Analog Integr Circ Sig Process (2012) 70:241–248
DOI 10.1007/s10470-011-9787-2

FPGA implementation and testing of a 128 FFT for a MB-OFDM receiver

Bruno Fernandes · Helena Sarmento

Received: 30 April 2010 / Revised: 5 September 2011 / Accepted: 14 September 2011 / Published online: 28 September 2011

Abstract In this paper we discuss the FPGA design of a 128-point Pipelined FFT processor for a multi-band (MB)-OFDM receiver. We implemented two parallel architectures based on the Xilinx Core Generator. The architectures use two or four smaller FFTs in parallel. Results show that time requirements are fulfill when combining Pipelined, Streaming input/output (I/O) architectures with Radix-2 and Radix-4 operations. For Virtex-4, we need four levels of parallelization, but for Virtex-5 only two. Both circuits were synthesized, placed and routed. We tested the circuits on the FPGA, defining test vectors and analyzing outputs signals with Chipscope.

Keywords UWB · MB-OFDM · FFT · FPGA

1 Introduction

Ultra-wideband (UWB) has been proposed as a technique to implement wireless personal area networks (WPAN) for short-range (<10 m) wireless technology that permits data transfers at very high rates, between 53.3 and 480 Mbps, with low power consumption (0.5 mW). It offers a practical solution to interconnect, by wireless connections, devices such as high definition television (HDTV), digital video recorders, digital camcorders and DVD systems.

B. Fernandes (✉)
INESC-ID, Lisbon, Portugal
e-mail: bruno.fernandes@xfel.eu

H. Sarmento
INESC-ID/IST/TU, Lisbon, Portugal
e-mail: helena.sarmento@inesc.pt

The Federal Communications Commission (FCC) in the USA defined a UWB signal as a signal which the bandwidth exceeds 500 MHz or is 20% of the center frequency. The FCC has also allocated 7.5 GHz, from 3.1 to 10.6 GHz, to unlicensed use, i.e. without licensing costs or control, as shown on Fig. 1.

MultiBand OFDM (MB-OFDM) [1] is a UWB technology. It divides the spectrum allocated to UWB into 14 bands of 528 MHz (Fig. 2). The fundamental principle behind OFDM is to split a data symbol into a set of independent smaller symbols. Each one of these sub-symbols are modulated on a separate sub-carrier (FDM). The orthogonality between sub-carriers permits to overlap the sub-carriers spectrum without mutual interference. All sub-symbols are transmitted simultaneously in time domain.

MB-OFDM supports data rates of 53.3, 80, 106.7, 160, 200, 320, 400 and 480 Mb/s. For data rates of 200 Mb/s and lower, the standard [1] establishes that the binary data is mapped onto a QPSK (quadrature phase-shift keying) constellation. When the data rate is 320 Mb/s or higher, the mapping is done onto a multi-dimensional constellation using a DCM (dual-carrier modulation) technique. In QPSK, the binary serial input is divided into groups of two bits. Each group converted into a complex number, representing one of the four QPSK constellation points. In DCM the input is divided into groups of 4 bits. Each group is converted in two complex numbers, representing two points in two different 16 QAM constellations.

The timing parameters associated with the physical layer of MB-OFDM are listed in Table 1. A total of 100 data sub-carriers and 10 guard sub-carriers are used per symbol. In addition, 12 pilot sub-carriers allow for coherent detection.

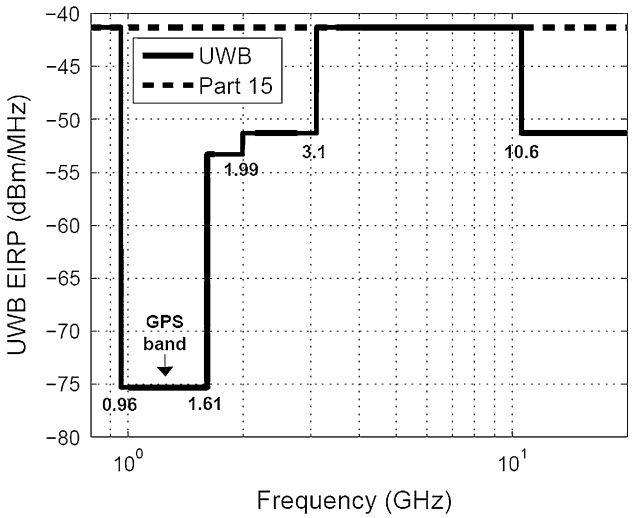

Fig. 1 FCC spectrum mask for UWB indoor transmission

$$s(t) = \begin{cases} \sum_{n=0}^{N_{FFT}-1} C_n e^{j2\pi\Delta ft} & t \in [0,\ T_{FFT}] \\ 0 & t \in [T_{FFT} + T_{SYM}] \end{cases} \quad (1)$$

