FPGA implementation of a 10 GS/s variable-length FFT for OFDM-based optical communication systems

Julián S. Bruno, Vicenç Almenar and Javier Valls

Abstract

The transmission rate in current passive optical networks can be increased by employing Orthogonal Frequency Division Multiplexing (OFDM) modulation. The computational kernel of this modulation is the fast Fourier transform (FFT) operator, which has to achieve a very high throughput in order to be used in optical networks. This paper presents the implementation in an FPGA device of a variable-length FFT that can be configured in run-time to compute different FFT lengths between 16 and 1024 points. The FFT reaches a throughput of 10 GS/s in a Virtex-7 485T-3 FPGA device and was used to implement a 20 Gb/s optical OFDM receiver.

Index Terms

FFT; FPGA implementation; OOFDM; PON; Real-time signal processing; IM/DD; QAM.

I. Introduction

The current wired and wireless access techniques will not be able to support the fast growing bandwidth demand in the access network market. Passive optical networks (PONs) are seen as one of the best candidates to implement broad-band access networks thanks to their low cost, high reliability and easy maintenance. Although nowadays these networks make use of low complexity modulation formats, in the near future it will be necessary to implement higher order modulation formats to increase the data throughput of these networks. So, Orthogonal Frequency Division Multiplexing is a clear candidate, as it was recently introduced into fiber communications due to its flexible dynamic bandwidth allocation, high spectral efficiency (SE) and strong resistance to chromatic dispersion (CD) and polarization mode dispersion (PMD) [1], [2].

OFDM is a multicarrier modulation technique where each symbol is composed of N samples which are generated by performing an N-point inverse fast Fourier transform (IFFT) on N complex data symbols. Thanks to the multiplexing, each subcarrier works at a data rate much lower than the one needed by a serial modulation format, given that CD increases with the square of the data rate while PMD increases linearly with the data rate [3], the electronic dispersion compensation (EDC) required at the receiver is easier when OFDM is used, this fact allows using high order modulation formats which increases the SE. Before transmission, a guard interval composed of the last N_{CP} samples is prepended to the OFDM symbol to make the system robust to dispersive channels, this guard extension is called cyclic prefix (CP). The length of the FFT is an important parameter when designing an OFDM communication system, for example, the ratio between N and N_{CP} determines the SE. As the CP length is selected according to the channel dispersion, which depends on the length of the optical link, once it is fixed, the FFT length determines the maximum achievable SE. Another benefit of using long FFT lengths is a lower out of band power interference, which simplifies the design of the analog filtering needed to reduce interferences with adjacent channels. On the other hand, long FFTs can introduce some detrimental effects, as high peak-to-average power ratio and high latency. Moreover, in the synchronization stage, a long CP helps in

J. S. Bruno is with the Laboratorio de Procesamiento Digital (DPLab), Universidad Tecnológica Nacional, Buenos Aires 1179, Argentina (e-mail: jbruno@electron.frba.utn.edu.ar).

V. Almenar and J. Valls are with the Instituto de Telecomunicaciones y Aplicaciones Multimedia (ITEAM), Universitat Politècnica de València, Valencia 46022, Spain.

the time synchronization phase, but a long FFT makes the system more sensitive to sampling frequency offset synchronization errors. As can be seen, selecting the FFT length is a trade-off between positive and negative effects in the optical communications system.

Our real-time OFDM receiver is equipped with 10-bit ADC chip with a maximum sampling rate of 5 GS/s. The EV10AQ190 ADC sends to the FPGA 4 samples in parallel at a rate of 1.25 Gbps via 40 low-voltage differential signaling (LVDS). The FPGAs has a dedicated serial-to-parallel converter (ISERDES) that can create a 4-bitwide parallel word to obtain 16 samples in parallel using a clock frequency of 312.5 MHz, which it is supported by most current FPGAs.

The FFT is one of the most challenging operators required in a very high-speed OFDM system. In 1969 Bergland [4] made a classification of FFT hardware architectures. They were divided into four big groups: sequential (memory-based), cascade (pipelined), parallel and array (fully parallel) implementation. Each of them supposes a different trade off between performance and complexity (area). To maximize performance, array hardware architectures should be used. Array implementation [5]–[7] maps each addition/multiplication of the FFT flow graph to an adder/multiplier in hardware. The problem is that, for large sizes, too many resources are used and this architecture becomes almost non-viable for FPGA devices. Sequential architectures [8], [9] are well suited to minimize the area. This type of hardware architectures calculates the FFT iteratively using data stored in a memory, its low throughput makes it not suitable for high-speed systems. Parallel architectures process N (FFT length) simultaneous samples during $\log_R(N)$ clock cycles, but due to the dependences between computational stages, they can not be pipelined. This fact limits its performance. Finally, pipeline FFT (PFFT) architectures [10] work with R entries and $\log_R(N)$ computational elements (CEs) in parallel, one for each stage of the FFT and R being the base (radix) of the algorithm. PFFT is the option chosen for high-speed applications since it is very regular, can be scaled, parallelized and pipelined.

To increase the processing rate of the PFFT while avoiding the considerable hardware of an fully parallel implementation, it would be desirable to introduce a partially parallel processor. This type of processor would have just sufficient parallel hardware in each stage to obtain the required increase in speed. These partially parallel processors might be called parallel pipeline FFT (PPFFT) [11] allows an increase in the processing speed while retaining the simple structure of the pipeline FFT and has a 100% of hardware utilization efficiency (HUE). PPFFTs [12]–[16] are classified into multi-path delay commutator (MDC) and single-path delay feedback (SDF) [17], MDC was shown to be the most efficient approach for high-throughput implementations [13].

In this paper, we present the architectural design of a radix-4 MDC decimation-in-frequency (DIF) parallel pipeline FFT processor with 4 data-path (O) that processes 16 input data in parallel (P) based on the architecture defined in [11]. The FFT processor is designed to compute variable-length FFT of 16, 64, 256 and 1024 points, which will allow us to test the effect of the FFT length in the optical OFDM (OOFDM) communication system. We propose a new reordering block that allows us to input or output the samples in natural order for the different FFT lengths and that is easily generalized for different parallelization degree and radix. The presented FFT architecture is more area-time efficient than other published high-speed FFT architectures when implemented in Virtex FPGA devices. The FFT processor reaches a throughput of 10 GS/s when is implemented in a Virtex-7 485T-3 and it was applied to implement a real-time optical OFDM receiver that operates at 5 GS/s.

This paper is organized as follows. In Sect. (II) we present the radix-4 decimation-in-frequency PPFFT architecture. Then in Sect. (III) the design of R4MDC DIF PPFFT variable length (16 through 1024) architecture is detailed. Sect. (IV) describes the hardware implementation and comparisons with other architectures from the literature. Sect. (V) shows the experimental setup used to evaluate our FFT design in a high-speed real-time optical receiver and the obtained results. Finally, the conclusions are summarized in Sect. (VI).

II. Parallel pipeline FFT fixed length algorithm

In [11] the radix-2 MDC decimation-in-time PPFFT architectures are described for: one, two and four data paths, and it is extended to radix-4 for a specific case. Fig. 1 shows the radix-4 MDC PPFFT generalized architecture with 4 datapaths (O=4) developed in this work, based on the ideas and guidelines described in [11]. An important feature of this architecture is that only two types of elements are used: computational elements (CE) and delay commutators (DC). It has $n = \log_4(N)$ stages, where each stage, numbered with index k, consists of O CEs with different twiddle factors. The switching speed and the number of delay elements of the DCs depends on the interconnected stages. Only minor changes are required to implement forward and inverse transforms with power of 4 lengths (R=4).

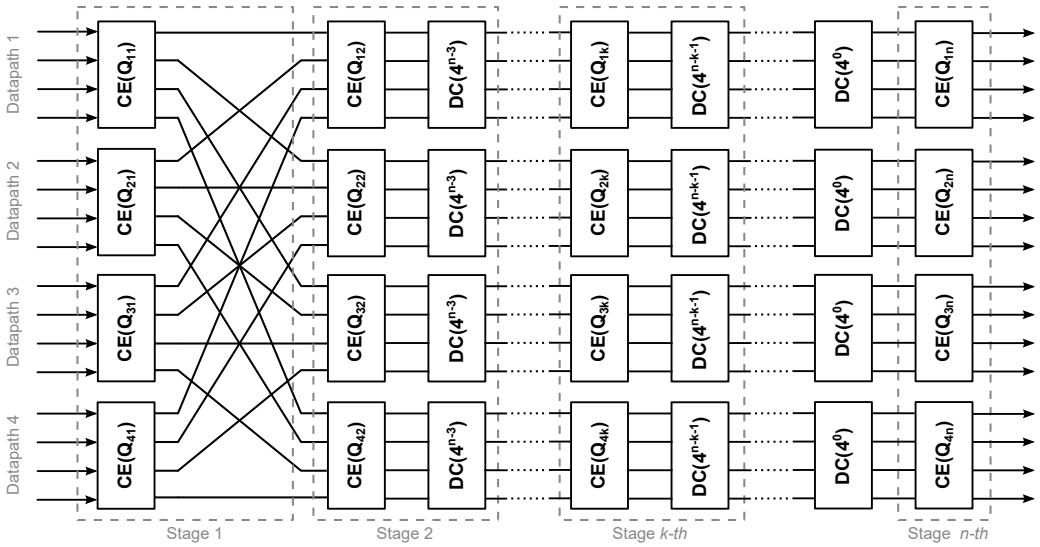


Fig. 1. R4MDC PPFFT architecture with 4 data path.

