DOI: 10.35598/mcfpga.2026.032

FPGA-Based Information Coding System in Noisy Channels with Increased Reliability

Ruslan Zub
ORCID 0009-0004-2650-3498
Computer Systems and Networks Department
National University "Zaporizhzhia Polytechnic"
Zaporizhzhia, Ukraine
zub2004@gmail.com

Oleksii Klymakov
ORCID 0009-0004-8704-8185
Computer Systems and Networks Department
National University "Zaporizhzhia Polytechnic"
Zaporizhzhia, Ukraine
alexklimakov2003@gmail.com

Iryna Zeleneva
ORCID 0000-0002-4042-4540
Computer Systems and Networks Department
National University "Zaporizhzhia Polytechnic"
Zaporizhzhia, Ukraine
irina.zeleneva@gmail.com

Svitlana Hrushko
ORCID 0000-0002-0064-408X
Computer Systems and Networks Department
National University "Zaporizhzhia Polytechnic"
Zaporizhzhia, Ukraine
grushkosveta@gmail.com

***Abstract*—Today, digital devices and the Internet of Things (IoT) are widely integrated into daily operations. Consequently, ensuring reliable data transmission over noisy communication channels has become a critical task. Standard software solutions often process data too slowly to correct errors in real time. This paper presents the development of a fast and highly reliable hardware system for coding and decoding data. The solution is implemented using Field-Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs). The system utilizes parallel convolutional encoding and Viterbi decoding. To achieve fault tolerance, Triple Modular Redundancy (TMR) combined with an adaptive majority controller was integrated into the design. Furthermore, a comparative analysis of hardware resource usage was conducted across various Altera/Intel chips (MAX II, MAX V, Cyclone III, Cyclone IV, and Cyclone V). The results demonstrate that simple encoders operate efficiently on very low-cost CPLDs, whereas complex decoders require the larger structural capacity of FPGAs, such as the Cyclone IV.**

Keywords—fault tolerance, Viterbi decoder, convolutional coding, majority voting, communication channel, CPLD, FPGA, VHDL, Quartus II.

I. INTRODUCTION

Modern society is rapidly advancing toward Industry 4.0, Industry 5.0, and the Internet of Things (IoT). Due to this trend, new and strict requirements are established for the design of electronic devices. One of the major challenges in building robust digital systems is ensuring that data is transmitted safely and correctly, even when communication channels are affected by heavy noise and interference [1].

When data is sent over the air or through cables, background noise can flip the bits (changing a 0 to a 1, or a 1 to a 0). If these errors are not corrected, critical failures may occur. For instance, a corrupted data packet can stop an industrial robot, provide incorrect readings on medical life-support equipment, or lead to a loss of control in aerospace systems. According to Shannon's noisy-channel coding theorem these errors can be successfully corrected if extra, mathematically calculated bits are added to the message before transmission [2].

This process is known as Forward Error Correction (FEC). One of the most effective FEC methods is the application of convolutional coding on the transmitter side and Viterbi decoding on the receiver side. This approach is highly efficient at detecting and correcting errors [3, 4].

However, performing Viterbi decoding with standard software (such as running a program on a regular processor) consumes too much time, as it requires a large number of mathematical operations. Modern networks demand data to be processed instantly, in real time. Software execution simply cannot meet these high-speed requirements [5].

To overcome the limitations of software, hardware acceleration is widely utilized. The coding logic is built directly into microchips called Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs). Hardware operates significantly faster because it can process multiple parts of the data simultaneously. It also provides predictable, extremely low processing delays [6].

Furthermore, some devices operate in very harsh environments, such as high radiation zones in space or high-temperature industrial factories. Under such conditions, the hardware itself can experience glitches. To prevent total system failure, additional backup logic must be built directly into the hardware architecture.

The main goal of this research is to design and test a fault-tolerant hardware data transmission system. The system was implemented using VHDL code for FPGAs and CPLDs. It utilizes three parallel convolutional encoders and three

Viterbi decoders. The accuracy of the processed data is ensured by implementing Triple Modular Redundancy (TMR) and an adaptive majority controller, which guarantees correct data output even if an entire communication channel fails.

II. System Architecture and Design

The proposed communication system consists of two main hardware parts. The first part is the transmission block, which contains the encoders. The second part is the receiving block, which contains the decoders and the majority controller.

To provide maximum protection against channel noise and hardware glitches, the system is designed based on the Triple Modular Redundancy (TMR) principle. This means the same data processing operation is performed three times simultaneously, and the results are compared.

A. Transmission Block (Convolutional Encoders)

In standard communication systems, a transmitter usually sends a single stream of encoded data. In the proposed fault-tolerant system, the transmission block duplicates the original input data and directs it into three separate, parallel convolutional encoders.

