Volume 3, Issue 10, October 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Different Approaches for OFDM Transmitter and Receiver Design in Hardware FPGA Design and Implementation with Performance Comparison

Danijela Efnusheva*, Goce Dokoski, Aristotel Tentov, Marija Kalendar
Computer Science and Engineering Department
Faculty of Electrical Engineering and Information Technology
Skopje, Rep. of Macedonia

Abstract—In this paper we investigate performances (bandwidth and speed) that could be obtained using basic logic structures for implementing improved Cooley-Tukey algorithm for IFFT/FFT in transmitter/receiver, as well the approach for implementing OFDM processing unit based on optimized ROM implementation. We implemented these techniques for OFDM in Virtex5 and Viterx7 FPGA boards, and we analyzed the used percentage of FPGA boards for two different approaches, improved Cooley-Tukey algorithm and minimized ROM implementation, versus maximum achievable performances under different modulation parameters

Keywords—FFT/IFFT, OFDM, Cooley-Tukey, ROM, FPGA, multi-gigabit speed.

I. INTRODUCTION

Orthogonal Frequency Division Multiplexing (OFDM) is the dominant transmission technique used in the 802.11 set of WLAN standards. It divides the available spectrum into many carriers, each one being modulated by a low rate data stream, [1]. OFDM is similar to FDMA in the way it provides multiple users access to multiple channels, spaced apart at precise frequencies. However, it utilizes the available bandwidth more efficiently, because of its adjacent subcarriers orthogonality. Actually, in OFDM system, the sampling for a single subcarrier is happening exactly when the other subcarrier signals have zero value amplitude, [2]. The advantage of such concept is that it mitigates the effects of multipath, and provides higher data rates, while avoiding Inter Symbol Interference (ISI), [3].

OFDM was initially developed in the 1960s, but it has become a recognized high-speed data transmission technique in the recent years. This is due to the latest technology development and progress, which caused cost decrease of the digital signal processing blocks that are the most essential part of each OFDM system, [4], [5]. In fact, the problem with a very large array of sinusoidal generators and coherent demodulators for generating and receiving signals, was overcame, since digital signal processors (DSP) for generating and demodulating the OFDM signal, were designed, [6], [7].

OFDM processing, which is carried out in digital domain, can be implemented as application specific integrated circuit (ASIC) or general purpose processor (GPP). GPP are at the limits of providing necessary computing performances requested by wire rates. On the other hand, ASIC circuits can reach high speed processing, but once they have been designed it is impossible to change them. This implied some novel technologies development, such as Field programmable gate array architecture (FPGA). This platform, which is a compromise between price, speed and programmability, is very suitable for hardware implementation of OFDM system, basically because it can be easily reprogrammed with novel and improved chip designs, [8], [9], [10]. Therefore, in this paper we present the hardware implementation of improved Cooley-Tukey and minimized ROM based OFDM models in Virtex5 and Viterx7 FPGA boards. The aim of this paper is to investigate the bandwidth and speed performances of the proposed models, under different modulation parameters.

In rest of this paper, we discuss state of the art OFDM system model in Section II, comment OFDM system model with transmitter and receiver model in Section III, present OFDM improved Cooley-Tukey FPGA based implementation in Section IV, introduce ROM-based and minimized ROM-based OFDM implementation in Section V. In Section VI we present results obtained during verification, testing and synthesizing proposed OFDM implementations on Virtex5 and Virtex7 FPGA boards, such as performance analysis results. Finally, Section VII concludes the paper.

II. STATE OF THE ART

Given the popularity and high applicability of the OFDM data encoding method, OFDM is the technology of choice for many fast wireless transfer technologies. Consequently, finding a suitable model for practical OFDM implementation is one of the real challenges among the current public research. Thus, many versatile implementations exist, each with different results, striving to achieve maximal data throughput. Research is based mainly on FPGA platforms ([8]-[10], [13]-[16], [18]), due to their high availability and low price, as well. The practical implementation of the OFDM encoding method mainly depends on the realization of the Fast Fourier Transformation (FFT/IFFT). Many methods exist

for completing this task, but most implementations use the Cooley-Tukey algorithm which enables decreasing the complexity of calculations to O (Nlog2N). Nevertheless, the algorithm implements quite many operations involving complex numbers dependent on N (the number of points used in the transformation). Consequently, numerous research papers ([9]-[12]) explore FFT/IFFT hardware implementations, aiming to reduce the number of multiplication operations and/or pipelining the operation execution, in order to produce hardware with less complexity. The authors in [9] present an FFT/IFFT processor implementation optimized for reducing the FPGA slice size (4, 5, 6%), and capable of varying the used points in the transformation from 256, 512 to 1024. Likewise, a processor architecture designed in [11] works with 8 transformation points, supplementing the process by pipelining operations, and completely eliminating the complex numbers multiplications expected from the Cooley-Tukey algorithm. Very similar optimization process has been presented in [12] with the use of a specially designed multiplication module.

In general, any single Discrete Fourier Transform (DFT) module will be limited by the clock speed of the FPGA board. For example, [13] proposes an FPGA implementation of the radix-2 decimation-in-time (DIT) Fast Fourier Transform (FFT) that operates on a 227MHz clock. This chip implemented on a Xilinx Virtex 2 Pro board. Regardless of the relatively slow clock that drives the FPGA platforms, some very impressive data throughputs have been achieved. For example, [14] describes a Cooley-Tukey based implementation of an IFFT for an Optical OFDM Transmitter with 12.1 Gbit/s, using Virtex 5 FX200T FPGA. The same transmitter is successfully applied in a 109 Gb/s (9×12.1 Gb/s) transmission system, [15]. Furthermore, [16] introduces a 101.5 Gb/s OFDM transmitter for optical communication through fiber with 16QAM modulated sub carriers, by using two Xilinx Virtex 5 FPGA boards that work in parallel.

In order to achieve higher data rates using an FPGA, parallelization is necessary to compensate for the low clock speeds. In [17] a high performance baseband transceiver is presented that utilizes a 4x4 MIMO-OFDM system in order to achieve a 1Gb/s speed. When it comes to WLANs, transmission speeds of up to 1 Gb/s that use the 60GHz radio band are starting to emerge [18].

As the extensive research of the field revealed that the Cooley-Tukey algorithm is the most used and most suitable algorithm for FFT/IFFT hardware implementation, in this paper we propose an approach for implementing FFT/IFFT computations with improved Cooley-Tukey algorithm, and other completely different approach which includes ROM and minimized ROM based solutions. Furthermore, this paper is not concerned only with merely implementing two different FFT/IFFT algorithms in hardware, but is also concerned with achieving greater data throughput rates.

