TSG-RAN Working Group 1 meeting #4
Tokyo, Japan
April 18-20, 1999

3GPP/TSGR.1#4(99)-446

TITLE:

Motorola Turbo Code Interleaver Ad Hoc 5 Work Plan Performance

SOURCE:

Motorola

ABSTRACT:

This contribution provides an analysis of the Motorola turbo code interleaver based on the Ad Hoc 5 Work Plan. Complexities of the proposed turbo code interleavers from Motorola, Hughes/Nortel, Canon, and the NTT DoCoMo merged proposal are also considered.

1.0 Key Points to Consider

- AWGN performance is very similar for the Motorola, Hughes/Nortel, and the NTT merged interleavers. BER differences are approximately 0.1 dB
- Performance in the more realistic mobile fading channels is virtually identical for all interleavers simulated.
- Hamming weight asymptote and free distance do not accurately predict performance.
- Complexity is a very important consideration in the choice of interleaver.
- The chosen interleaver should be of low complexity and capable of completely operating in hardware without the need for interfacing with a programmable DSP.
- The Motorola interleaver is the best choice considering performance and complexity.

2.0 Static Performance

AWGN simulations down to 1E-6 BER are shown in Figure 1 for the five random block sizes (720, 933, 1158, 7087, and 7686) chosen according to the Ad Hoc 5 work plan [7]. The results show **very similar performance** (within 0.1 dB) for Motorola, NTT Merged and the Hughes/ Nortel interleavers. We have attempted to simulate the Canon interleaver also, but we are not completely confident in our results because we are not certain we exactly understand the procedure of matching the selected information data rates to the interleaver size for the Canon interleaver.

