3GPP Turbo Decoder v4.0

Introduction

The Turbo Convolution Code (TCC) Decoder core is used in conjunction with a TCC Encoder to provide an extremely effective way of transmitting data reliably over noisy data channels, and is designed to meet the *3GPP Mobile Communication System* specification.

Features

- Drop-in module for Virtex™-4, Virtex-5, Virtex-6, Spartan™-3, Spartan-3E, Spartan-3A/3AN/3A DSP, Spartan 6 FPGAs
- Implements the 3GPP/UMTS specification [1]
- Core contains the full 3GPP interleaver
- Full 3GPP block size range supported, that is, 40 - 5114
- Dynamically selectable number of iterations 1-15
- Number representation: two's complement 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
- Fast Termination option
- Support for rate 1/3 or rate 1/5 coded input
- Available through the Xilinx CORE Generator™ 11.2 and later

Applications

The TCC Decoder core is designed to meet the *3GPP Mobile Communication System* specification [1].

General 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 defined by the Shannon limit [2].

When a decode operation is started, the core accepts the block size and the number of iterations from two input ports. A block data load stage follows, in which the systematic and parity data is read into the core in parallel on a clock-by-clock basis and stored in internal block RAM. The core then starts the decoding process and implements the required number of iterations. Decoding may optionally be terminated earlier if the Fast Termination unit is included. Finally, the decoded bit sequence is output. The entire sequence is automatically controlled from a single first data signal and requires no user intervention. All the interleaving operations required in the 3GPP specification are handled automatically within the core.

The core expects two's 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 two or three integer bits and between one and four fractional bits. The precision of the internal calculations can also be controlled using six or seven integer bits and between one and four fractional bits. (The number of input fractional bits must be less than or equal to the number of internal calculation fractional bits.)

Algorithm

The full TCC Decoder algorithm is extremely computationally intensive, and for this reason approximations must be made to make the algorithm usable in practice. The algorithm used in this core is MAX SCALE which produces a good trade-off of performance versus complexity; reference [3] describes this approach in more detail.

Code Rates

The core may be configured to assume that either rate 1/3 (as in the 3GPP standard) or rate 1/5 data will be used as input. Any rate matching algorithms should be implemented externally to this core, and the resulting input to the TCC Decoder must then be formatted in rate 1/3 or 1/5 format as appropriate. Punctured input bits should be replaced with a zero value (this represents equal probability of a hard 1 or 0 in the LLR format), while repeated input bits should be combined appropriately (for example, add and saturate to the required input width). It should be noted that although a core configuration which uses inputs in a format suitable for rate 1/5 coding will also support rate 1/3 by suitably puncturing the input parity values, there will be a significant additional memory requirement to store the data in rate 1/5 format.

Fast Termination

The number of iterations is generally determined by reading the ITERATIONS port at the start of each new decoding process. Each full iteration consists of two half iterations (SISOs), one using normal data and the second using interleaved data. The decoder normally continues iterative decoding for the specified number of iterations each consisting of two SISOs. However, the results from each SISO can be examined to determine whether sufficient decoding accuracy has been achieved and action taken to cease decoding upon this occurrence. This is commonly known as Fast Termination (or Early Termination). The most basic form of Fast Termination uses the hard data decision results on each SISO and compares against those for the previous SISO [4]. After the count of consecutive SISOs for which the data has matched has reached a threshold level, or the specified maximum iteration count has been reached, the decode operation is terminated and the decoded data is output. This function has been implemented as an optional module within the core and is made available to the user by changing a core parameter. The threshold level is read from the FT_THRES port at the start of each block decode operation.

Input/Output Pins

Figure 1 displays the signal names and Table 1 defines them.

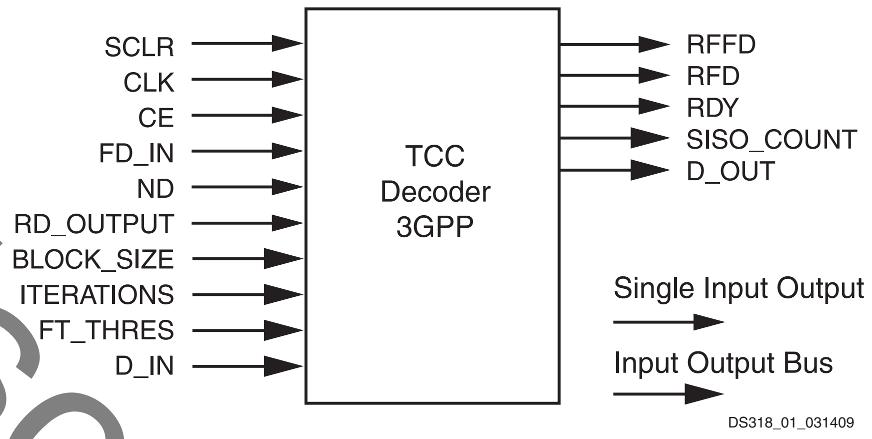


Figure 1: **TCC Decoder Input and Output Ports**

Table 1: **Core Signal Pinout**

Pin	Direction	Port width (bits)	Description
FD_IN	Input	1	**First Data In**: When asserted (High) on a valid clock edge, the decoding process is started.
ND	Input	1	**New Data**: When asserted (High) on a valid clock edge, a new input value is read from the D_IN port.
SCLR	Input	1	**Synchronous Clear**: When asserted (High) on a valid clock edge, the decoder is reset.
CE	Input	1	**Clock Enable**: When deasserted (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 the core must implement.
BLOCK_SIZE	Input	13	**Block Size:** The block size of the current decode operation.
FT_THRES	Input	3	**Fast Termination Threshold**: Specifies the threshold to which the count of matching SISO decode results is compared.
D_IN	Input	varies (see later)	**Data Input**: Consists of the systematic and parity data input.
RD_OUTPUT	Input	1	**Read Output**: When asserted (High) with RDY High, a new output value is output on the D_OUT port.
RFFD	Output	1	**Ready For First Data**: When asserted (High), the core is ready to start another decoder operation.

