International Journal of Research (IJR)

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 9 Issue 09
September 2022

TURBO ENCODER MODULE FOR IN VEHICLE SYSTEM

Muthu Shabarish K[1], S. Veerakumar[2], T. Devika[3]

[1]Research Scholar, Knowledge Institute of Technology, Kakapalayam, Salem, Tamil Nadu – 637504

[2, 3] Assistant Professor, Knowledge Institute of Technology, Kakapalayam, Salem, Tamil Nadu – 637504

ABSTRACT

This article investigates the design and implementation of the Turbo encoder as an integrated module in the in-vehicle system (IVS) chip. The Turbo encoder module was created using a field-programmable gate array (FPGA). For the encoding strategy, both serial and parallel computations are investigated. The two design strategies are discussed and argued. It is shown that by creating a parallel calculation technique employing a carry increment adder, Both chip size and processing speed have been increased. Reduced area enhances the application of reasoning. Using Xilinx tools, the Turbo encoder module was designed, simulated, and synthesized. Xilinx's Vertex Low Power is employed. The Turbo encoder module and IVS chip are intended to form a single programmable device.

Key words: Turbo encoder module, serial computation, parallel computation and carry increment adder.

I. INTRODUCTION

The Shannon limit, which Shannon first demonstrated in 1948, is the highest rate at which data can be sent via a noisy channel. Error-correcting codes may be created to get close to this capacity. As long as the codes may have an infinite length, this is achievable. For the past 60 years, coding theorists have been looking for real-world codes that can very nearly reach the Shannon limit. Turbo codes, a cutting-edge technique of error-correcting coding, were created in the 1990s. In 1993, Berrou created the concatenated forward error correction (FEC) method known as turbo coding.

In recent years, channel coding has been significantly influenced by the error-correcting technique known as "turbo coding." By employing simple component codes and huge interleavers, it beats all other known coding systems and achieves near Shannon limit error correction. Turbo codes are becoming commonplace in 3G. Turbo codes stand out for their use of interleavers, recursive systematic encoders, and iterative decoding.

The efficiency of data transmission in digital communications systems is increased via turbo coding. The turbo code was created after extensive theoretical research on polynomial selection, interleaver designs that impose weight distributions, decoder error

floors, and iterative decoding thresholds. Several spacecraft are now using a common set of turbo codes that have been implemented. LDPC codes at JPL are constructed using circulant and prototypes. Turbo codes enable communication over channels with limited power that is close to Shannon's limit.

To achieve this effect, however, a significant number of cycles are required, which increases latency. In order to allow for mistake detection and/or correction when there is channel noise, channel coding encrypts data sent across a communication channel. As a result, an important research area is how to apply turbo codes efficiently to meet real-time requirements.

After developing a channel code, there are always trades-offs between energy and bandwidth efficiency. Higher redundancy and a lower rate may commonly result in more error correction in a code. The communication system can handle more interference, communicate farther, and operate with less transmit power. Use smaller antennas and transmit at a greater data rate if more faults can be corrected. The code is energy-efficient as a result of these characteristics.

Low-rate codes, on the other hand, have a high overhead and hence consume more bandwidth. Long (low-rate) codes create a significant computational strain on conventional decoders since the complexity of decoding grows exponentially with code length. According to Viterbi, the major problem with channel coding is that encoding is straightforward but decoding is challenging.

The European Emergency Call (eCall) Telematics System was developed to increase the number of lives saved in automobile accidents. By March of 2018, the legally mandated system must be up and running. Following an accident, the EU eCall system provides a direct line of contact and data transmission between the cars and the emergency center. Through the data line, the emergency command center receives vital information for rendering aid in a crisis.

The in-vehicle system (IVS), the public safety answering point (PSAP), and the cellular communication channel are the three main parts of EU eCall. Following an automobile collision, the IVS automatically opens the data channel. The IVS collects the MSD, which contains the VIN number, GPS coordinates, and any other information needed to provide emergency assistance. It employs a cellular channel to transmit the MSD to the

adjacent PSAP in as little as four seconds. The PSAP dispatches an emergency personnel to the accident site.

The MSD signal is processed by the IVS modem utilizing a variety of components. Figure 1 depicts IVS module components. As a forward error correcting method, the IVS uses a Turbo encoder (FEC). Using digital data encoding, the Turbo encoder facilitates data transfer. When it comes to decreasing the BER in digital communications, turbo coding is among the most well-liked and successful coding systems. The modulator, demodulator-decoder, and cyclic redundancy check (CRC) modules are planned for and put into action on an FPGA device.

