3GPP2 Turbo Decoder v1.0

Features

- Drop-in module for Spartan™-3, Spartan-3E, Virtex™-II and Virtex-II Pro FPGAs
- Implements the CDMA2000/3GPP2 specification[1]
- Core contains the full 3GPP2 interleaver
- Full 3GPP2 block size supported, that is, 378-20730
- Core implements the MAX*, MAX or MAX SCALE algorithms
- Dynamically selectable number of Iterations 1-15
- 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
- To be used with Xilinx CORE Generator™ system v7.1i

Applications

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

General Description

The TCC decoder is used in conjunction with a TCC encoder to provide a reliable, extremely effective way to transmit data over noisy data channels. The turbo decoder core 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. In addition, all the interleaving operations required in the 3GPP2 specification are handled automatically within the core.

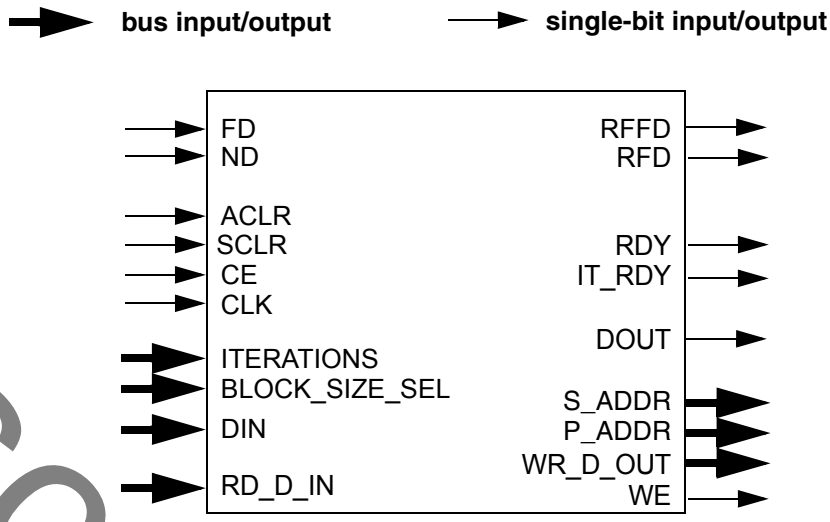

Figure 1: **Input and Output Ports for TCC Decoder**

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)

Algorithm Type

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***. A very good algorithmic approximation 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**. Produces lower BER performance than the MAX* algorithm, but provides the advantage of being less complex, and therefore requires fewer resources. In this case, the lookup table is not used, which reduces the algorithm accuracy and subsequently produces a slightly degraded BER performance (approximately 0.5dB compared to the MAX* algorithm).
3. **MAX SCALE**. 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 provides the best BER performance/resource requirement trade-off. Reference 3, *Improving the MAX Log MAP Turbo Decoder*, describes this approach in greater detail.

Sliding Window

A commonly used technique to reduce the resource requirements of the core is the use of a *sliding window* in the calculations. As the sliding window only stores a subset of the entire data set at any one time, the memory requirements are significantly reduced. Two sliding window sizes can be used with the core; 32 or 64.

Code Rates

The core operates with all the different code rates of the 3GPP2 specification, and 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 zeros, allowing the core to implement any puncturing scheme.

Input/Output Pins

Signal names are shown in Figure 1 and described in Table 1.

Table 1: **Core Signal Pinout**

Pin	Direction	Port width (bits)	Description
FD	Input	1	**First Data** When asserted (high) on a valid clock edge the decoding process is started
ND	Input	1	**New Data** When asserted (high) on a valid click edge a new input value is read from the DIN port
ACLR	input	1	**Asynchronous Clear** When asserted (high) the decoder asynchronously resets
SCLR	input	1	**Synchronous Clear** When asserted (high) on an active clock edge the decoder is reset
CE	input	1	**Clock Enable** When this is de-asserted (low) rising clock edges are ignored and the core is held in its current state
CLK	input	1	**Clock** All synchronous operations occur on the rising edge of the clock signal
ITERATIONS	input	4	**Iterations** The number of iterations that the core must implement
BLOCK_SIZE_SEL	input	4	**Block Size Select** The block size of the current decode operation
DIN	input	varies (see later)	**Data Input** Consisting of the systematic and parity data input
RD_D_IN	input	varies (see later)	**Read Data In** Only used with the external memory interface option. This is where the systematic and parity data are read into the core for decoding.
RFFD	output	1	**Ready For First Data** When asserted (high), the core is ready to start another decoder operation
RFD	output	1	**Ready For Data** When asserted (high) the core is ready to accept input on the DIN port

Table 1: **Core Signal Pinout** ***(Continued)***

Pin	Direction	Port width (bits)	Description
IT_RDY	output	1	**Iteration Ready** Asserted at the end of each iteration to indicate that the data on the DOUT port is valid. This can be used in fast termination circuits and is described later.
RDY	output	1	**Ready** Similar to the IT_RDY signal but only asserted on the iteration defined by the ITERATIONS port. This indicated when the entire decoding operation has been complete.
S_ADDR	output	14	**Systematic Address** Only used with an external memory interface. This is the address to control the reading and writing of the systematic data.
P_ADDR	Output	14	**Parity Address** Only used with an external memory interface. This is the address to control the reading and writing of the parity data.
DOUT	output	1	**Data Out** Decoded output from the core
WR_D_OUT	output	Varies	**Write Data Out** Only used with an external memory interface where this is the systematic and parity data loaded into the external memory. This data is then processed in the decoder
WE	output	1	**Write Enable** Indicates that there is valid data on the WR_D_OUT port to be written into memory

Functional Description

Clock: CLK

All operations of the core are synchronized to the rising edge of the CLK signal. If the optional CE pin is required, a valid clock signal occurs only when CE is high on a rising clock edge. If CE is low, the core is held in its current state.

