A 128 FFT Core Implementation for Multiband Full-Rate Ultra-Wideband Receivers

Bruno Fernandes
INESC-ID
Email: bruno.jesus.fernanes@gmail.com

Helena Sarmento
INESC-ID/IST/UTL
Email: helena.sarmento@inesc.pt

***Abstract*—MultiBand OFDM (MB-OFDM) is a short-range wireless technology that permits data transfers at very high rates, between 53.3 and 480 Mbps. MB-OFDM uses the already licensed radio spectrum, between 3.1 GHz - 10.6 GHz, in an unlicensed manner, i.e. without a licensing cost or control. MB-OFDM divides the spectrum allocated to UWB into 14 bands of 528 MHz. Each OFDM symbol is transmitted across a band.**

The FFT processor is a crucial block in multi-carrier systems like OFDM, being responsible by thje demodulation of the OFDM symbol. In this paper we discuss the design of 128-point Pipeline FFT modules optimized for use in OFDM based UWB receiver system to be implemented on a FPGA using Xilinx FFT CORE Generator.

***Index Terms*—UWB, MB-OFDM, FFT, FPGA**

I. Introduction

MultiBand OFDM (MB-OFDM) UWB [1] is a short-range wireless technology that permits data transfers at very high rates, between 53.3 and 480 Mbps. MB-OFDM uses the already licensed radio spectrum, between 3.1 GHz and 10.6 GHz, in an unlicensed manner, i.e. without a licensing cost or control. MB-OFDM divides the spectrum allocated to UWB into 14 bands of 528 MHz. Each OFDM symbol is transmitted across a band. An OFDM symbol is a set of independent smaller symbols, modulated on separated sub carriers (FDM). The orthogonality between sub-carriers permits overlapping of sub-carriers spectrum without mutual interference. The popularity of OFDM modulation, in current communications technologies, is due to the ability to define the signal in the frequency domain, using the inverse Fourier transform. At the receiver, the Fourier transform reverses the process to obtain modulated data.

The timing parameters associated with the OFDM PHY are listed in table I. 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. The time to process an OFDM symbol (IFFT and FFT period) is 242.42 ns (1/4.125 MHz). The FFT processor is one of the modules having high computational complexity in the physical layer of the MB-OFDM UWB system.

This paper presents the implementation of the FFT for a UWB receiver, using the FFT core generator from Xilinx [2]. Different architectures were analyzed in order to obtain a processing time less than 242.42 ns. A block (FFT_to_Dem), which extracts the data sub-carriers from the 128 FFT to send to a QPSK Demapper, was also implemented.

TABLE I
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	110
Number of guard sub-carriers	10
Number of null sub-carriers	0
Subcarrier frequency spacing	4.125 MHz
IFFT and FFT period	242.42 ns

Xilinx FFT IP Core offers four different architectures, all based on the Corey-Tukey algorithm [3]: the Pipeline, Streaming I/O architecture allows continuous frame transformations, at the cost of more hardware resources; Radix-4 and Radix-2 Burst I/O architectures allow multiple input data paths and the Radix-2 Lite Burst I/O version shares arithmetic blocks in order to reduce hardware resources on implementation.

This paper is organized as follows. Section II describes the capabilities of the FFT IP Core from Xilinx, which is used to implement the OFDM demodulation at the receiver side. In section III, the FFT and the FFT_to_Dem implementation details are discussed. Results are presented in section IV. Finally, section V concludes the paper presenting future work.

II. Xilinx FFT (Core) Generator

The Xilinx LogiCORE IP FFT [2] implements the Cooley-Tukey algorithm [3] to calculate the FFT. The FFT core computes an N-point forward DFT or inverse DFT (IDFT) where N can be 2^m (m = 3 ... 16).

