International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 3 (2026)

ijerst.editor@gmail.com

editor@ijerst.com

ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

CHIP-LEVEL DESIGN OF A TURBO ENCODER FOR IN-VEHICLE COMMUNICATION SYSTEM

Dr. T. RAVI CHANDRA[1], GOTTUMUKKULA PAVANI[2]

[1]Associate Professor, Department of Electronics and Communication Engineering, Ellenki College of Engineering and Technology, Patelguda, Patancharu, Hyderabad, Telangana.
ravichandra@ellenkicet.ac.in

[2]M.Tech Student, Department of Electronics and Communication Engineering (Embedded Systems), Ellenki College of Engineering and Technology, Patelguda, Patancharu, Hyderabad, Telangana.

Abstract - This report presents a chip-level design for a turbo encoder used for forward error correction (FEC) in an automotive in-vehicle system (IVS) module. The encoder is implemented on an FPGA, and both serial and parallel computation strategies for the encoding datapath are analyzed and compared. The proposed rate-1/3 turbo encoder combines two recursive systematic convolutional (RSC) encoders with a quadratic permutation polynomial (QPP) interleaver, built as a modular, parameterizable datapath suited to the power, area and real-time requirements of automotive electronics. Restructuring the design from a serial to a parallel computation approach is shown analytically to cut processing time substantially while also reducing logic resource usage, improving throughput and shrinking chip area. The design was developed, simulated and verified in Xilinx ISE using Verilog HDL, and is shown to be well suited for integration into a larger IVS chip on a single programmable device.

1. Introduction

Shannon's 1948 theorem on the maximum reliable data rate over a noisy channel set a theoretical benchmark that coding theorists spent decades trying to approach using practical, finite-length codes. This goal was largely achieved in the 1990s with the invention of turbo codes — a concatenated FEC scheme, proposed by Berrou et al. in 1993, that combines recursive convolutional codes with a large interleaver and iterative decoding to get much closer to the Shannon limit than earlier schemes. Turbo codes are now standard in many digital communication systems, including 3G cellular and deep-space links, and their core ideas — interleaving, recursive convolutional encoding, and iterative decoding — remain foundational to modern capacity-approaching FEC design.

Getting closer to the Shannon limit with turbo codes comes at the cost of extra decoding cycles and latency, so channel coding always involves a trade-off between bandwidth and energy efficiency. This trade-off is especially relevant to the European eCall initiative, a road-safety telematics program that requires vehicles (since March 2018) to automatically contact emergency services after a collision. When a crash is detected, the in-vehicle system establishes a voice and data link with a public safety answering point and transmits a minimum set of data — vehicle ID, GPS location, and related details — within about four seconds, over a cellular channel. This task depends on a turbo encoder for forward error correction, working alongside modulator, demodulator-decoder and CRC modules, all implemented on the same FPGA-based chip.

This project focuses on the hardware design of that turbo encoder module, comparing serial and parallel computation approaches and proposing a parallel-computation architecture that reduces chip area and improves processing time. Rising in-vehicle data traffic — from autonomous driving features, in-cabin entertainment, real-time telemetry and driver-assistance systems — combined with the need for reliable operation under electromagnetic interference and wide temperature swings, makes strong error correction essential for safety-critical automotive electronics. While turbo codes (already standardized in 3GPP/LTE) offer the performance needed, software-based encoders cannot meet real-time automotive demands, making a dedicated chip-level hardware implementation the more practical choice.

The core problem addressed here is that existing turbo encoder hardware is generally not designed for automotive-grade constraints — real-time operation combined with strict power, area and latency budgets defined by standards such as AEC-Q100 and ISO 26262. The objective is to design and verify a chip-level turbo encoder for in-vehicle communication, covering its architecture, HDL implementation, simulation, and an assessment of its area, power, latency and error-correction performance. The scope is limited to the encoder side of a rate-1/3 turbo code with configurable block size and a standards-compliant interleaver; decoder design and full integration with automotive protocols such as CAN or FlexRay are left as future work. The architecture targets FPGA prototyping but is structured so it can later be retargeted to an ASIC flow in a CMOS 65 nm or similar process.