To evaluate how different codes perform, the three encoders were not made identical. Instead, three different non-systematic convolutional encoders were designed using VHDL:

- Encoder 1 has parameters (2, 1, 4). This means for every 1 input bit, 2 output bits are generated (code rate 1/2). Its memory length is 4;
- Encoder 2 has parameters (2, 1, 5). The code rate is 1/2, but it has a longer memory length of 5. This provides slightly better error correction at the cost of minor added complexity;
- Encoder 3 has parameters (3, 1, 3). For every 1 input bit, 3 output bits are generated (code rate 1/3). The memory length is 3.

The internal structure of these encoders is straightforward. Each encoder consists of a basic shift register (a chain of D-flip-flops) connected to several modulo-2 adders (logic XOR gates). The XOR gates combine the current data bit with previous data bits to generate the new parity symbols.

B. Receiving Block (Viterbi Decoders and Controller)

The receiving block is significantly more complex than the transmitter. It receives three separate streams of encoded bits from the noisy communication channels.

Each stream is directed into its own dedicated Viterbi decoder. Decoder 1 processes the (2,1,4) stream, Decoder 2 processes the (2,1,5) stream, and Decoder 3 processes the (3,1,3) stream. The Viterbi algorithm analyzes the incoming bits and calculates the "Hamming distance"–a metric used to count how many transmission errors might have occurred. It constructs a map of possible original messages (a state trellis) and traces the most likely path to recover the correct original data.

After the three decoders complete their operation, their outputs are sent to the Adaptive Majority Controller (AMC). The AMC evaluates the three decoded bits simultaneously:

If all three bits are identical (for example, 1-1-1), the AMC confirms the data is perfect and passes it to the output.

If one channel is heavily corrupted by noise and produces an incorrect bit (for example, 1-1-0), the AMC applies a "2-out-of-3" voting rule. It determines that '1' is the majority, ignores the corrupted '0', and outputs the correct '1'.

Additionally, the AMC is designed to be adaptive. If it detects that a specific channel continuously provides incorrect answers, a status warning signal is generated. This feature allows system engineers to identify exactly which channel or sensor requires maintenance.

III. Hardware Implementation and Resource Analysis

Before deploying the system in a practical application, it was necessary to determine the hardware costs and select the appropriate microchip. To achieve this, the VHDL code was compiled and synthesized using Altera Quartus II software.

The design was tested on five different chips from various price categories and technological generations:

- CPLD MAX V (approximately $1.25) [7];
- CPLD MAX II (approximately $16.50) [8];
- FPGA Cyclone IV (approximately $27.80) [9];
- FPGA Cyclone III (approximately $44.00) [10];
- FPGA Cyclone V (approximately $54.00) [11];

The number of used Logic Elements (LE) or Adaptive Logic Modules (ALM) was carefully analyzed for each chip.

A. Encoder Implementation Results

When the transmitting part (the encoders) was compiled, it was found that the hardware requirements are incredibly low
flop

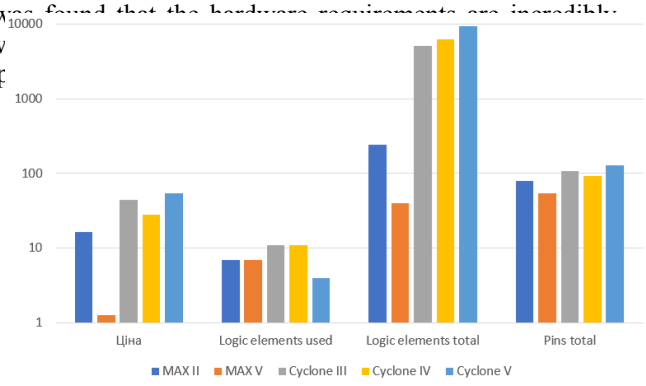

Fig. 1. Summary comparison of hardware costs and FPGA cost for the implementation of the transmitting part (encoders).

As shown in Fig. 1, implementing the encoders requires almost no physical space on the chip. Regardless of the chosen architecture, the three encoders combined use only between 4 and 11 logic elements. This represents less than 1% of the total capacity of even the smallest tested chips.

Because of this efficiency, purchasing an expensive FPGA (such as a Cyclone III or Cyclone V, costing over $40) solely for the encoders is economically unjustified. Instead, the most affordable ultra-budget CPLD, such as the MAX V priced at $1.25, fulfills the requirements perfectly. The older MAX II is also technically suitable, though less cost-effective.

