Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7922860

TURBO ENCODER MODULE FOR IVS WITH CIA

Dr.R.Jamuna[1], Dr.S.Bhavani[2] , Pooja. G[3] , Bharath Kumar.R[4]
Associate professor[1], Department of ECE, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu
Professor [2], Department of ECE, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu
PG[3] Student, Department of ECE, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu
PG[4] Student, Department of ECE, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu
rjamunaece@siet.ac.in, hodece@siet.ac.in ,
poojarajeswari98@gmail.com,bk66097@gmail.com

ABSTRACT
The creation and execution of the Turbo encoder are the primary subject matter of this essay. It functions as an integrated module in the chip for the in-vehicle system. A field-programmable gate array was used to make this encoder module. There are two ways utilised for encoding calculations in both serial and parallel fashion. These two design approaches are looked into. This leads us to the conclusion that two significant milestones are accomplished by adopting parallel approach for calculating in addition to CIA. Chip size and processing speed are the main and second factors, respectively. Both have gotten bigger. This has improved the reasoning application much more. Xilinx tools were used to design the Turbo encoder module. It was synthesised and simulated. The Xilinx vertex Low Power is utilised. The main purpose of the Turbo encoder module and in-vehicle system chip is to create a single programmable device.

KEYWORDS: *Turbo encoder module, In vehicle system, serial computation, parallel computation and carry increment adder*

I. INTRODUCTION

The Shannon limit, which was initially proved in the 1940s, is the maximum rate at which data may be transmitted across a noisy channel. This is possible up until the codes are infinitely long. Coding theorists have been looking for codes that will closely match and achieve the Shannon limit for a very long time—six decades, to be exact. In the 1990s, error-correcting coding, sometimes known as turbo codes, was developed. Concatenated forward error correction, or turbo coding, was developed by Berrou in the 1990s. The error-correcting method known as "turbo coding" has substantially altered channel coding in recent years. By utilising small interleavers and straightforward component codes, all other known coding methods have been defeated. A near Shannon limit error correction has also been achieved. These turbo codes are becoming more typical in 3G. Turbo codes stand out due to the usage of interleavers, recursive systematic encoders, and iterative decoding.

Turbo coding improves the efficiency of data transmission in digital communications systems. The turbo code was developed following substantial theoretical investigation into polynomial selection, interleaver designs, and iterative decoding thresholds. Numerous

spacecraft implementations now use a standard set of turbo codes. At JPL, LDPC codes—another sort of code—are created with the aid of prototypes and a circulator. These codes make it possible to communicate across weak channels that are almost at Shannon's limit. To produce this effect, however, a large number of cycles are necessary, which lengthens delay. Data delivered across a channel can be encrypted with channel coding to enable error detection and/or correction when there is channel noise. As a result, one of the key study areas is how to effectively apply turbo codes to fulfil real-time requirements.

After developing a channel code, trade-offs are inevitable, including those between energy and bandwidth efficiency. Higher redundancy and a lower rate can lead to more error correction in a code. The communication system can therefore withstand more disturbance. It has a greater range of communication and uses less transmit power. Use of smaller antennas and transition to a higher data rate allows for the correction of more faults. The code is hence energy-efficient. Low-rate codes, on the other hand, have a high overhead because of which they need more bandwidth. Low-rate codes, or long codes, place a significant computational burden on traditional decoders. This is due to the fact that as code length increases, decoding complexity also exponentially increases. According to Viterbi, the main issue with this channel coding is that while encoding is simple, decoding is difficult. The European Emergency Call (eCall) Telematics System was created to help save more lives in car accidents. The EU eCall system offers a direct line of communication and data transmission between the vehicles and the nearby emergency centre after an accident. The emergency command centre receives crucial information for providing aid in a disaster through the data connection.

