UDC 621.382.049:004.93'1
DOI: 10.15587/2706-5448.2025.336992

Oleksandr Vasyliev,
Oleh Filippenko,
Inna Filippenko,
Oleksandr Shkil

DEVELOPMENT OF HARDWARE-SOFTWARE MODEL FOR SIGNAL SPECTRUM COMPUTATION USING FAST FOURIER TRANSFORM BASED ON FPGA

The object of research is the implementation methods of an adaptive hardware-software model for signal spectrum analysis using Fast Fourier Transform (FFT), implemented on a Field-Programmable Gate Array (FPGA) followed by processing in the software part. This solution combines the advantages of hardware acceleration and software flexibility. The proposed model is aimed at solving the problem of creating an efficient tool for real-time signal processing, taking into account limitations in accuracy, latency, resource usage, and data retention for further processing and analysis. The model is designed with scalability in mind, both in terms of increasing the number of processing channels and extending the FFT length and precision level. Its development included stages of modeling, synthesis, debugging, and testing close to real-world conditions. The structure of the model was thoroughly designed, data representation formats and rounding procedures were optimized, and the FFT algorithm was adapted to the specifics of the chosen platform. Altogether, this ensured high accuracy of spectral analysis and efficient use of FPGA resources, as confirmed by experimental data. Practical testing of the system in real time was conducted, during which such parameters as result accuracy and power consumption were evaluated, considering the efficient use of logic elements and memory blocks. The obtained results logically reflect the advantages of the hardware-software implementation, the usage of optimized data formats and rounding procedures, as well as the successful adaptation of the FFT algorithm. This allowed achieving a balance between high spectral analysis accuracy at the level of 3.97 kHz with an FFT length of 16,384, a two-fold reduction in the required memory size, and a 0.25 ms decrease in FFT result transmission time. The practical applications of the developed model cover a wide range of fields, including embedded signal processing systems, modern real-time measurement devices, as well as mobile or energy-efficient systems, where real-time processing under low power consumption is critical. Thanks to its versatility, the model can be integrated into more complex digital signal processing systems, expanding their functionality.

Keywords: *model, Fast Fourier Transform, Field-Programmable Gate Array, Python, magnitude, rounding, accuracy, telecommunications.*

Received: 16.05.2025
Received in revised form: 04.07.2025
Accepted: 24.07.2025
Published: 29.08.2025

How to cite
Vasyliev, O., Filippenko, O., Filippenko, I., Shkil, O. (2025). Development of hardware-software model for signal spectrum computation using Fast Fourier Transform based on FPGA. Technology Audit and Production Reserves, 4 (2 (84)), 99–107. https://doi.org/10.15587/2706-5448.2025.336992

1. Introduction

In the modern world, digital signal processing is a critically important component of numerous technologies – from consumer devices to high-end scientific systems. Among the methods that enable the transformation of signals from the time domain to the frequency domain, the Fast Fourier Transform (FFT) holds a key position. Its applications span spectral analysis, telecommunications, medical diagnostics, radar systems, and autonomous platforms. With the growing prevalence of real-time computing systems, there is an increasing demand for high-performance, energy-efficient, and integrated solutions for FFT computation.

Despite many years of experience in using FFT, several important challenges still require more optimal solutions. These include the high-power consumption of classical implementations, the relative complexity of hardware realization of algorithms with complex arithmetic, low efficiency when processing large data arrays, and limited scalability of existing architectures. Many current implementations are tailored for use under fixed parameter conditions, which complicates their deployment in flexible or multifunctional systems.

Literature sources indicate an active search for new approaches to FFT implementation based on Field-Programmable Gate Arrays (FPGAs). For instance, [1] proposes a novel P-parallel FFT architecture that optimizes the number of multiplexers and memory utilization by managing access and merging memory banks. It is shown that the FPGA-based implementation demonstrates high hardware efficiency, making it promising for use in high-performance and compact systems such as 6G communication platforms. However, the evident advantages of this solution do not optimize the use of embedded FPGA memory resources, which is an important factor for applying the FFT algorithm in modern hardware.

In [2], the research of a high-performance 64-parallel FFT architecture based on memory with a length of 4096 points is investigated. This architecture is optimized for low latency and high throughput and is intended for use in 6G communication systems. However, it has limited scalability potential due to the specifics of the 64-parallel algorithm, which requires significant additional FPGA resources and does not support increasing the transform length, thereby limiting the spectral resolution of the solution.

ISSN-L 2664-9969; E-ISSN 2706-5448

Work [3] is dedicated to the development of a high-performance and flexible FFT architecture based on reconfigurable mixed-radix algorithms, featuring twiddle factor compression and conflict-free memory access, aimed at enhancing throughput and adaptability. The authors present optimization results for an implementation using a 40 nm CMOS technology.

Thus, works [1–3] are focused on static optimization, namely improving FFT implementations targeted at specific tasks. However, they do not offer flexibility or scalability. This limitation may complicate the use of such models in applications that require higher frequency resolution, for instance, when processing complex signals with a wide frequency range.

To reduce the impact of errors on digital signal processing modules, [4] investigates the effect of architectural choices on error rate, common-mode failure rate, and signal-to-noise ratio (SNR) across different FFT architectures. The authors propose a method for detecting the correlation between error rate and FPGA resource utilization, which is critical for systems where failures are unacceptable, such as in aerospace, nuclear installations, or medical devices.

Methods for improving phase rotation blocks for use in FFT are the focus of works [5–8], with particular attention to reliability, performance, latency, energy efficiency, and adaptive distortion correction.

In [5], the development of an innovative method for fault classification and diagnosis in Coordinate Rotation Digital Computers (CORDIC) processors implemented on FPGAs is investigated. The approach combines FFT and Convolutional Neural Networks (CNNs), with a focus on enhancing the accuracy of fault detection and classification. However, processing accuracy is not addressed.

Study [6] proposes an architecture that reduces hardware resource usage by employing radix-16 and CORDIC, enabling rotation computations without additional multipliers. This results in high performance with low power consumption and compactness, as confirmed by FPGA testing, where the architecture outperformed traditional radix-2/4 approaches. The method is intended for use in embedded systems where compactness and energy efficiency are critical, such as portable communication devices, IoT sensors, and signal processing systems with constrained resources. These systems require high performance with minimal latency and hardware overhead.

