**Report on Implementation of OFDM Transmitter using FPGA**

**Submitted By:**

**Kiran Kumar (UB# 50167774)**

**Darshan Godabanahal Malleshappa (50170064)**

**Mahathi Gottapu(50169663)**

**Behara Venkata Kali Suguna Praneeth(50169360)**

**ABSTRACT**

In wireless and mobile communications, multipath fading severely degrades the quality of information exchange. The orthogonal frequency division multiplexing (OFDM) technology is able to provide a high transmission data rate with enhanced communication performance at a relatively small bandwidth cost, together with proper estimation and compensation of channel effects. Therefore, it has been widely applied in many wireless and mobile networks, especially for the state-of-the-art communication standards. The unique structure of OFDM signals and the application of discrete Fourier transform (DFT) algorithm have significantly simplified the digital implementation of OFDM system. Among different kinds of implementations, field programmable gate array (FPGA) is a very cost-effective and highly flexible solution, which provides superior system performance and enables easy system upgrade.

In this project, a baseband OFDM transmitter system is designed and implemented using the FPGA technology. Starting from a brief theoretical study, this project in detail describes the system design and verification on the basis of hardware implementation. We used Quadrature Phase Shift Keying (QPSK) modulation and the system is simulated in ModelSim-Altera. Then, Altera FFT MegaCore Function has been used to generate the Inverse Fast Fourier Transform (IFFT) and a fixed-point model is created. Subsequently entire system is synthesized and implemented within Quartus II tools and targeted to DE-i150 board containing the Cyclone IV FGPA device family.

**Orthogonal frequency-division multiplexing (OFDM)**

It is a method of encoding digital data on multiple carrier frequencies. OFDM has developed into a popular scheme for wideband digital communication, used in applications such as digital television and audio broadcasting, DSL Internet access, wireless networks, powerline networks, and 4G mobile communications.

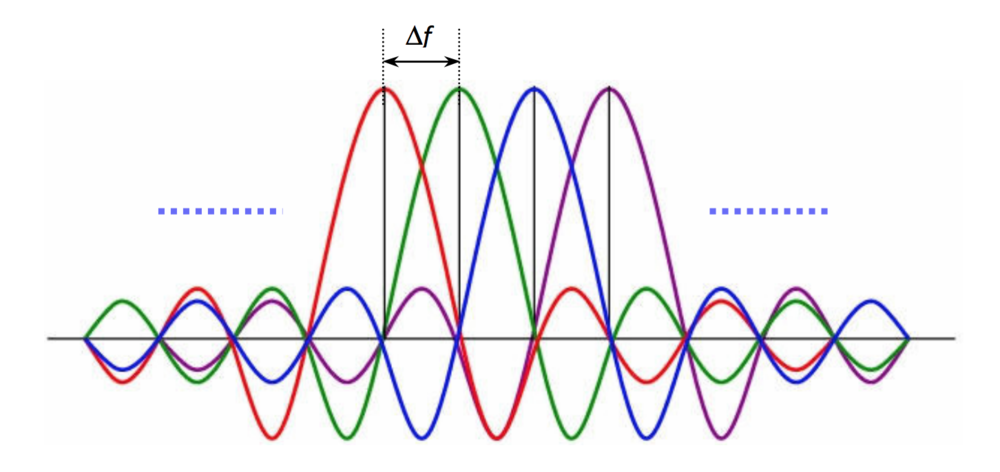
OFDM is a frequency-division multiplexing (FDM) scheme used as a digital multi-carrier modulation method. A large number of closely spaced orthogonal sub-carrier signals are used to carry data on several parallel data streams or channels. Each sub-carrier is modulated with a conventional modulation scheme (such as quadrature amplitude modulation or phase-shift keying) at a low symbol rate, maintaining total data rates similar to conventional single-carrier modulation schemes in the same bandwidth.

The primary advantage of OFDM over single-carrier schemes is its ability to cope with severe channel conditions (for example, attenuation of high frequencies in a long copper wire, narrowband interference and frequency-selective fading due to multipath) without complex equalization filters. Channel equalization is simplified because OFDM may be viewed as using many slowly modulated narrowband signals rather than one rapidly modulated wideband signal. The low symbol rate makes the use of a guard interval between symbols affordable, making it possible to eliminate intersymbol interference (ISI) and utilize echoes and time-spreading (on analogue TV these are visible as ghosting and blurring, respectively) to achieve a diversity gain, i.e. a signal-to-noise ratio improvement. This mechanism also facilitates the design of single frequency networks (SFNs), where several adjacent transmitters send the same signal simultaneously at the same frequency, as the signals from multiple distant transmitters may be combined constructively, rather than interfering as would typically occur in a traditional single-carrier system.

