Design of Convolutional Codes for varying Constraint Lengths

S. VikramaNarasimhaReddy[1], Charan Kumar .K[2], Neelima Koppala[3]

[1,2]M.Tech(VLSI) Student, [3]Assistant Professor, ECE Department,
SreeVidyanikethan Engineering College (Autonomous), A.Rangampet, Tirupati.

***Abstract*--- Convolutional codes play a vital role in wireless communication with the increase in usage of low-latency applications operating at high data rates. As the current technologies demand high data rates, the design of convolutional codes is necessary for varying lengths. They can be designed as recursive or non-recursive codes hence they represent a state machine. Non-recursive codes prove a better choice for single input based serially communicated applications. This paper concentrates on the design and verification of these codes using XilinxISE 10.1i Tool.**

***Keywords* – Convolutional Codes, Constraint Length, Data Rate, Latency, State machine.**

I. INTRODUCTION

Convolutional codes are introduced in 1955 by Elias. convolutional codes are one of the powerful and widely used class of codes, These codes are having many applications, that are used in deep-space communications, voiceband modems, wireless standards(such as 802.11) and in satellite communications [9] . Convolutional codes are plays a role in low-latency applications such as speech transmission and as constituent codes in Turbo codes.

Convolutional codes are used as inner codes with burst error correcting block codes as outer codes to form concatenated codes [6]. Errors in Viterbi-like decoding algorithms for convolutional codes tend to occur in bursts because they result from taking a wrong path in a trellis. The burst error correcting capability of the outer code is used to recover from such burst error patterns in the decoding of the inner code. Convolutional codes are easy to implement, because we can understand them in many different ways, and there is a way to decode them so as to recover the mathematically most likely message from among the set of all possible transmitted messages. Convolutional codes are a bit type block codes [8]. They involve the transmission of parity bits that are computed from message bits.

II. CONVOLUTIONAL CODES

Convolutional codes protect information by adding redundant bits to any binary data. The convolutional encoder computes each n-bit symbol ($n > k$) of the output sequence from linear operations on the current input k-bit symbol and the contents of the shift register(s). Thus, a rate k/n convolutional encoder processes a k-bit input symbol and computes an n-bit output symbol with every shift register update.

Convolutional codes are commonly specified by three parameters; (n,k,m).

n = number of output bits

k = number of input bits

m = number of memory registers

The quantity k/n is called as code rate. it is a measure of the efficiency of the code [1] . Commonly k and n parameters range from 1 to 8, m from 2 to 10 and the code rate from 1/8 to 7/8 except for deep space applications where coderates as low as 1/100 or even longer have been employed. here The quantity L is called the constraint length of the code and is defined by

Constraint Length, $L = k\,(m\text{-}1)$

The constraint length L represents the number of bits in the encoder memory that affect the generation of the n output bits. The constraint length L is also

referred to by the capital letter K, which can be confusing with the lower case *k*, which represents the number of input bits. In some books K is defined as equal to product the of *k* and *m.* Often in commercial specifications, the codes are specified by (r, K), where r = the code rate *k/n* and K is the constraint length.

III. CONVOLUTIONAL CODES TYPES

Convolutional codes are two types that are Recursive and non-recursive. Figures 1 and 2 illustrate Recursive and Non Recursive convolutional encoders. Convolutional encoders are finite-state machines [4]. Hence state diagrams provide considerable insight into the behavior of convolutional codes.

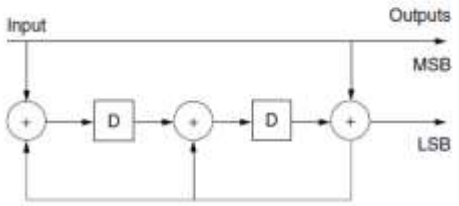

Fig 1: Recursive convolutional encoder.

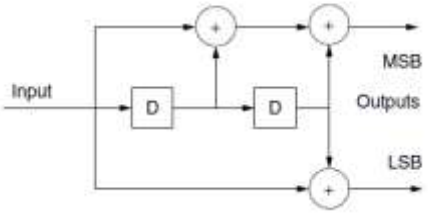

Fig 2: Non Recursive convolutional encoder.

Figures 3 and 4 provide the state diagrams for the encoders of Figs. 1 and 2, respectively. In state diagram the states are labeled so that the least significant bit is the one residing in the leftmost memory element of the shift register. [3] The branches are labeled with the 1-bit (single-bit) input and the 2-bit output separated by a comma. The most significant bit (MSB) of the two-bit output is the bit labeled MSB in Figs. 1 and 2.

