TSG-RAN WG-1
Meeting # 6
Espoo, Finland
July 13-16, 1999

TSG-R1#6(99)950

Title: Rate Matching Puncturing for 8-PCCC and Convolutional Code

Source: Nortel Networks[1]

0.0 Summary

In this contribution, we investigate the rate matching puncturing for 8-state PCCC and K=9 convolutional code. We have demonstrated the impact of parity loss due to puncturing to the Turbo decoder performance. And we propose Turbo interleaving based puncturing scheme and compared with Fujitsu and LGIC proposals. Based on the simulation results, we show that:

- For R=1/2 8-PCCC Code puncturing, with frame size 320, the proposed scheme has 0,35dB gain @BER=10^{-5} and 0.25dB gain @FER=10^{-4}.
- Applying current working assumption for convolutional code puncturing to 8-PCCC code puncturing which involves the systematic bits will always cause the performance loss. e.g. with frame size=320 20% puncturing rate, the loss is 0.25dB @BER=10^{-5}
- For R=1/3 8-PCCC rate matching puncturing, the prosed scheme is consistently better other proposals in **all** simulation cases, in particular the gain are:

  0.2dB, @FER=01^{-4}, with frame size=321, puncture rate=10%

  0.3dB, @FER=10^{-4}, with frame size=321, puncture rate=15%

  0.25dB,@ FER=10^{-3},with frame size 323, puncture rate=15%

In this contribution, we propose to use Turbo interleaver based puncturing scheme for 8-PCCC

- Code Rate R=1/2 8-PCCC code puncturing, and replace the current Burrou puncturing working assumption
- Code Rate R=1/3 8-PCCC rate matching puncturing.
- Code Rate R=1/2 8-PCCC rate matching puncturing
- To separate the systematic bits from Rate matching puncturing.

For K=9, R=1/3 convolutional code, we propose

- To puncture polynomial G0 and G1 and keep G2 separate from puncturing.

We propose to unify the Turbo code and Convolutional code puncturing scheme into a single framework.

1 Contact Persons: Wen Tong, Catherine Gauthier, Nortel Networks
1 Place des Freres Montgolfier, 78042 Guyancourt, France
Tel:+33 1 39 44 57 47
Fax:+33 1 39 44 50 12
e_mail: wentong@nortelnetworks.com / gauth@nortelnetworks.com

1.0 Puncturing of Turbo Code

The puncturing of basic rate 1/3 8-state PCCC can be classified into types: (1) Fixed puncturing parity bits to obtained higher rate codes; (2) Flexible rate matching puncturing. Optimal puncturing in (1) is investigated by several authors, while optimal puncturing in (2) is studied by Fujitsu and LGIC. However, the joint optimization of both (1) and (2) in both up-link and down link remains an open issue.

1.1 Fixed Rate Puncturing 8-State PCCC Code

Different code rates can be generated from rate R=1/3 through puncturing. The Berrou puncturing results in a R=1/2 code, the puncturing pattern is listed in Table 1.

TABLE 1. Berrou Puncturing for R=1/2 Code

x	x(0)	x(1)	x(2)	x(3)	x(4)	x(5)	x(6)	x(7)	x(8)	x(9)	x(10)
y_1	$y_1(0)$	$y_1(1)$	$y_1(2)$	$y_1(3)$	$y_1(4)$	$y_1(5)$	$y_1(6)$	$y_1(7)$	$y_1(8)$	$y_1(9)$	$y_1(10)$
y_2	$y_2(0)$	$y_2(1)$	$y_2(2)$	$y_2(3)$	$y_2(4)$	$y_2(5)$	$y_2(6)$	$y_2(7)$	$y_2(8)$	$y_2(9)$	$y_2(10)$

However, there is a degradation by using such a puncturing, this is so-called parity loss puncturing, see Figure 1. In this case, both parity bits corresponding to a given systematic bit are punctured, the impact of such a parity loss can be explain in Figure 2. Such a loss is high if Berrou puncturing scheme is used. see Figure 4.

FIGURE 1. Parity Loss Caused by Puncturing

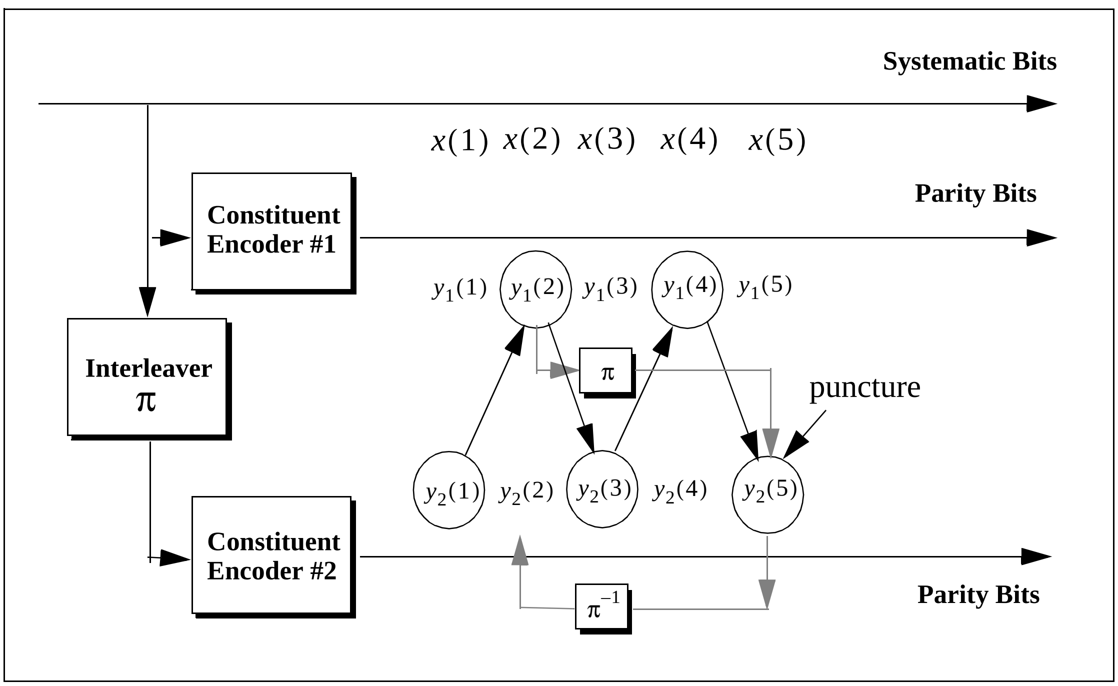


FIGURE 2. Impact of Turbo Decoder w.r.t. Parity Bit Loss

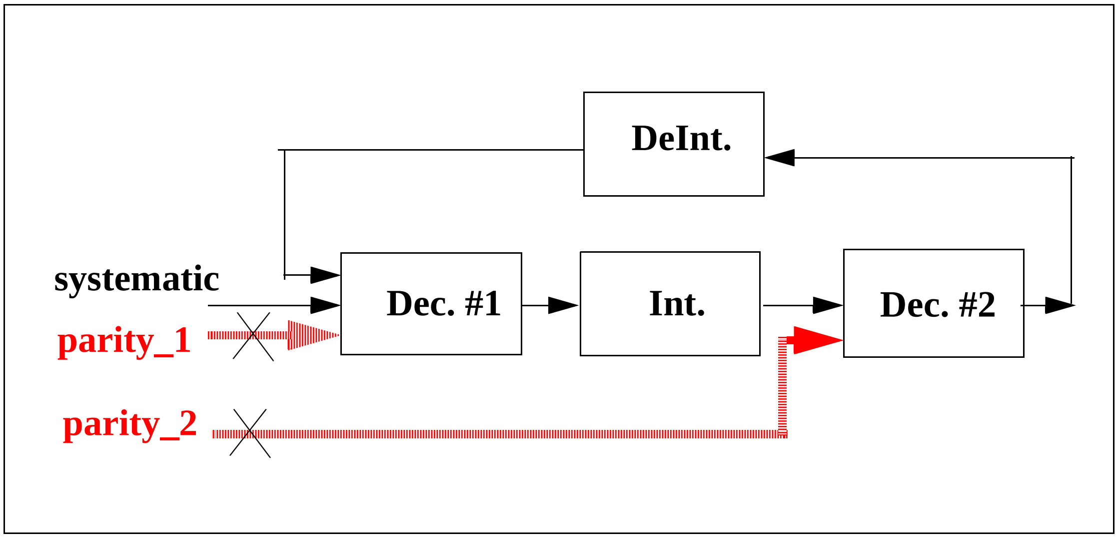
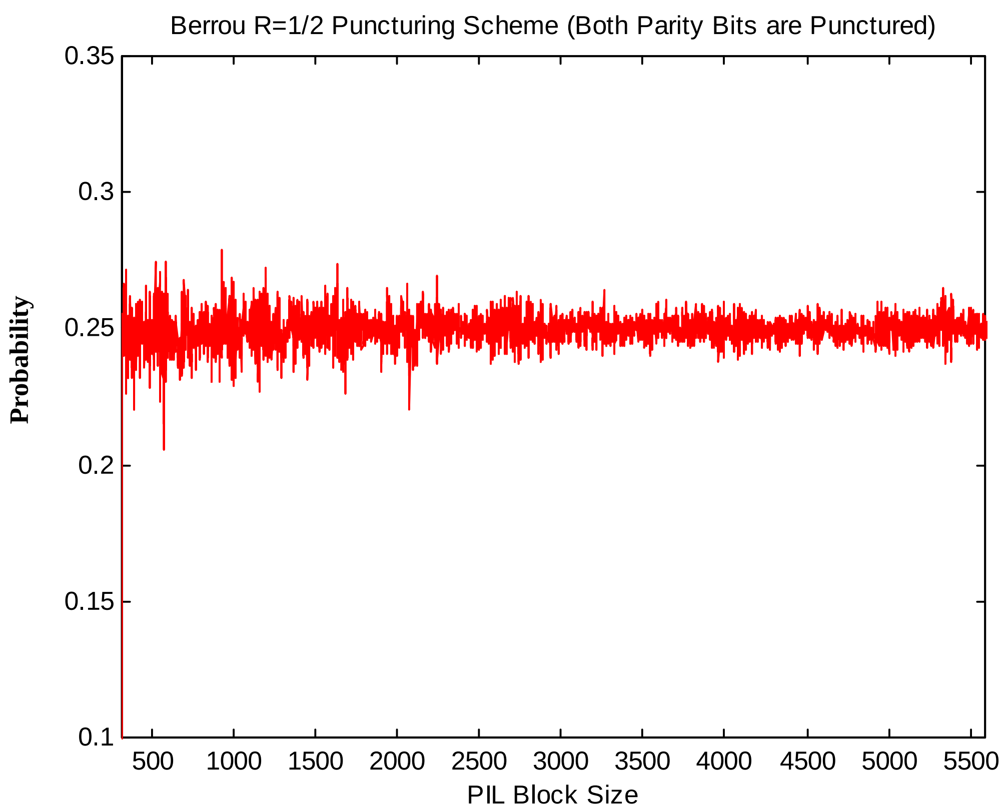