The main concept in OFDM is orthogonality of the subcarriers. The different orthogonal signal waveforms that differ in frequency are given by

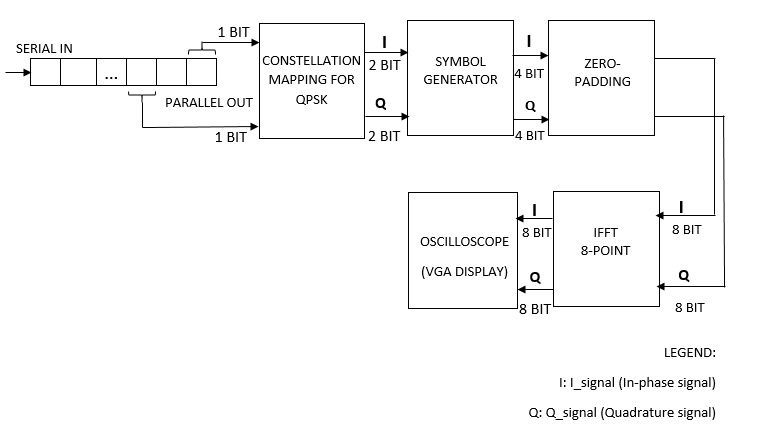
Sm(t)= p(t)cos 2π fc( t +2πmΔft )= p(t)cos 2π fm( t )

The orthogonality condition of the two signals in OFDM is given by



**OFDM subcarriers**

**IMPLEMENTATION**



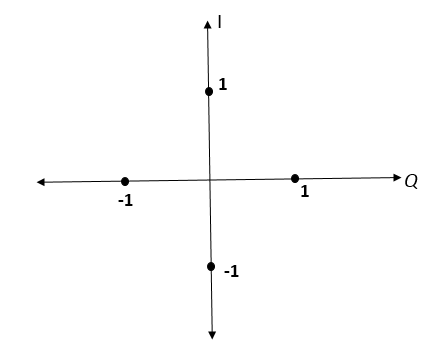
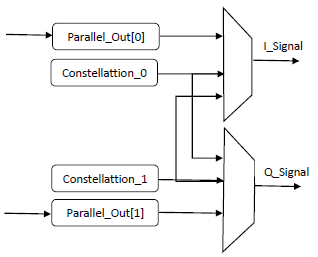
**OFDM Transmitter Block Diagram**

**Serial-in Parallel-out (SIPO)**

Here two parallel outputs are used, parallel\_out [0] and parallel\_out [1]. This configuration allows conversion from serial to parallel format. Once the data is clocked in, it may be either read off at each output simultaneously or it may be shifted out.

**Constellation Mapping for QPSK**

In this project Quadrature Phase Shift Keying (QPSK) is used as a digital modulation technique. In the QPSK constellation two bits are required to represent a point. The outputs of SIPO may be 1 bit 0 or 1. These outputs are used as select lines for mux to select the values of I\_signal (In-phase) and Q\_signal (Quadrature) components on the constellation graph. If the select line is 1, 2 bits 1 is chosen (01). If the select line is 0, 2 bits -1 is chosen (11). The outputs of QPSK modulator will be these I\_signal and Q\_signal.

****

**Constellation Mapping and Constellation Graph**

We are using 8 point IFFT in our OFDM transmitter and we are considering 1 bit data at each point. So, a total of 8 bits are required as input to this IFFT block. But, the length of the I\_signal (In-phase) and Q\_signal (Quadrature) is only 2 bits each. To solve this problem we apply concatenation followed by zero-padding on these signals.