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

Pin	Direction	Port width (bits)	Description
RFD	Output	1	**Ready For Data:** When asserted (High), the core is ready to accept input on the D_IN port.
RDY	Output	1	**Ready**: Asserted (High) when an entire decoding operation has been complete and data is available on the D_OUT port.
D_OUT	Output	varies (see later)	**Data Out**: This is the decoded output from the core.
SISO_COUNT	Output	4	**SISO Count:** Output that shows the current number of SISO operations performed. If read when RDY is High, it indicates the total number of SISO operations implemented in the current decode operation.

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 that is used to indicate if the next rising clock edge is valid. When CE is High, rising clock edges allow the decode process to continue, but if CE is Low, then the core operations are suspended and the core remains in its current state. All signals are ignored when CE is Low.

First Data In (FD_IN)

FD_IN is used to start the decode operation. When FD_IN is High on a valid clock edge, then the first data is read from the D_IN port. Simultaneously, the same clock edge is used to read the values of the BLOCK_SIZE, ITERATIONS, and FT_THRES (if selected) ports which define the block size, the number of iterations and the fast termination parameter. FD_IN should only be held High for a single clock cycle. If ND is selected, an FD_IN pulse is only valid if ND is asserted High; FD_IN is ignored if RFFD is Low.

New Data (ND)

ND is an optional pin that is used to indicate the presence of a new data on the D_IN port, including tail bits. For example, if the input block size is 378, then 378+6(tail bits) active High ND samples are required to load in the complete block before the decode operation commences. After all the expected input data has been read into the core, the ND signal is ignored until the next block can be input.

Synchronous Clear (SCLR)

SCLR is an optional pin which, when asserted High on a valid clock edge, resets the core 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 is automatically in the reset state, so no further SCLR is required before a decode operation can take place.

Iterations (ITERATIONS)

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

Block Size (BLOCK_SIZE)

BLOCK_SIZE is a 13-bit input port that represents the specific block size to be implemented. Like the ITERATIONS signal, BLOCK_SIZE is read on a valid clock edge when FD_IN is High. (Note that block size values do not include tail bits.) The core is designed to function correctly only if the block size is within the 3GPP specification, that is, 40-5114.

Fast Termination Threshold (FT_THRES)

FT_THRES is an optional port that allows users to increase average throughput by terminating decode operations before the maximum specified iteration count, provided that a certain condition has been met. The condition tested is that the number of consecutive SISO (half-iteration) decode operations that have completed with identical results to the previous SISO operation has reached a threshold value. The value for this threshold should be specified in the following way:

- 0: Fast Termination disabled
- 1: Terminate when results from two consecutive SISOs match
- 2: Terminate when results from three consecutive SISOs match and so forth, up to...
- 7: Terminate when results from seven consecutive SISOs match

The ITERATIONS input still applies when Fast Termination is enabled and acts as an upper limit for iterations.

Note: The fact that a number of SISO operations produce the same data is not a guarantee of correct decoded data. However, Fast Termination is generally an effective technique for minimizing the number of iterations applied to a specific block. Its performance, however, does depend on the amount of data corruption within each block.

Data In (D_IN)

The D_IN port is a a single-input bus that accepts the N input channels of the systematic and parity data. For example, if two integer and three fractional bits are required for each soft input value, then a total of five bits are required for each of the N channels. The arrangements of the D_IN port for two examples with rates 1/3 and 1/5 are displayed in Figure 2. For the rate 1/3 example, the 18-bit input port is represented by D_IN[17:0], indicating that bit 17 is the MSB and bit 0 the LSB. The parity value from the non-interleaved data is represented by RSC1_0, while the parity value from the interleaved data is represented by RSC2_0. Figure 2 shows how the user must map each of the systematic and parity bits to each of the D_IN bits to obtain the correct functionality.

The data input to the decoder core is assumed to be encoded using the Xilinx CORE Generator 3GPP Turbo Encoder core. The basic description of this core is given later in this data sheet to ensure the correct mapping between the encoder outputs and the decoder inputs.

See "Data Input Sequence" on page 9 for additional information about the correct data input sequence.

(a) Rate 1/3 encoded D_IN input, with 3 integer bits, 3 fractional bits per field.

17 — 12	11 — 6	5 — 0
RSC1 Systematic	RSC1 Parity0	RSC2 Parity0

(b) Rate 1/5 encoded D_IN input, with 2 integer bits, 3 fractional bits per field.

24 — 20	19 — 15	14 — 10	9 — 5	4 — 0
RSC1 Systematic	RSC1 Parity1	RSC1 Parity0	RSC2 Parity1	RSC2 Parity0

DS318_02_031409

Figure 2: **Construction of D_IN Port (Examples)**

Ready For First Data (RFFD)

When RFFD is asserted High, the core is ready to accept an FD_IN signal to start a new decode operation. When a valid FD_IN signal is detected, the RFFD signal is deasserted and remains deasserted until it is safe to start another block.