This architecture differs from a radix-4 pipelined FFT (R=4 and O=1) in that it allows to process 16 simultaneous samples (P=16, R=4 and O=4), quadrupling the throughput and keeping the HUE at 100%. The first and most expensive inter-stage data exchanger disappears, the amount of CEs is quadrupled (O=4), but not the amount of DCs. Both inputs and outputs of the algorithm need to be ordered in a way that is not natural, it depends on the FFT size.

In Fig. 1 it can be seen that after data exchange between first and second CEs, the 4 paths do not cross again. This happens regardless of the size of the FFT and, except for the first stage, the CEs and DCs of the 4 data path are equal. For each stage twiddle factors are a decimated version of the previous one and the last stage has all its rotation factors equal to unity. All these features allow us to design an FFT processor of configurable size with minimum additional logic.

A. Computational element

The computational element is a block that performs the FFT algorithm complex-number mathematical operations, it is usually called butterfly. A radix-4 CE corresponds to a four-point discrete Fourier transform (DFT) followed by a multiplication of each output by a complex twiddle factor. Thus there are four complex inputs (A, B, C and D), and four complex outputs (E, F, G and H), given by:

$$\begin{aligned} \mathrm{E} &= (\mathrm{A} + \mathrm{B} + \mathrm{C} + \mathrm{D}) * W_N^{0Q} \\ \mathrm{F} &= (\mathrm{A} - j\mathrm{B} - \mathrm{C} + j\mathrm{D}) * W_N^{1Q} \\ \mathrm{G} &= (\mathrm{A} - \mathrm{B} + \mathrm{C} - \mathrm{D}) * W_N^{2Q} \\ \mathrm{H} &= (\mathrm{A} + j\mathrm{B} - \mathrm{C} - j\mathrm{D}) * W_N^{3Q} \end{aligned} \tag{1}$$

These equations have 6 variables: 4 are the complex inputs to the block, one is Q and the other is N. The twiddle factor is represented as W_N=$e^{-j2\pi/N}$, a complex number with module one and variable phase. Q is an algorithm-position-dependent index that defines the value of 3 twiddle factors, the first twiddle factor is always 1 (i.e., W_N^0). Note that the radix-4 butterfly used for the standard formulation of the radix-4 FFT is sometimes referred to as a dragonfly [18] in the technical literature.

The single dragonfly described in Eq. (1) requires 12 complex-number additions, 3 complex-number multiplications and 8 trivial multiplications $(j, -j, -1)$. The 12 complex-number additions (24 real adders) and the 8 trivial multiplications are implemented with 16 real adders due to the fact that the operations are rearranged as Eq. (2). Fig. 2 shows the scheme of the computational element. It has 2 stages with 8 real adders each. Therefore, 8 adders are saved with respect to the direct implementation of Eq. (1).

$$
\begin{aligned}
Re(\mathrm{A}+\mathrm{B}+\mathrm{C}+\mathrm{D}) &= [Re(\mathrm{A})+Re(\mathrm{C})]+[Re(\mathrm{B})+Re(\mathrm{D})]\\
Im(\mathrm{A}+\mathrm{B}+\mathrm{C}+\mathrm{D}) &= [Im(\mathrm{A})+Im(\mathrm{C})]+[Im(\mathrm{B})+Im(\mathrm{D})]\\
\\
Re(\mathrm{A}-j\mathrm{B}-\mathrm{C}+j\mathrm{D}) &= [Re(\mathrm{A})-Re(\mathrm{C})]-[Im(\mathrm{D})-Im(\mathrm{B})]\\
Im(\mathrm{A}-j\mathrm{B}-\mathrm{C}+j\mathrm{D}) &= [Im(\mathrm{A})-Im(\mathrm{C})]-[Re(\mathrm{B})-Re(\mathrm{D})]\\
\\
Re(\mathrm{A}-\mathrm{B}+\mathrm{C}-\mathrm{D}) &= [Re(\mathrm{A})+Re(\mathrm{C})]-[Re(\mathrm{B})+Re(\mathrm{D})]\\
Im(\mathrm{A}-\mathrm{B}+\mathrm{C}-\mathrm{D}) &= [Im(\mathrm{A})+Im(\mathrm{C})]-[Im(\mathrm{B})+Im(\mathrm{D})]\\
\\
Re(\mathrm{A}+j\mathrm{B}-\mathrm{C}-j\mathrm{D}) &= [Re(\mathrm{A})-Re(\mathrm{C})]+[Im(\mathrm{D})-Im(\mathrm{B})]\\
Im(\mathrm{A}+j\mathrm{B}-\mathrm{C}-j\mathrm{D}) &= [Im(\mathrm{A})-Im(\mathrm{C})]+[Re(\mathrm{B})-Re(\mathrm{D})]
\end{aligned}
\tag{2}
$$

On the other hand, each complex-number multiplication of the dragonfly is implemented with 4 real multipliers and 2 real adders, as show in Fig. 2. The total cost of the CE is 8 complex adders and 3 complex multipliers (22 real adders and 12 real multipliers).

Within stage k, for k = 1, 2, ..., $\log_4(N)$, there are 4^{k-1} groups of dragonflies, with $(N/4)/4^{k-1}$ dragonflies per group. The twiddle factors for different groups in the same stage are equal [19]. The Q values per group for different FFT sizes can be expressed in steps of 4^{k-1} from 0 to $N/4-1$, and these values are summarized in Table I.

When N=1024, the values of Q for stage 2 (4 groups) are 0:4:255 (first:step:last) and when N=256 the values of Q for stage 1 (1 group) are 0:1:63. Although they seem to have no relation at all, if we analyze the twiddle factor W_N^{1Q}, W_N^{2Q} and W_N^{3Q} for these Q values, we will see that they are equivalent. In three different colors (blue, red and cyan) the equivalent rotation factors are highlighted in Table I.

TABLE I
Q VALUES FOR DIFFERENT FFT SIZES

N \ k	1	2	3	4	5
1024	0:1:255	0:4:255	0:16:255	0:64:255	0
256	0:1:63	0:4:63	0:16:63	0	-
64	0:1:15	0:4:15	0	-	-
16	0:1:3	0	-	-	-

Each computational element in Fig. 1 computes $N/16$ dragonflies in all stages. In the first stage there are $N/4$ different values of Q that are divided equally between Q_{11}, Q_{21}, Q_{31} and Q_{41}. In the second stage there are $N/16$ different values of Q, therefore the values of Q_{12} are the same as Q_{22}, Q_{32} and Q_{42}. The same fact happens in the remaining stages.

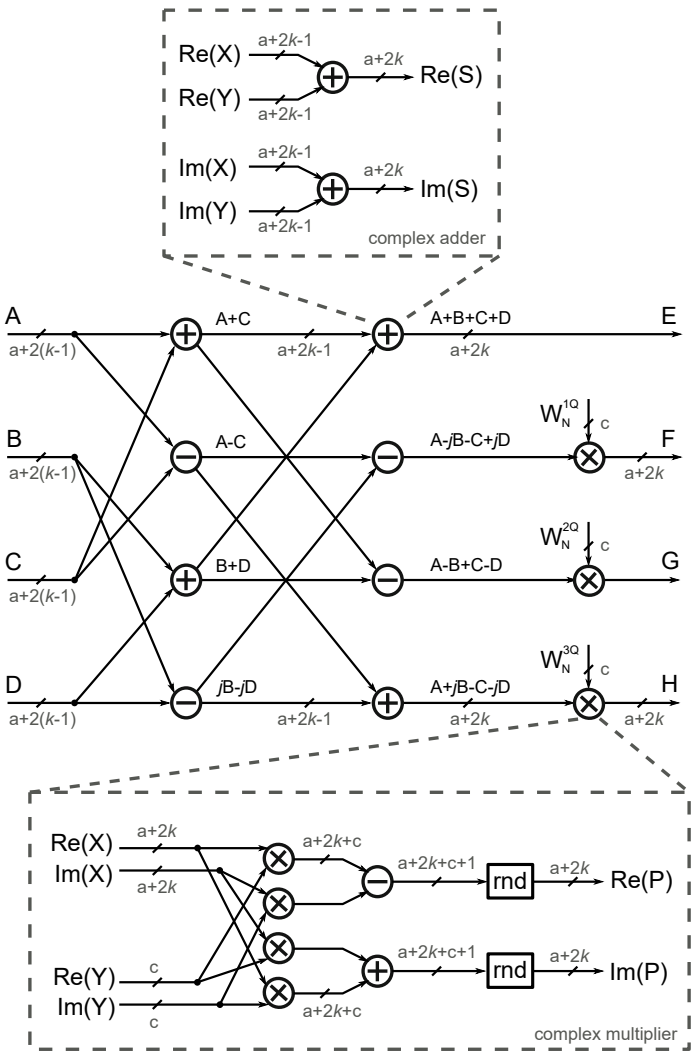


Fig. 2. Computational element radix-4 decimation-in-frequency.