Clock Enable: CE

CE is an optional pin used to indicate if the next rising clock edge is valid. When CE is high, rising clock edges allow the decoding process to continue, but if CE is low, the core operations are suspended and the core remains in its current state. All synchronous signals are ignored when CE is low.

First Data: FD

FD is used to start the decode operation. When FD is high on a valid clock edge, then the first data is read from the DIN port. Simultaneously, the same clock edge is used to read the values of the BLOCK_SIZE_SEL and the ITERATIONS ports which define the block size and the number of iterations to be implemented. FD should only be held high for a single clock cycle.

New Data: ND

The ND signal is optional and is used to indicate that there is new input data to be read from the DIN port. For example, if ND is required and the input block size is 378, then 378+6(tail bits) active high ND assertions are required to load in the complete block before the decoding operations commences. Once

all the expected input data has been read into the core, the ND signal is ignored until the next decoding block is started.

Asynchronous Clear: ACLR

The ACLR signal is optional and when it is asserted high, the core is reset to its initial state, that is, the core is ready to process a new block. Following the initial configuration of the FPGA, the core will automatically be in the reset state, so no further ACLR is required before a decoding operation can take place. The highly pipelined nature of the decoder core means that any ACLR signal actually creates some SCLR signals internal to the core. For this reason, the SCLR signal is recommended for use in this core.

Synchronous Clear: SCLR

The SCLR signal is optional and when it is asserted high on a valid clock edge, the core is reset to its initial state, that is, the core is ready to process a new block. SCLR is ignored if CE is low. Following the initial configuration of the FPGA, the core will automatically be in the reset state so no further SCLR is required before a decoding operation can take place.

ITERATIONS

The 4 bit input port represents the number of iterations the core will implement. Valid numbers are from 0001-1111 (binary) or 1-15(decimal). The ITERATIONS port is read on a valid clock edge when FD is high, which then defines the iterations for the decode operation.

Block Size Select: BLOCK_SIZE_SEL

The 4-bit port input port represents the specific block size to be implemented. Like the ITERATIONS signal, the BLOCK-SIZE_SEL port is read on a valid clock edge when FD is high. Table 2 shows the correct BLOCK_SIZE_SEL value for each of the 3GPP2 block sizes. All other values on BLOCK_SIZE _SEL are invalid. (Note that block size values do not include tail bits.)

Table 2: **Valid BLOCK_SIZE_SEL values**

Block Size	BLOCK_SIZE_SEL (binary)
378	0001
570	0010
762	0011
1,146	0100
1,530	0101
2,298	0110
3,066	0111
4,602	1000
6,138	1001
9,210	1010
12,282	1011
20,730	1100

Data In: DIN

The DIN port, a single-input bus, accepts the five input channels of the systematic and parity data. For example, if 2 integer and 3 fractional bits are required for each soft input value, a total of 5 bits is required for each of the five channels. The total bit width of the DIN port is therefore 5x5=25 bits. The arrangement of the DIN port for this example is shown in Figure 2. In this case, the 25-bit input port is represented by DIN[24:0], indicating that bit 24 is the MSB and bit 0 the LSB. The two parity values from the non-interleaved data are represented by RSC1_0 and RSC1_1, while the two parity values form the interleaved data are represented as RSC2_1 and RSC2_0. Figure 2 shows how the user must map each of the systematic and parity bits to each of the DIN bits in order to obtain the correct functionality.

The data input to the decoder core is assumed to be encoded using a corresponding Turbo Encoder core (Xilinx TCC_ENCODER v1.0). A basic description of the Turbo Encoder core is provided on page 8 to ensure the correct mapping between the encoder outputs and the decoder inputs. See Data Input Sequence for more information about the correct data input sequence.

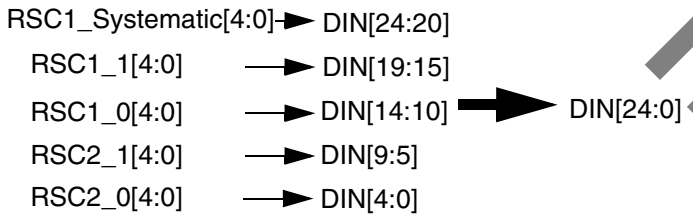

Figure 2: **Construction of DIN Port**

Read Data In: RD_D_IN

This port is only used where an external memory interface is required. This port is always the same width as the DIN port because it is the data port to write data to the external memory. It is constructed in the same way as the DIN port in that it consists of the five systematic and parity data inputs (Figure 2). Note that systematic and parity data is read in different orders during the decoding process, which means that one external memory block is required for the systematic data and a separate block is required for the parity data. These two memory areas are addressed by the S_ADDR and P_ADDR signals, respectively.

Ready For First Data: RFFD

When this optional signal is asserted high it indicates that the core is ready to accept an FD signal to start a new decoding operation. Immediately after a valid FD signal is detected, the RFFD signal will go low and remain low until it is safe to start another block.

Ready For Data: RFD

When this optional pin is asserted high it indicates that the core is ready to accept new input data. If RFD is required, then it will be high during the period that a specific block is input. When block size (plus the tail bits) of data has been input, the RFD signal will go low to indicate that the core is no longer ready to accept data.

READY: RDY

This signal is asserted high after completing the number of iterations defined by the ITERATIONS port on the valid FD signal. RDY is asserted high to indicate that the data on the DOUT port is now the final result of the decode operations. RDY is always asserted high for block size valid clock cycles.

Iteration Ready: IT_RDY

This optional port is similar to the RDY port in that when it is asserted high, it indicates that data is valid on the DOUT port. The IT_RDY signal is asserted high at the end of every iteration so that the user can monitor the decoded data at the end of each iteration. This is particularly useful when implementing fast termination algorithms where the decoding process can be stopped before all the iterations have been completed. This technique can result in a significant performance increase, as the lower the number of iterations, the greater the throughput.

