

## Introduction

The IEEE 802.16e CTC decoder core performs iterative decoding of channel data that has been encoded as described in Section 8.4.9.2.3 of the IEEE Std 802.16e-2005 specification and corrigendum IEEE P802.16Rev2/D0b (June 2007). The IEEE 802.16e code is a parallel concatenated convolutional code with an input data block of 2N bits. Through parallel processing with parameterizable number of SISOs, this LogiCORE™ IP decoder core is capable of achieving high throughput. The decoded data rate reaches up to 220 Mbps with five iterations using eight SISOs at 286 MHz clock frequency.

## Features

- Supports Virtex®-6, Virtex-5, Virtex-4, Spartan®-6, Spartan-3, Spartan-3E, and Spartan-3A DSP FPGA families
- Supports all interleaver block sizes of the CTC OFDMA PHY mode including the HARQ and IR HARQ modes: 24, 36, 48, 72, 96, 108, 120, 144, 180, 192, 216, 240, 480, 960, 1440, 1920, and 2400 pairs
- Performs parallel processing with parameterizable number of SISOs to achieve high throughput and reduce latency
- Supports dynamic block size switching without interruption
- Programmable number of iterations dynamically changeable per block
- Adaptive rate change via puncturing interface
- Uses MAX-LOG-MAP algorithm with extrinsic scaling
- Parameterizable options for soft data input and extrinsic bits
- Clock speed exceeds 162 MHz in Virtex-4 speed grade -10, 196 MHz in Virtex-5 speed grade -1, and 225 MHz in Virtex-6 speed grade -1
- Decoded data rate depends on block size and varies between 44 Mbps to 63 Mbps when targeting Virtex-4, between 53 Mbps to 76 Mbps when targeting Virtex-5, and between 61 Mbps to 88 Mbps when targeting Virtex-6 (slowest speed grade, five iterations, and four SISO options)

- Latency depends on block size and varies between 5  $\mu$ s to 76  $\mu$ s when targeting Virtex-4, between 4  $\mu$ s to 63  $\mu$ s when targeting Virtex-5, and between 4  $\mu$ s to 55  $\mu$ s when targeting Virtex-6 (slowest speed grade, five iterations, and four SISO option)
- Fully synchronous design with single clock domain
- Double-buffered input to accommodate burst or continuous data
- Available using the CORE Generator™ v11.2 software, which is included with the ISE® 11.2 software

## Functional Description

The IEEE 802.16e code is a parallel concatenated convolutional code as illustrated in [Figure 1](#). The input data block contains 2N bits. The input data is split into even and odd samples (A and B, each of length N bits) and is fed to the first constituent encoder. The constituent encoder is a double binary Circular Recursive Systematic Convolutional (CRSC) encoder that creates two parity bits,  $Y_1$  and  $W_1$ , for every pair of input bits, A and B. The input bits are passed to the output to form the systematic symbols as shown in [Figure 1](#).

Prior to encoding with the second constituent encoder, the input data is interleaved with the interleaver described in Section 8.4 of the specification. The interleaved data is fed to a second constituent encoder that is identical to the first one. The second constituent encoder creates two parity bits,  $Y_2$  and  $W_2$ , for every pair of the interleaved input bits. The systematic bits are not transmitted from the second constituent encoder.

The definition of the CRSC encoder is shown in [Figure 2](#). Each CRSC encoder is initialized to the circulation state at the beginning of every input block. The circulation state is calculated as described in Section 8.4 of the specification.

After a block of data has been encoded, it is typically used to drive a modulation scheme. The modulated signal is then transmitted over a channel and demodulated by a receiver. The combination of the transmitter, channel, and receiver results in some form of signal degradation.