III. OFDM SYSTEM MODEL

Generally, an OFDM system consists of transmitter, receiver and communication channel, which carries out the orthogonal OFDM symbols. The OFDM signal is generated with the Inverse Fast Fourier Transform (IFFT) function at the transmitter, while at the receiver end, the opposite operation is performed, known as the Fast Fourier Transform (FFT), [5], [10]. The iterative nature of the FFT/IFFT functions and their computational complexity makes OFDM system to be ideally realized as dedicated logic, with specialized hardware blocks. Therefore, in this paper we present two OFDM system models which implement different methods for FFT/IFFT computation. The first model is based on Cooley–Tukey algorithm, [6], [10], while the other has an optimized ROM implementation. We implement both models using Xilinx's Virtex 5 and Virtex 7 FPGA boards. It should be noted that in our approach, we consider only the OFDM transmitter and receiver modules, without modelling the communication channel, and its coding, interleaving and equalization techniques. Actually, we implement OFDM signal generation with OFDM transmitter and retrieving of transmitted symbols by OFDM receiver.

A. OFDM Transmitter

In order to generate OFDM signal, OFDM transmitter is consisted of modules purposed for QAM mapping, serial to parallel conversion, IFFT computation, cyclic prefix insertion and parallel to serial conversion. The block diagram of OFDM transmitter, showing how OFDM signal is processed on sending side, is presented in figure 1.

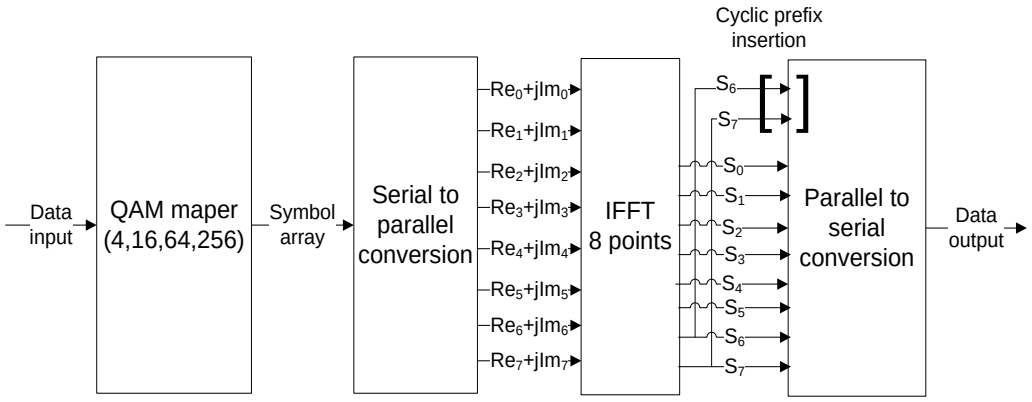


Fig. 1 Block diagram of OFDM transmitter

The first block in the OFDM transmitter is responsible for modulating the input data sequence, using QAM with several various signal constellations (4, 16, 64, and 256). According to a given constellation, complex value symbols (I+jQ) are obtained. Additionally, the information of the constellation I and Q components is presented as a real value (float or fix

point) number. In general, the preferred modulation technique depends on the communication channel quality, thus allowing larger constellation for interference free channels.

The modulated data symbols are then parallelized with the serial to parallel conversion module in N different sub streams, where N depends on the number of points, used in the IFFT block. In our approach, we consider OFDM system with 8 subcarriers. The input data symbols are distributed to the IFFT block input, one by one, and then they are transformed from frequency to time domain symbols. Actually, each sub stream modulates a separate subcarrier through the IFFT modulation block. This block executes the computations given in equation (1), where $X(k)$ are input symbols and $x(n)$ are outputs of the IFFT block.

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) W_N^{-nk},\ n = 0,1,\dots,N-1,\ W_N^{-nk} = e^{\frac{j2\pi nk}{N}} \quad (1)$$

The cyclic prefix is an extension of the IFFT-modulated symbol, obtained by copying the last L samples from the IFFT block output, in front of it. Generally the cyclic prefix length for 802.11 set of standards is N/4th portion of the OFDM symbol frame, which means that in an OFDM system with 8 subcarriers, the cyclic prefix length L is two. The cyclic prefix is required in order to eliminate the inter-symbol and inter-block interference (IBI). After the cyclic prefix insertion, the parallel OFDM symbols are back converted to serial stream, with the parallel to serial conversion module. This allows an OFDM symbol frame to be generated. This frame will be modulated at carrier frequency before its transmission through the channel.

B. OFDM Receiver

In order to reconstruct the generated OFDM signal, OFDM receiver has to comprise modules purposed for serial to parallel conversion, cyclic prefix removal, FFT computation, parallel to serial conversion and QAM decoding. The block diagram of OFDM receiver, illustrating OFDM signal processing on the receiver side, is presented in Figure 2.

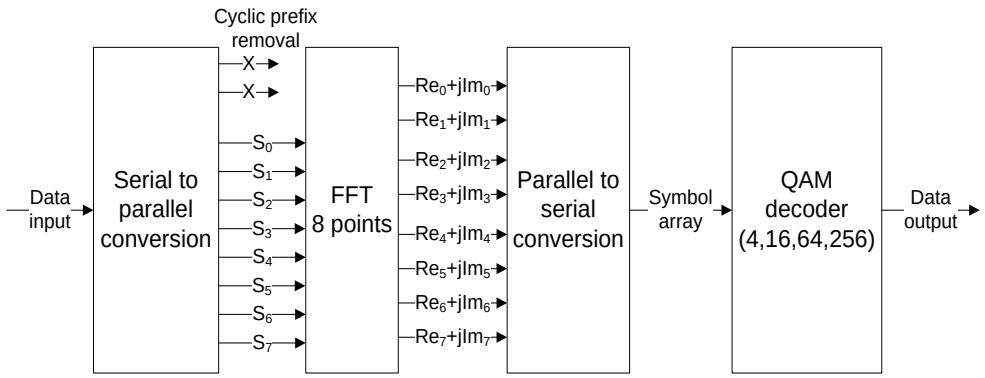


Fig. 2 Block diagram of OFDM receiver

The receiver performs inverse operations, in contrast to the OFDM transmitter. The input data of the OFDM receiver first goes through the serial to parallel conversion and cyclic prefix removal modules. Afterwards, the signal is passed to the 8-point FFT module, which converts it to frequency domain, and therefore retrieves the exact form of the transmitted symbols. This module executes the computations given in equation (2), where $x(n)$ are input symbols and $X(k)$ are outputs of the FFT block.

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk},\ k = 0,1,\dots,N-1,\ W_N^{nk} = e^{\frac{-j2\pi nk}{N}} \quad (2)$$

The FFT block output symbols are then serially converted, with a parallel to serial conversion module. Each of the symbols is then sent to the QAM decoder that enables symbols demapping, according to the appropriate constellation diagram (4, 16, 64, and 256). The QAM decoder defines a threshold in decision making, while decoding the QAM symbols, in order to deal with the modifications caused by the channel noise, timing de-synchronization or some other unwanted occurrences.

IV. OFDM COOLEY-TUKEY BASED IMPLEMENTATION

The discussed model of an OFDM system, presented in the previous section, has been implemented using aCooley–Tukey based method for IFFT/FFT computation. This method is the most general of all IFFT/FFT algorithms and is based on the "divide and conquer" approach. Its applicability is due to its recursive nature and ability to reduce the overall algorithm runtime from $O(N^2)$ to O(NlogN). However, the Cooley–Tukey algorithm is still not simple, as it requires N/2*logN multiplications and N*logN additions involving complex numbers, whose real and imaginary components are float- or fix-point real values.