Data Out: DOUT

This is the hard-coded output from the decoding process. Each block of data is output on a serial basis with RDY and IT_RDY indicating that the data is valid.

Systematic Address: S_ADDR

This is an optional 14 bit address port that is only used if an external memory interface is required. This port provided the address to the systematic data memory area. When WE is asserted high, data is written to this memory and when WE is low data, is read.

Parity Address: P_ADDR

This is similar to the S_ADDR port except that it provides the address for the parity memory block. Note that there are 4 soft input parity channels and only a single soft input systematic channel. The parity memory block will therefore always be four times larger than the systematic memory block.

Write Data Out: WR_D_OUT

This is an option port which is always the same width as the DIN and RD_D_DIN ports. When an external memory interface is required, data on the DIN port appears on the WR_D_OUT port ready for storage in the external memory. This port is configured in the same way as the DIN port (Figure 2).

Write Enable: WE

This is an optional port that is only required with an external memory interface. When the WE signal is high there is valid data on the WR_D_OUT port that must be written to memory.

Turbo Encoder

The data into the DIN port of the Turbo Decoder core must be generated using the Xilinx TCC Encoder v1.0 core, which provides the correct data format. A brief description of the data output requirements of the Turbo Encoder core is provide here for the purpose of identifying the input requirements for the decoder core.

Figure 3 shows the basic structure of the Turbo encoder. It consists of two identical Recursive Systematic Convolution (RSC) encoders: one processes the original input data, and the other processes an interleaved version of the original input data. As a general rule, the original input is delayed so that the first and successive outputs from both RSCs are matched to occur on the same clock cycle.

The output from each of the RSCs consists of the original input bit, the systematic bit, and two parity bits that are created by the circuit shown in Figure 4. Some of the RSC bits are not transmitted depending on the selected encoding rate. For example, in rate 1/5 for every one input bit, five output bits are generated.

Figure 4 shows that there is a control at the RSC input that switches between new data input and a feedback input. When *block_size* values have been output from the decoder, these control switches are switched over to create *tail bits*, which are used to force the RSCs to a known state. Each of the two RSCs create 3 sets of soft values during the tail bit generation. The output of the Turbo Encoder (and the input to the Turbo Decoder) will always consist of block_size+6 sets of soft values. See the 3GPP2 specification for more details.

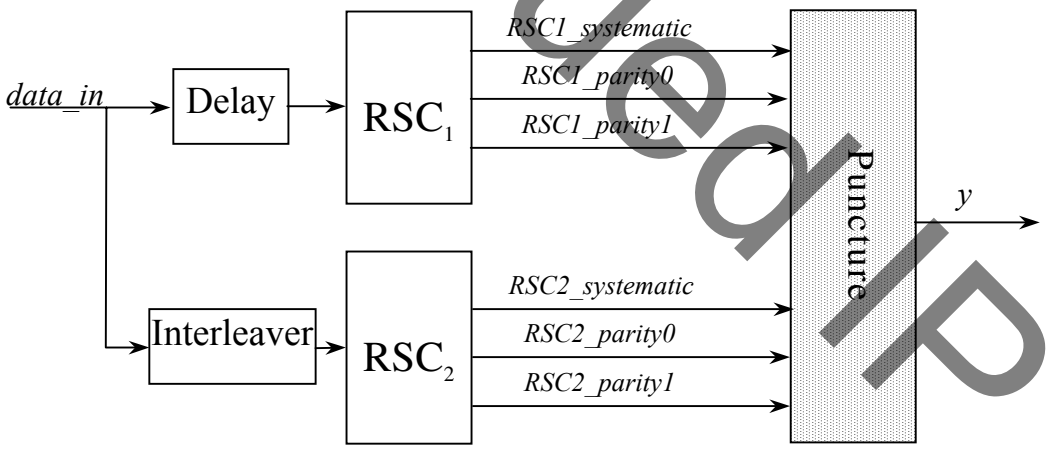

Figure 3: **Turbo Convolution Encoder**

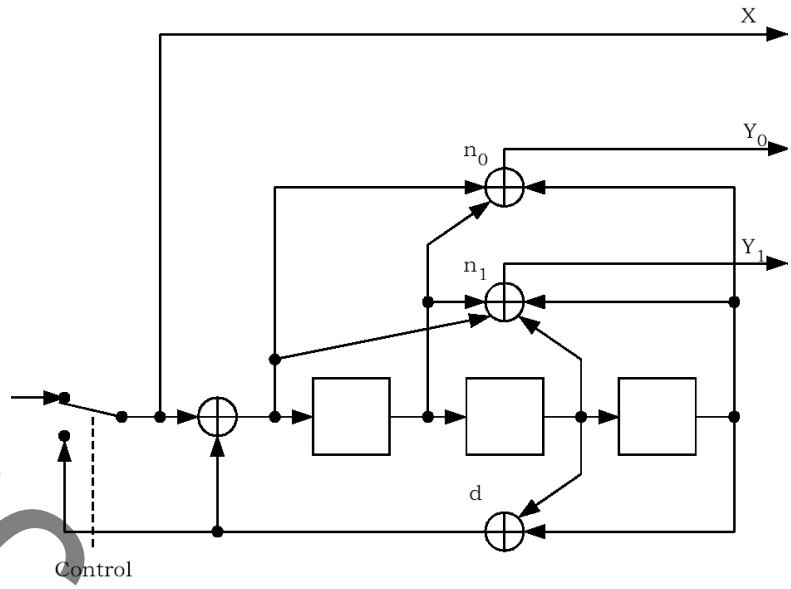

Figure 4: **Basic Recursive Systematic Convolution (RSC) Encoder**

Data Input Sequence

Figure 5 shows the data input sequence for an example case where *block_size* = 378. The input data consists of the output of the two RSC encoders and the convention used here is that RSC1_systematic(0) to RSC1_systematic (377) represent the 378 values generated during the encoder process. The same convention also applies to the parity channels. The figure shows the transition between data input at the end of the block and tail bit input period. During the RSC1tail bit input the RSC2 parity bits are not used and any value can be left on these ports at these times; they will be ignored by the core. The RSC1 inputs are also ignored during the RSC2 tail bit period, except the RSC1_systematic port is used for the RSC2_systematic values. This reduces the number of input bits required by the core.

