SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ DERGİSİ

Sakarya University Journal of Science
SAUJS

e-ISSN 2147-835X Founded 1997 Period Bimonthly Publisher Sakarya University
http://www.saujs.sakarya.edu.tr/en/

Title: Reconfigurable and Resource Efficient Implementation of a Parallel FFT Core in FPGA

Authors: Dursun BARAN

Recieved: 2021-02-09 00:00:00

Accepted: 2021-11-02 00:00:00

Article Type: Research Article

Volume: 25
Issue: 6
Month: December
Year: 2021
Pages: 1386-1393

How to cite
Dursun BARAN; (2021), Reconfigurable and Resource Efficient Implementation of a Parallel FFT Core in FPGA. Sakarya University Journal of Science, 25(6), 1386-1393, DOI: https://doi.org/10.16984/saufenbilder.877453
Access link
http://www.saujs.sakarya.edu.tr/tr/pub/issue/66341/877453

Sakarya University Journal of Science 25(6), 1386-1393, 2021

Reconfigurable and Resource Efficient Implementation of a Parallel FFT Core in FPGA

Dursun BARAN*[1]

Abstract

Resource efficient implementation of a highly reconfigurable, parallel and pipelined FFT core that provides 1.2GS/s throughput rate with 24-bits wide input samples for the real-time spectrum analysis applications is developed and realized. Physical placement constraints are used to improve the timing performance of implemented design in FPGA. Some design techniques to reduce the memory complexities of design are also provided. Full implementation of the design is completed and implementation details are provided.

Keywords: Parallel FFT Core, Real Time Spectrum Analysis, Energy-Efficient Design, FPGA

1. INTRODUCTION

Real time spectrum analysis is a commonly used technique in electronic counter surveillance applications, RF testing and emission measurements. Conventional swept spectrums are failing to demonstrate the intermittent signals that are available very occasionally. Those kind of signals are very critical for the RF emission measurements, the radar signal detection, the signal classification and similar applications. RF emission measurements are also getting more complex and require higher bandwidths as new communication technologies are introduced such as 5G and beyond [1]. In order to support larger RF bandwidths, higher speed ADCs (***A**nalog to **D**igital **C**onverter*) or parallel RF front ends are needed. The usage of parallel RF frontends becomes costlier and more complex as compared the usage of high speed ADCs. In addition, the sampling rates of current ADC devices already pass 5GSPS (***G**iga **S**ample **P**er **S**econd*) that enables to digitize more than 2.5GHz bandwidth without any complex RF front ends. After high speed ADCs, backend digital processing blocks must be able to process all digitized samples.

In order to perform spectrum analysis, RF channel power measurements or FFT (***F**ast **F**ourier **T**ransform*) techniques can be used. For real-time spectrum analysis applications, FFT technique is more suitable since it transforms the input samples from time domain to the frequency domain. Depending on the digital processing power, various FFT speeds, lengths and architectures can be successfully realized [15-16]. To perform FFT algorithm in digital processors efficiently, DFT (***D**iscrete **F**ourier **T**ransform*) technique is widely used. The DFT for a frame size of N is defined as (where k = 0, 1, …, N-1)

* Corresponding author: barandursun@yahoo.com
[1] The Scientific and Technological Research Council of Turkey, Ankara, Turkey
ORCID: https://orcid.org/0000-0001-9277-3796

