IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

The design of the chip for the turbo encoder module in the in-car system

KICHAGARE LATHA[1], Yeddula sreelatha[2]

[1]Department of ECE, PG Research Scholar, Sri Annamacharya Institute of Technology and Science, rajampet, Y.S.R Kadapa, A.P,

[2]Department of ECE, Assistant Professor, Sri Annamacharya Institute of Technology and Science, rajampet, Y.S.R Kadapa, A.P,

ABSTRACT:

Low complexity turbo-like codes based on simple two-state trellis or simple graph structure results in encoding with low complexity. Out of this Convolution encoder and turbo codes are widely used due to the excellent error control performance. The most popular communications encoding algorithm, the iterative decoding requires an exponential increase in hardware complexity to achieve greater encode accuracy. This project focuses on the realization of turbo encoder and decoder using Log-Map based Iterative decoding technique. The turbo codes are designed with the help of Recursive Systematic Convolutional and are separated by inter leaver, which (component used to rearrange the bit sequence) plays a vital role in the encoding process. This paper provides the design of a Turbo Encoder, which is parallel concatenation of Recursive Systematic Convolutional (RSC) encoders and inter leaver to reduce delay. The Turbo Encoder is designed by Verilog-HDL and Synthesized by Xilinx ISE.

Key words: chip, turbo encoder module and in-car system

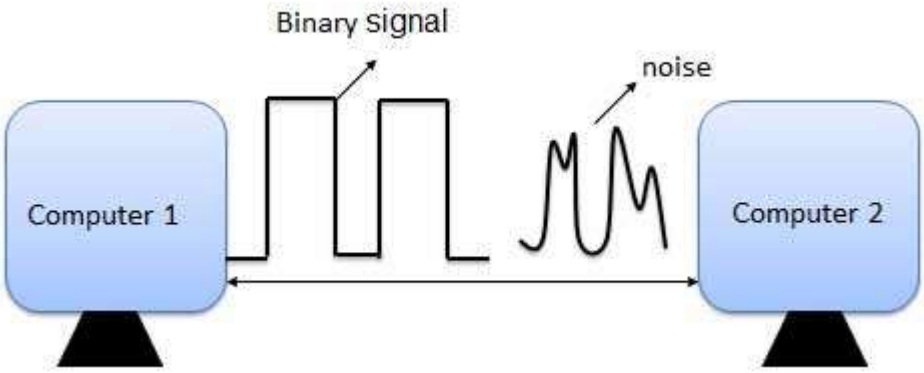
INTRODUCTION:

Error could be a condition once the output information doesn't match with the input information [1]. Throughout transmission, digital signals suffer from noise which will introduce errors within the binary bits movement from one system to another. That means a 0 bit might change to 1 or a 1 bit might change to 0. Whenever a message is transmitted, it might get push by noise or data may get corrupted [8]. To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if an error occurred throughout transmission of the message. A straight forward example of error-detecting code is parity check. Along with error-detecting code, we are able to pass some data to work out the original message from the corrupt message that we received [3]. This type of code is termed as an error-correcting code. Error-correcting codes also deploy the similar strategy as error-detecting codes however in addition; such codes also detect the exact location of the corrupt bit. In error-correcting codes, parity check encompasses an easy way to detect errors along with a sophisticated mechanism to work out the corrupt bit location. Once the corrupt bit is found, its value is reverted (from zero to one or one to zero) to urge the original message. To detect and correct the errors, extra bits are super imposed to the data bits at the time of transmission. The extra bits are referred to as parity bits [7]. They permit detection or correction of the errors. The data bits in conjunction with the parity bits form a code word. Parity checking is employed for error detection. It is the best technique for detecting and correcting errors. The MSB of an 8-bits word is employed as the parity bit and the remaining 7 bits are employed as data or message bits. The parity of 8-bits transmitted word is either even parity or odd parity. Even parity suggests that the amount of 1's within the given word as well as the parity bit should be even (2, 4, 6

...). Odd parity means the number of 1's in the given word including the parity bit should be odd (1, 3, 5 ...). The parity bit is set to zero and one depending on the type of the parity needed [10]. For even parity, this bit is set to one or zero such that the no. of "1 bits" in the entire word is even. For odd parity, this bit is set to one or zero such that the no. of "1 bits" in the entire word is odd. The theory of error-correcting codes was originated in the late 1940's by Richard Hamming, a mathematician who worked for Bell Telephone. Hamming's motivation was to program a computer to correct "bugs" which arose in punch-card programs. Hamming's overall motivation behind the theory of error-orrecting codes was to reliably enable digital communication. TURBO codes were first developed in a doctoral dissertation in 1963 by
R.G. Gallager. Gallager's work was largely ignored for approximately 30 years until connections were drawn between the iterative methods used for decoding both TURBO codes and Turbo codes. Orthogonal Lattice Square (TURBO) codes were first discovered by Gallager [1,2] in the early 1960's and have recently been rediscovered and generalized [3–14]. They have experienced an amazing comeback in the last few years. Unlike many other classes of codes TURBO codes are already equipped with very fast (probabilistic) encoding and decoding algorithms. It has been shown that these codes achieve a remarkable performance with iterative decoding that is very close to the Shannon limit. Consequently, these codes have become strong competitors to turbo codes for error control in many communication and digital storage systems where high reliability is required. An TURBO code is defined as the null space of a parity check matrix H with the following structural properties: (1) each row consists of _ "ones"; (2) each column consists of "ones"; (3) the number of "ones" in common between any two columns, denoted _, is no greater than 1; (4) both _ and are small compared to the length of the code and the number of rows in H [1, 2]. Since _ and are small, H has a small density of "ones" and hence is a sparse matrix. For this reason, the code specified by H is called an TURBO code. The TURBO code defined above is known as a regular TURBO code. If not all the columns or all the rows of the parity check matrix H have the same number of "ones" (or weights), an TURBO code is said to be irregular. Although TURBO codes have been shown to achieve outstanding performance, no analytic (algebraic or geometric) method has been found for constructing these codes. Gallager only provided a class of pseudo-random TURBO codes [1, 2]. Good TURBO codes that have been found are largely computer generated, especially long codes. Encoding of these long computer generated TURBO codes is quite complex due to the lack of code structure such as cyclic or quasi-cyclic structure. Furthermore, their minimum distances are either poor or hard to determine. TURBO codes can be classified into regular and irregular codes. With irregular codes improved performance is possible, since variable nodes with higher degrees (degree is number of adjacent nodes) collect more information from their adjacent check nodes and they get corrected first after a small number of iterations. They then help other variable nodes to get corrected through iterative decoding, similar to "wave effect". When all variable nodes have the same degrees, as in regular codes, this wave effect is not present and all variable nodes can get stuck during the decoding process.

