

# Implementation of FPGA Architecture for OFDM-SDR with an optimized Direct Digital Frequency Synthesizer

*Aluru Koteswara Rao, A. Anasuyamma, M. Tech*

*Department of Electronics and communication Engineering,  
Audisankara College of Engineering & Technology, gudur.*

*(Autonomous)*

**Abstract:** A Software Defined Radio (SDR) is defined as a radio in which the receive digitization is performed at some stage downstream from the antenna, typically after wideband filtering, low noise amplification, and down conversion to a lower frequency in subsequent stages - with a reverse process occurring for the transmit digitization. In an SDR, Digital Signal Processing in flexible and reconfigurable functional blocks define the characteristics of the radio. Researchers in the area of Communication and VLSI have a growing tendency to develop state-of-the-art architectures for SDR. The backbone of SDR framework in the digital domain is a direct digital frequency synthesizer (DDFS). Using DDFS, we can generate the high frequency carrier waves in digital domain and modulate the message on it. The optimization of the DDFS includes Lookup Table based designs and Coordinate Rotation Digital Computer (CORDIC) based designs. Lookup Table based designs require huge ROMs for implementation and are declared to be area-hungry. On the other hand, CORDIC based techniques use iterative algorithms for the computation of Sine and Cosine functions and are computationally inefficient. Each sample of Sine and Cosine requires 4 multiplications and 2 additions. However, the design in "*The Proposed Architecture for DDFS*" is even simpler and faster. It utilizes 2 adders and 2 multipliers to generate a sample of sine and cosine. This design is more area and time efficient than CORDIC and look-up table based approaches.

The main objective of This paper presents the framework for hardware implementation of SDR using Orthogonal Frequency Division Multiplexing (OFDM). The framework comprises of VLSI mapping of algorithms, Orthogonal Frequency Division Multiplexing (OFDM), 8 Phase Shift Keying (8PSK), Fast Fourier Transform (FFT) Algorithms and most importantly, the algorithm for Direct Digital Frequency Synthesis (DDFS). A digital frequency synthesizer with optimized time and area resources has been proposed for the SDR. This VLSI implementation of the DDFS computes the sine and cosine function on a single edge of clock, thus proving to be optimized in terms of area and speed.

In this design I want to use Verilog HDL as a description language for mapping Algorithms in VLSI. Xilinx Spartan 3 XC3S200 Field Programmable Gate Array (FPGA) was chosen as a Hardware Platform for the System Implementation.

**Keywords—** *Software Defined Radio, Direct Digital Frequency Synthesis, Orthogonal Frequency Division Multiplexing.*

## I. INTRODUCTION

A Software Defined Radio (SDR) is defined as a radio in which the receive digitization is performed at some stage downstream from the antenna, typically after wideband filtering, low noise amplification, and down conversion to a lower frequency in subsequent stages - with a reverse process occurring for the transmit digitization. In an SDR, Digital Signal Processing in flexible and reconfigurable functional blocks define the characteristics of the radio. As technology progresses, an SDR can move to an almost total Software Radio (SR), where the digitization is at (or very near to) the antenna and all of the processing required for the radio is performed by software residing in high-speed digital signal processing elements. The SDR will occur in the near term, migrating to the SR in the longer term, subject to the progression of core technologies. The need for such progression will be a function of the application. For example, a base station application may require and/or be

able by virtue of technology advances and design latitude to move to an SR. But a handset or portable terminal, because of numerous constraints, may not need or be able to progress beyond an SDR [1]. Researchers in the area of Communication and VLSI have a growing tendency to develop state-of-the-art architectures for SDR. Recent developments include the frameworks proposed in [2][3][4][5].

