Hardware Implementation of OFDM Baseband Transmitter

June 26, 2013

A THESIS

to be submitted by

Aluka Abhinav Ram

for the award of the degree

of

BACHELOR OF TECHNOLOGY

under the guidance of

Prof Nitin Chandrachoodan

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, MADRAS

CHENNAI-600036

CERTIFICATE

This is to certify that the thesis titled **"Hardware Implementation of OFDM Base-band Transmitter",** submitted by Mr. Aluka Abhinav Ram, to the Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai for the award of the degree of Bachelor of Technology, is a bonafide record of research work done by him under my supervision. The contents of this thesis, in full or parts, have not been submitted to any other institute or university for the award of any degree or diploma.

Dr. Nitin Chandrachoodan
Project Guide
Assistant Professor
Dept. of Electrical Engineering
IIT Madras, Chennai-600036
Place: Chennai
Date: 8th May 2013

ACKNOWLEDGMENT

I would like to express my deepest gratitude to Dr. Nitin Chandrachoodan for giving me this opportunity to work under his able guidance. His insights into various aspects of the project have been valuable and the constant motivation at all points have encouraged me to take this forward. Working for this project has improved my technical skills and broadened my understanding of many aspects of Electrical Engineering.

I would also like to thank Prof. Andrew Thangaraj and Prof. Radha Krishna Ganti for helping us understand the project from a communications perspective and hence making the implementation easier. They constantly looked over the project, clearing our doubts whenever we had any and suggested methods when we were totally lost trying to implement the various blocks. The tenacity and patience Ganti Sir had shown us when he actually sat down and written code to help us check our results will always be inspirational. I will always remember the fun we had during our meetings.

I would also like to thank my friends who have always been there when i needed them, the Electrical Engineering department for its infrastructure and opportunity to interact with some of the best minds across India both in terms of Professors and students, IIT Madras for giving me the best four years of my life and last but not the least, my family for making me who I am today.

Aluka Abhinav Ram

Abstract

This project is the hardware implementation of Orthogonal Frequency Division Multiplexing (OFDM), a technique with immense scope of usage that is already present widely in a variety of applications. It is being implementd on an FPGA (Field Programmable Gate Array) and hence can be modified at will based on the application.

Contents

List of Tables

List of Figures

1 Introduction

1.1 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 wide-band digital communication, whether wireless or over copper wires, used in applications such as digital television and audio broadcasting, DSL broadband internet access, wireless networks, and 4G mobile communications.

OFDM is essentially identical to coded OFDM and discrete multi-tone modulation, and is a frequency-division multiplexing (FDM) scheme used as a digital multi-carrier modulation method. The word "coded" comes from the use of forward error correction. 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, narrow-band 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 narrow-band signals rather than one rapidly modulated wide-band signal. The low symbol rate makes the use of a guard interval between symbols affordable, making it possible to eliminate inter-symbol interference and utilize echoes and time-spreading to achieve a diversity gain, i.e. a signal-to-noise ratio improvement. This mechanism also facilitates the design of single frequency networks, 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.

1.1.1 Advantages:

- High spectral efficiency as compared to other double sideband modulation schemes, spread spectrum, etc.
- Can easily adapt to severe channel conditions without complex time-domain equalization.
- Robust against narrow-band co-channel interference.
- Robust against inter-symbol interference and fading caused by multipath propagation.

- Efficient implementation using Fast Fourier Transform (FFT).
- Low sensitivity to time synchronization errors.
- Tuned sub-channel receiver filters are not required (unlike conventional FDM).
- Facilitates single frequency networks; i.e., transmitter macro-diversity.

1.1.2 Disadvantages:

- Sensitive to Doppler shift.
- Sensitive to frequency synchronization problems.
- High peak-to-average-power ratio (PAPR), requiring linear transmitter circuitry, which suffers from poor power efficiency.
- Loss of efficiency caused by cyclic prefix/guard interval.

1.2 Orthogonality

Conceptually, OFDM is a specialized FDM, the additional constraint being: all the carrier signals are orthogonal to each other.

In OFDM, the sub-carrier frequencies are chosen so that the sub-carriers are orthogonal to each other, meaning that cross-talk between the sub-channels is eliminated and inter-carrier guard bands are not required. This greatly simplifies the design of both the transmitter and the receiver; unlike conventional FDM, a separate filter for each sub-channel is not required.

The orthogonality requires that the sub-carrier spacing is Hertz, where TU seconds is the useful symbol duration (the receiver side window size), and k is a positive integer, typically equal to 1. Therefore, with N sub-carriers, the total pass-band bandwidth will be $B \approx N \cdot \Delta f$.

The orthogonality also allows high spectral efficiency, with a total symbol rate near the Nyquist rate for the equivalent base-band signal (i.e. near half the Nyquist rate for the double-side band physical pass-band signal). Almost the whole available frequency band can be utilized. OFDM generally has a nearly white spectrum, giving it benign electromagnetic interference properties with respect to other co-channel users.

A simple example: A useful symbol duration TU = 1 ms would require a sub-carrier spacing of (or an integer multiple of that) for orthogonality. N = 1,000 sub-carriers would result in a total pass-band bandwidth of $N \cdot \Delta f$= 1 MHz. For this symbol time, the required bandwidth in theory according to Nyquist is N/2TU = 0.5 MHz (i.e., half of the achieved bandwidth required by our scheme). If a guard interval is applied, Nyquist bandwidth requirement would be even lower. The FFT would result in N = 1,000 samples per symbol. If no guard interval was applied, this would result in a base band complex valued signal with a sample rate of 1 MHz, which would require a base-band bandwidth of 0.5 MHz according to Nyquist. However, the pass-band RF signal is produced by multiplying the base-band signal with a carrier waveform (i.e., double-sideband quadrature amplitude-modulation) resulting in a pass-band bandwidth of 1 MHz. A single-side band or vestigial sideban modulation scheme would achieve almost half that bandwidth for the same symbol rate (i.e., twice as high spectral efficiency for the same symbol alphabet length). It is however more sensitive to multipath interference.