In [7], a parallel rotator architecture is proposed, which addresses computational complexity and reduces processing latency. This significantly accelerates FFT computation and reduces overall latency, as confirmed by FPGA testing demonstrating increased throughput. The solution is relevant for systems that demand high throughput and low latency, such as wireless networks, MIMO systems, and other technologies where parallel processing is essential.

Study [8] focuses on using FFT to increase the accuracy of pulse decay detection. The goal is to enable a fast calibration procedure in preamplifiers to match different noise spectra and compensate for unpredictable signal fluctuations.

Studies [9–11] examine the use of FFT in high-speed and specialized signal processing systems, focusing on function integration, actual performance of FPGA-based solutions, and analytical optimization of accuracy and energy efficiency in fixed-point implementations.

The discussed solutions demonstrate potential for enhancing the performance and adaptability of FFT. It can be observed that the target applications of the reviewed works are either highly specialized or structurally complex for broader integration. Specifically, [1, 2] are aimed at 6G communication systems with requirements for low latency and high energy efficiency; [3] focuses on software-defined radio (SDR) systems with an emphasis on flexibility; [4, 5] address reliability and fault diagnostics in critical systems such as aerospace and nuclear technologies; [6, 7] are oriented toward telecommunications applications (5G, OFDM); [8] targets nuclear spectroscopy; [9] addresses optical communication systems (400G Ethernet); and [10, 11] are focused on plasma diagnostics and cyclostationary analysis. This highlights the relevance of conducting a study dedicated to the development of a universal or unified, adaptive, and simultaneously easy-to-implement model that combines hardware and software components, offering high performance while accounting for constraints in accuracy, throughput, energy consumption, and flexibility.

Thus, the need to address the challenge of developing an efficient, configurable, and unified architecture for FFT execution in modern signal processing systems underlines the significance of this research. The objective of research is to create an adaptive hardware-software model for signal spectrum computation using FFT, which, by leveraging the advantages of both software and hardware domains, will provide a balanced trade-off between performance, energy efficiency, and dynamic parameter control.

An important advantage of having a software component in the model is its potential for integration into complex systems. It should be noted that, under real-world conditions, certain computations are significantly more efficient when performed in software. For example, the MUSIC and ESPRIT algorithms, used for estimating the angle of arrival (AoA) in multichannel receiving systems such as antenna arrays, provide extremely high resolution and can distinguish signals arriving at closely spaced angles. However, they require matrix decomposition, operations with complex numbers, and high-precision arithmetic. Implementing these algorithms in hardware, particularly on FPGAs, is complicated by their computational complexity and dynamic parameter variation. Such adaptability requires a flexible architecture that is difficult to realize in hardware. In contrast, FPGAs are optimally suited for real-time streaming pre-processing. This hybrid approach allows for processor offloading and reduced latency, while high-level analysis is performed in the software domain.

To address the outlined challenges, it is necessary to develop an integrated hardware-software model that combines the advantages of hardware acceleration and software flexibility for FFT computation, optimized in terms of performance, accuracy, power consumption, and resource utilization. The use of modern approaches to FPGA-based FFT design is intended for constructing an optimized hardware-software model. The proposed model will enable the implementation of efficient and scalable solutions for real-time signal analysis, which is particularly relevant for telecommunications, medical diagnostics, radar systems, autonomous control, and other fields where real-time spectral processing determines overall system performance.

2. Materials and Methods

The object of this research is the implementation methods of an adaptive hardware-software model intended for signal spectral analysis using the Fast Fourier Transform (FFT). Within the scope of the research, a pipelined model capable of operating in real time has been developed. The goal of the design is to create a system that enables efficient signal processing with minimal latency, high accuracy, and convenient control. To ensure an optimal and unified implementation of the model, the following requirements were defined:

- a high-quality modern ADC for test signal input;
- inclusion of a reference signal generation block;
- implementation of filtering and post-FFT processing mechanisms;
- development of a reliable communication protocol for hardware control;
- integration of systems for data output, compression, and storage;
- capability for real-time result visualization.

The proposed hardware part of the model is shown in Fig. 1.

To meet real-time operation requirements, a pipelined FFT with decimation-in-time (DIT) was selected. The core idea of the FFT algorithm is to optimize and reduce the number of operations compared to the discrete Fourier transform (DFT). When this optimization is

applied to a pipelined implementation, intermediate samples become permuted during butterfly operations, resulting in output data being arranged in bit-reversed order. Thus, pipelined FFT implementations by default produce results in bit-reversed order. Modern IP cores for pipelined FFT provide the option to output results in natural order, but this requires additional memory for buffering. Since in bit-reversed order the sample corresponding to the zero-frequency bin is the first result of the pipelined FFT, and the sample corresponding to the first-frequency bin appears much later, the memory used for reordering must accommodate the entire FFT output. In other words, to obtain FFT results in natural order, memory usage requirements are effectively doubled.

This work proposes a method that leverages the described properties of FFT for optimization. The hardware part of the model outputs the FFT results in bit-reversed order, while the reordering into natural order is delegated to the software domain, where buffering memory is no longer a bottleneck. By eliminating hardware-based buffering for reordering, the actual data transmission latency is reduced. The buffering latency required for reordering to natural order at the given sampling rate is

$$t_d = \frac{2^N}{F_s}, \quad (1)$$

where t_d – the buffering delay time, F_s – the sampling frequency, and 2^N – the FFT length.

Considering that only the first Nyquist zone is required for signal analysis without aliasing, the samples falling outside this range can be filtered out. Thus, it is optimal to keep only the samples within the observation range, which is dynamically defined by the control software. To enable efficient signal analysis, the system filters out samples outside the first Nyquist zone and retains only those that fall within the range specified by the software parameters. This approach avoids redundant data transmission and ensures optimal resource utilization.