They are being designed to function as integrated modules for IVS chips. The hardware development of the Turbo encoder is examined in this study. Using FPGA technology, the Turbo encoder was created as an integrated module for the IVS modem. It goes through serial and parallel Turbo encoder computation methods. The Turbo encoder module is designed and implemented, and it also makes recommendations for a more efficient way to use the Turbo encoder. The evolution of the Turbo encoder's parallel calculation method demonstrates the advancements in chip size and processing speed.

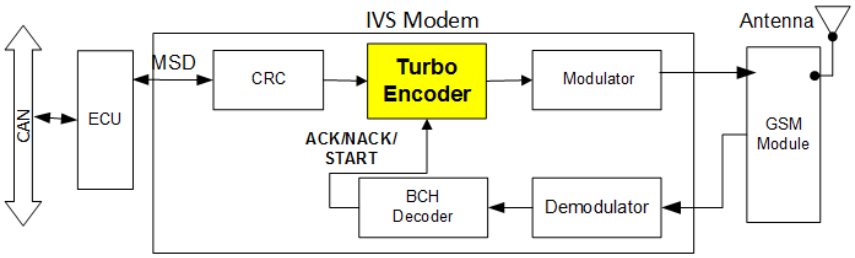

Fig. 1: The IVS block diagram.

II. LITERATURE SURVEY

The general summary of the 3GPP's "eCall data transfer; in-band modem solution," An in-depth analysis of the European Union's (EU) in-vehicle system (IVS) for the emergency call (eCall) system is presented here (EU). Designs for IVS modules are generated and implemented using field programmable gate array (FPGA) technology. Before being assembled into a system-on-chip (SoC) for the IVS electronic device, the modules are simulated, synthesized, and optimized.

Bench top tests to evaluate and test the developed modules have been completed. The hardware architecture and interfaces are covered. At various frequencies, the processing time of IVS signals is measured. We include an assortment of permitted frequencies and two

hardware interfaces. The IVS prototype platform's initial implementation technique makes use of cutting-edge FPGA architecture. This work might serve as the foundation for the eventual implementation of every IVS module on a single SoC chip.

III. TURBO ENCODER MODULE

Turbo encoding is one of the most efficient FEC algorithms utilized in digital communication. IVS employs a Turbo encoder module with a 1:3 coding rate.. The third generation partnership project (3GPP) standards include a full description of Turbo encoder features. A schematic of a 3GPP Turbo encoder is shown in Figure 1. The binary CRC parity bits are concatenated with the MSD data to create the input signal for the turbo encoders. The MSD data block size is 1148 bits. The module produces binary MSD-encoded data as an output. When employing the turbo coding technique with a 1:3 coding rate and thrills bits, the output length is 3456 bits. The structure of thrills has an impact on the Turbo encoder.

The Turbo encoder encrypts data using a parallel concatenated convolution approach (PCCC). Figure 2 depicts the PCCC's two component encoders, each of which has eight states. Zeros are the register's initial status. The utilized convolutional method is applied on the MSD bits in the first component. By taking one bit at a time, it produces one bit of parity1 bits. The second constituent adopts the same methodology as the first constituent, but after interleaving them using a mechanism created by the 3GPP, it demands the MSD bit.

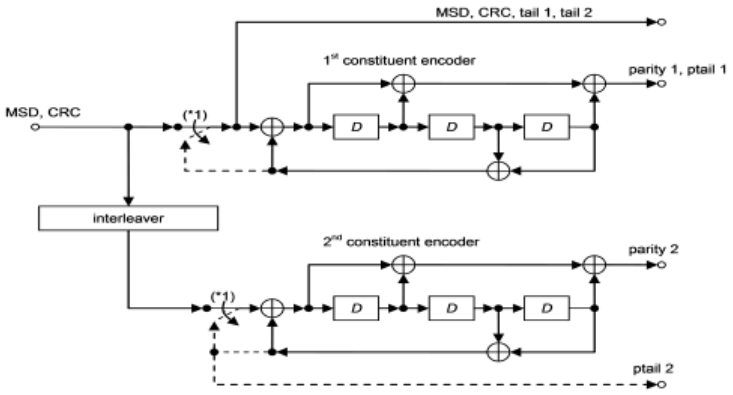

Fig. 2: The structure of the Turbo encoder.

