OFDM Symbol Detection Using FPGA

Mir Ahtesham Ali., Sultan Aljahdali., Nisar Hundewale.
College of Computers and Information Technology, Taif University, Taif, Saudi Arabia

mirali@tu.edu.sa, aljahdali@tu.edu.sa, n.hundewale@tu.edu.sa

Abstract - **Orthogonal Frequency Division Multiplexing is a frequency division multiplexing scheme. It is also known as multi-carrier or digital multi tone modulation. It has also been adopted for various standards including 802.11a . In this paper a mechanism is implemented to detect the beginning of the OFDM symbol based on Time synchronization.**

Keywords - **ofdm; csv; post route simulation; Lut**

I. INTRODUCTION

OFDM is a communication technique that can be seen as either a multiplexing technique or a modulation scheme. In OFDM the communication channel is divided into several equally spaced frequency bands. The user information data is divided and each subcarrier carries a portion of this information within each band. All the subcarriers are orthogonal to each other .

OFDM is a special kind of FDMA system in which the sidebands of individual subcarriers overlap, while the signals can be detected without the interference of the adjacent subcarriers. In FDMA systems there is a separation in frequency between adjacent subcarriers. On the other hand, OFDM subcarriers have overlap, but since the subcarriers are mathematically orthogonal, there is no interference between the adjacent channels. An OFDM system splits the available spectrum into N sub channels (subcarriers). If the original channel spectrum has a bandwidth of B, then each subcarrier in an OFDM system has a bandwidth of Fs=B/N. Each subcarrier will carry several bits of data using QAM or PSK technique. OFDM uses FFT and IFFT to divide the physical channel into N subcarriers [4]. Figure 1 shows the block diagram of an OFDM system. In the transmitter, the information data (input data) is encoded by either a block or convolution code for additional protection against noise and channel impairments. This encoded data is then mapped to either PSK or QAM mapping. The complex data is then fed into the input of the IFFT block. If the symbol that is assigned to the ith carrier of OFDM is defined as, where i = 0, 1,...., N-1, then the transmitted signal is represented as

$$S_a(t)=\sum_{i=0}^{N-1} ai \exp(2\pi j(f0 + ifs)t)$$

Where fo is carrier frequency and is the bandwidth of each ai sub channels. Note that each has a complex value.

After the symbols are passed through the IFFT, then a cyclic prefix is added to the output o f the IFFT. The role of cyclic prefix is very important to preserve the Orthogonality between OFDM symbols. The output samples will be converted into analog signal by using a (D/A) converter, and finally will be transmitted through the channel. At the receiver end the inverse operation is performed on the received data that contains the signal plus noise.

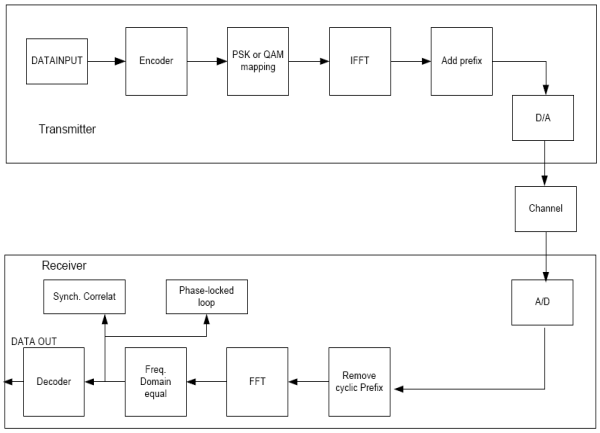

Figure 1: Structure of OFDM system

A. Cyclic prefix

Cyclic prefix are used in OFDM in order to combat the inter carrier interference and inter symbol interference introduced by the multipath channel through which the signal is propagated. In an OFDM symbol the cyclic prefix is a repeat of the end of the symbol at the beginning. The cyclic prefix should be longer or equal to the length of the channel impulse response. If the cyclic prefix is long enough, the transmitted signal is periodic and orthogonality of the frames is preserved.

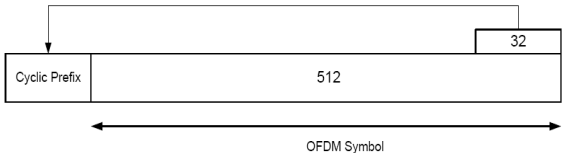

Figure 2: Cyclic prefix

B. Synchronization

Synchronization is very important in OFDM systems. In an OFDM system, phase noise and frequency offset create *inter symbol interference (IS*I) and inter *carrier* interference (ICI). The orthogonality of an OFDM system is preserved only when the receiver and transmitter use the same frequency. Otherwise, this frequency offset causes ICI and degradation in the performance of the overall system.