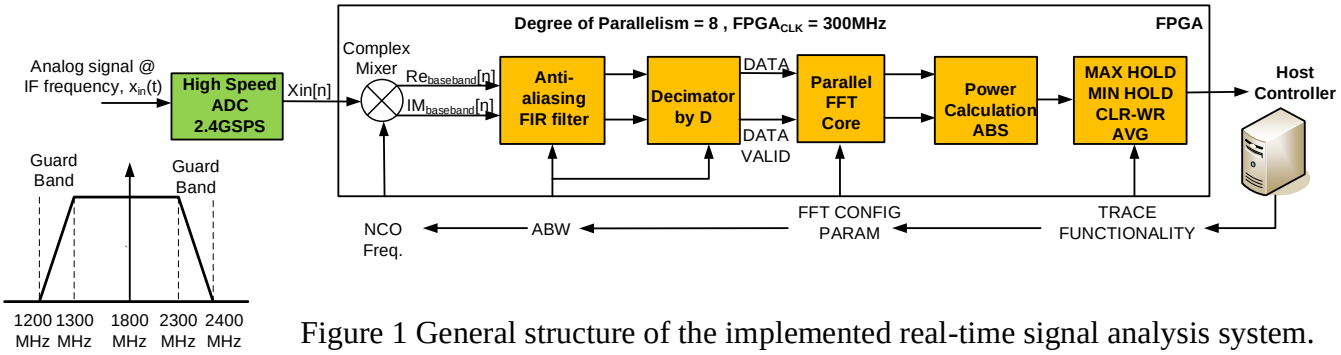

Figure 1 General structure of the implemented real-time signal analysis system.

$$X(k) = \sum_{n=0}^{N-1} x(n) * e^{-j2\pi kn/N} \qquad (1)$$

For custom applications, FPGA (***Field Programmable Gate Array***) is widely used computational environment since they are providing reprogrammable capability to developers with a reasonable cost and performance metrics. In terms of performance and power consumption, FPGA is worse than their corresponding ASIC realization. However, FPGA is able to provide more cost efficient implementation for prototype products since ASIC design requires a high budget for prototype design and tape-out cycles. Current FPGA devices provides lots of resources to designers that can be used to implement the real-time and parallel signal processing algorithms [8]. When a high speed ADC is used in front of FPGA device, FPGA internal clock cannot reach the same speed. This requires to implement a parallel and pipelined FFT core to realize real-time spectrum analysis [2-7] for high bandwidth applications.

In this paper, we provide the implementation details of a highly reconfigurable and resource efficient parallel FFT processing core to be used in spectrum monitoring applications. In addition to the developed parallel FFT core, a digital downconverter block, ADC interface and host controller connectivity circuits are also developed in HDL and realized in FPGA as well. The design detail of those auxiliary circuits will not be demonstrated in this work but they are used to realize the system. In Section II, the general structure of system is given. Section III provides design details of implemented parallel FFT core. Design optimization techniques are detailed in Section IV. Implementation results are given in Section V and Section VI concludes the work.

2. ARCHITECTURE OF THE REAL-TIME SPECTRUM ANALYSIS SYSTEM

The general structure of the implemented real-time signal analysis system for wideband applications is given in Figure 1. Firstly, the analog/RF signal is sampled by using and an ADC that has 2.4GSPS sampling rate that can provide 1.2GHz analysis bandwidth to the overall system if the digital processing hardware is able to support such a speed. Unfortunately, it is not possible to clock any commercially available FPGA device with a 2.4GHz signal. Parallel hardware design technique is used to reduce the FPGA clock requirement to support such demanding applications. The degree of parallelism is selected to be 8 in this design to make FPGA clock is feasible and the FPGA clock becomes 300MHz in this case that is easily supported by the current FPGA ICs.

Sampled signal is moved to the baseband using 8 complex mixers within FPGA. The required NCO (***Numerically Controlled Oscillator***) signal is generated within FPGA and the NCO frequency is controlled by the host controller to support direct digitization path as well. After complex mixing stages, the signal is fed to anti-aliasing filter stage that is implemented using FIR (***Finite Impulse Response***) filters in FPGA. After this filtering stage, the signal is moved to decimator stage to reduce the signal analysis bandwidth. The degree of decimation D is provided through the host controller as well. After decimation stage, the

signal is fed to a parallel FFT core to measure the signal power at each frequency bin index. The bandwidth of each frequency bin is calculated as

$$BW_{FFT-BIN} = \frac{ABW}{N_{FFT}} \quad (2)$$

where *ABW* is analysis bandwidth, N_{FFT} is the number of FFT point and $BW_{FFT\text{-}BIN}$ is the bandwidth of each FFT-BIN.