Each band (Fig. 2) transmits an ODFM symbol. The OFDM symbol is created by an IFFT, being mathematically described by Eq. 1. In Eq. 1, C_n coefficients are the complex numbers placed on data, pilots or guard sub-carriers of the nth symbol, N_{FFT} is the FFT size, T_{FFT} the IFFT period, T_{SYM} the symbol interval and Δf the sub-carrier spacing (Table 1).

At the receiver the symbol is demodulated by an FFT calculation. The FFT processor, besides the Viterbi decoder, has the higher computational complexity at the receiver (Fig. 3). The time to process an OFDM symbol (IFFT and FFT period) is 242.42 ns (1/4.125 MHz).

The main objective of our work is to prototype the MB-OFDM receiver on a FPGA. FPGAs, due to its flexibility, can be used to test and analyze implementation issues. Moreover, high-end FPGA include DSP blocks, high-speed serial input/output (I/O) and memory structures to support high data throughput that are particularly interesting to prototype digital processing in transceivers. Prototyping of wireless data communication systems, using high end FPGAs, has been reported in literature for mobile

Table 1 Timing-related parameters

Sampling frequency	528 MHz
Total number of sub-carriers (FFT size)	128
Number of data sub-carriers	100
Number of pilot sub-carriers	12
Number of guard sub-carriers	10
Total of sub-carriers used	122
Sub-carrier frequency spacing	4.125 MHz
IFFT and FFT period	242.42 ns

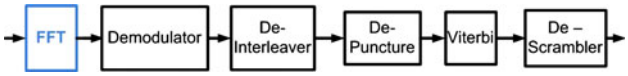

Fig. 3 The UWB receiver

technologies [2] and for short range high data rate (MB-OFDM) [3, 4].

The FFT (Fig. 3) is a crucial block in multi-carrier systems like OFDM, being responsible by the demodulation of the OFDM symbol. Due to the required performance and resources involved, the FFT implementation is a key factor in the receiver design.

Since the Cooley–Tukey algorithm [5] was proposed, many architectures appeared for FFT hardware implementation. One of the key issues is the way twiddle factors are handled. They can be stored in memories, optimizing the FFT performance, or calculated on the fly, reducing memory storage. Twiddle factors calculation is usually performed by a CORDIC algorithm. A survey on CORDIC algorithms for FPGA implementation is described in [6]. An FFT implementation in a FPGA with both CORDIC and stored twiddle factors, using Radix-2 algorithm, is propose in [7].

In general, FFT algorithms can be categorized in two groups. When FFT is to be performed on N points where $N = 2^m$ (m being a natural number), the design uses Radix-2, Radix-4, split-radix or, more recently, Radix-2^2 and Radix-2^4 algorithms. However, when N is not equal to 2^m, a PFA (prime factor algorithm) [8] or WFTA (Winograd Fourier transform algorithm) [9] are preferable. According

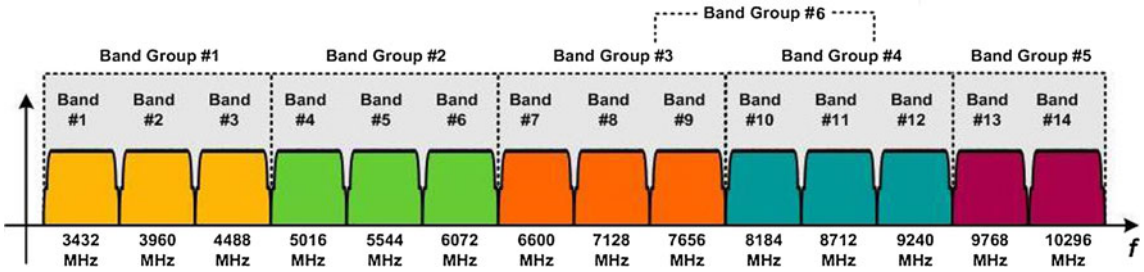

Fig. 2 MB-OFDM band group allocation

to [10], for hardware implementation, when N is a power of 2, is preferable to adopt Radix-4 and split-radix, while PFA is preferable for lengths having coprime factors. Architectures implementing the Radix-2^2 and Radix-2^4 have shown to reduce the hardware resources when compared with the other architectures [11, 12].

