Turbo Decoder V1_0

17th June, 2003 **Preliminary Product Brief**

Features

- Drop-in module for Spartan-3™, Virtex-II™ and Virtex-II Pro™ FPGAs.
- Implements the 3GPP2 specification[1].
- Core contains the full 3GPP2 interleaver.
- Full 3GPP2 block size supported i.e. 378-20730.
- Core implements the MAX*, MAX or MAX SCALE algorithms.
- Dynamically selectable number of Iterations 1-16.
- Number representation: twos compliment fractional numbers:
  - Data input: 2 or 3 integer bits and 1 to 4 fractional bits.
  - Internal Calculations: 6 or 7 integer bits and 1 to 4 fractional bits.
- Sliding window size of 32 or 64.
- Works with all 3GPP2 code rates.
- Internal or external RAM data storage.

Applications

This version of the TCC (Turbo Convolution Code) decoder is designed to meet the 3GPP2 mobile communication system specification[1].

Basic Description

The TCC decoder is used in conjunction with a TCC encoder to provide an extremely effective way of transmitting data reliably over noisy data channels. The Turbo decoder operates very well under low signal to noise conditions and provides a performance close to the theoretical optimal performance as defined by the Shannon limit[2].

When a decoding operation is started, the core accepts the block size and the number of iterations from two input ports. The systematic and parity data is read into the core in parallel on a clock by clock basis. The core then starts the decoding process and implements the required number of decode iterations. Finally the decoded bit sequence is output. The entire sequence is automatically controlled from a single FD (First Data) signal and requires no user intervention. Also all the interleaving operations required in the 3GPP2 specification are handled automatically within the core.

The core expects twos complement fractional numbers as inputs and also uses this format for the internal calculations. Each fractional input number represents the Log Likelihood Ratio (LLR) divided by 2 for each input bit. This LLR value can be considered to be the confidence level that a particular bit is a one or zero. The user can trade off accuracy against speed and complexity by selecting the numerical precision that is required. The input data can have 2 or 3 integer bits and between 1 and 4 fractional bits. The precision of the internal calculations can also be controlled with the 6 or 7 integer bits and between 1 and 4 fractional bits. (The number of input fractional bits must be less than or equal to the number of internal calculation fractional bits)

The core will operate with all the different code rates of the 3GPP2 specification. The core always assumes that rate 1/5 data will be used as input, for different code rates the appropriate parity bits in the sequence are replaced by zero's. Thus the core can implement any puncturing scheme.

The core has been extensively tested in order to optimize performance. Features such as algorithm type and numerical precision have been investigated against BER performance to determine the optimum trade off between core speed and complexity. The available parameters that can be used with this core are the results of this extensive testing and represent a range of parameters which offer excellent performance with minimum complexity.

The full TCC decoder algorithm is extremely computationally intensive and therefore approximations must be made to make the algorithm usable in practice. The approach taken here is to provide the user with three algorithm choices:

1. MAX* - This is a very good approximation which is used when accuracy rather than algorithm simplicity is required. BER performance of this approach is the best of all three algorithms although this increases core complexity and resource requirements. In this algorithm a small lookup table is used to increase the accuracy of some non-linear operations.
2. MAX - This produces lower BER performance than the MAX* algorithm but it has the advantage of being less complex and therefore requires less resources. Here the

lookup table is not used hence the algorithm accuracy is reduced producing a slightly degraded BER performance (approximately 0.5dB compared to the MAX* algorithm).

3. MAX SCALE - This produces BER performance very close to the MAX* (within approximately 0.1 to 0.2dB)but with the complexity of the MAX algorithm. If the small reduction in BER performance is acceptable this will provide the best BER performance/resource requirement trade-off. Reference 3 describes this approach in more detail.

A second approximation technique is to use a sliding window in the calculations. As the sliding window only stores a subset of the entire data set at any one time then the memory requirements are significantly reduced. The range of parameters provided represent the best performance/area trade-off.

The number of iterations is determined by reading an input port at the start of each new decoding process. However, the intermediate results from each iteration can be made available to the user by changing a core parameter. With this approach the user can monitor the output from each iteration and stop the decoding process when sufficient decoding accuracy has been reached.

Performance and Resource Usage

Performance Assumptions

All the following figures describing resource and performance characterization assume a Virtex-II device with a -6 speed grade. Resource requirements are given for both the MAX* and MAX SCALE algorithms.

Resource Estimates

Assuming an Input data width of 2 integer bits and 3 fractional bits and internal calculations of 6 integer bits and 3 fractional bits then the estimated size requirements are as follows:

1. MAX* Algorithm

- Slices = 1690
- Block memory = 15
- Hardware multipliers =2

2. MAX SCALE Algorithm

- Slices=1490
- Block Memory=15
- Hardware multipliers=2

Performance Estimate

It is estimated that the maximum clock rate of the MAX* and MAX SCALE algorithms will be in the region of 97MHz and 110MHz respectively (assuming the same bit widths used in the size estimation).