The essential components of EU are the in-vehicle system, cellular connectivity, and the public safety answering point. The IVS immediately opens the data channel after a car accident. The MSD is then collected. The MSD includes the VIN number, GPS location, and other data required to deliver emergency assistance. The PSAP then dispatches emergency responders to the scene of the accident. The IVS modem uses a number of components to process the signal from MSD. The elements of the IVS module are displayed below. A Turbo encoder is used by the IVS as a forward error correcting technique. The encoder uses digital data encoding to facilitate data transfer. The most popular and effective coding method for reducing BER in digital communications is turbo coding. On an FPGA device, the modules like the modulator, demodulator, and cyclic redundancy check (CRC) are planned for and implemented. They are made to work with IVS chips as integrated modules. This paper investigates the hardware evolution of the Turbo encoder. Using FPGA technology, the Turbo encoder was developed as an integrated module for the IVS modem. Both serial and parallel Turbo encoder computation techniques are used. The Turbo encoder module has now been developed and put into practice. It offers suggestions for the most effective usage of the Turbo encoder. The parallel computation technique used by the Turbo encoder has evolved to reflect improvements in chip size and processing performance.

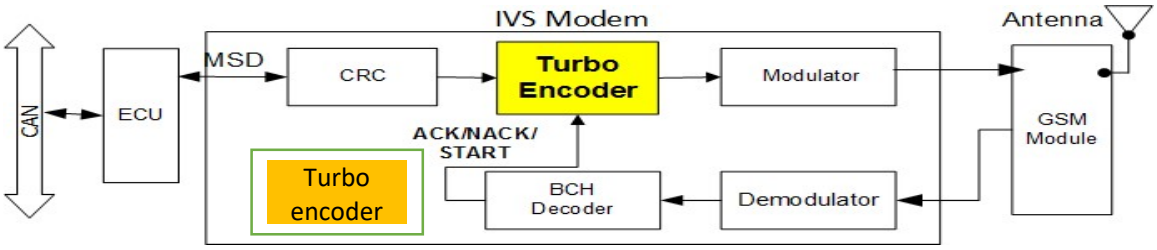


Fig. 1: The IVS block diagram

II. LITERATURE SURVEY

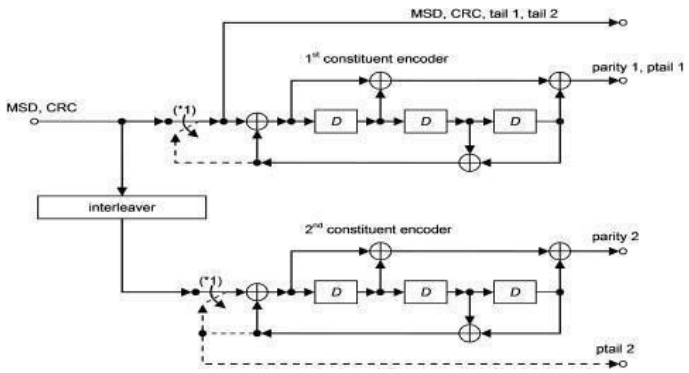
The 3GPP's "eCall data transfer; in-band modem solution," in general terms. The in-vehicle system (IVS) for the emergency call system is presented with a thorough analysis. FPGA technology is used to create and implement module designs. Before being put together into a system-on-chip for the IVS electrical device, the modules are modelled, synthesized, and optimized.

Bench top tests, for example, have been finished, as have tests to check the developed modules. Also covered are the hardware architecture and interfaces. The processing time of IVS signals is assessed at various frequencies. The initial implementation technique of IVS prototype platform is based on the state-of-the-art FPGA architecture. This might eventually serve as the basis for the integration of every IVS module into a single SoC chip.

III. TURBO ENCODER MODULE

One of the most successful FEC methods for digital transmission is this kind of encoding. A Turbo encoder module with a 1:3 coding rate is used in the automotive system. The definition of Turbo encoder features is included in the Third Generation Partnership Project (3GPP) specifications. Figure 1 depicts a schematic of a 3GPP Turbo encoder. The MSD data is concatenated with the binary CRC parity bits. Its purpose is to produce the turbo encoders' input signal. The MSD data block has a 1148-bit length. The module generates binary MSD-encoded data as its output. When using the turbo coding technique with a 1:3 coding rate and thrills bits, the output length is 3456 bits. The thrills' organizational structure has an effect on the encoder module.

Data is encrypted by the Turbo encoder utilising a parallel concatenated convolution technique. The two component encoders for the PCCC are shown in Figure 2, and each has eight states. The register has a zero initial status. The MSD bits are subjected to the convolutional approach that is being used in the first component. By extracting one bit at a time, it creates one bit of



parity1. The second ingredient uses the same approach as the first constituent. It requires the MSD bit after interleaving them using a 3GPP-designed method.