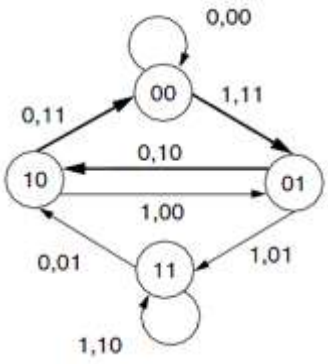

Fig 3 : recursive state diagram

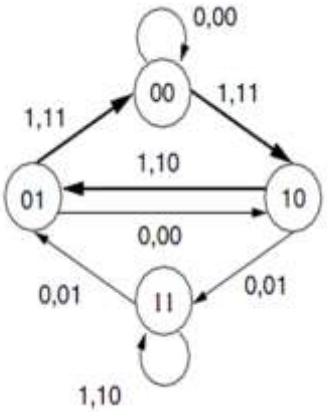

Fig 4 : Non recursive state diagram

Nonrecursiveencoder requires only a single nonzero input. But recursive encoder requires three nonzero inputs. Thus the nonrecursive shift register has a finite impulse response, and the recursive shift register has an infinite impulse response. This difference is not particularly important for convolutional codes decoded with Viterbi, but it is extremely important to convolutional encoders used as constituents in Turbo codes, which are constructed by concatenating convolutional codes separated by interleavers. Only recursive encoders (with infinite impulse responses) are effective constituents in Turbo codes [10]. Thus, equivalent encoders can produce dramatically different performance as constituents in Turbo codes, depending on whether or not they meet the requirement for an infinite impulse response.

IV. (2,1,3) CONVOLUTIONAL ENCODER

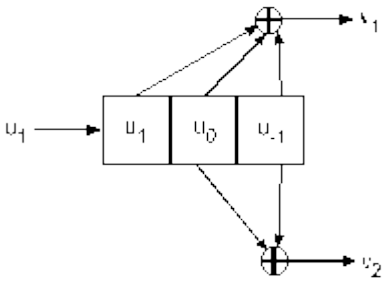

Fig 5 :(2,1,3) convolutional encoder

The (2,1,3) code in Fig. 5 has a constraint length of 2. The last two registers below hold two bits. The very first register holds the incoming bit. This means that 2 bits or 4 different combinations of these bits can be present in these memory registers. These 4 different combinations determine what output we will get for v_1 and v_2, the coded sequence. The number of combinations of bits in the shaded registers are called the states of the code and are defined by

$$\text{Number of states} = 2^L$$

where L = the constraint length of the code and is equal to k (m - 1).

Think of states as sort of an initial condition. The output bit depends on this initial condition which changes at each time tick.

Let's examine the states of the code (2,1,3) shown above. This code outputs 2 bits for every 1 input bit. It is a rate ½ code. Its constraint length is 2. The total number of states is equal to 4 [2].

The eight states of this (2,1,3) code are: 00, 01, 01, 11. And the state diagram is shown in Fig 6.

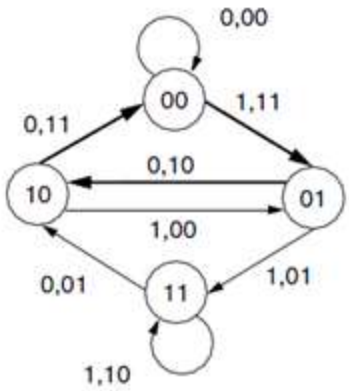

Fig 6 : (2,1,3) state diagram.

V. (2,1,4) CONVOLUTIONAL ENCODER

The (2,1,4) code in Fig. 7 has a constraint length of 3. The shaded registers below hold these bits. The un shaded register holds the incoming bit. This means that 3 bits or 8 different combinations of these bits can be present in these memory registers. These 8 different combinations determine what output we will get for v_1 and v_2, the coded sequence.

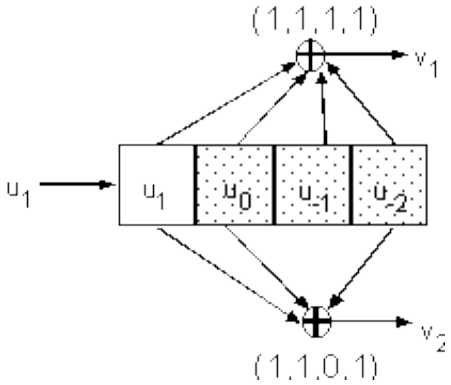

Fig 7 : (2,1,4) convolutional encoder

The number of combinations of bits in the shaded registers are called the states of the code and are defined by

$$\text{Number of states} = 2^L$$

where L = the constraint length of the code and is equal to k (m - 1).

Think of states as sort of an initial condition. The output bit depends on this initial condition which changes at each time tick.

Let's examine the states of the code (2,1,4) shown above. This code outputs 2 bits for every 1 input bit. It is a rate ½ code. Its constraint length is 3. The total number of states is equal to 8.

