EMERGING TECH CONFERENCE – Edge Intelligence

Volume 03, 2024, Page 81 – 84

Proceedings of Emerging Tech Conference:
Edge Intelligence 2024

A Low Power FPGA Implementation of LDPC Encoder for Space Applications

Christos Sidiras and Vasilis F. Pavlidis
Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Thessaloniki, Greece
chrisidi@ece.auth.gr, vpavlid@ece.auth.gr

Abstract

The emergence of 5G, 6G technologies and the expanding Internet of Things (IoT) necessitate advanced data encoding techniques, with Low-Density Parity-Check (LDPC) codes being particularly notable for the high error correction efficiency and low computational complexity. An LDPC encoder with a 4/5 data rate, as specified in CCSDS standards is developed in this work, tailored for transmitting scientific data from a nano-satellite to a ground station. Implemented in SystemVerilog and designed for FPGA deployment, the encoder employs a streaming architecture and the AXI4 Stream protocol for communication interfaces. The resulting solution aims lower power consumption while demonstrating good resource utilization and throughput performance.

Keywords: LDPC code, space communication, quasi-cyclic LDPC encoder, FPGA implementation, power reduction.

1 Introduction

The advent of 5G and 6G technologies and the increasing proliferation of IoT, require all types of telecommunication modulation techniques and data encoding, thereby escalating the demand for efficient data encoding methods. One of these methods is LDPC codes for which there is extensive literature, covering various aspects of their design, implementation, and optimization. LDPC codes exhibit their advantages over other code classes, and are, therefore, integrated in many communication standards, such as 10GBase-T Ethernet (IEEE 802.3an), Digital Video Broadcasting (DVB-S2) and standard for Wireless LANs (IEEE 802.11n). These codes are known for their ability to perform near the Shannon limit, providing excellent error correction capabilities with relatively low complexity [1]. As a result, LDPC codes are highly favored in applications that require high reliability, particularly in space applications. The implementation presented in this paper is part of the telecommunication subsystem of the Acubesat satellite developed as part of the European Space Agency's "Fly Your Satellite! 3" program.

2 Low Density Parity Check codes

LDPC codes are a type of block error-correcting codes introduced by Robert Gallager in the early 1960s. These codes are characterized by a sparse (low percentage of “1”) parity-check matrix H, which represents a set of parity-check equations that valid codewords must satisfy, thereby ensuring $H * c^T = 0 \ (mod\ 2)$

for any valid codeword c. The generator matrix G, derived from H, is used to produce the codewords, such that $c = u \cdot G$, where u is the input message. Regular LDPC codes are defined by the parameters (w_r, w_c), where w_r is the number of ones in each row and w_c is the number of ones in each column of the parity-check matrix H [**2**].

The sparsity of the parity-check matrix H is essential for facilitating efficient encoding and decoding processes. Iterative decoding algorithms, such as belief propagation, leverage this sparsity to identify and correct errors with minimal computational complexity. Deriving the generator matrix G from H is a common method for encoding LDPC codes. Alternative encoding methods [**3**] include systematic encoding techniques that directly construct codewords to satisfy the parity-check equations

3 LDPC Encoder Design

The object is to design a robust and fully functional LDPC encoder with a data rate of 4/5, as defined in CCSDS standards [**4**]. The encoder exclusively provides the scientific data of the transfer from a nano-satellite experiment to the ground station. The intellectual property core (IP) to be extracted follows a streaming architecture, making use of the AMBA AXI4-Stream protocol [**5**] for its interfaces. Developed in SystemVerilog, the encoder's IP core is intended for deployment on a Field Programmable Gate Array (FPGA).

3.1. LDPC Design Specifications

Space communication protocol recommends using two classes of LDPC codes. One class is AR4JA codes which offers a variety of block lengths (1,024, 4,096 and 16,384 bits) and rates (1/2, 2/3, 4/5) and exhibit good performance in terms of $Eb/N0$, emphasizing reliable communication in noisy environments. The other class is $C2$ code, which is a regular (8,176, 7,156) LDPC code that achieves high transmission speeds.