The backbone of SDR framework in the digital domain is a direct digital frequency synthesizer (DDFS). Using DDFS, we can generate the high frequency carrier waves in digital domain and modulate the message on it, and then convert it to analog form using a digital to analog converter (DAC) before antenna. This is referred to be the digital up-conversion (DUC) [6][7]. The counter-part of digital up-converter, on the receiver end, is digital down converter (DDC). VLSI has been a key technology for mapping communication and signal processing algorithms in hardware. A key challenge for researchers has been the optimization of the VLSI circuit in terms of area as well as time. Conventional methods include pipelining, retiming, parallel processing etc. Unconventional methods refer to evolutionary and genetic

algorithms, genetic programming and Cartesian genetic programming [8][9][10][11]. The optimization of the DDFS includes Lookup Table based designs and Coordinate Rotation Digital Computer (CORDIC) based designs. While [12] have used Lookup Table based approaches, CORDIC based approaches have been used in for the generation of Sine and Cosine. Janiszewski has adapted a hybrid scheme for the design of Numeric Controlled Oscillator (NCO) using both the Lookup Table based and CORDIC based approach. Lookup Table based designs require huge ROMs for implementation and are declared to be area-hungry. On the other hand, CORDIC based techniques use iterative algorithms for the computation of Sine and Cosine functions and are computationally inefficient. Singleton used basic trigonometric identities to compute the values of Sine and Cosine. Each sample of Sine and Cosine requires 4 multiplications and 2 additions. However, the design is even simpler and faster. It utilizes 2 adders and 2 multipliers to generate a sample of sine and cosine. This design is more area and time efficient than CORDIC and look-up table based approaches. But the Architecture for VLSI implementation proposed, as shown in Fig. 1 has utilized registers being triggered on the positive and negative edges of the same clock. Such designs are not synthesizable on Field Programmable Gate Arrays (FPGA). So, in this work, a novel architecture has been proposed which generates the Sine and Cosine function using the single clock edge. The proposed architecture reduces the required hardware resources while realizing a synthesizable efficient design.

The remaining of the paper is organized as; Section II briefly describes the overall framework and the functionality of each block of the framework. Detailed discussion on the implementation of the blocks and especially the FFT block has been made in section III. DDFS is the key component of the DUC and DDC blocks and is the focus of this paper. Thus, a separate section i.e. section IV has been dedicated for the discussion of DDFS. Finally, section V concludes the paper.

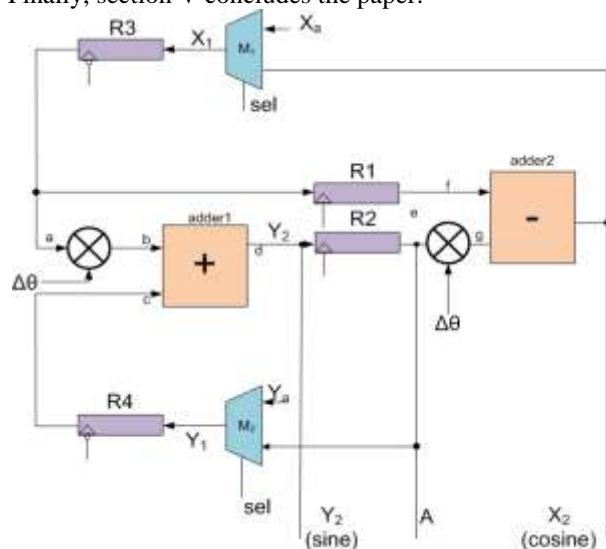


Fig. 1 Architecture proposed

## II. METHODOLOGY

This work contains simulation and Verilog HDL implementation of OFDM based transmitter and receiver system. After floating point simulation of the framework, Verilog HDL has been used for fixed point simulation and description of hardware details. The transmitter first

converts the input data from a serial stream to parallel sets. Each set of data contains one information bit for each carrier frequency.

Then, parallel data are modulated to the orthogonal carrier frequencies. The IFFT converts the parallel data into time domain waveforms. Finally, these waveforms are combined to create a single time domain signal for transmission. The channel simulation will allow for us to examine the effects of noise and multipath on the OFDM scheme. By adding small amount of random data to the transmitted signal, simple noise can be simulated. Multipath simulation involves adding attenuated and delayed copies of the transmitted signal to the original. This simulates the problem in wireless communication when the signal propagates on many paths. For example, a receiver may see a signal via a direct path as well as a path that bounces-off of a building. The receiver basically performs the inverse of the transmitter by first separating the data into parallel streams. The FFT converts the parallel data streams into frequency domain data. The data are now available in modulated form on the orthogonal carriers. Demodulation down-converts this information back to the baseband. Finally, the parallel data are converted back into a serial stream to recover the original signal. The overall block diagram of the system has been shown in Fig. 2.