FIGURE 3. 3GPP-PIL Interleaver Parity Loss (R=1/2)



Therefore to avoid such a puncturing, we use even puncturing on parity bit#1 and apply the non-punctured bits position in parity bit#1 to Turbo interleaver and punctured the corresponding position on bit#2. The proposed Turbo code puncturing scheme is listed in Table.2

TABLE 2. Proposed Fixed Puncturing for R=1/2 Code

x	x(0)	x(1)	x(2)	x(3)	x(4)	x(5)	x(6)	x(7)	x(8)	x(9)	x(10)
y_1	$y_1(0)$	$y_1(1)$	$y_1(2)$	$y_1(3)$	$y_1(4)$	$y_1(5)$	$y_1(6)$	$y_1(7)$	$y_1(8)$	$y_1(9)$	$y_1(10)$
address		**1**		**3**		**5**		**7**		**9**	
Int[address]	**9**	**7**					**5**	**1**		**3**	
y_2	$y_2(0)$	$y_2(1)$	$y_2(2)$	$y_2(3)$	$y_2(4)$	$y_2(5)$	$y_2(6)$	$y_2(7)$	$y_2(8)$	$y_2(9)$	$y_2(10)$

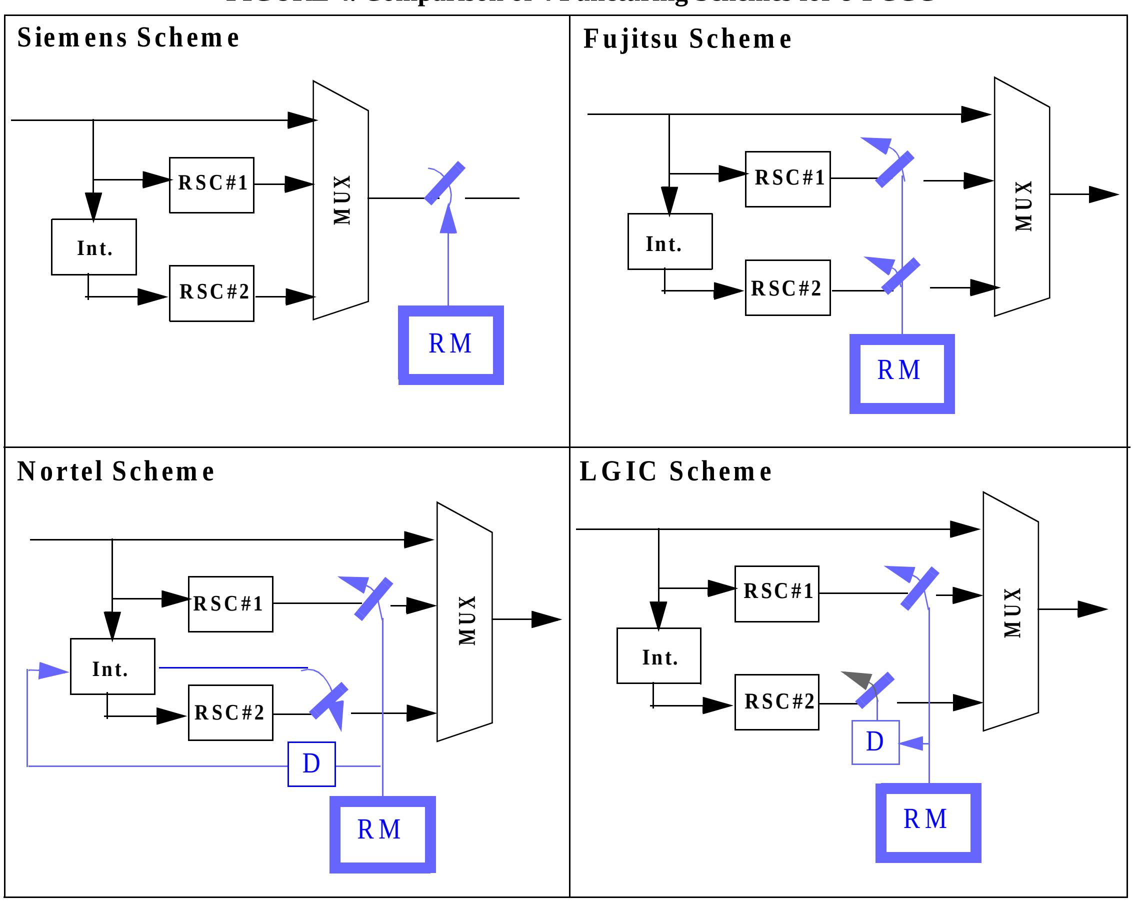
The gain of proposed R=1/2 puncturing pattern over the Berrou puncturing pattern is demonstrated in Section 2.0

1.2 Flexible Rate Matching Puncturing

1.2.1 Rate Matching for Code Rate 1/3

The objective of rate matching puncturing for Turbo code is to allow to apply arbitrary puncturing rate up to 20% of the coded block size. We propose to use the rate matching algorithm to determine the P1 position of puncturing for parity bits#1 by using Siemens algorithm, and to use the un-punctured bits in parity bits#1 adjacent to the punctured bits mapped by using turbo interleaver onto parity bits#2 positions. The principle of interleaved puncturing scheme is listed in Table 3.

FIGURE 4. Comparison of 4 Puncturing Schemes for 8-PCCC

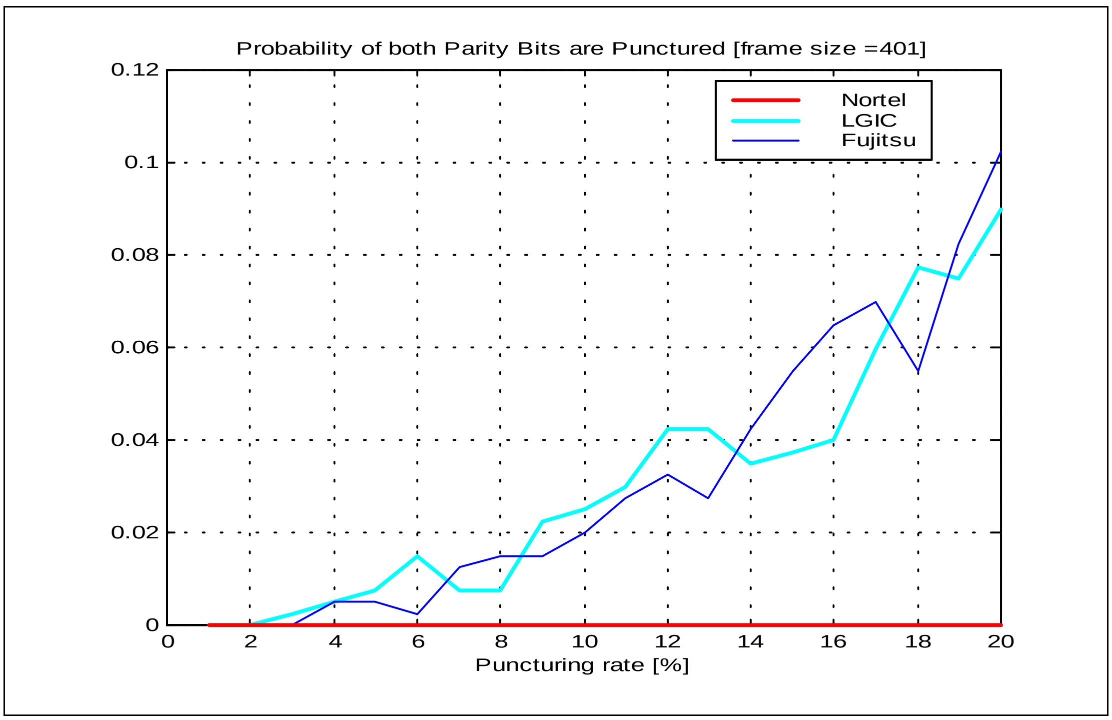


For the parity loss defined in section 1.1, we can show for both LGIC and Fujitsu scheme suffer from such a loss. See Figure 3. As we can see, for Fujitsu and LGIC puncturing schemes, the parity loss is puncturing rate dependent. For Nortel scheme, it is guaranteed, for code rate 1/3 case, there are no parity loss.

TABLE 3. Proposed Flexible Puncturing for R=1/3 Code

x	x(0)	x(1)	x(2)	x(3)	x(4)	x(5)	x(6)	x(7)	x(8)	x(9)	x(10)
y_1	$y_1(0)$	$y_1(1)$	$y_1(2)$	$y_1(3)$	$y_1(4)$	$y_1(5)$	$y_1(6)$	$y_1(7)$	$y_1(8)$	$y_1(9)$	$y_1(10)$
address		**1**				**5**				**9**	
Int[address]	**9**						**5**	**1**			
y_2	$y_2(0)$	$y_2(1)$	$y_2(2)$	$y_2(3)$	$y_2(4)$	$y_2(5)$	$y_2(6)$	$y_2(7)$	$y_2(8)$	$y_2(9)$	$y_2(10)$

FIGURE 5. Comparison of LGIC/Fujitsu Schemes



1.2.2 Rate Matching for Code Rate 1/2

If the base Turbo code rate is 1/2, then the flexible rate matching puncturing up to 20% is not an easy task. We propose to use the same principle as for the R=1/3 for the flexible puncturing of R=1/2 code as listed in Table 3.