Fig. 2: The Turbo Encoder's structure.

Each of the input data's parity1 and parity2 components is 1148 bits long. Together, the tail consists of 12 pieces. All of them are pushed by shift register input. The tail bits are used at the junctions where the encoded data blocks stop. Figure displays the output data format for the Turbo encoder.

MSD+CRC	tail1	tail2	Parity 1	ptail1	Parity 2	ptail2

Fig.3: It is the Turbo encoder's output buffer.

Interleaver

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

where

$$g_1(D) = 1 + D^2 + D^3$$
$$g_0(D) = 1 + D + D^3$$

It is the transfer function that is used by the PCCC.

Additionally, it displays the input bits for the encoder, $x_1, x_2, ..., x_K$, the binary outputs for the first and second elements, and K, which stands for the quantity of input bits needed by the Turbo encoder.

K stands for how many input bits the Turbo encoder needs.
The encoder output is shown as follows:

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

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

$d_K^{(0)}$, $d_K^{(1)}$, and $d_K^{(2)}$

K are set apart from one another using trellis bits. The shift register's tail bits are utilized to generate the trellis bits after encoding all of the input bits. Once the top switch is lowered and the second constituent is turned off, the three tail bits finish the first component. The trellis bits are present in both of the Turbo encoder's output bits.

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

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

The primary goal of the 3GPP Turbo encoder's internal interleaver is to provide a predictable relationship between Xk and X'k for any. For the employed Turbo encoder, there is a special method for creating an internal interleaver that is documented in. In this work, the internal interleaver for the used Turbo encoder is built in accordance with the 3GPP standard technique.

Verilog HDL was used in the creation of the Turbo encoder. Verilog can read hexadecimal files to extract data that can be used as interleaver information. The block length of the input data, parity1, and parity2 is 1148 bits. There are a total of 12 tail components. They are pushed together by shift register input. At the intersections of the encoded data block, the tail bits are inserted. The output structure of the Turbo encoder is shown in Figure 3.

The built-in Turbo encoder interleaver complies with 3GPP specifications. A rectangular matrix is used to organise the interleaver sections. Elements in the matrix total 1148. These MSD bits are reordered by the interleaver. Users of interleavers employ an intra- and inter-row strategy. B1, B2,..., BK, where K = 1148, stand for MSD bits.

According to 3GPP specifications, the interleaver matrix is a rectangular RxC matrix. Total size is 20 by 58 pixels. The interleaver matrix is affected by the following actions: 1. The characters b1, b2,..., bK, where bK = BK and K = 1148, also stand in for the input bits of the matrix. The zeros and ones are also used to cushion the remaining elements. Following 3GPP rules, both intra-row and inter-row permutation are used. Except for the padded bits, every component of the computed interleaver matrix is stored as a hexadecimal file. The Turbo encoder was developed using the Verilog HDL language, which can read hexadecimal files to obtain data and use it as interleaver data.

The Turbo encoder modules made using these technologies. Verilog HDL is used for the module's register transfer level. The input, output, and essential parameters of the encoding technique are defined in multiple registers. Serial and parallel computation are the two additional encoding techniques. These two are looked at in this work. One clock cycle is used in the serial computation method for processing a bit at a time. Also creates the input and output registers, serially computes tail bits, and reads the MSD's input information.

It produces output data and ends the process of encoding. Although this method has been developed and put into practice, the time spent processing that corresponds to the actions of other modules may be quite long. The serial calculation pseudo code module is displayed.

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: Turbo encoder's serial computation's pseudocode

The Turbo encoder is created in Verilog adopting a parallel computing strategy. When applying a parallel computing approach, multiple tasks in the serial computation technique overlap. The parallel Turbo encoder includes two developed functionalities. The majority of the processing time required for this encoding approach is handled by the two procedures. The MSD data must be used in its whole in order to employ the Turbo encoding approach.