N-point IFFT/FFT modules which implement radix 2 Cooley–Tukey algorithm perform parallel computation of the even-indexed and the odd-indexed outputs, by using two interleaved N/2-point IFFT/FFTs. This approach allows recursive algorithm execution in $\log_2 N$ phases. In each phase the input data sequence is divided into two N/2 point data sequences, allowing parallel "butterfly" processes to be performed, computing sums or differences for the symbols of the first and the second half of the input signal, respectively. The odd-numbered samples of the IFFT/FFT require the pre-multiplication of the input sequence with a so-called twiddle/conjugated twiddle (W_N^{nk}/W_N^{-nk}) factors, accordingly. For illustrative purposes, an 8-point radix 2 Cooley–Tukey FFT computation is shown in figure 3.

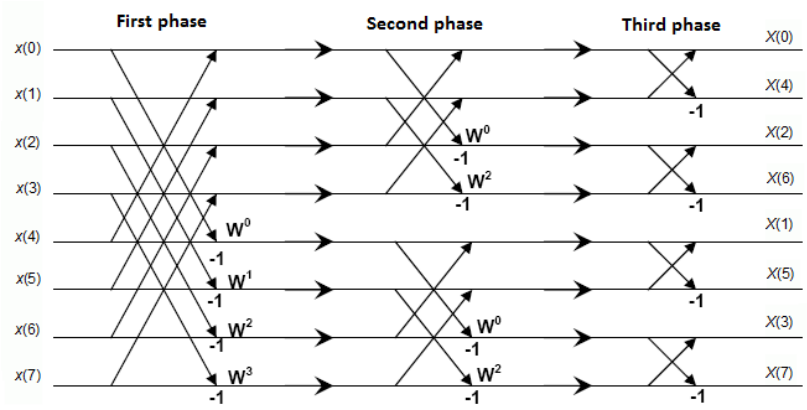


Fig. 3 8-point radix 2 Cooley–Tukey FFT algorithm

The IFFT/FFT modules have the main role in each OFDM system. In the proposed OFDM system, 8 point IFFT and FFT modules have been realized. This choice has been made since 8 point modules complete IFFT/FFT calculations in only three phases and thus require minimal logic, compared to the IFFT/FFT modules with more points (128, 256…). We believe that designing an optimal version of 8-point IFFT/FFT module would be a good starting point for designing a 16- or 64-point IFFT/FFT module, in the future. For the 8-point IFFT and FFT modules of the OFDM system model, shown in Figures 1 and 2, we propose a simplified hardware implementation of the Cooley–Tukey algorithm, with reduced number of multiplications by Twiddle factors. Assuming that Twiddle(0) = 1, Twiddle(-0) = 1, Twiddle(-2) =j and Twiddle(2) = -j, the multiplications by one are not taken into account and the multiplications with j and -j are performed in such a way that the products (a+jb)*(j) and (a+jb)*(-j) are directly computed as -b +ja and b-ja in hardware. This way, the IFFT/FFT modules execute only two multiply operations with the factors: Twiddle(1) and Twiddle(3) or Twiddle(-1) and Twiddle(-3), accordingly, in the first phase of the Cooley–Tukey algorithm. As a result of these improvements, the number of adders in the IFFT/FFT modules realized in Virtex 5 (Virtex 7) FPGA, is decreased by 29.63% (9.75%) / 38.77% (19.44%), accordingly.

The proposed IFFT/FFT modules perform operations over real and imaginary components of symbols that represent such 16-bit real numbers with fixed-point, [19]. Comparing to the 16-bit IEEE half-precision floating-point representation, the fixed-point approach is more convenient for use in the proposed IFFT/FFT modules. This is confirmed by several experiments, which show that the IFFT/FFT Virtex 5 (Virtex 7) FPGA implementations that use floating-point numbers, increase the complexity 15,2(13)/17(11) times, accordingly. Actually, these modules utilize 76(13)%/68(11)% of the slice resources of Virtex 5 (Virtex 7) FPGA components, while the IFFT/FFT modules which work with fixed-point numbers utilize only 5(1)%/4(1)% of the slice resources of Virtex 5 (Virtex 7) FPGA components.

In order to achieve high throughput and continuous data flow, the IFFT/FFT modules implement pipeline architecture, divided into control and data path. The control path is used to generate signals that control the operation of the functional units and the register transfers in the data path. On the other hand, the data path is purposed to enable pipeline execution of the IFFT/FFT computations in four stages (Figure 4): operands load (and multiplication by 1/N for IFFT), executing the operations in the first phase of the Cooley–Tukey algorithm, executing the operations in the second and third phase of the same algorithm (radix-4) and enabling the calculated results at the output. During the pipeline execution, the results of each stage are placed in intermediate registers, thus allowing continuous data flow between the sequential pipeline stages. The final result of the IFFT/FFT computations is obtained after four clock cycles.

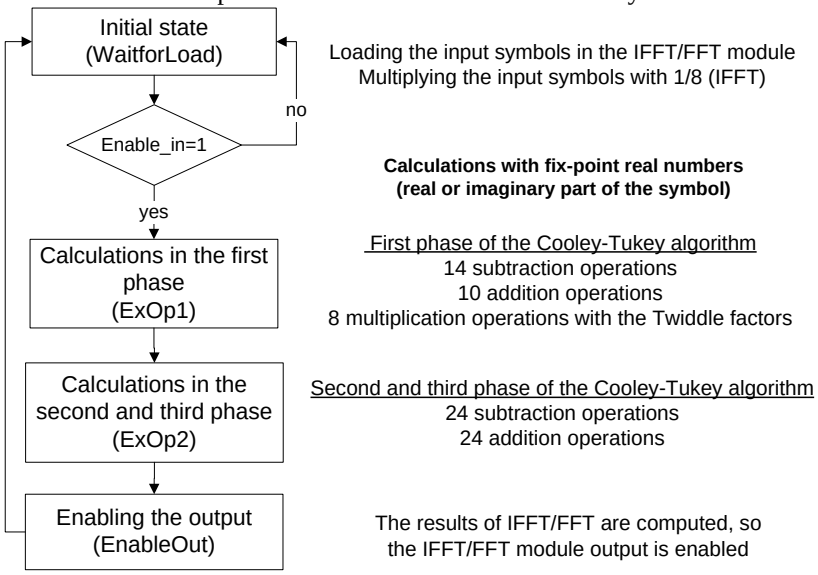


Fig. 4 State diagram of the Cooley–Tukey based IFFT/FFT module

Intermediate registers store and transfer the intermediate results (imaginary and real parts of a symbol) from one to another stage in the pipeline. The IFFT/FFT modules implement three sets of 32 16-bit registers. The input symbols are loaded in the first set of registers: i0re-i15re and i0im-i15im, while the results of the next two pipeline stages are placed in: x0re - x15re and x0im-x15im, y0re - y15re and y0im-y15im set of registers, accordingly. Without using these intermediate registers, pipelined execution will not be enabled, but the number of registers would have decreased three times. However, the complexity of the IFFT/FFT module, in terms of occupied Virtex5 or Virtex7 FPGA slice resources, will change to a maximum of 1%. Thus we can justify the use of intermediate registers.