A. Architecture Options

The FFT core provides four architecture options to offer a trade-off between core size and transform time: Pipelined, Streaming I/O; Radix-4, Burst I/O; Radix-2, Burst I/O; and Radix-2 Lite, Burst I/O. For all architectures:

- Three arithmetic options are available: full precision unscaled arithmetic; scaled fixed-point (configured by user); block floating-point (run-time adjusted scaling).
- Run-time parameters can be configured: forward or inverse transform, scaling value between stages of the FFT and length of data points used in cyclic prefix.
- Bit/digit reversed or natural output order can be defined .

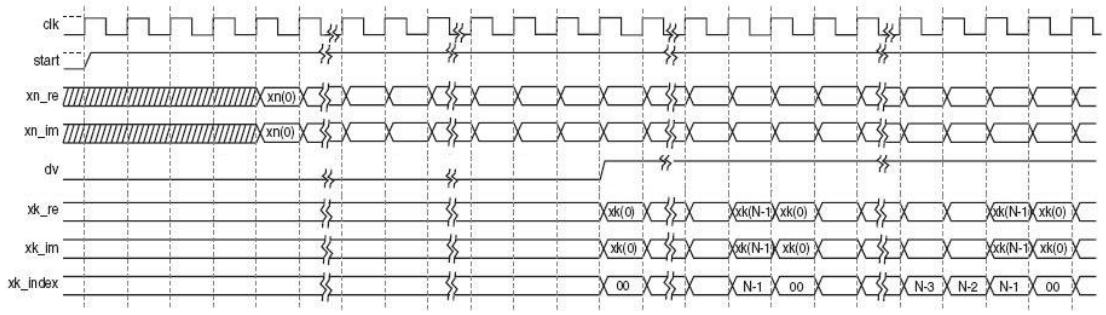

Fig. 1. Timing for Continuous-Mode [2]

- Input must be in two-complement format, output is available in two-complement or single precision floating point format.

1) Architecture Pipeline, Streaming I/O: Allows continuous data processing, using several radix-2 butterfly processing engines. Each butterfly has its own memory banks to store input and intermediate data. This architecture has the ability to simultaneously perform transform calculations on the current frame of data, load input data for the next frame of data, and unload the results of the previous frame of data. For a continuously stream of input frame data, which must presented in natural order, the FFT, after calculation latency, is continuously unloading output results.

2) Architecture Radix-4, Burst I/O: With the radix-4, Burst I/O solution, one radix-4 butterfly processing engine is used. The transform is calculated after a full frame has been loaded. Data is unloaded after the computation has finished. Data loading and unloading can be overlapped only if the unload is in digit reversed order.

3) Architecture Radix-2, Burst I/O: The FFT core uses one radix-2 butterfly processing engine and its behavioral for data I/O and transform calculation is similar to the radix-4 architecture previously mentioned. Overlap of data load and unload is possible in the same conditions.

4) Architecture Radix-2 Lite, Burst I/O: This architecture differs from the previous one in that the butterfly processing engine uses one shared adder/subtractor, reducing resources at the expense of an additional delay per butterfly calculation. Again, data can be simultaneously loaded and unloaded if the output samples are in bit reversed order.

The Pipeline, Streaming I/O architecture, which allows continuous data processing, is suitable for real-time applications where input data arrive in sequential. The architecture offers the highest throughput [2] making it the more appropriate for high rate wireless applications. The main challenging constraint to be met by the implementation is a computation time of 242.42ns for a 128-point FFT. Due to this advantages, we will use FFTs Pipelined, Streaming I/O as a base architecture for the design.

B. Timing for the Pipelined, Streaming I/O Architecture

As mentioned before, the Pipeline, Streaming I/O architecture allows continuous data processing. There are two modes for the pipeline streaming architecture: Impulse-Mode where the start signal to initiate the FFT calculation is set every N clock cycles, and Continuous-mode where the start signal is always set. Timing for Continuous-mode is presented on Figure 1, the same name signals are used in the following description. Timing for the Impulse-mode is similar, but input data shall be available on the next (clock) period after the start impulse.