The input data, parity1, and parity2 each have a length of 1148 bits. There are 12 tail parts altogether. Shift register input pushes them. The tail bits are used at the junctions where the encoded data blocks end. Figure depicts the format of the Turbo encoder's output data.

International Journal of Research (IJR)
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 9 Issue 09
September 2022

MSD+CRC	$tail_1$	$tail_2$	Parity 1	$ptail_1$	Parity 2	$ptail_2$

Fig.3: The output buffer of the Turbo encoder.

Interleaver

Indicate the PCCC's used transfer function as:

$$G(D) = \left(1, \frac{g_1(D)}{g_0(D)}\right) \quad (1)$$

where

$$g_1(D) = 1 + D^2 + D^3$$

$$g_0(D) = 1 + D + D^3$$

Moreover, it shows the encoder's input bits. $x_1, x_2, ..., x_K$, Moreover, it shows the encoder's input bits. $x'_1, x'_2, ..., x'_K$, and the first and second elements' respective binary outputs as $z_1, z_2, ..., z_K$ and $z'_1, z'_2, ..., z'_K$, K represent the number of input bits required by the Turbo encoder.

The output from the encoder looks like this:

$$d_K^{(0)} = x_K, d_K^{(1)} = z_K, d_K^{(2)} = z'_K$$

where $K = 0, 1, ..., K - 1.$

K represents the number of input bits required by the Turbo encoder.

This is how the encoder output is represented: $d_K^{(0)}$, $d_K^{(1)}$, and $d_K^{(2)}$, K is separated from one another by trellis bits. The shift registers' tail bits are then used to create the trellis bits after encoding all of the input bits. Once the top switch is lowered and the second constituent is switched off, the three tail bits in Figure 2 complete the first constituent. The Turbo encoder's output bits contain the trellis bits.

$$d_K^{(0)} = x_K, \quad d_{K+1}^{(0)} = z_{K+1}, \quad d_{K+2}^{(0)} = x'_K, \quad d_{K+3}^{(0)} = z'_{K+1}$$

$$d_K^{(1)} = z_K, \quad d_{K+1}^{(1)} = x_{K+2}, \quad d_{K+2}^{(1)} = z'_K, \quad d_{K+3}^{(1)} = x'_{K+2}$$

$$d_K^{(2)} = x_{K+1}, d_{K+1}^{(2)} = z_{K+2}, d_{K+2}^{(2)} = x'_{K+1}, d_{K+3}^{(2)} = z'_{K+2}$$

where $K = 0, 1, ..., K - 1.$

The internal interleaver of the 3GPP Turbo encoder aims to create a predictable connection between x_K and x'_K for any $40 \leq K \leq 5114$. For the used Turbo encoder, there is a specific method for building an internal interleaver that is described in. In this work, the internal interleaver for the used Turbo encoder is constructed using the 3GPP standard methodology.

The Turbo encoder is designed with Verilog HDL. Hexadecimal files can be read by Verilog to give information that can be utilized as interleaver data. The input data, parity1, and parity2 each have a length of 1148 bits. There are 12 tail parts altogether. Shift register input pushes them. At the intersections of the encoded data blocks, the tail bits are inserted. Figure 3 depicts the Turbo encoder's output structure.

The Turbo encoder interleaver is built in accordance with 3GPP requirements [8]. A rectangular matrix organizes the interleaver portions. The interleaver matrix has 1148 elements. 1148 MSD bits form the interleaver matrix. The interleaver reorders MSD bits. Interleaver users use intra- and inter-row approaches. B1, B2... BK, where K = 1148, indicate MSD bits.

According to 3GPP guidelines, the interleaver matrix is a rectangular RxC matrix. It's 20 by 58 pixels. The following activities influence the interleaver matrix: 1. the matrix's input bits are represented by the letters b1, b2... bK, where bK = BK and K = 1148. The remaining elements are padded with zeroes and ones. Both intra-row and inter-row permutation follow 3GPP guidelines. Except for the padded bits, all elements of the computed interleaver matrix are saved as hexadecimal files. The Verilog HDL language is used to create the Turbo encoder. Verilog is capable of reading hexadecimal files in order to obtain data and utilize it as interleaver data.

The technologies are used to produce the Turbo encoder module. The module's register transfer level is Verilog HDL (RTL). The Turbo encoding technique's input, output, and necessary parameters are defined in several registers. The two alternative encoding methods—serial and parallel computation—are examined in this work. The serial computation method uses one clock cycle to process one bit. It builds the input and output registers, reads the input data from the MSD, and serially computes the tail bits.