ERROR DETECTION AND CORRECTION TECHNIQUE:

Error is a condition when the output information does not match with the input information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from one system to other. That means a 0 bit may change to 1 or a 1 bit may change to 0.



Error-Detecting codes

Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if an error occurred during transmission of the message. A simple example of error-detecting code is **parity check**.

Error-Correcting codes

Along with error-detecting code, we can also pass some data to figure out the original message from the corrupt message that we received. This type of code is called an error-correcting code. Error-correcting codes also deploy the same strategy as error-detecting codes but additionally, such codes also detect the exact location of the corrupt bit.

In error-correcting codes, parity check has a simple way to detect errors along with a sophisticated mechanism to determine the corrupt bit location. Once the corrupt bit is located, its value is reverted (from 0 to 1 or 1 to 0) to get the original message.

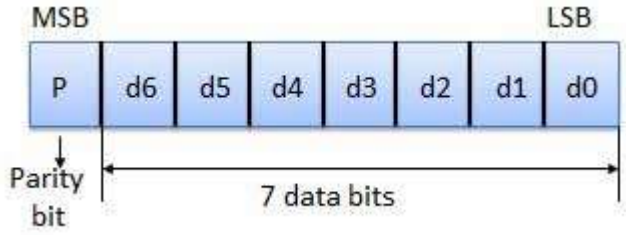
How to Detect and Correct Errors?

To detect and correct the errors, additional bits are added to the data bits at the time of transmission.

- The additional bits are called **parity bits**. They allow detection or correction of the errors.
- The data bits along with the parity bits form a **code word**.

Parity Checking of Error Detection

It is the simplest technique for detecting and correcting errors. The MSB of an 8-bits word is used as the parity bit and the remaining 7 bits are used as data or message bits. The parity of 8-bits transmitted word can be either even parity or odd parity.



Even parity -- Even parity means the number of 1's in the given word including the parity bit should be even (2,4,6,....).
Odd parity -- Odd parity means the number of 1's in the given word including the parity bit should be odd (1,3,5,....).

Use of Parity Bit

The parity bit can be set to 0 and 1 depending on the type of the parity required.

- For even parity, this bit is set to 1 or 0 such that the no. of "1 bits" in the entire word is even. Shown in fig. (a).
- For odd parity, this bit is set to 1 or 0 such that the no. of "1 bits" in the entire word is odd. Shown in fig. (b).

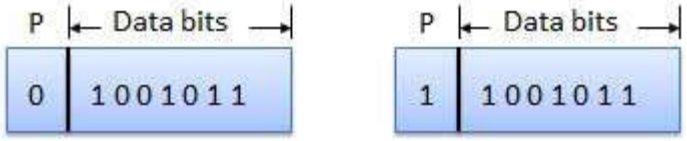


Fig. (a)

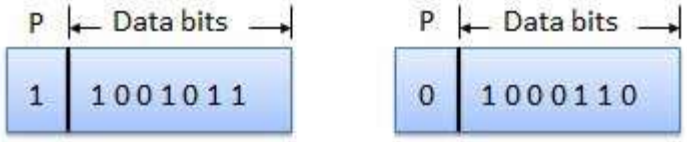
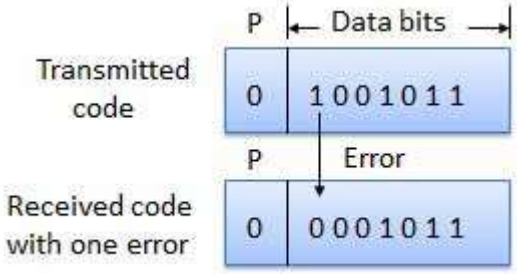


Fig. (b)

How Does Error Detection Take Place?

Parity checking at the receiver can detect the presence of an error if the parity of the receiver signal is different from the expected parity. That means, if it is known that the parity of the transmitted signal is always going to be "even" and if the received signal has an odd parity, then the receiver can conclude that the received signal is not correct. If an error is detected, then the receiver will ignore the received byte and request for retransmission of the same byte to the transmitter.