II. IMPLEMENTATION

A. Algorithm for symbol detection

To avoid the inter symbol interference (ISI) and to maintain the orthogonality of the OFDM signal, the last 32 samples in the frame are duplicated at the start of the frame as shown in figure 2. This technique is referred to as adding cyclic prefix. The synchronization algorithm uses the correlation between these identical data blocks to estimate the time and frequency offset.

$$\gamma\,(\mathrm{k}) = \sum_{k=M}^{m+M-1} r(k) r*(k+N)$$

M= length of the CP
N = length of the symbol
K = Sample No.
m = Sample from where we need to start correlation.

$$\mathrm{abs} = \left|\ \mathrm{Re}\{\gamma\}\ \right| + \left|\ \mathrm{Im}\{\gamma\}\ \right|$$

Symbol start is a point K in the signal where we find the max(abs).

Phase Offset = $\mathbf{tan^{-1}}\left(\frac{\mathbf{Im\{\gamma\}}}{\mathbf{Re\{\gamma\}}}\right)$ at max (abs) or symbol start.

B. Reference Model(MATLAB)

The reference model is implemented in matlab, a transmitter – channel – receiver is modeled in matlab to do the system level simulation. Major blocks constitute the memory, correlation unit and the peak detector unit. These blocks are described below.

B.1 Memory

A 544x16 bit memory is used for storing the received data, the input data consist of complex values of 16 bits, 8 bit real and 8 bit imaginary.

B.2 Correlator

Taking advantage of the cyclic prefix(part of the data in the beginning is a repeat of the data at the end) the correlation algorithm is implemented and the data in the memory is correlated with the cyclic prefix value in memory. The output of the correlator is a complex value with 22bit real and 22 bit imaginary.

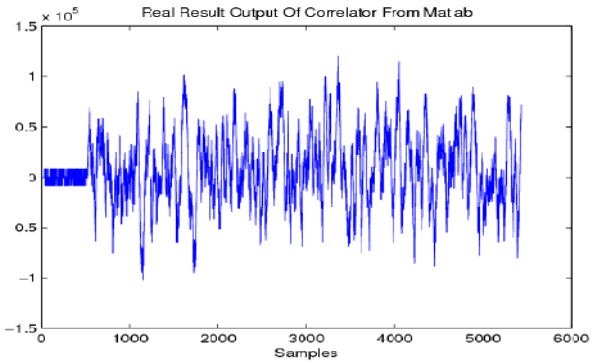

Figure 3: Real Output from correlator

B.3 Absolute

The complex value from the correlator is converted to absolute value.

Sqrt ($\mathrm{Re}\{\gamma\}^2 + \mathrm{Im}\{\gamma\}^2$)

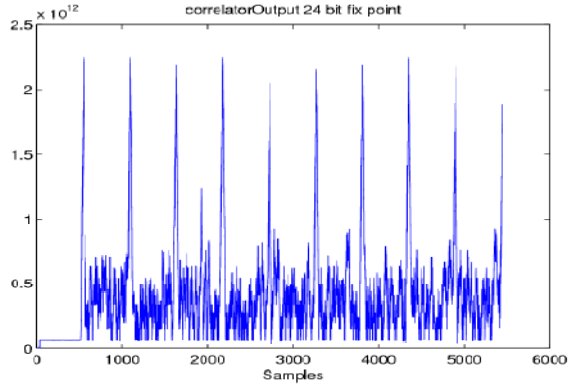

Figure 4: 24bit fixed point output of the absolute module

B 4. Peak Detector

Peak detector is used to find the maximum of the correlation result.
SymbolStart = max{ $\left|\ \mathrm{Re}\{\gamma\}\ \right| + \left|\ \mathrm{Im}\{\gamma\}\ \right|$ }

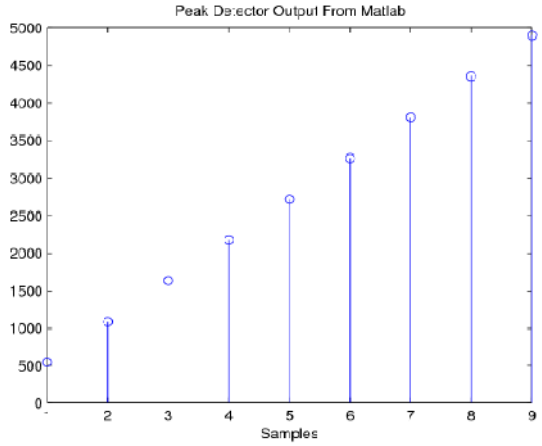

Figure 5: Output of the peak detector

From the figure 5 it can be noted that the first symbol start is detected at 544 samples. All other Proceeding samples are 544 samples apart. Can be seen in 24bit fixed point output of the absolute module

III. HARDWARE IMPLEMENTATION

A . Hardware Architecture.