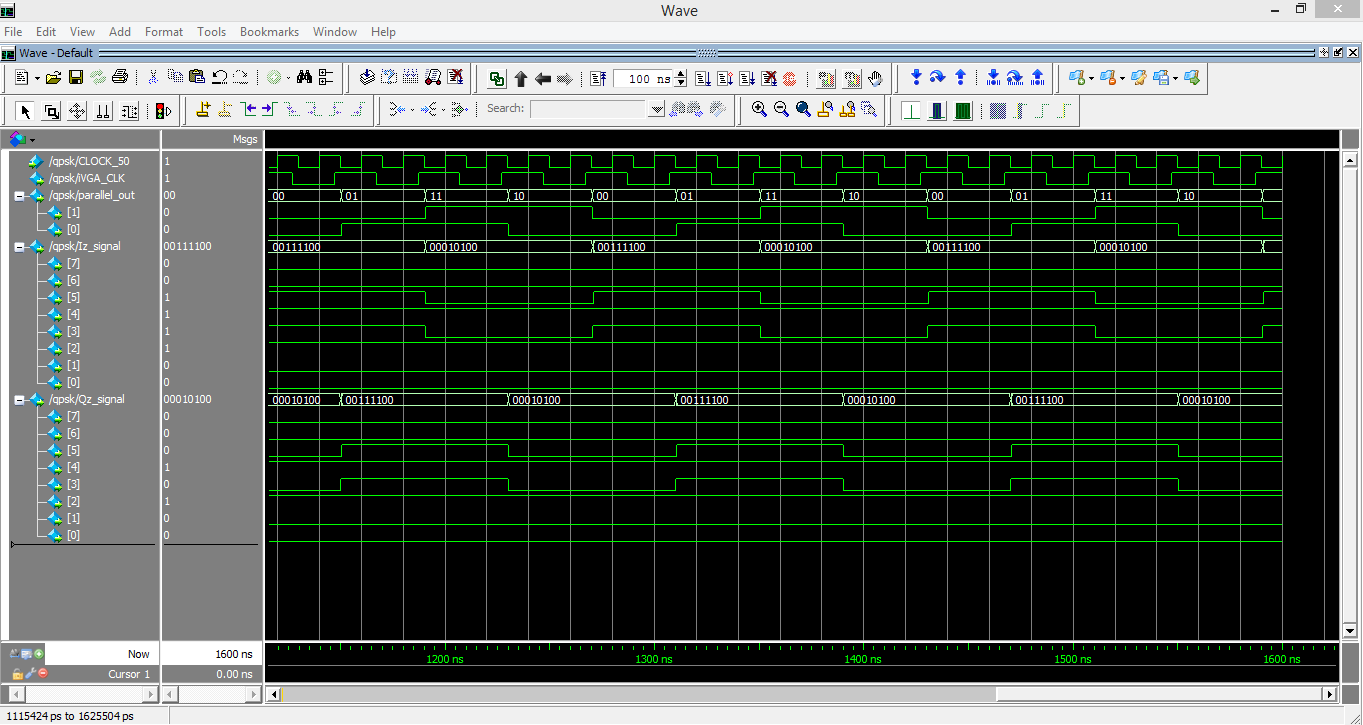
**Symbol generator**

It is used to concatenate 2 bits of data 2 times to make it 4 bits. Hence the outputs of QPSK modulator are concatenated 2 times.

**Zero-padding**

It is used to make 4 bits data to 8 bits. Here 2 bit zero’s are added at the beginning and at the end of signal generator output respectively. Zero padding doesn’t change the Fourier transform, it only increases the density of the samples. This style of zero-padding will help us distinguish between signals.

Finally Iz\_signal (8 bits Inphase signal) and Qz\_signal (8 bits Quadrature phase signal) are fed as inputs to the 8-point IFFT block. In our project all the above mentioned operations are carried out in **qpsk.v** module.



**Simulation of I and Q signals after zero-padding using Model Sim tool**

**FFT IP Core**

The FFT IP core is a high performance, highly-parameterizable Fast Fourier transform (FFT) processor. The FFT IP core implements a complex FFT or inverse FFT (IFFT) for high-performance applications.

The FFT MegaCore function implements:

* Fixed transform size FFT
* Variable streaming FFT

In our project we have implemented Variable streaming IFFT.

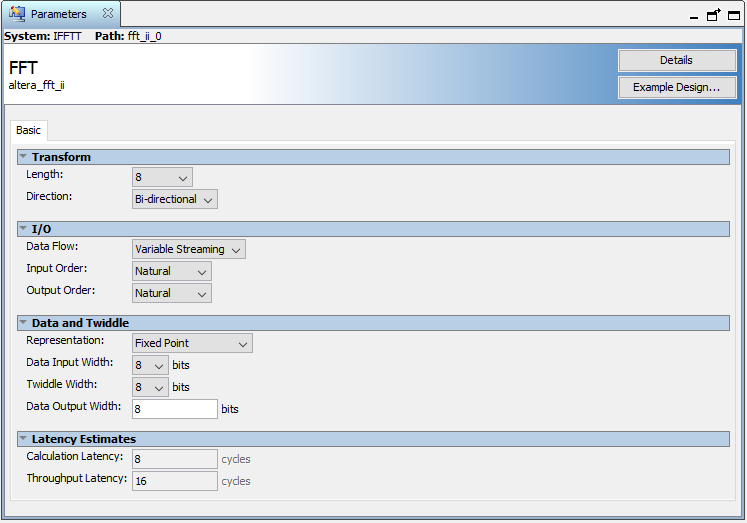
**Fixed-Point Variable Streaming IFFTs:**