The example in Figure 5 assumes a rate of 1/5 where all five of the encoder outputs are used. (Note that the RSC2_systematic data is never transmitted except during the RSC2 tail-bit period.) Where different puncture rates are used a number of the decoder input values will not be transmitted and these must be set to zero during the decoder input stage. This is straightforward and is left to the user to implement, if required. Refer to the 3GPP2 specification for more details.

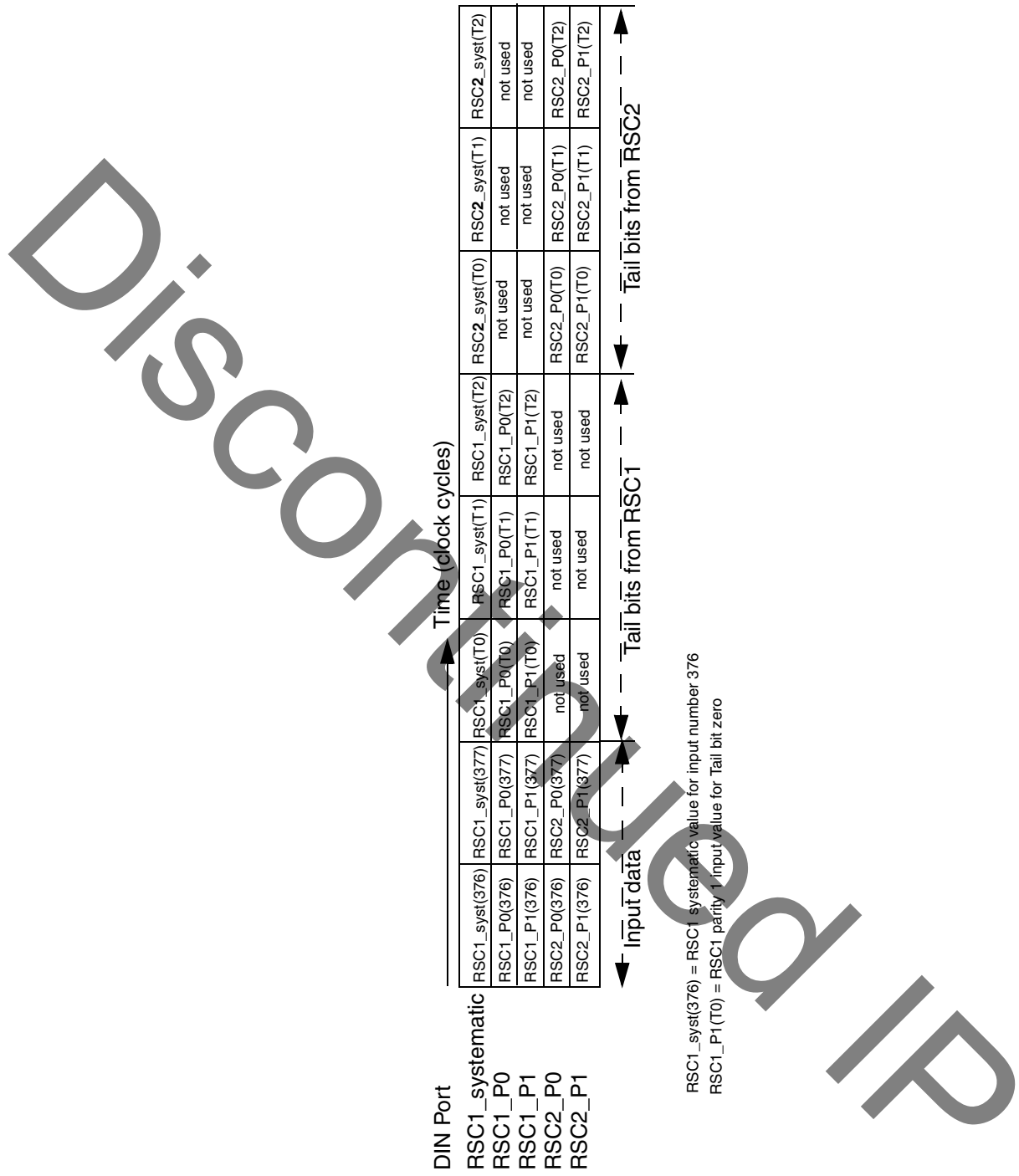

Figure 5: **Data and Tail Bit Input Sequence into the Decoder Core**

Data Input Format

The input to the DIN port of the decoder is proportional to the Log Likelihood Ratio of the received data i.e. LLR(x).l

$$LLR(x) = \ln\left(\frac{Pr(x = 1)}{Pr(x = 0)}\right)$$

This is the natural logarithm of the probability that a received symbol is a one or zero. The actual required input of the TCC_DECODER core is LLR(x)/2. Knowledge of the input data format and the noise characteristics is required to calculate LLR(x) accurately. The Turbo decoder will still function if an estimate of the LLR is made but best performance will be obtained with an accurate calculation of the LLR.

Assume that there is a BSPK (Binary Phase Shift Keying) input signal, x, where a value of 1.0 represents a transmitted logic 1 and -1.0 represents a transmitted logic zero. Also, assuming that the input has been corrupted by Random Gaussian noise with a mean μ and a variance σ^2 the LLR(x) can be calculated from

$$LLR(x) = \ln\left(\frac{\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{(x-\mu^2)}{2\sigma^2}\right\}}{\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{(x+\mu^2)}{2\sigma^2}\right\}}\right)$$

This produces:

$$LLR(x) = \frac{2\mu x}{\sigma^2}$$

meaning that the required input into the DIN port of the decoder is given by:

$$DIN = \frac{LLR(x)}{2} = \frac{\mu x}{\sigma^2}$$

For a mean of 1, the input to the DIN port of the decoder is simply the values output from the encoder divided by the variance. If for example, the mean of the input signals is 3 instead of 1 the input values are simply divided by 3 so that they are re-scaled to have a mean of one. This will ensure optimal operation of the Turbo decoder.