It completes the encoding and then generates the output bits. Despite the fact that the approach has been developed and implemented, it has been discovered that there is a

significant processing time that may be coupled with the operations of other modules. Figure 4 depicts the Turbo encoder module's serial calculation pseudo code.

```
Pseudocode code for Serial Computation of Turbo encoder
module TURBO_SERIAL ( inputs, outputs;)
        define REGISERS and PARAMETERS;
      always @(posedge clock, posedge reset)
        begin
         if (reset) output=0;
         else begin
            repeat1 (1148)  {
                        Read MSD input data}
          If (repeat1 is done)
            repeat2 (1148)  {
                        Build output register for MSDinput;
                        Build output register for Parity1;
                        Build output register for Parity2; }
          If (repeat2 is done)
            repeat3 (1148) {
                        Process Parity1 bits;  }
          If (repeat3 is done)
            Repeat4 (3) {
                        Process tail1 bits;
                        Process ptail1 bits;     }
          If (repeat4 is done)
            Repeat5 (1148) {
                        Process Parity2 bits;  }
          If (repeat5 is done)
            Repeat6(3) {
                        Process tail2 bits;
                        Process ptail2 bits;     }
          If (repeat6 is done)
            Repeat7(3456) {
                        Generate output bits     }
           end   end
endmodule;
```

Fig 4: The pseudo code for serial computation of the Turbo encoder.

In Verilog, the parallel computing approach is employed to construct the Turbo encoder. Using a parallel computing method, several operations in the serial computation technique are overlapping. The parallel Turbo encoder has two developed functionalities. The two procedures handle the vast bulk of the processing time needed for the encoding strategy. The MSD data must be used in its whole for the Turbo encoding strategy to be used. Figure 5 depicts the pseudo code for the parallel approach used by the Turbo encoder. Verilog is used to create pseudo codes for both serial and parallel computing processes.

International Journal of Research (IJR)
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 9 Issue 09
September 2022

```
Pseudocode code for Parallel Computation of Turbo encoder
module TURBO_PARALLEL ( inputs, outputs;)
        define REGISERS and PARAMETERS;
 function [3455:0] codedMSD1;
            input [1147:0] MSDdata;
               begin
                  repeat1 (1148) {
                        Build output register for MSDinput;
                        Build output register for Parity1;
                        Build output register for Parity2; }
                  If (repeat1 is done)
                     repeat2 (1148) {
                       call  CodedMSD2 (codedMSD1) }
                  end
      endfunction
 function [3455:0] codedMSD2;
              input [3455:0] codedMSD1;
                begin
                repeat3 (1148) {
                        Process Parity1 bits; }
                 If (repeat3 is done)
                   repeat4 (3) {
                        Process tail1 bits;
                        Process ptail1 bits;    }
                If (repeat4 is done)
                   repeat5 (1148) {
                        Process Parity2 bits; }
                If (repeat5 is done)
                   repeat6(3) {
                        Process tail2 bits;
                        Process ptail2 bits;    }
                  end
      endfunction
  always @(posedge clock, posedge reset)
     begin
      If (reset) output=0;
      else begin
         repeat1 (1148) {
                  Read MSD input data    }
          If (repeat1 is done) {
              call  CodedMSD1 (MSDdata)   }
          Repeat7 (3456) {
                  Generate output bits    }
        end  end
endmodule;
```

Fig 5: The pseudo code for parallel computation of the Turbo encoder.

CARRY INCREMENT ADDER:

An 8-bit increment adder has two four bit RCA (Ripple carry adders) built in. First ripple carry adder combines preset number of 4-bit inputs to provide partitioned sum and carry variation. First block's RCA is applied to conditional increment block's CIN. As a result, the ripple carry output is used to generate the first four bit total. Regardless of what the first RCA output is, the second RCA block will do the addition operation and output results to the conditional increment block.

The first RCA block's input CIN is always set to a low value. The conditional increments block is made up of half adders. The amount of the increment will be determined by the cout value of the 1st RCA block. The carry increment block's half adder performs the increment operation in this case. The output total of the second RCA is therefore obtained using the carry increment block. The Carry Increment Adder's design schematic is shown in Figure.