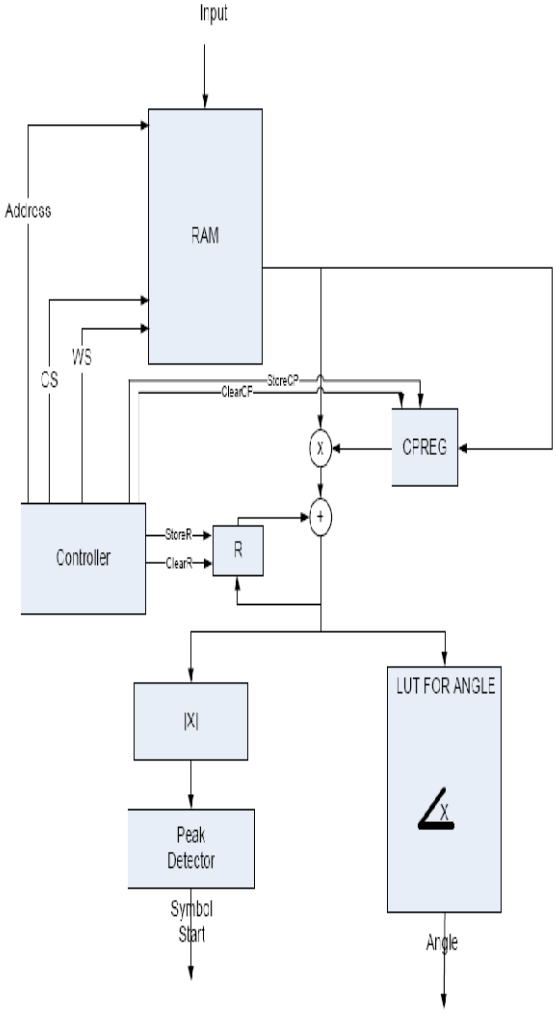

Figure 6: Hardware architecture of OFDM symbol detector

B. Architecture Implementation

Main modules of the OFDM Symbol Detector architecture are:

1. Correlator
2. Controller
3. Memory
4. Absolute Module
5. Peak detector
6. LUT

B.1 Correlator

The correlator module is used for correlating the input Data with the cyclic prefix, in order to detect the start of the symbol. It constitutes of a complex adder, complex multiplier, 2 Flip-flops (CP flip-flop and Result Flip-flop) controlled by the controller.

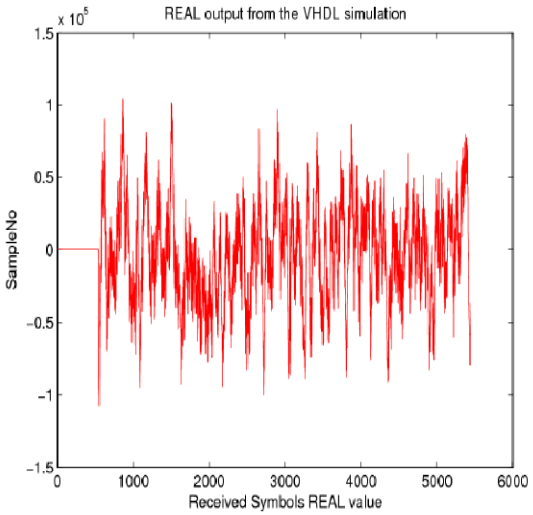

Figure 7: Correlator output real from vhdl

B.2 Controller

Controller module is responsible for controlling the entire block in the design and it is implemented with a Finite State machine (FSM) as shown in figure 8. It consist s of the following states. Init, Waitstate, WriteSRAM, ReadCP, StoreCP, ReadData, StoreData, Result.