Several works present FFT implementation for MB-OBFDM in FPGAs [4, 13, 14, 15]. In [4], the FFT is implemented with the Pipeline, Streaming I/O from Xilinx Core Generator. Four parallel 128-point FFTs were implemented, operating at a maximum frequency of 136 MHz. Since the architecture allows four frames to be process in parallel, a clock frequency of 132 MHz (528/4 MHz) is needed. The outputs of each FFT are combined, using the unfolding transformation and multiple-phase clocking. Unfolding is a technique that can be applied to a DSP program to create a new program that describes more than one iteration of the original program [16].

A complete FFT was developed from scratch in [13], using both Radix-4 and Radix-2 algorithms, with eight parallel Radix-4 butterflies. More recently, Nuo Li and Meijs [14] propose a FFT design, also made from scratch, which uses the Radix 2^2 parallel algorithm. The Radix 2^4 algorithm has also been implemented in a 128 FFT architecture for MB-OFDM systems [15].

In our work we decided to use FFTs Cores, generated by Xilinx Core Generator [17]. The Pipelined, Streaming I/O offers the highest throughput [17], making it the more appropriate for high rate wireless applications. It allows for continuous frame transformations, at the cost of more hardware resources, being suitable for our application where input data arrive sequentially. By itself, the Pipelined, Streaming I/O can not process a symbol in 242.42 ns when Virtex-4 and Virtex-5 devices are used. Therefore, we parallelize the FFT calculation to decrease the clock frequency at the cost of increasing the amount of used FPGA resources.

Different levels of parallelization were analyzed in order to obtain a processing time less than 242.42 ns. We started by parallelizing the calculation, even and odd indexed data computed separately. Final values are computed with a Radix-2 butterfly. That architecture only fulfilled the requirements on a Virtex-5 xc5 versus x50t with speed rate of -1. A second architecture, with four 32 points FFT in parallel, achieved the 242.42 ns on a Virtex-4 xc4sx35 with a speed rate of -10 [18].

For the time being, besides the FFT, we also developed the following blocks: QPSK demodulator, DCM demodulator [19], Interleaver [20] and Viterbi decoder [21]. The extraction of the 100 data sub-carriers from the OFDM symbol is preform by a module, FFT_to_Dem, which is included in our FFT block (Fig. 3). The data is then send to the QPSK or DCM demodulators.

This paper is organized as follows. Section 2 describes the capabilities of the FFT IP Core from Xilinx used to implement the OFDM demodulator. In Sect. 3, the FFT implementation details are discussed. Results are presented in Sect. 4 Finally, conclusions are presented in Sect. 5.

2 Xilinx FFT (Core) Generator

In a discrete Fourier transform (DFT), the $X(k)$ sequence $(k = 0, \ldots, N-1)$ is obtained from the $x[n]$ sequence where $n = 0, \ldots, N-1$, according to Eq. 2,

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk} \tag{2}$$

where W_N^{nk}, referred to as *twiddle factor*, is defined by Eq. 3.

$$W_N^{nk} = e^{-jnk\frac{-2\pi}{N}} \tag{3}$$

The FFT is an efficient algorithm for computing the DFT. It is based on the fundamental principle of decomposing the DFT of a N length sequence into successively smaller DFTs. Algorithms that decompose the $x[n]$ sequence into successively smaller sub-sequences are called decimation-in-time algorithms, while decimation-in-frequency algorithms decompose the output $X(k)$ [5]. Using Radix-2, the $x[n]$ sequence is separated in two sequences, one for the $n/2$ even indexes and other for the odd. Equation 2 can be rewritten as Eq. 4, taking advantage of the properties of the twiddle factors.

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x[2r] W_{N/2}^{2rk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x[2r+1] W_{N/2}^{2rk} \tag{4}$$

The computation of the original DFT (Eq. 2) is now reduced to the computation of two DFTs with half the complexity. This procedure can be recursively applied until data is reduced to transforms of only two points. The Radix-4 algorithm rearranges $x[k]$ in four sequences, the elementary computations being done by 4-point DFTs.