* Fixed-point representation or Single-precision floating-point.
* Bit-accurate MATLAB models.
* Radix-22 single delay feed-back implementation for fixed-point IFFT.
* Input and output orders: natural order, bit-reversed or digit-reversed, and DC-centered

(-N/2 to N/2).

* Support for 8 to 32-bit data and twiddle width (foxed-point FFTs).
* Transform direction (FFT/IFFT) specifiable on a per-block basis.

In our project we have generated the IP core by specifying IP core options and parameters in the parameter editor as shown in the below figure.



Here we have assigned for I/O Data flow parameter as Variable Streaming and Transform length as 8 (for Variable Streaming this value is the maximum FFT length). The IFFT operation is configured to accept 8-bit natural order inputs and produces 8-bit natural order outputs due to this the IFFT will be operating in Engine with bit reversal mode, which increases the throughput latency to 16 clock cycles. For Data Representation parameter we have selected Fixed Point Representation.

**Avalon-ST interface signals**

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal Name** | **Direction** | **Size** | **Description** |
| clk | Input | 1 | Clock signal that clocks all internal FFT engine components. |
| reset\_n | Input | 1 | Active-low asynchronous reset signal. This signal is detected on the rising edge of clk, and it must be asserted at least one clk clock cycle. Therefore, reset\_n can be safely deasserted on or following the clk rising edge that follows the first clk rising edge after reset\_n assertion. |
| sink\_eop | Input | 1 | Indicates the end of the incoming FFT frame. |
| sink\_error | Input | 2 | Indicates an error has occurred in an upstream module, because of an illegal usage of the Avalon-ST protocol. The following errors are defined:   * 00 = no error * 01 = missing start of packet (SOP) * 10 = missing end of packet (EOP) * 11 = unexpected EOP   If this signal is not used in upstream modules, set to zero. |
| sink\_imag | Input | 8 | Imaginary input data, which represents a signed number of data precision bits. |
| sink\_ready | Output | 1 | Asserted by the FFT engine when it can accept data. It is not mandatory to provide data to the FFT during ready cycles. |
| sink\_real | Input | 8 | Real input data, which represents a signed number of data precision bits. |
| sink\_sop | Input | 1 | Indicates the start of the incoming FFT frame. |
| sink\_valid | Input | 1 | Asserted when data on the data bus is valid. When sink\_valid and sink\_ready are asserted, a data transfer takes place. |
| source\_eop | Output | 1 | Marks the end of the outgoing FFT frame. Only valid when source\_valid is asserted. |
| source\_error | Output | 2 | Indicates an error has occurred either in an  upstream module or within the FFT module  (logical OR of sink\_error with errors generated in the FFT). |
| source\_imag | Output | 8 | Imaginary output data. |
| source\_ready | Input | 1 | Asserted by the downstream module if it is able to accept data. |
| source\_real | Output | 8 | Real output data. |
| source\_sop | Output | 1 | Marks the start of the outgoing FFT frame. Only valid when source\_valid is asserted. |
| source\_valid | Output | 1 | Asserted by the FFT when there is valid data to output. |
| fftpts\_in | Input | 4 | The number of points in this FFT frame. If this value is not specified, the FFT cannot be a variable length. The default behavior is for the FFT to have fixed length of maximum points. Only sampled at SOP. |
| fftpts\_out | Output | 4 | The number of points in this FFT frame synchronized to the Avalon-ST source interface. Variable streaming only. |
| Inverse | Input | 1 | Inverse FFT calculated if asserted. Only sampled at SOP. |

The outputs of the QPSK block I\_Signal and Q\_Signal is connected to sink\_real and sink\_imag of the IFFT block respectively. And the outputs of the IFFT block source\_real and source\_imag is given as input to the Oscillator.

**call.v**: This module operates the FFT megafunction block. The inputs sink\_error, sink\_imag, fftpts\_in and source\_ready are controlled from this block.