TABLE 4. Proposed Rate Matching Puncturing for R=1/2 Code

x	x(0)	x(1)	x(2)	x(3)	x(4)	x(5)	x(6)	x(7)	x(8)	x(9)	x(10)
y_1	$y_1(0)$	$y_1(1)$	$y_1(2)$	$y_1(3)$	$y_1(4)$	$y_1(5)$	$y_1(6)$	$y_1(7)$	$y_1(8)$	$y_1(9)$	$y_1(10)$
address1		1		3		5		7		9	
Int[address1]	9	7					5	1		3	
address2			2				6				
Int[address2]				6					2		
y_2	$y_2(0)$	$y_2(1)$	$y_2(2)$	$y_2(3)$	$y_2(4)$	$y_2(5)$	$y_2(6)$	$y_2(7)$	$y_2(8)$	$y_2(9)$	$y_2(10)$

1.3 Flexible Puncturing of Convolutional Code

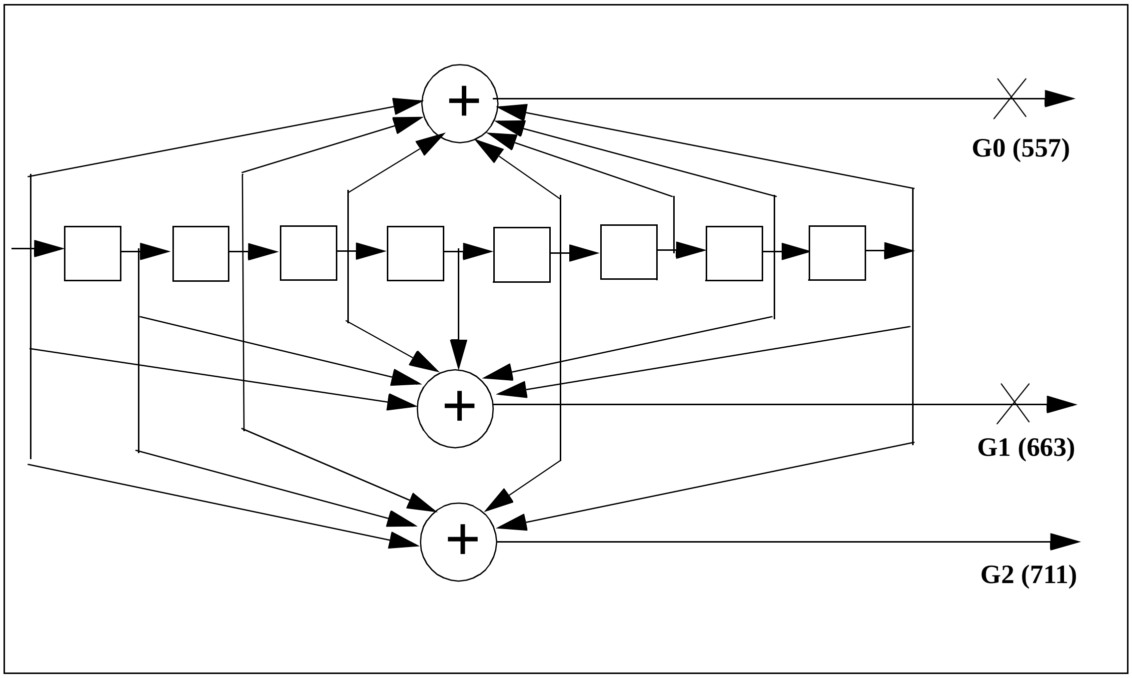
In current 3GPP working assumption, the rate matching puncturing for convolutional code is applied on the encoder output coded bits stream based on Siemens rate matching algorithm. It is claimed that such a puncturing scheme is optimum, however, in this contribution, we will demonstrate working assumption is not necessary optimum, or at least it is not a unique way to achieve optimum performance. In order to have a unified puncture scheme with Turbo code, we present the following observation listed in Table 3

TABLE 5. 3GPP Convolutional Code

	(0,1)	**(0,2)**	**(1,2)**	**(0,1,2)**
polynomials	(557/663)	(557/711)	(663/711)	(557/663/711)
d_{min}	9	10	12	18

Form Table 3, we can see that the polynomial 557 is the weakest in terms of free distance, therefore, the puncturing should be applied to this polynomial at least, we can show if the puncturing is applied to the polynomial 557 and 663, we can achieve the best performance. The performance comparison for Nortel, Siemens is shown in Section 2.0

FIGURE 6. Flexible Puncturing of R=1/3 Convolutional Code



1.4 Complexity Analysis

It can be seen from Figure 4 that 4 proposed schemes have the same complexity. Basically, for Turbo interleaving, we have to buffer the entire information bit block anyway, in this case, puncturing is a simple address labeling operation for Nortel, Fujitsu and LGIC schemes.

1.5 Unified Flexible Puncturing for Turbo and Convolutional Code

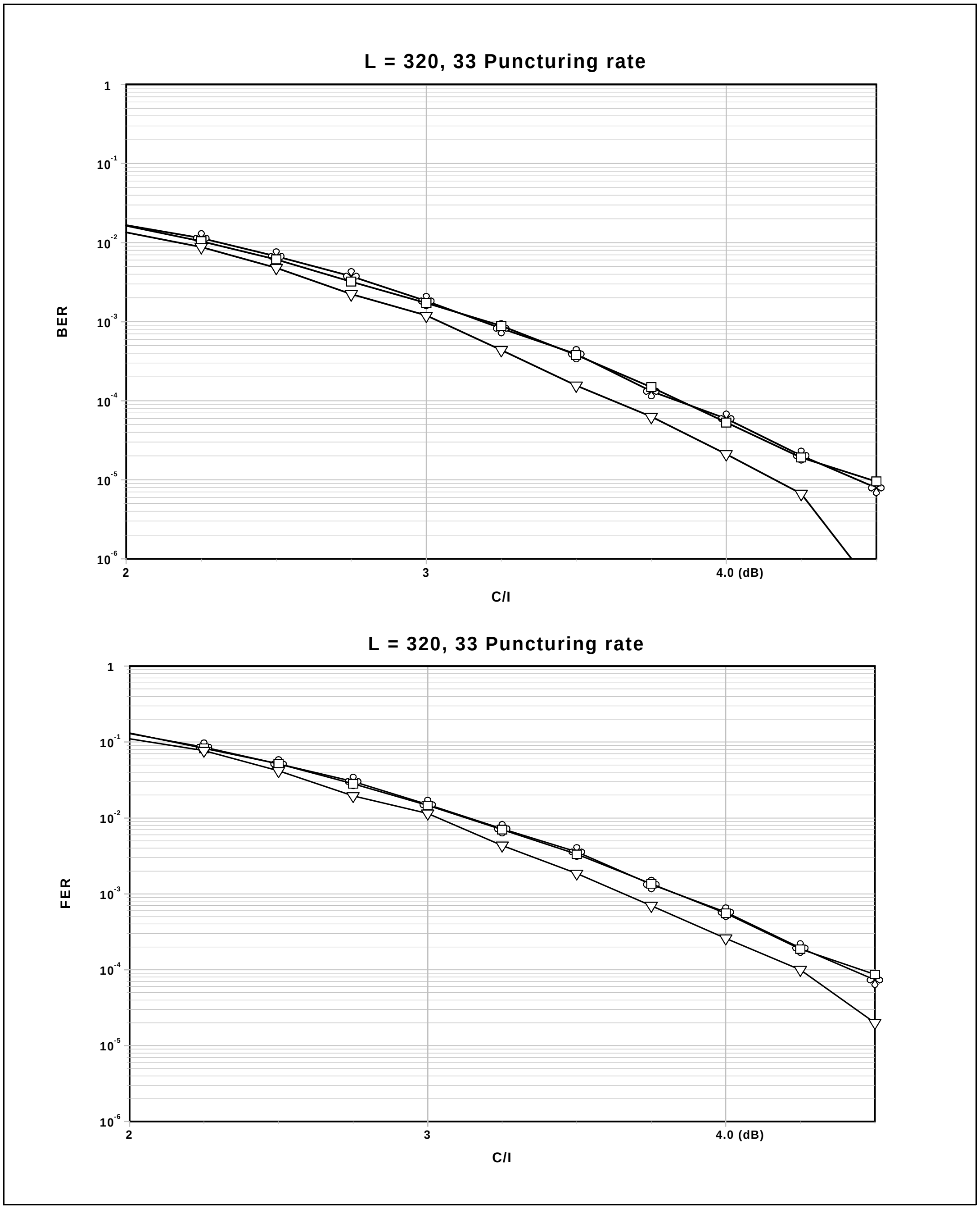
As we can see from Section 1.2 and 1.3, the best flexible puncturing scheme for 8-PCCC Turbo code is to keep the systematic bit stream un-punctured, and to keep the branch of G2 for the convolutional code un-punctured.

2.0 Performance Simulation Results

2.1 Simulation Result on Optimum R=1/2 Code Puncturing

In Figure 7 we show that the Nortel interleaver based puncturing is significantly better than the Berrou puncturing. This confirms our analysis of the parity loss impact due to puncturing.

FIGURE 7. Comparison of Berrou Puncturing and Proposed Scheme



2.2 Evaluation of Parity Loss for Flexible Puncturing of 8-PCCC

Figure 7-11 are the parity loss evaluation for 8-PCCC for different frame size. It can be seen that such a loss different w.r.t. different puncturing rate, block size and different schemes. Nortel interleaving puncturing scheme can avoid such a loss for any puncturing rate and block size.