1. Init: Initializes registers and counters to initial state.

2. Waitstate: Waits for next sample indicated by asserted valid_i.

3. WriteSRAM: writes in to the RAM.

4. ReadCP: reads data from the RAM. Reference by the CP address.

5. StoreCP: stores the data fetched in the readCP state in a register.

6. Read Data: reads data from the RAM. Referenced by the Data Address.

7. StoreDATA:stores the data fetched in the ReadData state.

8. Resultstate: Calculates the correlation and if the calculation is completed asserts valid_o.

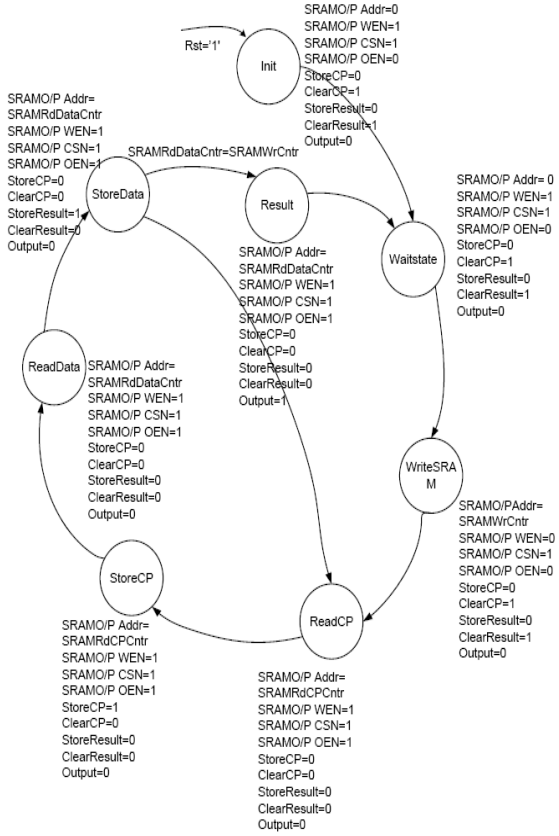

Figure 8: State machine for controller

B.3 Memory

The memory module is used for storing the received OFDM symbols, the memory module is a 544 x 16 bit RAM generated using the memaker tool. The functioning of this memory is controlled by a controller.

B.4 Absolute Module

The correlator output consists of the real and imaginary outputs, we use an absolute module to calculate the absolute value which is given to the peak detector for detecting the peaks. The trade-off between computational complexity and performance is crucial. Minor changes in the performance often lead to large reductions in hardware. i.e. calculation of the abs(γ) value is simplified. Since we are only interested in comparing the numbers, abs(γ2) will give the same result. The computation sqrt(Re{γ}2+ Im{γ}2) will be reduced to Re{γ}2+ Im{γ}2. Simulations show that abs can be further reduced to {|Re{γ}|+|Im{γ}|} without too much loss. The algorithm implemented in hardware is

$$\hat{\theta} = \{|\mathrm{Re}\{\gamma\}| + |\mathrm{Im}\{\gamma\}|\}$$

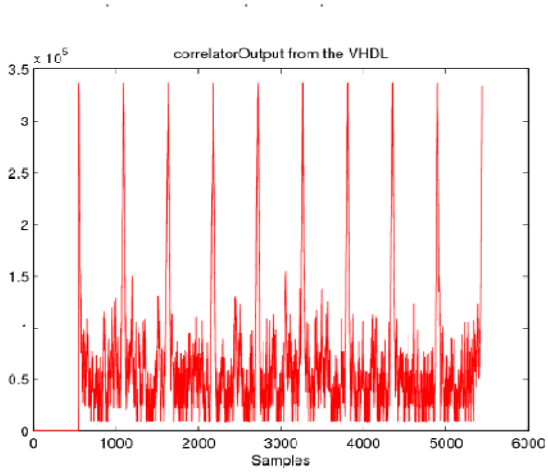

Figure 9: vhdl output of the absolute module

B.5 Peak Detector

Peak detection is performed by a Finite state machine (FSM) as shown in figure 10. The FSM detects the symbol start by comparing the amplitude difference of the correlation, with a predefined threshold and validates the symbol timing by comparing the amplitude difference with another predefined threshold. Heuristics method is applied by using two thresholds.

As shown in the figure 10, in the Wait state the amplitude of the correlation is compared with threshold 1. If it is greater than or equal to threshold 1 then it enters the Detect state. The incoming amplitude of the correlation in Detect state is stored as prev value and as long as this value is less than the pres value, it is in Detect state. And when the prevvalue is greater than or equal to the presvalue then it goes to the stable state. In this state is the prev value is less than the threshold2 then it enters the wait state.