Finally, the total cost of the N-point P-parallel radix-4 PPFFT architecture is calculated as:

$$\begin{aligned} 8 \cdot P/4 \cdot \log_4(N) \quad & \text{complex adders} \\ 3 \cdot P/4 \cdot [\log_4(N) - 1] \quad & \text{complex multipliers} \end{aligned} \tag{3}$$

1) Fixed precision: In the FFT algorithm if the entry ($x_{[n]}$) is bounded between ± 1, the output ($X_{[n]}$) will be bounded between $\pm N$. This implies that the data path has to grow $\log_2(N)$ bits, because otherwise overflows could appear. There are different techniques that avoid the growth of the data path ($|X_{[n]}| < 1$) and the appearance of overflows.

From a hardware-implementation point of view, the complex multipliers (Eq. (1)) of the different CEs are the most critical elements, and from the point of view of the fixed-point error analysis, to truncate or round each product produces a roundoff error. In [18] the analysis about the effect of different scaling techniques to avoid overflows is presented.

By extrapolating the results of Signal-to-noise ratio (SNR) obtained in [18] to the architecture used in this work, it was decided to allow the magnitude of the signal to grow and not to perform any type of scaling. That is, if a bits are used to quantize the input signal (real and imaginary part), $a + \log_2(N)$ bits are required at the output of the FFT. On the other hand, to guarantee a high operating frequency, it was decided to use the hardware adders and multipliers available in the FPGA (*i.e.* the DSP48 component)

to implement the complex multipliers. These two decisions impose certain restrictions on the design that are detailed below. The operand widths of the hardware multipliers available in the FPGA are 25x18 bits. By considering that 18 bits are more than enough to quantize the twiddle factors, 25 bits were chosen for the data-path and 18 bits for the coefficients. These two constraints limit the output-data width of the penultimate stage ($k = \log_4(N) - 1$) to 25 bits, for the real part and for the imaginary part, since the twiddle factors of the last stage are equal to 1 (multipliers are not used). If it is considered that the growth of the data-path per stage is 2 bits, the maximum output data width is 27 bits, for both the real and imaginary part, and it restricts the input-data width to 23, 21, 19 and 17 bits ($27 - \log_2(N)$) for the FFT lengths of 16, 64, 256, 1024 points, respectively.

Figure 2 shows the flow graph of the Eq. (1) and the fixed-point format is detailed, where a represents the FFT input-data width, c the number of bits with which the twiddle factors are quantized and k is the stage number. Because no scaling technique was used, after the two stages of complex adders the data width increases by 2 bits ($a + 2k$). The implementation scheme of the complex multiplier is also detailed in Fig. 2. As a multiplication by a twiddle factor only rotates the input, only the phase of the data is affected, so the input/output-data width of the complex-number multiplier is the same ($a + 2k$). Inside the complex multiplier, the rounding error is only introduced at the output of each adder.

In order to confirm that our finite-precision FFT works well in our OFDM communication system, a simulation of the whole system was done, using as input OFDM signals. In this work, the biggest FFT length is 1024 points, the highest transmitted modulation order is 512-QAM and the input data width is 10 bits (ADC). Figure 3 presents the bit error rate (BER) performance of OFDM against SNR for different finite precision versions of the radix-4 DIF FFT algorithm: floating point, fixed point with rounding and fixed point with truncation. It is observed that the curves of the 3 approaches are practically the same as the theoretical one for the SNR values from 24 dB to 30 dB, what satisfies an error-free transmission using soft-decision forward error correction (SD-FEC) or hard-decision FEC (HD-FEC). Therefore, it is demonstrated that the data widths used and the lack of a scaling technique are adequate to meet the needs of the target system.

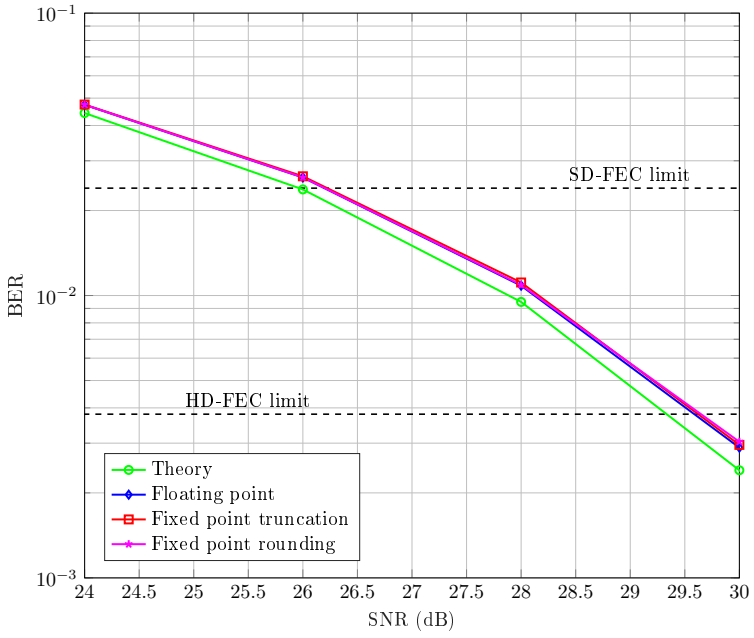


Fig. 3. BER vs SNR for 512-QAM 1024-point FFT 10/20-bit input/output data width and 18-bit twiddle factor data width.

B. Delay Commutator

The delay commutator reorders complex data between computational stages as required by the pipelined FFT algorithm. The DC for radix-R pipelined FFT architecture, as described in [19], is shown in Fig. 4.

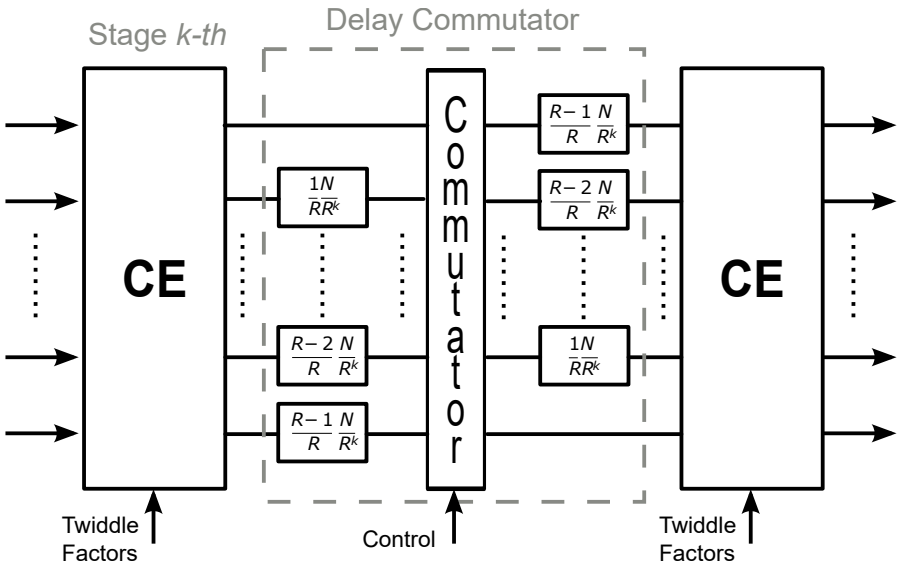


Fig. 4. k-th Delay Commutator stage of radix-R pipelined FFT.

With R=4 and N=4^n, when data enter to the DC they pass through 4 parallel delay lines. No delay is applied to data in the first row, in the second row they are delayed by 4^{n-k-1}, in the third row by $2 \cdot 4^{n-k-1}$ and in the fourth row by $3 \cdot 4^{n-k-1}$. Then data pass through a switch to exchange the rows and columns. Lastly, data are delayed again. If data arrive to the module every T seconds, the switch must modify its selection every $f \cdot T$ seconds; where f=4^{n-k-1}. As N increases, the delays introduced in the data path by the DCs of each stage increase, which implies an increase in the latency of the system. Table II shows the f values for different N values in PFFT with R=4 and in [20] a DC model is presented.

TABLE II
f VALUES FOR DIFFERENT FFT SIZES WITH R=4.

N/n \ k	1	2	3	4
1024/5	64	16	4	1
256/4	16	4	1	-
64/3	4	1	-	-
16/2	1	-	-	-

In parallel pipelined FFT with radix-4 and processing 16 simultaneous samples (P=16, R=4 and O=4), the first (k=1) and most expensive inter-stage data exchanger disappears causing the total latency of the PPFFT architecture (see Fig. 1 and Table II) to decrease significantly with respect to that of the PFFT architecture. The latency of the radix-R PPFFT architecture is given in Eq. (4). It should be noted that it is independent of the radix used and decreases with the degree of parallelism P.

$$\text{Lat} = \frac{N - P}{P} \tag{4}$$

The routing of data that occurs in DC with f=4 is graphically depicted in Fig. 5. Input data numbered from 0 to 63 are shown on four parallel streams at the left of Fig. 5. It transforms four streams of data separated by 16 points into 4 streams where data are separated by 4 points. This process is derived and explained in greater detail in [19].