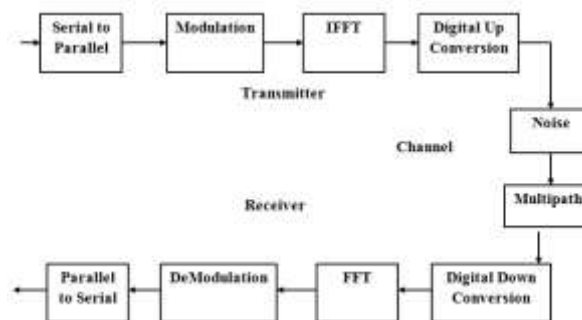


Fig. 2 Block Diagram of OFDM Software Defined Radio

## III. BLOCK DESCRIPTION AND IMPLEMENTATION

### A. Serial-to-parallel Conversion and Parallel-to-Serial Conversion

The Serial-to-Parallel converter takes 8 bits at a time and produces four parallel streams of two bits each. It takes 8 clock cycles. The ASM chart for serial to parallel conversion has been shown in Fig. 3. On the receiver end, the parallel-to-serial converter takes four parallel streams, each one consisting of two bits and transforms it into a stream of 8 bits serial data.

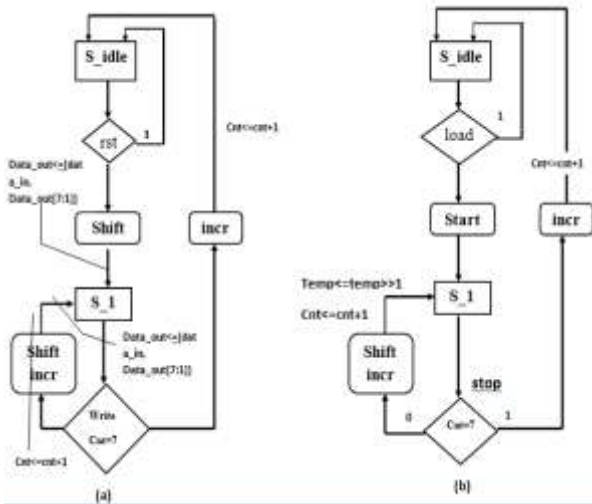


Fig.3. ASM Chart for (a) Serial to Parallel Conversion (b) Parallel to Serial Conversion

### B. Modulation and Demodulation

In this work, we have used Quadrature Phase Shift Keying (QPSK) Modulation. The data is divided into chunks of a set of bits, each containing 3 bits of data, leading to 8 distinct combinations. Each combination is assigned a different symbol. In order to ensure maximum likelihood, the separation between angles should be as large as possible. Hence, an angle assigned to each symbol is  $22.5^\circ$ ,  $67.5^\circ$ ,  $112.5^\circ$ ,  $157.5^\circ$ ,  $202.5^\circ$ ,  $247.5^\circ$ ,  $292.5^\circ$ ,  $337.5^\circ$ . Let us consider  $A= 0.3826$  &  $B= 0.9238$

TABLE I. CORRESPONDING ANGLE OF QPSK MODULATION

Bit Sequence	Modulation Angle	Mapped Sequence
000	$22.5^\circ$	$A+jB$
001	$67.5^\circ$	$B+jA$
010	$112.5^\circ$	$-A+jB$
011	$157.5^\circ$	$-B+jA$
100	$202.5^\circ$	$-A-jB$
101	$247.5^\circ$	$-B-jA$
110	$292.5^\circ$	$A-jB$
111	$337.5^\circ$	$B-jA$

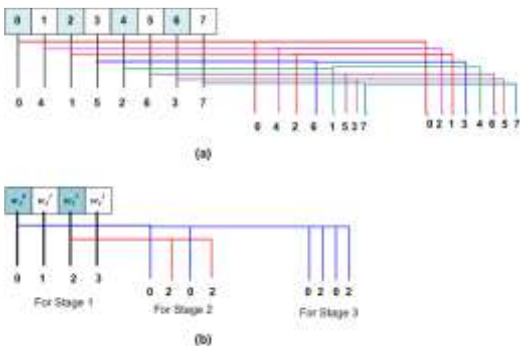


Fig. 4 Memory Representation. (a) Input Memory. (b) Twiddle Memory

### C. Fast Fourier Transform

The FFT machine is partitioned into data path and controller i.e. the Finite State Machine (FSM). It simplifies

