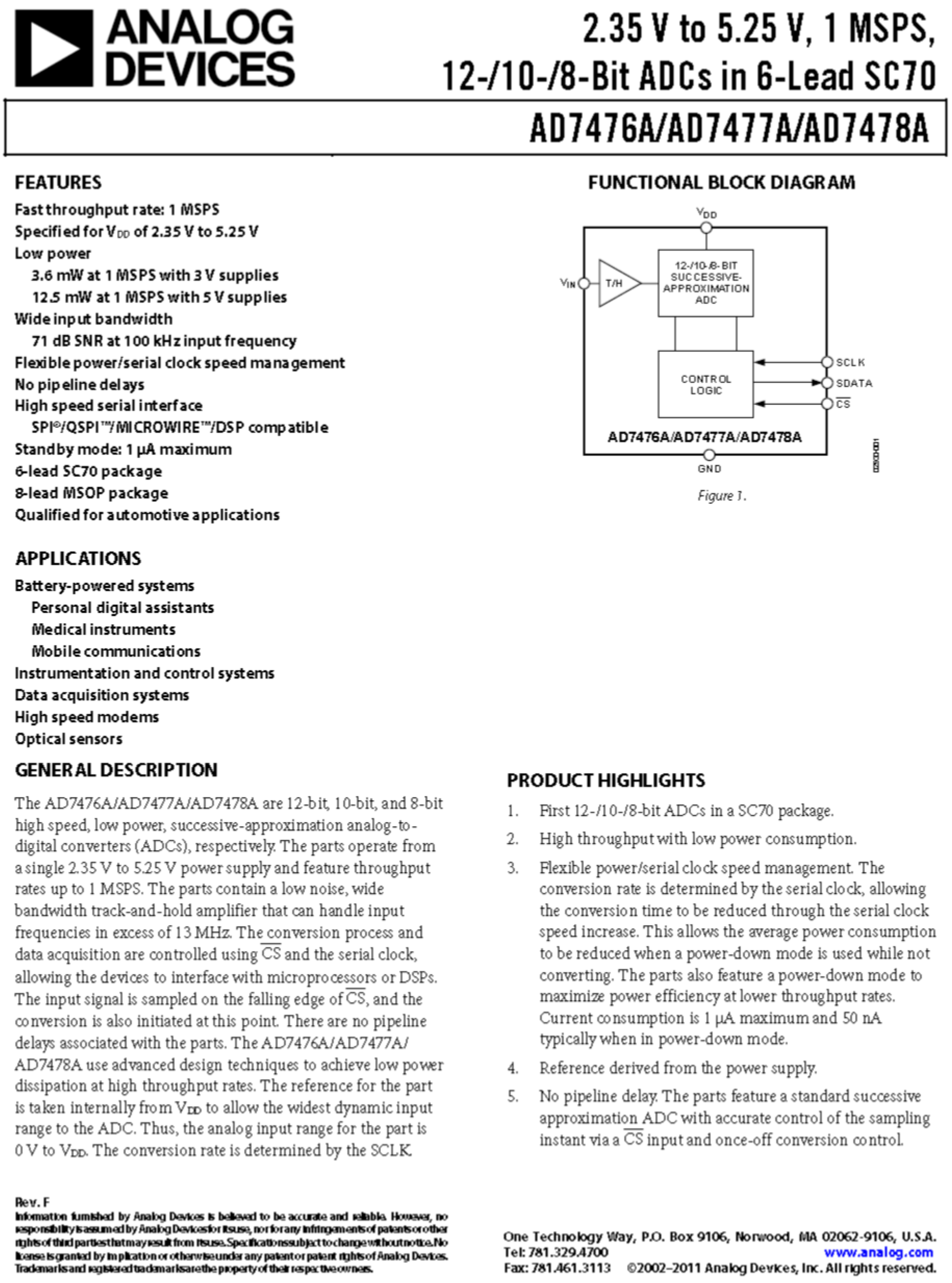
XA Artix-7 FPGA Summary Tables

Table 1: **XA Artix-7 FPGA Device-Feature Table**

Device	Logic Cells	Configurable Logic Blocks (CLBs)		DSP48E1 Slices[2]	Block RAM Blocks[3]			CMTs[4]	PCIe[5]	GTPs	XADC Blocks	Total I/O Banks[6]	Max User I/O[7]
		Slices[1]	Max Distributed RAM (Kb)		18 Kb	36 Kb	Max (Kb)						
XA7A12T	12,800	2,000	171	40	40	20	720	3	1	2	1	3	150
XA7A15T	16,640	2,600	200	45	50	25	900	5	1	4	1	5	210
XA7A25T	23,360	3,650	313	80	90	45	1,620	3	1	4	1	3	150
XA7A35T	33,280	5,200	400	90	100	50	1,800	5	1	4	1	5	210
XA7A50T	52,160	8,150	600	120	150	75	2,700	5	1	4	1	5	210
XA7A75T	75,520	11,800	892	180	210	105	3,780	6	1	4	1	6	285
XA7A100T	101,440	15,850	1,188	240	270	135	4,860	6	1	4	1	6	285

Notes:
1. Each 7 series FPGA slice contains four LUTs and eight flip-flops; only some slices can use their LUTs as distributed RAM or SRLs.
2. Each DSP slice contains a pre-adder, a 25 x 18 multiplier, an adder, and an accumulator.
3. Block RAMs are fundamentally 36 Kb in size; each block can also be used as two independent 18 Kb blocks.
4. Each CMT contains one MMCM and one PLL.
5. XA Artix-7 FPGA Interface Blocks for PCI Express support up to x4 Gen 2.
6. Does not include configuration Bank 0.
7. This number does not include GTP transceivers.

Appendix C. 2 ADC Chip Analog Devices AD7476 Datasheet



ANALOG DEVICES

2.35 V to 5.25 V, 1 MSPS, 12-/10-/8-Bit ADCs in 6-Lead SC70

AD7476A/AD7477A/AD7478A

FEATURES

Fast throughput rate: 1 MSPS
Specified for V_{DD} of 2.35 V to 5.25 V
Low power
  3.6 mW at 1 MSPS with 3 V supplies
  12.5 mW at 1 MSPS with 5 V supplies
Wide input bandwidth
  71 dB SNR at 100 kHz input frequency
Flexible power/serial clock speed management
No pipeline delays
High speed serial interface
  SPI®/QSPI™/MICROWIRE™/DSP compatible
Standby mode: 1 µA maximum
6-lead SC70 package
8-lead MSOP package
Qualified for automotive applications

APPLICATIONS

Battery-powered systems
  Personal digital assistants
  Medical instruments
  Mobile communications
Instrumentation and control systems