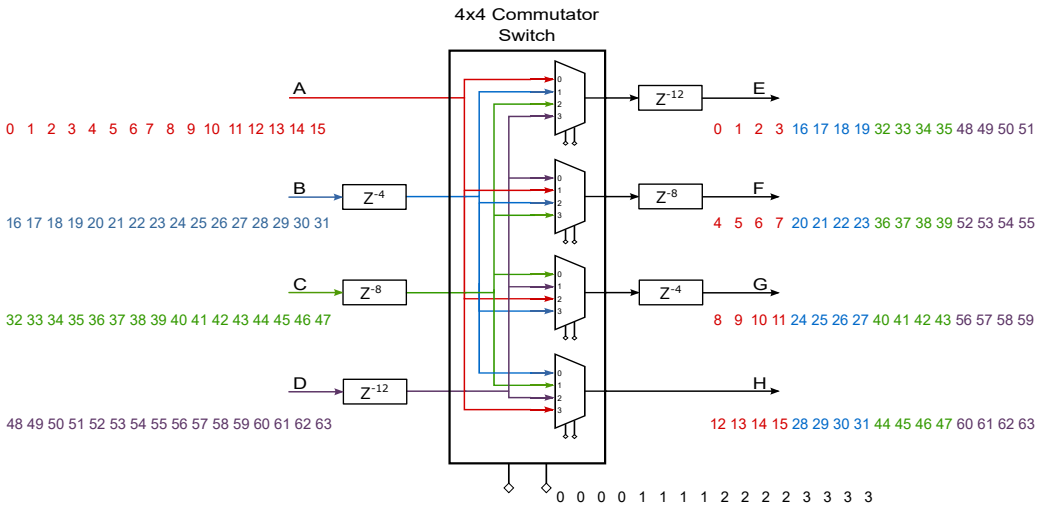


Fig. 5. Data patterns through Delay Commutator with f=4.

C. Input/Output Data Reordering

In the target application, the OFDM communication system, input samples are provided to the FFT in natural order and output samples are also required in natural order. Under these circumstances, reordering circuits are required before and after the FFT to adapt input and output orders. The inputs and outputs of the PPFFT algorithm are not organized in natural ordering, but its order depends on the FFT size and on the number of parallel samples.

Input and output samples can be interpreted as a matrix of index of $P \times L$, where L=N/P. The natural order and input order index matrix for N-point PPFFT with 16 parallel samples are presented in Table III and Table IV respectively, where M=$N/4$. The output order index matrix is obtained applying base-4 digit-reversal [21] transform to the input order index matrix.

TABLE III
NATURAL ORDERING.

Input Index \ Cycle	1	2	.....	$N/16$
0	0	16	.....	$16(L-1)$
1	1	17	.....	$16(L-1)+1$
⋮	⋮	⋮		⋮
15	15	31	.....	$N-1$

It is important to note that to obtain an input order index matrix from a natural order index matrix, it is necessary to perform permutations in both dimensions. The same fact happens to obtain a natural order index matrix from an output order index matrix. As an example, Fig. 6 shows data ordering at input and output for a 64-point PPFFT. It is observed that the rearrangement applied to the input samples is different from the one applied to the output samples and how a rearrangement of samples in both dimensions is necessary.

III. PARALLEL PIPELINE FFT VARIABLE LENGTH ALGORITHM

In this section, we present the architectural design of a run-time configurable variable-length FFT processor which can perform 16-, 64-, 256- and 1024-point FFT.

TABLE IV
INPUT ORDERING.

Input Index \ Cycle	1	2	.....	$N/16$
0	0	1	.....	$L-1$
1	M	$1+M$	.....	$L-1+M$
2	$2M$	$1+2M$	.....	$L-1+2M$
3	$3M$	$1+3M$	.....	$L-1+3M$
4	L	$L+1$	.....	$2L-1$
5	$L+M$	$L+1+M$	.....	$2L-1+M$
6	$L+2M$	$L+1+2M$	.....	$2L-1+2M$
7	$L+3M$	$L+1+3M$	.....	$2L-1+3M$
8	$2L$	$2L+1$	.....	$3L-1$
9	$2L+M$	$2L+1+M$	.....	$3L-1+M$
10	$2L+2M$	$2L+1+2M$	.....	$3L-1+2M$
11	$2L+3M$	$2L+1+3M$	.....	$3L-1+3M$
12	$3L$	$3L+1$	.....	$4L-1$
13	$3L+M$	$3L+1+M$	.....	$4L-1+M$
14	$3L+2M$	$3L+1+2M$	.....	$4L-1+2M$
15	$3L+3M$	$3L+1+3M$	.....	$4L-1+3M$

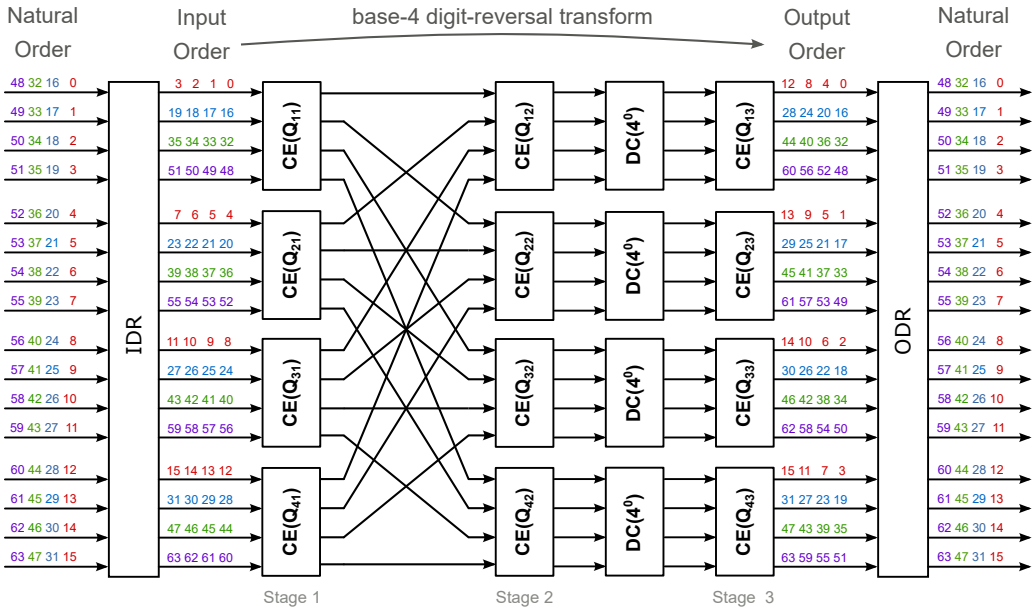
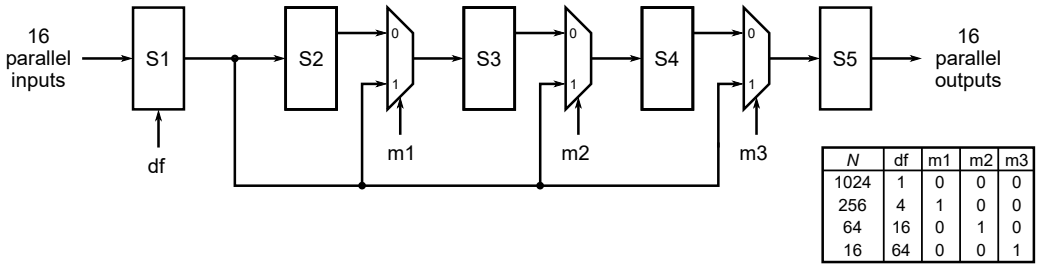


Fig. 6. Input/Output data patterns through 64-point R4MDC DIF PPFFT with 4 datapath (N=64 and P=16).

Analyzing the structure shown in Fig. 1 and the Q and f values presented in Tables I and II, respectively, it can be concluded that a smaller length can be computed eliminating stages of the 1024-point FFT architecture by using dynamic bypass. For this purpose, multiplexers are introduced in the architecture, which allows carrying out a data transfer between stages modifying the data-path. The block diagram of the proposed variable-length FFT processor based on the radix-4 multipath delay commutator architecture is depicted in Fig. 7, where rectangular blocks represent the CE and DC of each stage (named S1 to S5). Note that the blocks S_k correspond with the stages of Fig. 1.

As explained in Section II this architecture has the peculiarity that the DCs of the stage 1 are replaced by direct wired connections between the 1$^{\text{st}}$-stage and the 2$^{\text{nd}}$-stage of the CEs, and these connections are the same in all cases regardless of the FFT length. In order to take advantage of this fact, we need a method to generate the twiddle factors of the first stage for different FFT lengths from the twiddle factors of 1024-point FFT. If Table I is analyzed, it can be seen that the Q values at stage 1 for N=256 are



N	df	m1	m2	m3
1024	1	0	0	0
256	4	1	0	0
64	16	0	1	0
16	64	0	0	1

Fig. 7. R4MDC PPFFT variable length architecture with 4 data path. S_k symbolizes the FFT stages including the CE and DC blocks.

the same as N=1024 but decimated by 4. For N=64 the decimation factor is 16 and for N=16 is 64. In general, df=1024/N, where df is the decimation factor. As an example, for the case of N=64 a bypass must be created between the CEs inputs of stage 2 and 4 (m=010), and the twiddle factors of the first stage are decimated by df=16. Note that the reordering of the input and output data also change.