Real and imaginary parts of input data, (xn_re and xn_im in figure 1) must be available three clock period after the asserting of the start signal. The Data Valid (dv) signal is set when data (xk_re and xk_im) are present at the output. The xk_index signal indicates the index of the output data.

III. FFT Implementation

An OFDM symbol has a bandwith of 528 Mhz and, therefore, arrives into the 128-FFT every 242.42ns (Table I). The required clock frequency for 128-FFT is 528 MHz. Several works addressed this issue [4] [5] [6] to implement the FFT for the OFDM demodulation process.

In [4], a block with four parallel 128-point FFT Pipeline, Streaming I/O from Xilinx core generator operating at a maximum frequency of 136 MHz was implemented. The output of each is combined using the unfolding transformation and multiple-phase clocking. Since the architecture allows 4 frames to be process in parallel, a clock frequency of 132MHz (528/4 MHz) is needed.

A complete FFT was developed from scratch in [5], using both radix-4 and radix-2 algorithms, with 8 parallel radix-4 butterflies. More recently, Nuo Li et al [6] propose a FFT design, also made from scratch, which uses the $Radix - 2^2$ Parallel algorithm [6].

In this paper we describe the implementation of pipeline FFT architecture in a Virtex-4 FPGA, using FFTs from Xilinx Core Generator as the base for the FFT design. To achieve the time requirements for OFDM demodulation, we parallelized the FFT calculation, decomposing the 128-point input in smaller sequences. The FFT Core Generator was used to compute these smaller FFTs. Smaller FFTs output is then combined to obtain the final values, maintaining continuous data processing. The arithmetic functions on the architectures are implemented using the Complex Multiplier and Adder/Subtracter functions for the Xilinx Core Generator,

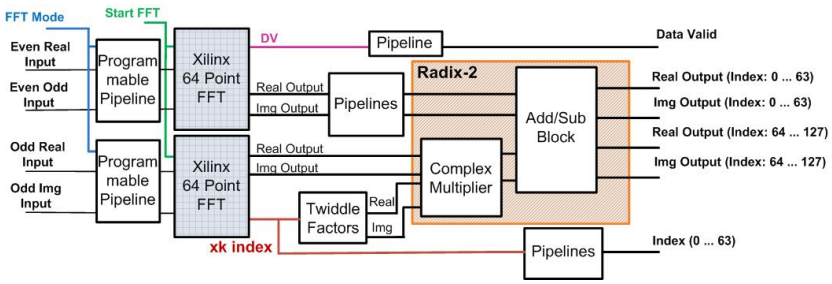

Fig. 2. 2FFTpipe Architecture

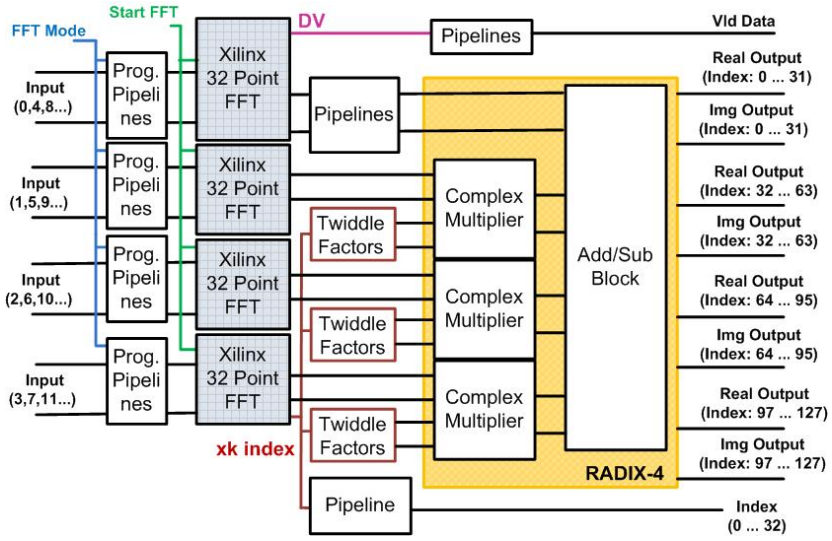

Fig. 3. 4FFTpipe Architecture

since they are optimized for Xilinx FPGAs. Also, all implementations use full-precision unscaled arithmetic, with an 8 bit input and output. The final values preserve the input fractional bits .