One common source of faults in a distributed system involves data corruption. This can occur during transmission or during storage in disk or memory. Data corruption can occur as a result of two failure modes – random and systematic. A random error is just that – data is corrupted in a random and unpredictable way. This might occur due to noise on a transmission line, conflicts with other software in a mutual exclusion violation, interrupted communications – the possibilities are endless.

A systematic error occurs in a predictable way. For example, a communications system might erroneously drop the last bit of every message. Or perhaps the Nth bit of every message is toggled. Perhaps anytime a specific sequence of bits is encountered a error is introduced.

To address these problems, data is often coded. Data coding simply means replacing a data word with a code word. 'Word' in this case may mean a bit sequence of length N – rather than 4 or 8 bytes. While coding is also used for security, in this sense we will use coding for error detection and possible recovery.

There is no magic in error detection and recovery codes. In general, the larger the code word with respect to the original data word, the more information is present and the more error detection and recovery can occur. In other words, there is information replication. There is a trade-off; longer messages have a higher probability of errors.
Data codes are either separable or non-separable. A separable code is easily stripped off a data message. A non-separable code requires the original data message be reconstructed.
Data coding is utilized in communications, memory systems, storage media, and anywhere data corruption is possible. In general, the type of coding utilized depends upon the nature of expected errors. No one coding scheme is practical and efficient for all types and modes of faults.

TURBOS CODES:

The theory of error correcting codes has presented a large number of code constructions with corresponding decoding algorithms. However, for applications where very strong error correcting capabilities are required these constructions all result in far too complex decoder solutions. The way to combat this is to use concatenated coding, where two (or more) constituent codes are used after each other or in parallel - usually with some kind of interleaving. The constituent codes are decoded with their respective decoders, but the final decoded result is usually sub-optimal. This means that better results might be achieved with a more complicated decoding algorithm - like the brute-force trying of all possible codewords. However, concatenated coding offers a nice trade of between error correcting capabilities and decoder complexity. Concatenated coding is illustrated in Figure 1. Here we see the information frame illustrated as a square - assuming block interleaving - and we see the parity from the vertical encoding and the parity from the horizontal encoding. For serial concatenation the parity bits from one of the constituent codes are encoded with the second code and we have parity of parity. If the codes are working in parallel, we do not have this additional parity. The idea of concatenated coding fits well with Shannon's channel coding theorem, stating that as long as we stay on the right side of the channel capacity we can correct everything - if the code is long enough. This also means that if the code is very long, it does not have to be optimal. The length in itself gives good error correcting capabilities, and concatenated coding is just a way of constructing - and especially decoding - very long codes

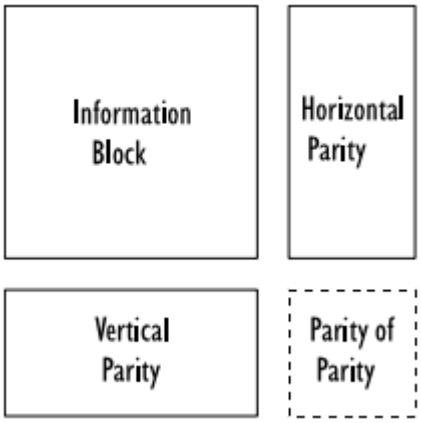


Figure : Concatenated coding

A generic Turbo encoder (Barbulescu et al 1999) has been shown in BELOW Figure. The input sequence of the information bits is organized in blocks of length N. The first block of data will be encoded by the Recursive Systematic Convolutional codes RSC ENCODER1 block, which is recursive systematic encoder. The same block of information bits is interleaved by the interleaver, and encoded by RSC ENCODER2, which is also systematic recursive encoder. The code word is framed by concatenating out put code words Xk, Y1k, and Y2k
Due to similarities with product codes (Orhan Gazi and Ali Ozgur Yilmaz 2006), it can be called the RSC ENCODER1 block as the encoder in the horizontal dimension and the RSC ENCODER2 block as the encoder in the vertical dimension. The interleaver block, rearranges the order of the information bits of input to the second encoder. The main purpose of the Interleaver (Sadjadpour et al 2000) is to increase the minimum distance of the Turbo code such that after correction in one dimension the remaining errors should become correctable error patterns in the second dimension. Ignoring for the moment the delay for each block, we assume both encoders output data simultaneously. This is rate 1/3 Turbo code, the output of the Turbo encoder being the triplet (Xk, Y1k, and Y2k). This triplet is then modulated for transmission across the communication channel, which is Additive White Gaussian Noise channel. Since the code is systematic, Xk is the input data at time k. Y1k, and Y2k are the two parity bits at time k. The two encoders do not have to be identical.
The basic idea of turbo codes is to use two convolutional codes in parallel with some kind of interleaving in between. Convolutional codes can be used to encode a continuous stream of data, but in this case we assume that data is configured in finite blocks - corresponding to the interleaver size. The frames can be terminated - i.e. the encoders are forced to a known state after the information block. The termination tail is then appended to the encoded information and used in the decoder. The system is illustrated in below Figure .