IV. Hardware Implementation

This architecture was designed to work with different Xilinx FPGA devices and to obtain the best possible throughput, therefore it is a parameterized design, with a very careful segmentation and a great control of the fanout, so that the maximum working frequency of the design is very close to the maximum switching frequency of the slowest logic element inside the FPGA.

The FFT module was designed in fixed point arithmetic using System Generator and the input and output sample reordering modules were described in the Verilog language. The Xilinx System Generator for DSP is a plug-in to Simulink that enables designers to develop high-performance DSP systems for Xilinx FPGAs. The FFT length can be configured at run-time and it can take the following values: 16, 64, 256 and 1024. All modules were verified using a MATLAB finite precision model and implemented on different Xilinx Virtex FPGAs using Xilinx Vivado 2016.3 and Ise 14.7 software tools.

A. Computational element

This block is the core of the PPFFT architecture and it is the only one that performs arithmetic operations (complex addition, subtraction and multiplication) and is repeated in each stage of the systolic architecture, because of that, the dragonfly design is critical.

To implement the complex-number multiplier involved in Eq. (1), 4 multipliers and 2 adders are used. There are less expensive ways to implement a complex multiplier, but when using the DSP48 blocks, this is the optimal way, as the whole implementation of the complex-number multiplier can be done with resources of the DSP48 block. The complex-number multiplier has a latency of 4 cycles. To implement the additions and the trivial multiplications (left side of Fig. 2) 16 adders (LUT-based) pipelined in two stages are employed.

The twiddle factors ([-1, +1]) are pre-calculated and stored in small distributed logic ROMs and their outputs are registered to compensate for the delay in the data path, decrease the possibility of a critical path and give greater flexibility to the placement. These factors are constant trigonometric coefficients that multiply data modifying its phase but not its amplitude. These phase rotation factors have 1 integer bit and 17 fractional bits.

If rounding were used at the output of the CE's adders of Fig. 2, six additional adders would be required for each CE. This increases the hardware resources used and the latency of the operator. However, if truncation is used, the FEC-limit is met (see Fig. 3) and the computational cost is not increased. For this reason, it is decided to use truncation.

Finally, the module inputs and outputs are registered to reduce critical paths and the total latency of the CE is 9 clock cycles. The only exception is the last stage CEs, which does not have complex multipliers and whose total latency is 3 clock cycles.

B. Delay commutator

The delay elements are implemented with shift register LUTs (SRLs) followed by one flip-flop (FF). The module inputs/outputs data were registered to reduce critical paths, often a critical path is generated between input A and output H of Fig. 5. The fan-out for the input selection signal is 4 times the input data width. This large fan-out is reduced by using a balanced register tree.

The selection signals for the DCs that are along the FFT architecture are generated from the selection of two bits of a 6-bit counter that works with a clock frequency equal to the sampling frequency divided by 16. For example, for a value of f=4 (see Fig. 5), bits 2 and 3 of the counter are used as selection signals.

C. Input/Output data reordering

The input and output rearrangements are different and depend on the FFT length and the number of parallel samples, as commented in Section II-C, and can be modified in real time. Only few works in the literature [12], [16] propose a solution for these two modules, taking into account a variable length and the continuous process of P samples in parallel; both of them oriented to VLSI designs. This level of complexity forced us to design a general configurable solution that would work for the 8 cases.

The proposed solution is based on a two-dimensional matrix of $16 \times N/16$ for each output of the data reordering module, where rows represent the 16 inputs (spatial dimension) that are available in the same clock cycle and columns represent those inputs in different clock cycles (time dimension). This two-dimensional matrix stores the N samples of one FFT, as shown in Table III. As it was explained in Sect. (II-C) it is necessary to perform permutations in both spatial and time dimensions to obtain the output order from the input order. Besides, there are cases (*e.g.* 256 or 1024-FFT) where each output depends on any of the 16 inputs, as can be seen in Table IV (*e.g.* for N=1024 and P=16, L=64 and M=256). For this reason, this two-dimensional matrix has to be replicated 16 times. On the other hand, to avoid the loss of samples due to overwriting (outputs cannot be generated until all the N samples are already stored) the technique of double buffering or ping-pong buffering is also employed. Therefore, a total of 16 matrices of $16 \times 2N/16$ samples are needed.

To implement this two-dimensional matrix, we use one block RAM simple dual port with a width of $16 \cdot d$ bits (d is the sample word length) and memory depth of $2N/16$. Therefore, to select one sample it is necessary to address one column of the RAM and select the proper row (a d-bit sample) with a 16-to-1 multiplexer. The generation of the RAM reading address and the multiplexer input selection are done by means of ROMs that have this information stored for the different FFT lengths and the different outputs of the module. Consequently, 4 ROMs for the addresses and 4 ROMs for the selection of multiplexers are used, besides, a counter is used to generate the input address. As it was said before, this two-dimension matrix is replicated 16 times as depicted in Fig. 8, which shows the internal scheme of the reordering module. This design takes advantage of the use of block RAMs, which are available in FPGA devices.

The outputs of the multiplexers and the inputs and outputs of the memories were registered as well as the fan out of the control signals were reduced to obtain a working frequency as high as possible.

D. FPGA-implementation results

This section shows implementation results in different FPGA devices of the designed variable-length and fixed-length FFT core.

The total latency of implemented PPFFT with fixed or variable length is given by the sum of the DCs delays of each stage (see Eq. (4)) and the delays introduced by registers used to pipeline the architecture (which are required to guarantee the maximum operating frequency of the FPGA device). The total latency of our implementation is given in Eq. (5).

$$\text{Lat}_{\text{hw}} = \text{Lat} + \text{pipeline lat.} = \frac{N-P}{P} + (11n - 8) \tag{5}$$

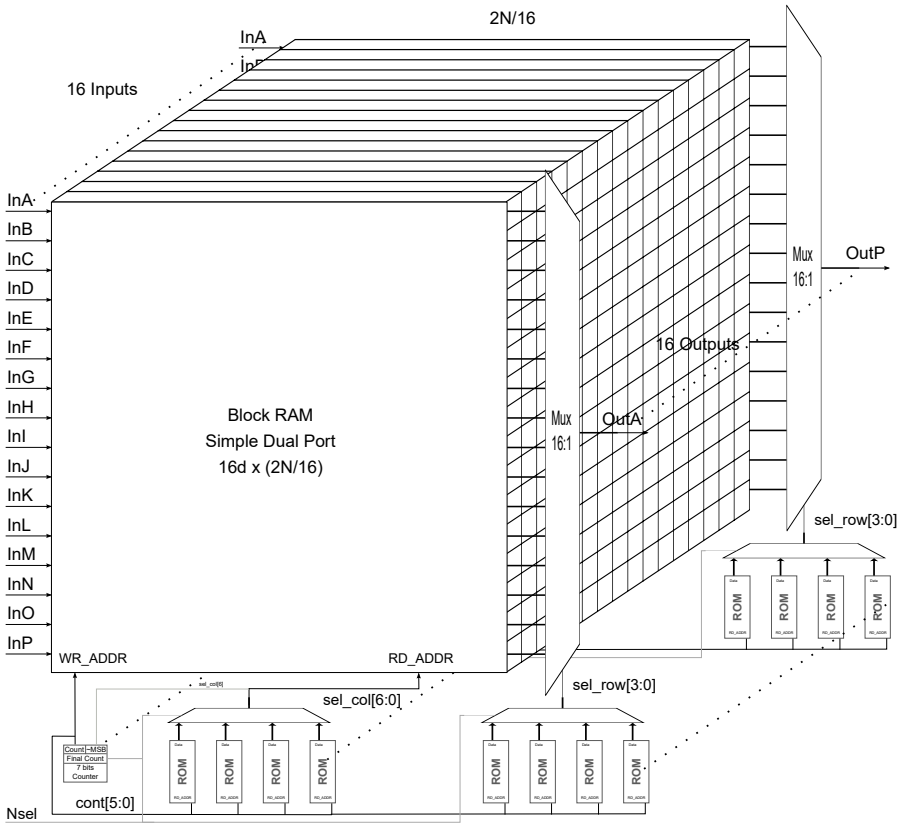


Fig. 8. Input/Output buffer reorder hardware implementation.

1) Variable-length: Using a Virtex-7 485T-3, which has a maximum operating frequency (MOF) of 740 MHz, the FFT module requires 34,441 FFs (5.67%), 13,897 LUTs (4.58%), 8,204 Slices (10.81%), 148 BRAMs (14.37%) and 192 DSP48s (6.86%), and the maximum clock frequency is 645 MHz, which guarantees a throughput of 10.23 GS/s. The FFT was configured to process data in natural order with 20-bit and 40-bit word length for complex inputs and outputs, respectively.