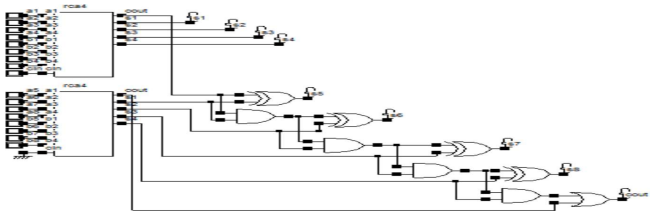
```
Pseudocode code for Parallel Computation of Turbo encoder
module TURBO_PARALLEL ( inputs, outputs;)
      define REGISER5 and PARAMETER5;
function [3455:0] codedMSD1;
            input [1147:0] MSDdata;
               begin
                  repeat1 (1148) {
                     Build output register 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;
```

Pseudocodes for serial and parallel computing operations are produced using Verilog.

Fig 5 demonstrates python coding for the parallel strategy employed.

CARRY INCREMENT ADDER

There are two four bit ripple carry adders incorporated inside an 8-bit increment adder. The primary ripple carry adder creates a partitioned sum and carry variation by combining a predetermined amount of a four-bit entries. The CIN of conditional increment block is applied by the ripple carry adder of the first block. The first four bit total is thus produced using the ripple carry output. The subsequent RCA block will carry out the adding operation and transmit



the results to the conditional increment block irrespective of what the initial RCA output is.

Fig6: carry increment adder.

It is standard practice to set the input CIN of the first RCA block to a low value. Half adders make up the conditional increments block of it. The amount of increment is determined by the

count value of the first RCA block. In this scenario, the increment operation is carried out using the carry increment block's half adder. Therefore, the output total of the second RCA is obtained by using the carry increment block. The design schematic for the Carry Increment Adder is displayed.

RESULTS :

RTL Diagram: An RTL diagram represents the transfer of logic to registers. RTL stands for register transfer logic. It is frequently referred to as "designer viewpoint" since it resembles the creator's intendedoutcome.

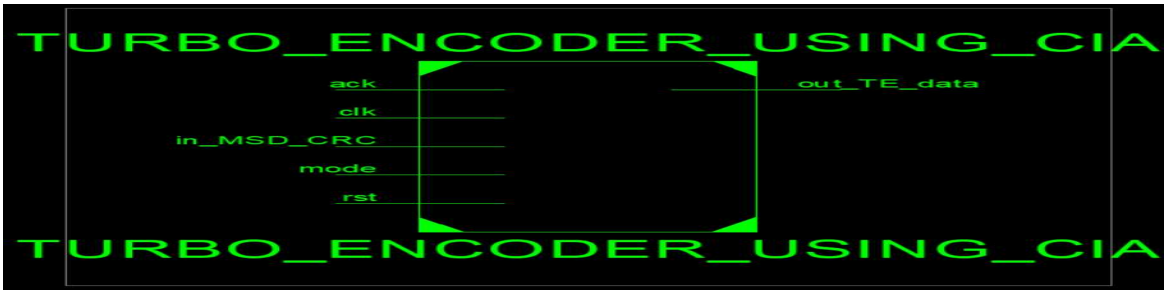


Fig 7: Turbo encoder RTL Schematic Using CIA

Technology Schematic: Register transfer logic (RTL) diagrams show the movement of logic to registers. Because it resembles the designer's intended outcome and is occasionally referred to "designer viewpoint."