Ready For Data (RFD)

When RFD is asserted High, it indicates that the core is ready to accept new input data. RFD remains High during the input of the entire block. When block size (plus tail bits) of data has been input, the RFD signal goes Low to indicate that the core is no longer ready to accept data.

Ready (RDY)

RDY is asserted High after completing the number of iterations defined by the ITERATIONS port on the valid FD_IN signal, or when the block decode operation has been terminated early (Fast Termination). RDY is asserted to indicate that the data on the D_OUT port is now the final result of the decode operation.

Data Out (D_OUT)

D_OUT is the hard-coded output from the decode process. Each block of data is output on a serial basis with RDY indicating that the data is valid. The output width of the D_OUT port can be changed so that multiple data bits are output on each clock cycle. The RDY signal automatically compensates for any changes in the output bit width. For example, with a single bit output width, a decoded block size of 40 is output in 40 cycles, with RDY active for the 40 cycles of valid output data. If the output width is set to 32, a decoder block size of 40 is output in 2 cycles. RDY is also only active for 2 cycles in this case. Thirty-two values are valid on the first RDY cycle, while only the eight LSBs are valid on the second active RDY.

Read Output (RD_OUTPUT)

When selected, the optional RD_OUTPUT pin can be used to control the data output. When RDY first goes High to indicate that new data is valid, the first data is also output on the D_OUT port. When RD_OUTPUT is selected, the output remains in this state until RD_OUTPUT is asserted. Each time RD_OUTPUT is asserted, the next valid output data value is put onto the D_OUT port. RD_OUTPUT, therefore, is effectively a read enable for the data output.

The core is pipelined to allow one block's result to be in the output stage (RDY active) while the next block is being processed. If the next block's decode operation finishes before the previous block has been fully output, then the RFFD signal is held Low until the output operation has completed. RDY always goes Low between different output blocks to indicate the end and start of a new block.

Siso Count (SISO_COUNT)

The optional SISO_COUNT port can be used to monitor the number of SISO operations (half iterations) that the core implements within the decode operation. If this port is enabled with a fixed number of iterations, then as RDY is active it simply indicates a value of 2 x iterations. It is most useful if this port is used in conjunction with the fast termination circuit. If the maximum number of iterations was set to 5, for example, then the maximum value of SISO count is 2x5=10. However, if the RDY signal becomes active when SISO_COUNT=8, for example, it means that the core has met the fast termination criteria after only four iterations. By monitoring the SISO_COUNT value and adjusting the FT_THRES value, the user can attempt to increase system throughput by implementing only the minimum number of iterations to correctly decode the data.

Turbo Encoder

The data input to the Turbo Decoder core must be generated by the Xilinx 3GPP Turbo Encoder to provide the correct data format. For clarity, a brief description of the data output requirements of the Turbo Encoder is provided here to identify the input requirements of the decoder core.

Figure 3 shows the basic structure of the Turbo Encoder. It consists of two identical Recursive Systematic Convolution (RSC) Encoders, one unit that processes the original input data, and a second unit that processes an interleaved version of the original input data. Usually, the original input is delayed so that the first and successive outputs from both RSCs occur on the same clock cycle.

The output from each of the RSCs consists of the original input bit, the systematic bit, and either one (coding rate 1/3) or two (coding rate 1/5) parity bits created by the circuit shown in Figure 4. Note that for 3GPP implementations, where the coding rate is always 1/3, the Y0 output corresponds to RSC*_0 and the Y1 output is unused.

Figure 4 shows the control at the RSC input that switches between new data input and a feedback input. When *block_size* values have been output from the encoder, these control switches are switched over to create *tail bits* that are used to force the RSC units to a known state. Each of the two RSC units creates three sets of data values during the tail bit generation. The output of the Turbo Encoder (and the input to the Turbo Decoder) always consists of (block_size+6) sets of data values. See the 3GPP specification for additional information.

