CHIP DESIGN FOR TURBO ENCODER MODULE FOR IN-VEHICLE SYSTEM

[1]Dr. John Paul Pulipati, Principal, [2]A Suresh Reddy, [3]D Thirupathi,
[4]D. Rajendra Prasad,
[1]principal@mrce. in,Malla Reddy College of Engineering
[2]Professor, Dept. of ECE,Malla Reddy College of Engineering
[3]Assistant Professor, Dept. of ECE,Malla Reddy College of Engineering
[4]Assistant Professor, Dept. of ECE,Malla Reddy College of Engineering

Abstract— This paper studies design and implementation of the Turbo encoder to be an embedded module in the in-vehicle system (IVS) chip. Field programmable gate array (FPGA) is employed to develop the Turbo encoder module. Both serial and parallel computations for the encoding technique are studied. The two design methods are presented and analyzed. Developing the parallel computation method, it is shown that both chip size and processing time are improved. The logic utilization is enhanced by 73% and the processing time is reduced by 58%. The Turbo encoder module is designed, simulated, and synthesized using Xilinx tools. Xilinx Zynq-7000 is employed as an FPGA device to implement the developed module. The Turbo encoder module is designed to be a part of the IVS chip on a single programmable device.

Index Term: Turbo encoder module; field programmable gate array; emergency call; in-vehicle system chip.

I. INTRODUCTION

The European emergency call (eCall) system is a telematics system designed to save more lives in vehicle accidents. It is a governmental mandatory system that is to be implemented by March 2018 [1][2]. The EU eCall system provides an immediate voice and data channel between the vehicles and an emergency center after car accidents. The data channel provides the emergency center with the necessary data for emergency aids.

The EU eCall system main parts includes the in-vehicle system (IVS), the public safety answering point (PSAP), a cellular communication channel. The IVS activates the data channel automatically when a car accident occurs. The IVS collects the minimum set of data (MSD) that includes GPS coordinates, the VIN number, and all required data for an emergency aid. It sends the MSD to the closest PSAP through a cellular channel in up to 4 seconds [1]. The PSAP sends the emergency team to the location of the accidents.

The IVS modem employs multiple modules for the MSD signal processing. The modules of the IVS are shown in Figure

1. The IVS employs a Turbo encoder as a forward error cor- recting (FEC) [1]. The Turbo encoder implements the digital data encoding technique in data transmissions. Turbo coding is one of the most popular and efficient coding technique to improve bit error rate (BER) in digital communications [3] [4]. The cyclic redundancy check (CRC) [5], the modulator [6], the demodulator-decoder [7] modules are projected and implemented on an FPGA device. They are developed to be embedded modules of the IVS chip.

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-12, 2017
DOI: 10.21276/ijcesr.2017.4.12.40

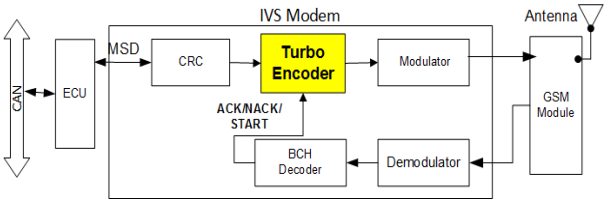


Fig. 1: The IVS block diagram.

This work studies the hardware development of the Turbo encoder. It employs FPGA technologies to develop the Turbo encoder to be an embedded module in the IVS modem. It discusses serial and parallel computation techniques for the Turbo encoder. It does not only design and implement the Turbo encoder module, but also proposes a better solution for the turbo encoder implementation. The improvement of the chip size and processing time are exhibited by developing the parallel computation technique for the Turbo encoder.