Reference [22] presents a radix-2 MDC PPFFT module that processes 16 inputs in parallel with dynamic length programmability (4 through 1024), an input word length of 16-bits, flow control and fixed point arithmetic. The proposed FFT architecture (without input/output reorder hardware) is implemented in two FPGA devices from Xilinx and these results are compared to [22] in Table V. The amount of DSP48 used by [22] is almost double, the latency is 30% higher and the maximum frequency is 23% (V5) and 32% (V7) lower than the design proposed in this work. The throughput improvement with respect to [22] is approximately 1.2 GS/s and 3.3 GS/s for Virtex-5 (MOF=500 MHz) and Virtex-7 (MOF=740 MHz) respectively.

TABLE V
COMPARISON OF THE PROPOSED R4MDC TO R2MDC ARCHITECTURE FOR THE COMPUTATION OF A PPFFT VARIABLE LENGTH 16-BIT INPUT.

Device	Author	Slices	DSP48s	BRAMs	Max. Freq. (MHz)	Max. Latency (cycles)	Throughput (MS/s)
Virtex-5 SX95T-2	[22]	-	408 (63%)	8(3%)	251	145	4,016
	Proposed	5,985 (41%)	192 (30%)	0	324.5	110	5,191
Virtex-7 485T-3	[22]	-	408 (15%)	8 (1%)	443	145	7,088
	Proposed	5,365 (7%)	192 (6%)	0	645.2	110	10,323

In [12], [16] variable length FFT designs are presented for VLSI devices and in [23], [24] for FPGA devices. These designs reach few GS/s, as they are focused for WiMax and LTE applications, so they cannot meet the optical OFDM's throughput requirements.

2) Fixed-length: Due to the lack of implementation results of variable-length FFT architectures on FPGA, below in Table VI we present results for fixed length FFT architectures [5]–[7], [13]–[15], [25]–[27] with similar features to the ones of our proposed work. Xilinx devices used in these works have a MOF between 450 MHz and 650 MHz.

TABLE VI
COMPARISON OF DIFFERENT ARCHITECTURES FOR THE COMPUTATION OF FIXED LENGTH FFT.

N	Author	P	Architecture			Input Data width	Slices	DSP48s	BRAMs	Max. Freq. (MHz)	Lat_{hw} (cycl.)	Through-put (MS/s)	Efficiency ((MS/s) /slices)
			FFT Type	DC Type	R								
64	[26] [1] [6]	64	FP	-	4, 2	16	9,875	208	0	400	-	25,600	2.59
	[5] [1]	64	FP	-	4	10	10,264	208	0	125	-	8,000	0.78
	[14] [2]	32	PP	MDC	2^2	16	3,838	192	0	333	17	10,656	2.78
	Proposed [3]	16	PP	MDC	4	16	2,328	96	0	625.0	14	10,000	4.29
	[14] [2]	16	PP	MDC	2^2	16	2,235	96	0	334	19	5,344	2.39
	[13] [1]	16	PP	MDC	2^2	16	2,657	96	0	245	19	3,921	1.47
256	[6] [2]	256	FP	-	4	5	53,425*	0	0	312	19	79,872	1.49
	[7] [5]	256	FP	-	16	13	51,383*	0	0	182	19	46,592	0.91
	[25] [2]	256	Custom	-	4	6	48,335	0	0	248.3	-	15,891	0.33
	[14] [2]	64	PP	MDC	2^2	16	9,085	576	0	198	26	12,672	1.39
	[15] [4]	32	PP	SDF	$2^3, 2^5$	12	21,802*	0	0	219.5	-	7,024	0.32
	Proposed [3]	16	PP	MDC	4	16	3,394	144	0	606.1	51	9,698	2.85
	[26] [1] [6]	16	Custom	-	-	16	5,638	104	64	400	-	6,400	1.13
	[14] [2]	16	PP	MDC	2^2	16	3,105	144	0	297	38	4,752	1.53
	[13] [1]	16	PP	MDC	2^2	16	3,754	144	0	252	38	4,033	1.07
1024	[27] [3] [6]	64	Custom	-	-	12	25,795	720	334	250	-	16,000	0.62
	[14] [2]	64	PP	MDC	2^2	16	13,257	768	0	200	45	12,800	0.96
	[26] [1] [6]	32	Custom	-	-	16	15,512	288	128	400	-	12,800	0.82
	Proposed [3]	16	PP	MDC	4	16	5,007	192	0	555.6	110	8,889	1.77
	[13] [1]	16	PP	MDC	2^2	16	5,044	192	0	229	93	3,666	0.72
	[14] [2]	16	PP	MDC	2^2	16	4,186	192	0	221	93	3,536	0.84

[1] Virtex-5. [2] Virtex-6. [3] Virtex-7. [4] Kintex-7. [5] Virtex UltraScale. [6] Comercial product.
* Considering that one slice contains 4 LUTs and 4 FFs. FP: Fully Parallel. PP: Parallel Pipeline.

The most effective way to increase the throughput in these architectures is to use a higher number of FFT inputs, that is, to increase the value of P. To achieve high throughput is important to use a device with high MOF and to pay attention to the architectural design in order to minimize the critical path. In [5]–[7], array type FFT architecture (P=N) are used, because of that their area and throughput are both very high. However, their maximum clock frequency is much lower than the MOF of the used devices and their efficiencies are lower than the obtained by our work. In [25] an architecture with P=N is presented but it is not fully parallel because has an internal multiplexer and its HUE is not 100%. This modified architecture has no advantage with respect to [6], [7], since it uses almost the same resources and obtains 3 times lower output rate. In order to make fair comparisons, a measure of the efficiency, defined as the rate between the throughput and the number of slices, is included in Table VI.

In [13], [14] a radix-2^2 MDC architecture is used, with 16 data entries (P=16) of 16 bits and a Virtex-5 and Virtex-6 device, respectively. The results presented in the present work are, from the point of view of efficiency, significantly better (approximately 2.5 and 2.1 times) than in [13], [14] for the different lengths (N=64, 256, 1024). The hardware resources, number of slices and DSP48s, used in [13], [14] are similar to those used in the present work. The difference in efficiency is due to the fact that our FFT architecture achieves a clock frequency very close to the device's MOF.

For N=256, the present work compared with [26] has an efficiency 150% higher and in relation to the remaining works the improvement is even greater. With N=1024, there are two commercial products:

[26] and [27]. Both have the highest throughput due to their high degree of parallelism. The present work has an efficiency 185% greater than [27] and a 215% greater than [26].

In this work, we processed 16 inputs in parallel and the obtained latency is very similar to the one of other works that use the same parallelization degree. For example, for N=1024 and P=16 (Lat=64) our latency is 110 cycles and latency in [13], [14] is 93 cycles. For N=256 and P=16 (Lat=15) our latency is 51 cycles and latency in [13], [14] is 38 cycles. This slight increase in latency is due to the level of pipeline used in the CEs and DCs hardware implementation (right side of Eq. (5)), which allowed us to double the maximum operating frequency of our design.

V. Real-Time Experimental Setup

The real-time optical OFDM modem for low-cost intensity modulation direct detection (IM/DD) communication systems is shown in Fig. 9. The OFDM samples are generated offline using MATLAB and they are sent to a Xilinx FPGA evaluation board VC707 combined with an Euvis DAC MD657B (12 bits) operating at 5.0 GS/s to produce the required analog electrical signal. The FPGA sends to the DAC 4 data at the same time in double data rate (DDR) mode via 48 LVDS. The FPGA has a dedicated parallel-to-serial converter (OSERDES) which can serialize up to 8:1, we have selected 4:1 to relax the internal clock. Therefore to achieve a throughput of 5 GS/s we need to read 16 samples in parallel using a clock frequency of 312.5 MHz.

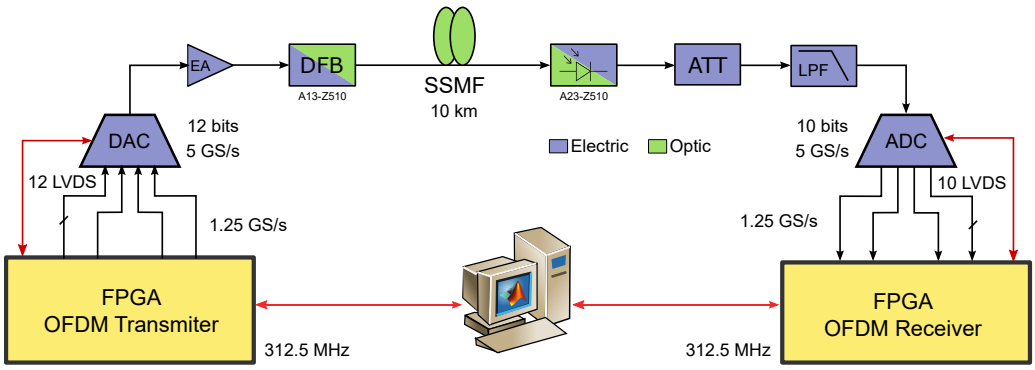


Fig. 9. Experimental setup for the Real-Time IM/DD OFDM-PON system.