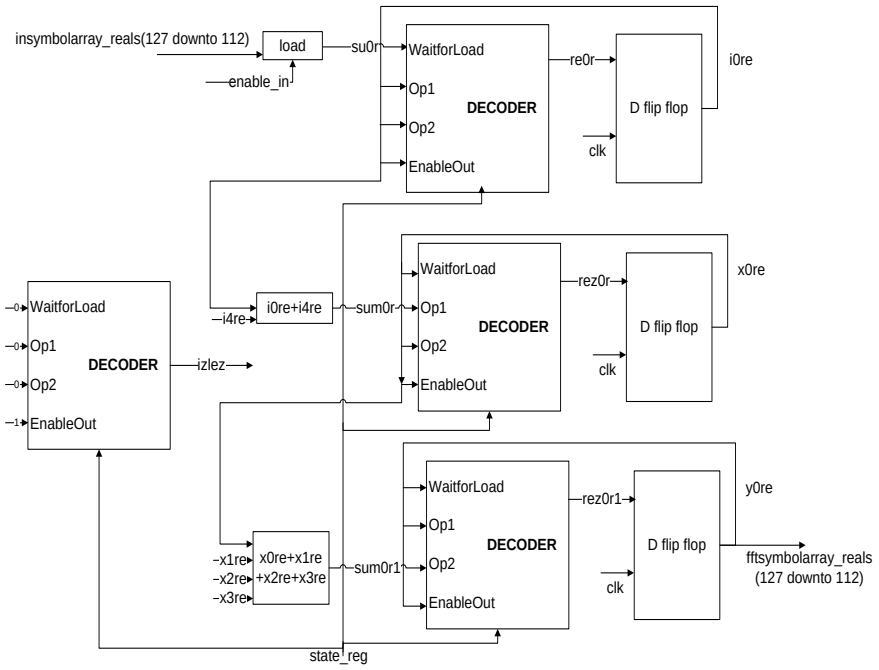


Fig. 5 Part of the datapath in the Cooley–Tukey based IFFT module

A complete OFDM system comprises several components in addition to the IFFT/FFT block. Therefore, for the proposed OFDM system model, we realized several modules for QAM mapping/decoding (4, 16, 64, 256), as well as, blocks for serial to parallel (1/8, 1/10, 2/8, 2/10) and parallel to serial conversion (8/1, 10/1, 8/2, 10/2), extended with the ability to insert and remove cyclic prefix (length 2), accordingly. The QAM 4/16/64/256 mapping modules are realized with a decoder function, which outputs real and imaginary components for each 2/4/6/8 input bit stream. On the other hand, the QAM 4/16/64/256 decoding modules include several branching functions, which define the specific ranges for generating the output bits of the decoded symbol. The SP1/8 (SP2/8) and PS8/1 (SP2/8) conversion modules are realized as shift registers which load and output one/two symbols each clock cycle. The conversion time of a SP1/8 and PS8/1 module is eight cycles, which is additionally extended by two cycles when the modules implement cyclic prefix insertion and cyclic prefix removal, accordingly. Assuming that each symbol arrives in a cycle, the SP1/8 module implements two additional buffers which are intended to hold the input symbols, while the module performs cyclic prefix insertion. Another approach to dealing with cyclic prefix delay is to use SP1/10 module which assumes that eight input symbols arrive in ten cycles. This way the cyclic prefix insertion time doesn't cause additional delay, regarding to the input symbol frequency. The PS8/2 (SP2/8) and PS10/2(SP2/10) modules allow parallel processing of two symbols at a given moment, and thus decrease the conversion time for eight symbols by half. These modules are used only if the OFDM system utilizes two QAM mapping/decoding modules that map/decode two symbols simultaneously. As a result, the conversion is performed in 4 and 5 cycles, instead of 8 and 10 cycles, accordingly. Given that IFFT/FFT computations are executed in 4 cycles as well, we allow pipeline data flow from one to another module, in such a way that each module is fully utilized all the time. This way we double the throughput for the overall OFDM system. Besides that, the proposed OFDM modules will not utilize much hardware resources, so we can further increase the data rates, by extending the number of points (16 or 64) in the proposed IFFT/FFT modules, and/or by using multiple transmitter/receiver modules that will work in parallel.

The modules of the OFDM system are implemented in VHDL, [20], [21]. The VHDL models of the OFDM system modules are developed using ISE Design Suite Tool from Xilinx, and their functionality is simulated with Xilinx ISim simulator. For the verification of the IFFT/FFT modules, the same input test sequence is applied to the VHDL test bench and the fft/ifft embedded MATLAB functions, and then the outputs are compared. After the verification process, the modules are integrated in the OFDM transmitter and receiver VHDL blocks, and then synthesized on a Virtex 5 XC5VLX110T and Virtex 7 XC7VX330T FPGA boards, separately. The purpose is to compare the resource usage and performance characteristics of the proposed OFDM transmitter and receiver components on Virtex platforms from different families (5 and 7), and with diverse capabilities.

For this research, we have implemented six OFDM transmitter and four OFDM receiver models, depending on the input symbol frequency and cyclic prefix presence or absence. Each of these models additionally supports four different types of QAM mapping: QAM4, QAM16, QAM64 and QAM256.

Realized models of OFDM transmitters are the following:

- model 1: doesn't insert cyclic prefix in an OFDM frame where symbols arrive with frequency of one symbol in one cycle;
- model 2: inserts cyclic prefix in an OFDM frame where symbols arrive with frequency of one symbol in one cycle;
- model 3: inserts cyclic prefix in an OFDM frame where symbols arrive with frequency of eight symbols in ten cycles;
- model 4: doesn't insert cyclic prefix in an OFDM frame where symbols arrive with frequency of two symbols in one cycle;
- model 5: inserts cyclic prefix in an OFDM frame where symbols arrive with frequency of two symbols in one cycle;
- model 6: inserts cyclic prefix in an OFDM frame where symbols arrive with frequency of eight symbols in five cycles;

Realized models of OFDM receivers are presented below:

- model 1: receives an OFDM frame where symbols arrive with frequency of one symbol in one cycle, without cyclic prefix;
- model 2: receives an OFDM frame where symbols arrive with frequency of one symbol in one cycle, with cyclic prefix;
- model 3: receives an OFDM frame where symbols arrive with frequency of two symbols in one cycle, without cyclic prefix;
- model 4: receives an OFDM frame where symbols arrive with frequency of two symbols in one cycle, with cyclic prefix;

The intention with developing various transmitter and receiver models is to compare their performances regarding to the modulation technique, cyclic prefix presence/absence, and input symbol frequency. It is expected that, best results will be achieved in the case when more symbols, carrying more information, are processed simultaneously each processing cycle, in a pipeline manner.

V. ROM-BASED AND MINIMIZED ROM-BASED OFDM PROCESSING UNITS

In this paper an alternative approach to the OFDM implementation is also considered, which is directly derived from the Discrete Fourier Transform (DFT), eq. (3) and the Inverse DFT, eq. (4).