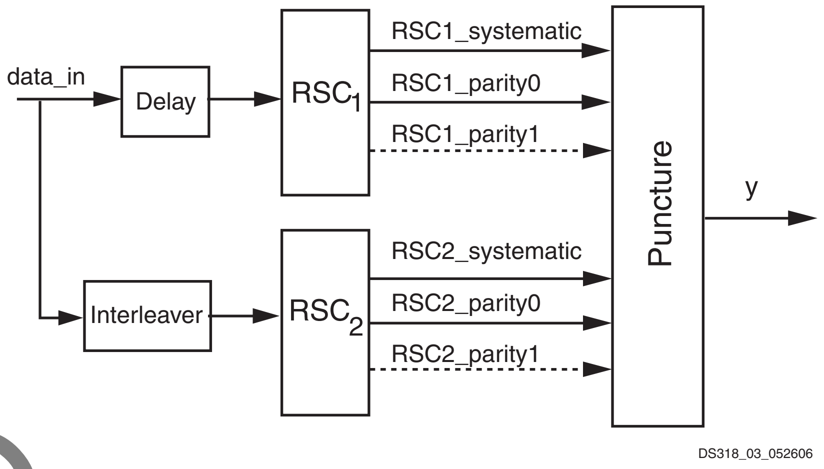


Figure 3: **Turbo Convolution Encoder**

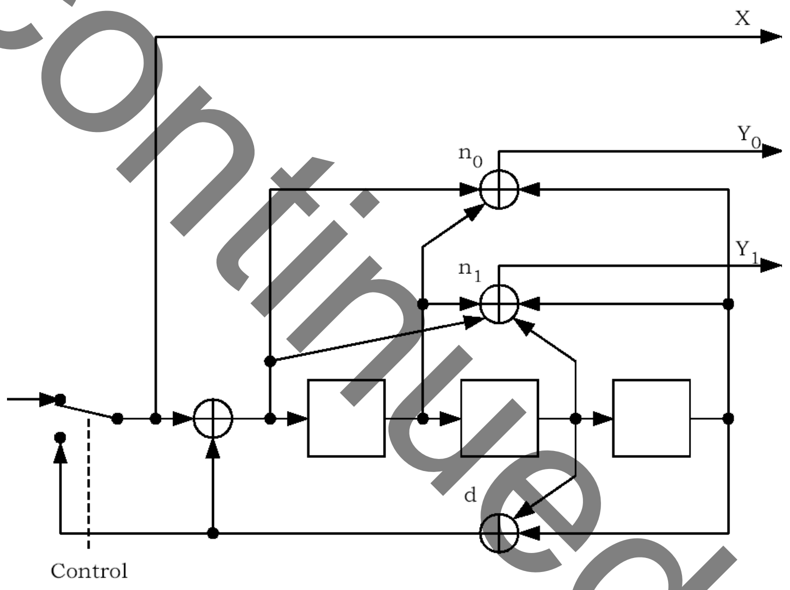
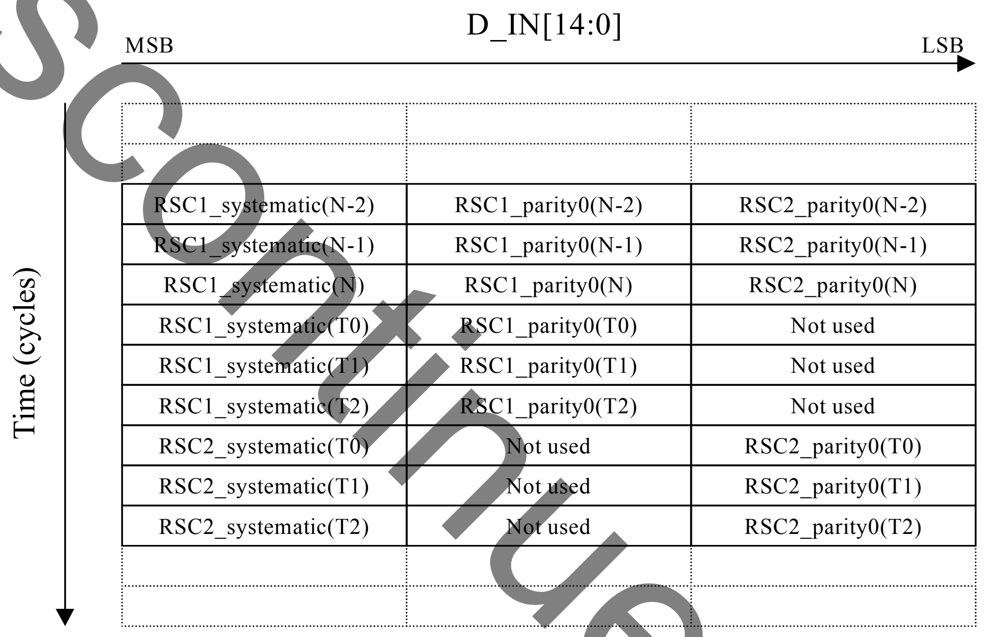


Figure 4: **Basic Recursive Systematic Convolution (RSC) Encoder**

Data Input Sequence

Figure 5 shows the data input sequence for a rate 1/3 example case where *block_size* = N. The input data consists of the output of the two RSC encoders. Figure 5 shows the transition between data input at the end of the block and tail bit input period. There are three tail bit cycles for each RSC (indicated by T0-T2 in the diagram), giving six cycles of tail bit input in total. During the RSC1 tail 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 parity inputs are similarly ignored during the RSC2 tail bit period; however, the systematic input bits are taken from the encoder RSC2_systematic output values this time. The example in Figure 5 assumes a normal 3GPP rate of 1/3 with no rate matching. If rate 1/5 coding is used, there would be two parity fields for each RSC rather than the single field illustrated, but the pattern of blanking of parity bits and the switching of systematic output from RSC1 to RSC2 would remain the same.



RSC1_systematic(N-2)	RSC1_parity0(N-2)	RSC2_parity0(N-2)
RSC1_systematic(N-1)	RSC1_parity0(N-1)	RSC2_parity0(N-1)
RSC1_systematic(N)	RSC1_parity0(N)	RSC2_parity0(N)
RSC1_systematic(T0)	RSC1_parity0(T0)	Not used
RSC1_systematic(T1)	RSC1_parity0(T1)	Not used
RSC1_systematic(T2)	RSC1_parity0(T2)	Not used
RSC2_systematic(T0)	Not used	RSC2_parity0(T0)
RSC2_systematic(T1)	Not used	RSC2_parity0(T1)
RSC2_systematic(T2)	Not used	RSC2_parity0(T2)

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

Data Input Format

The input to the D_IN port of the decoder is proportional to the Log Likelihood Ratio of the received data, that is, LLR(x).