First, the electrical OFDM signal is passed through an electrical amplifier (EA), with an operating frequency from 30 MHz to 3 GHz and an small signal gain of 27 dB, before optical conversion. A single-mode 1550 nm directly modulated linear optically isolated distributed feedback (DFB) laser is driven by the amplified electrical OFDM signal with a +3.6 dBm optical output power and without inline optical amplifier and is injected into 10 km standard single-mode optical fiber (SSMF) with optical attenuation of 0.18 dB/km. The received base-band OFDM signal is obtained via a high performance InGaAs photodiode (PD) with a bandwidth from 30 MHz to 3 GHz.

The converted electrical OFDM signal is attenuated (ATT), then passed through a low pass filter (LPF) with 3 dB bandwidth of 2,343 MHz and with rejection band (2.65 GHz to 7 GHz) attenuation of 40 dB and finally sampled at 5 GS/s by an ADC E2V EV10AQ190A (10 bits and 3.2 GHz analog bandwidth) combined with a Xilinx FPGA evaluation board VC707. The ADC sends to the FPGA 4 sampled data at the same time in DDR mode via 40 LVDS. The FPGA has a dedicated serial-to-parallel converter (ISERDES) which can create a 4-wide parallel word. Therefore to achieve a throughput of 5 GS/s we need to process 16 channels in parallel using a clock frequency of 312.5 MHz.

The receiver's RF power is manually adjusted to maintain an optimum peak-to-peak signal level at the ADC input. Finally, the captured samples are processed in real-time inside the FPGA and are sent to a PC via gigabit Ethernet to calculate in MATLAB the BER of the received bits. The following stages are applied to the base-band received signal and are implemented in the FPGA device: time symbol synchronization

[28], cyclic prefix removal, FFT, channel estimation and compensation (based on [29]) and high level QAM de-mapping. The implemented DSP functions ([1], [2]) included in the OFDM receiver are shown in Fig. 10.

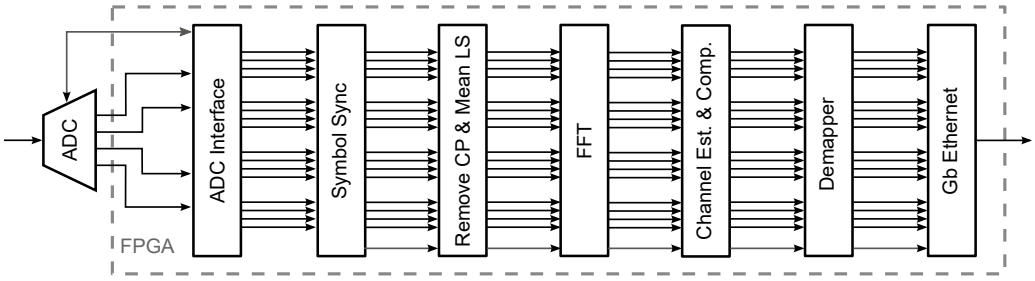


Fig. 10. FPGA OFDM Receiver.

There is about 15 dB power fading in the highest frequency subcarrier due to the excess low-pass filtering of the system (DAC, analog amplifier, fiber optic link, low-pass filter and ADC). To improve the system performance in IM/DD OFDM-PON systems, digital frequency-domain pre-emphasis is used.

In this experiment the number of OFDM subcarriers is set to $N = 1024$, but due to the Hermitian symmetry only 512 are defined: 475 transport data, 7 at the low, 5 at the middle and 24 at the high frequency are set to zero, and DC is used to bias the laser. This carrier arrangement results in a bandwidth of 2.349 GHz. The data-carrying subcarriers are modulated with different M-QAM ($M = 512, 256, 128, 64$) symbols. The cyclic prefix length is set to 16 samples. To reduce the high peak-to-average power ratio (PAPR) of the OFDM signal, digital clipping is applied at the OFDM transmitter output. We use a common clock source to avoid the sampling clock frequency offset between DAC and ADC, but their phases are not aligned.

A. Experimental results and discussions

The Table VII summarizes the performance of the real-time experimental setup for different QAM order symbols. The symbol rate and the number of data subcarriers are equal in all cases, then the raw bit rate depends on the modulation format.

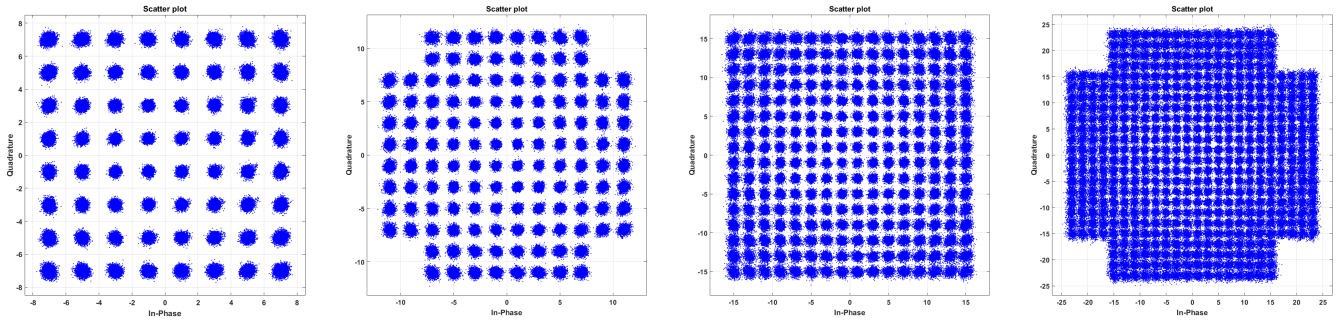
TABLE VII
PROPOSED SYSTEM PERFORMANCE FOR 64-, 128-, 256- AND 512-QAM MODULATION.

Modulation	Raw Bitrate	BER	EVM max	EVM min	SE (bit/s/Hz)
64-QAM	13.70 Gb/s	3.69×10^{-6}	8%	2%	5.833
128-QAM	15.99 Gb/s	1.48×10^{-5}	9%	2%	6.808
256-QAM	18.27 Gb/s	1.02×10^{-3}	12%	2%	7.779
512-QAM	20.55 Gb/s	6.97×10^{-3}	12%	2%	8.750

The magnitude of the error vector (EVM) is a measure of the quality of the modulation according to Eq. (6), where $|E_t|$ is the power of the normalized ideal constellation or the transmitted constellation, $E_{r,i}$ is the received signal vector, $E_{t,i}$ is the transmitted signal vector and I is the number of transmitted symbols [30]. The EVM allows us to evaluate the quality of a communications system measuring the dispersion of the received symbols. In an OOFDM system, the average EVM per subcarrier is measured and the quality of the modulation is determined for different frequencies. Table VII shows the best (min.) and worst (max.) values of EVM, obtained for the different QAM modulation orders.

$$\text{EVM} = \frac{\sigma_{err}}{|E_t|},\ \sigma_{err}^2 = \frac{1}{I}\sum_{i=1}^{I}|E_{err,i}|^2,\ E_{err,i} = E_{r,i} - E_{t,i} \tag{6}$$

After 10 km SSMF transmission, the BER was measured. We have taken a BER value of 3.8×10^{-3} as a reference because it is commonly considered a FEC threshold to obtain an error free transmission when a HD-FEC with 7% redundancy is employed [31]; on the other hand, in case a SD-FEC with 20% redundancy were employed, the FEC threshold would be 2.4×10^{-2} [32]. Measurements show that a BER better than the HD-FEC threshold is successfully achieved for 64-, 128- and 256-QAM symbols. For 512-QAM a BER better than the SD-FEC threshold is achieved.



(a) 64-QAM (b) 128-QAM (c) 256-QAM (d) 512-QAM

Fig. 11. 64-, 128-, 256- and 512-QAM constellations after 10 km SSMF transmission.

Taking into account the redundancy used in SD-FEC and HD-FEC, the error-free bit rate for 512-QAM is 16.44 Gb/s and for 256-QAM is 16.99 Gb/s. This slight difference in the bitrate and the reduction in hardware cost when a HD-FEC is employed, make the 256-QAM modulation the best option for this transmission system. The 64-, 128-, 256- and 512-QAM constellations after 10 km SSMF transmission are shown in Fig. 11.

The curve of BER versus the received optical power (ROP) for a 64-QAM modulation is shown in Fig. 12. To elaborate this curve, it was necessary to modify the experimental setup of Fig. 9. At the output of the optical fiber, an optical variable attenuator (OVA) and an optical coupler 99-1 were connected. At the output of the photodetector, a chain of electrical attenuators and amplifiers were connected to normalize the amplitude of the electrical signal to the values allowed by the ADC. When the ROP is beyond about -7.5 dBm, the BER is less than HD-FEC limit (7% overhead). When the ROP is beyond about -9.5 dBm, the BER is less than SD-FEC limit (20% overhead).

VI. Conclusions

In this paper, the implementation of a variable-length parallel-pipelined FFT architecture suitable for high-speed optical OFDM communication systems was presented. The architecture can be configured in run-time to compute FFTs of lengths between 16, 64, 256 and 1024 points, and process 16 data in parallel. This will allow us to experiment with the effect of the FFT length on the performance of the OOFDM system. Our results show that it can operate at 10 GS/s when implemented in current FPGA devices. An OFDM receiver was implemented using this FFT architecture and a transmission rate of 20 Gb/s is reached transmitting data through 10 km of single-mode optical fiber.