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N} \tag{3}$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i2\pi kn/N} \tag{4}$$

When it comes to OFDM processing, the Fourier transformations are only needed over a relatively small set of values – the QAM choice of symbols as well as the exponential coefficients in the DFT/IDFT sum. This is the case for both the transmitting and the receiving side. The sender maps its incoming data stream into the QAM symbol values and feeds them into the IDFT module, whereas the receiver, rounds up the received values to the nearest symbol of the QAM constellation, and again computes a Fourier transform over the same symbol set.

The limited number of OFDM inputs allows us to consider the possibility of storing the DFT/IDFT products pre-computed in look-up tables or a ROM memory. In this way multiplications are avoided completely and the hardware is reduced to adders and simpler logic components such as decoders, counters and shifters.

For a given constellation, the mapping type determines the number of possible symbol values – for example: M different symbols for digital M-QAM mapping. The number of OFDM points (N) is also fixed for a given system or a standard. As can be seen in equation (4), the coefficients for the IDFT can be expressed as: exp(i2πkn/N)/N, where k and n are in the range [0, N-1]. Therefore, the number of possible products can be calculated as:

$$ROMSize = N * M \text{ words} \tag{5}$$

For example, an 8-point OFDM transmitter, with a constellation of 4 symbols, will only need two 32-word ROM memories and two 8-word adders. The word length should be at least 16 bit in order to provide sufficient precision. Therefore, we have calculated the coefficients and the products for 4-, 16-, 64- and 256- QAM constellations, and synthesized the according OFDM modules on specific Virtex 5/ 7 FPGA platforms. The amount of memory needed is 32, 128, 512 and 2048 words for each case accordingly.

Similarly as in the previous section, the following variations of transmitter models were implemented:

- model 7: ROM – based transmitter with cyclic prefix, and one transmitted symbol per clock cycle;
- model 8: ROM – based transmitter without cyclic prefix, and one transmitted symbol per clock cycle;

Additionally we experimented with two QAM modules working in parallel in the following models:

- model 11: ROM – based transmitter with cyclic prefix, and two transmitted symbols per clock cycle;

- model 12: ROM – based transmitter without cyclic prefix, and two transmitted symbols per clock cycle;

The only difference between the OFDM sender and receiver are their coefficient values, so their respective synthesis results are very similar in the case of the ROM–based implementation. However, we include these models in the results for completeness, and to make an appropriate comparison with the Cooley-Tukey-based implementations:

- model 5: ROM – based receiver with cyclic prefix, and one received symbol per clock cycle;
- model 6: ROM – based receiver without cyclic prefix, and one received symbol per clock cycle;
- model 9: ROM – based receiver with cyclic prefix, and two received symbols per clock cycle;
- model 10: ROM – based receiver without cyclic prefix, and two received symbols per clock cycle;

In an attempt to optimize the ROM-based OFDM processing unit we try to minimize the memory. For this purpose, we treated every memory output as a logical function of its address bits and then use the Quine-McCluskey algorithm to minimize them. We used the QMC Logic Minimizer tool [22] that outputs the results as a minimal canonical sum of products. Additionally, we developed a greedy algorithm to parse the minimal canonical products form by grouping the common terms. In this way we make a further minimization of the number of gates.

The proposed algorithm can be expressed as a recursive function with the following pseudo code:

```
function groupTerms(sum):
begin
    for each product in sum:
        for each term in product:
            count[term] ++;
    mostFrequentTerm = max(count);
    for each product in sum:
        if (mostFrequentTerm in product) then
            productsWithMostFreqTerm.add(product)
        else
            productsWithoutMostFreqTerm.add(product)
        end if;
    return array( mostFrequentTerm
            , groupTerms(productsWithMostFreqTerm)
            , groupTerms(productsWithoutMostFreqTerm );
end;
```

This algorithm provides the minimal non-canonical form of the Boolean functions and should result in a minimal gate count, while still maintaining high clock speeds. This will be true for ASIC circuits, although it may not achieve the most optimal resource usage on the FPGA platform, because of its slice structure, resulting in more gate stages than the ROM-based solution.

The transmitter models that are implemented and discussed are the following:

- model 9: Minimized ROM – based transmitter with cyclic prefix, and one transmitted symbol per clock cycle;
- model 10: Minimized ROM – based transmitter without cyclic prefix, and one transmitted symbol per clock cycle;
- model 13: Minimized ROM – based transmitter with cyclic prefix, and two transmitted symbols per clock cycle;
- model 14: Minimized ROM – based transmitter without cyclic prefix, and two transmitted symbols per clock cycle;

The respective receiver models are also presented:

- model 7: Minimized ROM – based receiver with cyclic prefix, and one received symbol per clock cycle;
- model 8: Minimized ROM – based receiver without cyclic prefix, and one received symbol per clock cycle;
- model 11: Minimized ROM – based receiver with cyclic prefix, and two received symbols per clock cycle;
- model 12: Minimized ROM – based receiver without cyclic prefix, and two received symbols per clock cycle;

VI. PERFORMANCE ANALYSES AND EXPERIMENTAL RESULTS

The results of the various hardware implementations of OFDM transmitter and receiver models are analysed through reports generated by Xilinx ISE Design Suite tool. Of particular interest for this research is to examine how much Virtex 5 and Virtex 7 FPGA slice logic resources are utilized by the proposed models, and to investigate their performance capabilities (bandwidth). For calculating the bandwidth of the OFDM transmitter and receiver modules we use the following equation:

Bandwidth (b/s) = (Number of bits in symbol) *
(OFDM transmitter/receiver module frequency) *
(Symbol processing frequency) (6)

Symbol processing frequency =

= 8 symbols/8 cycles= 1, for models 1, 8 and 10 of OFDM transmitters and models 1, 6 and 8 of OFDM receivers

= 8 symbols/10 cycles = 0.8, for models 2, 3, 7 and 9 of OFDM transmitters and models 2, 5 and 7 of OFDM receivers

= 8 symbols/4 cycles= 2, for models 4, 12 and 14 of OFDM transmitters and models 3, 10 and 12 of OFDM receivers

= 8 symbols/5 cycles = 1.6, for models 5, 6, 11, 13 of OFDM transmitters and models 4, 9 and 11 of OFDM receivers

Number of bits in symbol = 2 for QAM4, 4 for QAM16, 6 for QAM64 and 8 for QAM256

Figure 6 and Figure 7 show the percentage of slice hardware resources required for Virtex 5 and Virtex 7 FPGA implementation of the proposed OFDM transmitter and receiver models. As the results show, the various transmitter and

receiver models occupy minimal percentage of slice resources, in the range of 3-6% for Virtex 5 implementations, and up to 1% for Virtex 7 implementations. The worst results (11% for Virtex5 and Virtex7 implementations) are obtained for OFDM transmitter models 7-14, and OFDM receiver models 5-12, in the case of QAM 256 symbol mapping, due to the increased ROM memory size (2048x32), required for storing the DFT/IDFT pre-computed products. Although, the minimized ROM approach avoids the use of ROM memories and implements the logic by using a smaller number of look-up tables, a small decrease in FPGA resource usage can be noticed.