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

This is the natural logarithm of the probability that a received symbol is a one or zero. The actual required input of the Turbo 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 still functions if an estimate of the LLR is made, but best performance is 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 Gausian noise with a mean μ and a variance σ^2, the LLR(x) can be calculated from:

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

This produces the following:

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

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

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

For a mean of 1, the input to the D_IN 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 1. This ensures 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:

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

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

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

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 0 dB, 1.8 dB, and 3 dB for code rates 1/2, 1/3, and 1/4, respectively (note that 3GPP specifies a fixed 1/3 code rate). 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 2 dB, 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 Bit Error Rate (BER) figures are quoted as using *scaled values,* it is assumed that the noise variance is accurately known. The less accurate the determination of noise variance of the input is, the greater the degradation in BER performance.

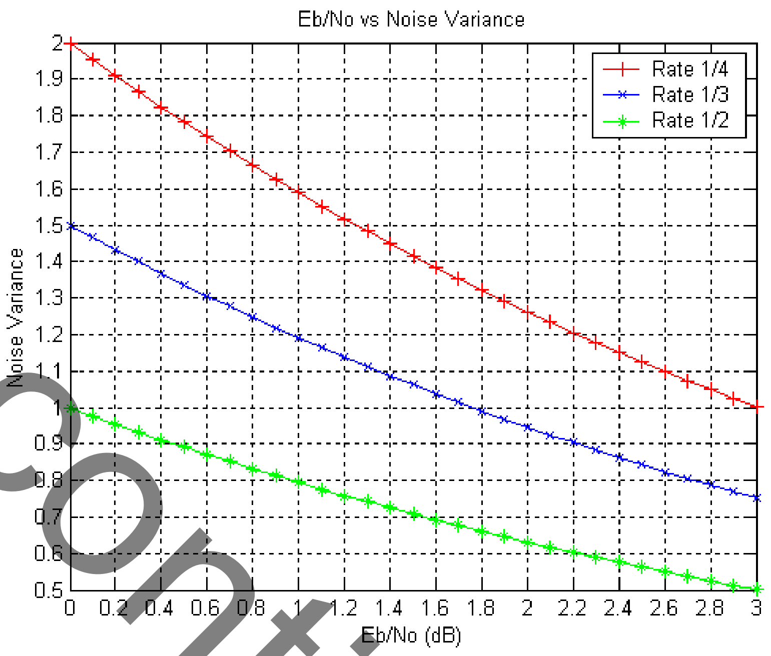


Figure 6: **Plot of Noise Variance against Eb/No for Different Rates**

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 can be changed on the rising edge of the clock. 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 7 shows the input timing for the decoder. The processing is started when a valid FD_IN is asserted on a rising clock edge. A valid FD_IN occurs when FD_IN, CE, and ND are all High. On receiving a valid FD_IN pulse, the RFFD signal goes Low to indicate that the core is no longer ready to receive a first data pulse. RFFD remains Low until the core is ready to process another block of data.

The first input data, d0, is read from the D_IN port on the same clock edge as the valid first data pulse is detected (Figure 7). Also at this time, the ITERATIONS, BLOCK_SIZE, and FT_THRES (if selected) inputs are read to determine the cores operation on this data block.

The core reads 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.

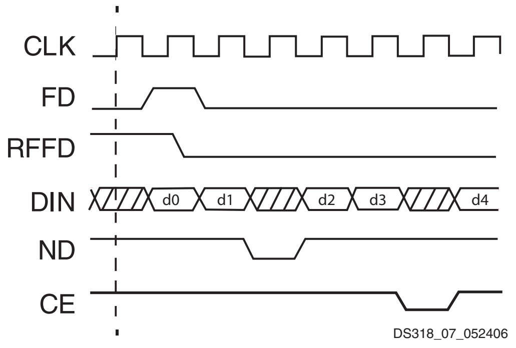


Figure 7: **Start of Input Timing**

As shown in Figure 8, at the end of the input cycle, after BLOCK_SIZE+6 data values have been input, the RFD signal goes Low to indicate that all input data has been read. After 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.

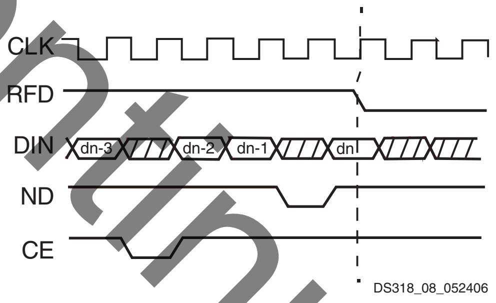


Figure 8: **End of Input Timing**

After the decoder has performed the required number of iterations, the RDY signal is driven High to indicate that there is valid data on the DOUT port (Figure 9).

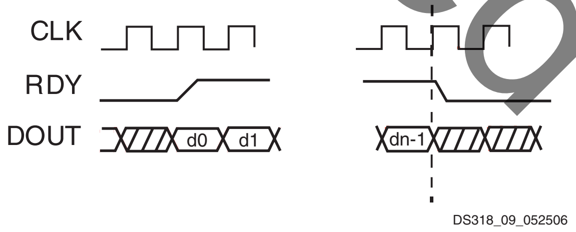


Figure 9: **Output Timing**

Performance and Resource Usage

The core has been extensively tested to optimize performance. Features such as algorithm type and numerical precision have been investigated with reference to BER performance and resource usage 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.