The software part of the model is designed as a modular application with the following structure: modules for handling communication interfaces, a module for converting FFT output into natural order, a module for compression and storage, a graphical user interface (GUI) module, a control command generation module, and a module for dynamic real-time plotting. The proposed modular structure of the software component is shown in Fig. 2.

When developing the software application according to the proposed model, special attention should be given to its modularity. In this case, modularity provides the software component with greater flexibility and simpler adaptation to new tasks. Such a multithreaded design approach allows each module to be executed in a separate software thread. This enables the use of modern multicore systems and thus accelerates the processing of incoming signals. This is particularly beneficial in modern hybrid processor systems, where cores have different architectures and performance characteristics. The proposed modular software structure leverages the advantages of energy-efficient computing platforms and therefore does not become a bottleneck in the operation of the hardware-software model. Modularity ensures the flexibility and adaptability of the software component while maintaining its versatility.

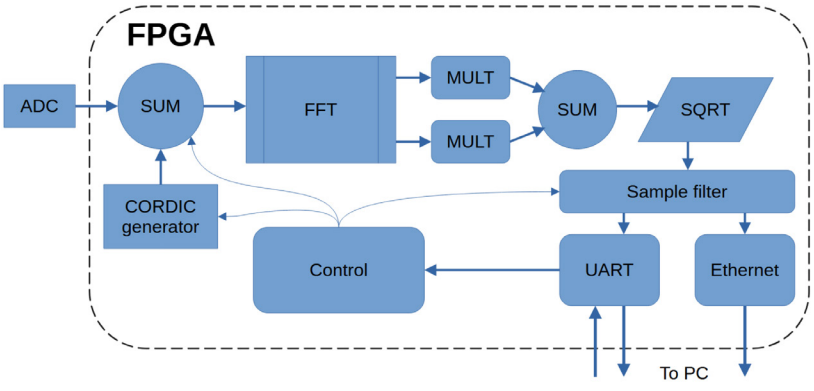

Fig. 1. Proposed hardware structure of the model

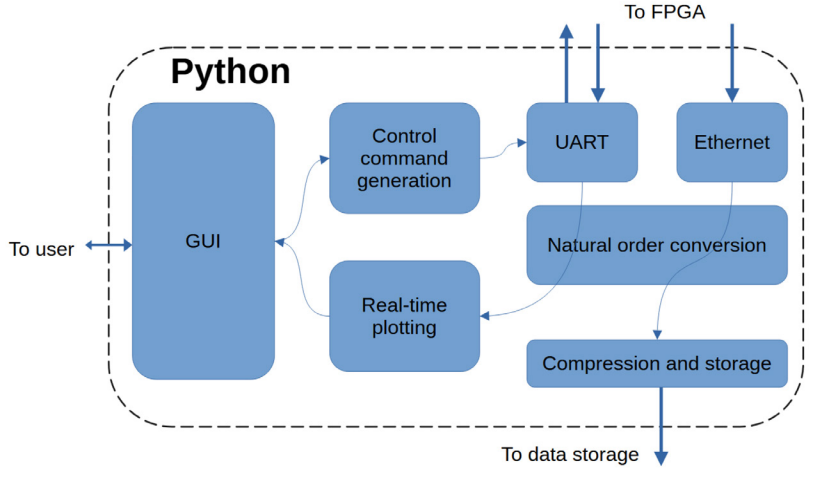

Fig. 2. Proposed software structure of the model

An important function of the proposed model's software is the reordering of FFT output into natural order, aimed at minimizing the use of FPGA hardware resources. As previously noted, the hardware optimization in the model lies in avoiding the use of memory buffers for converting FFT output into natural order, which is further complicated by filtering based on the observation range. Based on this range and the expected output order of the FFT, the software component restores the natural sample order after transmission via one of the interfaces. This makes it possible to universally optimize FPGA resource usage regardless of the FFT input data width or transform length. It is the high degree of integration between the software and hardware components of the proposed model that enables optimization of both FPGA resource usage and transmission latency.

3. Results and Discussion

3.1. Hardware Implementation

The development, debugging, and testing of the model were carried out according to an iterative-incremental methodology, which involves step-by-step and parallel implementation, with each stage adding new functionality to the already developed part of the system. Overall, the process covered five key areas: hardware implementation, interface integration, software component development, preliminary calculations, experimental testing and analysis.

To demonstrate the proposed model, the Xilinx Artix-7 35t FPGA was selected as the hardware platform. It provides sufficient resources, supports modern interfaces and IP cores, and is cost-effective. The FPGA is part of the A7 Lite carrier board, which includes the required interface converters and AD9226 ADC (12-bit, 65 MSPS) was added to the setup. The FPGA design was developed using the Xilinx Vivado IDE, which supports simulation, synthesis, implementation, programming, and

ISSN-L 2664-9969; E-ISSN 2706-5448

debugging. To reduce development time and ensure stability, IP cores were used.

According to the proposed model, the Xilinx FFT 9.1 IP core was selected for performing the FFT. It features a pipelined architecture with a transform length of 16,384 samples and a butterfly structure based on radix-2 decimation-in-time. At the operating sampling frequency, the core accepts one input sample and produces one complex result per clock cycle, with a latency of N cycles from the first input sample to the first output result, where N is the FFT length. The computations are performed in fixed-point format without scaling, which must be taken into account during subsequent processing.

As part of the FFT post-processing in the proposed model, the signal spectrum is calculated as follows

$$|X_k| = \sqrt{\mathrm{Re}(X_k)^2 + \mathrm{Im}(X_k)^2}. \quad (2)$$

The amplitude squared is computed using Xilinx Multiplier 12.0 blocks, while square root calculation is implemented using the Xilinx CORDIC 6.0 IP core. The use of IP cores is justified by their flexibility in resource utilization and by the acceleration of design routing. The adder is implemented in SystemVerilog. These blocks introduce minimal latency and require few hardware resources, yet they are considerably more efficient and faster compared to software-based implementations.

For test signal generation, the design includes a sine wave generator based on the Xilinx CORDIC 6.0 core with a custom control system. This generator is convenient for use during debugging or as an additional signal source during testing.