OFDM requires very accurate frequency synchronization between the receiver and the transmitter; with frequency deviation the sub-carriers will no longer be orthogonal, causing inter-carrier interference (i.e., cross-talk between the sub-carriers). Frequency offsets are typically caused by mismatched transmitter and receiver oscillators, or by Doppler shift due to movement. While Doppler shift alone may be compensated for by the receiver, the situation is worsened when combined with multipath, as reflections will appear at various frequency offsets, which is much harder to correct. This effect typically worsens as speed increases, and is an important factor limiting the use of OFDM in high-speed vehicles. Several techniques for ICI suppression are suggested, but they may increase the receiver complexity.

1.3 Implementation using the FFT algorithm

The orthogonality allows for efficient modulator and demodulator implementation using the FFT algorithm on the receiver side, and inverse FFT on the sender side. Although the principles and some of the benefits have been known since the 1960s, OFDM is popular for wide-band communications today by way of low-cost digital signal processing components that can efficiently calculate the FFT.

The time to compute the inverse-FFT or FFT transform has to take less than the time for each symbol. Which for example for DVB-T (FFT 8k) means the computation has to be done in 896 ţs or less. The computational demand approximately scales linearly with FFT size so a double size FFT needs double the amount of time and vice versa. As a comparison an Intel Pentium III CPU at 1.266 GHz is able to calculate a 8 192 point FFT in 576 ţs using FFTW. Intel Pentium M at 1.6 GHz does it in 387 ţs. Intel Core Duo at 3.0 GHz does it in 96.8 ţs.

1.3.1 Guard interval for elimination of inter-symbol interference:

One key principle of OFDM is that since low symbol rate modulation schemes (i.e., where the symbols are relatively long compared to the channel time characteristics) suffer less from inter-symbol interference caused by multipath propagation, it is advantageous to transmit a number of low-rate streams in parallel instead of a single high-rate stream. Since the duration of each symbol is long, it is feasible to insert a guard interval between the OFDM symbols, thus eliminating the inter-symbol interference.

The guard interval also eliminates the need for a pulse-shaping filter, and it reduces the sensitivity to time synchronization problems.

A simple example: If one sends a million symbols per second using conventional single-carrier modulation over a wireless channel, then the duration of each symbol would be one microsecond or less. This imposes severe constraints on synchronization and necessitates the removal of multipath interference. If the same million symbols per second are spread among one thousand sub-channels, the duration of each symbol can be longer by a factor of a thousand (i.e., one millisecond) for orthogonality with approximately the same bandwidth. Assume that a guard interval of 1/8 of the symbol length is inserted between each symbol. Inter-symbol interference can be avoided if the multipath time-spreading (the time between the reception of the first and the last echo) is shorter than the guard interval (i.e., 125 microseconds). This corresponds to a maximum difference of 37.5 kilometers between the lengths of the paths.

The cyclic prefix, which is transmitted during the guard interval, consists of the end of the OFDM symbol copied into the guard interval, and the guard interval is transmitted followed by the OFDM symbol. The reason that the guard interval consists of a copy of the end of the OFDM symbol is so that the receiver will integrate over an integer number of sinusoid cycles for each of the multipaths when it performs OFDM demodulation with the FFT. In some standards such as Ultrawideband, in the interest of transmitted power, cyclic prefix is skipped and nothing is sent during the guard interval. The receiver will then have to mimic the cyclic prefix functionality by copying the end part of the OFDM symbol and adding it to the beginning portion.

1.4 Simplified equalization

The effects of frequency-selective channel conditions, for example fading caused by multipath propagation, can be considered as constant (flat) over an OFDM sub-channel if the sub-channel is sufficiently narrow-banded (i.e., if the number of sub-channels is sufficiently large). This makes frequency domain equalization possible at the receiver, which is far simpler than the time-domain equalization used in conventional single-carrier modulation. In OFDM, the equalizer only has to multiply each detected sub-carrier (each Fourier coefficient) in each OFDM symbol by a constant complex number, or a rarely changed value.

Our example: The OFDM equalization in the above numerical example would require one complex valued multiplication per sub-carrier and symbol (i.e., complex multiplications per OFDM symbol; i.e., one million multiplications per second, at the receiver). The FFT algorithm requires [this is imprecise:

over half of these complex multiplications are trivial, i.e. = to 1 and are not implemented in software or HW]. complex-valued multiplications per OFDM symbol (i.e., 10 million multiplications per second), at both the receiver and transmitter side. This should be compared with the corresponding one million symbols/second single-carrier modulation case mentioned in the example, where the equalization of 125 microseconds time-spreading using a FIR filter would require, in a naive implementation, 125 multiplications per symbol (i.e., 125 million multiplications per second). FFT techniques can be used to reduce the number of multiplications for an FIR filter based time-domain equalizer to a number comparable with OFDM, at the cost of delay between reception and decoding which also becomes comparable with OFDM.

If differential modulation such as DPSK or DQPSK is applied to each sub-carrier, equalization can be completely omitted, since these non-coherent schemes are insensitive to slowly changing amplitude and phase distortion.

In a sense, improvements in FIR equalization using FFTs or partial FFTs leads mathematically closer to OFDM, but the OFDM technique is easier to understand and implement, and the sub-channels can be independently adapted in other ways than varying equalization coefficients, such as switching between different QAM constellation patterns and error-correction schemes to match individual sub-channel noise and interference characteristics.

Some of the sub-carriers in some of the OFDM symbols may carry pilot signals for measurement of the channel conditions (i.e., the equalizer gain and phase shift for each sub-carrier). Pilot signals and training symbols (preambles) may also be used for time synchronization (to avoid inter-symbol interference, ISI) and frequency synchronization (to avoid inter-carrier interference, caused by Doppler shift).

OFDM was initially used for wired and stationary wireless communications. However, with an increasing number of applications operating in highly mobile environments, the effect of dispersive fading caused by a combination of multi-path propagation and Doppler shift is more significant. Over the last decade, research has been done on how to equalize OFDM transmission over doubly selective channels.

1.5 Channel coding and interleaving

OFDM is invariably used in conjunction with channel coding (forward error correction), and almost always uses frequency and/or time interleaving.