the architecture design and understanding. The data path presented in this work contains a single butterfly stage, consisting of  $N/2$  butterflies for  $N$ -order FFT, registers for holding real and imaginary data as well as the intermediate results of the various stages of FFT, real and imaginary twiddle memories and buses. The 8-point FFT design, in this work, consists of butterfly stage, twiddle memories, input register banks that also store the values calculated during the various stages of FFT, multiplexers for steering signal. There are 3 stages in 8-point FFT/IFFT, so input memories and twiddle memories give 3 outputs, one for each stage, as shown in the Fig. 4. In this way, the design of FSM is simplified so that it keeps track of the FFT current stage only, i.e. the stage being computed at that

Particular time. The butterfly stage comprises of  $N/2$  butterflies. The FFT data path has been shown in Fig. 5. Each sub-stage on the FFT takes input data in different order, and the twiddle values used in each sub-stage are also different. Thus, this FFT design simultaneously generates output for each sub-stage. The FSM selects one out of the three outputs, depending upon the stage of FFT that is being calculated. The state transition graph for the FFT FSM has been shown in Fig. 6. The number of states in FSM depends upon the FFT order. Initially, when the FFT machine is reset, the FSM enters the IDLE state, which means that the FFT machine is waiting for data to be processed. Once 8 samples of data are received by the serial-to-parallel converter before the FFT, a signal *Readyin* is asserted.

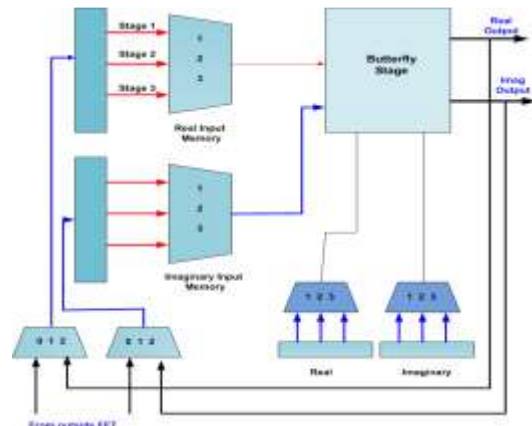


Fig. 5 FFT Data Path

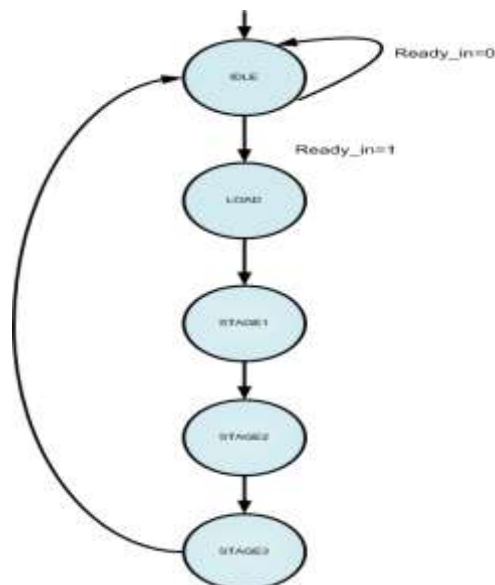


Fig. 6 State Transition Graph for FFT FSM

This signal is originated by serial-to-parallel converter. After the assertion of *Readyin* signal, FFT FSM goes to the *LOAD* state; otherwise it stays in the *IDLE* state. In the *LOAD* state, the FSM loads the input memories (real and imaginary) with input, and starts the FFT operation. On the next positive edge of the clock cycle, the FSM enters *STAGE-1* state, which means that stage 1 of the 8-point FFT is calculated. The number of clock cycles the FSM stays in *STAGE-1* state depends upon the specific implementation of the multipliers in the butterfly. Faster multiplier implementation means reduced clock cycles per computation stage of the FFT. After the first stage of FFT computation is calculated, the FSM enters *STAGE-2* state, which is similar in operation to the *STAGE-1*. Following this, the FSM enters *STAGE-3*, and returns to the *IDLE* state after completion of *STAGE-3*, and the whole cycle starts again.