FIGURE 8. Puncturing of Frame Size of 320 and 321

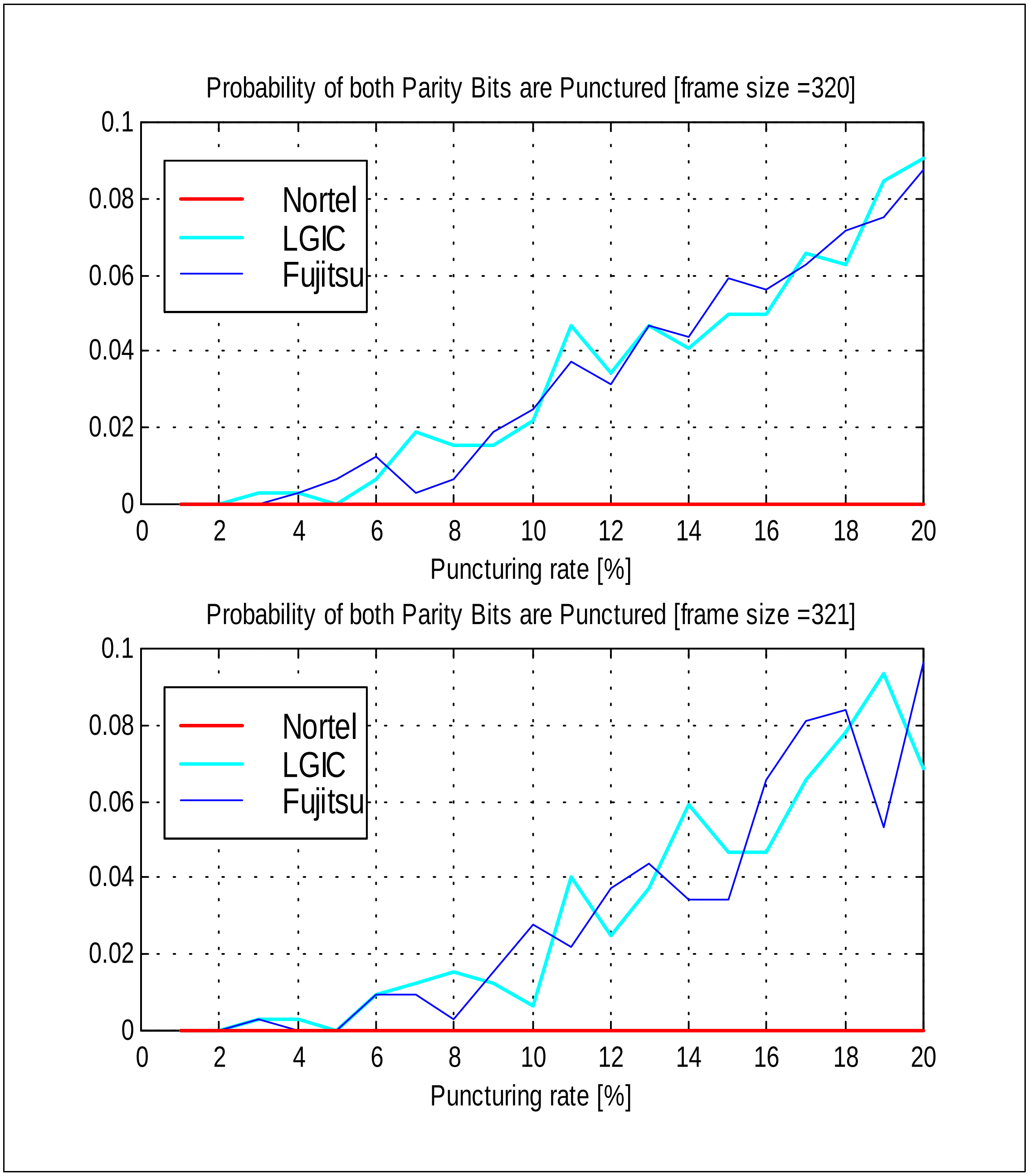


FIGURE 9. Puncturing of Frame Size of 322 and 323

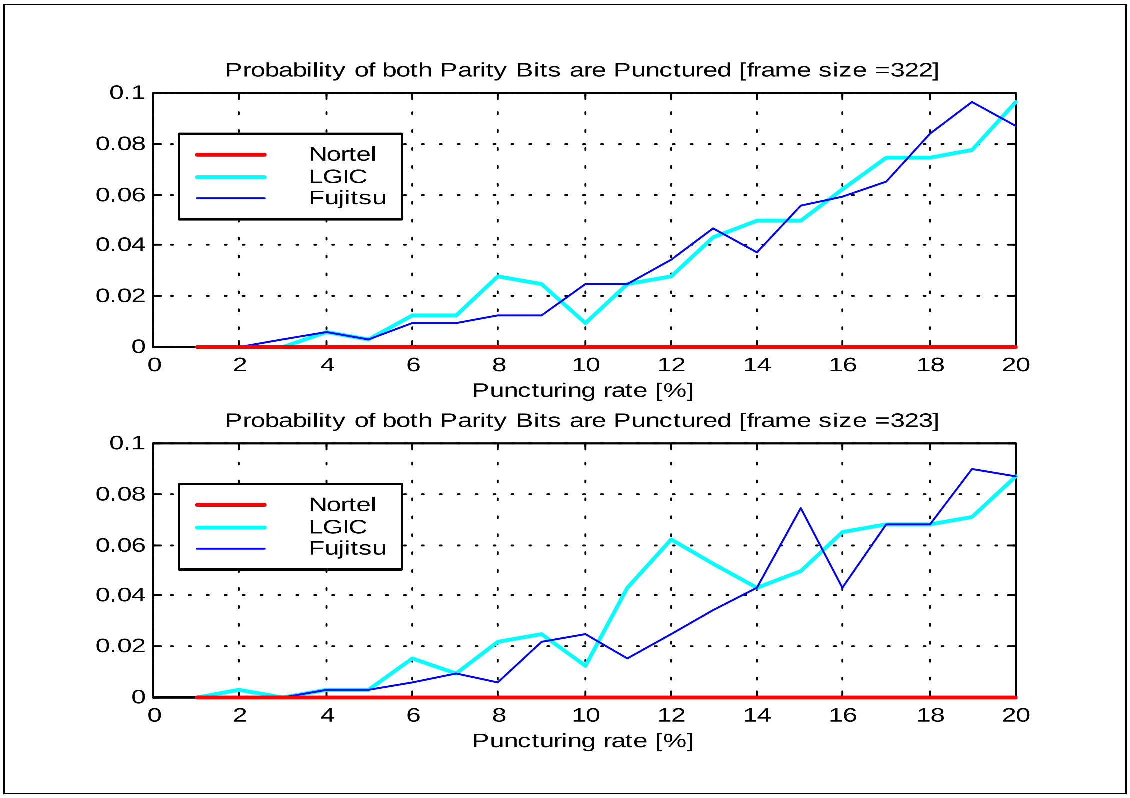


FIGURE 10. Puncturing of Frame Size of 640 and 641

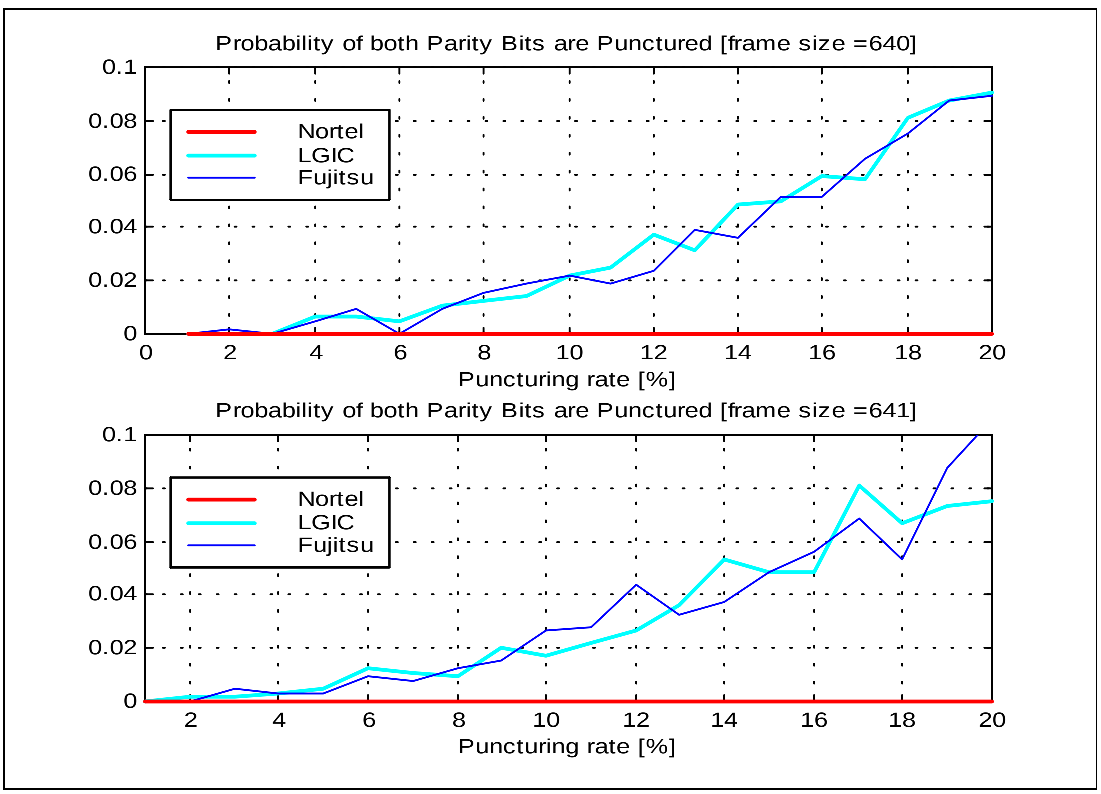
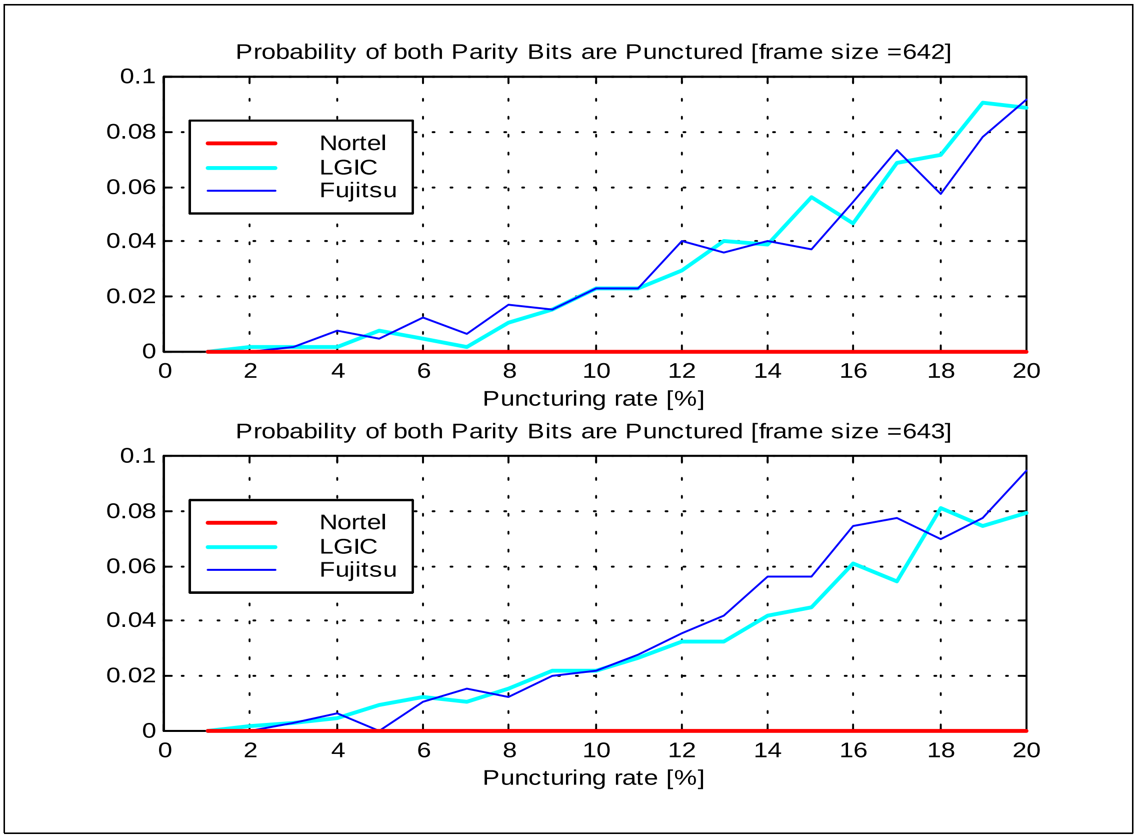