Frequency (sub-carrier) interleaving increases resistance to frequency-selective channel conditions such as fading. For example, when a part of the channel bandwidth fades, frequency interleaving ensures that the bit errors that would result from those sub-carriers in the faded part of the bandwidth are spread out in the bit-stream rather than being concentrated. Similarly, time interleaving ensures that bits that are originally close together in the bit-stream are transmitted far apart in time, thus mitigating against severe fading as would happen when traveling at high speed.

However, time interleaving is of little benefit in slowly fading channels, such as for stationary reception, and frequency interleaving offers little to no benefit for narrow-band channels that suffer from flat-fading (where the whole channel bandwidth fades at the same time).

The reason why interleaving is used on OFDM is to attempt to spread the errors out in the bit-stream that is presented to the error correction decoder, because when such decoders are presented with a high concentration of errors the decoder is unable to correct all the bit errors, and a burst of uncorrected errors occurs. A similar design of audio data encoding makes compact disc (CD) playback robust.

A classical type of error correction coding used with OFDM-based systems is convolutional coding, often concatenated with Reed-Solomon coding. Usually, additional interleaving (on top of the time and frequency interleaving mentioned above) in between the two layers of coding is implemented. The choice for Reed-Solomon coding as the outer error correction code is based on the observation that the Viterbi decoder used for inner convolutional decoding produces short errors bursts when there is a high concentration of errors, and Reed-Solomon codes are inherently well-suited to correcting bursts of errors.

Newer systems, however, usually now adopt near-optimal types of error correction codes that use the turbo decoding principle, where the decoder iterates towards the desired solution. Examples of such error correction coding types include turbo codes and LDPC codes, which perform close to the Shannon limit for the Additive White Gaussian Noise (AWGN) channel. Some systems that have implemented these codes have concatenated them with either Reed-Solomon (for example on the MediaFLO system) or BCH codes (on the DVB-S2 system) to improve upon an error floor inherent to these codes at high signal-to-noise ratios.

1.6 OFDM system model

This section describes a simple idealized OFDM system model suitable for a time-invariant AWGN channel.

1.6.1 Transmitter:

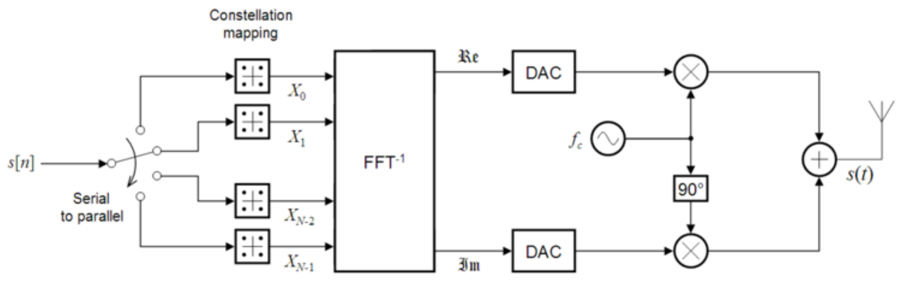


Figure 1: idealized OFDM transmitter

An OFDM carrier signal is the sum of a number of orthogonal sub-carriers, with base-band data on each sub-carrier being independently modulated commonly using some type of quadrature amplitude modulation (QAM) or phase-shift keying (PSK). This composite base-band signal is typically used to modulate a main RF carrier.

By inverse multiplexing a serial stream of binary bits, these are first demultiplexed into parallel streams, and each one mapped to a (possibly complex) symbol stream using some modulation constellation (QAM, PSK, etc.). Note that the constellations may be different, so some streams may carry a higher bit-rate than others.

An inverse FFT is computed on each set of symbols, giving a set of complex time-domain samples. These samples are then quadrature-mixed to pass-band in the standard way. The real and imaginary components are first converted to the analogue domain using digital-to-analogue converters (DACs); the analogue signals are then used to modulate cosine and sine waves at the carrier frequency, , respectively. These signals are then summed to give the transmission signal, .

1.6.2 Receiver:

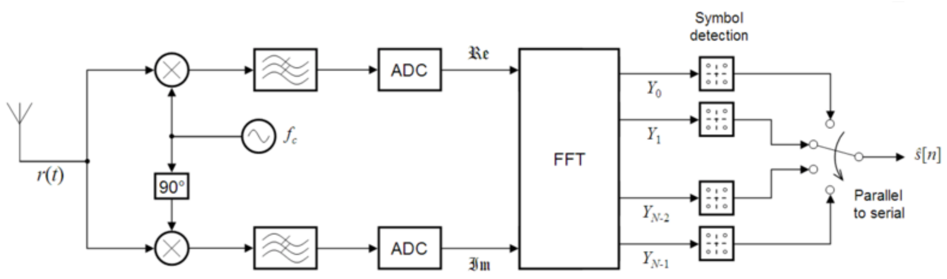


Figure 2: idealized OFDM Receiver

The receiver picks up the signal , which is then quadrature-mixed down to baseband using cosine and sine waves at the carrier frequency. This also creates signals centered on , so low-pass filters are used to reject these. The baseband signals are then sampled and digitized using analog-to-digital converters (ADCs), and a forward FFT is used to convert back to the frequency domain.

This returns parallel streams, each of which is converted to a binary stream using an appropriate symbol detector. These streams are then re-combined into a serial stream, , which is an estimate of the original binary stream at the transmitter.

2 OFDM for current application

2.1 Design Parameters

The design parameters for the current implementation of an OFDM Baseband Transmitter are given as follows.

Parameter	**Value (number of samples/bits)**
Transmitter input per frame	4608 (bits)
LDPC input per computation	1536 (bits)
LDPC output per computation	2304 (bits)
Constellation Mapping Scheme	QPSK
Fast Fourier Transform size	512 (samples)
Cyclic Prefix size	32 (samples)
one OFDM output symbol length	544 (samples)
Block size (number of OFDM symbols)	9
Preamble size	64 (samples)
Transmitted output	4960 (samples)

Table 1: Design parameters set for current Implementation

2.2 Symbol and Frame design

2.2.1 Symbol design

The input to the Inverse Fast Fourier Transform (IFFT) which is of length 512 real and imaginary samples is designed with Pilots and nulls placed in strategic locations. The nulls are located at the start and end of the 512 point input. They are also placed for a large part at the center of the 512 points to ensure that there is no DC component in the transmitted signal. The pilots are located at regular intervals in between the data in order to estimate the channel at the receiver end.