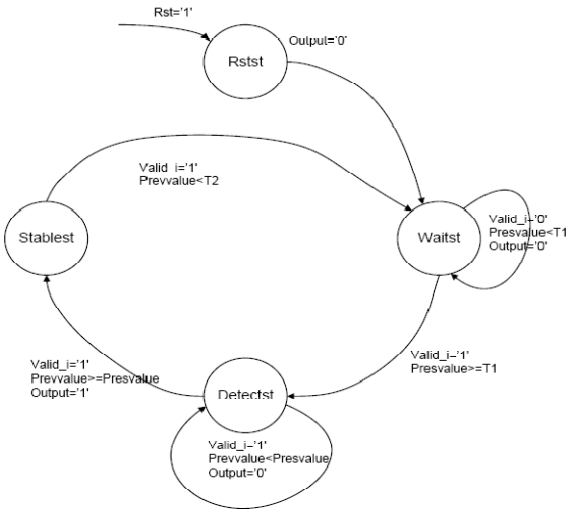

Figure 10: State machine for peak detector

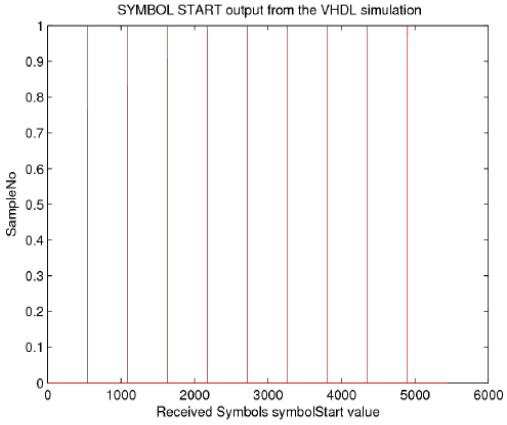

Figure :11 Peak detector output from vhdl

The output of the peak detector from vhdl which shows '1' when we have max(abs). This is then plotted in matlab showing corresponding sample where it is high.

B.6 Look Up Table

It consists of pre calculated values from the equation below

$$\text{LutData}=\tan^{-1}\left(\frac{\text{Im}\{\gamma\}}{\text{Re}\{\gamma\}}\right)$$

Where both real and imaginary are of width 5 bits. So the address of the LUT is of 10 bit.
Address = {real (4 down to 0) imag(4 down to 0)}. The Output of the LUT is shown in the figure below.

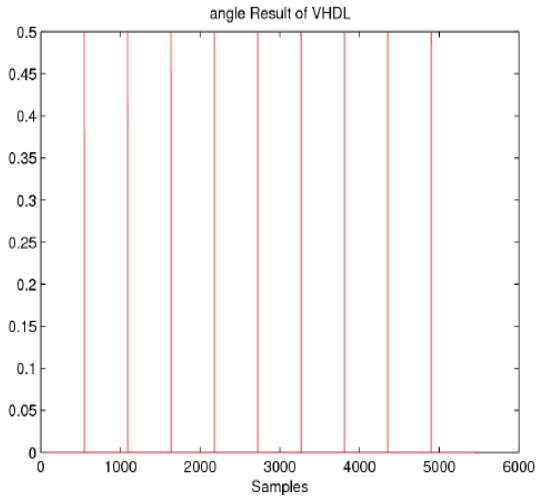

Figure 12: Output of LUT from vhdl

The above is the result from the VHDL simulation. The result is the corresponding angles at the point where the symbol starts. This was simulated with data having phase offset of 0.5

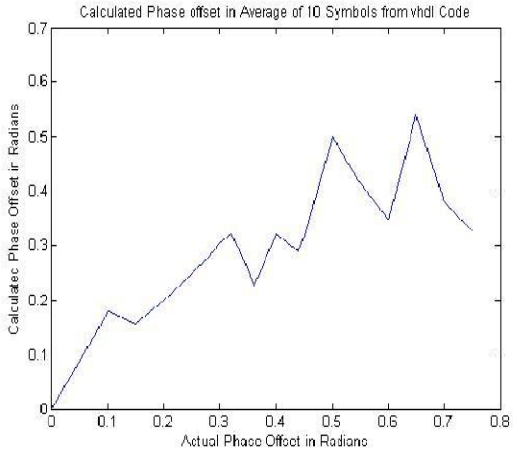

Figure 13 : Phase offset

X-axis represents the actual angle transmitted and y- axis shows the LUT output. As we can see from the above figure that there is an error in the output, this is shown in the figure below.