FIGURE 11. Puncturing of Frame Size of 642 and 643



As we can see parity loss due to the puncturing is different with puncturing rate and block size for Fujitsu and LGIC scheme, in the performance simulation, we can observe the similar result as the parity loss that for different puncturing rate one scheme could be better than the other, we can see also the proposed puncturing scheme has no parity loss in all the case, this leads to a consistent better performance than Fujitsu and LGIC, simulation results confirm this claim.

2.3 Simulation Results Comparison of 8-PCCC Puncturing Schemes

2.3.1 Simulation Condition

The simulation condition is based on the AWGN environment agreed in selection of Turbo Code puncturing scheme.

FIGURE 12. Information Block Size=320, Punctured Bits=192

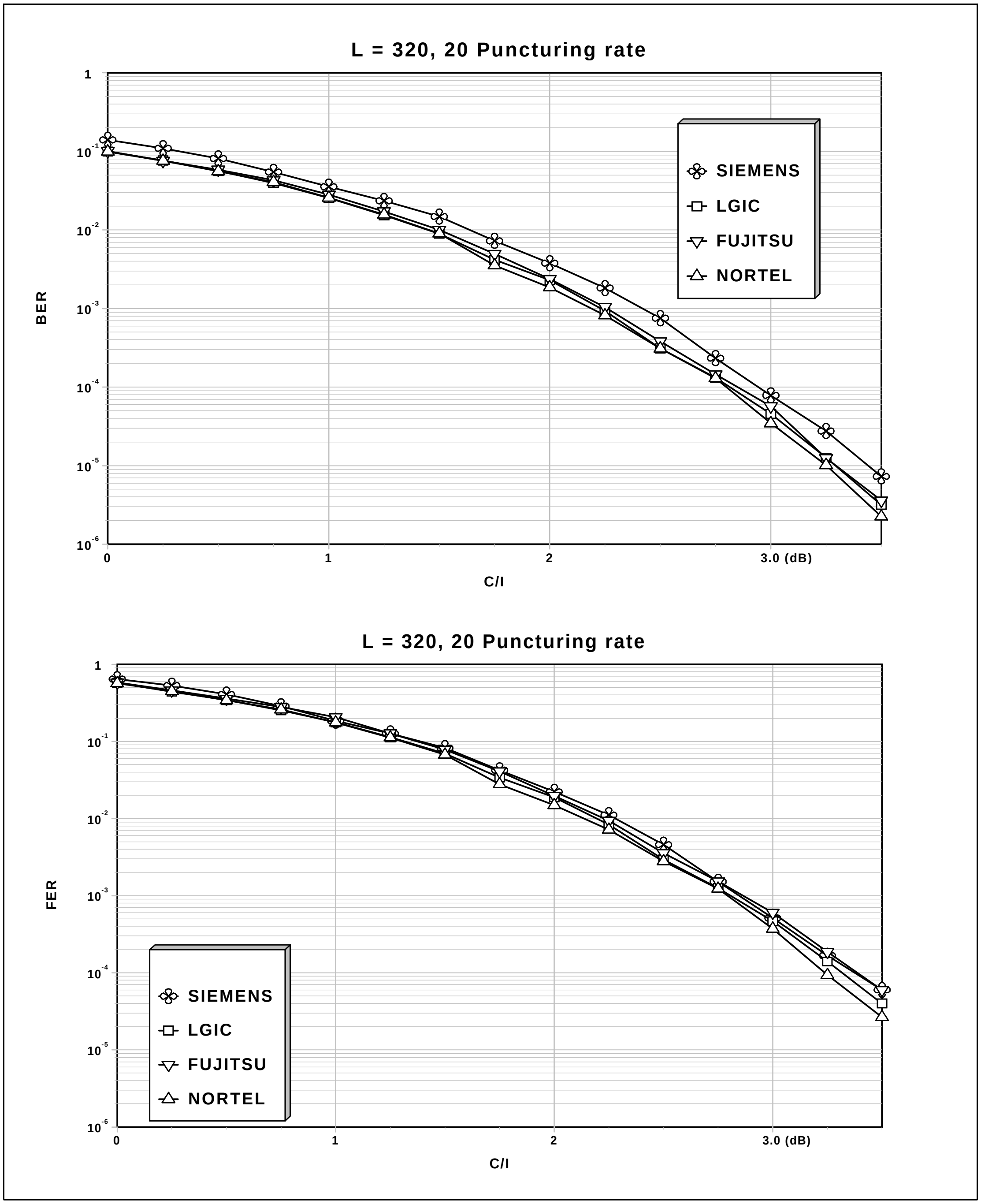


FIGURE 13. Information Block Size=320, Punctured Bits=97

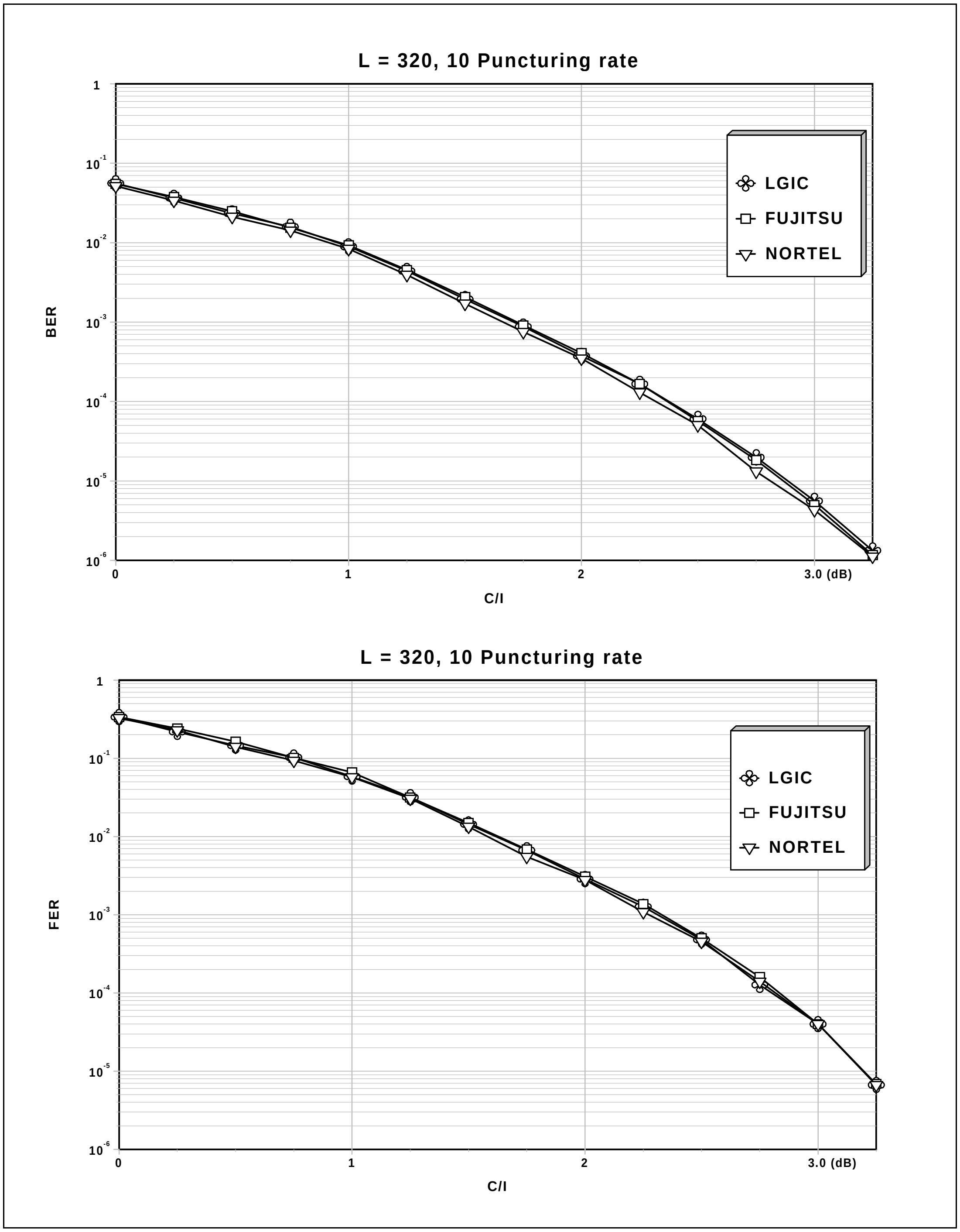


FIGURE 14. Information Block Size=321, Punctured Bits=97

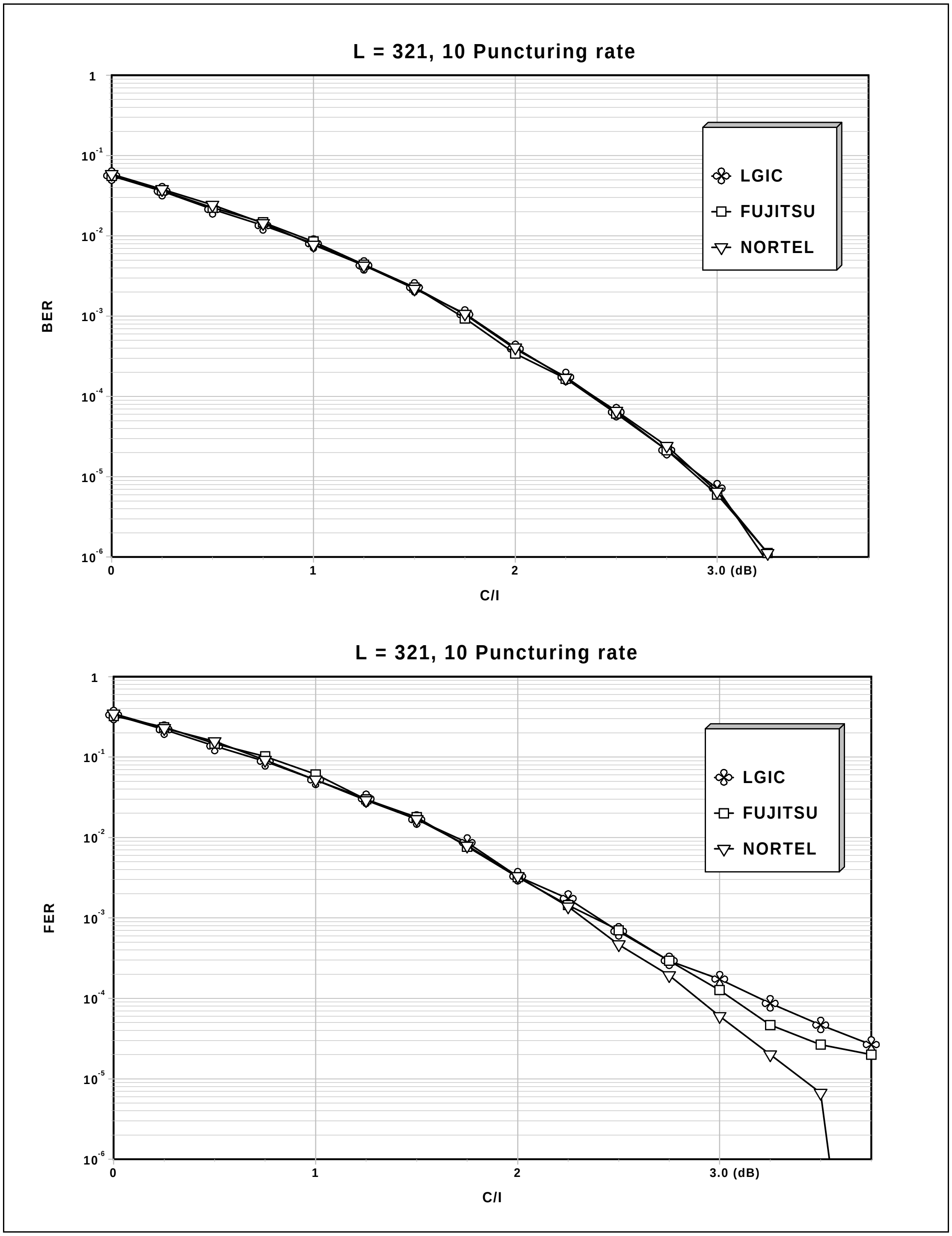


FIGURE 15. Information Block Size=321, Punctured Bits=146

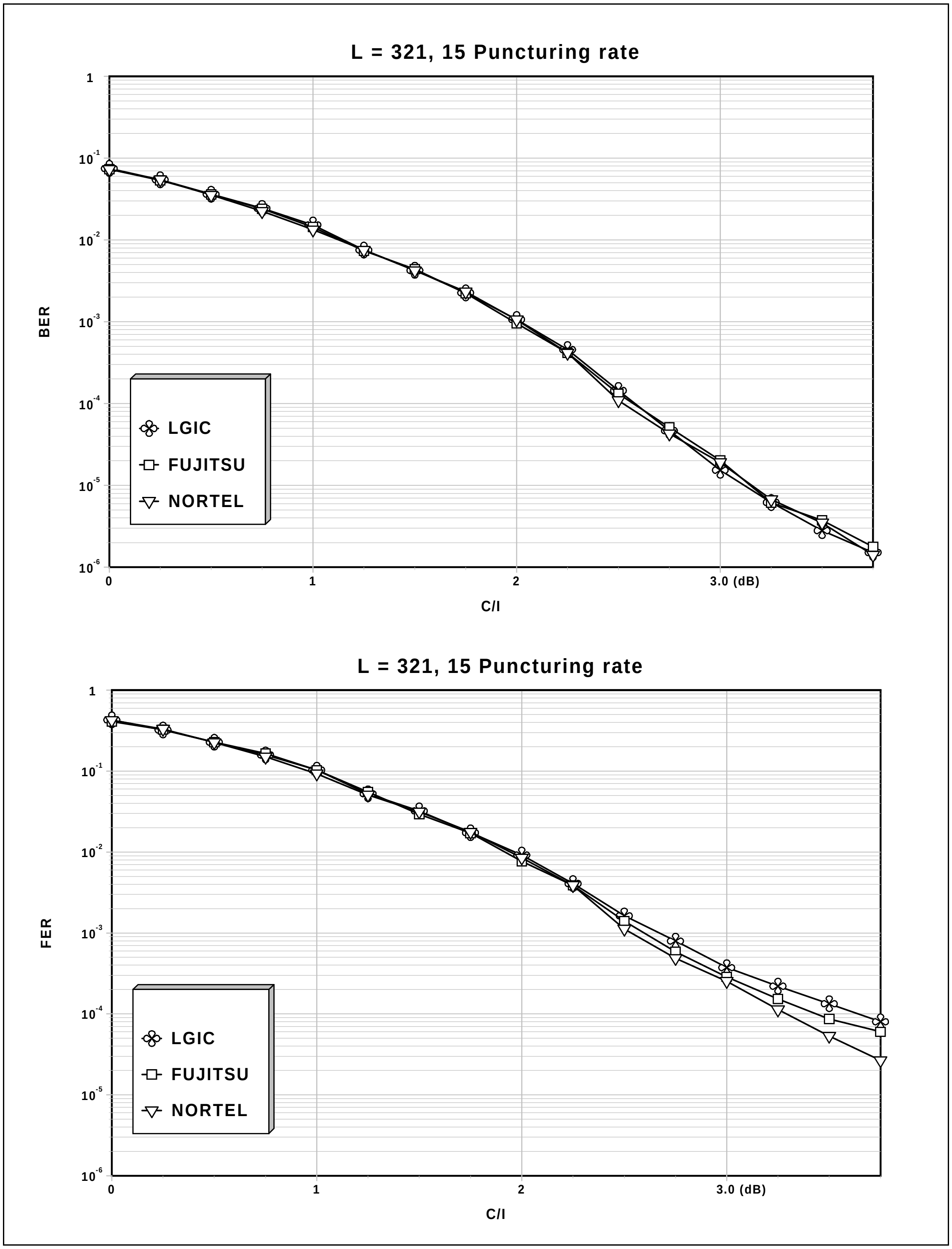
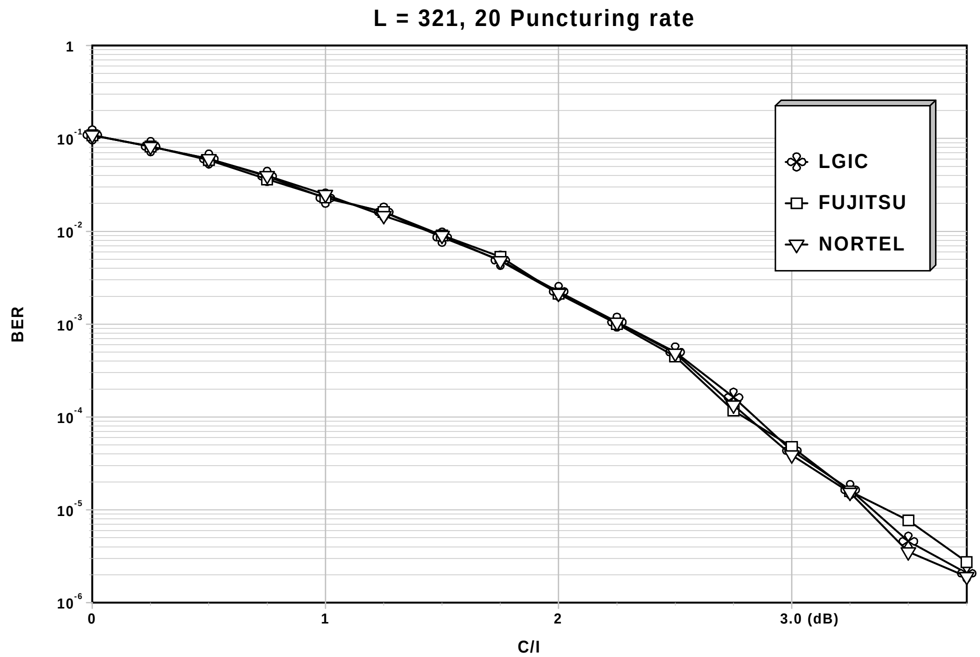