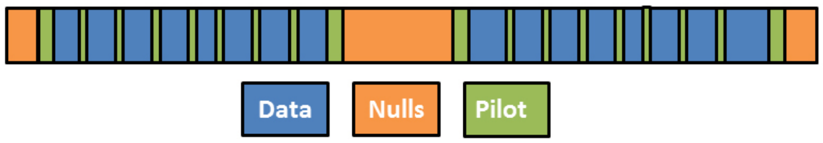


Figure 3: Input to IFFT

At the output a 32 sample Cyclic Prefix which is a copy of the last 32 points in the 512 point IFFT output is added. This acts as a guard interval to eliminate inter symbol interference. It is also used for the second stage frequency error estimation and correction through a cross correlation function. Hence each OFDM symbol consists a total of 544 samples.

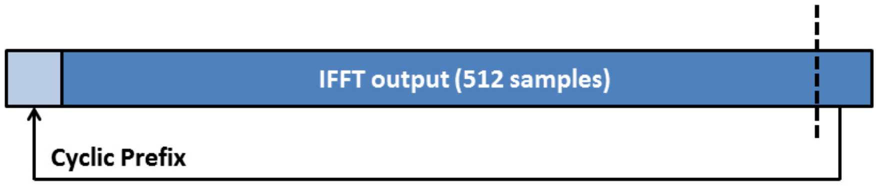


Figure 4: Output of IFFT and cyclic prefix addition

2.2.2 Frame design

Each OFDM frame consists of 9 OFDM symbols of 544 points each. A Preamble of size 64 samples are added at the beginning of this block in order to detect the arrival of the OFDM frame. This frame is detected at the starting point of the receiver using the Schmidl and Cox algorithm. The preamble is also used for a first stage frequency error estimation and correction operation. Hence each frame consists of 4960 samples of data, where each sample is represented by a complex number.

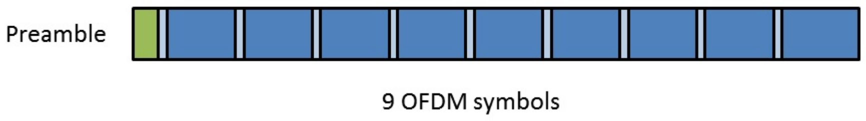


Figure 5: One OFDM frame

3 Hardware Implementation

3.1 Implementation Parameters

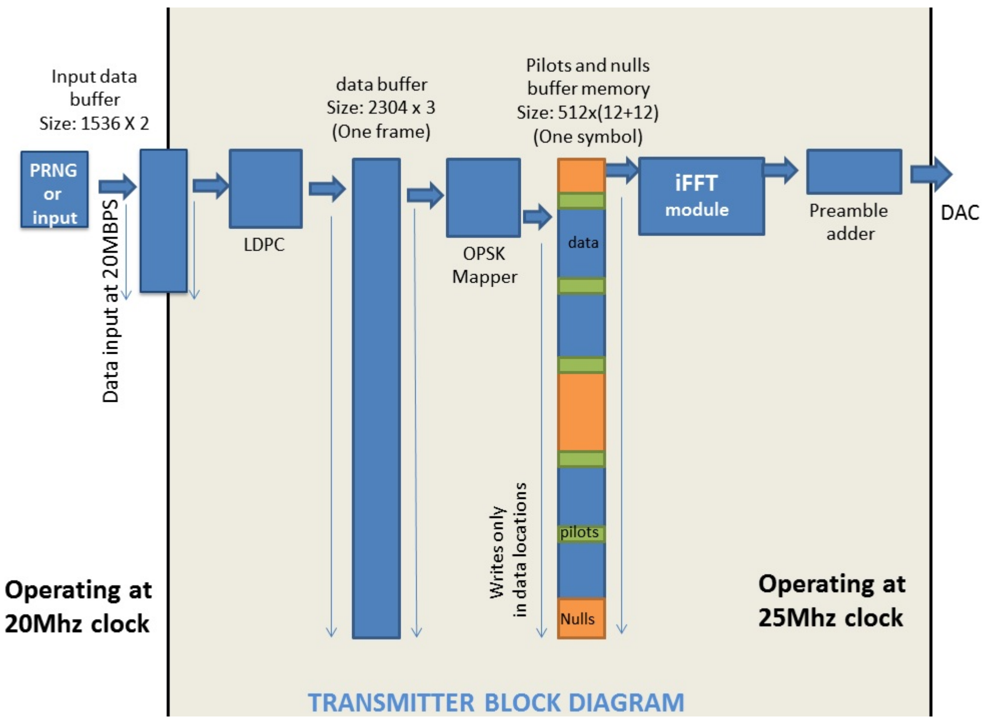
The current design for transmitter is implemented in Verilog for FPGA. Xilinx ISE Design Suite (version 14.3) was used for this and ISIM was used to test the outputs of the hardware implementation. The relevant parameters are given below:

Parameter	**Value**
Language used	Verilog
Target device	XC5VTX150T (Virtex – 5 FPGA)
Data input clock frequency	20 MHz (T = 50ns)
Internal and computation clock frequency	25 MHz (T = 40ns)
Data output rate	21.527 MHz
Data input	stream of bits
output bit width	16 bits (for each real and imaginary)
Data output	complex sample

Table 2: Implementation parameters

3.2 Transmitter Blocks

The blocks in the transmitter have been designed keeping in mind the changing parameter values and requirement for a run time configurable FFT point size. Each block is defined to perform based on certain control inputs received from outside and it gives out certain control signals for the working of other blocks.



Input data buffer Size: 1536 X 2
PRNG or input
Data input at 20MBPS
data buffer Size: 2304 x 3 (One frame)
LDPC
Pilots and nulls buffer memory Size: 512x(12+12) (One symbol)
OPSK Mapper
data
pilots
Nulls
Writes only in data locations
iFFT module
Preamble adder
DAC
Operating at 20Mhz clock
Operating at 25Mhz clock
TRANSMITTER BLOCK DIAGRAM

Figure 6: Transmitter showing various blocks

These blocks are then integrated in order to make a complete state machine that reverts back to its original state at the end of each operation. The blocks interact with each other using these control inputs and outputs and this ’hand-shaking’ design ensures that the transmitter works robustly under all circumstances and changed parameters.