Assuming the MAX SCALE algorithm running at 110MHz then the performance of the core is as follows:

- Block size =378, Iterations=3, 12.5Mbits/sec
- Block size =378, Iterations=5, 8Mbits/sec
- Block size =20730, Iterations=3, 14.5Mbits/sec
- Block size =20730, Iterations=5, 9Mbits/sec

As the Turbo decoder is a block code algorithm it is possible to increase the data throughput by using multiple cores in parallel.

Data Storage

The core must store the systematic and the parity data values for use in the calculations. There is one systematic input and four parity inputs. Assuming that the input data has 2 integer bits and 4 fractional bits then a 6x5=30bits wide input bus is required. The maximum block size is 20736 bits so the data storage requirement is 20736x30=622080bits. This can be internal or external memory but if internal memory is required then this data storage will require 32 additional virtex2 memory blocks.

BER Performance

Figure 1 shows the Bit Error Rate (BER) performance of the core.The results in this figure show the effect of performance for different block sizes against Eb/No.

These results have been generated using the Nallatech XtremeDSP Development Kit. The user programmable FPGA within the kit was programmed to create the encoded data, noisy channel and the decoder. Finally, an FPGA based BER calculation circuit detected the errors in the bits to produce the results of Figure 1.

The results have been generated with the following parameters:

- Number of iterations=5
- Sliding window size=32
- Input Bit width= 2 integer and 3 fractional bits
- Internal calculation width = 6 integer bits and 3 fractional bits
- Rate 1/3
- Error count is the minimum number of errors detected to generate each point on the curve.

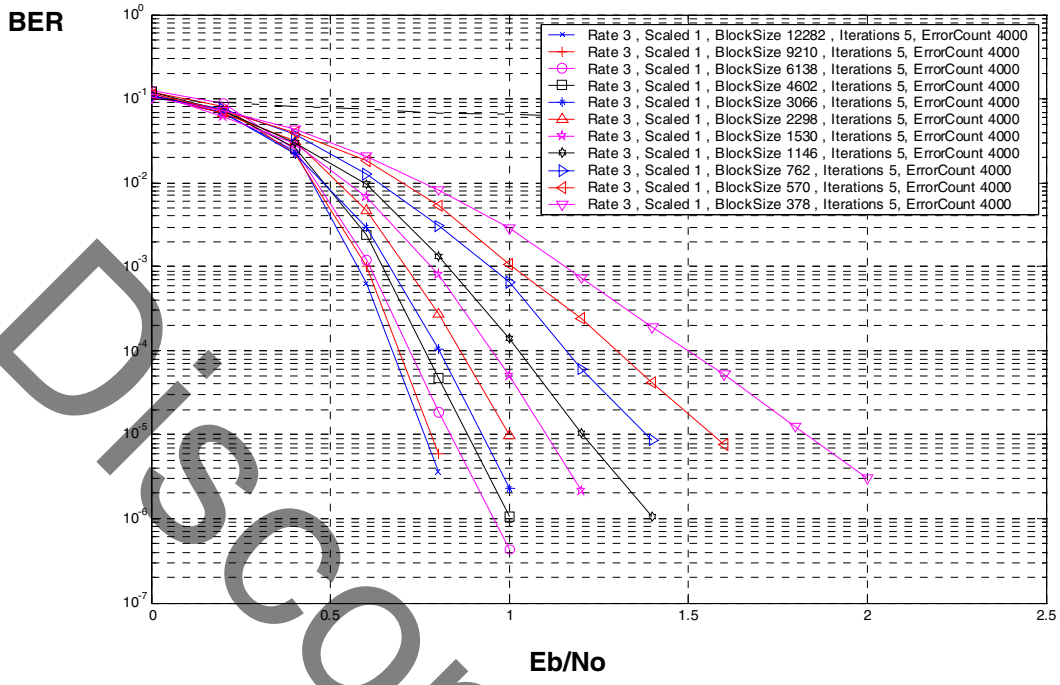

Figure 1: **Measured BER Performance of the core**

References

[1] 3GPP2 C.S0002-C Physical Layer Standard for CDMA2000 Spread Spectrum Systems. Version 1.0 Release C, 28th May 2002.

[2] C Berrrou, A. Glavieux, and P Thitimajshima, "Near Shannon Limit Error-correcting Coding and Decoding Turbo Codes", IEEE Proc 1993 Int Conf. Comm., pp1064-1070.

[3] Improving the MAX Log MAP Turbo Decoder, J Vogt and A Finger, Electronics Letters 9th November 2000, Vol36 No 23, pp1937-1939.