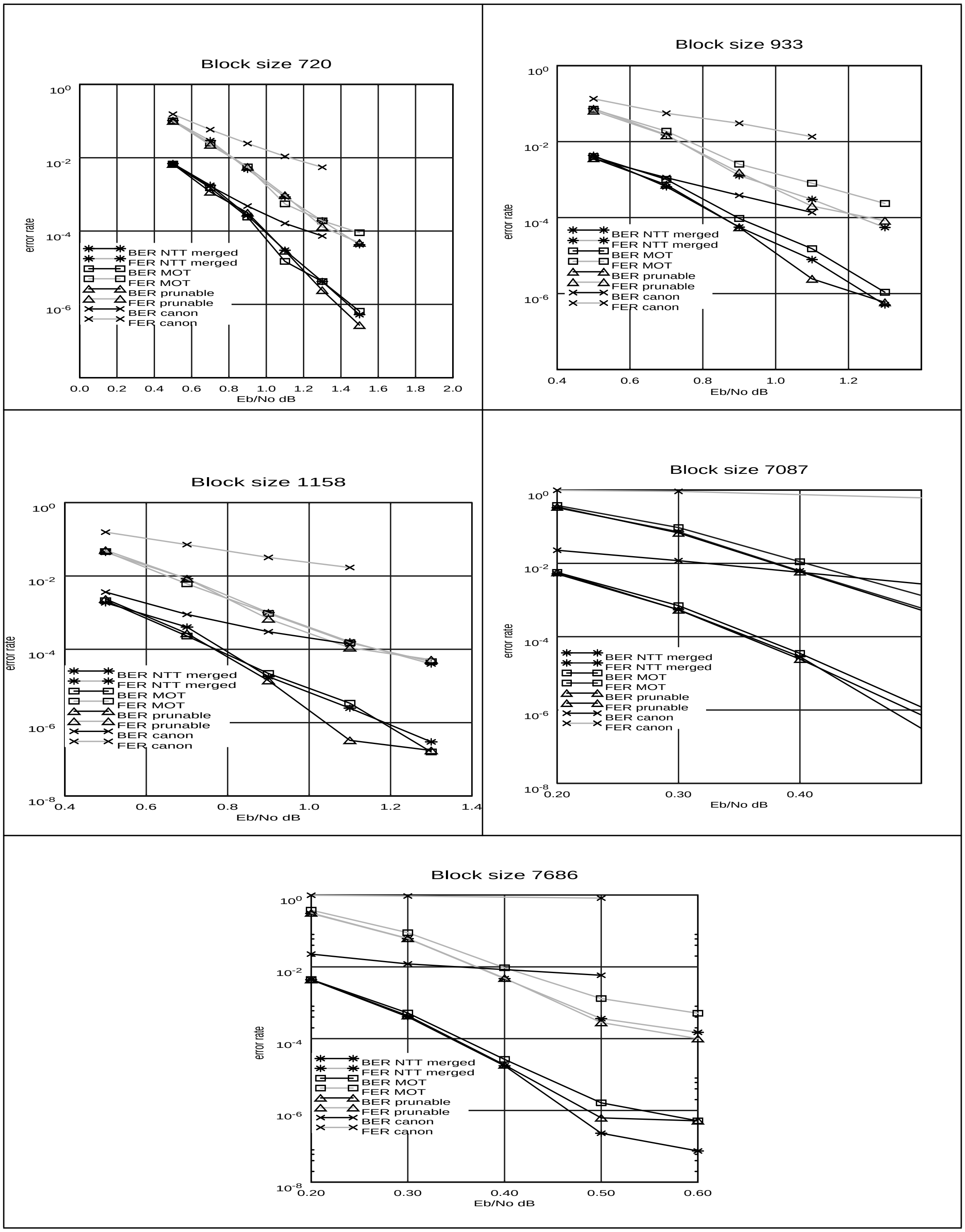

Figure 1: AWGN comparison of turbo interleavers with 8 states 8 iterations

3.0 ETSI Phase II

As required in the work plan, fading simulations have also been done according to ETSI phase II parameters. These are shown in Figures 2 -7. We consider these simulations to be **much more realistic** for mobile channels than the static simulations. It is very important to note that for all interleavers simulated, **the performance is virtually identical**.

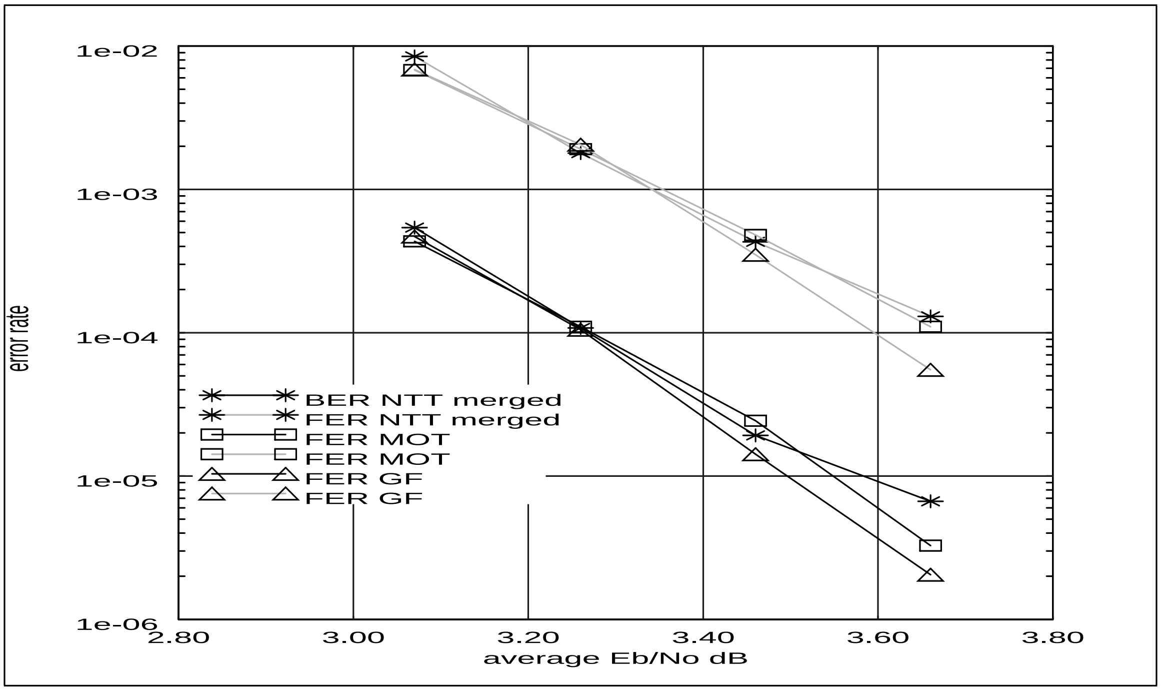

Figure 2: Turbo interleaver comparison using 8 states 8 iterations turbo code, data rate 64kbps, 10 ms interleaving interval, 3 kmph mobile speed