Data Input Format in Relation to Eb/No

It is common practice to measure the error correction performance of different algorithms using the standard measurement:

$$\frac{Eb}{No} = \frac{\text{Energy per bit}}{\text{Noise Density}}$$

It can be shown that for white gaussian noise Eb/No can be related to the noise input by

$$\frac{Eb}{No} = 10 \times \log\left(\frac{1}{2 \times rate \times \sigma^2}\right) dB$$

Plotting the noise variance against Eb/No produces the results shown in Figure 6. This Figure shows that with the noise variance set to 1, the decoder will only be operating optimally at Eb/No values of 0dB, 1.8dB, and 3dB for rates 1/2, 1/3, and 1/4 respectively. For example, if the user requires the decoder to operate at different Eb/No values, the user can either calculate the variance value in real time or use Figure 6 to provide an estimate of the variance. For example, if the decoder is expected to operate when Eb/No is around 2dB then appropriate variance values are 0.63, 0.95, and 1.25 for rates 1/2, 1/3, and 1/4, respectively.

For the purposes of this data sheet when BER (Bit Error Rate) figures are quotes as using *scaled values* it is assumed that the noise variance is accurately known. The less accurate the noise variance of the input, the greater the degradation in BER performance.

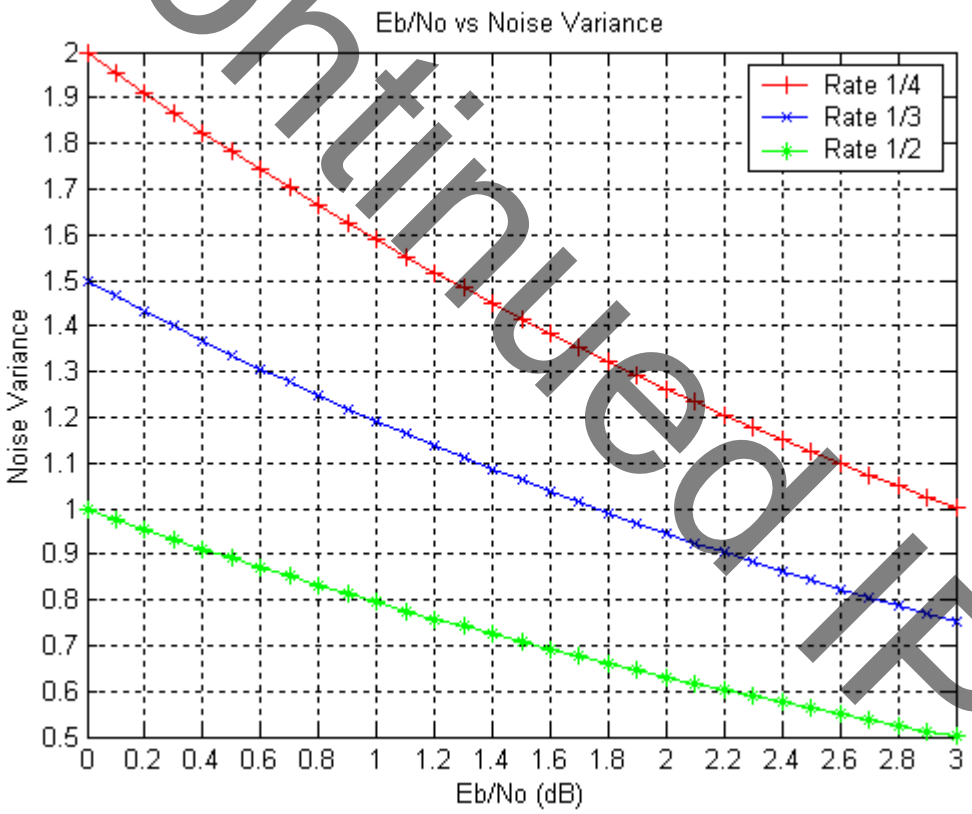

Figure 6: **Plot of Noise Variance against Eb/No for Different Rates**

External Memory Interface

The raw input data from the DIN port is used during each decoding iteration, so a RAM must be provided to store the data. This data store can be internal or external to the core.

If the data memory is internal to the core no user action is required. The core handles all storage and addressing functions internally. If the memory storage is external to the core the external memory ports are used to create the memory interface. These ports are defined in a previous section, but additional detail is included in this section.

Assuming that a 25bit wide DIN port has been selected (as shown in Figure 1), Figure 7 shows the detail of how the external memory ports are connected. The systematic and parity memory blocks are assumed to be generic memory blocks. If these are external to the FPGA, then it is likely that there is a single bi-directional data port on the memories, rather than the separate ports shown in Figure 7. These simple changes are left for the user to implement, as they will be specific to the design.

Note that the systematic memory is only 5 bits wide in this example, compared to the parity memory which is 20 bits wide. The parity memory will always be 4 times as wide as the systematic memory due to the fact that there are four parity input channels and only one systematic. The maximum block size in the 3GPP2 specification is 20736 (including 6 tail bits) giving a total memory requirement for this example as follows;

- Systematic Memory requirement=5 bits x 20736 = 103,680 bits
- Parity Memory = 4 channels x 5 bits x 20736 = 414,720 bits

If all this data storage was provided by FPGA block memory, 32 Virtex-II type BRAMS would be required.

It is important to note that the systematic memory is read in a linear and an interleaved sequence. This memory block must therefore be capable of true random access at the full system clock rate. Parity memory is always addressed as an increasing count from zero to block size+tail bits.