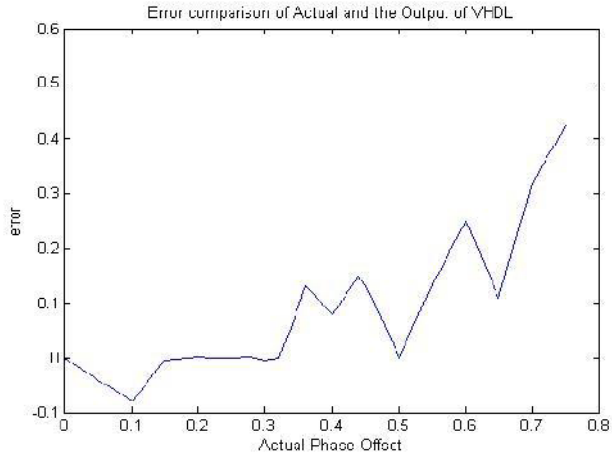

Figure 14:Error result from LUT

As seen from the figure, up to a phase offset of 0.4 errors is less, after 0.4 the error exceeds drastically, and at 0.8 the error is about 0.4, which is very large. And needs to be improved.

IV.PREPARATION FOR VERIFICATION

A. Synthesis and Post Route simulation

The synthesis and post route simulation has been done using the Xilinx ISE tool version 10.1 .synthesis process translates the VHDL code in to a device net list format. The design was synthesized and possible errors and warnings in the synthesis report were overcome. Special focus was on removing latches in design as they affect the functionality of the design.

B. Timing Summary

Speed grade: -4 Minimum period: 8.899ns Minimum input arrival time before clock: 2.700ns Maximum output required.Time after clock: 46.440ns Maximum combinational path delay: 7.882ns (Note: Max frequency without DCM was 22.02 MHz) Post-Route simulation models interconnect delay, as well as gate delay. This type of simulation will most accurately match the behavior of the actual hardware. After synthesis we generate the post place and route simulation model and selected post-Route simulation on the source for drop down list and run the Post-Route simulation model and verified the simulation results with that of the Matlab reference model.

C. UCF (User Constraint File)

The UCF file is an ASCII file specifying constrains on the logic design. We have used PACE under the Xilinx Constraints Editor to create constraints within a UCF file . This file is useful for specifying the inputs and output pins to the board. By using the Data sheet of Spartan 3 FPGA board, we know which pins are dedicated and which are free for the user to be used as I/O and clock for test signal.

D. CSV (comma separated vector)

The sequences of samples which are used for testing are generated by pattern generator and these samples are to be specified to the pattern generator in the .CSV file, these-axise you begin to format your paper, first write and samples are specified in the hexadecimal values. This file has been generated using Matlab.

V. FPGA VERIFICATION

While simulation offers an important level of design verification, it does not provide sufficient understanding of the design performance in the system because the stimulus is just an approximation of the actual physical environment .FPGA verification offers the only mean of understanding what is happening inside the FPGA as it operates in the system at clock speed .

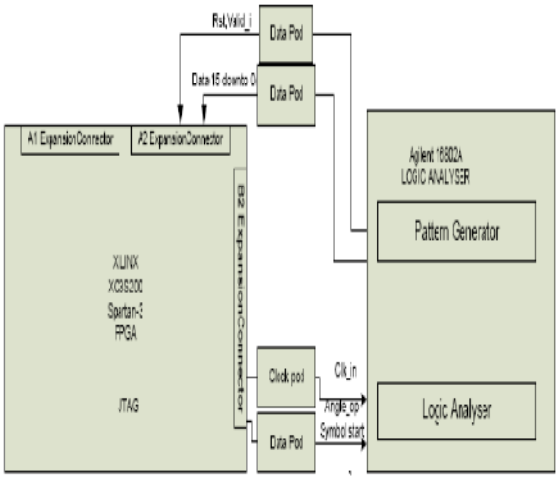

Figure 15: Block diagram of logic analyzer setup

A. Logic Analyzer

The design is verified on Agilent 16802A portable logic Analyzer .It consists of 68 channels and up to 1M acquisition memory depth. It consist of inbuilt pattern generator , which is used to generate a sequence of samples given as input to the design , the data to be generated by the pattern generator is given as input file to the pattern generator which is in .CSV format created in Matlab.

B. Verification

In our design we are using an internal clock of 10MHz from the FPGA board (generated using DCM) and fed back to the logic analyzer using a clock pod. And the input samples from pattern generator are triggered on this clock. The input data sequence which consists of 16 bit complex data, 1 bit reset and 1 bit valid_i are generated from pattern generator and are fed at A2 expansion port using the data pods, as shown in figure15. The output data sequence which consists of 8 bit angle output and 1 bit symbol start value are fed to the logic analyzer from the B2 expansion ports using data pods, as shown in figure.2.Project was created with the following specifications: family Spartan 3, Device XC3S200, package FT256 and speed 4.