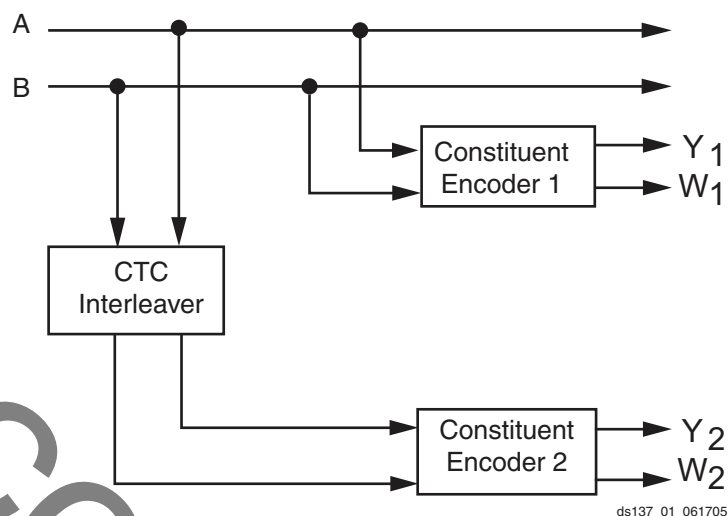


Figure 1: Block Diagram of IEEE 802.16e CTC Encoder

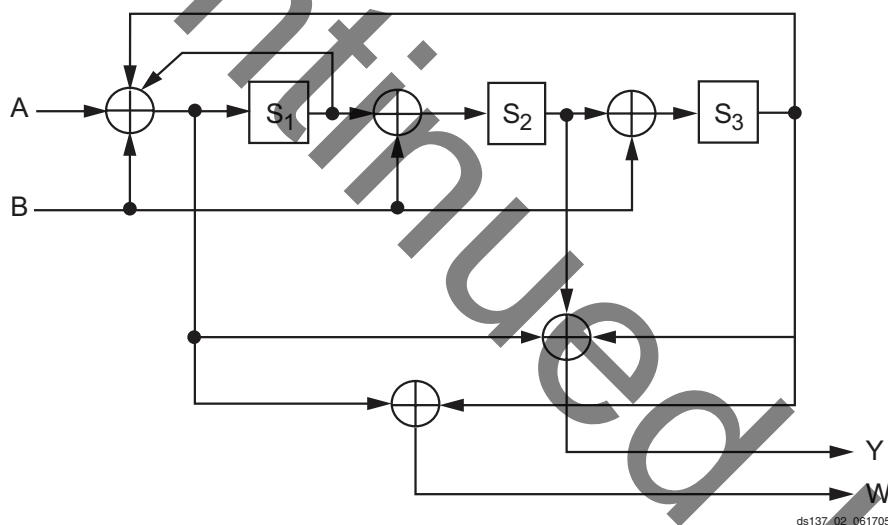


Figure 2: CRSC Constituent Encoder from IEEE 802.16e Specification

The data input to the IEEE 802.16e CTC decoder core is in the form of log-likelihood ratios (LLRs) on each code bit. As such, the demodulator output symbols must be used in conjunction with knowledge of the modulation scheme (that is, the constellation) to derive the LLRs for each code bit. It is the output of this LLR pre-processor that is used to drive the decoder. The number format for the LLRs is true two's complement with each sample quantized to widthd bits.

Figure 3 shows a simplified block diagram of the decoding process. The non-interleaved systematic samples, A and B, along with the parity samples from encoder 1,  $Y_1$  and  $W_1$ , are processed by SISO decoder 1. The interleaved systematic samples, A' and B', along with the parity samples from encoder 2,  $Y_2$  and  $W_2$ , are processed by SISO decoder 2.