The single butterfly model used in this work saves hardware resources. The butterfly stage takes input in the **Q8.8** format, and gives output in **Q8.8** format. There are two input memories for storing real and imaginary parts of the complex inputs. As FFT is a parallel process, so the memories do not give or store data serially but rather store or provide the data in chunks of particular number of bits, depending upon the order of FFT. The input memories are not only used to store the inputs at the beginning of the FFT operation, but also used to store output of the butterfly stage at the completion of the subsequent FFT stages. This helps in reducing the memory required for the FFT machine. The input memories are designed so that they can be read at the positive edge of the clock and data can be written to them at the negative edge of the clock. This helps in synchronization and ensures that data is not read before the completion of the modification of data in the memories. The twiddle memories are designed to give output at the positive edge of the clock.

The architecture for IFFT is almost similar to that of FFT except the division by 8 after completion of IFFT operation. The division by 8 in IFFT architecture is accomplished by shift registers and a change in twiddle factors. The rest of architecture of the data path and the FSM is the same.

#### IV. DIRECT DIGITAL FREQUENCY SYNTHESIS (DDFS)

The DUC is a digital circuit which implements the conversion of a complex digital baseband signal to a real pass band signal. The input complex baseband signal is sampled at a relatively low sampling rate, typically the digital modulation symbol rate. The Digital Down Converter (DDC) is the counter-part on the receiver end. The detailed description on DUC and DDC can be found. This section focuses on the efficient hardware implementation of DDFS, which is backbone of the DUC and DDC.

##### A. Mathematical Representation

Based on the differential relationship between Sine and Cosine  
i.e.

$$\frac{d}{d\theta}(\sin \theta) = \cos \theta \quad (1)$$

$$\frac{d}{d\theta}(\cos \theta) = -\sin \theta \quad (2)$$

Khan et al. [23] has suggested the equation (3) and (4), and then proposed the architecture, as shown in Fig. 1.

$$Y_c = Y_p + \Delta \theta X_p \quad (3)$$

And

$$X_c = X_p - \Delta \theta Y_p \quad (4)$$

$Y_c$  is the current value of the Sine and  $Y_p$  is the previous value of Sine. Similarly,  $X_c$  is the current value of Cosine and  $X_p$  is the previous value of Cosine.  $\Delta \theta$  is the *Frequency Control Word*. By putting equation (3) into equation (4), we get

$$\begin{aligned} X_c &= X_p - \Delta \theta (\Delta \theta X_p + Y_p) \\ X_c &= X_p - \Delta \theta^2 X_p - \Delta \theta Y_p \\ X_c &= (1 - \Delta \theta^2) X_p - \Delta \theta Y_p \end{aligned} \quad (5)$$

The value of  $\Delta \theta$  in the last equation is very small. For  $\Delta \theta$  being represented in **Qn.m** format, it is in the range of  $2^{-m} \leq \Delta \theta \leq 2^{-x}$ , where lowest possible value of  $x$  is 1. As the value of  $\Delta \theta$  is very small, so it is further reduced after taking square.. Thus its contribution becomes less significant in equation (5). Therefore  $\Delta \theta$  can be ignored in calculating the values of Cosine and Sine and the the simplified equation is

$$X_c = X_p - \Delta \theta Y_p \quad (6)$$

From equation (1) and equation (6), it is clear that the current samples of Sine and Cosine are generated using the previous values of Sine and Cosine.

Eq. 1 and 6 are initialized with known seed values of 1 and 0, i.e.  $X_i = 1$  and  $Y_i = 0$ . The values of  $\Delta \theta$  can be determined by equation (7)

$$\Delta \theta = \frac{2\pi f}{f_{clk}} \quad (7)$$

Where,  $f$  is the desired frequency, and  $f_{clk}$  is the clock frequency of the system clock.

##### B. Proposed Architecture

The proposed architecture for VLSI implementation of the DDFS is shown in Fig. 8.