Resource Requirements

Table 2 and Table 3 show the resource requirements and the performance of the Turbo Decoder core for Virtex-5 and Virtex-6 parts. These results have been obtained for coding rate 1/3, with the state metrics being 6 integer and 3 fractional bits and the input being 2 integer and 3 fractional bits. Results are shown for the core with both no options and all optional functionality and ports.

Table 2: **Resource Requirements and Performance – Virtex-5 parts**

	Core 1		Core 2	
Options			ce,nd,sclr,rd_ouput,ft_thres,siso_count	
Xilinx Part	XC5VLX50		XC5VLX50	
LUT/FF Pairs[1]	4717		4750	
Slice LUTs	3483		3790	
Slice Registers	4115		4146	
Block RAMs (36k)	6		6	
Block RAMs (2x18k)	4		4	
DSP Blocks	0		0	
Speed Grade	-1	-3	-1	-3
Max Clock Freq[1,2]	299 MHz	385 MHz	299 MHz	381 MHz

Notes:

1. Area and maximum clock frequencies are provided as a guide. They may vary with new releases of the Xilinx implementation tools.
2. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.

Table 3: **Resource Requirements and Performance – Virtex-6 parts**

	Core 3		**Core 4**	
Options			ce,nd,sclr,rd_output,ft_thres,siso_count	
Xilinx Part	XC6VLX75T		XC6VLX75T	
LUT/FF Pairs(1)	3765		3910	
Slice LUTs	3712		3858	
Slice Registers	4062		4099	
Block RAMs (36k)	6		6	
Block RAMs (18k)	7		7	
DSP Blocks	0		0	
Speed Grade	-1	-3	-1	-3
Max Clock Freq(1,2)	285 MHz	349 MHz	291 MHz	367 MHz

Notes:

1. Area and maximum clock frequencies are provided as a guide. They may vary with new releases of the Xilinx implementation tools.
2. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.

BER Performance

BER performance results have been generated for the core by hardware co-simulation using MATLAB® products and System Generator. An FPGA-based hardware design was created to encode data, add noise, and then decode. A BER calculation circuit was used to detect differences between the original and decoded data to produce the results of Figures 10 through 13. This data, therefore, represents the true performance of the core.

Figure 10 shows the BER performance of the core for various block sizes against Eb/No, while Figure 11 illustrates the effect on BER of increasing the iteration count. The results of Figures 10 through 13 have been generated with the following parameters:

- Input data width = 2 integer bits and 3 fractional bits
- Internal calculation width = 6 integer bits and 3 fractional bits

The input data has also been scaled as described in "Data Input Format" on page 9.

The function of the Fast Termination feature is illustrated by Figures 12 and 13. The former demonstrates how closely the fast termination function can match the BER performance of a decoder that does not utilize fast termination, while the latter shows the potential throughput improvement in terms of reduction in *average* half-iterations (SISOs) required to complete the decode operation.