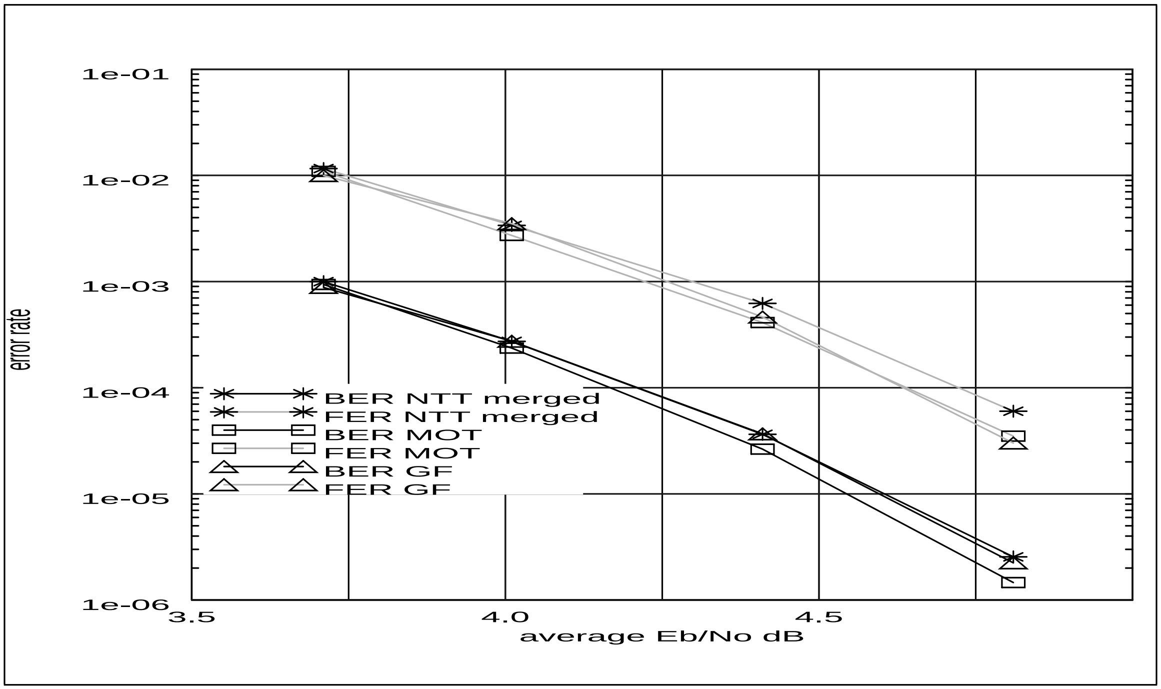

Figure 3: Turbo interleaver comparison using 8 states 8 iterations turbo code, data rate 64kbps, 10 ms interleaving interval, 30 kmph mobile speed

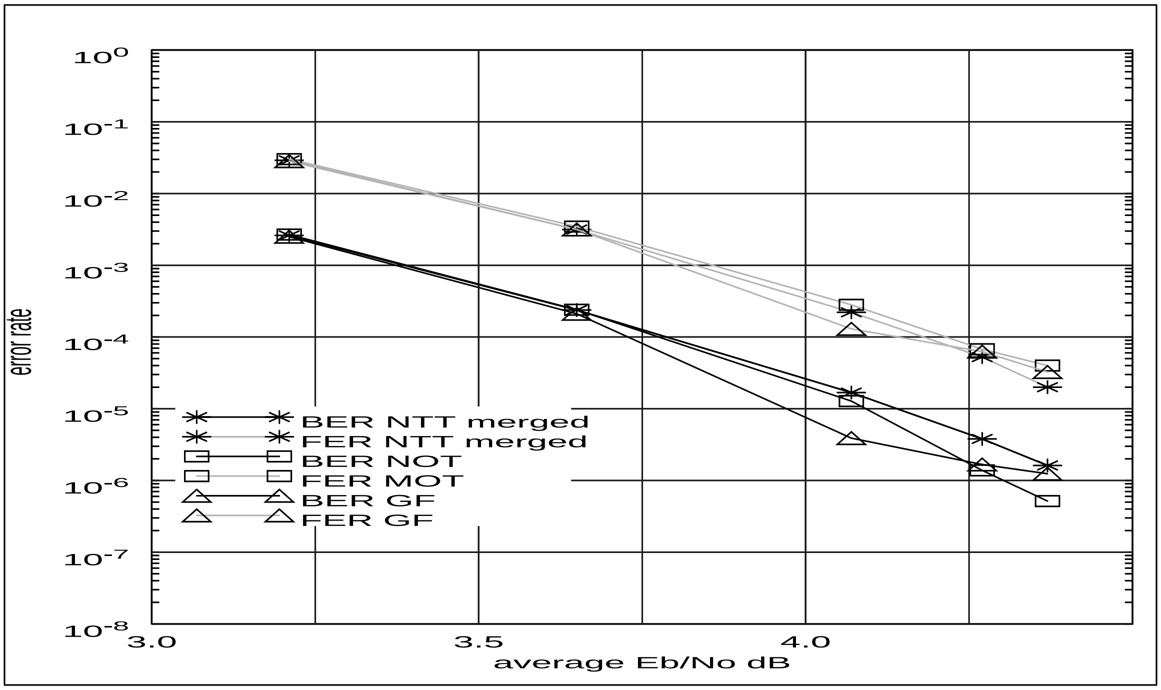

Figure 4 Turbo interleaver comparison using 8 states 8 iterations turbo code, data rate 32 kbps, 10 ms interleaving interval, 3 kmph mobile speed