The ’master_reset’ and ’clk’ - clock signals are common to all blocks. The clock signal is self explanatory but the master reset is used to initialize the memories to their respective values along all blocks along with its use to reset the entire transmitter to its initial state.

The pilot-nulls values and their locations, the preamble values, and the IFFT parameters and other implementation parameters like data widths, FFT point size, Cyclic Prefix size, Preamble size etc., are all stored in separate individual files and are included in the main code using the ’include(“filename.v”) function. This way, any change that is made to these parameter files will be reflect directly in the verilog code. These include files are created from the fixed point C code to ensure consistency between the fixed point simulations and actual implementation.

3.2.1 QPSK mapper

The QPSK mapper is straight forward. It takes in input as two bits at a time and gives out a corresponding complex mapped sample one clock cycle later.

Input	Output value
00	5792+i*5792
01	5792-i*5793
11	-5793-i*5793
10	-5793+i*5792

Table 3: QPSK mapping values

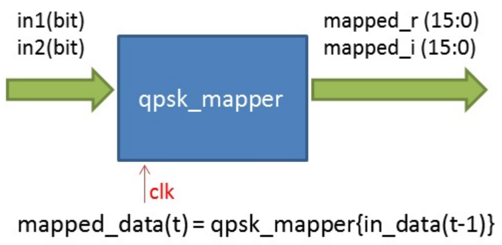


Figure 7: QPSK mapper

3.2.2 Pilots-Nulls adder

The pilot-null adder, as its name suggests organizes the data samples, pilots and nulls in the respective locations to the 512 point input of the IFFT block. It is a memory block of 512 complex samples of width 16 bits for each of real and imaginary components. The values for pilots and nulls are hard-coded into this memory block on giving an data high to the master reset input. This is done at the very beginning after starting the device.

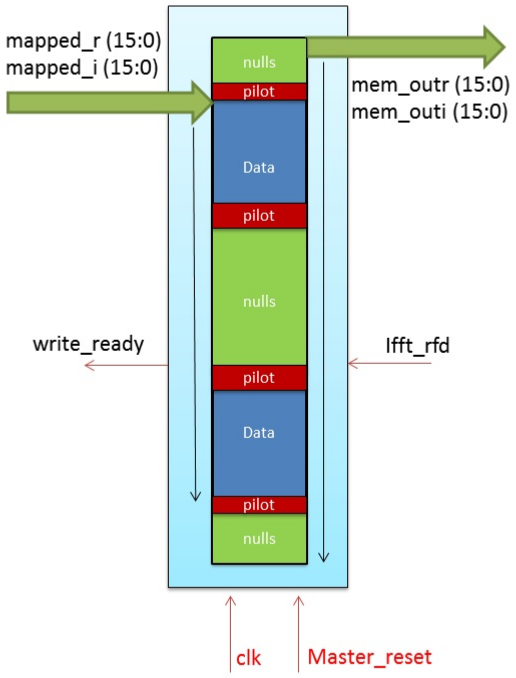


Figure 8: Pilot-Null adder

The IFFT module sends a 'ready for data' high signal as soon as it is ready to take inputs. When this happens two counters start within the pilot-null adder and a 'write ready' high signal is sent to the LDPC buffer saying that it is ready to accept data. One of them is a 'read counter' that counts serially from 0 to 511.

A 'write counter' starts after a suitable delay writes the data only into the data locations. The initial delay is to account for the latency involved in getting data from the LDPC buffer through the QPSK mapper. After starting, this counter increments serially when the data locations are continuous but jumps by a suitable increment when it encounters a pilot or null location. By this we ensure that the incoming data is being written only in the data locations without touching the pilot and null locations in between.

The ready for data signal from the IFFT module is high for 512 clock cycles and low for 32 clock cycles. The low signal is to account for the Cyclic Prefix that is added at the output of the 512 point IFFT. The write ready signal that goes to the LDPC buffer is high for 384 cycles and low for the remaining 160 cycles for a single OFDM symbol.

3.2.3 Preamble adder

The preamble adder sits at the output of the IFFT module. It consists of two memories, each of size equal to the preamble length and of bit widths equal to the complex output of the IFFT module. One of these memories is a right shift memory which as its name suggests, does a right shift operation at the positive edge of every clock cycle. The other is a fixed memory which stores the values of the preamble.

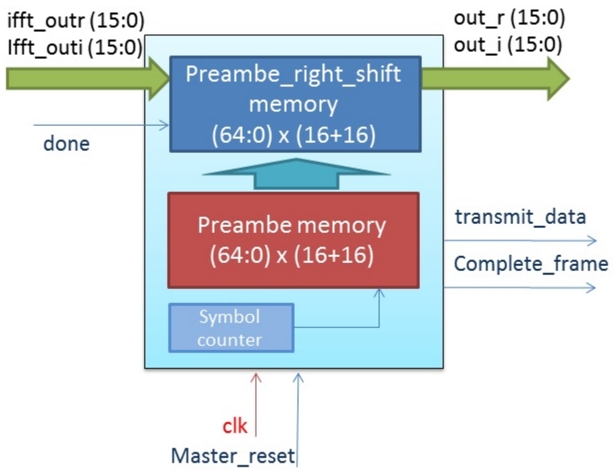


Figure 9: Preamble adder

At the very beginning, the master reset will load the preamble right shift memory with the respective values of the preamble.

The preamble adder block waits for the IFFT module to signal the arrival of a transformed output through a 'done' high signal. The right shift operation starts only after this and will continue until one frame has been transmitted. A counter within the block will count the number of symbols that have passed out for the current frame from the start of the right shift operation. After the required number of symbols have passed out (represented by the parameter - block size), the right shift operation stops and the preamble is loaded to the right shift memory in order to transmit the next frame. The last register in the right shift memory is connected to the output and the frame is transmitted through here.

The Preamble adder sends an output high 'transmit_data' signal when it starts the right shift operation. This indicates that it is transmitting the frame at the end of the right shift memory. At the end of the frame this signal goes low and toggles a 'frame done' signal to high which is toggled back to low after one clock cycle. This indicates the successful transmission of one frame of data.