It is also possible to access FFT results prior to spectrum calculation, or even raw unprocessed data. This is achieved by routing data to output interfaces from various stages of the model's hardware pipeline. Such flexibility simplifies hardware verification and tuning, and provides additional adaptability for integration with external applications.

The model was implemented on the FPGA using the Xilinx Vivado IDE. During implementation, optimizations were applied that did not affect the processing logic but ensured that the required timing constraints were met. The base clock frequency is 50 MHz, while the main processing blocks operate at a sampling frequency of 65 MHz, and the Ethernet subsystem runs at 125 MHz in accordance with the Gigabit Ethernet PHY RGMII standard [12]. Thus, the project was implemented using three clock domains 50, 65, and 125 MHz for different functional blocks, including FFT, UART interfaces, and Ethernet. Frequencies were selected based on the specifications of peripheral devices and internal logic. Resource constraints were taken into account during synthesis and were carefully analyzed. The Xilinx Vivado IDE supports optimization parameters that help accelerate development and implementation through partial compilation, elaboration, placement, and routing. The iterative development approach allowed for efficient utilization of FPGA hardware resources while reducing the time required for implementation and testing in each development cycle.

3.2. Communication Interface Integration

The model includes two primary communication channels between the hardware component and the host computer: UART and Ethernet. The UART interface is used for control and for transmitting decimated data for real-time plot during debugging. The Ethernet protocol is used to transmit complete computation results, with further compression and storage. Ethernet communication operates in point-to-point mode, minimizing the influence of external devices on the model evaluation process. Data transmission over Ethernet is performed using the UDP protocol without delivery confirmation. Therefore, the packet format includes an additional prefix containing service information that defines the sample indices in the transmitted packet and serves to ensure data integrity (Table 1). The packet number identifies the sequential number of the packet, which corresponds to the order of the performed FFT operation. The frame number is used to enumerate data when packet fragmentation occurs due to the inability to transmit the entire dataset in a single packet. This mechanism allows the receiver to reconstruct the full data sequence correctly, even if packets do not arrive in the original order. The sample indices field defines the range of spectral data indices contained in the current packet. This also serves as a synchronization check between the hardware and software components of the model, which is essential for reliable synchronization and correct interpretation of received data during real-time processing.

Control of the hardware component is performed via the UART protocol. Control words have a fixed structure, ensuring ease of implementation in hardware and high reliability. Table 2 presents the format of the 64-bit control word used to configure the operating modes of the processing and data transmission system in the FPGA. The word fields allow setting the signal source, the type of transmitted data, sine wave generator parameters, and the range of samples to be transmitted. Additionally, it provides control over start, stop, and data transmission modes via UART or Ethernet. For programming convenience and simple addition of new features, reserved fields are included in the control word.

Table 1

Prefix format for data transmission over Ethernet protocol

Byte	Functional description	Value range
0	Packet number	0–255
1	Frame number	0–31
2–3	Starting sample index of data in the packet	0–8191
4–5	Ending sample index of data in the packet	0–8191

Table 2

Control word format

Bit(s)	Functional description	Value range
0	Control of the hardware FSM for data transmission via UART and Ethernet (activated by bit 7). If bit 1 is set to 0, a single dataset of specified size will be transmitted	1 – start 0 – stop
1	Start continuous data transmission, it can only be stopped by issuing a stop command via bits 1 and 0	1 – start 0 – stop
2	Use an external (ADC) data source for FPGA. If bits 2 and 3 are both active, the data from both sources are halved and summed	1 – active 0 – inactive
3	Use internal FPGA data source (reference CORDIC generator). If bits 2 and 3 are both active, the data from both sources are halved and summed	1 – active 0 – inactive
6–4	Select type of data to transmit	111 – spectrum data (square root of sum of squares) 101 – real part of FFT result 001 – imaginary part of FFT result 000 – unprocessed data
7	Enable Ethernet transmission with further storage	1 – active 0 – inactive
11–8	Reserved for CORDIC generator amplitude control	–
15–12	Reserved with no specific function	–
31–16	CORDIC phase increment per clock cycle (defines the output frequency of the reference generator). Full phase rotation equals 51470 steps. Default = 800	1–25735
45–32	Index of the first sample to be retained for transmission. Samples with indices below this value will be filtered out. Default = 0	0–8191
47–78	Reserved for alignment	–
61–48	Index of the last sample to be retained for transmission. Samples with indices above this value will be filtered out. Default = 8191	0–8191
63–62	Reserved for alignment	–

Data transmitted via the UART interface is structured as a packet with a prefix that represents the sequence number of the data words and serves as additional control data for verifying the integrity of the control packet. Table 3 shows the prefix format added to each data packet during transmission over the serial interface to ensure correct reception and synchronization. The prefix includes patterns for identifying the beginning and end of the prefix, as well as specifying the range of samples contained in the packet. This ensures the correct transmission and processing of data on the receiving side.

Table 3

Prefix format for data transmission

Bytes	Functional Description	Value
0–1	Start bytes	0x9B, 0xFA
2–3	Index of the first sample in the packet	0–8191
4–5	Index of the last sample in the packet	0–8191
6–8	End bytes	0x0E, 0xBF, 0x9A

3.3. Software Implementation

The software for visualization and experiment control was implemented in Python, providing platform independence and rapid development. Only open-source libraries were used, such as numpy, matplotlib, threading, and gzip. The graphical design of the application was created using the PyQt6 library with the PyQt6 UI code generator, into which a plotting window and control elements were integrated, as shown in Fig. 3.

The application allows forming control commands, configuring the hardware, plotting data, and saving results. The control command is created by aggregating and concatenating values set in the graphical part of the software application. The application allows selecting input and output data, choosing which part of the processed data to transmit, enabling and configuring the network connection and the reference signal generator. The full list of control word fields is provided in Table 2. After the control word is generated, all necessary prefixes and postfixes are added, and it is transmitted via the UART protocol.