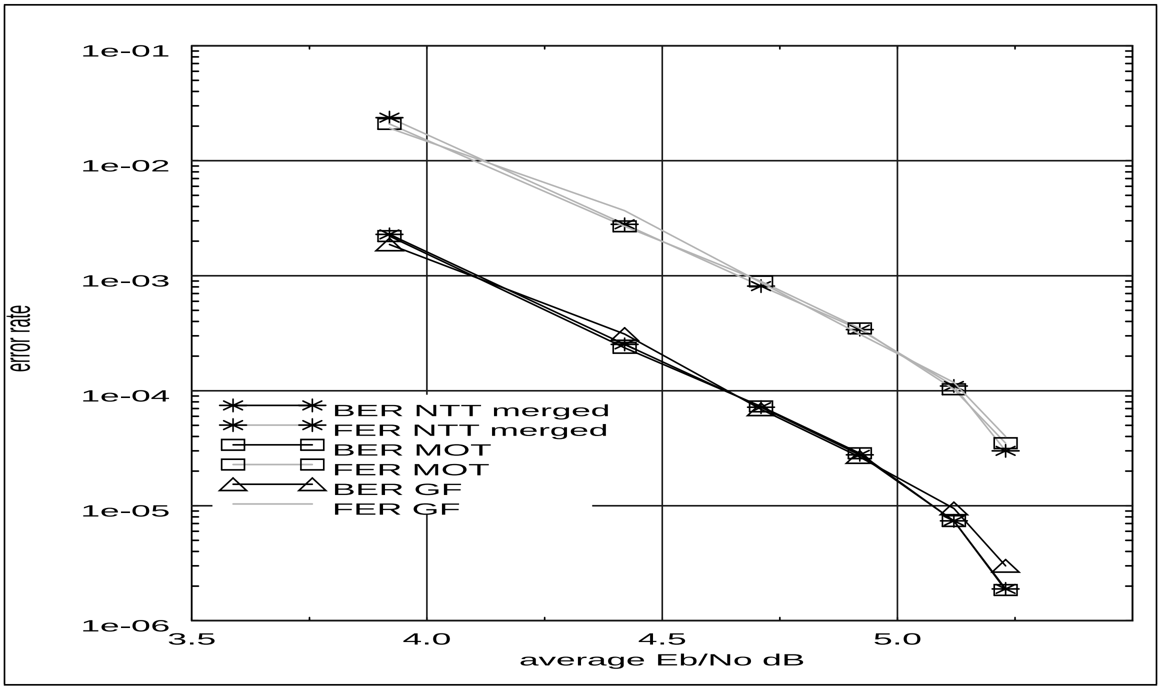

Figure 5 Turbo interleaver comparison using 8 states 8 iterations turbo code, data rate 32 kbps, 10 ms interleaving interval, 30 kmph mobile speed