The eight states of this (2,1,4) code are: 000, 001, 010, 011, 100, 101, 110, 111. And the state diagram is as shown in Fig 8.

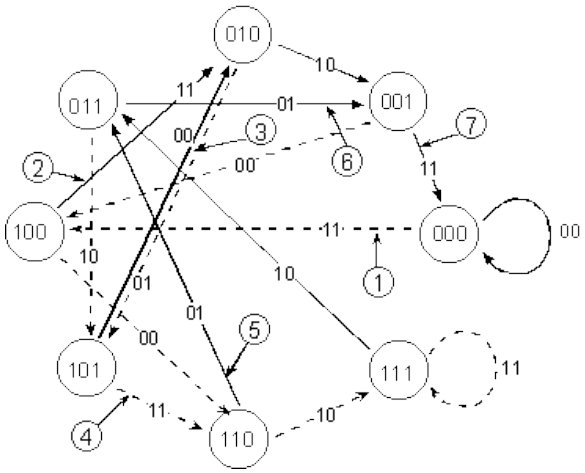

Fig 8: (2,1,4) state diagram

VI. (2,1,7) CONVOLUTIONAL ENCODER

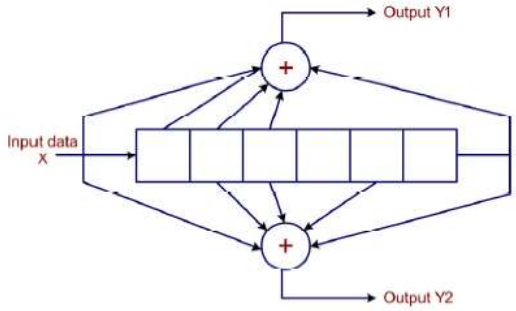

Fig 9 : (2,1,7) convolutional encoder

The (2,1,7) code in Fig. 9 has a constraint length of 6. The shaded registers below hold two bits. The unshaded register holds the incoming bit. This means that 6 bits or 64 different combinations of these bits can be present in these memory registers. These 64 different combinations determine what output we will get for y_1 and y_2, the coded sequence. The number of combinations of bits in the shaded registers are called the states of the code and are defined by

$$\text{Number of states} = 2^L$$

where L = the constraint length of the code and is equal to k (m - 1).

Think of states as sort of an initial condition. The output bit depends on this initial condition which changes at each time tick.

Let's examine the states of the code (2,1,7) shown above. This code outputs 2 bits for every 1 input bit. It is a rate ½ code. Its constraint length is 7. The total number of states is equal to 64. The sixty four states of this (2,1,7) code are: 000000,0000 01, 000010,…….111111 11. State table is shown in Table 1.

TABLE I: (2,1,7) state table

Input	Present state	Next state	output
0	000000	000000	00
1	000000	100000	11
0	000001	000000	11
1	000000	100000	00
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
0	111111	011111	00
1	111111	111111	11

VII. DECODING CONVOLUTIONAL CODES

There are three major families of decoding algorithms for convolutional codes: sequential, Viterbi, and maximum a posteriori (MAP). Viterbi originally described the decoding algorithm that bears his name in 1967 [6]. And also Forney's work introducing the trellis structure and showing that Viterbi decoding is maximum-likelihood in the sense

that it selects the sequence that makes the received sequence most likely.

In 1974, Bahl et al. proposed MAP decoding, which explicitly minimizes bit (rather than sequence) error rate. Compared with Viterbi, MAP provides a negligibly smaller bit error rate (and a negligibly larger sequence error rate). These small performance differences require roughly twice the complexity of Viterbi, making MAP unattractive for practical decoding of convolutional codes [5]. However, MAP decoding is crucial to the decoding of Turbo codes.

When convolutional codes are used in the traditional way (not as constituents in Turbo codes), they are almost always decoded using some form of the Viterbi algorithm, and the rest of this section focuses on describing it. The goal of the Viterbi algorithm is to find the transmitted sequence (or codeword) that is closest to the received sequence [7]. As long as the distortion is not too severe, this will be the correct sequence.

VIII. RESULTS AND DISCUSSION

The designed convolutional codes are developed using Verilog HDL and are simulated and synthesized in XilinxISE10.1i environment. The FPGA device selected is Spartan 3E with XC3S500E family, fg320 package and -5 speed grade.

The simulations are carried for various inputs and outputs of the convolutional encoders.The simulation results for (2,1,3) convolutional code is shown in Fig.10 where the obtained output sequence is 00111000010111 for the given input sequence of 01011100.