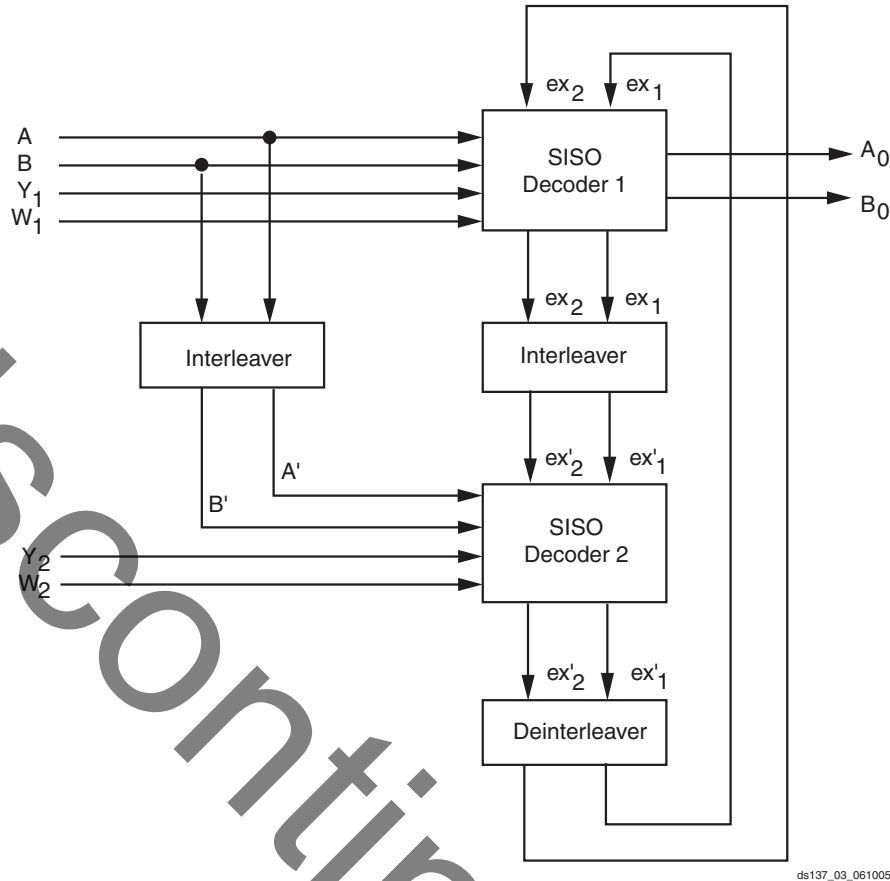


Figure 3: Block Diagram of CTC Decoding Algorithm

In addition to using the channel data, each SISO decoder uses extrinsic information from the other decoder to update its own extrinsic information output. The extrinsic information from SISO decoder 1 must be interleaved before being processed by SISO decoder 2. Similarly, the extrinsic information from SISO decoder 2 must be de-interleaved before being processed by SISO decoder 1. A half iteration occurs every time a single decoder finishes generating new extrinsics. A full iteration contains two half iterations.

The order in which decoding begins is somewhat arbitrary, but there are some practical advantages to starting with the interleaved data. In particular, by starting with SISO decoder 2, a full iteration occurs when decoder 1 has finished updating its output. The estimated information bit sequences, A<sub>0</sub> and B<sub>0</sub>, from decoder 1 is in non-interleaved order. In contrast, by starting with decoder 1, the full iteration ends with decoder 2, and the estimated bit sequences, A<sub>0</sub> and B<sub>0</sub>, must be de-interleaved before being processed downstream.

Another practical consideration in choosing the decode order has to do with the nature of the channel. In an Additive White Gaussian Noise (AWGN) channel, the order of the decoding should have no impact on Bit Error Rate (BER) performance. However, in a fading channel, the first half iteration should have some advantage if it is based on the interleaved data. For these reasons, the IEEE 802.16e CTC decoder core processes the interleaved data on the first half iteration.

It should be noted that the actual implementation of the decoding algorithm is different than what is indicated in Figure 3. For example, the SISO decoder is time shared between the two half iterations. Therefore, the hardware that implements SISO decoder 1 is the same hardware that implements SISO

decoder 2. Similarly, a single circuit is used to perform both the interleaving and de-interleaving functions for the extrinsics.

Although there is a factor of two savings in hardware by sharing resources between half iterations, the implementation of each decoder has a P-fold complexity increase over that shown in Figure 3. In particular, to increase data throughput, P SISO processors are used to process the data on each half iteration. This is accomplished by processing the same data block by different SISO processors for the 480, 960, 1440, 1920 and 2400 blocks and by processing different data blocks by different SISO processors for the remaining blocks.

The decoder implementation contains an input buffer that allows new code blocks to be written while still processing the previous code blocks (that is, double buffered). The input buffer stores the channel data in a set of P memories that are independently accessed by P SISO processors. During the interleaved iteration, systematic data is read in an interleaved fashion, while parity data is read in a non-interleaved fashion. During the non-interleaved iteration, the systematic and parity data are both read in a non-interleaved fashion.