A. FFTs Architecture

An 128-point FFT Pipeline, Streaming I/O generated by the Xilinx FFT core could not achieve the time requirements for Virtex-4 devices. Therefore, we decide to parallelize the FFT calculation. The smaller FFTs are generated by the FFT core Generator from Xilinx and have a Pipeline, Streaming I/O Architecture. We assumed that the stream input is first written into memory blocks and then directly read, according to the input format of the architectures. Two architectures were implemented, using two or four smaller FFT in parallel. Firstly, we implement two 64 points FFTs. This architecture is referred as 2FFTpipe. Even and odd index data are computed separately. A radix-2 block computes the final values.

As presented on Figure 2, the signal to start the FFT calculation connects to the start input of the Xilinx Core Generator FFTs, mentioned in section II-B. The signal *FFT Mode*, when '0', enables Impulse-Mode timing otherwise timing is Continuous-mode. The Programmable Pipelines block in Figure 2 delay the input values for Impulse or Continuous-Mode timing. The real and imaginary twiddle factors are stored separately in RAM blocks. The xk index signal, from one of the FFT block, is used to address twiddle factor values for the complex multiplication.

TABLE II

xk index	TF Values	
	TF Real	TF Img.
0 ... 31	RAM Real	RAM Img.
31 ... 63	RAM Img.	-(RAM Real)

To reduce the use of memory block, only 32 of the 64 twiddle factor are stored. By the proprieties of the twiddle factors and equation 1 we can obtain all the twiddle factors (TF) as presented on table II.

$$cos(\phi + \frac{\pi}{2}) = -sin(\phi) \quad (1)$$

As can be seen on Figure 2, DV and xk index signals are delayed to be synchronous with the final outputs values. Likewise, the FFT even output values are delayed due to the latency of the complex multiplier.

The clock frequency needed is 256 MHz, when the 128 points input is parallelized in two subsequences. In a Virtex-4 xc4sx35, with a speed rate of -10, the maximum clock frequency obtained, although close, didn't fulfill the 242.42 ns processing time requirements.

A second architecture with four 32 points FFT in parallel, presented on Figure 3, was also implemented. The 128-point input is decomposed in 32 sequences. A radix-4 is used at the output of the 4 FFTs. All the twiddle factors are stored in separate block RAMs. A frequency of 132 MHz (528 MHz/4) is now needed.

In order to prepare data for sub-carriers demodulation, data sub-carriers must be extracted from the 128 FFT points. At the output of the FFT, a block (FFT_to_Dem block) is used to separate and store, in different memories, the pilots and data sub-carriers. The sub-carriers demodulator block needs to process simultaneously four data sub-carriers, one from each group of 25 data sub-carriers (an OFDM symbol has 100 data sub-carriers). Therefore, the FFT_to_Dem block can receive directly from the output of the 2FFTpipe (Figure 2). In order to use this block with the 4FFTpipe architecture (Figure 3), a buffer to arrange the FFT output data to the FFT_to_Dem block is necessary.

IV. Results

Both architectures were synthesized, implemented and floor planned. Test bench waveforms were created for each implementation. For simulation purposes, FFT data input is read from a text file. Output data from the FFT_to_Dem block is interpreted and written to text files (4 text files for the 25 data sub-carriers groups and 1 for the pilot sub-carriers).

Simulations proved the correct behavior of both architectures, both in Impulse and Continuous mode. FFT output values were compared with values obtained by the FFT function in MATLAB. Sub-carriers values generated by a MATLAB simulation of a MB-OFDM system were used and compared with the ones extracted in the FFT_to_Dem block.