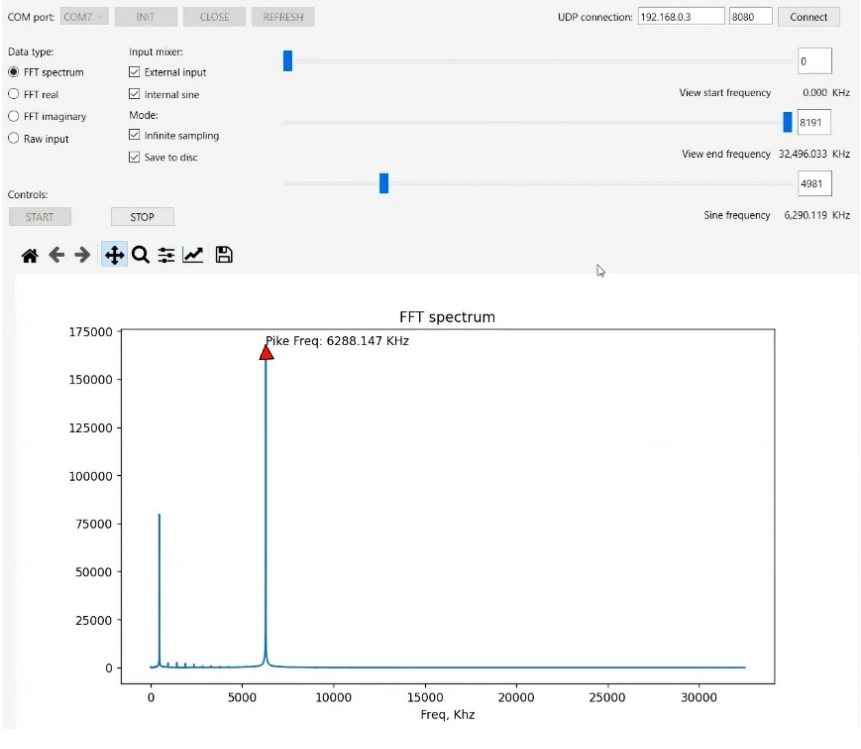

Fig. 3. Python application user interface

During the generation of the control word, information about the hardware configuration parameters is gathered by the software component. The software acts as the source of control information for filtering samples in the hardware according to the observation range. Based on this information, buffering, data integrity validation, and reordering of samples into natural order are performed.

Data is stored in binary form, in a format corresponding to the transmission protocol, namely 3 bytes per word. Since unprocessed data may also be saved, in such cases one word contains two 12-bit ADC samples. To save space, the data is compressed using open-source gzip library.

3.4. Preliminary Calculations

Based on the developed data format tables, it is possible to estimate the overhead introduced during network transmission of uncompressed data. In the proposed model, the maximum payload per packet is 256 words, each 3 bytes long. This requires adding the following service data: 7 bytes of physical preamble, 1 byte of start frame delimiter (SFD), 14 bytes of Ethernet header, 20 bytes of IPv4 header, 8 bytes of UDP header, and 6 bytes of prefix. Thus, to transmit 768 bytes of payload, a packet of 824 bytes is formed. The relative network overhead is approximately 7.3%, resulting in a total bandwidth usage of ~ 836 Mbps under a theoretical maximum of 1 Gbps, which falls within acceptable limits. This calculation does not include service packets at the network layer (e. g., ARP, SNMP, etc.).

Latency estimation is based on the following calculations. The transmission speed over the serial interface is 1.5 Mbaud or 150 kBps. One complete dataset contains 16,384 samples; therefore, the sampling, processing, and buffering time equals $16{,}384 \cdot 2 = 32{,}768$ cycles. At a sampling rate of 65 MSPS, this corresponds to $5.04 \cdot 10^{-4}$ s. Only the first Nyquist zone is transmitted, i. e., half of the full frame, with each sample occupying 3 bytes. Hence, the total amount of data for transmission is $8{,}192 \cdot 3 = 24{,}576$ bytes, and the transmission time over the serial interface is $24{,}576/150{,}000 = 0.164$ s. This demonstrates that processing time within the FPGA and software does not introduce significant delay, and the total display latency is approximately 0.17 s.

The transmission speed via the Ethernet interface is 1 Gbps. The size of the full sample frame is also 24,576 bytes, and the additional data required for transmission over the network introduces a 7.3% overhead. Therefore, the Ethernet transmission time is $24{,}576 \cdot 1.073/125 \cdot 10^6 = 2.1 \cdot 10^{-4}$ s, and the total storage delay is $2.1 + 5.04 = 7.14 \cdot 10^{-4}$ s. According to (1), the transmission delay reduction achieved by eliminating hardware buffering equals $16{,}384/65 \cdot 10^6 = 2.52 \cdot 10^{-4}$ s. The calculated delay parameters demonstrate the real-time capability of the implemented hardware-software model. Therefore, the system demonstrates the ability to operate in real time with minimal latency.

After completing all development stages in the Xilinx IDE, a detailed report on resource utilization can be generated. Fig. 4 shows the FPGA resource usage statistics after project synthesis and implementation. The implemented design uses FPGA resources efficiently, staying within acceptable limits. This is partly achieved by using the bit-reversed FFT output order, which eliminates the need for memory used to convert data into natural order. The highest utilization is observed in DSP blocks (70%) and BRAM (50%), which is

ISSN-L 2664-9969; E-ISSN 2706-5448

typical for systems performing intensive digital signal processing. Other resources remain available with a noticeable margin.

Resource	Utilization	Available	Utilization %
LUT	5104	20800	24.54
LUTRAM	1009	9600	10.51
FF	8864	41600	21.31
BRAM	25	50	50.00
DSP	63	90	70.00
IO	33	250	13.20
BUFG	6	32	18.75
MMCM	2	5	40.00

Fig. 4. FPGA resource usage

The 50% BRAM usage is due to its role in storing intermediate FFT results and buffering during UART and Ethernet transfers. DSP blocks are utilized at 70% because they are used in FFT and multiply operations, but they could be replaced with LUTs without affecting performance or timing. In this implementation, their use is justified as it optimizes routing and accelerates implementation. In other words, the high utilization of these blocks is not critical.

Power consumption and thermal characteristics were evaluated using tools integrated into the IDE. Fig. 5 shows the estimated power consumption and thermal behavior of the FPGA during operation. The results indicate that the FPGA operates in an energy-efficient mode with low thermal load. The power consumption of 0.535 W is moderate, and the operating temperature of 58.5°C ensures stable operation without overheating, even without active cooling.