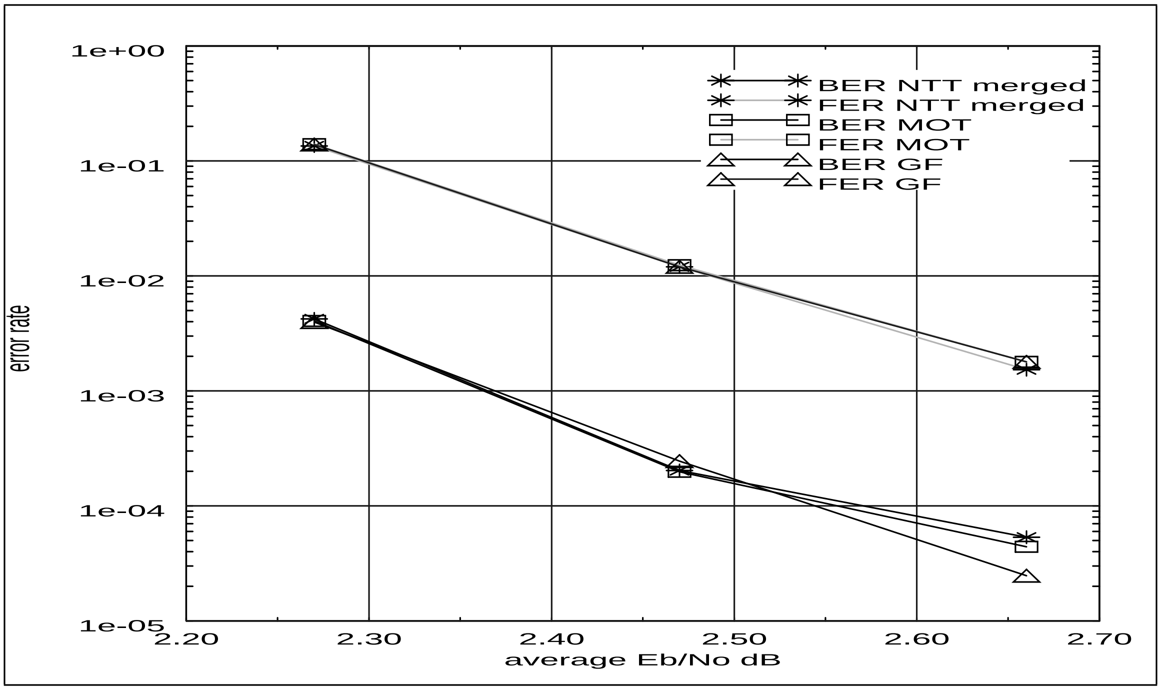

Figure 6 Turbo interleaver comparison using 8 states 8 iterations turbo code, data rate 64 kbps, 80 ms interleaving interval, 3 kmph mobile speed

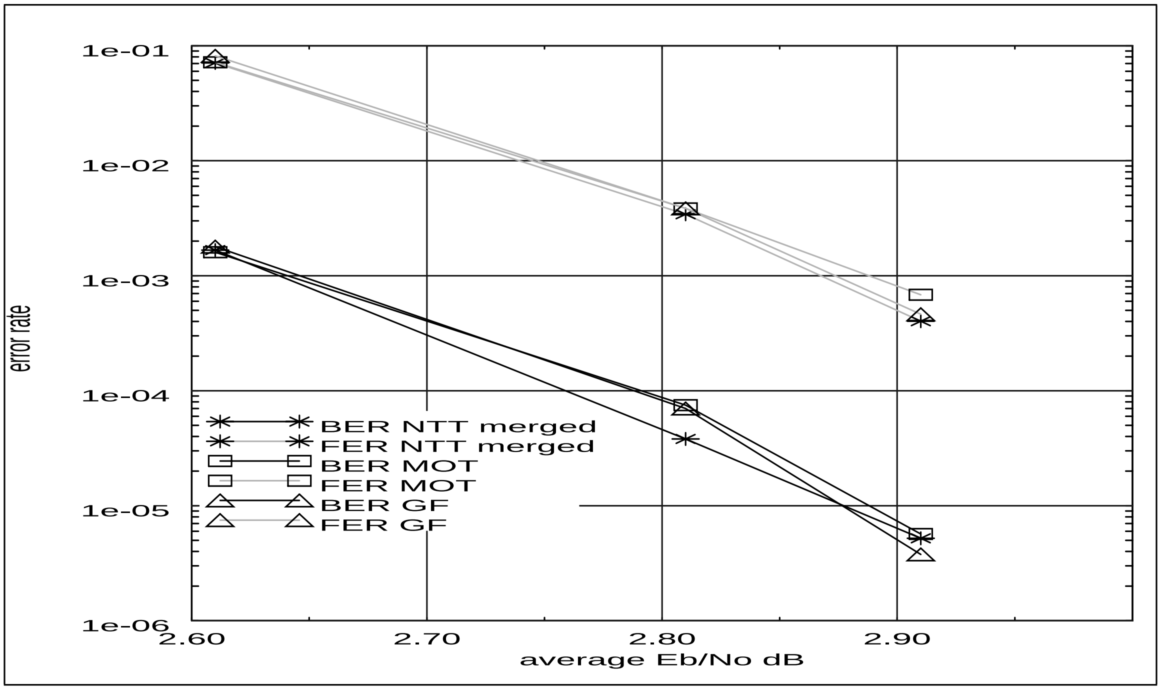

Figure 7 Turbo interleaver comparison using 8 states 8 iterations turbo code, data rate 64 kbps, 80 ms interleaving interval, 30 kmph mobile speed

4.0 Spectral Analysis

According to the work plan from the last Ad Hoc 5 meeting, the best method for choosing an interleaver is to consider three criteria:

1. Simulation performance
2. Hamming Weight and Free Distance analysis
3. Complexity

We do not feel that HWA and Free Distance should be considered before complexity because these do not necessarily provide an accurate assessment of performance.