Project files were added, synthesized, design implemented and .bit file was generated using Generate programming File tab. The design which is now in bit format is loaded in to the FPGA by using the ISE IMPACT tool. Connections from the logic analyzer and pattern generator are given to the FPGA board as specified in the .UCF file, as shown in figure15.

C. Verification procedure

The CSV file is generated in matlab for different angles for (0.1 to 0.9). This file is imported to the pattern generator by using the tab FILE → Import. The Bus signal and the sampling frequency of the Logic analyzer are set as shown in the below figure.15. We run the pattern generator and observe the simulation waveform in Logic analyzer. Output waveforms from the Logic Analyzer are shown in figure 17 and figure19.

VI. RESULTS

Symbol start and Angle result From the Logic Analyzer

Below are the outputs of Logic analyzer, the results here are shown in Hex but actual output from the design is in 8 bit unsigned binary number, which when converted to decimal and divided by 27 (The LUT stores the angle in fix point which has been multiplied by 27 in matlab, so here it is divide by 27) and then converted to degrees by dividing it by π gives the corresponding angle. The input data (I.e. CSV file) generated corresponds to specific angle, for verification purpose the angle output from the

logic analyzer should be same as the angle of the input data. The figure 16 below **shows Post-Route** simulation results for 0.5 phase shift and the figure 17 is the angle output for the data that has a phase shift of 0.5, the calculation for this angle is shown below.

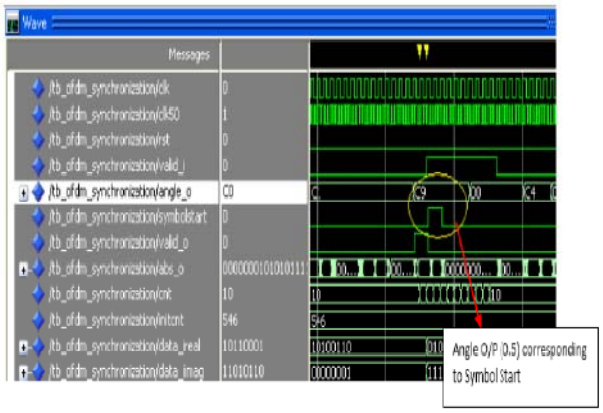

Figure 16: Post Route simulation result for 0.5 phase shift

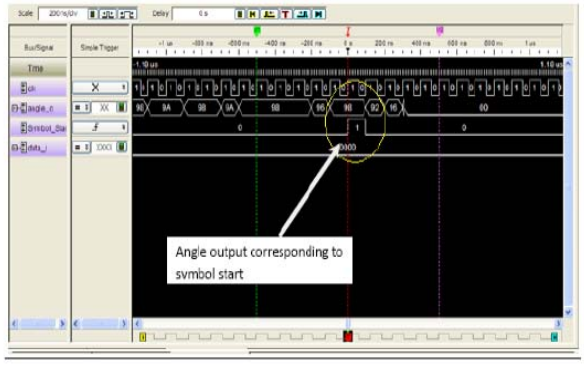

Figure 17: Logic Analyzer output for 0.5 phase shift in symbol

LUT output at the point where we receive symbol start is Hex=$C9$ which is equal to an Angle of 0.4997.

$(C9)16$= $(201)10$ *which is in fixed point*

$LUT\ output\ in\ floating\ point$=20127=1.57

$Angle\ Value$=1.57π=0.4997.

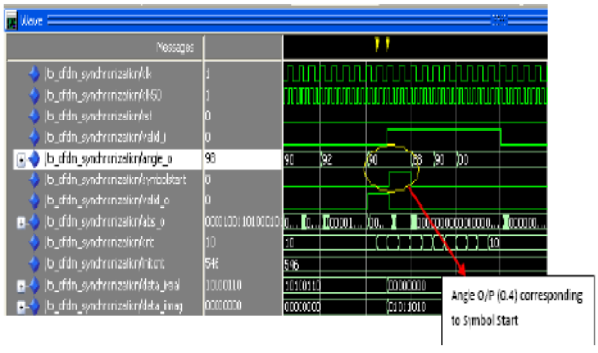

Figure 18: Post-Route simulation result for 0.4 phase shift

The figure 18 is the angle output for the data that has a phase shift of 0.4, the calculation for this angle is shown below.