Worse results (6% for Virtex5 and Virtex7 implementations) are also obtained for OFDM Cooley–Tukey based transmitter models 2 and 5, since they implement additional buffers in the serial-to-parallel conversion and cyclic prefix insertion module. This buffer keeps the symbols that are waiting to be processed during the one or two cycles delay, caused by the cyclic prefix insertion. However, the other proposed modules utilize minimal resources, allowing them to be further extended and improved.

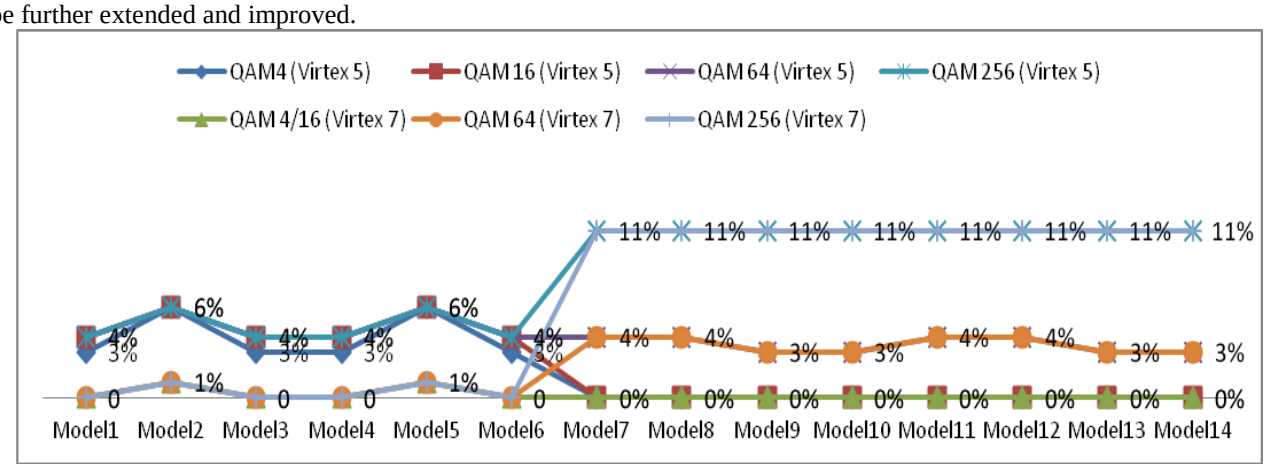


Fig. 6 FPGA slice resource utilization of OFDM transmitter models

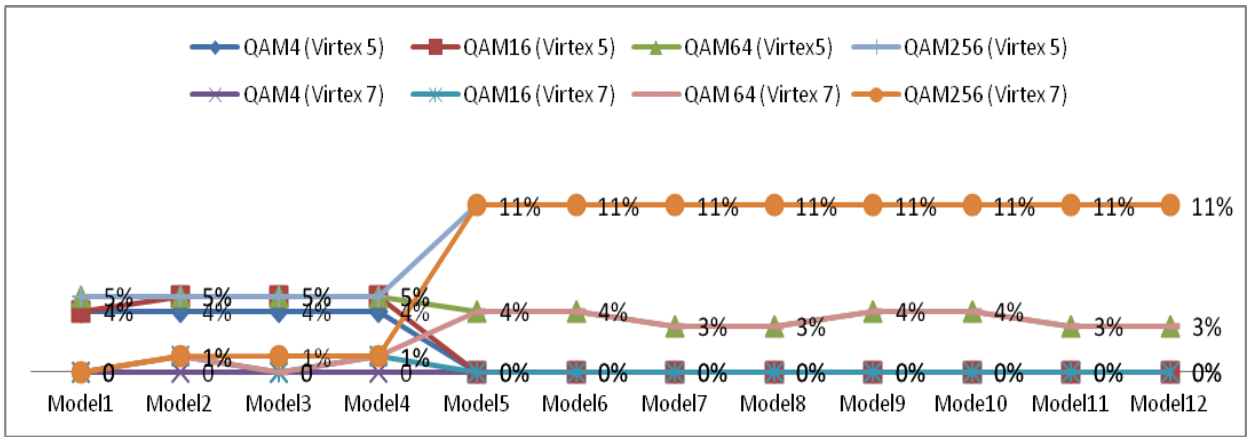


Fig. 7 FPGA slice resource utilization of OFDM receiver models

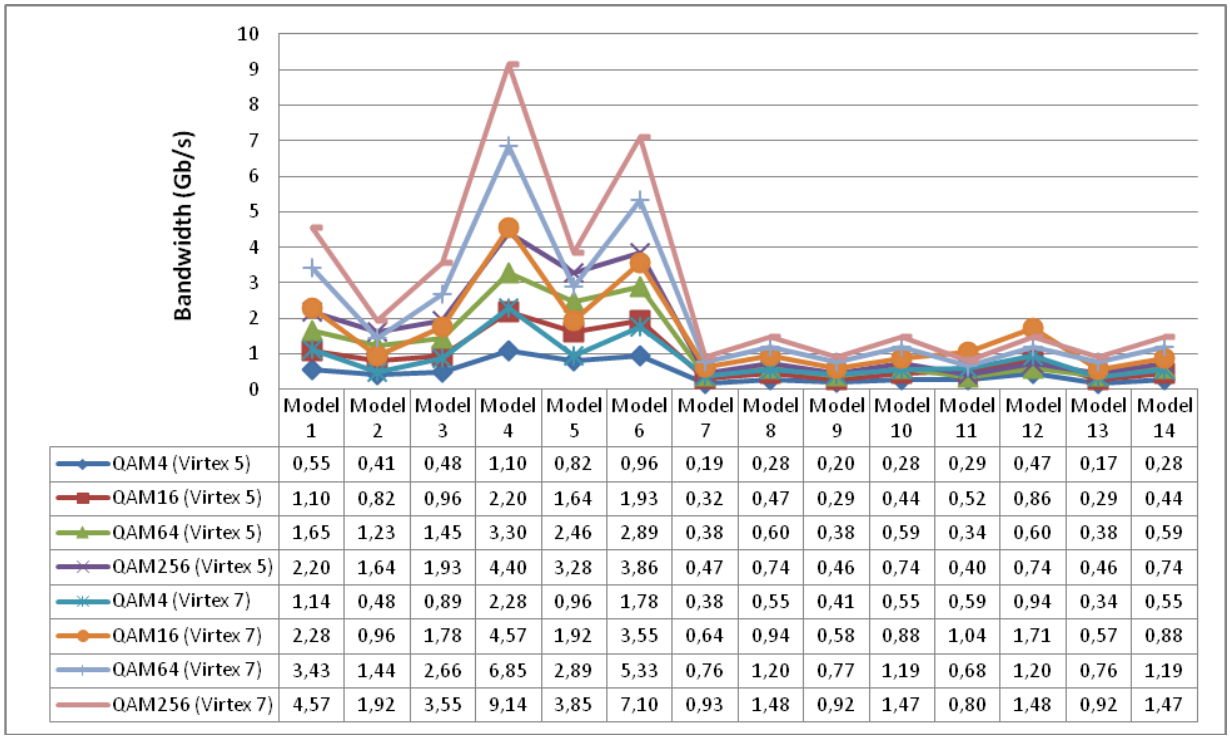
Figure 8 and Figure 9 present the bandwidth results obtained by various OFDM transmitter and receiver models, implemented on Virtex 5 and Virtex 7 FPGAs. According to the result, the highest bandwidth (9.14 Gb/s on sending side, and 9.88 Gb/s on receiving side) is attained with the Cooley–Tukey based model, that doesn't perform cyclic prefix insertion/removal and uses QAM256 mapped symbols. On the other hand, the worst performance of the Cooley–Tukey based models is obtained with the Virtex 5 implemented OFDM transmitter/receiver model 2 that uses QAM4 mapping (0.41/0.37 Gb/s). Generally all Virtex 7 realizations (improved Cooley–Tukey, ROM and Minimized ROM) provide higher bandwidth then the Virtex 5 ones. The most obvious difference between the two boards is the working frequency, which is almost two times faster on the Virtex 7. This is mostly because of the board's improved characteristics.