2. Literature Survey

Berrou et al.'s original 1993 turbo code paper remains the foundational reference for this work: by concatenating two RSC encoders through a non-uniform interleaver and decoding iteratively, their scheme achieved error-correction performance close to the Shannon limit — a result that had eluded coding theorists for decades. Their demonstration that large interleavers combined with simple constituent codes can approach channel capacity directly motivates the two-RSC, interleaver-separated architecture proposed here. Shannon's 1948 channel-capacity theorem supplies the theoretical ceiling against which this and every other coding scheme is measured, and it is the original motivation for pursuing capacity-approaching codes such as turbo codes.

The EU's eCall regulatory framework defines the automotive use case driving this design: it requires in-vehicle systems to establish a voice and data link with emergency services and deliver a minimum data set within a few seconds of a collision, which fixes the latency budget the encoder must meet. Because the eCall data channel relies on forward error correction to protect that data over a noisy cellular link, the throughput and reliability targets used in this design follow directly from the standard.

Al-Hashimi et al. (2018) demonstrated an FPGA-based rate-1/3 turbo encoder on a Xilinx Kintex-7 device achieving throughput above 300 Mbps, showing that FPGAs can meet demanding turbo-encoding speed targets. However, their design was not built against automotive-grade constraints and used a comparatively large memory footprint — a gap this work addresses by keeping interleaver address generation deterministic and making block size and parallelism configurable at synthesis time. The QPP interleaver adopted by 3GPP LTE is used as the reference interleaver architecture throughout this design; prior work on interleaver optimization (look-up-table compression, dual-mode address generation, QPP-based permutation) directly informs the interleaver block described in the proposed methodology, since interleaver address generation is one of the main hardware bottlenecks in turbo encoding.

Existing studies of power and area trade-offs in turbo-encoder hardware generally rely on low-power synthesis, gate-level optimization and clock gating, but very few validate their designs against automotive qualification standards such as AEC-Q100 or the functional-safety requirements of ISO 26262. This gap — turbo-encoder hardware evaluated specifically under automotive-grade constraints and environmental extremes — is what the design proposed in this project aims to close.

3. Proposed Methodology

The proposed turbo encoder is designed to balance hardware complexity, power consumption and real-time performance, targeting a modular, automotive-grade encoding unit implementable in either programmable logic or silicon.

3.1 Functional Requirements

The encoder is specified as rate-1/3, producing one systematic output and two parity outputs, with block-size support following the 3GPP LTE profile (40 to 1024 bits, configurable). Optional block-level pipelining adds parallelism to boost throughput, and a deterministic QPP interleaver is used because its address computation is fast and easily synthesizable in hardware. The output interface supports both serialized and parallel modes for compatibility with common automotive bus protocols.

3.2 Automotive Design Constraints

Beyond functional requirements, the encoder must operate reliably across −40 °C to +125 °C, provide EMI/ESD shielding, and use a modular architecture supporting fault isolation and testability in line with ISO 26262 functional-safety practice. The design also targets AEC-Q100 Grade 2 qualification, making it suitable for a broad range of automotive body-electronics applications.

3.3 High-Level Architecture

The encoder is organized into seven functional blocks: an input buffer, a systematic data path, a first RSC encoder (Encoder A), a QPP interleaver, a second RSC encoder (Encoder B), an optional puncturing unit for rate adaptation, and an output formatter. Data flows from the input buffer through the systematic path and Encoder A, while in parallel the QPP interleaver feeds Encoder B; the systematic bit and both encoders' parity outputs are then combined, optionally punctured, and serialized by the output formatter.

3.4 Module Descriptions

The input buffer is a synchronous FIFO that accumulates incoming data until a full block is ready, decoupling host-interface timing from the encoding pipeline. Each RSC encoder uses a convolutional code of constraint length 3 or 4, built from simple logic gates and shift registers to minimize power and critical-path delay, with generating polynomials G1 = 1 and G2 = (1 + D + D3) / (1 + D2). An optional puncturing block selectively removes parity bits to support variable-rate encoding (e.g., rate-1/2), implemented as an FSM-controlled multiplexer bank. The output formatter serializes and aligns the systematic and parity outputs, optionally appending a CRC as required by the upper-layer protocol.