Parity check matrix of the selected AR4JA code with 4,096 block size and 4/5 rate is a quasi cyclic matrix consists of 512 × 512 sparse submatrices, such as identity, zero or permutation matrices all strictly implied by CCSDS standard. Generator matrix G produced by the aforementioned parity check matrix has the form of $G = [I_{4096}\ W]$, where I_{4096} is the square identity matrix of dimension equal to 4,096 and W is a dense 4,096 × 1,024 matrix of circulants each one of size 128 × 128. This type of LDPC codes can be assumed to be systematic, meaning that the first 4,096 bits of information encoded are the same as the input bits while next 1,024 bits are the parity ones. Apart from block length and information rate, detailed specifications are listed in Table 1.

3.2. Design Process

The IP core must have a specific set of interface signals in order to be compliant with AXI4 Stream protocol. A slave interface accepting the message and a master interface exporting the codeword are designed and only the signals that attribute to the handshake are used, as depicted in Figure 1(A). The provided Figure 1(B) represents a Recursive Convolutional Encoder (RCE) recommended by the standard. A detailed description of the encoder operation follows:

- First row bits of generator matrix, stored in memory, are loaded in registers from right to left. These bits are multiplied with the first single bit of the message and the result is stored in registers inside RCEs.
- The second bit of the message enters sequentially the encoder and is multiplied with the first-row

bits of generator matrix. The result is added exclusively with the previously stored one. This process is repeated for 128 cycles and for eight consecutive times.

- Every 128 passing cycles, the next row circulants of the generator matrix are loaded.
- When all 4,096 bits of the message have been processed, the encoder proceeds to the parity phase. The memory is disabled, demuxes change their output and the encoder starts executing a right shift operation for 1,024 cycles extracting calculated parity bits.

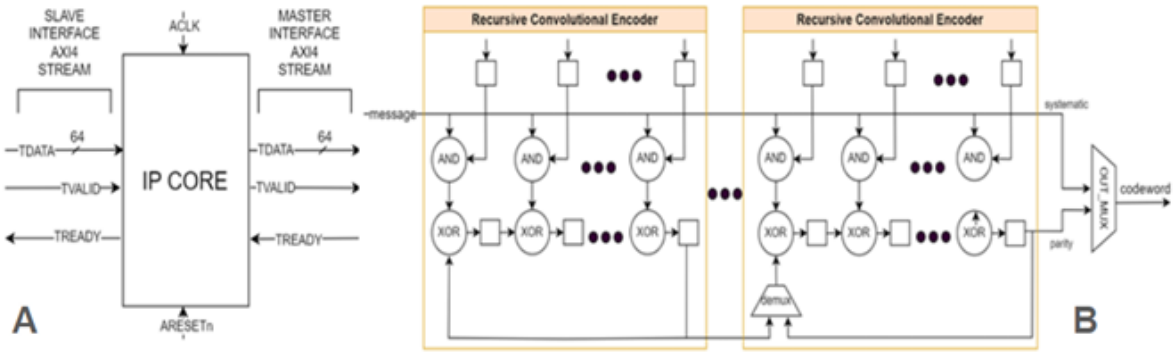

Figure 1: LDPC encoder (A) interface block diagram of the encoder, (B) internal architecture

The three subsystems (AXI-slave, encoder, AXI-master) within the IP core are almost independent and can work simultaneously, thus accelerating the encoding process. Rows of generator matrix are stored in a block memory due to the high efficiency in managing large data sets and the ability to provide rapid access times. Additionally, Block RAM (BRAM) of FPGAs offers increased parallelism and lower power, both of which are vital for the specific design. The FSM controls the different phases of the encoder and communicate with the FSMs of the interfaces.