Data acquisition systems
High speed modems
Optical sensors

FUNCTIONAL BLOCK DIAGRAM

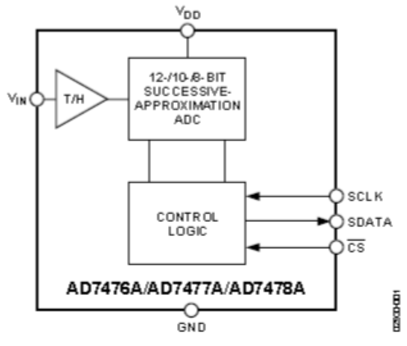


Figure 1.

GENERAL DESCRIPTION

The AD7476A/AD7477A/AD7478A are 12-bit, 10-bit, and 8-bit high speed, low power, successive-approximation analog-to-digital converters (ADCs), respectively. The parts operate from a single 2.35 V to 5.25 V power supply and feature throughput rates up to 1 MSPS. The parts contain a low noise, wide bandwidth track-and-hold amplifier that can handle input frequencies in excess of 13 MHz. The conversion process and data acquisition are controlled using $\overline{CS}$ and the serial clock, allowing the devices to interface with microprocessors or DSPs. The input signal is sampled on the falling edge of $\overline{CS}$, and the conversion is also initiated at this point. There are no pipeline delays associated with the parts. The AD7476A/AD7477A/AD7478A use advanced design techniques to achieve low power dissipation at high throughput rates. The reference for the part is taken internally from V_{DD} to allow the widest dynamic input range to the ADC. Thus, the analog input range for the part is 0 V to V_{DD}. The conversion rate is determined by the SCLK.

PRODUCT HIGHLIGHTS

1. First 12-/10-/8-bit ADCs in a SC70 package.
2. High throughput with low power consumption.
3. Flexible power/serial clock speed management. The conversion rate is determined by the serial clock, allowing the conversion time to be reduced through the serial clock speed increase. This allows the average power consumption to be reduced when a power-down mode is used while not converting. The parts also feature a power-down mode to maximize power efficiency at lower throughput rates. Current consumption is 1 µA maximum and 50 nA typically when in power-down mode.
4. Reference derived from the power supply.
5. No pipeline delay. The parts feature a standard successive approximation ADC with accurate control of the sampling instant via a $\overline{CS}$ input and once-off conversion control.