II. TURBO ENCODER MODULE

The turbo encoder technique is one of the most powerful FEC techniques in digital communication [8]. The IVS em-

ploys a Turbo encoder module with 1/3 code rate. The Turbo

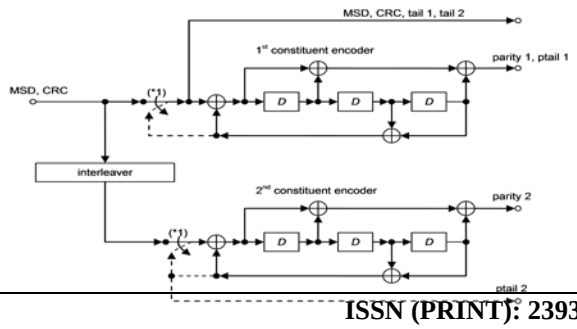
encoder functionalities are detailed in the third generation part-

nership project (3GPP) standards. The 3GPP Turbo encoder is illustrated in Figure 2 [8]. The input signal of the turbo encodes is the MSD data appended with the CRC parity bits in binary. The block length of the MSD data is 1148 bits.

Implementing the turbo coding technique with 1/3 coding rate The output of the module is the MSD encoded data in binary. and thrills bits, the length of the output is 3456 bits. The thrills

structure has an impact of the Turbo encoder [9].

The Turbo encoder employs a parallel concatenated con- volutional code (PCCC). The PCCC uses two constituent encoders with eight states as it is shown in Figure 2. The initial status of the register are zeros. The first constituent takes the MSD bits and implements the employed convolutional technique. It takes one bit at a time and generates one bit of



parity1 bits. The second constituent implements an identical technique of the first constituent, but it calls for the MSD bit after they are interleaved with a 3GPP designed interleaver technique [8].

The length of the input data, parity1, and parity2 are 1148 bits. There are 12 bits of the tail bits. They are driven from the shift register feedback. The tail bits are applied for end points between the encoded data blocks. The output structure of the Turbo encoder is illustrated in Figure 3.

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

The output of the interleaver, xJ2, xJ2, ..., xJK , is the bit sequence that are read out from the R × C matrix column by column starting with y1J and ending with y(JR C). The appended zero bits for R C > K are removed from the

output.

There are 1148 elements in the interleaver matrix. The 1148 bits of the MSD data are the elements of the interleaver matrix. The interleaver reorganizes the MSD bits in a systematic order. The intra-row and inter-row techniques of the interleaver elements organizations.

Denote the MSD bits as B1, B2, ..., BK, where K = 1148.

According to the algorithm that is explained in the above

mathematical modeling, the interleaver matrix is a rectangular matrix, where R is the number of rows and C is the number of columns of the matrix. The interleaver matrix is designed for an input data block that consists of 1148 bits. As a result, the size of the matrix is 20 X 58. The following steps are implemented to drive the

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-12, 2017
DOI: 10.21276/ijcesr.2017.4.12.40

interleaver matrix:

where bK = BK and K = 1148. The remaining elements are 1. The input bits of the matrix are denoted as b1, b2, ..., bK, padded with zeros.

2. The intra-row and inter-row permutation are performed according to 3GPP.

3. The calculated elements of the interleaver matrix, except for the padded bits, are stored in a file in hexadecimal format. The Turbo encoder is developed in Verilog HDL language.

Verilog has ability to read the hexadecimal file to get the data and use it as the interleaver data.

The length of the input data, parity1, and parity2 are 1148 bits. There are 12 bits of the tail bits. They are driven from the shift register feedback. The tail bits are applied for end points between the encoded data blocks. The output structure of the Turbo encoder is illustrated in Figure 3.

The employed interleaver for the Turbo encoder is designed according to 3GPP standards [8]. The interleaver elements are organized in a rectangular matrix. There are 1148 elements in the interleaver matrix. The 1148 bits of the MSD data are the elements of the interleaver matrix. The interleaver reorganizes the MSD bits in a systematic order. There are intra-row and inter-row techniques of the interleaver elements organizations.

Denote the MSD bits as B1, B2, ..., BK, where K = 1148.