After implementing the parallel FFT core, the signal powers at each bin is calculated using an absolute value calculation block. The absolute value calculation simply implemented using two multipliers, an adder and a square root operation. The final stage of the real-time spectrum analysis system is to calculate the max-hold/min-hold/clr-wr/avg trace functions in real time as shown in Figure 1. The block should be able to process all samples without any data loss to provide the real-time spectrum analysis feature. When a new data transfer is requested from the host controller, this block must be able to continue to calculate the trace functionality while transferring data. The data transfer to host PC is implemented using a DMA (**D**irect **M**emory **A**ccess) connectivity to support raw data transfer. The output of parallel FFT core data does not require very high bandwidth connectivity when some trace functionality is selected instead of the raw data transfer.

As seen from Figure 1, the system is highly reconfigurable in terms of the NCO frequency, the analysis bandwidth (ABW), the FFT configuration parameters and the trace functionality. Depending on the ABW configuration, DATA and DATA VALID signal characteristics are changed and FFT core must be able to handle those changes. The summary of design specifications for the spectrum analysis system is given in Table 1. The maximum analysis bandwidth is 1.2GHz and user will use 1GHz bandwidth out of it. 100MHz guard bands (1200MHz to 1300MHz and 2300MHz to 2400MHz) are defined at the left and the right side of IF frequency bands to prevent any signal aliasing may occur between Nyquist bands as shown in Figure 1.

Table 1 Real-Time Spectrum Analysis System Design Specifications

Technical Specifications	Value
ADC Sampling Rate	2.4GHz
SFDR	73dBc (typical)
SNR	58.5dB (typical)
FPGA Clock Frequency	300MHz
ABW	1.2GHz ~ 1MHz
IF Frequency	1800MHz
Sampled IF Bandwidth	1.2GHz - 2.4GHz
Trace Functionalities	MAX-HOLD, MIN-HOLD, AVG, CLR-WR
FFT Core Point	32k - 16
FFT Windowing	Supported
*Spectrum Sweep Speed	73.24THz/Sec

* 32K FFT size and IF bandwidth is 1.2GHz

3. PARALLEL FFT PROCESSING CORE

Four consecutive baseband real and complex data samples are packed into a DATA register and a DATA VALID signal conveys the valid trigger to load data into FFT core. The timing waveform of DATA_VALID signal is shown in Figure 2 for various ABW values. DATA register is 192-bits wide and includes 4 real and 4 imaginary parts of the sampled data as given in Figure 3. DATA_VALID signal depends on the analysis bandwidth and the worst case scenario happens when the analysis bandwidth is set to 1200MHz as shown in Figure 2, DATA and DATA_VALID signals are synchronous with respect to the FPGA_CLK.

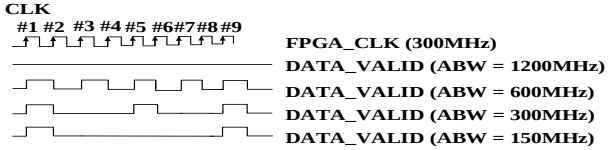

Figure 2 Timing waveform of DATA_VALID signal with respect to FPGA Clock.

DATA [191:168]	DATA [167:144]	DATA [143:120]	DATA [119:96]	DATA [95:72]	DATA [71:48]	DATA [47:24]	DATA [23:0]
IM[n-3]	RE[n-3]	IM[n-2]	RE[n-2]	IM[n-1]	RE[n-1]	IM[n]	RE[n]

Figure 3 Content of DATA register to be processed by parallel FFT core.