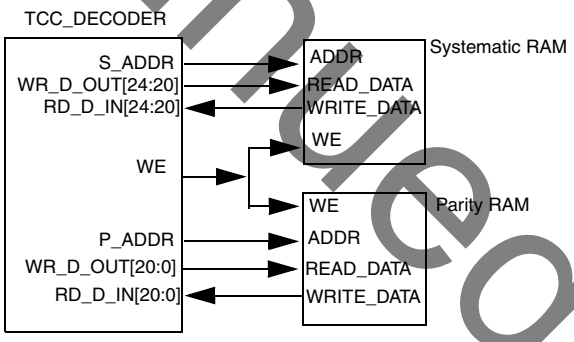

Figure 7: **Example of External Memory Interface**

Signal Timing

The turbo decoder core is a synchronous core operating on the rising edge of the clock. All input signals are read and all output signals may be changed on the rising edge of the clock. The only exception to this is the asynchronous clear signal, ACLR. When an optional CE signal is used the core state does not change when CE is low; all input signals are ignored and the core outputs remain the same. If the optional CE signal is not used the core operates as though CE is permanently high (enabled).

Figure 8 shows the input timing for the decoder. The processing is started when a valid FD is asserted on a rising clock edge. A valid FD occurs when FD, CE, and ND are all high. On receiving a valid FD pulse the RFFD signal will go low to indicate that the core is no longer ready to receive a first data pulse. RFFD will remain low until the core is ready to process another block of data.

The first input data, d0, is read from the DIN port on the same clock edge as the valid FD pulse is detected (Figure 8). Also at this time the ITERATIONS and BLOCK_SIZE_SEL inputs are read to determine the cores operation on this data block.

The core will read the next input data values on successive rising edges of the clock unless CE or ND is low, in which case the input data is ignored. During the data input process the RFD signal remains high to indicate that the core is ready to accept further input data.

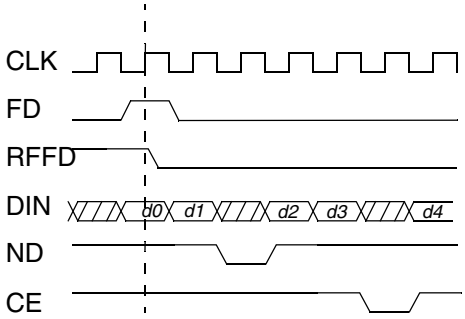

Figure 8: **Start of Input Timing**

As shown in Figure 9, at the end of the input cycle, after BLOCK_SIZE+6 data values have been input, the RFD signal will go low to indicate that all input data has been read. Once RFD is low further ND signal changes are ignored. The RFD signal going low also indicates that the core is moving from its input to its decoding phase.

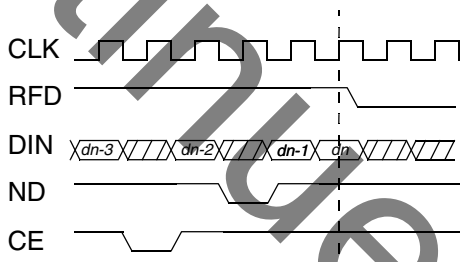

Figure 9: **End Of Input Timing**

After the decoder has performed the required number of iterations the RDY signal will be driven high to indicate that there is valid data on the DOUT port (Figure 10).

If selected, the IT_RDY signal performs in the same way as the RDY signal except that the IT_RDY signal indicates valid output data for each iteration. For example, if two iterations are required the relationship between the RDY and the IT_RDY signals will be as shown in Figure 11. In this case IT_RDY is active to indicate that data is available from each iteration while RDY only indicates when the valid data is output on the final iteration.

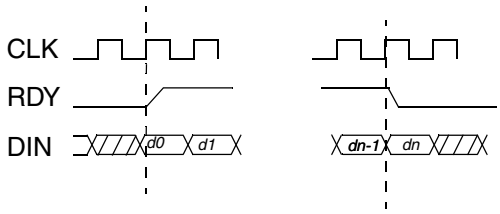

Figure 10: **Output Timing**

When the core approaches the end of the data output phase, it will take RFD and RFFD high to indicate that core is ready to accept a new block. These signals will go high before the last data has been output in order to maximize throughput.

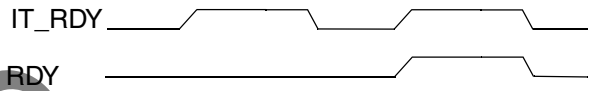

Figure 11: **Relationship between RDY and IT_RDY**

Performance and Resource Usage

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 excellentperformance. 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.

BER Performance

Figure 12 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. Figures 13, 14, and 15, show further results for the performance of the core.

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 12.The results have been generated with the following parameters:

- Sliding window size=32
- Input Bit width= 2 integer and 3 fractional bits
- Internal calculation width = 6 integer bits and 3 fractional bits
- Error count is the minimum number of errors detected to generate each point on the curve
- The input data has been scaled as described in the *Data Input Format* section
- MAX SCALE algorithm

Resource Requirements

Table 3 shows the resource requirements and the performance of the Turbo Decoder core.

The results for the case where internal memory is used assume that all available optional signals are used (excluding the internal memory interface). For the case where the external memory interface is used, then SCLR, ACLR, CE, and IT_RDY have not been used in the core. This change in the selection of optional signals and the use of internal or external memory accounts for the variation in the slice and block memory counts.

These results have been obtained using the MAX SCALE algorithm with the state metrics being 6 integer and 3 fractional bits and the input being 2 integer and 3 fractional bits.

It can be expected that the maximum clock frequency and the slice count will vary slightly according to the application, the version of the software implementation tools, and the tool options selected.