Acknowledgements

This work was supported by the Spanish *Ministerio de Economía y Competitividad* under project TEC2015-70858-C2-2-R with FEDER funds.

References

[1] J. Armstrong, "OFDM for Optical Communications," *Lightwave Technology, Journal of*, vol. 27, no. 3, pp. 189–204, February 2009.

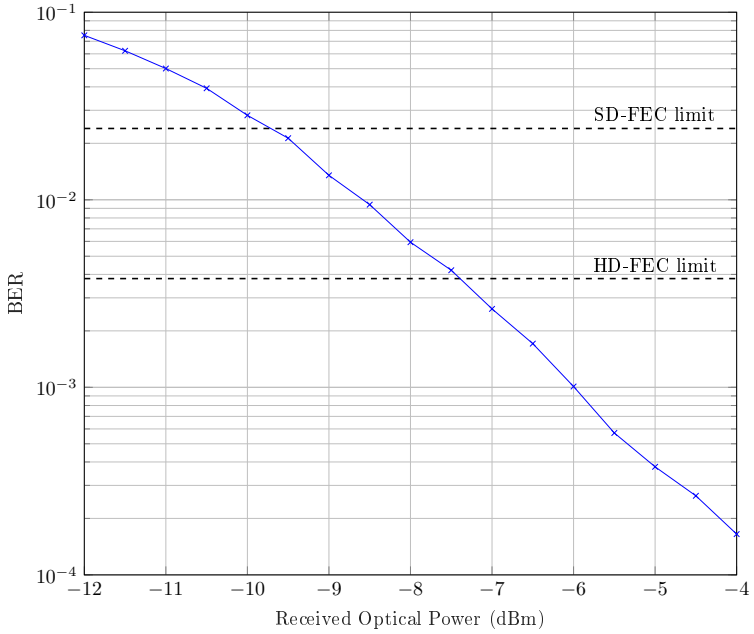


Fig. 12. The BER performance of the OOFDM system vs. ROP for a 64-QAM modulation.

[2] F. Buchali, R. Dischler, and X. Liu, "Optical OFDM: A promising high-speed optical transport technology," *Bell Labs Technical Journal*, vol. 14, no. 1, pp. 125–146, Spring 2009.

[3] G. P. Agrawal, *Fiber-Optic Communication Systems*, 4th ed. Wiley, 2010.

[4] G. Bergland, "Fast Fourier transform hardware implementations - An overview," *Audio and Electroacoustics, IEEE Transactions on*, vol. 17, no. 2, pp. 104–108, 1969.

[5] M. Jamali, J. Downey, N. Wilikins, C. Rehm, and J. Tipping, "Development of a FPGA-based high speed FFT processor for wideband Direction of Arrival applications," in *Radar Conference, 2009 IEEE*, 2009, pp. 1–4.

[6] G. Polat, S. Ozturk, and M. Yakut, "Design and Implementation of 256-Point Radix-4 100 Gbit/s FFT Algorithm into FPGA for High-Speed Applications," *ETRI Journal*, vol. 37, no. 4, pp. 667–676, Aug 2015.

[7] J. Kim, J. Lee, and K. Cho, "Design of 256-point FFT processor for 100 Gb/s coherent optical OFDM system," in *2016 IEEE International Symposium on Consumer Electronics (ISCE)*, Sept 2016, pp. 61–62.

[8] M. Garrido, M. . Sánchez, M. L. López-Vallejo, and J. Grajal, "A 4096-Point Radix-4 Memory-Based FFT Using DSP Slices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 1, pp. 375–379, Jan 2017.

[9] Z. G. Ma, X. B. Yin, and F. Yu, "A novel memory-based fft architecture for real-valued signals based on a radix-2 decimation-in-frequency algorithm," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 9, pp. 876–880, Sept 2015.

[10] H. L. Groginsky and G. Works, "A Pipeline Fast Fourier Transform," *Computers, IEEE Transactions on*, vol. C-19, no. 11, pp. 1015–1019, 1970.

[11] J. A. Johnston, "Parallel pipeline fast Fourier transformer," *Communications, Radar and Signal Processing, IEE Proceedings F*, vol. 130, no. 6, pp. 564–572, 1983.

[12] K.-J. Yang, S.-H. Tsai, and G. Chuang, "MDC FFT/IFFT Processor With Variable Length for MIMO-OFDM Systems," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 4, pp. 720–731, 2013.

[13] M. Garrido, J. Grajal, M. Sanchez, and O. Gustafsson, "Pipelined Radix- 2^k Feedforward FFT Architectures," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 1, pp. 23–32, 2013.

[14] M. Garrido, M. Acevedo, A. Ehliar, and O. Gustafsson, "Challenging the limits of FFT performance on FPGAs (Invited paper)," in *2014 International Symposium on Integrated Circuits (ISIC)*, Dec 2014, pp. 172–175.

[15] H. S. Kang, S. H. Chang, I. K. Hwang, and J. K. Lee, "A design and implementation of 32-paths parallel 256-point FFT/IFFT for optical OFDM systems," in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, Jan 2016, pp. 1–1.

[16] G. R. Locharla, K. K. Mahapatra, and S. Ari, "Variable length mixed radix MDC FFT/IFFT processor for MIMO-OFDM application," *IET Computers Digital Techniques*, vol. 12, no. 1, pp. 9–19, 2018.

[17] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Custom Integrated Circuits Conference, 1998. Proceedings of the IEEE 1998*, 1998, pp. 131–134.

[18] J. G. Proakis and D. G. Manolakis, *Digital signal processing: principles, algorithms, and applications*, 4th ed. Prentice Hall, 2006.

[19] B. Gold and T. Bially, "Parallelism in fast Fourier transform hardware," *Audio and Electroacoustics, IEEE Transactions on*, vol. 21, no. 1, pp. 5–16, 1973.

[20] E. Swartzlander, W. Young, and S. Joseph, "A radix 4 delay commutator for fast Fourier transform processor implementation," *Solid-State Circuits, IEEE Journal of*, vol. 19, no. 5, pp. 702–709, 1984.

[21] P. Papamichalis and C. Burrus, "Conversion of digit-reversed to bit-reversed order in FFT algorithms," in *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, 1989, pp. 984–987 vol.2.

[22] C. Eddington and B. Ray, "Using the Parallel FFT for Multigigahertz FPGA Signal Processing," *Xcell Journal*, no. 82, pp. 50–55, 2013. [Online]. Available: {https://issuu.com/xcelljournal/docs/xcell_journal_issue_82/50}

[23] P. P. Boopal, M. Garrido, and O. Gustafsson, "A reconfigurable FFT architecture for variable-length and multi-streaming OFDM standards," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, May 2013, pp. 2066–2070.

[24] J. Yang, D. Zhang, Y. Gong, and B. Liu, "A Novel Design of Pipeline MDC-FFT Processor Based on Various Memory Access Mechanism," in *2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, Oct 2016, pp. 62–65.

[25] M. Dreschmann, J. Meyer, M. Hubner, R. Schmogrow, D. Hillerkuss, J. Becker, J. Leuthold, and W. Freude, "Implementation of an ultra-high speed 256-point FFT for Xilinx Virtex-6 devices," in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, 2011, pp. 829–834.

[26] E. Dillon. Dual Parallel FFT. [Online]. Available: {http://www.dilloneng.com/fft_ip}

[27] L. RFEL. HyperSpeed Plus FFT. [Online]. Available: http://www.rfel.com/index.php/signal-processing/ip-cores/fast-fourier-transforms/hyperspeed-plus-fft/

[28] J. S. Bruno, V. Almenar, J. Valls, and J. L. Corral, "Low-Complexity Time Synchronization Algorithm for Optical OFDM PON System Using a Directly Modulated DFB Laser," *J. Opt. Commun. Netw.*, vol. 7, no. 11, pp. 1025–1033, Nov 2015.

[29] M. J. Canet, J. Valls, V. Almenar, and J. Marín-Roig, "FPGA implementation of an OFDM-based WLAN receiver," *Microprocessors and Microsystems*, vol. 36, no. 3, pp. 232 – 244, 2012.

[30] R. Schmogrow, B. Nebendahl, M. Winter, A. Josten, D. Hillerkuss, S. Koenig, J. Meyer, M. Dreschmann, M. Huebner, C. Koos, J. Becker, W. Freude, and J. Leuthold, "Error Vector Magnitude as a Performance Measure for Advanced Modulation Formats," *IEEE Photonics Technology Letters*, vol. 24, no. 1, pp. 61–63, Jan 2012.

[31] International Telecommunication Union, "Forward error correction for high bit-rate DWDM submarine systems," Feb. 2004, ITU-T Recommendation G.975.1.

[32] L. Nelson, G. Zhang, M. Birk, C. Skolnick, R. Isaac, Y. Pan, C. Rasmussen, G. Pendock, and B. Mikkelsen, "A robust real-time 100G transceiver With soft-decision forward error correction [Invited]," *Optical Communications and Networking, IEEE/OSA Journal of*, vol. 4, no. 11, pp. B131–B141, Nov 2012.