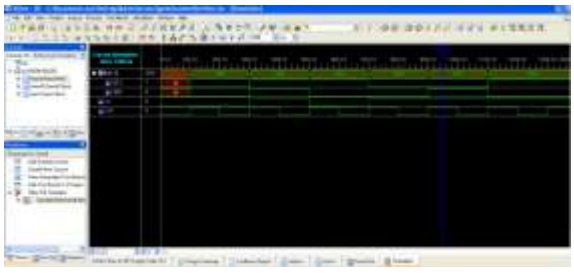

Fig.10: The simulation output of (2,1,3) Convolutional code

The simulation results for (2,1,4) convolutional code is shown in Fig.11 where the obtained output sequence is 00111101111001 for the given input sequence of 01011100.

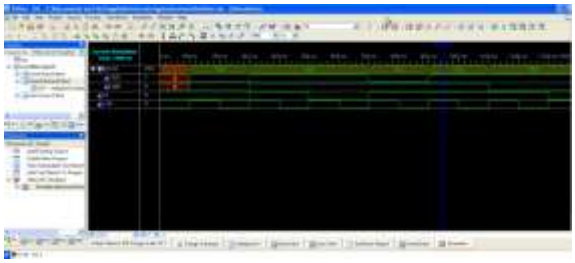

Fig.11: The simulation output of (2,1,4) Convolutional code

The simulation results for (2,1,7) convolutional code is shown in Fig.12 where the obtained output sequence is 0011100010101111 for the given input sequence of 01011100.

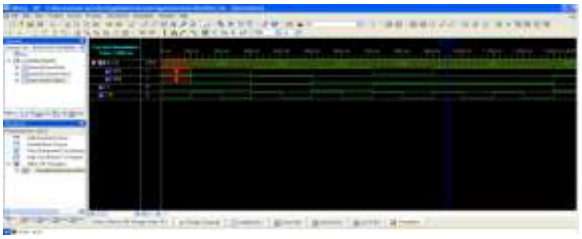

Fig.12: The simulation output of (2,1,7) Convolutional code

The obtained synthesis results are represented in Table.2 as shown below. With same code rate, it compares the codes for area i.e., in terms of slices, flip-flops, LUTs (look up tables) and the critical path delay.

TABLE II: Synthesis Results of the codes

Convolutional Codes / Parameters	**(2,1,3)**	**(2,1,4)**	**(2,1,7)**
Code rate	½	½	½
Constraint Length	2	3	6
No.of.LUTs	2	5	14
No.of.slices	2	3	10
Sliced F/Fs	4	5	8
Combinational Path delay	2.589ns	2.676ns	4.645ns

From above table, it is clear that even though the code rate is kept same, as the constraint length is increasing the combinational path delay and the

number of output bits increase with an optimum increase in area. Hence, optimally the code rates can be used based on applications.

IX. CONCLUSION

The design of Convolutional codes are basically encoders which are very important for extremely low error probabilities used at high data rates of wireless communication applications. For the basic design of the codes can bedeveloped based on code rate and constraint length. As the constraint length is increasing the parallel implementationhas to be used for feasibility and high data rates with less delay and optimum area utilization. The non-recursive and recursive codes can be designed based on the application of single or multiple inputs considered at a time. These codes are suitable for high data rate based wireless communication applications which include deep space communications, voice band modems, satellite communication, etc.

REFERENCES

[1] W. W. Peterson and E. J. Weldon, Jr., Error Correcting Codes, 2nd ed. Cambridge, MA: The MIT Press, 1972.

[2] S Lin and D.J. Costello, Error control coding. Englewood Cliffs, NJ: Prentice Hall, 1982.

[3] A. M. Michelson and A.H. Levesque, Error control techniques for digital communication.New & Sons, 1985.

[4] V. Pless, Introduction to the theory of Error-Correcting codes, 3rded. New York: john Wiley & Sons,1998.

[5] A. J.Viterbi, "Error bounds for convolutional coding and an asymptotically optimum decoding algorithm", IEEE Tran. on Inform. Theory, Vol. 2, Pp. 260-269, Apr. 1967.

[6] A. J. Viterbi, "Convolutional codes and their performance in communication systems", IEEE Transaction Communication Technology, Vol.19, pp. 751-772, Oct. 1971

[7] A. J. Viterbi and J. K. Omura, "Principles of Digital Communications and Coding", McGraw-Hill, NY, 1979.

[8] B. Sklar, Digital Communications: "Fundamentals and Applications", 2nd edition, Prentice-Hall, Upper Saddle River, N J, 2001. 4 G. C. Clark Jr. and J. B. Cain, "Error-Correction Coding for Digital Communications", Plenum Press, NY, 1981.

[9] J. H. Yuen, "Modulation and Coding for Satellite and Space Communications", IEEE Proceedings, Vol. 78, No. 7, pp. 1250-1266, July 1990.

[10] Wang Xinmei, Xiao Guozhen, Error correcting code - Principles and Methods, Xidian University Press, 2001 :378-458.