Table 3: **Resource Requirements and Performance**

Slices	Block Rams	Hardware Multiplier	FPGA	Max Clock Rate	Comment
1442	15	2	XC2V500 FG256-6	110MHz	With external memory interface
1620	47	2	XC2V500 FG256-6	111MHz	With internal memory
1442	15	2	XC2VP20 FF896-7	142MHz	With external memory interface
1618	47	2	XC2VP20 FF896-7	135MHz	With internal memory

Core Performance

The throughput for the core can be calculated using Table 4, which shows the number of clock cycles that the core takes to decode data with various number of iterations. These numbers of clock cycles are those measured from the first input into the core (that is, when FD is active) to the last decoded data being output from the core. The total number of clock cycles depends on the window size that has been selected and Table 4 shows the performance with window sizes of 32 and 64.

Within Table 4 the variable **BS** represents the block size. For example, if the throughput of the core is required for a block size of 378, with 3 iterations and a window size of 32, then the total number of cycles is (7 x 378)+ 533 = 3179.

The number of cycles taken by the core does not vary if the algorithm is changed.

Assuming

- 2 integer, 3 fractional bits for the input,
- 6 integer, 3 fractional bits for the internal precision,
- MAX SCALE algorithm,
- window size of 32,

the maximum clock rate in a XC2VP20 FF896-7 was measured to be 135MHz. Using the values calculated in Table 4 and the clock rate the throughput can be calculated as shown in Table 5.

Table 4: **Turbo Decoder Clock Cycles**

Number of Iterations	Window Size 32	Window Size 64
1	(3xBS) + 201	(3xBS) + 362
3	(7xBS) + 533	(7xBS)+ 950
5	(11xBS) + 865	(11xBS)+1538
7	(15xBS) + 1198	(15xBS) + 2126
9	(19xBS)+1530	(19xBS)+2714

Table 5: **Throughput Rates (Mbits/sec)**

Block Size	3 iterations	5 iterations	7 iterations	9 iterations
378	16.0	10.1	7.4	5.8
1146	18.0	11.4	8.4	6.6
3066	18.8	11.9	8.7	6.9
9210	19.1	12.1	8.9	7.0
20730	19.2	12.2	8.9	7.0