2.1 Xilinx FFT IP Core

Xilinx FFT IP Core [17] follows the Cooley–Tukey algorithm [5] to calculate the FFT. The FFT Core can compute any N-point forward or inverse DFT (IDFT) as long as $N = 2^m$, with $m = 3, \ldots, 16$. Xilinx FFT IP Core implements Radix-4 and Radix-2 decomposition.

The Core provides four different optional architectures: Pipeline, Streaming I/O; Radix-4 Burst I/O; Radix-2 Burst I/O and the Radix-2 Lite Burst I/O. These different options present a trade-off between core size and transform time.

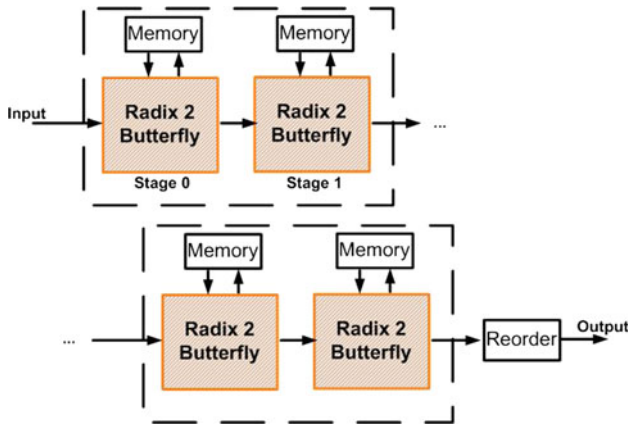

Fig. 4 Pipelined, Streaming I/O

All architectures use memories (block or distributed RAM) to store twiddle factors. Fractional bits are truncated or rounded after the multiplication on the butterflies and the data path can be increased to retain all integer bits.

The Pipelined, Streaming I/O architecture pipelines several Radix-2 butterflies to allow continuous data processing (Fig. 4). This architecture is the one that requires more hardware, but offers the best throughput. Each butterfly has its own memory banks to store input and intermediate data, a implementation usually refereed as R2SDF (Radix-2 single-path delay feedback), optimizing memory usage [22].

R2SDF has the ability to simultaneously perform transform calculations on the current frame of data, load input data for the next data frame, and unload the results from the previous frame of data. It is also possible to compute frames with gaps in between. The Pipelined, Streaming I/O architecture uses decimation-in-frequency algorithm.

3 FFT implementation

We parallelized the FFT calculation, decomposing the 128-point input into smaller sequences. The FFT Core Generator was used to generate these smaller FFTs. FFTs output are then combined to obtain the final values. Arithmetic functions needed to obtain the final output values are implemented using the Complex Multiplier and Adder/Subtracter functions from Xilinx Core Generator. We decided to use FFTs Cores with Pipeline, Streaming I/O architecture since it offers the best throughput and also because we want to maintain continuous data processing, making the architecture suitable for wireless applications where input data arrive sequentially.

Our implementation adopts full-precision unscaled arithmetic, with an 8 bit input and output based on [23]. In [23], requirements for the numerical precision, in a practical implementation of MB-OFDM, are discussed. The performance of a MB-OFDM receiver and transmitter, with different floating point representations and different channel models is compared. The work concludes that 8 bits for the FFT input and output data and 11 bits for the internal data are good data representations.

3.1 FFTs architecture

Two architectures were implemented, using two and four smaller FFTs in parallel. The input stream is written into memory blocks and then read according to the input format of the FFT architectures.

Firstly, we implement two 64-point FFTs that we refer as the 2FFTpipe architecture. Even and odd-indexed data are computed separately. A Radix-2 block computes the final values.

The signal to start the FFT calculation, as presented on Fig. 5, is the same to start both 64-point FFTs. Delay exists at the inputs, according to the specifications of Xilinx FFT Core. Real and imaginary twiddle factors are stored in block RAMs. The *xk* index signal, from one of the smaller FFTs, is used to address twiddle factor values needed in the complex multiplication. *DV* and *xk* index signals are delayed to be synchronous with the final outputs values (Fig. 5). Likewise, the FFT even output values are delayed due to the latency of the complex multiplier.