Because it is difficult to simulate all possible interleaver block lengths, two approximations to the actual bit error rate have been proposed. The first, free distance asymptote, is based only on the terms of the union bound with minimum weight while the second, Hamming Weight Asymptote, includes multiple terms of this bound. In either case this performance measure alone does not always give an accurate portrayal of the performance at the BER of interest. HWA and free distance only consider certain output weights corresponding to low input weight sequences.

At sufficiently high SNR the actual performance may approach these asymptotes but it is not clear at what SNR and BER this occurs. Large multiplicities of higher weight terms can cause the true performance to be the opposite of what is predicted by the measures. There may be other weights with a high multiplicity which contribute significantly to BER at E_b/N_o levels of interest for typical mobile applications.

An example is shown in Figure 8 where the simulated performance for interleaver block length 625 is shown along with the Hamming Weight Asymptote. Even at a BER as low as 1E-6, the Motorola interleaver is superior to the NTT merged proposal. This is in contrast to what the Hamming Weight Asymptote suggests. These tools may be useful in finding frame lengths which may have high floors but simulation must be done to confirm these predictions. As another example, the Canon interleaver has excellent HWA (of the 25 random block sizes chosen for evaluation, all but one have an HWA of 0) but this is not indicative of simulated performance.

While these measures may ultimately give information about high SNR/low BER performance, it is difficult to judge their importance relative to other measures such as complexity because it is not clear at what BER levels they are accurate.

5.0 Complexity

In realistic mobile channel conditions (fading), all interleavers will perform identically. Hence, we feel that complexity should be considered of primary importance in selecting the interleaver; however, we are concerned that complexity is being ignored. From a subscriber manufacturer’s point of view, we would like to implement the turbo code interleaver in hardware without the need for DSP intervention.

To the best of our understanding, the NTT Merged would have to be implemented in DSP. It is very algorithmic and control oriented rather than a simple data flow type routine like the Motorola interleaver.

The Hughes/Nortel routine can probably be implemented in hardware but we think it is approximately 5 times larger and would require 2600 bits of memory compared to Motorola’s 40.

Where possible, the complexities are given in terms of percentage of the overall turbo coder. The gate count for the turbo coder was determined using the statement in [1] that a 9 by 5 bit multiplier is 5% of the overall coder complexity, and the assumption that such a multiplier is about 600 gates. This leads to a gate count of 12000 for the turbo coder, and this is the number used in the

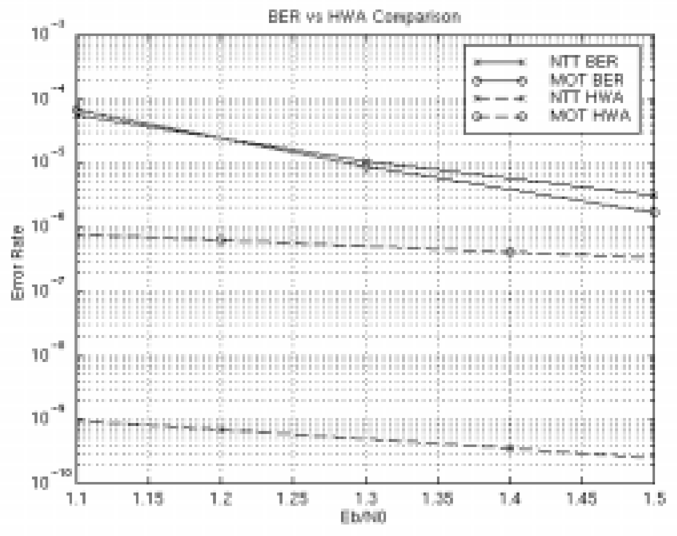

Figure 8: BER comparison and HWA comparison for Motorola and NTT Merged interleavers. HWA does not predict BER performance.

relative complexity analysis. Please note that it is difficult to say specifically what the complexity of the implemented turbo coder will be, as it will vary greatly depending upon implementation and clock speed. Thus, raw gate count estimates for each interleaver have also been given for comparison.

5.1 Motorola

The gate count estimate for the Motorola interleaver is shown below in Table 1.