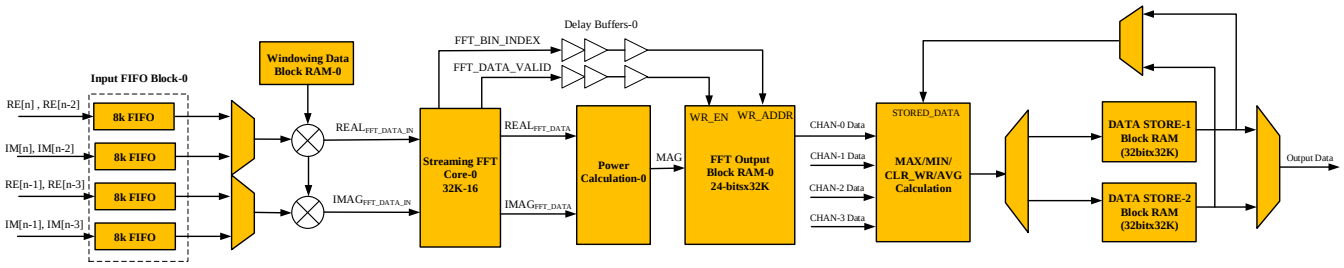

Figure 4 The block diagram of implemented parallel FFT core (one channel out of four is demonstrated)

Parallel FFT core must handle to calculate DFT of incoming data samples without any loss. When the analysis bandwidth is set to 1200MHz, the parallel FFT core must be able to process all sampled data. This requires 4 FFT cores each running at 300MHz FPGA clock. In addition to FFT cores, some memory hardware is required to temporarily store the incoming data. In order to minimize FPGA resource utilization, minimum number of memory should be used. The block diagram for a channel of implemented parallel FFT core is given in Figure 4.

Incoming data is first stored to 8k FIFO (***First In First Out***) blocks with 24 bits wide. When there is some amount of data available in FIFO, they are moved to FFT cores by the help of empty signal triggers generated from the FIFO blocks. 4 FIFO memory blocks are used for each FFT channel to support 2 Real and 2 Imaginary data samples. When analysis bandwidth is set to 1200MHz, 4 Real and 4 Imaginary data come at each clock cycle that necessitates 8 FIFO memory blocks if they are clocked with the FPGA clock itself. As seen from Figure 4, the need for memory is high in this design and it is important to reduce the amount of memory usage. So, input FIFO is clocked with 2xFPGA_CLK to capture 4 Real and 4 Imaginary data at each FPGA clock when the analysis bandwidth is set to 1200MHz. For other analysis bandwidths, there is no need to clock input FIFO blocks with 2xFPGA_CLK. In those cases, input FIFO clock is derived with FPGA_CLK itself. The clocking scheme of input FIFO block is shown in Figure 5. By using this clocking scheme, 4*8K*24-bits wide memory resources will be saved at expense of a clock multiplier and a BUFG-CTRL that is a controlled clock buffer driver from Xilinx FPGA family [9].

There are various kinds of windowing functions such as Hanning, Blackman Harris and similar coefficient sets that can be applied to DFT calculations [10]. It is very memory consuming to store all windowing coefficients into FPGA memory, therefore a single block RAM is reserved for windowing coefficient set that can be written from host controller to load windowing coefficients before the start of FFT core. The data comes from input FIFO and windowing coefficient RAMs are multiplied before entering FFT blocks. In the system, four block RAMs are reserved for windowing coefficients to support the four parallel cores.

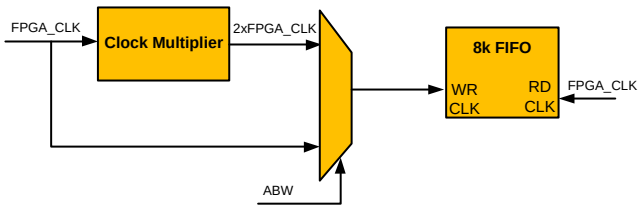

Figure 5 Clocking scheme of input FIFO memory blocks.

A reconfigurable FFT core is used to calculate DFT function as shown in Figure 4 [11]. The outputs of FFT core is provided to power calculation block to convert the complex output of FFT block to signal power amplitude. FFT bin index is used to generate write address of the FFT output block RAM. FFT_DATA_VALID signal is used to generate write enable signal of FFT Output Block RAM as shown in Figure 4.

In order to compensate the delay of power calculation block, delay buffers are added to FFT_BIN_INDEX and FFT_DATA_VALID signals. The amount of delay depends on the implementation of power calculation block. 4

parallel FFT cores are able to fill the FFT Output Block RAMs depending on the FFT configuration register provided from the host controller. If only one channel is sufficient, the rest of channels will be disabled. In order to support 1.2GHz analysis bandwidth case, 4 channel of FFT cores are required. For lower analysis bandwidth cases, the unused FFT channels are reserved for overlapping purpose. The supported overlap percentages for selected analysis bandwidth is provided in Table 2.