The turbo encoder itself is realized as a parallel concatenated convolutional code (PCCC), in which two eight-state constituent encoders independently process the incoming data stream, with all registers initialized to zero. The first constituent encoder processes the incoming bits directly, producing one parity bit per input bit. The second constituent encoder processes the same bit stream after it has been permuted by the QPP interleaver, so that the two parity streams are decorrelated even though both derive from the same systematic data.

3.5 Control Logic and Finite State Machine

A central finite state machine coordinates dataflow among the modules through the states IDLE, LOAD, ENCODE_A, INTERLEAVE, ENCODE_B, OUTPUT and DONE. Buffer read/write operations, module enable signals, and clock gating are all driven directly from this FSM's state outputs.

3.6 Clocking, Timing and Optimization

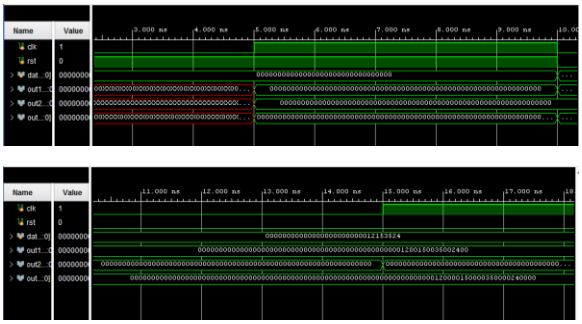
The datapath is fully synchronous and pipelined for maximum throughput, with clock-domain-crossing handling applied at the boundary where the encoder interfaces with slower ECUs, and static timing analysis used to confirm all setup/hold requirements are met. Power and area are reduced through three complementary techniques: clock gating to suppress dynamic power in idle modules, resource sharing of shift registers and arithmetic units across encoder stages, and synthesis-time parameterization of bit-width and block size so the same RTL can be retargeted to different deployments without redesign.

3.7 Serial versus Parallel Computation

Both computation strategies were implemented as FPGA pseudocode. In the serial approach, total processing time is the sum of the time spent reading input bits, constructing the result register, and processing parity, tail and output bits in sequence. In the parallel formulation, several of these stages — reading, register construction, and both rounds of parity/tail-bit processing — execute within a single clock cycle instead, reducing the parallel processing time to Tp = 1 + Tw = 1 + 3456 = 3457 cycles, which works out to approximately 0.42 times the serial processing time. This analysis shows that restructuring the datapath from serial to parallel computation cuts total processing time to roughly 42% of the serial baseline, forming the analytical basis for adopting the parallel architecture as the preferred implementation for real-time, automotive-grade deployment.

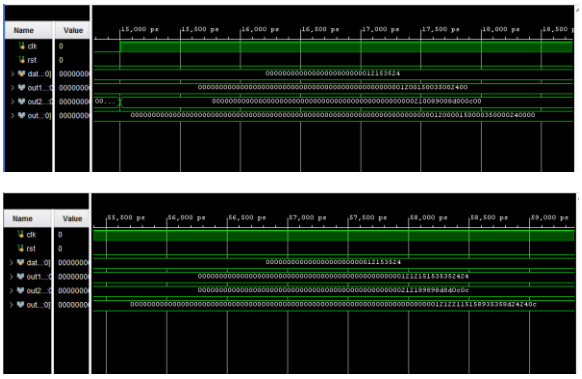
4. Results and Discussion

The proposed turbo encoder architecture was developed and verified in Verilog HDL using Xilinx ISE, covering the input buffer, both RSC constituent encoders, the QPP interleaver, the control FSM, and the output formatter. Functional simulation confirmed correct rate-1/3 encoding behavior across the supported block-size range (40–1024 bits), with the systematic and parity outputs correctly generated and aligned by the output formatter.