Module	Gate Equivalents	Comment
Counter	60	5-bit counter
Bit Reversal	0	
LFSR	154	(10g FF + 3g XOR) per bit * 8 bits + 50g augmentation circuitry
Shift Left m Bits	60	Muxes required to shift a 5 bit value from 4-12. (3g/mux)*20 = 60g
Adder	155	15g per bit * 5 bits + 10g per bit * 8 bits
Comparator	130	10g per bit * 13 bits
Output Address Latch	130	10g FF per bit * 13 bits
LUT	40	5x8 LUT (40g), 1g per bit
TOTAL	729	
RELATIVE TO CODER	6%	729g interleaver / 12000g coder = 6%

Table 1: Motorola interleaver gate count / memory

Note that this implementation can support any block size from 288 bits up to 8,224 bits using a look up table (LUT) which contains only 5, 8-bit values (this can be a very small ROM but more likely would be synthesized into logic gates). **No other parameters are required**. If larger block sizes are required, this interleaver could easily be modified. For example, to handle a block size of 81920 bits (1.024 Mbps, 80 ms block), this interleaver would only grow to 1050 gates.

The 729 gates are all that is required to implement the interleaver. No additional parameters need to be stored and no additional memory is required. This interleaver is very well suited for hardware implementation. Interleaver addresses can be generated real-time. They do not need to be generated prior to use and stored. The interleaver is very flexible and easily adapts to changing block size.

5.2 NTT Merged

There are essentially 6 steps to the NTT DoCoMo Merged Interleaver:

1. Decide on number of rows, N, and assign appropriate PIP
2. Decide on number of columns, M, and set appropriate flags
3. Create pattern of N primes >=7 that are not factors of (M - 1)
4. Create index of (M - 1) multiples of the primitive root
5. Create interleaver pattern
6. Puncture interleaver pattern

Each of these steps can be broken down into pseudo-code to understand the high complexity of this interleaver.

Step 1: Decide on number of rows, N, and assign appropriate PIP

```
if size = 2281 - 2480 or 3161 - 3210
  N = 20
  Use PIP2
if size = 481 - 530
  N = 10
  Use PIP1
otherwise
  N = 20
  Use PIP3
```

Step 2: Decide on number of columns, M, and set appropriate flags

```
M = size / N rounded up
find M, (M - 1), or next highest number > M that is prime
if M prime
  no action necessary
else if (M - 1) prime
  M = M - 1
  set flag plus1 = 1
else
  find first prime greater than M
  set M = to this prime
  set flag minus1 = 1
  size = (M - 1) * N
  if original size = 481 - 530
```

```
    clear flag minus1 = 0
    size = M * N
```

Step 3: Create pattern of N primes >= 7 that are not factors of (M - 1)

```
for i = 7 to 100
  if i is prime
    if (M - 1) % i <> 0
      pattern[count] = i;
      increment count
```

Step 4: Create index of (M - 1) multiples of the primitive root

```
index[1] = 1 - minus1
for i = 2 to (M - 1)
  index[2] = (((index[i - 1] + minus1) * RPN[M]) % M) - minus1
```

Step 5: Create interleaver pattern

```
initialize indexselect[1..N] to 0
for i = 1 to (M - minus1 + plus1)
  for j = 1 to N
    if i = M
      interleaver[j + i * N] = M + (M + 1) * PIP[j]
    else if i = (M - 1)
      interleaver[j + i * N] = (M + plus1) * PIP[j]
    else
      interleaver[j + i * N] = index[indexselect[j]] + (M - minus1 + plus1) * PIP[j]
      indexselect[j] = (indexselect[j] + pattern[j]) % (M - 1)
```

Step 6: Puncture

```
for i = 1 to size
  if interleaver[i] > size
    puncture interleaver[i]
```

By breaking the interleaver down into sections like these, and examining the pseudo code for each, the high complexity of this interleaver becomes evident. In reference [1], it is explained that the intra-row permutation circuit (step 5 above) is the main operation circuit for the interleaver. It is then concluded that the complexity of the interleaver is only 7% of the overall complexity of the turbo coder, with the 9 by 5 bit multiplier being the majority of that. However, as can be seen above, there is another 9 by 5 bit multiplier operation in step 4. Depending on the method of implementation, these two multipliers alone could put the implementation complexity at 10% of the turbo coder using the numbers given in the referenced document. In addition to this, there are numerous adders, modulo operators, and another multiplier used to determine the final interleaver size once the number of columns M has been set. These operations would have be performed prior to using the interleaver. But this would have to be done every time the block size changed, possibly every frame. Furthermore, the algorithmic nature of this interleaver, coupled with the high number of conditional branches, special cases, and reliance upon the determination of primes, does not make it well-suited for hardware implementation. Implementation would likely be carried out in DSP.