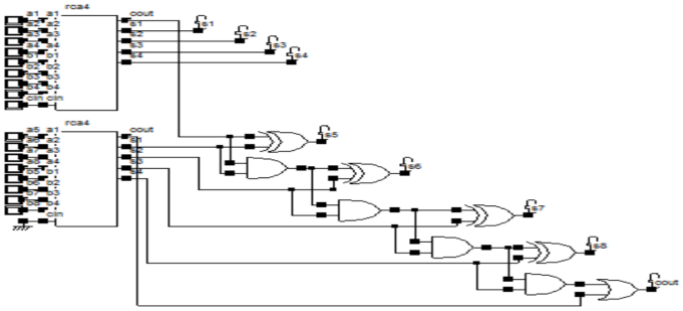

Fig6: carry increment adder

IV. RESULTS

RTL Schematic: RTL diagram, which stands for register transfer logic, denotes the transfer of logic to registers. Since it mimics the desired result of the designer, it is sometimes known as "designer viewpoint."

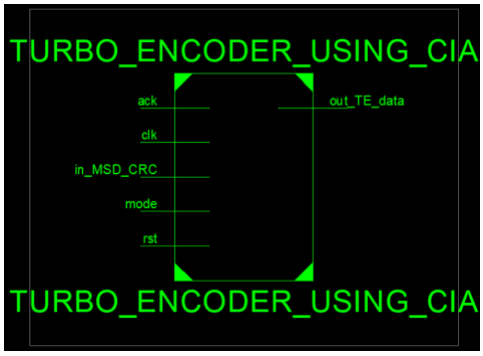

Fig 7: RTL Schematic of turbo encoder using CIA

Technology Schematic: RTL diagram, which stands for register transfer logic, denotes the transfer of logic to registers. Since it mimics the desired result of the designer, it is sometimes known as "designer viewpoint."

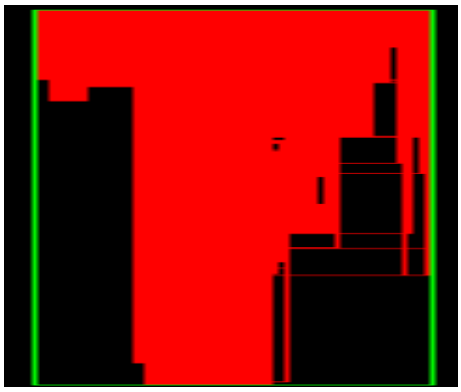

Fig8: view technology schematic of turbo encoder using CIA

Simulation: The schematic is used to confirm the connections and building blocks, whereas the simulation is the process that determines how it works. When the simulation window is started, the tool's home screen changes from implementation to simulation and limits the output to wave forms. It is adaptable enough to provide many radix number systems in this case. In this project, parallel and serial approaches are both used.

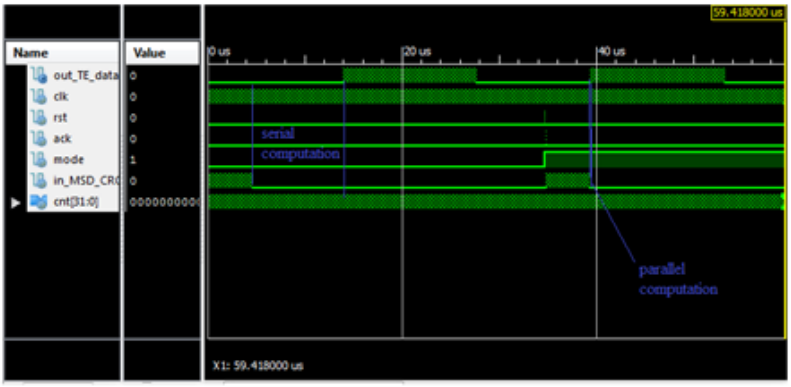

Fig 9: simulated wave form of turbo encoder

Parameter	Serial computation	Parallel computation
Required time to receiving output (ns)	**9218**	**22**

Table1: Clock comparison table

Table 2 demonstrates that the parallel processing strategy required fewer clock cycles, which reduces power consumption as the clock pulse alone accounted for 50% of the design's power usage.