Further modifications are exploited to reduce the power consumed by the circuit. Memory, registers, and FIFOs are disabled between long periods of inactivity. FIFOs of interfaces are implemented with BRAM to save more slice registers and power and all designs with BRAM operate with gated clocks. The latter technique enhances significantly (up to 80%) power savings since memory of this implementation occupies most resources on FPGA, as can be also seen in Table 1.

3.3. Encoder Verification

The primary functional verification of the LDPC encoder is conducted using a C++ script. This script demands a report file in .txt format, where details, such as seed, index of test cases, and captured input messages and output codewords, are documented. These report files hold information generated by either simulated testbenches or during measurements on the FPGA. The C++ script scans these report files for the relevant messages. For each message, the script triggers another C++ software program that simulates a software version of the LDPC encoder, and compares the resulting SW-generated codeword with the corresponding HW-generated codeword. This process is iterated for each tested message and stores the comparison results.

Verification aims to achieve approximately 100% code and functional coverage using the Xcelium and Integrated Metrics Center (IMC) tools by Cadence. Code coverage encompasses several aspects: block,

expression, toggle and FSM. Functional coverage, on the other hand, is divided into data oriented, performed through covergroups that monitor input and output data and control-oriented verification, carried out using SV Assertions that mostly track AXI4-Stream transactions.

4 Evaluation and Integration on FPGA

The proposed LDPC encoder is also integrated on the XC7Z020CLG400-1 FPGA of family Zynq-7000 of AMD Company [6]. The encoder has been enriched with Processing System (PS), AXI Direct Memory Access (DMA), interconnects and Integrated Logic Analyzer (ILA) to monitor waveforms. Vivado 2023.2 and Vitis 2023.2 are used to generate the encoder bitstream and to assist bit transactions between Software and Hardware, respectively. The implementation results of the encoder (not including the remainder design) are reported in Table 1. Power consumption is estimated by the provided tools, however, power measurements on board include additional components, such as voltage regulators and active peripherals, leading to deviations and therefore are omitted. Regarding throughput, estimations indicate the maximum achievable performance, whereas real measurements reflect the actual performance tested with a 10 ns period cycle.

Logic Utilization	**Used**	**Available**	**Utilization**	**[7] Xilinx XC6VLX240T rate=1/2, length=2048**	**[8] Kintex UltraScale KCU105**	**Specifica -tions**
Number of Slice Registers	1249	106400	1.17%	3317	3670	< 2500
Number of Slice LUTs	1228	53200	2.31%	2211	6770	< 5500
Number of Block RAM	16.5	140	11.79%	32	1	-
Power consumption	**Estimation**			**Estimation**	**Measurement**	
Power [mW]	117			N/A	1371	< 2500
Performance	**Estimation**	**Measurement**		**Estimation**	**Estimation**	
Throughput [Mbps]	136	99.3		180	30000	> 0.5

Table 1: Design specifications and implementation results

5 Conclusion

In this paper, the implementation of an LDPC encoder, fully applicable to CCSDS standard for deep space applications and to AXI4-Stream communication protocol is presented. The circuit exploits the structure of generator matrix of quasi-cyclic LDPC code and aims for low power, while in parallel, performance and resource utilization objectives are satisfied. Furthermore, the proposed circuit is verified 100% with commercial tools and implemented on a space-grade FPGA.

References

[1] T.K. Moon, Error Correction Coding Mathematical Methods and Algorithms. Utah , USA, 2005.

[2] S. J. Johnson, Low-density parity-check codes. Cambridge, USA: Cambridge University Press, 2009.

[3] W. C. Huffman and V. Pless, Fundamentals of Error-Correcting Codes. Cambridge University Press, 2003.

[4] Consultative Committee for Space Data Systems, "TM synchronization and channel coding recommended standard," Standard 131.0-B-5, 2017.

[5] [ARM Ltd., "AMBA® 4 AXI4-Stream Protocol," Specification Version 1.0, 2010.

[6] AMD, "Zynq-7000 SoC Data Sheet ," Overview DS190, 2023.