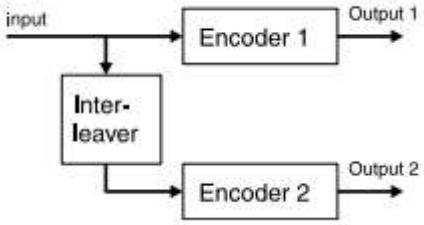


Fig: turbo encoder

We can regard the turbo code as a large block code. The performance depends on the weight distribution - not only the minimum distance but the number of words with low weight. Therefore, we want input patterns giving low weight words from the first encoder to be interleaved to patterns giving words with high weight for the second encoder. Convolutional codes have usually been encoded in their feed-forward form, like (G1,G2)=(1+D ,1+D+D). However, for these codes a single 1, i.e. the sequence 2 2 ...0001000..., will give a codeword which is exactly the generator vectors and the weight of this codeword will in general be very low. It is clear that a single 1 will propagate through any interleaver as a single 1, so the conclusion is that if we use the codes in the feed-forward form in the turbo scheme the resulting code will have a large number of codewords with very low weight. The trick is to use the codes in their recursive systematic form where we divide with one of the generator vectors. Our example gives (1,G2/G1)=(1,(1+D+D /(1+D)). This operation 2) 2 does not change the set of encoded sequences, but the mapping of input sequences to out-put sequences is different. We say that the code is the same, meaning that the distance properties are unchanged, but the encoding is different. In Figure 3 we have shown an encoder on the recursive systematic form. The output sequence we got from the feed-forward encoder with a single 1 is now obtained with the input 1+D =G1. More important is the fact that a single 1 gives a codeword of semi-infinite 2 weight, so with the recursive systematic encoders we may have a chance to find an interleaver where information patterns giving low weight words from the first encoder are interleaved to patterns giving words with high weight from the second encoder. The most critical input patterns are now patterns of weight 2.

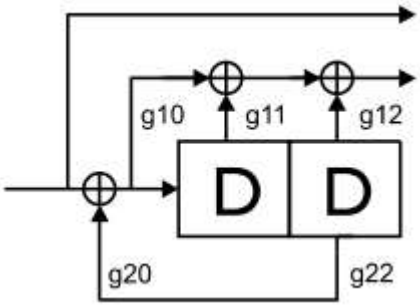


Figure 3 Recursive systematic encoder

For the example code the information sequence ...01010... will give an output of weight 5. Notice that the fact that the codes are systematic is just a coincidence, although it turns out to be very convenient for several reasons. One of these is that the bit error rate (BER) after decoding of a systematic code can not exceed the BER on the channel. Imagine that the received parity symbols were completely random, then the decoder would of course stick to the received version of the information. If the parity symbols at least make some sense we would gain information on the average and the BER after decoding will be below the BER on the channel. One thing is important concerning the systematic property, though. If we transmit the systematic part from both encoders, this would just be a repetition, and we know that we can construct better codes than repetition codes. The information part should only be transmitted from one of the constituent codes, so if we use constituent codes with rate 1/2 the final rate of the turbo code becomes 1/3. If more redundancy is needed, we must select constituent codes with lower rates. Likewise we can use puncturing after the constituent encoders to increase the rate of the turbo codes. Now comes the question of the interleaving. A first choice would be a simple block interleaver, i.e. to write by row and read by column. However, two input words of low 6 . . . . . 0 0 0 0 0 . . . . . 0 1 0 1 0 . . . . . 0 0 0 0 0 . . . . . 0 1 0 1 0 . . . . . 0 0 0 0 0 . . . . . Figure 4 Critical pattern in block interleaver weight would give some very unfortunate patterns in this interleaver. The pattern is shown in Figure 4 for our example code. We see that this is exactly two times the critical twoinput word for the horizontal encoder and two times the critical two-input pattern for the vertical encoder as well. The result is a code word of low weight (16 for the example code) - not the lowest possible, but since the pattern appears at every position in the interleaver we would have a large number of these words.

Turbo Encoder is a parallel concatenation of Two Recursive Systematic Convolutional (RSC) Encoders and is separated by interleaver. RSC Encoders generates twodifferent codes one is systematic output and the second one is parity bits. But each RSC encoders takes different bit stream as an input. The first one will take original data as input and second will take interleaved data as input. Interleaving is a process in which bits are rearranged by using the desired algorithm. The Turbo Encoder gives output of 28 bits

which is a combination of input data and an output of two RSC encoders which is thrice the length of the input bits as it shown in below Fig

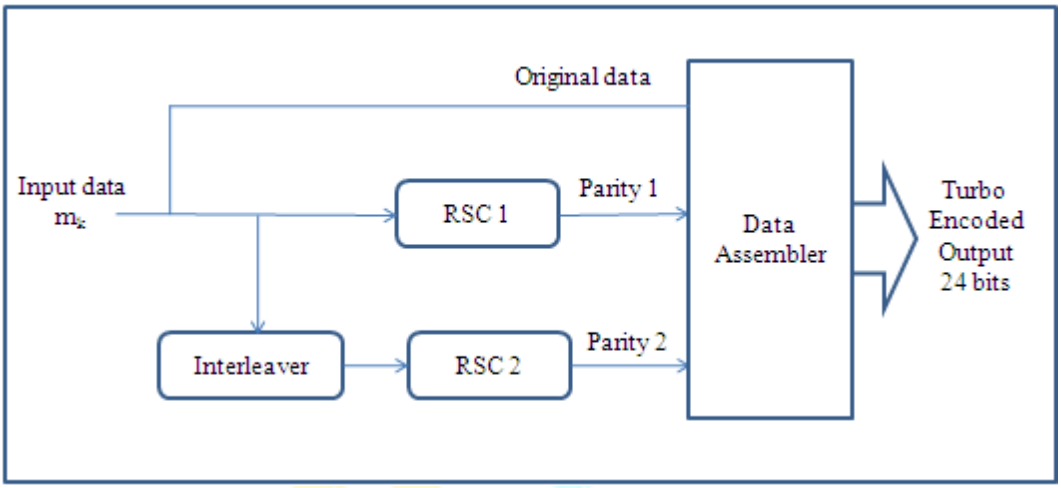


Fig: Turbo Encoder Block Diagram

RSC ENCODER:

The Turbo encoder is designed with two RSC encoders and is separated by block interleavers shown in below Fig . The input which is given to Turbo Encoder is passed through the RSC 1 and Interleaver. The interleaved is given to RSC 2. RSC 1 generates original information bits and parity 1, RSC 2 generates originals information bits and parity 2. The data bits, parity 1 and parity 2-bit streams are given to Data Assembler and it gives 24-bit Turbo Encoded data.