To reduce memory size, only 32 of the 64 twiddle factor are stored. All the 64 twiddle factors needed can be obtain

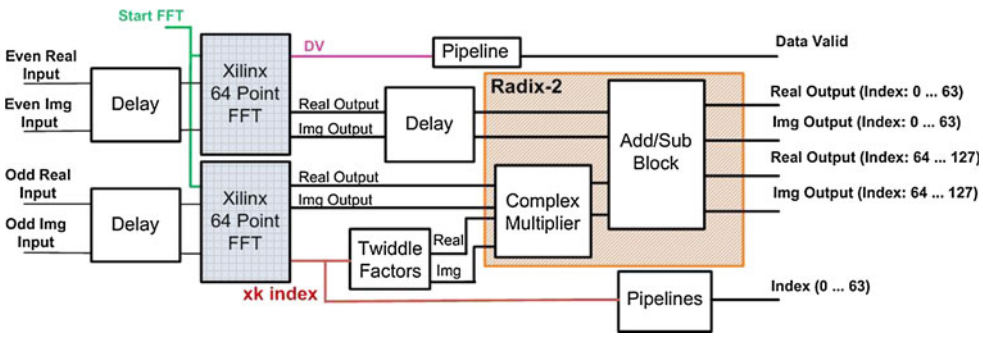

Fig. 5 2FFTpipe architecture

Table 2 Twiddle factors

xk index	Twiddle factor values	
	Real	Img.
0 … 31	RAM Real	RAM Img.
31 … 63	RAM Img.	−(RAM Real)

with only half of the values due to Eq. 5, as stated on Table 2.

$$cos\left(\phi + \frac{\pi}{2}\right) = -sin(\phi) \tag{5}$$

A clock of 264 MHz (528/2 MHz) was required for the 2FFTPipe. In a Virtex-5 device this requirement is fulfilled, but not on Virtex-4.

To improve performance, a second architecture with four FFT in parallel (Fig. 6) was implemented. The 128-point input is decomposed in four 32-point sequences. A Radix-4 is used to combine the output of the 4 FFTs. The twiddle factors are stored in three separate block RAMs. This architecture is able to work at 132 MHz, fulfilling the time requirements for a MB-OFDM receiver in both Virtex-4 and Virtex-5 devices.

4 Results

Both architectures were synthesized, implemented and floor planned. Test bench waveforms were created for each implementation and used on behavioral and post layout simulations. The generated bit streams were downloaded to the FPGA and Chipscope was used to validate the designs on the hardware. The same input data was used on all simulations and on both architectures to compare deviations in the output values.

Simulations preformed on Xilinx ISE 10.1 proved the correct behavior of both architectures. Frames were calculated continuously and with gaps between them. FFT output values obtained with the MATLAB9 FFT function, the 2FFTPipe architecture and the 4FFTPipe architecture, as presented on Fig. 7, are very similar. To improve the figure clearness, we present only half the values. The other half is analogous.

The output values are 8 bits wide with 3 bits representing the fractional part. Although similar, values obtained from the 2FFTPipe architecture present bigger differences to the MATLAB values, particularly in the 59th and 64th sub-carriers values.

Since the output from Xilinx FFT Core is truncated, deviations on the output values are expected. By parallelizing the FFT computation, the associated error is distributed along the different sequences, therefore diminishing it.

At the demodulation process, the output values are analyzed to find the QAM constellation point they are closed to. Comparing results obtain by MATLAB, we concluded that the 2FFTpipe output errors do not affect the demodulation process.

Chipscope Pro 10.1 was used to validate the designs implemented on the FPGA. A signal from a Chipscope VIO (virtual input/output) module is used to start the FFT calculation and to read RAMs, which contain the data input. We used the internal frequency of the crystal oscillator and increased it with digital clock manager (DCM) blocks to obtain a frequency higher than 264 MHz for the 2FFTPipe and higher than 132 MHz to the 4FFTPipe.