The Cooley–Tukey based OFDM transmitter models 4, 5 and 6 and OFDM receiver models 3 and 4 process two symbols each cycle, and thus allows bandwidth doubling, in contrast to the OFDM transmitter models 1, 2 and 3 and OFDM receiver models 1 and 2, accordingly. This happens because the IFFT/FFT computations are performed in 4 cycles, so each module is fully utilized all the time, and the overall system works in pipeline mode. Considering this and the results shown in Figure 8 and 9, we suggest the use of model 6 transmitter (5.33 and 7.1 Gb/s) and model 4 receiver (4.89 and 6.52 Gb/s), which include cyclic prefix processing, over QAM64 or QAM256 mapped OFDM frames.

Another important observation is that the ROM-based and minimized ROM-based models (transmitters: 7-14 and receivers: 5-12) generally show poorer characteristics on the Virtex 5/7 FPGA boards. They achieve lower processing throughputs, compared to the Cooley-Tukey models, and do not provide for much lower device utilization. Only when implementing QAM4, QAM16 and QAM64 on the Virtex 5 board, they utilize 1% FPGA slices less than the Cooley-Tukey based models. To a large extent, this result depends on to the nature of the FPGA technology and its synthesis process. The logic described in the VHDL language is automatically mapped by the synthesis tool into FPGA slices.

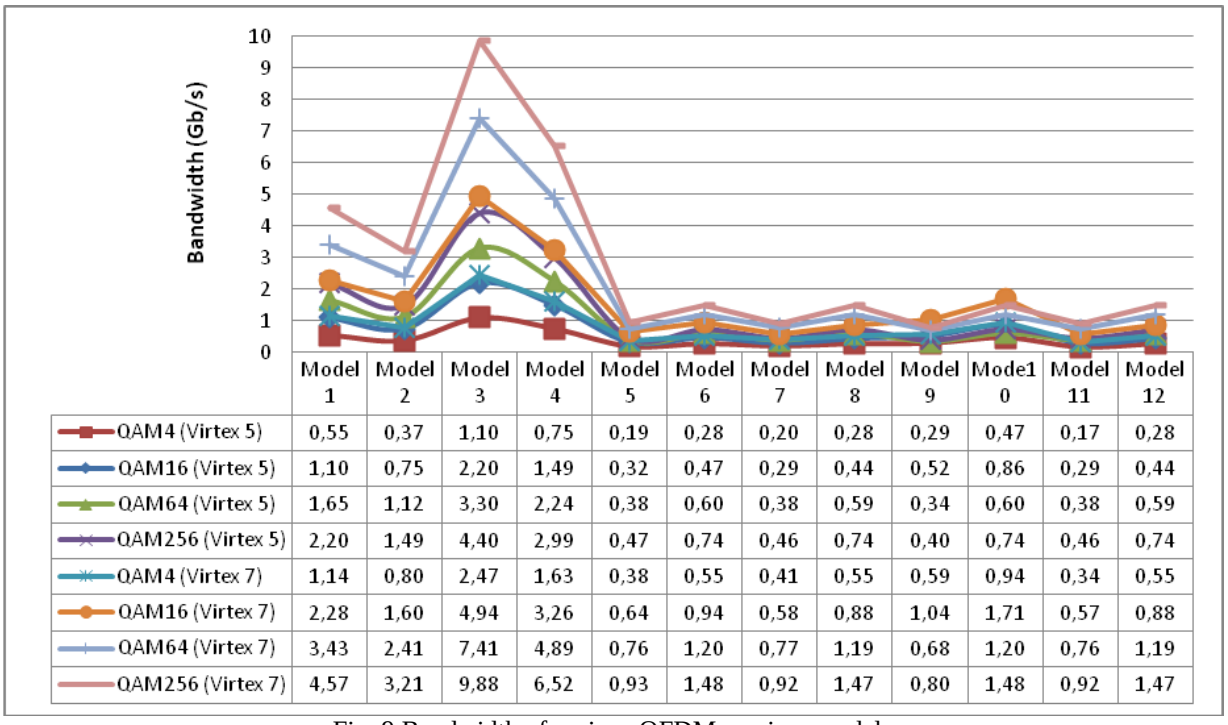
These slices that consist of look-up tables, decoders and flip-flops, have a fixed delay, so the final performance depends on the way they are allocated. Therefore, some improvements may be achieved if more optimal mapping is performed.

Generally, if we compare the performance results of the OFDM transmitter models with the OFDM receivers, we can recognize that the bandwidth attained on the sending side is much higher than on the receiving side. Actually, by comparing the model 6 transmitter with the model 4 receiver, we show that the transmitter achieves by 14.92% (Virtex 5), i.e. 34.02% (Virtex 7) better performances. On the other hand, the ROM and minimized-ROM models show similar results on the sending and receiving side.



	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9	Model 10	Model 11	Model 12	Model 13	Model 14
QAM4 (Virtex 5)	0,55	0,41	0,48	1,10	0,82	0,96	0,19	0,28	0,20	0,28	0,29	0,47	0,17	0,28
QAM16 (Virtex 5)	1,10	0,82	0,96	2,20	1,64	1,93	0,32	0,47	0,29	0,44	0,52	0,86	0,29	0,44
QAM64 (Virtex 5)	1,65	1,23	1,45	3,30	2,46	2,89	0,38	0,60	0,38	0,59	0,34	0,60	0,38	0,59
QAM256 (Virtex 5)	2,20	1,64	1,93	4,40	3,28	3,86	0,47	0,74	0,46	0,74	0,40	0,74	0,46	0,74
QAM4 (Virtex 7)	1,14	0,48	0,89	2,28	0,96	1,78	0,38	0,55	0,41	0,55	0,59	0,94	0,34	0,55
QAM16 (Virtex 7)	2,28	0,96	1,78	4,57	1,92	3,55	0,64	0,94	0,58	0,88	1,04	1,71	0,57	0,88
QAM64 (Virtex 7)	3,43	1,44	2,66	6,85	2,89	5,33	0,76	1,20	0,77	1,19	0,68	1,20	0,76	1,19
QAM256 (Virtex 7)	4,57	1,92	3,55	9,14	3,85	7,10	0,93	1,48	0,92	1,47	0,80	1,48	0,92	1,47