According to the 3GPP standards [8], the interleaver matrix

is a RxC rectangular matrix, where R is the number of rows and C is the number of columns. The size of the matrix is 20 X 58. The following steps are implemented to drive the interleaver matrix:

III. FPGA DESIGN FOR THE TURBO ENCODER MODULE

FPGA technologies are employed to develop and implement

the designed Turbo encoder module. The register transfer level (RTL) of the module is developed in Verilog HDL. There are multiple registers defined for the input, output, and necessary parameters to implement the Turbo encoding technique. This work studies two methods to execute the encoding, which are serial computation and parallel computation.

The serial computation method processes one bit in one clock cycle. It reads the input data of the MSD, builds the input and output registers, and calculates the parity1, parity2, and the tail bits in a serial process. After performing the encoding, it generates the output bits. Although the method is designed and implemented, it is noted that there is a long processing time that can be overlapped with the other processes in the module. Figure 4 shows the pseudocode of the serial computation of the Turbo encoder module.

The parallel computing technique is employed to develop the Turbo encoder in Verilog. There are many processes in the serial computation technique that are overlapped by using parallel computing technique. There are two functions developed in the parallel Turbo encoder. The two functions implements almost all the processing time of the encoding technique. The Turbo encoding technique needs the MSD data as a whole package to implement the encoding. Figure 5 shows the pseudocode of the implemented parallel technique

```
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 pseudocode for serial computation of the Turbo encoder

for the Turbo encoder. Both Pseudocodes of the serial and parallel computing techniques are designed in Verilog and implemented on an FPGA device.

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-12, 2017
DOI: 10.21276/ijcesr.2017.4.12.40

The processing time of the Turbo encoder module (in clock cycles) is denoted by Ts for the serial computation and by Tp for the parallel technique, one has,

Ts = Tr + Tb + Tparity1 + Ttail1 + Tparity2 + Ttail2 + Tw (6)

Ts = 1148 + 1148 + 1148 + 3 + 1148 + 3 + 3456 = 8054

where Tr is the time for reading the 1148 bits of the MSD,

Tb is the processing time to build the output register, Tparit1 is the time of processing parity bits, Ttail is the time of processing tail bits, and Tw is the time of generating output bits.

Note that the Tr + Tb + Tparity1 + Ttail1 + Tparity2 + Ttail2

are processed in one clock cycle in the parallel computing

Tp = 1 + Tw = 1 + 3456 = 3457 (7)

Then one has,

Tp = 0.42Ts (8)

Eq. 8 reveals that the parallel computing can improve the

proceeding time of the Turbo encoder by 58%.

A. Simulation and Verification

Xilinx tools are utilized to simulate the developed modules. A test bench is designed to simulate the Turbo encoder. Verilog

```
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 pseudocode for parallel computation of the Turbo encoder.

HDL is employed to design the test bench. There are two MSD data that are simulated for each of the serial computation and parallel computation of the Turbo encoder. Figure 6 shows the simulation results for the serial computation of the Turbo encoder module. The simulation indicates that the structure output data is correct. However, there is a long time that can be overlapped, which is colored in red in the output trace.

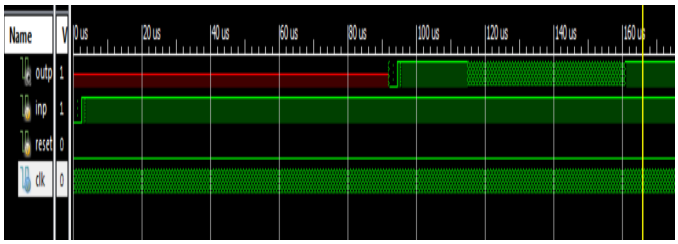


Fig. 6: The simulation result of the Turbo encoder with serial computation.

Figure 7 shows the simulation result of the parallel comput- ing technique. The module starts to generate the output in the beginning. Note that the MSD is encoded in a shorter time compare with the serial computation simulation. This is the result of parallel computing of multiple process in one clock cycle.