3.2.4 IFFT module

This is at the heart of the transmitter block and it houses the Xilinx FFT core to perform the Inverse Fast Fourier Transform. This block is constructed in order to fix the scaling schedule and cyclic prefix length. These parameters can be changed in the main include parameters file that is included at the top of this module.

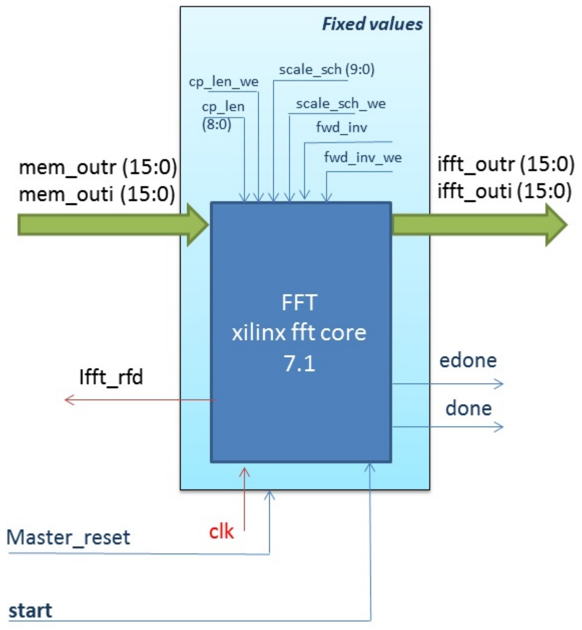


Figure 10: IFFT module

This module waits for a input 'start' high signal from the LDPC buffer. It sends a 'ifft_rfd - ready for data' high to the pilot null adder and starts taking in data. This signal is high for 512 cycles and low for 32 cycles to account for the inclusion of the Cyclic Prefix at the output. As soon as the computation is done it sends a 'done' high signal for one clock cycle.

3.2.5 Xilinx FFT core 7.1

The Xilinx FFT IP core version 7.1 was chosen among the lot of IP cores available online on OpenCores.org. It was chosen as it provides a variety of options for customization with respect to the architecture of implementation and performance versus resource optimization. It also adds the Cyclic Prefix automatically and has the capability to configure the FFT point size during run time.

The FFT and IFFT operation are implemented using the Radix - 2 or Radix

- 4 butterfly structure in stages. The number of these stages is equal to$\log_2(N)$, where N is the FFT point size. At each stage the radix structure involves a multiplication with a phase factor, truncation operation and an addition. A certain scaling factor is given at the end of each radix operation and this is set using the scaling schedule input.

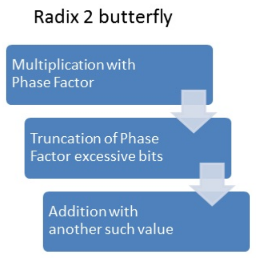

Radix 2 butterfly
Multiplication with Phase Factor
Truncation of Phase Factor excessive bits
Addition with another such value

Figure 11: radix 2

Parameter	**Choice**
Architecture option	Pipelined I/O Streaming
Data width	Fixed point: 16 bits wide
QPSK mapped values and pilot values	5792, -5793
Mode of scaling	Scaled

Table 4: IFFT parameters

The architecture option is chosen to be Pipelined as it has a latency of only one clock cycle after start is high. Other schemes have larger initial latencies. The data width was fixed at 16 bits after running simulations on the fixed point C code. The QPSK mapped values have been chosen to occupy 13 bits in order to carry forward as much precision as possible at the end of the truncation operation of each radix 2 stage. The scaling mode and scaling schedule were fixed after checking with the fixed point C code as well.

The following parameters were fixed after extensive testing and comparing values with the MATLAB output. The results of these tests will be discussed in the next sub section.

3.3 Testing of Implementation outputs

3.3.1 Comparision of FFT core with MATLAB

The inputs and outputs of the Verilog implementation of the FFT block have been loaded and tested against the floating point outputs of MATLAB. We observe that the difference between the MATLAB and Verilog outputs for the same input data differs by a maximum of two bits in the LSB. The fixed point C code verifies that this difference is acceptable for the transmission.

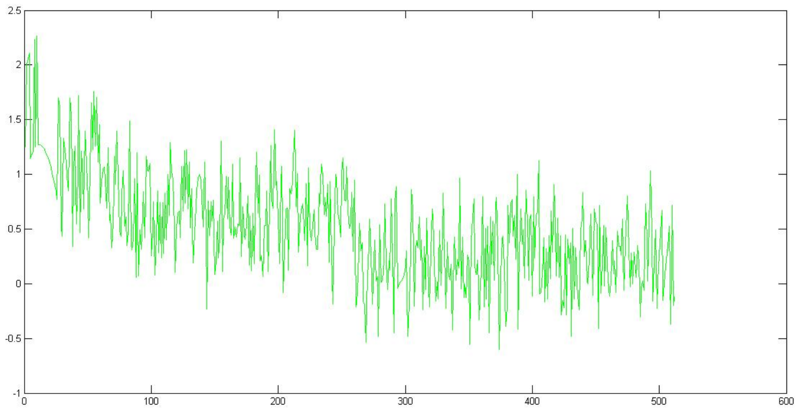


Figure 12: Difference of MATLAB and Verilog outputs for the same input for a 512 point IFFT

3.3.2 Verilog Test Benches

Verilog test benches were written for each block individually and for the integrated module to verify functional behavior, contiguity of data and timing. A result for the final transmitted data is shown below. The input for this has been taken from a pseudo random number generator (PRNG). This is a 22 bit right shift register that has gives the XOR of the last two bits as an input to the first bit, forming a large number of states and hence an output bit sequence with a very large period.