FIGURE 16. Information Block Size=321, Punctured Bits=195



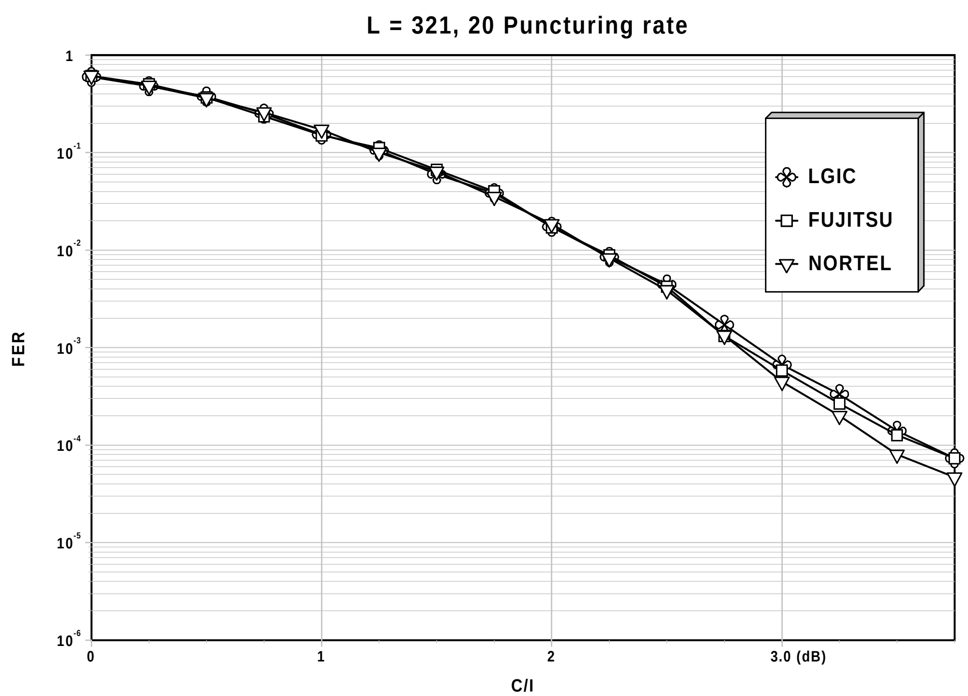


FIGURE 17. Information Block Size=323, Punctured Bits=195

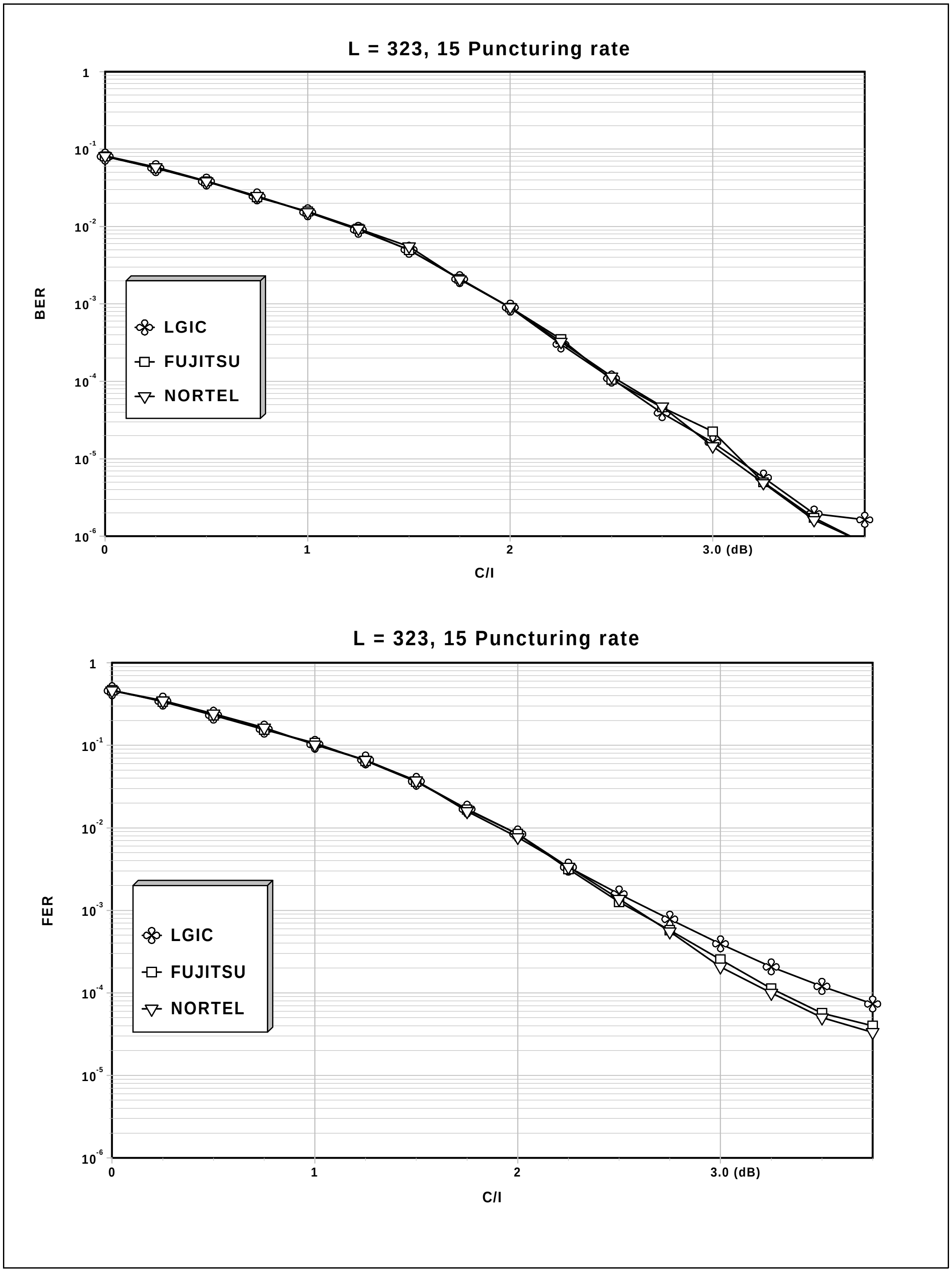
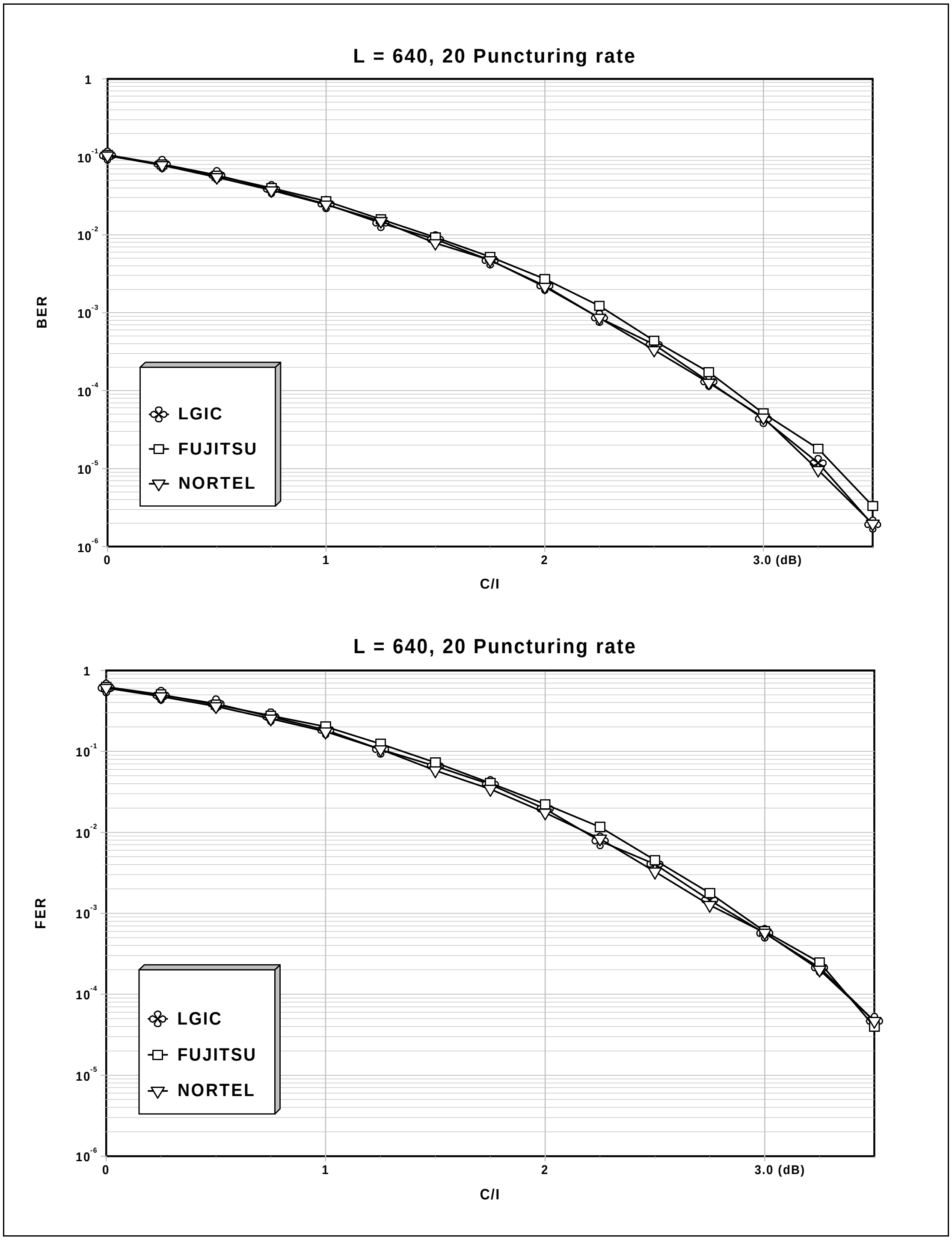


FIGURE 18. Information Block Size=640, Punctured Bits=384



2.4 Simulation Puncturing of 3GPP Convolutional Code

Figure 19 and Figure 20 show the performance comparison of 3 branch puncturing based on Siemens algorithm and the proposed 2 branch puncturing scheme. As we can see the proposed scheme is at least the same or better than Siemens puncturing.