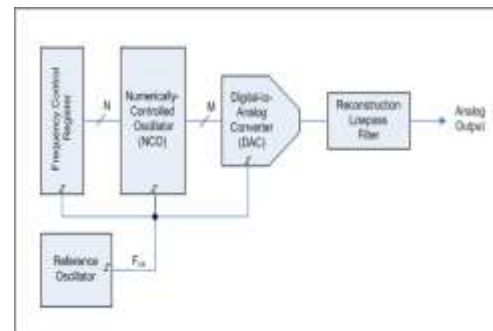


Fig.7 - Direct Digital Synthesizer block diagram

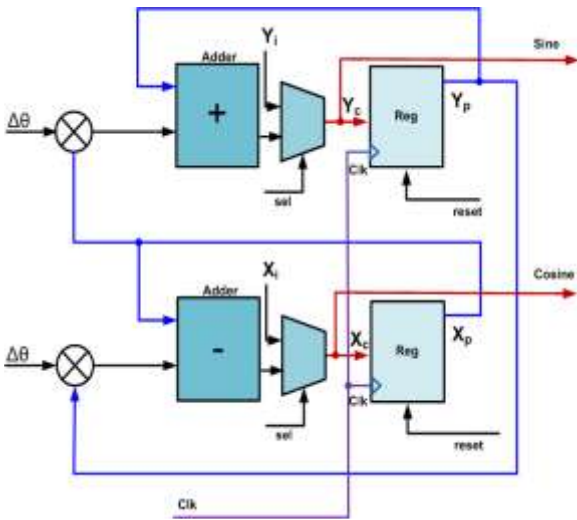


Fig. 8 The Proposed Architecture for DDFS

This architecture utilizes two adders, two multiplexers, two multipliers and two registers. The presentation is general and applicable to any bit length. The data path elements are 32 bits wide. As compared to the architecture proposed, the required number of registers has been reduced to two, instead of four. Depending upon the number of bits used, it results in considerable reduction of hardware resources.

Initially  $Y_i$  and  $X_i$  are fed to the Registers because  $sel$  is 1. On the next positive edge of  $clk$ , these seed values are used to compute the values of Sine and Cosine. After the first clock cycle,  $sel$  is 0 and now the multiplexers only act as simple wires for rest of the clock cycles. The previous value of Cosine is multiplied by  $\Delta\theta$  and added with the previous value of Sine to generate the current value of Sine,  $Y_c$ . Similarly, the previous value of Sine,  $Y_p$  is multiplied by  $\Delta\theta$  and subtracted from the previous value of Cosine,  $X_p$  to generate the current value of Cosine,  $X_c$ . The transmitter is as shown in below figure:

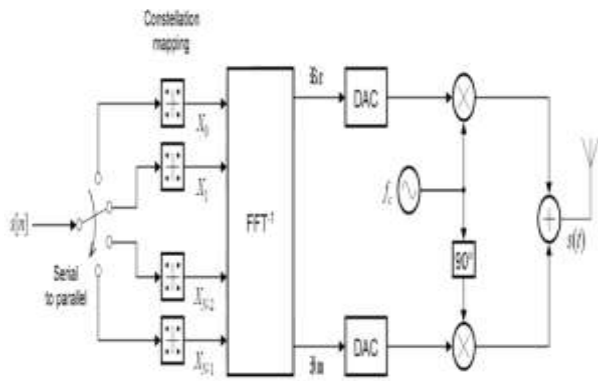


Fig.9 Transmitter block diagram

The receiver is as shown in below figure:

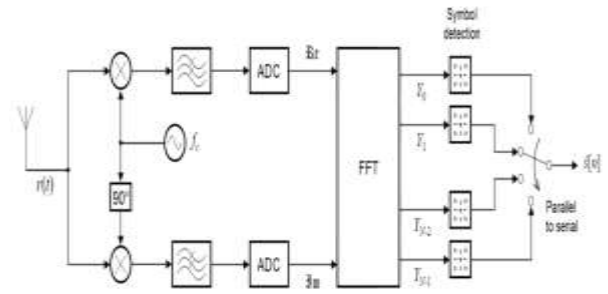


Fig.10 Receiver block diagram

The proposed OFDM Block Diagram is as shown in below figure:

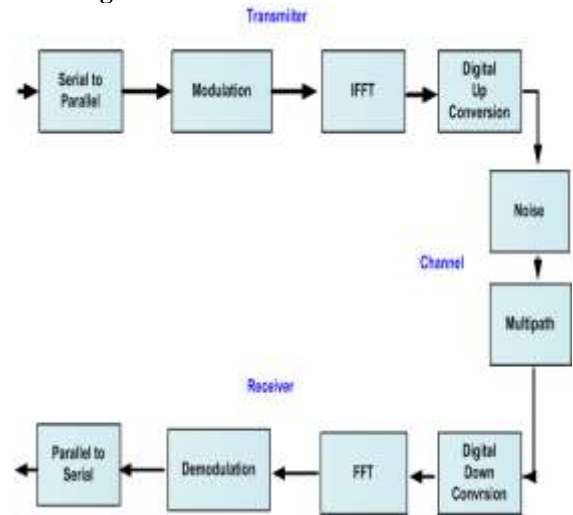


Fig. 11 Proposed OFDM block diagram

### V SIMULATION RESULTS

The simulation of proposed OFDM 8 PSK block is carried out by using Verilog HDL language in Xilinx tool.

The RTL schematics and simulation results of OFDM 8 PSK transmitter, OFDM 8 PSK receiver and for the main block are shown in below figures.