Total On-Chip Power:	**0.535 W**
Junction Temperature:	**26,5 °C**
Thermal Margin:	58,5 °C (20,7 W)
Effective ϑJA:	2,8 °C/W

Fig. 5. Estimated power consumption of the FPGA

According to the detailed power distribution (Fig. 6), the transition from idle to active mode increases power consumption only due to Logic, DSP, Signals, and BRAM blocks. The maximum estimated increase is up to 164 mW, which is a favorable indicator for the target application and demonstrates efficient utilization of hardware resources.

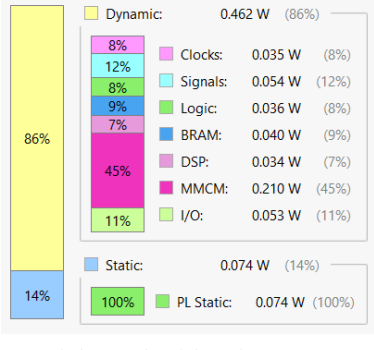

Fig. 6. Detailed power breakdown by FPGA components

To evaluate the accuracy loss, it is necessary to determine the bit width of the numbers used in calculations. The number format (fixed-point or floating-point) does not affect the result itself. At the input, 12-bit data from the ADC is obtained. The FFT length is 16,384 or 2^{14}. This means that after the transformation, the bit width of the real and imaginary parts of the complex result increases by 14 bits, resulting in 26 bits. The IP core includes one additional overflow protection bit, resulting in a 27-bit output. This bit width does not correspond to an integer number of bytes, which complicates data transmission and processing. The nearest byte-aligned size is 24 bits, which requires rounding.

The error introduced by rounding the real and imaginary parts of the complex result was evaluated. Discarding 3 least significant bits introduces a new quantization step of $2^3 = 8$. The resulting quantization error is equal to half the step size

$$\Delta_1 = \frac{2^3}{2} = 4. \quad (3)$$

The signal magnitude is defined as the square root of the sum of the squares of the real and imaginary parts

$$M = \sqrt{\mathrm{Re}^2 + \mathrm{Im}^2}. \quad (4)$$

Squaring doubles the bit width and adds 1 overflow protection bit, giving

$$2n+1 = 2 \cdot 24 + 1 = 49. \quad (5)$$

The sum of two such numbers add 1 more bit, yielding $n = 50$ bits. After square root extraction, the result has the following bit width

$$\frac{n}{2} + 1 = 25 \text{ bit}. \quad (6)$$

Discarding 1 least significant bit after square root computation gives a quantization step of $2^1 = 2$. Accordingly, the quantization error is

$$\Delta_2 = \frac{2^1}{2} = 1. \quad (7)$$

To calculate the total error across stages, the square root sum of the squares is used

$$\Delta_{total} = \sqrt{\Delta_1^2 + \Delta_2^2} = \sqrt{17} \approx 4.1231. \quad (8)$$

The maximum signal value for 25-bit resolution is

$$A_{\max} = 2^{24} = 16777216. \quad (9)$$

Relative error

$$\varepsilon = \frac{\Delta_{total}}{\Delta_{\max}} \cdot 100 = \frac{4.1231}{16777216} \cdot 100 \approx 2.457 \cdot 10^{-5}\%. \quad (10)$$

To evaluate the accuracy in decibels, the signal-to-noise ratio (*SNR*) formula is used

$$SNR = 20\log_{10}\left(\frac{\Delta_{\max}}{\Delta_{total}}\right) = 20\log_{10}\left(\frac{16777216}{4.1231}\right) \approx 132.2 \text{ dB}. \quad (11)$$

3.5. Testing, Measurement, and Analysis

Testing of the implemented hardware-software model was performed. Test signals originated from the internal FPGA generator, the external generator via the ADC input, or a combination of both. The externally applied signals met the ADC's input specifications. To approximate real application conditions, a stable generator with signal modulation capability was used. The tests demonstrated the functionality of the implemented hardware-software model.

In the demonstrated test, a discrepancy was observed between the set frequency of the generated signal and the frequency detected in the spectrum (Fig. 7). Specifically, the generator produced a signal with a frequency of 6290.119 kHz, while spectral analysis showed a value of 6288.147 kHz. The corresponding error was 1.972 kHz.