FIGURE 19. Information Block Size=640, Punctured Bits=192

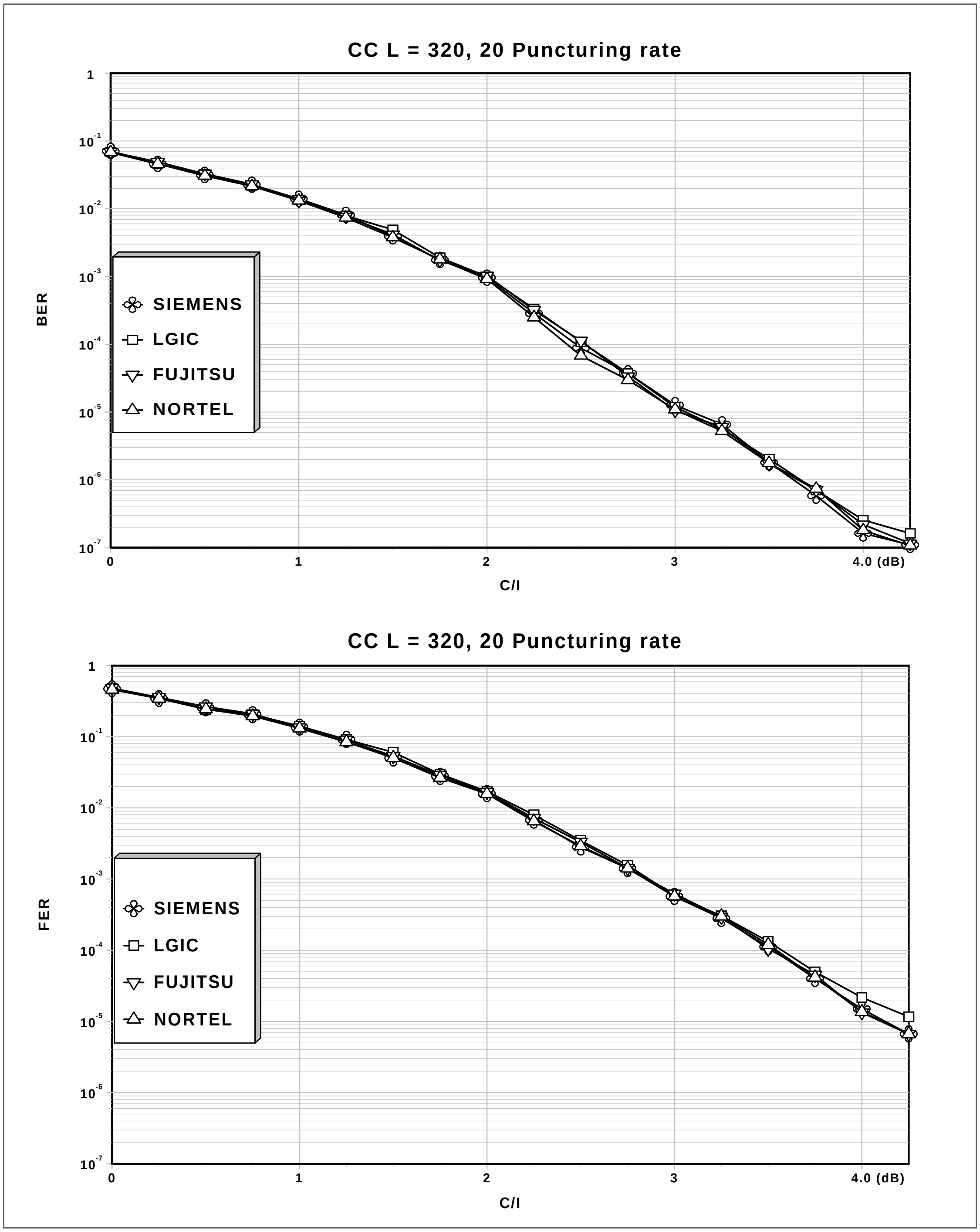
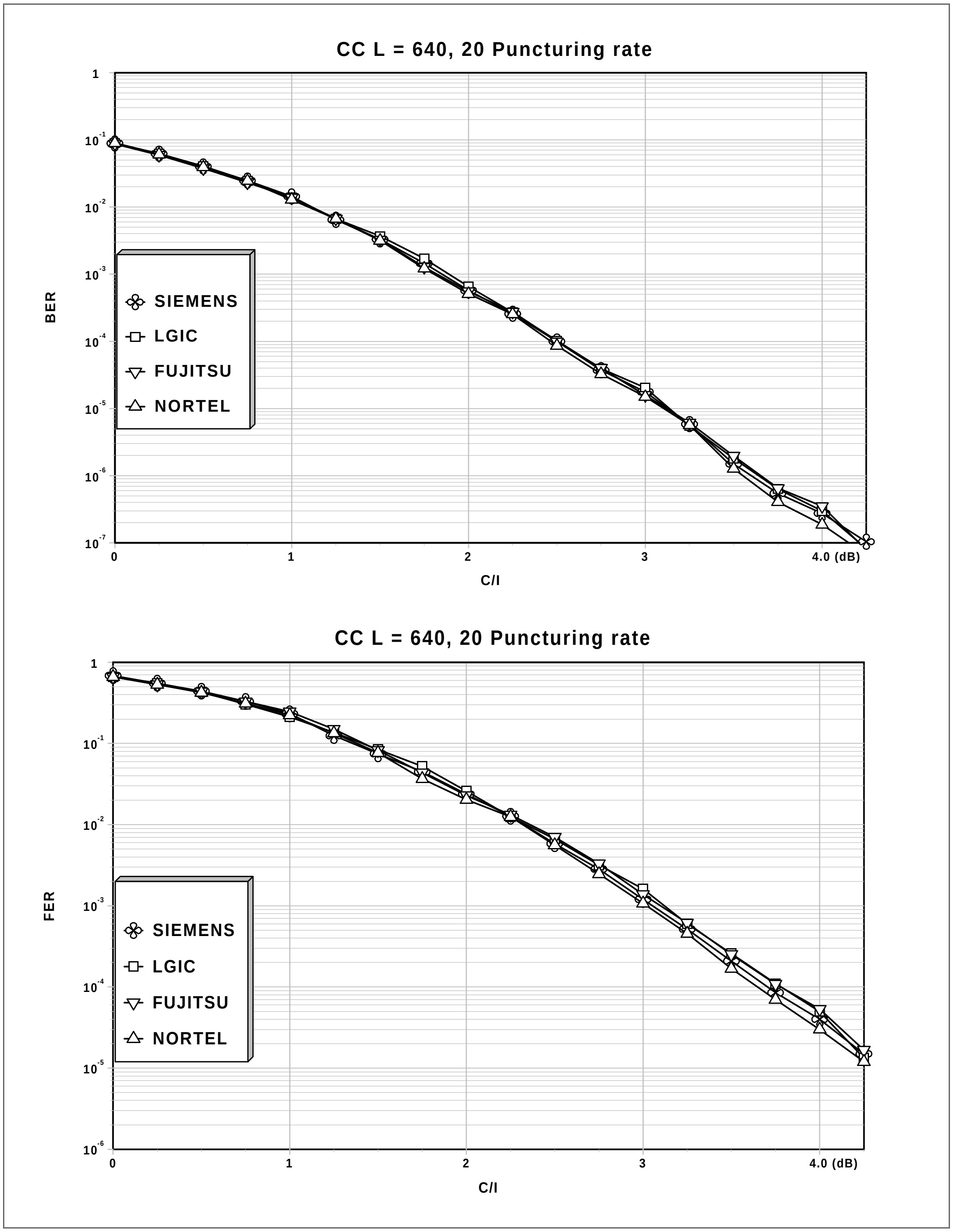


FIGURE 20. Information Block Size=640, Punctured Bits=384



3.0 Summary and Conclusions

In this contribution, we have demonstrated the impact of parity loss due to puncturing to the Turbo decoder performance. And we propose Turbo interleaving based puncturing scheme and compared with Fujitsu and LGIC proposals. Based on the simulation results, we show that:

- For R=1/2 8-PCCC Code puncturing, with frame size 320, the proposed scheme has 0,35dB gain @BER=10^{-5} and 0.25dB gain @FER=10^{-4}.
- Applying current working assumption for convolutional code puncturing to 8-PCCC code puncturing which involves the systematic bits will always cause the performance loss. e.g. with frame size=320 20% puncturing rate, the loss is 0.25dB @BER=10^{-5}
- For R=1/3 8-PCCC rate matching puncturing, the prosed scheme is consistently better other proposals in **all** simulation cases, in particular than gain are:

  0.2dB, @FER=01^{-4}, with frame size=321, puncture rate=10%

  0.3dB, @FER=10^{-4}, with frame size=321, puncture rate=15%

  0.25dB,@ FER=10^{-3},with frame size 323, puncture rate=15%

In this contribution, we propose to use Turbo Interleaver based puncturing scheme for

- Code Rate R=1/2 8-PCCC code puncturing, and replace the current Burrou puncturing working assumption
- Code Rate R=1/3 8-PCCC rate matching puncturing.
- Code Rate R=1/2 8-PCCC rate matching puncturing
- To separate the systematic bits from Rate matching puncturing.

For K=9, R=1/3 convolutional code, we propose

- To puncture polynomial G0 and G1 and keep G2 separate from puncturing.

We propose to unify the Turbo code and Convolutional code puncturing scheme into a single framework.