Table 2 Supported Overlap Percentages

Analysis Bandwidth	Supported Overlap Percentage
1200MHz	0%
600MHz	0%, 50%
300MHz	0%, 25%, 50%, 75%
Lower Bandwidths	0%, 25%, 50%, 75%

Power calculation block calculates the mathematical operation given in eq. 3. Square root operation is applied to reduce the bit widths of FFT Output Block RAMs. It is an important optimization since this design has a serious bottleneck about the memory resources.

$$MAG = SQRT(REAL_{FFT\_DATA} * REAL_{FFT\_DATA} + IMAG_{FFT\_DATA} * IMAG_{FFT\_DATA}) \quad (3)$$

CHAN-0 Data is the output data from FFT Output Block RAMs that is the magnitude of FFT output data for channel-0 as shown in Figure 4. There will be CHAN-1 Data, CHAN-2 Data and CHAN-3 Data depending on the activated FFT channels. MAX/MIN/CLR_WR/AVG block calculates the trace function selected by the host controller. As name implies, MAX holds the maximal values of data at each FFT_BIN_INDEX and MIN holds the minimal values. CLR_WR function update the content of DATA_STORE block rams with the recent output of FFT cores. AVG function accumulates the values in DATA_STORE rams for a given number of averages. Division operation is completed in host controller side since the division operation is quite costly when implemented in FPGA. Two block RAMs are used to store final output data to prevent any data loss when the data is requested by the host controller. One RAM block stores the current FFT data and other RAM block transfers the stored data to the host controller. Whenever a data read request is generated from the host controller, DATA_STORE ram is switched. One RAM block is cleared and then start to store the new incoming data values. The data in other RAM block will be sent to the host controller.

4. DESIGN OPTIMIZATIONS AND TIMING ENCLOSURE TECHNIQUES

This design requires to use large memory blocks to support 32K FFT size and the real-time analysis features. However, the available memory resources are limited in FPGA [8]. Therefore, the memory usage must be reduced as much as possible to realize the design. There are two resource optimization techniques are applied to design. First one is to drive the clock of input FIFO with 2xFPGA_CLK to write the incoming data for 1.2GHz ABW case. Also, the system starts to operate whenever any data is stored in input FIFO to reduce the size of input FIFO memory as well. This will reduce the FIFO depth to 8K instead of 32K. All RAM memory blocks are implemented using FPGA internal RAM resources.

In order to improve the timing enclosure of the design, physical placement constraints are used. All block RAMs, DSP units and other configurable logic blocks are constrained to fit the specified region of FPGA to get better performance values. In order to implement large block RAMs, synthesizer will use multiple small block RAMs to realize it. As an example, Xilinx Virtex-7 series has internal 18Kb and 36Kb block RAMs to implement larger RAM blocks [8]. Small block RAMs are placed in predetermined regions horizontally or vertically. Therefore, constraining synthesizer to use small block RAMs within the specified regions improve the timing and the implementation time results considerably. Similarly, the FFT cores, the complex multipliers and the power calculation blocks are constrained to be placed to a predetermined region within the target FPGA.

5. RESULTS

The parallel FFT core given in Figure 4 and preceding digital down converter stages are implemented in Xilinx Virtex-7 VX690T model FPGA using HDL. ADC12D800RF model analog to digital converter is used to realize the full system [13]. Target FPGA_CLK period is set to 3.33ns to satisfy 300MHz in Xilinx PlanAhead design environment [14]. Four FFT core blocks are constrained with the physical placement constraints and the final placement result is demonstrated in Figure 6. As shown in this figure, all large RAM blocks are constrained to be placed to nearby small block RAMs. In addition to RAM blocks, four FFT and corresponding power calculation blocks are constrained to place some predefined regions within the FPGA architecture. By using this design technique, the placer successfully meets the timing closures for the complete design. In addition, the buffering technique is commonly used to improve the timing performance of the design similar to ASIC (***A**pplication **S**pecific **I**ntegrated **C**ircuit*) counterparts [12]. After determining the failing paths, extra buffers are added to meet the timing constraints. In order to determine the failing paths, the static timing analyzer of Plan Ahead tool is used.