#### RTL Schematic Of OFDM 8 Psk Transmitter:

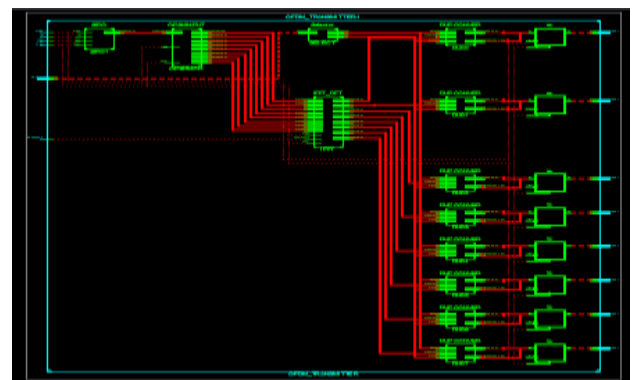


Fig.12 RTL of Transmitter

#### Transmitter Simulation Results:

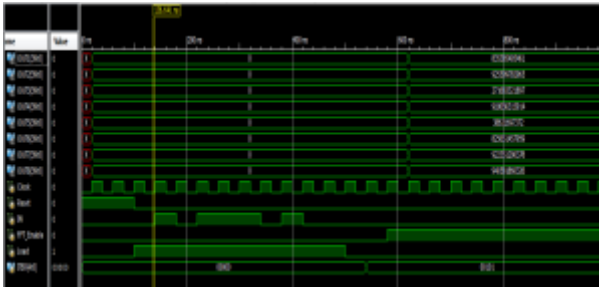


Fig.13 OFDM 8 PSK Transmitter simulation results

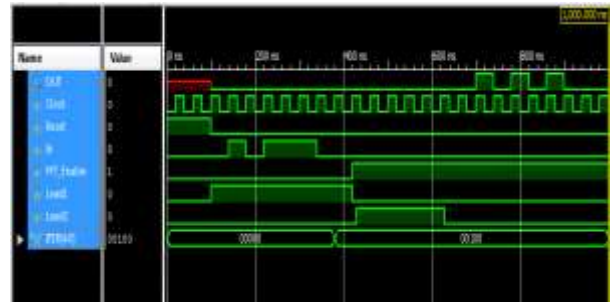


Fig.17 Simulation results of OFDM

**OFDM 8-PSK Receiver RTL:**

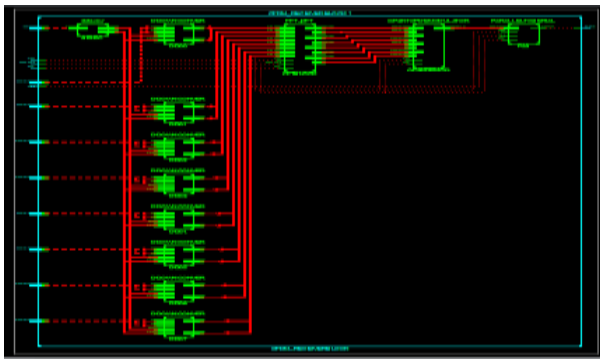


Fig.14 RTL of Receiver

**Receiver result:**



Fig.15 Simulation results of OFDM 8 PSK receiver

**OFDM RTL Schematic:**

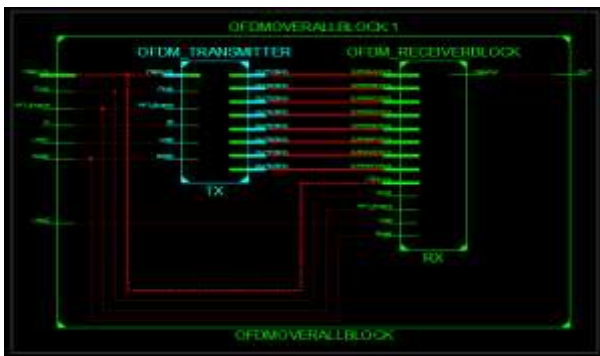


Fig.16 RTL of OFDM block

**OFDM Final Result:**

**VI CONCLUSION**

In this paper, we presented simplified hardware architecture for OFDM Software Defined Radio. This paper also presented an area and speed optimized architecture for Direct Digital Frequency Synthesis, one of the backbone for SDR. The proposed framework consumes lesser silicon area and is realizable on FPGA. This is due to the fact that it generates the sine and cosine values on a single edge of the system clock. The required memory resources are extremely less as only two 32-bit registers have been used in the architecture. The future goals include the reduction of the critical path, so that the framework may work at even higher clock rate and the range of frequency generated may be increased. Besides, the design may be optimized leading to a more intelligent framework to help in realization of a more ideal Cognitive Radio.

**REFERENCES**

- [1] W. Tuttlebee, *Software Defined Radio Enabling Technologies*. John Wiley and Sons, 2002.
- [2] X. Qi, L. Xiao, and S. Zhou, "A novel GPP-based Software-Defined Radio architecture," in *7th International ICST Conference on Communications and Networking in China (CHINACOM)*, 2012, pp. 838 – 842.
- [3] E. Nicollet, "DSP software architecture for Software Defined Radio," in *IEE Colloquium on DSP enable Radio*, 2003, pp. 1 – 9.
- [4] V. Barral, J. Rodas, J. A. Garcia-Naya, and C. J. Escudero, "A novel, scalable and distributed software architecture for softwaredefined radio with remote interactionpp. 80 – 83.," in *19<sup>th</sup> International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 80 – 83.
- [5] N. Ali, "Novel architecture for software defined radio," in *IEEE International Conference Microwaves, Communications, Antennas and Electronic Systems (COMCAS)*, pp. 1 – 4.
- [6] T. Hentschel, *T. Hentschel, Sampling Rate Conversion in Software Configurable Radios*. Artech House Mobile Communication Series, 2002.
- [7] R. B. Staszewski, K. Muhammad, and D. Leipold, "Digital RF Processor DRPTM for Cellular Phones," Dallas, TX 75243, USA.
- [8] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. John Wiley and Sons, 1999.
- [9] T. Weise, *Global Optimization Algorithms - Theory and Applications*. 2008.
- [10] W. B. Langdon, *Genetic Programming and Data Structures*. Springer, 1998.

[11] J. Miller and P. Thomson, "Cartesian genetic programming," in *Third European Conference on Genetic Programming EuroGP2000*, 2000, pp. 121–132.

[12] D. A. Sunderland, S. S. Strauch, H. Wharfield, T. Peterson, and C. R. Cole, "D. A. Sunderland, R. A. Cmos/sos frequency synthesizer lsi circuit for spread spectrum communications," *IEEE Journal of Solid-State Circuits*, vol. 19, no. 4, pp. 497–566.

#### Author's Profile:



**Aluru Koteswara Rao** received his B.Tech degree in Electronics and communication Engineering from Gokula Krishna College of Engineering, Near RTC Depot, Sullurpet

mandal, Nellore District, affiliated to JNTU Anantapur. He is currently pursuing M.Tech VLSI in Audisankara college of Engineering and Technology, Gudur(Autonomous), SPSR Nellore (Dist), affiliated to JNTU Anantapur.



**A. Anasuyamma** received her M.Tech in VLSI from Audisankara college of Engineering and Technology, Gudur(Autonomous), SPSR Nellore (Dist). She has 3 years teaching experience. She is presently working as Assistant Professor in the department of ECE Audisankara College of Engineering and Technology, Gudur (Autonomous), Affiliated to JNTU, Anantapur.