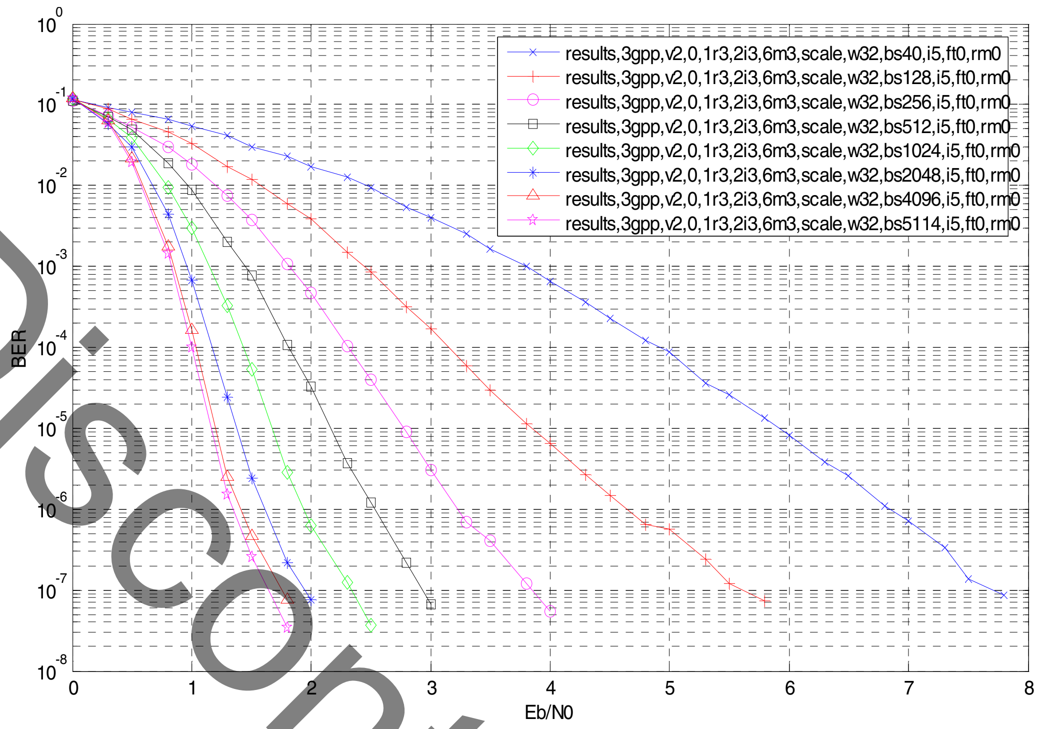


Figure 10: **Comparison of Different Block Sizes with Five Iterations**

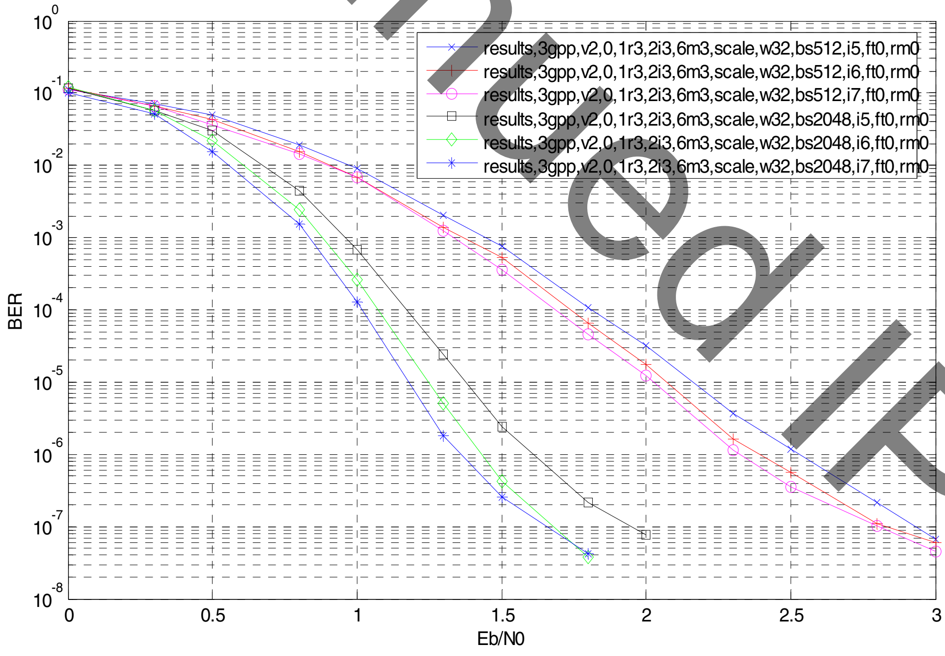


Figure 11: **Increasing the Number of Iterations for two Block Sizes**

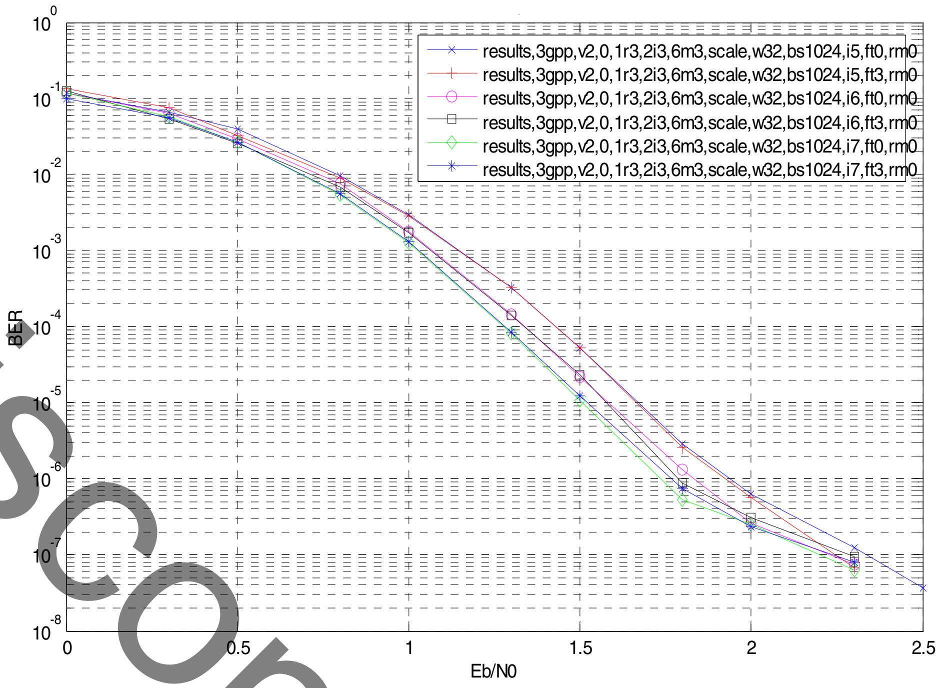


Figure 12: **Using Fast Termination with Fixed Block Size and Different Numbers of Iterations**