TABLE III
Post Layout results

	Module with 2FFT Pipe	Module with 4FFTpipe
Frequency (MHz)	255.82	223.86
Slices	1,975	2,817
4 input LUTS	2,638	3,789
FIFO16/RAMB16s	6	18
DSP48s	43	81

Table III presents post layout results (maximum clock frequency and number of resources used), obtained through Xilinx ISE, for both architectures, including the FFT_to_Dem block. The maximum clock frequency is obtained by the best case achievable of setup clock in the Place and Route report. The results on the Synthesis report indicate that the maximum clock frequency is limited by the arithmetic operations on the FFT architecture. The 4FFTpipe is well suited for a MB-OFDM receiver. Although it did not achieve the time requirements, the results for the FFT2pipe are also presented since, due to the speed rate of our FPGA and the frequency obtain, we believe that the architecture is suitable for other FPGAs devices.

Table IV presents synthesis results for the 2FFTpipe and the 4FFTpipe, and for [5] and [6] which we have the results. The 2FFTpipe architecture are also presented since it uses less logic resources then the 4FFTpipe and, as previously mentioned, can be viable in different FPGA devices.

TABLE IV
Synthesis results comparison

	[5]	[6]	2FFTpipe	4FFTpipe
Frequency (MHz)	76.67	167.30	255.82	223.86
Output Lentgh (bits)	11	15	15	15
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

Comparing the logic resources used in [5], we have greatly improved the architecture of the FFT. In fact, in our previous implementation all the DSP blocks were used and extra logic blocks were used to implement the remaining arithmetic. The arithmetic operations needed are now fully implement in DSP blocks.

The algorithm used in [6] uses fewer arithmetic operations, therefore the number of DSP blocks is reduced. Although the FFT implemented is suitable, the proposed architectures can work at higher frequency and use less logic resources. By using Xilinx IP Core Generator as a base design the propose architectures are compatible with Xilinx FPGAs and gives researchers and designers the possibility to compare their results with results presented here.

V. Conclusion

This paper presents the study of two different architectures to implement an 128-point FFT module to perform the OFDM demodulation process in a receiver for UWB MB-OFDM technology. Implementations are based on the Xilinx Core Generator FFT.

Results show that combining FFT architectures Pipelined/Streaming I/O with Radix-2 and Radix-4 operations, an pipeline FFT module that meets the requirements for a UWB receiver is viable. The *4FFTpipe* architecture will be used on the MB-OFDM receiver under development. However further implementations of different FFT algorithms with Xilinx FFT IP core architectures can improve the results presented.

References

[1] Ecma International. *High Rate Ultra Wideband PHY and MAC Standard*, December 2008. Standard ECMA-368.

[2] Xilinx Inc. *Logic Core Fast Fourier Transform v6.0*, DS260 September 2008.

[3] J.W.Cooley and J.W.Tukey. An algorithm for machine computation of complex fourier series. *Math Comput*, 1965.

[4] R. Simon Sherratt, Oswaldo Cadenas, and Nomita Goswami. A Low Clock Frequency FFT Core Implementation for Multiband Full-Rate Ultra-Wideband (UWB) Receivers. *IEEE Transactions on Consumer Electronics*, 2005.

[5] N. Rodrigues, H. Neto, and H. Sarmento. A ofdm module for a mb-ofdm receiver. *Design & Technology of Integrated Systems in Nanoscale Era, 2007. DTIS. International Conference*, 2007.

[6] Nuo Li and Nick van der Meijs. Asic FFT Processor for MB-OFDM UWB System. Master's thesis, Faculty of Electrical Engineering, Mathematics and Computer Science Delft University of Technology, 2008.

[7] R. Simon Sherratt and S. Makino. Numerical Precision Requirements on the Multiband Ultra-Wideband System for Practical Consumer Electronic Devices. *IEEE Transactions on Consumer Electronics*, 2005.