Rev. F

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 www.analog.com
Fax: 781.461.3113 ©2002–2011 Analog Devices, Inc. All rights reserved.

Appendix C. 3 Texas Instruments DAC121S101 DAC Chip Datasheet

TEXAS INSTRUMENTS

DAC121S101, DAC121S101-Q1

SNAS265J – JUNE 2005 – REVISED SEPTEMBER 2015

DAC121S101/-Q1 12-Bit Micro Power, RRO Digital-to-Analog Converter

1 Features

- DAC121S101-Q1 is AEC-Q100 Grade 1 Qualified and is Manufactured on an Automotive Grade Flow.
- Ensured Monotonicity
- Low Power Operation
- Rail-to-Rail Voltage Output
- Power-on Reset to Zero Volts Output
- Wide Temperature Range of –40°C to +125°C
- Wide Power Supply Range of 2.7 V to 5.5 V
- Small Packages
- Power Down Feature
- Key Specifications
  – 12-Bit Resolution
  – DNL -0.15, +0.25 LSB (Typical)
  – 8-μs Output Settling Time (Typical)
  – 4-mV Zero Code Error (Typical)
  – Full-Scale Error at −0.06 %FS (Typical)
  – 0.64-mW (3.6-V) / 1.43-mW (5.5-V) Normal Mode Power Consumption (Typical)
  – 0.14-μW (3.6-V) / 0.39-μW (5.5-V) Power-Down Mode (Typical)

2 Applications

- Battery-Powered Instruments
- Digital Gain and Offset Adjustment
- Programmable Voltage and Current Sources
- Programmable Attenuators
- Automotive

3 Description

The DAC121S101 device is a full-featured, general-purpose, 12-bit voltage-output digital-to-analog converter (DAC) that can operate from a single 2.7-V to 5.5-V supply and consumes just 177 µA of current at 3.6 V. The on-chip output amplifier allows rail-to-rail output swing and the three wire serial interface operates at clock rates up to 30 MHz over the specified supply voltage range and is compatible with standard SPI™, QSPI, MICROWIRE and DSP interfaces. Competitive devices are limited to 20-MHz clock rates at supply voltages in the 2.7 V to 3.6 V range.

The supply voltage for the DAC121S101 serves as its voltage reference, providing the widest possible output dynamic range. A power-on reset circuit ensures that the DAC output powers up to zero volts and remains there until there is a valid write to the device. A power-down feature reduces power consumption to less than a microWatt.

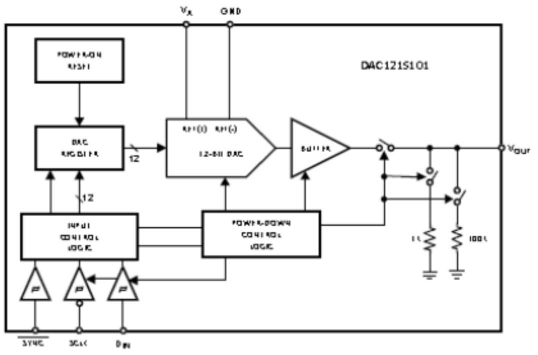
The low power consumption and small packages of the DAC121S101 make it an excellent choice for use in battery operated equipment.

Device Information(1)

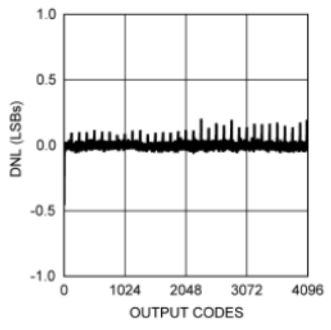
PART NUMBER	PACKAGE	BODY SIZE (NOM)
DAC121S101	SOT (6)	2.90 mm × 1.60 mm
	VSSOP (8)	3.00 mm × 3.00 mm
DAC121S101-Q1	SOT (6)	2.90 mm × 1.60 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Simplified Block Diagram



DNL vs. Output Code



RESUME

ABDALLA.M.MOHAMED ELSALHIN completed his secondary education at the General Secondary School, after which he started a bachelor’s program in the College of Technology, Electrical and Electronic Engineering, majoring in Communications in 2018. Then, in 2020, he started a master’s degree. Education at Karabük University Department of Electrical and Electronic Engineering.