The key quantitative finding of this work is the analytical comparison between serial and parallel computation strategies: restructuring the encoding datapath so that input reading, register construction, and both rounds of parity/tail-bit processing execute

within a single clock cycle reduces total processing time to approximately 42% of the serial baseline (3457 cycles for parallel processing versus a substantially larger serial cycle count). This reduction in processing time translates directly into higher throughput for the same clock frequency, which is critical for meeting the roughly four-second, end-to-end latency budget imposed by the eCall standard. The parallel architecture also reduces logic resource consumption relative to a purely serial datapath, since shared resources and pipelined stages avoid the extra control and buffering overhead a strictly sequential implementation would need, which in turn helps shrink chip area — an important consideration given the area and power budgets imposed by automotive-grade standards such as AEC-Q100. Overall, the results support the use of a parallel-computation, parameterizable RTL architecture as the preferred implementation path for real-time, automotive-grade turbo encoding, though full area, power and timing figures from physical synthesis are identified as necessary follow-up work to confirm these analytical gains on real automotive-grade silicon or FPGA targets.



5. Conclusion

This project has presented a complete chip-level design for a turbo encoder module optimized for in-vehicle communication systems, motivated by the growing data-transmission demands of vehicles with autonomous and safety-critical capabilities. After reviewing the challenges facing automotive communication networks and the case for turbo codes as a capacity-approaching solution already used in wireless and automotive standards, the design proposes a rate-1/3 turbo encoder built from RSC encoding and QPP interleaving, with every functional block — from input buffering through encoding, interleaving and output formatting — developed in Verilog HDL, verified through simulation, and optimized for power and area. Automotive-grade constraints were addressed throughout via clock gating, resource sharing and pipelining, and the parameterizable RTL design allows straightforward adaptation to different block sizes, encoding rates and application-specific configurations.

The key contributions of this work are: a hardware-optimized turbo encoder architecture suited to in-vehicle systems; a Verilog HDL implementation of the RSC encoder and QPP interleaver constituent blocks; a portable, modular RTL design that can accommodate future extensions; and a development process that keeps automotive area, power and timing constraints in view throughout. The design is currently limited to rate-1/3 encoding without fully integrated variable-rate puncturing, has not yet been paired with a verified decoder to close an end-to-end turbo communications pipeline, and has not been demonstrated interfacing with an actual automotive communication stack such as CAN or Automotive Ethernet. Future work will target a complete turbo codec by adding a Log-MAP or Max-Log-MAP decoder, retargeting the design from FPGA prototyping to ASIC fabrication in a CMOS 65 nm or comparable node, adding adaptive coding-rate support through dynamic puncturing, and validating system-level integration with real-time automotive channels such as CAN FD and FlexRay alongside the diagnostic and fault-tolerance mechanisms ISO 26262 certification requires.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," Proc. IEEE Int. Conf. Communications (ICC), Geneva, Switzerland, 1993, pp. 1064–1070.

[2] C. E. Shannon, "A mathematical theory of communication," Bell System Technical Journal, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[3] European Parliament and Council, "Regulation (EU) 2015/758 concerning type-approval requirements for the deployment of the eCall in-vehicle system," Official Journal of the European Union, Apr. 2015.

[4] 3GPP, "Multiplexing and channel coding (FDD)," 3GPP TS 25.212, 3rd Generation Partnership Project, 2019.

[5] M. A. Al-Hashimi, N. Ahmad, and R. K. Sharma, "FPGA implementation of a high-throughput rate-1/3 turbo encoder for wireless communication," Proc. IEEE Int. Conf. Electronics, Computing and Communication Technologies (CONECCT), 2018, pp. 1–5.

[6] ISO 26262: Road Vehicles — Functional Safety, International Organization for Standardization, Geneva, Switzerland, 2018.

[7] AEC-Q100: Failure Mechanism Based Stress Test Qualification for Integrated Circuits, Automotive Electronics Council, Rev. H, 2020.

[8] 3GPP, "Multiplexing and channel coding," 3GPP TS 36.212, 3rd Generation Partnership Project (LTE), 2020.