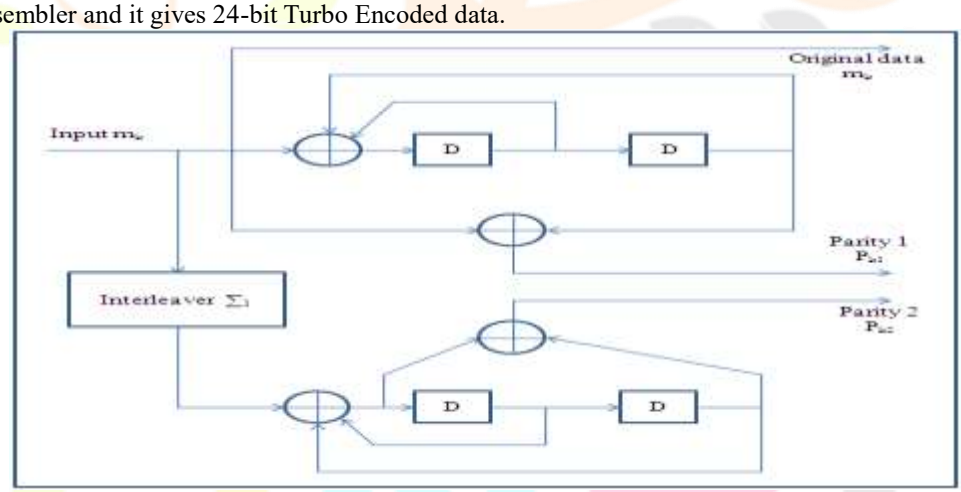
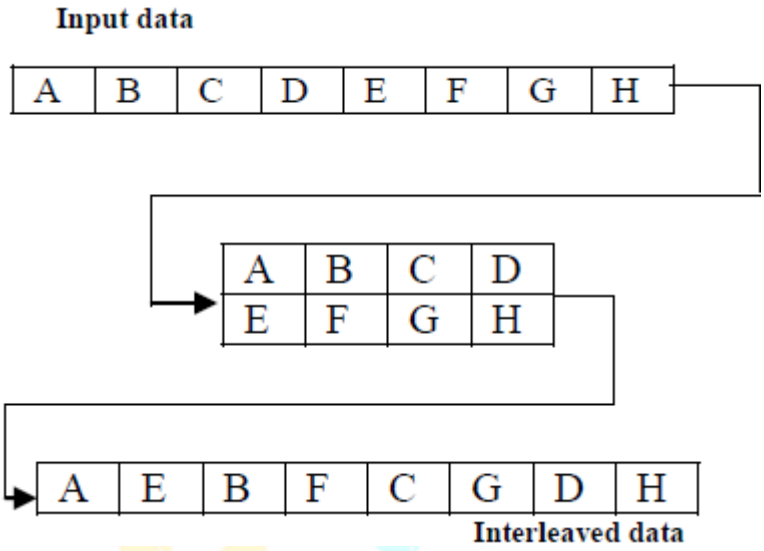


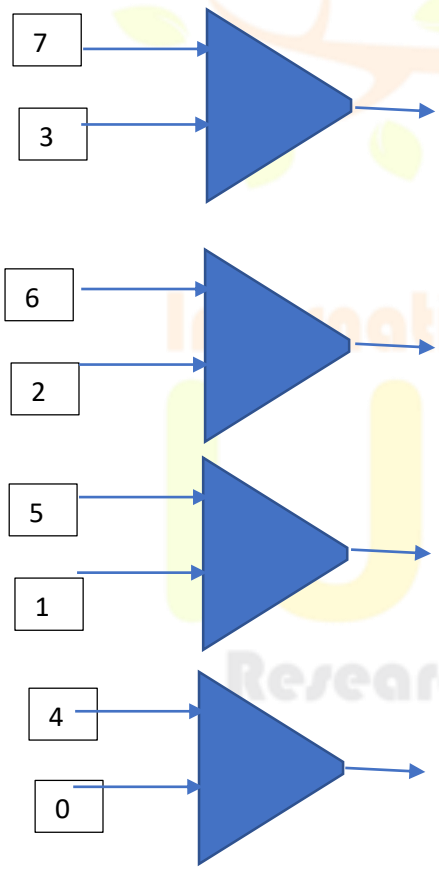
Fig : Recursive Systematic Convolutional Encoder

INTERLEAVER:

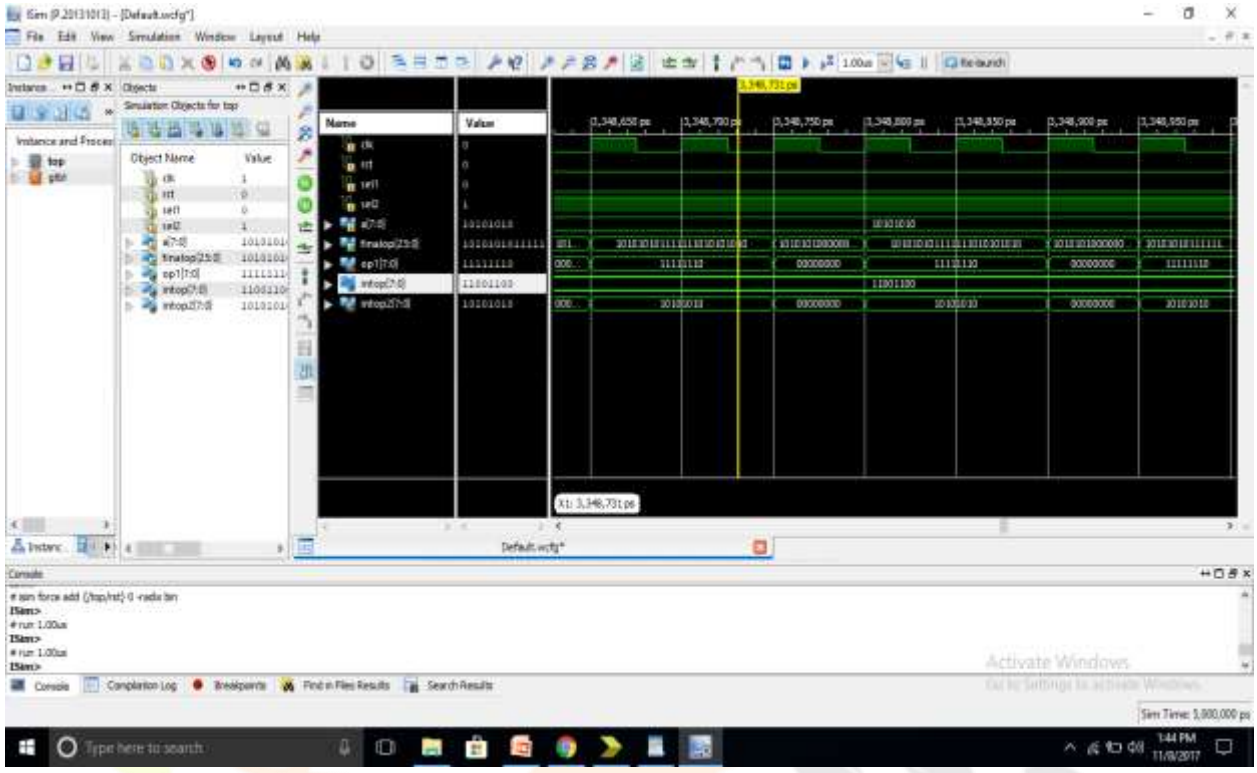
The interleaver design (Khandani 1998) is a key factor, which determines the good performance of a Turbo code. Shannon (1948, 1949) showed that large block-length random codes achieve channel capacity. The 23 pseudo-random interleaver makes the code appear random. In this work the pseudo Random Interleaver has been used. Block interleaver [14] or Matrix interleaver is a most popular interleaver used in digital data transmission that is illustrated in Fig 6. When compared with the other interleavers, it is very simple and easy to design and implement. A block interleaver writes data in a matrix row wise from left to right and top to bottom. After all the information bits are writing into a matrix, it reads the data in column wise from top to bottom and left to right. The output of the interleaver is applied to the RSC 2.



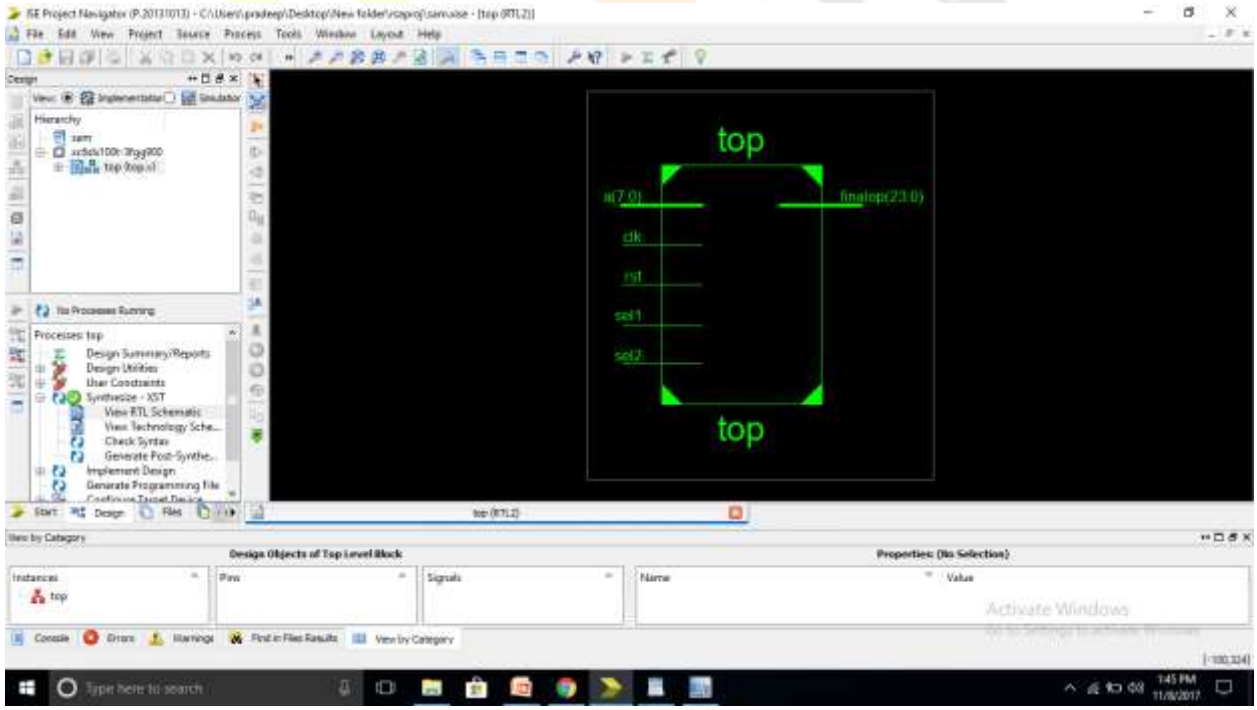
MODIFIED INTERLEAVER DSIGN:

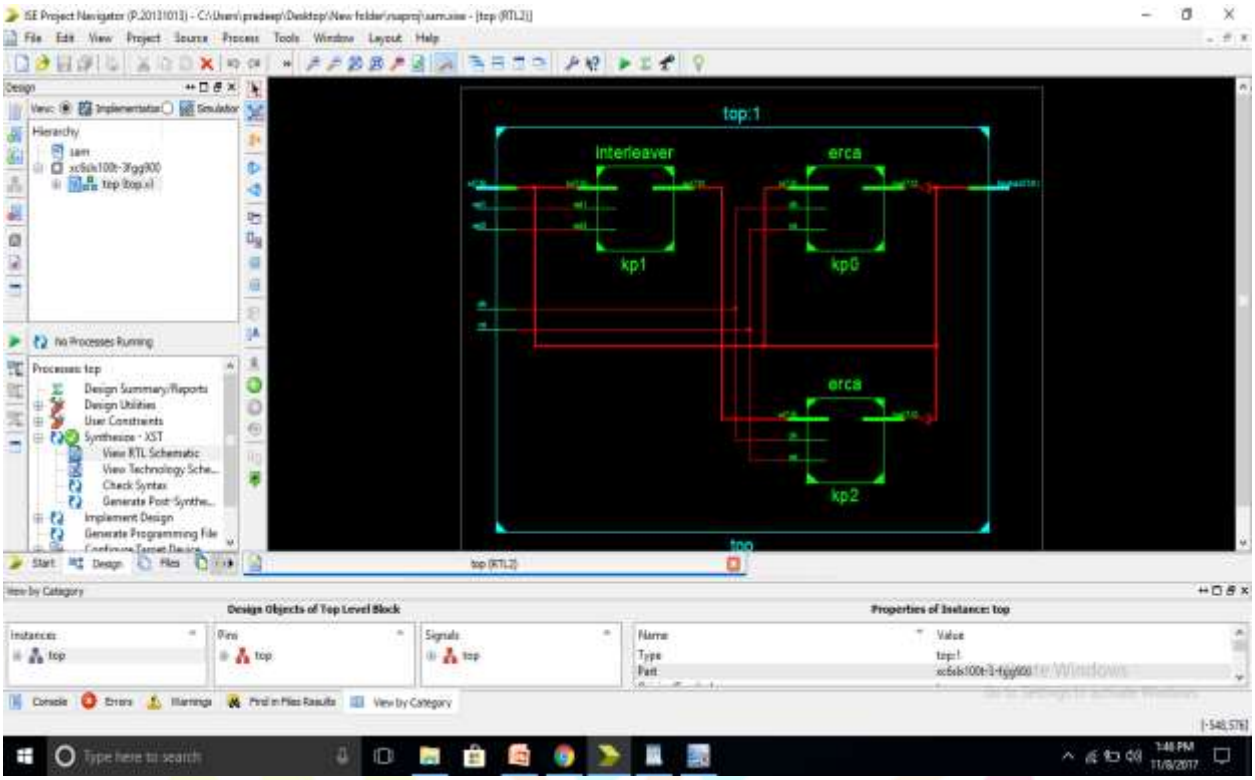


RESULT:



RTL:





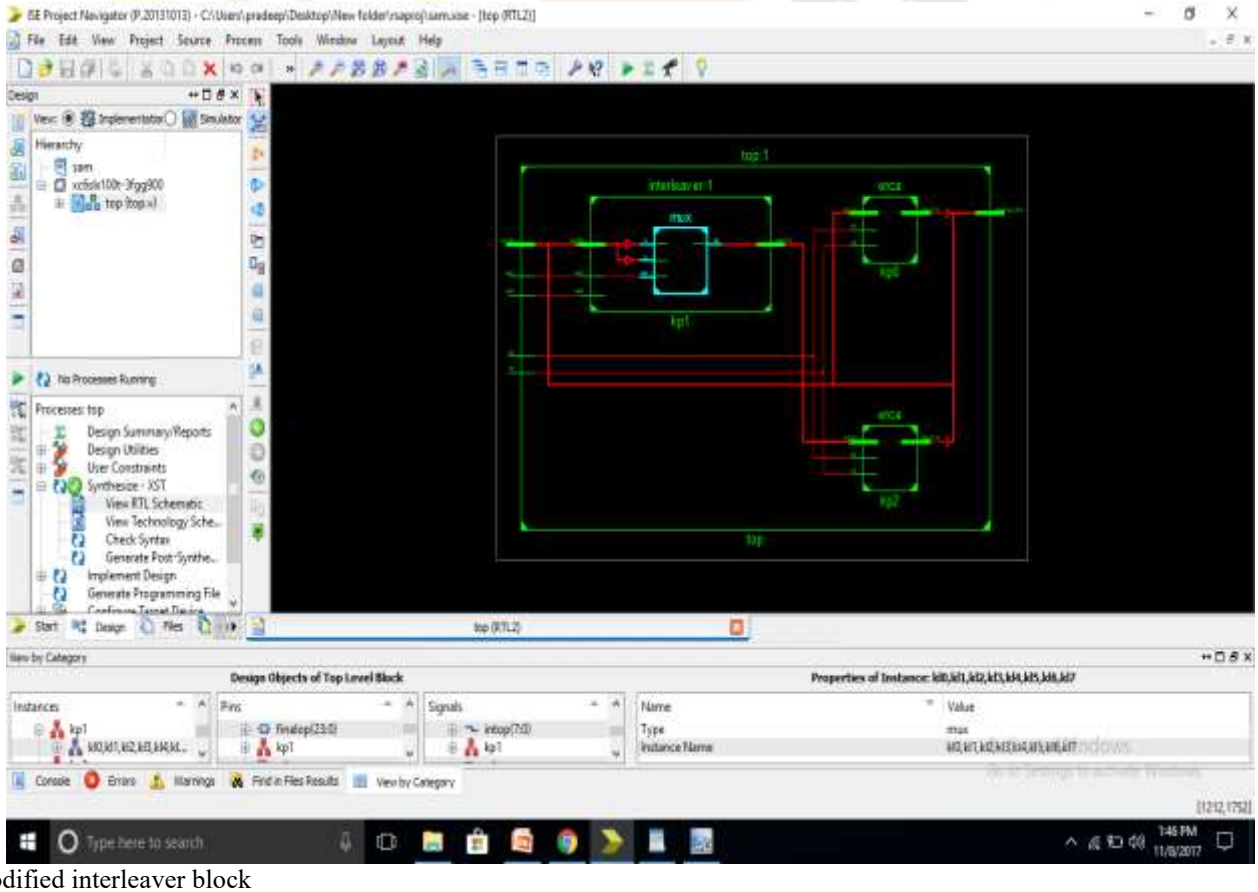


Fig: Modified interleaver block

CONCLUSION:

The Turbo Encoder is designed using Verilog-HDL and simulated using Xilinx ISE 14.7, by considering 8-bit input stream and 16-bit input stream. It is observed through the simulation results that the Turbo Encoder with 8-bit input is more efficient when compare with existing methodologies. These parameters can have negative effects when very low BERs are simulated. Another cause for the limitation in BER performance is a poor interleaver design. Due to highly correlated sequences, the BER decreases to a certain level from the decoding process. SO, interleaver is designed in efficient manner.

REFERENCES:

[1] Jakob Dahl Andersen, "Turbo Codes Extended with Outer BCH Code", Electronics Letters, vol. 32 No. 22, Oct. 1996.
[2] J. Dahl Andersen and V. V. Zyablov, "Interleaver Design for Turbo Coding", Proc. Int. Symposium on Turbo Codes, Brest, Sept. 1997.
[3] Jakob Dahl Andersen, "Selection of Component Codes for Turbo Coding based on Convergence Properties", Annales des Telecommunication, Special issue on iterated decoding, June 1999.
[4] L. R. Bahl, J. Cocke, F. Jelinek and R. Raviv, " Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," IEEE Trans. Inform. Theory, vol IT- 20, pp 284-287, March 1974.
[5] S. Benedetto and G. Montorsi, "Performance Evaluation of Turbo-codes", Electronics Letters, vol. 31, No. 3, Feb. 1995.
[6] S. Benedetto and G. Montorsi, "Serial Concatenation of Block And Convolutional Codes", Electronics Letters, Vol. 32, No. 10, May 1996.
[7] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon Limit Error-correcting Coding and Decoding : Turbo-codes(1)", Proc. ICC 93, pp. 1064-1070, May 1993.
[8] C. Berrou and A. Glavieux, "Near Optimum Error Correcting Coding and Decoding: Turbo Codes", IEEE trans. on Communications, Vol. 44, No. 10, Oct. 1996.
[9] R. J. McEliece, E. R. Rodemich and J.-F. Cheng, "The Turbo Decision Algorithm", Presented at the 33rd Allerton Conference on Communication, Control and Computing, Oct. 1995. 21
[10] L. C. Perez, J. Seghers and D. J. Costello, Jr, "A Distance Spectrum Interpretation of Turbo Codes", IEEE Trans. on Inform. Theory, Vol. 42, No. 6, Nov. 1996.
[11] Steven S. Pietrobon, "Implementation and Performance of a Serial MAP Decoder for use in an Iterative Turbo Decoder", Proc. Int. Symposium on Information Theory, Whistler, Canada, Sept. 1995.

Biography of authors:

Author 1:

KICHAGARE LATHA, She was a PG Research Scholar in Sri Annamacharya Institute of Technology and Science, rajampet, Y.S.R Kadapa, A.P. She was interest in cmos Analog IC design in VLSI. It is a crucial for applications in communications, signal processing, and various consumer electronics. And She also interest in semiconductor memory design and testing and physical design automation in VLSI system. It can create a reliable, high performance memory components suitable for a wide range of applications.

Author 2:

Yeddula sreelatha, she was working as an assistant professor in Sri Annamacharya Institute of Technology and Science, rajampet, Y.S.R Kadapa, A.P. She was interested in digital electronics, VLSI, Digital signal processing, Image processing, AI and ML.

.