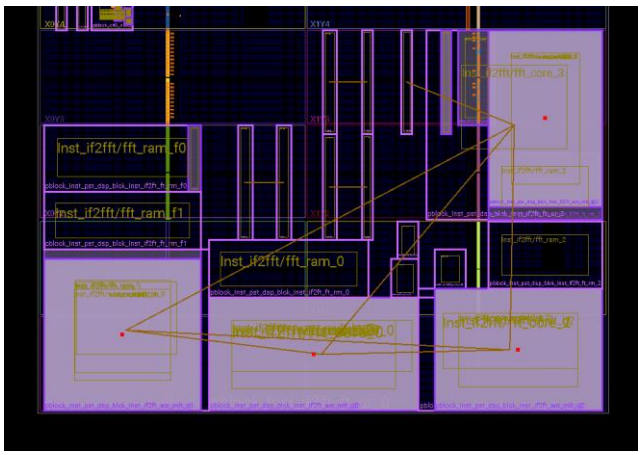

Figure 6 FPGA layout after place and route cycle.

FPGA design steps are followed to generate the final programming file. The resource utilization of parallel FFT core within FPGA is given in Table 3. By placing the FFT cores to specified region of FPGA, routing complexity of the design will be reduced in addition to the timing improvements.

Table 3 Resource utilization of the parallel FFT core

FPGA Resource	Utilization
SLICEL	6839
SLICEM	6119
DSP48E1	508
RAMBFIFO36E1	282
RAMBFIFO18E1	412
BUFGCTRL	1
MMCM (Mixed-Mode Clock Manager)	1

The digitizer board [13] is placed in a PC hardware with a suitable graphic card, a processor and a RAM. The system is connected a HD monitor over HDMI connector. The system is connected to a signal generator using a SMA cable and signal frequency is set to a value within the IF frequency bandwidth. An application program is developed for the parallel FFT core to demonstrate the performance of the system. An example graphic from the developed application is shown in Figure 7.

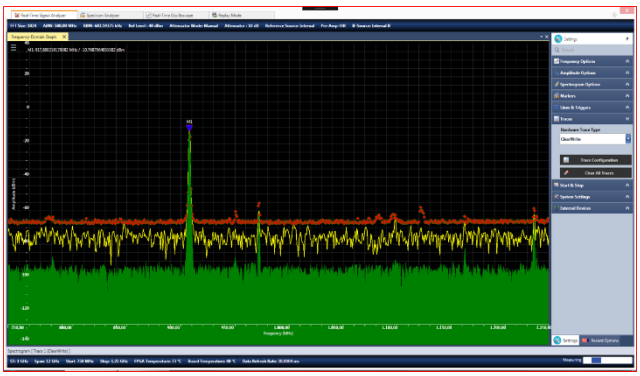

Figure 7 A graphical user interface (GUI) example for the developed FFT core. Each color shows different traces namely max-hold, min-hold, clr/wr and avg.

6. CONCLUSION

A highly reconfigurable and resource efficient implementation of a parallelized FFT core design is demonstrated in this work. The implemented four parallel FFT cores are able to operate simultaneously with 300MHz FPGA clock that provides 1.2GS/s throughput. Various optimization techniques used to reduce the FPGA

memory usage and improve the timing enclosure. By using physical placement constraints, the timing performance of the design is improved. The proposed design is implemented in a 7 Series Xilinx FPGA and the final programming file is generated. FPGA is programmed with the generated file to demonstrate successful operation of the developed parallel FFT core. In future, custom coded FFT cores may be developed to improve the performance of the parallel FFT core furthers in terms of speed and RF specifications.

Acknowledgments

Many thanks to Enes Karav and my colleagues in Sensor and Antenna Systems department for their invaluable technical discussions.

Funding

This work is supported by informatics and information and security research center (*BILGEM*).