LUT output at the point where we receive symbol start is Hex=98 which is equal to an Angle of 0.3779

$(98)16$=$(152)10$ *which is in fixed point*

The Above calculations show how the output is represented in the real angle value from LUT. $LUT\ output\ in\ floating\ point$=15227=1.1875

$Angle\ Value$=1.1875π=0.3779.

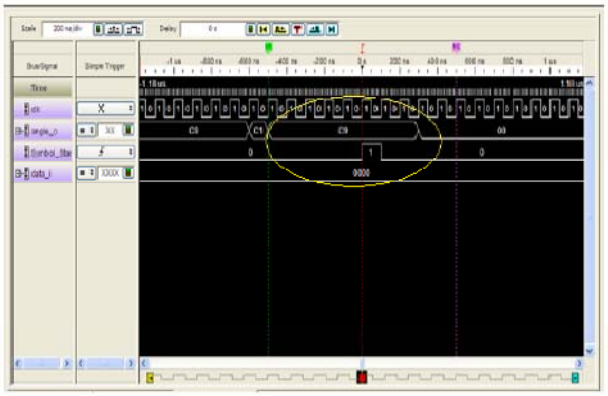

Figure 19:Logic Analyzer output for 0.9 phase shift in symbol

VII. CONCLUSION

The Results estimated from the system was observed to be correct for limited amount of phase shift from 0.35 -0.55 due to limitations of the LUT (LUT used in the design which is 2^{10}=1024 words wide. To detect phase shift from 0-1 we would require a 232=4294967296 words wide LUT. Therefore it is better to use Cordic for angle estimation). The design has been verified on FPGA and it can be used for fabrication. The system can be used for OFDM symbol detection in less ISI.

REFERENCES

[1] Jan-Jaap van de Beek,Magnus Sandell and Per Ola B¨orjesson "ML Estimation of Time and Frequency Offset in OFDM Systems" IEEE Transaction on signal processing Vol.45,No.7,July 1997.

[2] Johansson, Martin Nilsson, Peter Nilsson"An OFDM Timing Synchronization ASIC"Department of Applied Electronics, Lund University IEEE Electronics,Circuits and systems 2000.

[3] J. A. C. Binghum, “Multicarrier Modulation for Data Transmission: An Idea Whose Time Has Come,” IEEE Communication Magazine, vol. 28, pp. 5–14, May 1990.

[4] Bruce Fette, Roberto Aiello, PraphulChandra, DanielM. Dobkin, Alan Bensky, Douglas Miron, David A. Lide, Farid Dowla and Ron Olexa."RF & Wireless Technologies"pages(803-813).

[5] Saberinia, E.; Tang, J.; Tewfik, A.H.; Parhi, K.K.; Dept. of Electr. & Comput. Eng., Univ. of Nevada, Las Vegas, NV “ Pulsed-OFDM Modulation for Ultrawideband Communications". IEEE Vehicular Technology Society , April 2008.

[6] Wu Yan; Sumei Sun; Zhongding Lei; Inst. for Infocomm Res., Singapore. “ A low complexity VBLAST OFDM detection algorithm". Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference.

[7] Reddy,P.S; reddy, G.R “ Performance Comparision of Autocorrelation and CPRDIC Algorithm Implemented on FPGA for OFDM based WLAN”.ICCSN ‘09

[8] Dick, C.; Harris, F.; “FPGA implementation of an OFDM PHY” IEEE Signals,Systems and computers, 2003.

[9] Garcia, J.; Cumplido,R.; “On the design of an FPGA-based OFDM modulator for IEEE 802.16”. IEEE Reconfigurable computing and FPGA, 2005.

[10] Merlyn,M.; FPGA implementation of FFT processor with OFDM transceiver”IEEE Signal and Image processing (ICSIP).2010.

[11] M.Faulkner.”FPGA Implementation of an OFDM-WLAN Synchronizer”, Second IEEE International Workshop on Electronic Design Test and Applications, 2004.

[12] P.Nilsson.”An OFDM timing synchronization ASIC”,ICECS 2000 7th IEEE International Conference on Electronic Circuits and Systems(Cat No 00EX445) ICECS-00, 2000.

[13] B.Tarokh.”Construction of ofdm m-gam sequences with low peak-to-average power ratio”,IEEE Transactions on Communications,1/2003.

[14] Equardo Marques.”Executing Algorithm for Dynamic Dataflow Reconfigurable Hardware-The Operators Protocol”,2006 IEEE International Conference on Reconfigurable Computing and FPGAs (Reconfig 2006),09/2006.

[15] Wen, J.H..”Frame Synchronization, channel estimation scheme and signal compensation using regression method in OFDM systems”,Computer Communications,20080625.