5.3 Hughes/Nortel Prunable

Based on Motorola's understanding, Table 2 shows estimated gate counts for the elements of the Prunable interleaver. Row permutation and the final address comparator (for puncturing) and the output latch are similar to the Motorola interleaver. In addition a column permutation must be done which requires a 9 x 9 bit addition and a modulo computation. Furthermore the prunable interleaver parameters must be stored. This requires a minimum of 2608 bits of ROM. Also a 32x8 bit RAM must be used to store the Row state.

Module	Gate Equivalents	Comment
Counter	60	5-bit counter
Bit Reversal	0	
Shift left m bits	60	Muxes required to shift a 5 bit value from 4-12. (3g/mux)*20 = 60g
Adder	225	15g per bit * 9 bits + 10g per bit * 9 bits
Comparator	130	10g per bit * 13 bits
Output Address Latch	130	10g per bit * 13 bits
Multiplier	600	5 bit by 9 bit multiplier, ~600g
Sub-total	1205	
LUT	2608	2608 bits, 1g per bit
RAM	1024	32x8 bit, 4g/bit for row state storage
TOTAL	4837	
RELATIVE TO CODER	40.3%	4837g interleaver / 12000g coder

Table 2: Hughes/Nortel gate count / memory

If larger block sizes were desired, the ROM table would grow significantly. The other logic would grow similarly to the Motorola interleaver.

5.4 Canon

The Canon algebraic interleaver is a low complexity solution, as it involves only an adder and a modulo operation. The table below shows the hardware implementation complexity, using a gate count for the interleaver as given in [2].

Module	Gate Equivalents	Comment
Interleaver	220	As given in [2]
LUT	420	420 bits total for e values
TOTAL	640	
RELATIVE TO CODER	5%	640g interleaver / 12000g coder = 5%

Table 3: Canon gate count / memory

This analysis shows that this algebraic interleaver is comparable to the Motorola interleaver in terms of complexity. Canon recommends their interleaver for small block sizes [7]. This interleaver suffers from a highly structured nature that seems to greatly affect performance, particularly for large block sizes. Specifically, Canon has stated that their interleaver's performance suffers from 0.5 to 1.0 dB in comparison to a CDI interleaver implementation at a block size of 5120 [3].

6.0 Conclusion

AWGN performance is very similar for the Motorola, Hughes/Nortel, and the NTT merged interleavers. BER differences are approximately 0.1 dB. Performance in the more realistic mobile channels is virtually identical for all interleavers simulated. Hamming weight asymptote and free distance do not accurately predict performance. Complexity is a very important consideration in the choice of interleaver. The chosen interleaver should be of low complexity and capable of completely operating in hardware without the need for interfacing with a programmable DSP.

The preceding analysis has shown that the hardware complexity of the turbo code interleaver proposed by Motorola is equivalent to the Canon Interleaver, approximately 1/6 the complexity of the Hughes/Nortel approach and appears to be of much lower complexity compared to the NTT merged proposal. Simulation results show the Motorola interleaver outperforms the Canon interleaver. The Motorola interleaver does not require any ROM or RAM thereby simplifying the implementation as well as any self-test circuitry. Furthermore, the latency of the Motorola interleaver can be reduced to a single clock cycle with only a 20% increase in complexity over the two-clock version. At the mobile, in order to reduce power consumption the turbo decoder would most likely be implemented such that each decoder time step would require only a single clock. The ability to operate the turbo interleaver at the same clock frequency would simplify the implementation and further reduce power consumption.

Also note that since the Motorola turbo code interleaver has only one parameter (the LFSR polynomial) for each non-pruned interleaver length, it is much simpler to specify and verify. Given that all the proposed interleaver methods are very similar in performance, the turbo code interleaver should be chosen based on simplicity.

7.0 References

[1] “A Study on Merge Interleaver for the Turbo Codes,” NTT DoCoMo, TSGR1#3(99)217
[2] “1 Dimensional Algebraic Interleavers for Turbo Codes (AL-C): Description, Complexity, and Summary of Performances,” CANNON CRF, TSGR1#2(99)069
[3] “Algebraic Interleavers for Turbo Codes,” CANNON CRF, Tdoc SMG2 UMTS-L1 674/98
[4] “A Low Complexity and Flexible Turbo Interleaver with Good Performance,” Hughes Network Systems and Nortel Networks, TSGW1#2(99)101
[5] “Description of a Class of Flexible Turbo Interleavers,” Hughes Network Systems and Nortel Networks
[6] “A Proposal for Turbo Code Interleaving,” Motorola, TSGR1#3(99)239
[7] “Report from Ad Hoc 5 (24 March 1999),” Ad Hoc 5, TSGW1#3(99)319
[8] “A study on Turbo Interleaver Flexibility,” NTT DoCoMo, TSGR1#2(99)095