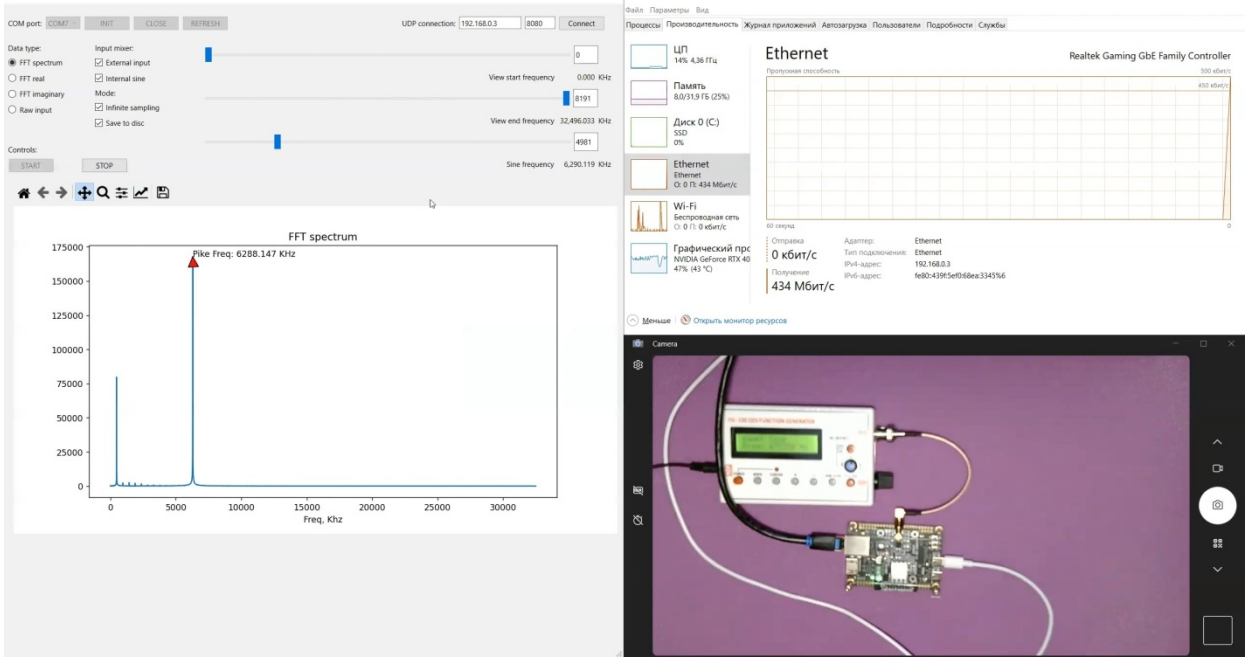

Fig. 7. Screenshot during testing

The identified deviation is reasonable considering the technical specifics of the spectral analysis implementation. The calculated width of one frequency bin is 65 MHz/16,384 = 3.97 kHz. Thus, the 1.972 kHz error corresponds to half the resolution of the implemented system. Despite the deviation, the result is considered acceptable within the accuracy limits of the chosen method.

An evaluation of the system's compliance with real-time operation conditions was carried out. Since all data is processed by streaming FFT without losses, the assessment of real-time performance should distinguish between display latency and full data transmission latency. Display latency depends on the speed of the serial interface. The latency of full lossless data transmission is affected by the speed of the network interface according to preliminary calculations.

Power consumption was measured using the ATORCH A3 device, and the results were graphically recorded (Fig. 8). It is not possible to separate power consumption by carrier board components (FPGA, ADC, network controller, etc.), so the measurements were taken in idle and active modes, with and without Ethernet connection.

a *b* *c* *d*

Fig. 8. Power consumption measurement results:
a – idle mode without Ethernet connection; *b* – active mode without Ethernet connection; *c* – idle mode with Ethernet connection; *d* – active mode with Ethernet connection

According to the measurements, the active operation of the FPGA increases power consumption by 59–69 mW, which corresponds to the previously performed theoretical estimation and confirms the correctness of the calculations obtained using the IDE tools.

Based on the obtained results, it is appropriate to conduct a comparative analysis with modern studies aimed at improving FFT applications. In particular, compared to [2], the proposed model and its hardware-software implementation feature a longer transform length of 16,384 samples, which provides higher spectral analysis accuracy of 3.97 kHz in the presented implementation. The model demonstrates a significant advantage in hardware power consumption, ranging from 1.4 to 1.9 W depending on the mode. At the same time, it is necessary to consider both the broader functionality of the model and the fact that this value already includes the ADC, FPGA, and other components of the carrier board, making it impossible to separate the power consumption of individual components, which limits the precision of evaluating the impact of the implemented solutions on power usage. Meanwhile, the display and storage latencies are 0.17 s and 0.74 μs, respectively, which are higher compared to the solution in [2], which was specifically optimized for minimal latency. The study [3] also uses a shorter transform, and under scaling conditions, the resource efficiency is comparable. However, the power consumption of that solution is approximately half that of the proposed universal model. The works [1–11] do not implement real-time result transmission and do not evaluate the impact of rounding on computational accuracy, which in this study reaches 132 dB and represents a significant aspect. One of the key advantages of the proposed model is software integration, which allows halving the required FPGA memory resources and enables support for more complex spectral analysis algorithms that require the flexibility of software implementations.

The model has potential for further development and improvement. In the future, scaling is possible both horizontally (increasing the number of channels) and vertically (increasing FFT length, accuracy, speed, etc.). One of the current limitations is the use of an FFT IP core that does not provide amplitude scaling control, which complicates the processing of weak signals in high dynamic range scenarios and also requires refinement. Adaptive processing, power-saving modes using variable clocking of individual blocks, and functional expansion for real-time analysis of multiple data channels can be introduced. Horizontal scaling opens the possibility of implementing cross-spectral signal analysis. At the same time, it should be noted that increasing the number of channels at the same data width significantly increases the amount of transmitted data, and the impact of this factor on system efficiency and further development requires additional research. However, such changes can significantly expand the application areas of the model, enabling the implementation of more tasks and integration into a wider range of modern applications. In this context, it is appropriate to consider optimizing methods for correlation analysis with attention to resource efficiency, as well as the potential for scaling by combining multiple model instances into a unified system.

ISSN-L 2664-9969; E-ISSN 2706-5448

Particular attention should be given to FPGA memory resources, as they may limit both the transform length and computation accuracy during further development. Increasing the transform length or number of channels involves several challenges that require careful planning. For example, doubling the transform length may bring the utilization of certain FPGA resources close to 100%, which is related both to the limited on-chip memory and potential complications during implementation and routing. Overcoming these challenges would increase the model's suitability for autonomous systems requiring high-precision real-time digital signal processing.

The current implementation complies with the design principles and uses under 50% of FPGA resources, enabling additional modifications if needed. In its current form, the model is suitable for real-time single-channel signal analysis with the ability to store data for further processing, and for use in projects where processing time is critical. It is suitable for implementing digital signal processing algorithms, including adaptive methods such as MUSIC or ESPRIT, which are more appropriately executed in a software environment. Thanks to the use of standard interfaces and reliable data transmission and control protocols, the model can be easily integrated into computer-based signal analysis systems, including those capable of running Python software.

Thus, the proposed adaptive architecture optimizes hardware resource usage and the latency of result transmission for real-time operation while maintaining the required accuracy and low power consumption. This is achieved through the combination of a hardware pipeline structure with software integration and dynamic use of UART/Ethernet interfaces, extending the theoretical foundations of digital signal processing for applied systems.