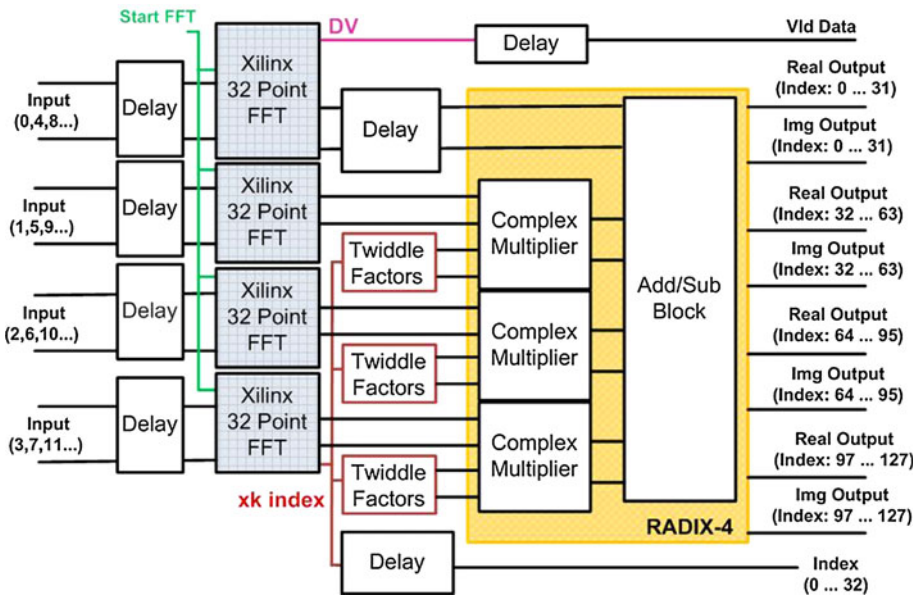

Fig. 6 4FFTpipe architecture

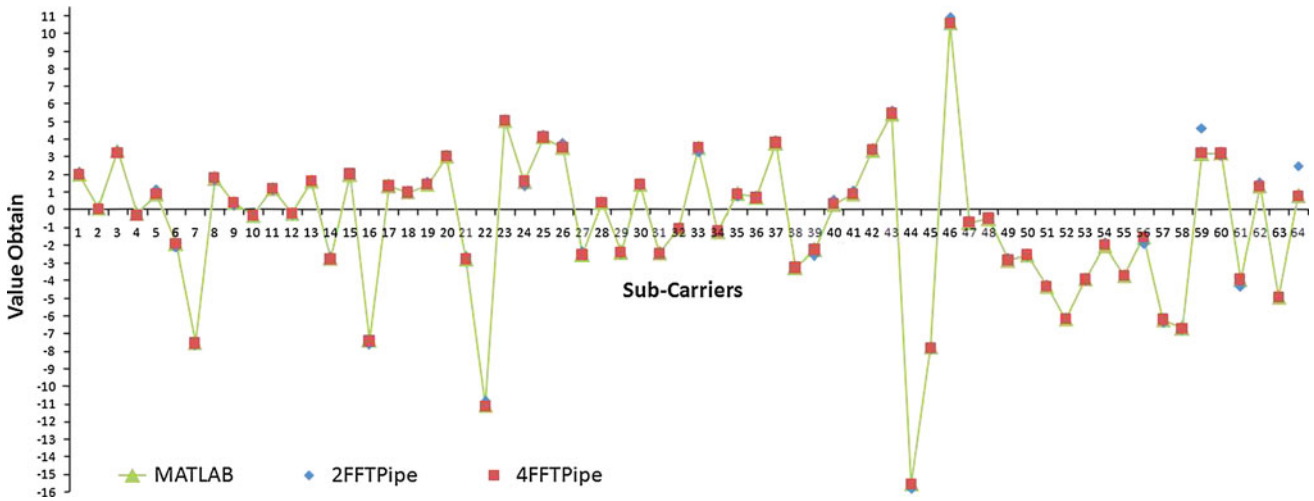

Fig. 7 FFTs output values comparison

Fig. 8 Results obtain from simulation (**a**) and Chipscope (**b**)

Figure 8(a) depicts output values from a behavioral simulation and Fig. 8(b) values from Chipscope, both for the 2FFTpipe architecture. Output values are interpreted as unsigned values. Both values are similar. Indice and vld_fft signals, respectively the Index and Data Valid on Fig. 5, are synchronous with the output data in both scenarios.

Table 3 presents the maximum clock frequency and resources used, including the FFT_to_Dem block. The maximum clock frequency achievable is reported in the Place and Route report. Synthesis report indicates that the frequency is limited by the FFT arithmetic operations.

As mentioned in 3.1, to fulfill time requirements, for the 4FFTPipe architecture, a frequency of 132 MHz is required. The results indicate that the 4FFTpipe can implement, on a Virtex-5 or a Virtex-4 FPGA, the MB-OFDM demodulator receiver. The 2FFTpipe architecture, that needs a clock frequency of 264 MHz, is only suitable for Virtex-5 device.