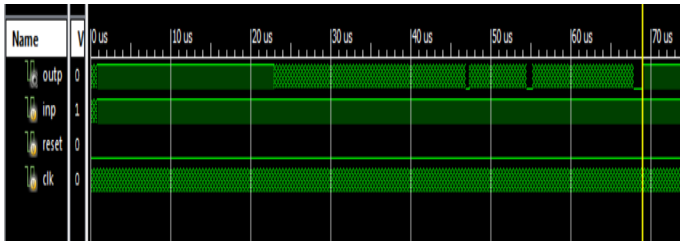


Fig. 7: The simulation result of the Turbo encoder with parallel computation.

B. Hardware Implementation

FPGA has been widely used in the SoC design [10][11]. Zynq-7000 FPGA device is employed to implement the devel- oped Turbo encoder module. Both serial and parallel computa- tion methods are implemented on the FPGA device separately. The Turbo encoder module is synthesized and loaded on the FPGA device. The synthesis reports show that the parallel computation does not only save processing time, but also reduces the utilized logics and the chip

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-12, 2017
DOI: 10.21276/ijcesr.2017.4.12.40

size. Figures 8 and 9 show the utilization flip flops and look-up tables (LTU) for the serial and parallel computation respectively.

Fig. 8: The logic utilization of the Turbo encoder with serial computation.

The impact of the parallel computation on the logic uti- lization is shown in Figure 10. Note that the utilized flip flop and LUTs are significantly reduced when parallel computation is employed. The LUT utilization is improved by 73% and the flipflop utilization enhanced by 75%. It is proven that the FPGA technologies strongly supports parallel computation for the Turbo encoder. By reducing the size of the Turbo encoder module, the IVS can be implemented on a single programmable chip with less hardware constrains.

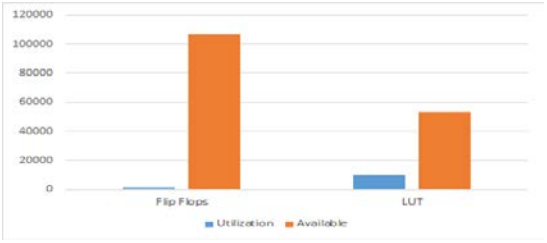


Fig. 9: The logic utilization of the Turbo encoder with parallel computation.

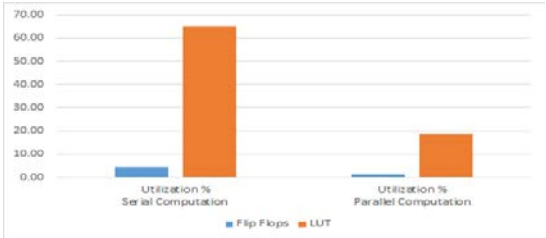


Fig. 10: The logic utilization of the Turbo encoder, serial vs parallel computation.

IV. CONCLUSION

The Turbo encoder module is designed and implemented to be an embedded module in the IVS modem. FPGA tech- nologies are employed to develop the Turbo encoder module. Xilinx tools and Verilog HDL are employed to design and simulate the module. Both serial and parallel computation techniques are studied for the encoding process. It is shown that the parallel computation can improve the chip size and processing time of the module. Comparing with the serial computation technique, the parallel computation encoding, improves the processing time by 58% and logic utilization by 73%. The processing time enhancement can be seen in both simulation and analyzing the chip processing.

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 timesaved lives saved.

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

[4] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang "Design and Implemen- tation 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 Demodula- tor/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. Muralikrishna, G.L. Madhumati, H. Khan, K.G. Deepika, "Recon- figurable system-on-chip design using FPGA," in 2nd International Conference on Devices, Circuits and Systems (ICDCS), 2014, pp.1-6.

[11] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, "Features, design tools, and application domains of FPGAs," IEEE Trans. Ind. Electron., vol. 54, no. 4, pp. 18101823, Aug. 2007.