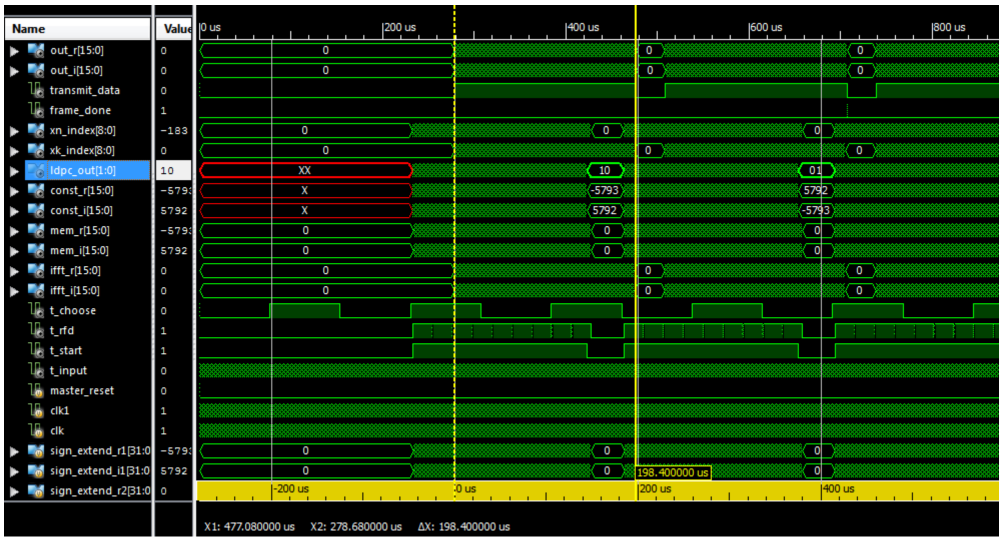


Figure 13: ISIM simulator results showing transmission of the first and second frame of data

3.4 Integration and timing

After designing each block individually, they were integrated with each other and other blocks such as the input buffer, LDPC encoder and LDPC buffer. The behavior of the input buffer, LDPC encoder and LDPC buffer will be discussed in detail in another report. This sub section will cover the description of the PRNG, QPSK mapper, pilot null adder, IFFT module and preamble adder.

3.4.1 Working of the current implementation

This sub section explains the working of each block, the relationship each block has with another and its behavior in response to inputs it gets from other blocks. The exact clocks have been used in the simulation shown above as well.

- The PRNG starts transmitting valid bits as soon as the master reset operation has been performed. This fills an input Buffer. Both the write operation in this buffer and the PRNG work at 20Mhz clock. It sends a signal to the LDPC encoder as soon as it is filled with 1536 bits of data.
- The imput buffer sends the data to the output at 25Mhz when requested by the LDPC encoder. Every block post this point will operate at 25Mhz. The LDPC encoder takes about 44 clock cycles to take the data input and finish computation.
- It then takes 24 clock cycles to transfer the 2304 bits to the LDPC buffer. Three such operations will fill up the LDPC buffer completely with 6912

bits that are required for one single frame of data.

- The LDPC buffer toggles an output flag to 'high' once it is filled with one frame of data i.e. 6912 bits. This flag remains high until the internal counter has counted from 1 to 6912 and then goes to 'low' indicating that all data for one frame has been sent to the next block. Having this block sees to it that we dont transmit anything until we have the entire frame, removing the requirement to toggle for valid data at the output.
- This output flag is used to start the FFT core. Responding to this the FFT core in the IFFT module sends a 'ready for data' high signal to the pilot-null adder saying that it is accepting data through its inputs to fill the internal buffer of 512 complex samples. This 'ready for data' signal stays high for 512 cycles and goes low for 32 cycles.
- The pilot null adder senses the 'ready for data' high signal from the IFFT module and sends a 'write ready' signal to the LDPC buffer to start its internal counter and start sending data to the QPSK mapper. This 'write ready' signal stays high for 384 cycles and low for 160 cycles. The internal read counter of the LDPC buffer will increment only when this 'write ready' signal is high. It will retain its previous value if it is low.
- This internal read counters is used as a means to know when the LDPC buffer is free to take the next 2304 bits from the LDPC encoder. As soon as it passes the 2304 and 4608 mark, it raises a flag and tells the LDPC encoder that the first and second one thirds of the buffer are free and can be filled with the next set of LDPC encoded bits.
- A latency of 3 clock cycles is observed from the time the 'write ready' signal is sent till the time we get valid data at the input of the pilot null adder block. To account for this, we start the writing operation after three clock cycles. This ensures that we dont lose any bits in between.
- The IFFT module then does the computation and starts sending data to the output. A done signal goes high for one clock cycle just as it is about to transmit the data. This signal is sent to the preamble adder block.
- The preamble adder block detects the 'done' high signal and starts the right shift operation, thereby sending the data out along with the preamble at its start.
- The output flag from the LDPC buffer automatically goes to zero after all 6912 bits have been sent to the pilot null adder. hence this ensures that the IFFT module doesnt take any stray bits at its input after transmitting 9 OFDM symbols.
- The preamble adder counts for 9 OFDM symbols through the 'done' signal and then resets the preamble to make it ready for the next frame of data.

3.4.2 Timing

- The input buffer takes about 76800ns i.e. 1536 clock cycles at 20Mhz clock, T = 50ns to fill with 1536 bits of data. This operation is repeated through out without any gap.

- The LDPC encoder takes 1760ns i.e. 44 clock cycles at 25Mhz clock, T = 40ns to take input and compute parity matrix. This operation happens only after 1536 bits of data fill up the input buffer.

- The LDPC buffer takes 960ns i.e. 24 clock cycles at 25Mhz clock, T = 40ns to fill one LDPC output i.e. 2304 bits. This operation happens only after the LDPC encoder finishes computing.

- The 'write ready' flag from the pilot null block to the LDPC buffer is high for 15360ns and low for 6400ns i.e. 384 cycles and 160 cycles at 25Mhz clock, T = 40ns.

- The 'ready for data' signal from the IFFT module to the pilot null adder stays high for 20480ns and low for 1280ns i.e. 512 cycles and 32 cycles at 25Mhz clock, T = 40ns.

- After an 'output flag' high is detected at the output of the LDPC buffer it takes 65280ns for the internal couter to read through the first 2304 bits i.e. 1632 cycles at 25Mhz clock, T = 40ns.

- At the output the transmitter takes 198400ns to transmit 4960 complex samples i.e. 4960 cycles at 25Mhz clock, T = 40ns.

These individual processes happen serially in some cases and in parallel in some other. We notice that the filling of input buffer takes more time than most other processes and hence it would be the rate determining step. We observe that the transmission of one frame of data is done before the LDPC buffer is filled with data for the next frame. The following diagrams clearly show the timing of various operations in the transmitter.