Table 3 Post layout results

	2FFTPipe		4FFTpipe	
	Virtex-4	Virtex-5	Virtex-4	Virtex-5
Freq. (MHz)	255.82	297.353	223.86	245.10
Flip-flops	2.758	2.640	3.923	3.923
LUTs	2.739	2.169	3.693	3.468
Occupied slices	2.074	1.122	2.739	1.634
FIFO/RAM	5	3	10	5
DSP48s	43	43	81	81

Table 4 Synthesis results comparison

	[13]	[14]	2FFTpipe	4FFTpipe
Freq. (MHz)	76.67	167.30	255.82	223.86
Output bits	11	15	8	8
Slices	7,390	1,052	1,667	2,579
4 Input LUTS	12,749	3,600	2,099	3,304
DSP48s	48	20	43	81

The difference on slice and LUTs distribution (Table 3) is due to the devices architectures. For instance, Virtex-5 uses 36 kbits memory blocks, while Virtex-4 uses 18 kbits. Therefore, as expected, memory usage in a Virtex-5 is reduced by half. Also, Virtex-5 LUTs have six inputs signals which reduces the number of LUTs used when implementing some functions.

Table 4 compares synthesis results obtained from the 2FFTpipe and 4FFTpipe architectures, and values reported by [13] and [14]. Results in [13] and [14] are for a Virtex-4 device. For the 2FFTpipe and 4FFTpipe architectures, the FFT_to_Dem block are not included. Although the 2FFTpipe architecture is not viable for an Virtex-4 FPGA device, results are presented for comparison purposes.

Comparing logic resources with our previous implementation [13], we greatly improved the design. In fact, all the DSP blocks and extra logic blocks were used to implement the arithmetic operation in that architecture. Now, the arithmetic operations are fully implemented in DSP blocks.

The Radix-2^2 algorithm requires fewer arithmetic operations. Results presented in [14] show that the number of DSP blocks is significantly lower, but at the expense of a development from scratch. We presented a suitable FFT implementation for MB-OFDM hardware based on a commercial IP core.

5 Conclusion

In this paper we present two FFTs architectures suitable for UWB MB-OFDM receivers. They implement an 128-point FFT based on the Xilinx Core Generator FFT. The architectures use already develop and tested IP cores, fulfilling the numerical precision for a practical implementation and being able to maintain continuous data processing.

Results show that Xilinx FFT Pipelined/Streaming I/O architectures, combined with Radix-2 and Radix-4 operations permits to meet MB-OFDM requirements for a receiver. The two parallel architectures were implemented and validated on a Virtex-5 FGPA. Results were analyzed with Chipscope. Both architectures will be used on the current development of a MB-OFDM receiver.

Acknowledgments This work has been performed under the project "UWB Receiver: baseband processing using reconfigurable hardware"—PTDC/EEAELC/67993/2006, and was partially supported by FCT (INESC-ID multi-annual funding) through the PIDDAC Program funds."