B. *Decoder and System Implementation Results*

The situation is entirely different for the receiving block. The three Viterbi decoders and the AMC controller involve inte mei

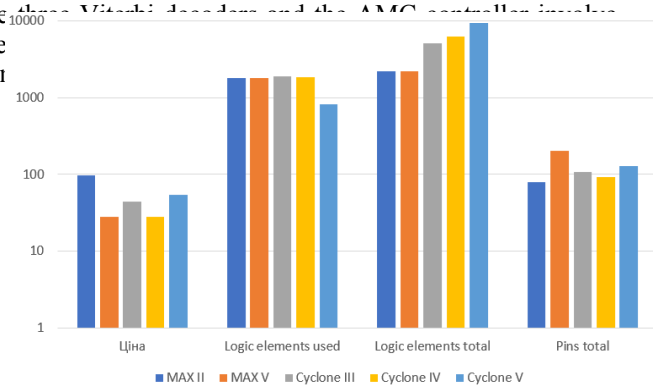

Fig. 2. Summary comparison of parameters and implementation results of the receiver on FPGAs (logarithmic scale).

Fig. 2 illustrates that the receiver consumes a massive amount of hardware resources. If an attempt is made to implement the receiver on the budget CPLD MAX V, it consumes 1797 out of 2210 logic elements, meaning the chip is 81% full. While the system functions correctly, the chip is operating near its absolute limit. There is no capacity left to add other features or upgrades in the future. The older CPLD MAX II yields the identical resource usage but at a disproportionately high financial cost.

Therefore, for the receiver, FPGAs are strictly required. The Cyclone IV FPGA accommodates the entire receiving system easily, utilizing only 1854 logic elements. This accounts for just 30% of the chip's total volume, leaving 70% completely free for future system expansions. The older Cyclone III also provides sufficient space but is less cost-effective. The newest device, the Cyclone V , which utilizes an advanced ALM architecture, required only 820 ALMs (9% of its capacity). However, the Cyclone V represents the most expensive option among the tested devices.

IV. Conclusion

The developed fault-tolerant data transmission system successfully demonstrates the capability to automatically correct errors and reconfigure the communication path even in the event of a total failure of one of the channels.

The comparative hardware research confirms that different components of communication systems require fundamentally different hardware platforms. For simple, repetitive tasks like convolutional encoding, the use of ultra-budget CPLDs (such as Altera MAX V) provides the optimal economic and technical solution. However, for implementing complex mathematical algorithms like Viterbi decoding combined with hardware redundancy, FPGAs are mandatory. The Cyclone IV FPGA proved to be the most balanced choice for the receiver, offering a cost comparable to high-end CPLDs while maintaining a 70% resource margin for integrating additional system functionalities, aligning perfectly with the scalable demands of modern IoT and smart industrial systems.

References

[1] M. Zhan et al., "Channel Coding for High Performance Wireless Control in Critical Applications: Survey and Analysis," IEEE Access, vol. 6, pp. 29648–29664, 2018. [Online]. Available: https://www.academia.edu/87095486/Channel_Coding_for_High_Performance_Wireless_Control_in_Critical_Applications_Survey_and_Analysis.

[2] P. W. Shor, "Shannon's Noisy Coding Theorem," lecture notes, Massachusetts Institute of Technology. [Online]. Available: https://math.mit.edu/~shor/18.310/noisy-coding-notes.pdf.

[3] V. R. Kulkarni et al., "Performance Analysis of Channel Coding Techniques in Wireless Communication," Int. J. Sci. Res. Sci. Eng. Technol. [Online]. Available: https://ijsrset.com/paper/8191.pdf.

[4] Digital cellular telecommunications system (Phase 2+); Channel coding, ETSI EN 300 909 V6.2.0, Apr. 1998. [Online]. Available: https://www.etsi.org/deliver/etsi_en/300900_300999/300909/06.02.00_40/en_300909v060200o.pdf.

[5] S. K. Singh et al., "Implementation of Viterbi Decoder," Int. J. Appl. Inf. Syst. [Online]. Available: https://research.ijais.org/xplore/number1/xplore1306.pdf.

[6] S. Brown and J. Rose, "Architecture of FPGAs and CPLDs: A Tutorial," IEEE Des. Test Comput., vol. 13, no. 2, pp. 42-57, 1996. [Online]. Available: https://www.eecg.utoronto.ca/~brown/papers/apps.pdf.

[7] Altera Corporation, MAX V Device Handbook, Volume 1. [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/6165/mv5v1_01.pdf.

[8] Altera Corporation, MAX II Device Handbook, Volume 1. [Online]. Available: https://datasheet.octopart.com/EPM240T100C4N-Altera-datasheet-7274037.pdf.

[9] Altera Corporation, Cyclone IV Device Handbook, Volume 3. [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/2491/Cyclone%20IV%20Device%20Handbook%2C%20Volume%203_Web.pdf.

[10] Altera Corporation, Cyclone III Device Handbook, Volume 1. [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/602/Cyclone_III_Device_Family_Overview.pdf.

[11] Altera Corporation, Cyclone V Device Handbook. [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/607/Cyclone_V_Device_Overview_Web.pdf.