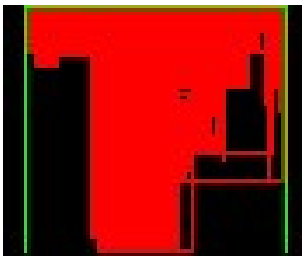


Fig 8: view of schematic of the turbo encoder technology

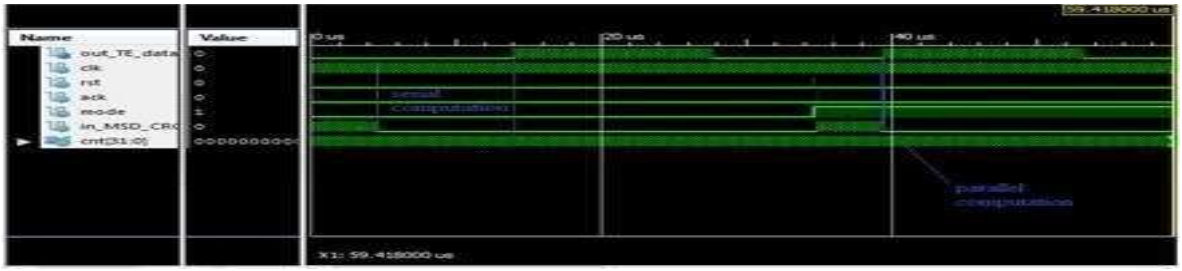


Fig 9: simulated wave form

Simulation: The simulation is employed to ensure how it functions while the design is used to build connections between components and building sections. When the simulation window commences, the tool's interface changes from execution to simulation, and it only allows wave shapes as output. You may supply a variety of radix number systems in this situation because it is so versatile. Both parallel and serial techniques are used in this project.

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

Table1: Clock comparison table

Table 2: shows that using a parallel processing approach used less clock cycles. These designs requires less electricity because the clock pulse accounted for fifty percent of the overall power consumption.

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

Table2 Table of parameter comparisons

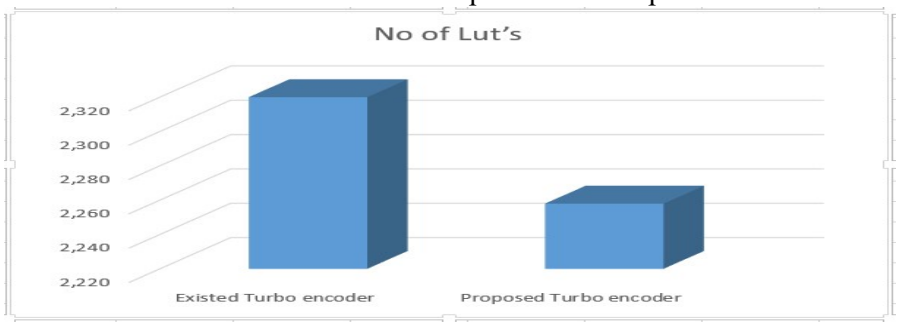


Fig10: Comparative LUT bar graph

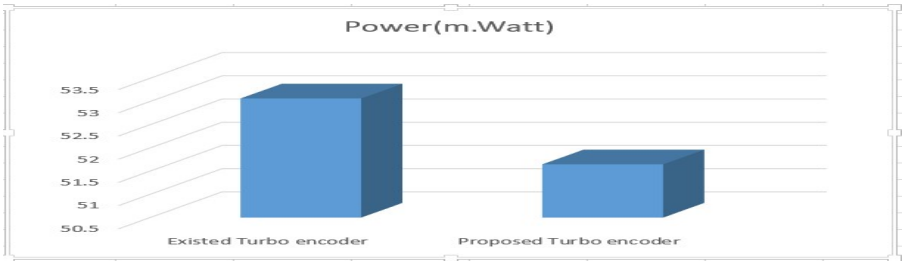


Fig11: which is Power comparison bar graph

Parameters: Area, delay, and power are the three variables that are considered in VLSI; we may use these variables to contrast one architecture with another. The need for power and space is also taken into account. The parameters are gathered and the HDL is written in verilog. It is collected with the aid of the programme XILINX 14.7ise.

V. CONCLUSION

Turbo modules are created using the carry increment adder encoder. The Turbo encoder module was created using FPGA technology. The module is built and simulated using Xilinx tools and Verilog HDL. The encoding process looks into parallel and serial methods as well. in addition to parallel and serial computation techniques. Parallel computation can increase processing speed and chip size of the module. The serial computing method increases logic usage by 9218 clock pulses while accelerating computation. The parallel processing encoding, however, only needs twenty two clock pulses. The proposed structure will also consume less energy and space. Both simulation and chip processing analyses show how the processing speed has been increased.

REFERENCES

[1] 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. International Journal of Research (IJR) e-ISSN: 2348-6848 p-ISSN: 2348-795 Vol. 9 Issue 09 September 2022 Copyright © authors 2022 49

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

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

[4]] 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.

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

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

[9] “Technical Specification Group Radio Access Network along with Multiplexing and channel coding (FDD),” 3GPP, Tech. Rep. TS22.212.

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