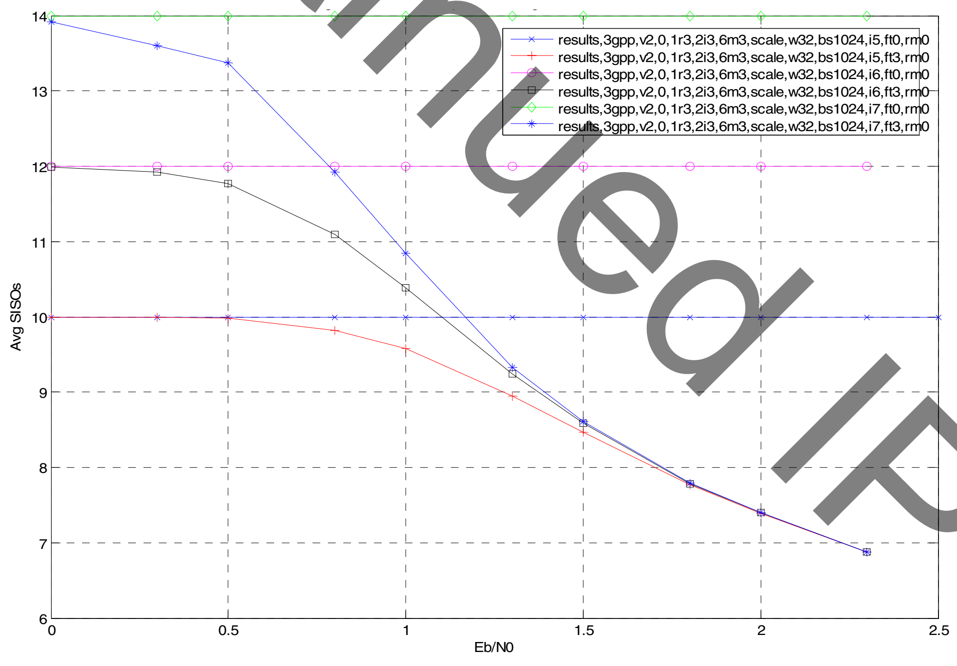


Figure 13: **Average Number of SISO operations within Each Decoder Operation**

Core Throughput Performance

Table 4 shows the number of clock cycles from the first data input to the first data output for various iteration and block size combinations. Therefore, this represents the latency of the core in clock cycles. Data input can operate in parallel with data output and, hence, the figures in Table 4 can also be used to calculate the throughput of the core. Table 5 shows the throughput of the core assuming a clock frequency of 300 MHz. (In reality, the next block can be started a few cycles before the previous block of data is output; this gives slightly improved performance figures than the results shown in Table 5.)

Table 4: **Turbo Decoder Latency**

Block Size	Number of Cycles for Different Number of Iterations			
	3	5	7	9
40	803	1291	1779	2267
64	1144	1856	2568	3280
128	1592	2560	3528	4496
256	2872	4608	6344	8080
512	4664	7424	10184	12944
1024	8632	13696	18760	23824
2048	16184	25600	35016	44432
4096	30520	48128	65736	83344
5114	37658	59346	81034	102722

Table 5: **Turbo Decoder Throughput (Mbits/s)**

Block Size	Throughput for Different Number of Iterations (Mbits/s)			
	3	5	7	9
40	14.94	9.29	6.74	5.29
64	16.78	10.34	7.47	5.85
128	24.12	15.00	10.88	8.54
256	26.74	16.66	12.10	9.50
512	32.93	20.68	15.08	11.86
1024	35.58	22.42	16.37	12.89
2048	37.96	24.00	17.54	13.82
4096	40.26	25.53	18.69	14.74
5114	40.97	26.00	19.04	15.02

Throughput can be seen to increase with block size. This is attributable to the fact that there are processing overheads which become more efficient when the block size increases. Reductions in throughput would be observed when the block size is changing often, because there is an overlap period from one block to the next. This overlap period is only efficiently utilized where the subsequent block size is the same as or larger than that of the previous block. For more information about the 3GPP Turbo Decoder performance, contact your local Xilinx sales representative.

Note: The overall performance of the decoder core can vary according to the tool version used, the complexity of the overall design, and the chosen Xilinx FPGA.

References

1. 3G TS.25.212 V3.3.0 (2000-06), *Multiplexing and Channel Coding (FDD)*, Technical Specification Group Radio Access Network, 3rd Generation Partnership Project.
2. C. Berrou, A. Glavieux, and P. Thitimajshima, *Near Shannon Limit Error-correcting Coding and Decoding Turbo Code*s, IEEE Proc 1993 Int Conf. Comm., pp1064-1070.
3. J. Vogt and A. Finger, *Improving the MAX Log MAP Turbo Decoder*, Electronics Letters 9th November 2000, Vol36 No 23, pp1937-1939.
4. A. Matache, S. Dolinar, and F. Pollara, *Stopping Rules for Turbo Decoders*, TMO Progress Report 42-142, August 15, 2000.

Evaluation

An evaluation license is available for this core. The evaluation version of the core operates in the same way as the full version for several hours, dependent on clock frequency. Operation is then disabled and the data output will not change. If you notice this behavior in hardware, it probably means you are using an evaluation version of the core. The Xilinx tools warn that an evaluation license is being used during netlist implementation. If a full license is installed in order for the core to run on hardware, delete the old XCO file and recreate the core from new.

Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Refer to the IP Release Notes Guide (XTP025) for further information on this core. There will be a link to all the DSP IP and then to the relevant core being designed with.

For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for the core being used. The following information is listed for each version of the core:

- New Features
- Bug Fixes
- Known Issues

Ordering Information

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. 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
France

This Xilinx LogiCORE™ product is provided under the terms of the SignOnce IP Site License. To evaluate this core in hardware, generate an evaluation license, which can be accessed from the Xilinx IP Evaluation page.

After purchasing the TCC Decoder core, you will receive instructions for registering and generating a Full license, which can be requested and installed from the TCC Decoder product page for use with the Xilinx CORE Generator. The CORE Generator is bundled with ISE® Foundation at no additional charge. Contact your local Xilinx sales representative for pricing and availability on Xilinx LogiCORE products and software.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/11/04	1.0	Initial release.
01/18/06	2.0	Updated for v2.0 decoder and Xilinx 8.1
09/28/06	3.0	Updated for v3.0 decoder and Xilinx 8.2
05/17/07	3.1	Updated for v3.1 decoder and Xilinx 9.1
06/24/09	4.0	Updated for v4.0 decoder and Xilinx 11.2

Notice of Disclaimer