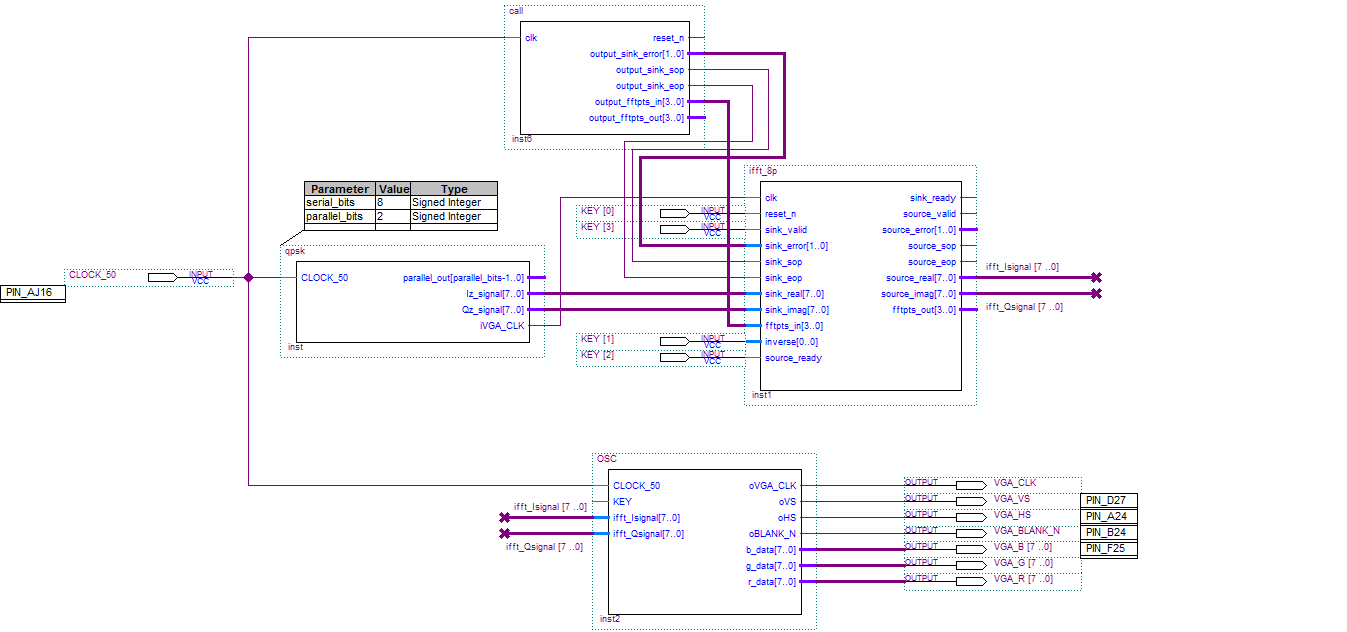
**ifft\_8.v**: This module instantiates IFFT block.

**Oscilloscope**

A basic digital oscilloscope and a signal generator using DE2i-150 and VGA monitor is used.

**OSC.v**: This is the oscilloscope module which

* Instantiates the signal generator module.
* Has the display constraints for the coordinates X and Y
* Instantiates the VGA module.
* **Signal\_generator.v:** Generates the required signal to display I and Q signals.
* **VGA\_Controller.v:** Calculates the coordinates and assigns the sync for the VGA display.



**Top level design of OFDM Transmitter**

**References:**

[1] M.Narasimhulu & Mr.P.Sravan Kumar ‘Hardware Implementation of OFDM Transmitter and Receiver Using FPGA’ International Journal & Magazine of Engineering, Technology, Management and Research Volume No: 2 (2015), Issue No: 3 (March).

[2] Hongyan Zhou ‘Design and FPGA Implementation of OFDM System with Channel Estimation and Synchronization’ Concordia University Montréal, Québec, Canada June 2013.

[3] Tirumala Rao Pechetty, Mohith Vemulapalli ‘An Implementation of OFDM Transmitter and Receiver on Reconfigurable Platforms’ International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

Vol. 2, Issue 11, November 2013.

[4] Naveen Kumar N, Rohith. S & H Venkatesh Kumar ‘FPGA Implementation of OFDM Transceiver using Verilog - Hardware Description Language’ International Journal of Computer Applications (0975 – 8887) Volume 102– No.6, September 2014.

[5] https://www.altera.com/literature/ug/ug\_fft.pdf