Fig. 8 Bandwidth of various OFDM transmitter models



	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9	Model 10	Model 11	Model 12
QAM4 (Virtex 5)	0,55	0,37	1,10	0,75	0,19	0,28	0,20	0,28	0,29	0,47	0,17	0,28
QAM16 (Virtex 5)	1,10	0,75	2,20	1,49	0,32	0,47	0,29	0,44	0,52	0,86	0,29	0,44
QAM64 (Virtex 5)	1,65	1,12	3,30	2,24	0,38	0,60	0,38	0,59	0,34	0,60	0,38	0,59
QAM256 (Virtex 5)	2,20	1,49	4,40	2,99	0,47	0,74	0,46	0,74	0,40	0,74	0,46	0,74
QAM4 (Virtex 7)	1,14	0,80	2,47	1,63	0,38	0,55	0,41	0,55	0,59	0,94	0,34	0,55
QAM16 (Virtex 7)	2,28	1,60	4,94	3,26	0,64	0,94	0,58	0,88	1,04	1,71	0,57	0,88
QAM64 (Virtex 7)	3,43	2,41	7,41	4,89	0,76	1,20	0,77	1,19	0,68	1,20	0,76	1,19
QAM256 (Virtex 7)	4,57	3,21	9,88	6,52	0,93	1,48	0,92	1,47	0,80	1,48	0,92	1,47

Fig. 9 Bandwidth of various OFDM receiver models

VII. CONCLUSION

The throughput achievable with a single ROM-based OFDM module on an FPGA board is limited by the board's technology and its working frequency. In order to reach higher transmission speeds, some form of parallelization becomes a necessity. Therefore, the Cooley–Tukey based OFDM models perform IFFT/FFT computations in 4 cycles, in

such a way that the overall OFDM system can work in pipeline mode. Actually, the appropriate Cooley–Tukey based transmitter and receiver models which include cyclic prefix processing, over QAM64 or QAM256 mapped OFDM frames achieve throughput of 5.33/7.1 Gb/s and 4.89/6.52 Gb/s on Virtex 5/7 FPGA board. Higher performances of up to 140 (130) Gb/s may be obtained if for example, we can fit 20 instances of the model 6 transmitter (or model 4 receiver) on the Virtex 7 board, and put them to work in parallel. Of course this will introduce an additional output delay due to the use of input and output buffering. If we consider the minimized ROM-based implementation, its main advantage over the other models should be observed when its logic is implemented in ASIC technology, and this is planned as a next phase in our research.

REFERENCES

[1] Manushree Bhardwaj, Arun Gangwar, Devendra Soni, "A Review on OFDM: Concept, Scope & its Applications", *IOSR Journal of Mechanical and Civil Engineering*, Volume 1, Issue 1, pp. 07-11, 2012.

[2] Louis Lit win, Michael Pugel, "The principles of OFDM", *RF Signal Processing Journal*, pp 30-48, 2001.

[3] Marius Oltean, "An Introduction to Orthogonal Frequency Division Multiplexing", Analele Facultatii din Oradea, Fascicola Electrotehnica, Sectiunea Electronica, pp. 180-185, 2004.

[4] Charan Langton, "Orthogonal Frequency Division Multiplex (OFDM) Tutorial", Tutorials in Communication Engineering, 2004.

[5] Mark Elo, "Orthogonal Frequency Division Multiplexing", White Paper, 2007.

[6] Predrag Spasić, Ljubomir Zelenbaba, "Principi OFDM sistema za radio komunikacije", In Proc. of *TELFOR'02*, 2002.

[7] Ahmad R. S. Bahai and Burton R. Saltzberg, *Multi-Carrier Digital Communications - Theory And Applications of OFDM*, USA: Kluwer Academic Publishers, 2002.

[8] Manjunath Lakkannavar, Ashwini Desai, "Design and Implementation of OFDM (Orthogonal Frequency Division Multiplexing) using VHDL and FPGA", *International Journal of Engineering and Advanced Technology (IJEAT)*, Volume-1, Issue-6, August 2012.

[9] Somayeh Mohammady, Nasri Sulaiman, Roslina M. Sidek, Pooria Varahram, M. Nizar Hamidon, "FPGA Implementation of Inverse Fast Fourier Transform in Orthogonal Frequency Division Multiplexing Systems", Universiti Putra Malaysia (UPM)

[10] Kamaru Adzha Bin Kadiran, "Design and implementation of OFDM transmitter and receiver on FPGA hardware", M. Eng. thesis, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 2005.

[11] Mounir Arioua, Said Belkouch, Mohamed Agdad, Moha M'rabet Hassani, "VHDL implementation of an optimized 8-point FFT/IFFT processor in pipeline architecture for OFDM systems", In proc. of *Multimedia Computing and Systems (ICMCS'11) International Conference*, 2011.

[12] Chandan. M, S. L. Pinjare, Chandra Mohan Umapthy, "Optimised FFT design using Constant Co-efficient Multiplier", *International Journal of Emerging Technology and Advanced Engineering*, 2012.

[13] M. Krstic, M. Piz, M. Ehrig, E. Grass, "OFDM Datapath Baseband Processor for 1 Gbps Datarate", in Proc. of *IFIP/IEEE Intl. Conf. on VLSI and System-on-Chip (VLSI-SoC'08)*, 2008.

[14] Michael Bernhard, Joachim Speidel, "Implementation of an IFFT for an Optical OFDM Transmitter with 12.1 Gbit/s ", *ITG Symposium on Photonic Networks*, Germany, 2010.

[15] Dischler Roman, Klekamp A., Bernhard M.,Efinger Daniel, "Realisation of a real-time 12.1 Gb/s optical OFDM transmitter and its application in a 109 Gb/s transmission system with coherent reception", *35th European Conference on Optical Communication*, Germany, 2009.

[16] R. Schmogrow, M. Winter, B. Nebendahl, D. Hillerkuss, J. Meyer, M. Dreschmann, M. Huebner, J. Becker, C. Koos, W. Freude1, and J. Leuthold, "101.5 Gbit/s Real-Time OFDM Transmitter with 16QAM Modulated Subcarriers", *OSA/OFC/NFOEC 2011*, vol. A247, pp. 529-551, April 1955.

[17] Shaminder K.; Rajesh M., "FPGA Implementation of OFDM Transceiver using FFT Algorithm", *International Journal of Engineering Science and Technology (IJEST)*, April 2012

[18] Toal C., Sezer S., Burns D., Xaio P., Fusco V., "A 1Gbps FPGA-based wireless baseband MIMO transceiver", *IEEE International SOC Conference (SOCC'12)*, pp. 202-207, September 2012.

[19] David Bishop. (2010)Fixed point package user's guide. [Online]. Available: http://www.eda.org/fphdl/fixed_ug.pdf.

[20] John L. Hennessy, David A. Patterson, *Computer architecture: a quantitative approach*, USA:Elsevier, 2012.

[21] Peter J. Ashenden, *The Designer's Guide to VHDL Third Edition (Systems on Silicon)*, California, USA: Morgan Kaufmann Publishers, 2008.

[22] (2013) SourceForge website: QMC Logic Minimizer. [Online]. Available: http://sourceforge.net/projects/qmclm/