Parameter	Existed Turbo encoder	Proposed Turbo encoder
No of Lut's	2,320	2,258
Power(m.Watt)	53.067	51.648

Table2: parameter comparison table

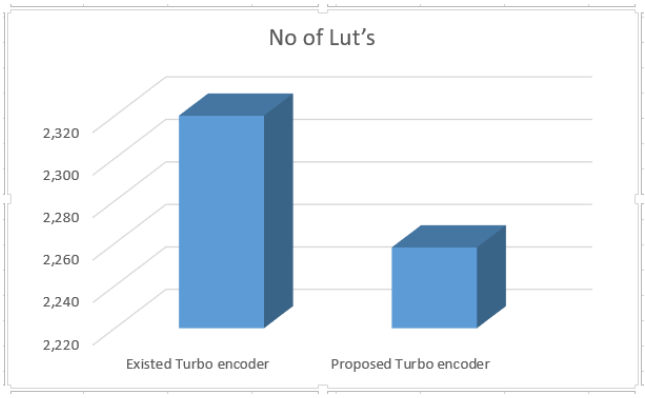

Fig10: LUT comparison bar graph

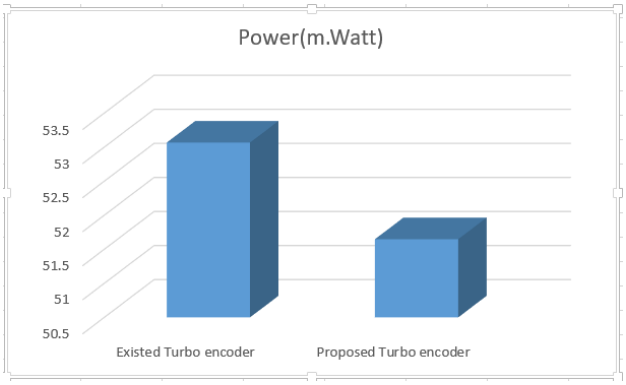

Fig11: Power comparison bargraph

Parameters: Three factors that are taken into account in VLSI are area, latency, and power; one can compare one architecture to another using these factors. Here, space and power requirements are taken into consideration. The HDL is written in verilog, and the parameters are gathered using the software XILINX 14.7ise.

V. CONCLUSION

In order to function as an integrated module in the IVS modem, the carry increment adder-encoder turbo module was designed and built. FPGA technology was used in the development of the Turbo encoder module. With the use of Xilinx tools and Verilog HDL, the module is developed and simulated. In addition to serial and parallel calculation strategies, the encoding procedure also investigates parallel and serial techniques. It is demonstrated that parallel computation may boost the module's chip size and processing speed. In compared to the serial computing technique, which speeds up processing and boosts logic consumption by 9218 clock pulses, the parallel computation encoding takes just 22 clock pulses. The planned building will also consume less energy and space. The enhanced processing speed is demonstrated by both simulation and chip processing analyses.

REFERENCES

[1] “eCall data transfer; in-band modem solution; general description,” 3GPP, Tech. Rep. TS26.267.

[2] Eroupean Commission, “eCall: Time saved / lives saved,” Press Release, Brussels, August 21, 2015. Website: http: //ec:europa:eu/digital –agenda/en/ecall - time - saved - lives - saved.

[3] A. Saleem et at. "Four-Dimensional Trellis Coded Modulation for Flexible Optical Communications," IEEE Journal of Light wave Technology. vol. 35, no. 2, pp. 151-158, Nov. 2017.

[4] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang "Design and Implementation of a Parallel Turbo-Decoder ASIC for 3GPP-LTE," IEEE Journal of Solid-state Circuits., vol. 46, no. 1, pp. 8-17, Jan. 2011.

[5] M. Nader and J. Liu, "Design and implementation of CRC module of eCall in-vehicle system on FPGA," SAE Technical 2015-01-2844, 2015, doi: 10.4271/2015-01-2844.

[6] M. Nader and J. Liu. "Developing modulator and demodulator for the EU eCall in-vehicle system in FPGAs" in IEEE, 2016 International Conference on Computing, Networking and Communications (ICNC),Hawaii, USA, Feb. 15-18, 2016, pp. 1-5.

[7] M. Nader and J. Liu. "FPGA Design and Implementation of Demodulator/Decoder Module for the EU eCall In-Vehicle System" in International Conference on Embedded Systems and Applications (ESA'15), Las Vegas, USA, July 27-30, 2015, pp. 3-9.

[8] "Technical Specification Group Radio Access Network; Multiplexing and channel coding (FDD)," 3GPP, Tech. Rep. TS22.212.

[9] D. Viktor, K. Michal, and D. Milan "Impact of trellis termination on performance of turbo codes," in ELEKTRO, 2016, pp.48-51.

[10] B. Murali krishna, G.L. Madhumati, H. Khan, K.G. Deepika, "Reconfigurable system-on-chip design using FPGA," in 2nd International Conference on Devices, Circuits and Systems (ICDCS), 2014, pp.1-6.