4. Conclusions

The hardware-software model for real-time signal spectrum computation using the Fast Fourier Transform has been proposed, implemented, and tested. The use of a hardware pipelined architecture enabled efficient real-time processing. The software part provided convenient control, visualization during configuration, and flexibility of software-based solutions. Tight hardware-software integration allows for optimal FPGA resource utilization, reduced latency in transmitting processed signals, and makes the proposed model a promising platform for executing frequency-domain analysis algorithms with hardware acceleration. Thus, the model addresses key challenges of digital signal processing: ensuring high performance, computational accuracy, energy efficiency, and optimal resource utilization.

The complete development cycle was carried out: the FPGA-based hardware platform was selected, the hardware was developed using SystemVerilog, the supporting software was created in Python, and data exchange protocols between system components were implemented. Functional simulation and implementation were conducted using the IDE, allowing evaluation of resource usage and power consumption. Accuracy loss was assessed, demonstrating the benefits of hardware-software optimization and the appropriateness of rounding for effective real-time implementation. Experimental testing confirmed the correct functioning of the system and its ability to process signals in real time without data loss. Low data transmission latency and high throughput were achieved, further confirming the efficiency of the proposed solution.

The developed model is integrable and scalable, with support for increasing transform length, adding new channels, or expanding functionality (e. g., cross-spectral analysis) without major changes to the architecture. It also supports integration with other software systems. The study advances the theoretical foundations of digital signal processing by combining pipelined architecture with adaptive control and implementing a hardware-software compromise, complemented by a comprehensive analysis of experimental results. The proposed model enables efficient processing of dynamic signals in telecommunications, radar, and autonomous control applications. The results obtained are of both theoretical and practical value and can be integrated into other signal processing projects or used as a basis for further research in the field of digital signal processing.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this research, including financial, personal, authorship, or any other nature that could affect the research and its results presented in this article.

Financing

The research was conducted without financial support.

Data availability

The manuscript has no associated data.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the presented work.

References

1. Kaya, Z., Garrido, M., Takala, J. (2023). Memory-Based FFT Architecture With Optimized Number of Multiplexers and Memory Usage. *IEEE Transactions on Circuits and Systems II: Express Briefs, 70 (8),* 3084–3088. https://doi.org/10.1109/tcsii.2023.3245823
2. Kaya, Z., Garrido, M. (2023). Low-Latency 64-Parallel 4096-Point Memory-Based FFT for 6G. *IEEE Transactions on Circuits and Systems I: Regular Papers, 70 (10),* 4004–4014. https://doi.org/10.1109/tcsi.2023.3298227
3. Yang, C., Wu, J., Xiang, S., Liang, L., Geng, L. (2023). A High-Throughput and Flexible Architecture Based on a Reconfigurable Mixed-Radix FFT With Twiddle Factor Compression and Conflict-Free Access. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 31 (10),* 1472–1485. https://doi.org/10.1109/tvlsi.2023.3298943
4. García-Astudillo, L. A., Lindoso, A., Entrena, L., Martín, H., García-Valderas, M. (2021). Error sensitivity study of FFT architectures implemented in FPGA. *Microelectronics Reliability, 126,* 114298. https://doi.org/10.1016/j.microrel.2021.114298
5. Xie, Y., Chen, H., Zhuang, Y., Xie, Y. (2023). Fault Classification and Diagnosis Approach Using FFT-CNN for FPGA-Based CORDIC Processor. *Electronics, 13 (1),* 72. https://doi.org/10.3390/electronics13010072
6. Changela, A., Zaveri, M., Verma, D. (2020). FPGA implementation of high-performance, resource-efficient Radix-16 CORDIC rotator based FFT algorithm. *Integration, 73,* 89–100. https://doi.org/10.1016/j.vlsi.2020.03.008
7. Dang, T.-H., Tran, V.-N., Nguyen, L.-C. (2023). A parallel rotator for FFT/IFFT applied in multi-carrier wireless communication systems. *Digital Signal Processing, 141,* 104190. https://doi.org/10.1016/j.dsp.2023.104190
8. He, J., Bao, Z., Li, H., Li, Q., Li, Z., Liu, P. et al. (2023). Implementation of an Adaptive Trapezoidal Shaping Method Based on FFT. *IEEE Transactions on Nuclear Science, 70 (9),* 2234–2239. https://doi.org/10.1109/tns.2023.3307475
9. Song, J., Li, Y., Qiu, J., Hong, X., Guo, H., Yang, Z. et al. (2023). Low-Complexity FPGA Implementation of 106.24Gbps DP-QPSK Coherent Optical Receiver With Fractional Oversampling Rate Based on One FIR Filter for Resampling, Retiming and Equalizing. *Journal of Lightwave Technology, 41 (16),* 5244–5251. https://doi.org/10.1109/jlt.2023.3258072
10. Yao, Y., Zhang, J., Liu, Y., Ruan, T., Li, W., Lian, H. et al. (2023). Development of a multifunctional real-time data processing system for interferometers on EAST. *Journal of Instrumentation, 18 (11),* C11013. https://doi.org/10.1088/1748-0221/18/11/c11013
11. Li, C. J., Li, X., Lou, B., Jin, C. T., Boland, D., Leong, P. H. W. (2023). Fixed-point FPGA Implementation of the FFT Accumulation Method for Real-time

Cyclostationary Analysis. *ACM Transactions on Reconfigurable Technology and Systems, 16 (3),* 1–28. https://doi.org/10.1145/3567429

12. *Reduced Gigabit Media Independent Interface (RGMII)* (2020). Texas Instruments Incorporated. Available at: https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/138/6661.RGMIIv1_5F00_3.pdf

Oleksandr Vasyliev, *PhD Student, Department of Design Automation, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, ORCID: https://orcid.org/0009-0009-9975-2942*

✉***Oleh Filippenko,*** *PhD, Associate Professor, Department of Infocommunication Engineering V. V. Popovsky, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, e-mail: oleh.filippenko@nure.ua, ORCID: https://orcid.org/0000-0003-4616-250X*

Inna Filippenko, *PhD, Associate Professor, Department of Design Automation, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, ORCID: https://orcid.org/0000-0002-3584-2107*

Oleksandr Shkil, *PhD, Associate Professor, Department of Design Automation, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, ORCID: https://orcid.org/0000-0003-1071-3445*

✉*Corresponding author*