The Declaration of Conflict of Interest/ Common Interest

"No conflict of interest or common interest has been declared by the authors. "

Authors' Contribution

The first and only author contributed 100% to this study.

The Declaration of Ethics Committee Approval

"This study does not require ethics committee permission or any special permission."

The Declaration of Research and Publication Ethics

"The authors of the paper declare that they comply with the scientific, ethical and quotation rules of SAUJS in all processes of the paper and that they do not make any falsification on the data collected. In addition, they declare that Sakarya University Journal of Science and its editorial board have no responsibility for any ethical violations that may be encountered, and that this study has not been evaluated in any academic publication environment other than Sakarya University Journal of Science."

REFERENCES

[1] C. Fager, T. Eriksson, F. Barradas, K. Hausmair, T. Cunha and J. C. Pedro, "Linearity and Efficiency in 5G Transmitters: New Techniques for Analyzing Efficiency, Linearity, and Linearization in a 5G Active Antenna Transmitter Context," in *IEEE Microwave Magazine*, vol. 20, no. 5, pp. 35-49, May 2019.

[2] C. Eddington, B. Ray, "Using parallel FFT for multi-gigahertz FPGA signal processing", EE Times Magazine, https://www.eetimes.com/using-parallel-fft-for-multi-gigahertz-fpga-signal-processing/

[3] X. Zou, Y. Liu, Y. Zhang, P. Liu, F. Li and Y. Wu, "FPGA Implementation of Full Parallel and Pipelined FFT," 2012 8th International Conference on Wireless Communications, Networking and Mobile Computing, Shanghai, 2012, pp. 1-4.

[4] H. Kanders, T. Mellqvist, M. Garrido, K. Palmkvist and O. Gustafsson, "A 1 Million-Point FFT on a Single FPGA," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 10, pp. 3863-3873, Oct. 2019

[5] M. Dreschmann et al., "Implementation of an ultra-high speed 256-point FFT for Xilinx Virtex-6 devices," 2011 9th IEEE International Conference on Industrial Informatics, Caparica, Lisbon, 2011, pp. 829-834.

[6] Shousheng He; Torkelson, M.; " *A new approach to pipeline FFT processor* ," Parallel Processing Symposium, 1996, Proceedings of IPPS '96, The 10th International, April 1996.

[7] V. Iglesias, J. Grajal, M. A. Sánchez and M. López-Vallejo, "Implementation of a Real-

Time Spectrum Analyzer on FPGA Platforms," in IEEE Transactions on Instrumentation and Measurement, vol. 64, no. 2, pp. 338-355, Feb. 2015.

[8] https://www.xilinx.com/products/silicon-devices/fpga.html

[9] https://www.xilinx.com/support/documentation/user_guides/ug472_7Series_Clocking.pdf

[10] S. Rapuano and F. J. Harris, "An introduction to FFT and time domain windows," in *IEEE Instrumentation & Measurement Magazine*, vol. 10, no. 6, pp. 32-44, December 2007.

[11] https://www.xilinx.com/support/documentation/ip_documentation/xfft/v9_1/pg109-xfft.pdf

[12] B. R. Zeydel, D. Baran and V. G. Oklobdzija, "Energy-Efficient Design Methodologies: High-Performance VLSI Adders," in *IEEE Journal of Solid-State Circuits*, vol. 45, no. 6, pp. 1220-1233, June 2010.

[13] https://www.pentek.com/products/detail.cfm?model=78741

[14] https://www.xilinx.com/products/design-tools/planahead.html

[15] Palmer J., Nelson B. (2004) A Parallel FFT Architecture for FPGAs. In: Becker J., Platzner M., Vernalde S. (eds) Field Programmable Logic and Application. FPL 2004. Lecture Notes in Computer Science, vol 3203. Springer, Berlin,

[16] K. Nguyen, J. Zheng, Y. He and B. Shah, "A high-throughput, adaptive FFT architecture for FPGA-based space-borne data processors," 2010 NASA/ESA Conference on Adaptive Hardware and Systems, 2010, pp. 121-126