References

1. ECMA International (2008). High rate ultra wideband PHY and MAC standard. Standard ECMA-368, Dec. 2008, ECMA International.
2. Cui, Q., Li, Y., & Tao, X. (2010). Gbps wireless communication system design and transmitter implementation. *IEEE International Conference on Broadband Network and Multimedia Technology*, Beijing, China.
3. Santhi, M., Lakshminarayanan, G., Sundaram, R., & Balachander, N. (2009). Synchronous pipelined two-stage radix-4 200 Mbps MB-OFDM UWB Viterbi decoder on FPGA. *SoC Design Conference* (*ISOCC*), Busan, Korea.
4. Simon Sherratt, R., Cadenas, O., & Goswami, N. (2005). A low clock frequency FFT core implementation for multiband full-rate ultra-wideband (UWB) receivers. *IEEE Transactions on Consumer Electronics, 51*(3).
5. Cooley, J. W., & Tukey, J. W. (1965). An algorithm for machine computation of complex Fourier series. *Mathematics of Computation, 19*(90).
6. Andraka, R. (1998). A survey of CORDIC algorithms for FPGA based computers. *Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays*, California, USA.
7. Sllab, A., Fahmy, H., & Rashwan, M. (2009). Optimized hardware implementation of FFT processor. *4th International Design and Test Workshop* (*IDT*), Riyadh, Saudi Arabia.
8. Kolba, D., & Parks, T. (1977). A prime factor FFT algorithm using high-speed convolution. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-25, 281–294, Aug.
9. Winograd, S. (1976). On computing the discrete Fourier transform. *Proceedings of the National Academy of Sciences of USA, 73*(4).
10. Duhamel, P., & Vetterli, M. (1997). *The digital signal processing handbook*. Boca Raton, FL: CRC Press and IEEE Press.
11. Shousheng, H., & Torkelson, M. (1996). A new approach to pipeline FFT processor. *Parallel Processing Symposium*, Lund, Sweden.
12. Santhi, M., Arun Kumar, S., Praveen Kalish, G. S., Murali, K., Siddharth, S., & Lakshminarayanan, G. (2008). A modified radix-2^4 SDF pipelined OFDM module for FPGA based MB-OFDM UWB systems. *International Conference on Computing, Communication and Networking*, *ICCCn*, Dec. 2008, Tamil Nadu, India.
13. Rodrigues, N., Neto, H., & Sarmento, H. (2007). A OFDM module for a MB-OFDM receiver. *International Conference on Design and Technology of Integrated Systems in Nanoscale Era* (*DTIS*), Rabat, Morocco.
14. Li, N., & van der Meijs, N. (2008). A radix 2^2 based parallel pipeline FFT processor for MB-OFDM UWB system. *International Conference on System on Chips* (*SOC*), Tampere, Finland.
15. Cho, S.-I., Kang, K.-M., & Choi, S.-S. (2008). Implementation of 128-point fast fourier transform processor for UWB systems. *Wireless Communications and Mobile Computing Conference. IWCMC International*, Crete Island, Greece.
16. Parhi, K. (1999). *VLSI digital signal processing*: *Design and implementation*. Chichester: Wiley.
17. Xilinx Inc. (2008). Logic core fast Fourier transform v6.0, DS260 Sep. 2008, Prentice Hall.

18. Fernandes, B., & Sarmanto, H. (2010). A 128 FFT core implementation for multiband full-rate ultra-wideband receivers. *Latin American Symposium on Circuits and Systems* (*LASCAS*), Iguasu Falls, Brasil.
19. Véstias, M., Santos, H., & Sarmento, H. (2010). A DCM demapper for MB-OFDM on FPGA. *International Conference on Consumer Electronics* (*ICEE*), Jan. 2010, Las Vegas, USA.
20. Fernandes, B., & Sarmento, H. (2010). A parallel de-interleaver implementation for multiband full-rate ultra-wideband receivers. *International Conference on Design of Circuits and Integrated Systems* (*DCIS*) *XXV*, Lanzarote, Spain (submitted to publication).
21. Santos, H., Véstias, M., Neto, H., & Sarmento, H. (2009). Tradeoffs in the design of a Viterbi decoder for a MB-OFDM receiver. *International Conference on Design of Circuits and Integrated Systems* (*DCIS*) *XXIV*, Nov. 2009, Zaragoza, Spain.
22. Wold, E. H., & Despain, A. M. (1984). Pipeline and parallel-pipeline FFT processors for VLSI Implementation. *IEEE Transactions on Computers, 33*(5).
23. Simon Sherratt, R., & Makino, S. (2005). Numerical precision requirements on the multiband ultra-wideband system for practical consumer electronic devices. *IEEE Transactions on Consumer Electronics, 51*(2).

Bruno Fernandes was born in Lisbon, Portugal, in 1984. He received is Licentiate and Master degree in Physics Engineer at Faculdade de Ciências of the University of Lisbon. During these years, he collaborated with CERN has a digital hardware designer in a project for the ATLAS detector. During 2009 he collaborated in a telecommunications project in INESC-ID focusing on UWB technology. In 2010 he accepted an invitation by the University of Lisbon to be an Assistant Professor. Currently he works on the European XFEL project as an Electronic Engineer. His research topics include hardware design and embedded systems.

Helena Sarmento graduated and received her MSc and PhD degrees in Electrical and Computer Engineering at Instituto Superior Técnico, Technical University of Lisbon. She joined Instituto Superior Técnico in 1979, as teaching assistant, and is since 1996 an Associate Professor in the Electrical and Computer Department, Area of Electronics. From 1981 to 2000 she was a researcher at INESC where she leaded a group on CAD for Electronics and a group on Wireless Communication group. Since 2001, she is a senior researcher at INESC-ID. Currently is research work is done in Electronic Systems Design and Automation Group of the Embedded Electronic Systems action line. Her current interests include baseband processing for wireless communications systems, with particular emphasis on UWB, and system level design with short range wireless technologies.