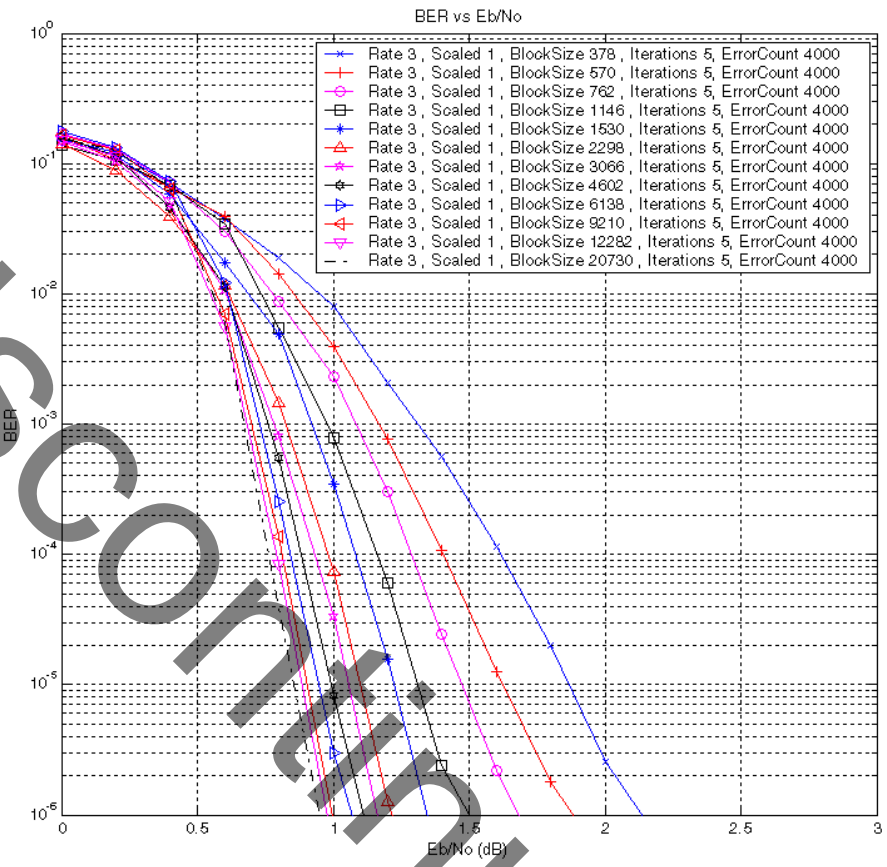

Figure 12: **Measured BER Performance of the MAX_SCALE Algorithm**

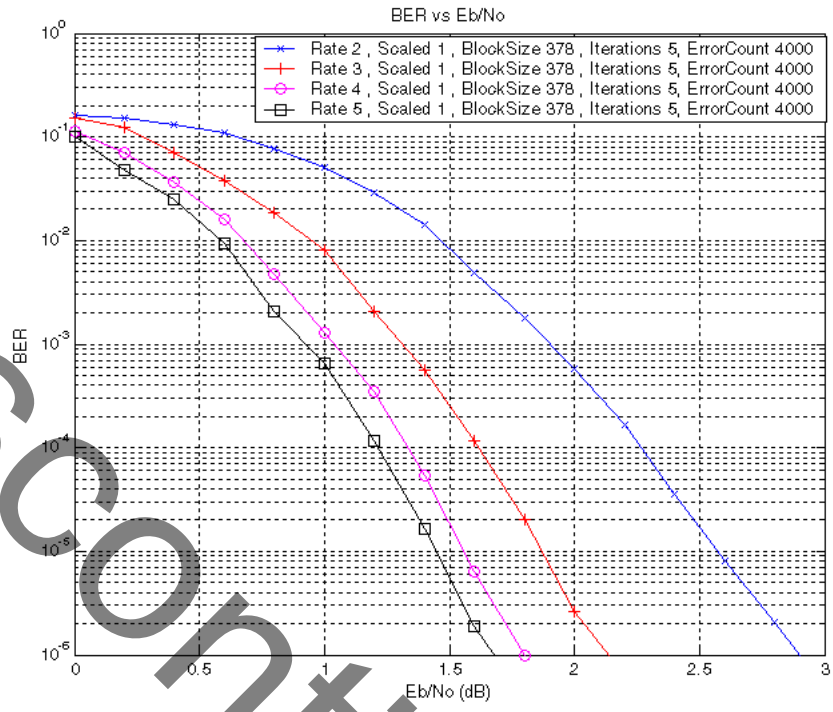

Figure 13: **Measured BER Performance for Different Code Rates**

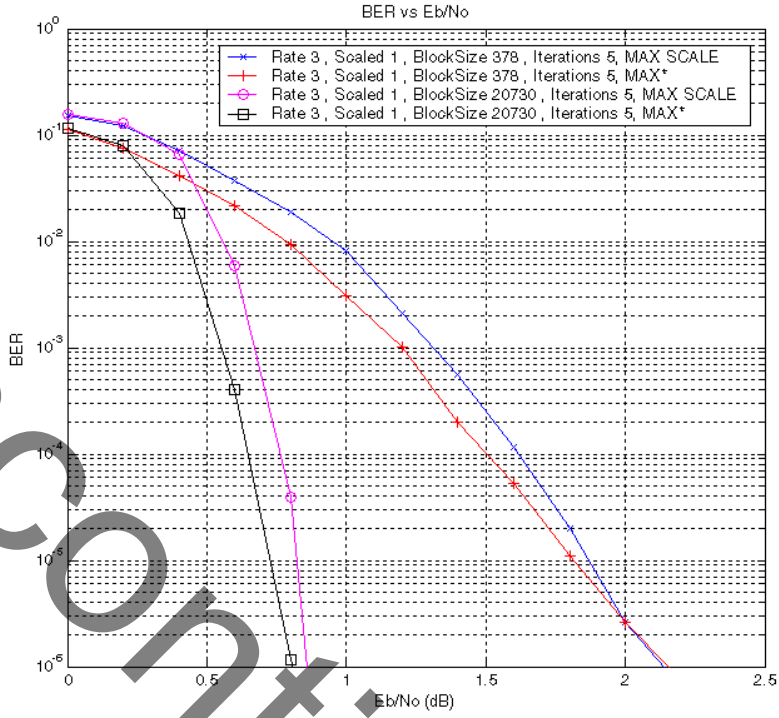

Figure 14: **Comparison of MAX* and MAX_SCALE Algorithms**

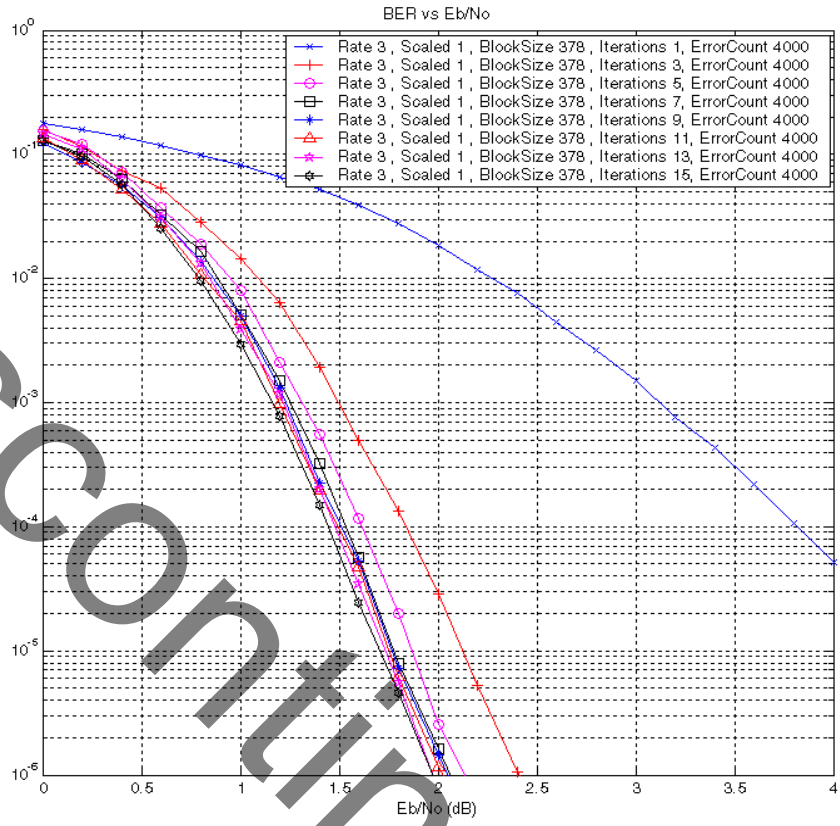

Figure 15: **Measured BER Performance for Different Numbers of Iterations**

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.

Ordering Information

This Xilinx LogiCORE™ product is provided under the terms of the SignOnce IP Site License. For additional information about the core and how to obtain a license for the core, please see the TCC Decoder product page. For pricing and availability of Xilinx LogiCORE products and software, please contact your local Xilinx sales representative or visit the Xilinx Silicon Xpresso Cafe.

France Telecom, for itself and certain other parties, claims certain intellectual property rights covering Turbo Codes technology, and has decided to license these rights under a licensing program called the Turbo Codes Licensing Program. Supply of this IP core does not convey a license nor imply any right to use any Turbo Codes patents owned by France Telecom, TDF, or GET. Please contact France Telecom for information about its Turbo Codes Licensing Program at the following address: France Telecom R&D, VAT /TURBOCODES 38, rue du Général Leclerc 92794 Issy Moulineaux Cedex 9.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/11/03	1.0	Initial Xilinx release.
04/22/04	1.1	Added further performance data.
4/28/05	2.0	Added support for Spartan-3E.