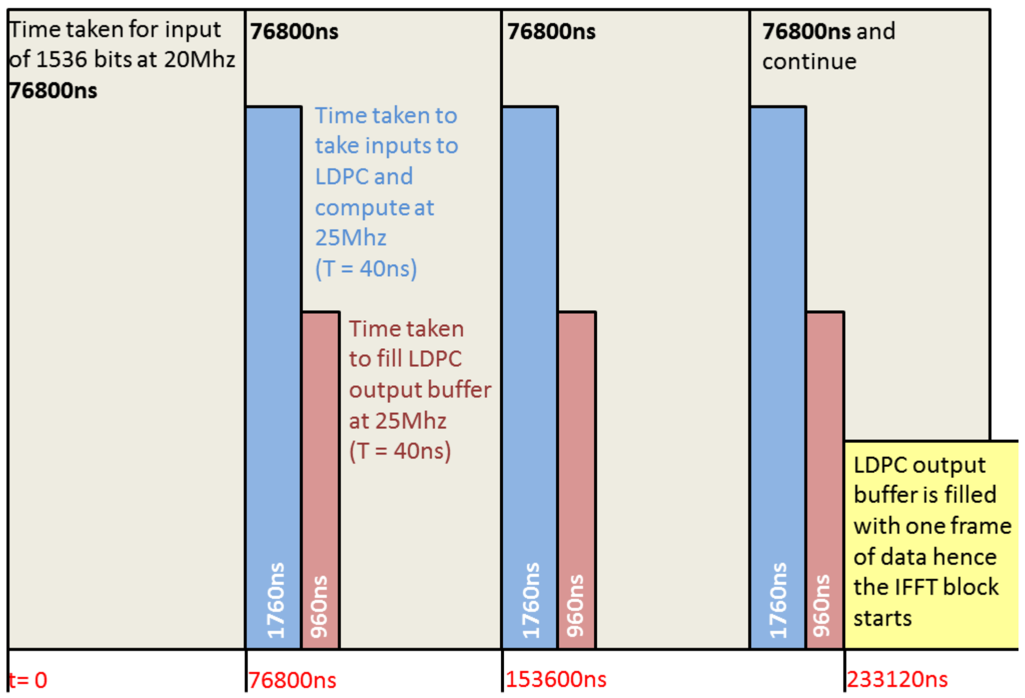

Time taken for input of 1536 bits at 20Mhz
76800ns
76800ns
76800ns
76800ns and continue
Time taken to take inputs to LDPC and compute at 25Mhz (T = 40ns)
Time taken to fill LDPC output buffer at 25Mhz (T = 40ns)
LDPC output buffer is filled with one frame of data hence the IFFT block starts
1760ns
960ns
1760ns
960ns
1760ns
960ns
t= 0
76800ns
153600ns
233120ns

Figure 14: Timing at the beginning of before transmitting the first frame

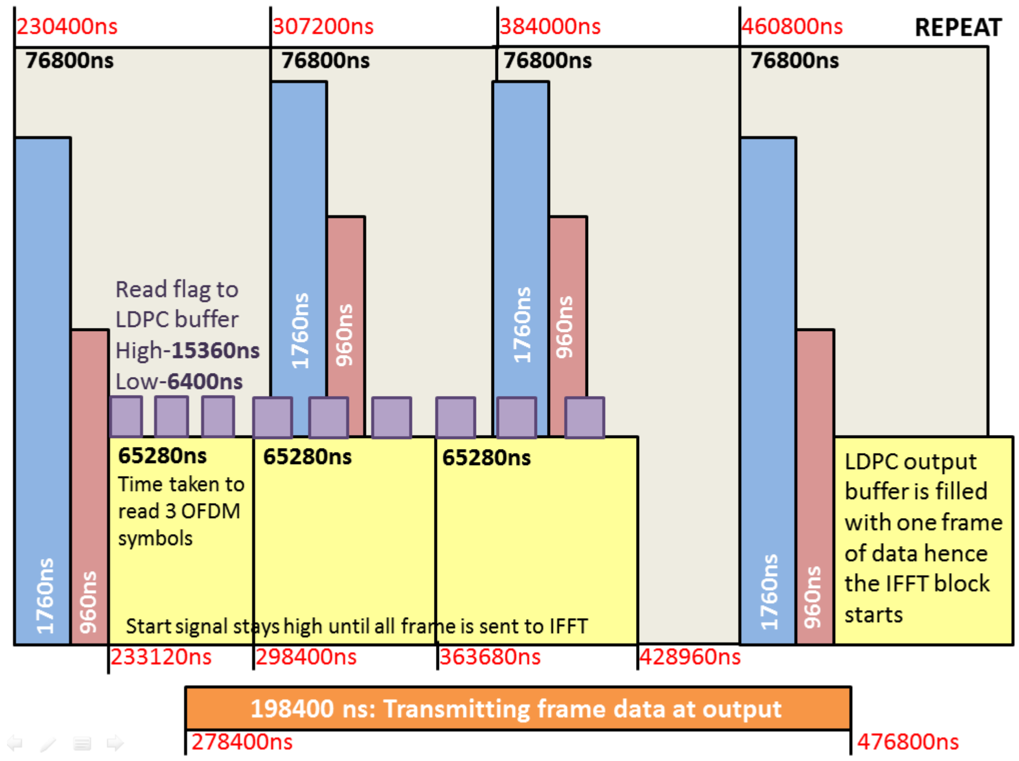

230400ns
307200ns
384000ns
460800ns
REPEAT
76800ns
76800ns
76800ns
76800ns
1760ns
960ns
Read flag to LDPC buffer High-15360ns Low-6400ns
1760ns
960ns
1760ns
960ns
65280ns Time taken to read 3 OFDM symbols
65280ns
65280ns
1760ns
960ns
LDPC output buffer is filled with one frame of data hence the IFFT block starts
Start signal stays high until all frame is sent to IFFT
233120ns
298400ns
363680ns
428960ns
198400 ns: Transmitting frame data at output
278400ns
476800ns

Figure 15: Timing for transmission of any frame