In addition to an input buffer, the core also contains an output buffer. At the end of the final iteration, soft decoded data is written independently to the output buffer by all P SISOs. The soft decoded data stored in the output buffer drives the core output. Like the input buffer, the output buffer contains two pages of memory so that soft decoded data from new code blocks can be written while data from the previous code blocks is still being read.

## CTC Decoder Interface

A block diagram of the CTC decoder core interface is shown in Figure 4. The port definitions are given in Table 1.

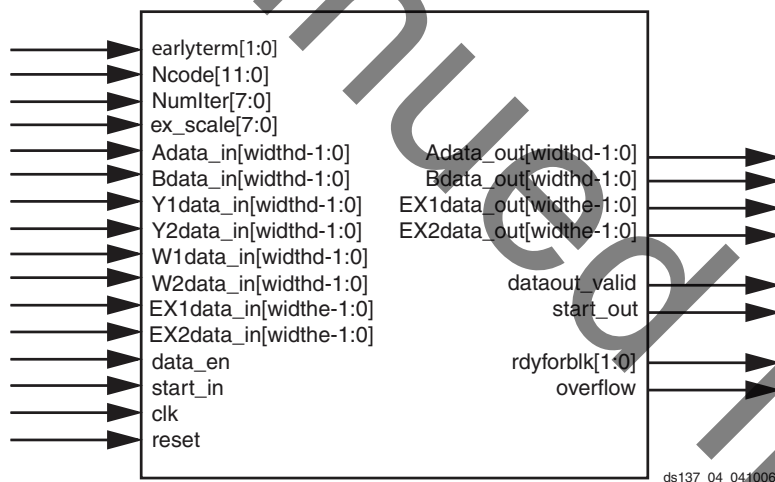


Figure 4: CTC Decoder Core Interface Signals

**Table 1: Core Port Definitions**

Signal	Direction	Description
earlyterm	Input	Early termination enable signal. When 00, early termination is disabled (the iterative process stops after a number of iterations given by NumIter), when 01 early termination scheme1 is enabled (comparing hard decisions over two successive iterations), and when 10, early termination scheme2 is enabled (comparing hard decisions over three successive iterations).
Ncode	Input	Data block length N in pairs.
NumIter	Input	Number of full iterations.
ex_scale	Input	Extrinsic scaling. The extrinsic data is scaled by a fractional number. Therefore, the input signal ex_scale is equal to 256 multiplies by that fractional number.
Adata_in	Input	First received non-interleaved systematic data.
Bdata_in	Input	Second received non-interleaved systematic data.
Y1data_in	Input	First received non-interleaved parity data.
Y2data_in	Input	First received interleaved parity data.
W1data_in	Input	Second received non-interleaved parity data.
W2data_in	Input	Second received interleaved parity data.
EX1data_in	Input	First received extrinsic data.
EX2data_in	Input	Second received extrinsic data.
data_en	Input	Data input enable. The input data is clocked into the core on the rising edge of clk when data_en = 1.
start_in	Input	Marks the start of an input code block. Must pulse High with the first data_en of a new code block.
clk	Input	Data input clock. Core is clocked on the rising edge of clk.
reset	Input	Active high synchronous reset.
Adata_out	Output	First output data.
Bdata_out	Output	Second output data.
EX1data_out	Output	First extrinsic output data.
EX2data_out	Output	Second extrinsic output data.
dataout_valid	Output	High when the Adata_out and Bdata_out signals contain valid data. Used as an enable signal for downstream processing of the Adata_out and Bdata_out signals.
start_out	Output	Marks the start of a decoded output block. Pulses High with the start of a new output block.
rdyforblk	Output	This signal is High when the decoder is ready for a new input block. If this signal is Low, the decoder is not ready for another input block. Pulsing start_in High when this signal is Low corrupts the operation of the core. Bit one controls the flow of the 480, 960, 1440, 1920, and 2400 blocks, while bit zero controls the flow of the remaining blocks.
overflow	Output	Input buffer overflow signal. This signal goes High when the start_in signal goes High while the rdyforblk signal is still Low.

## CORE Generator Parameters

Figure 5 shows the CORE Generator Graphical User Interface (GUI) of the IEEE 802.16e CTC decoder core. Table 2 has a detailed description of the GUI parameters.

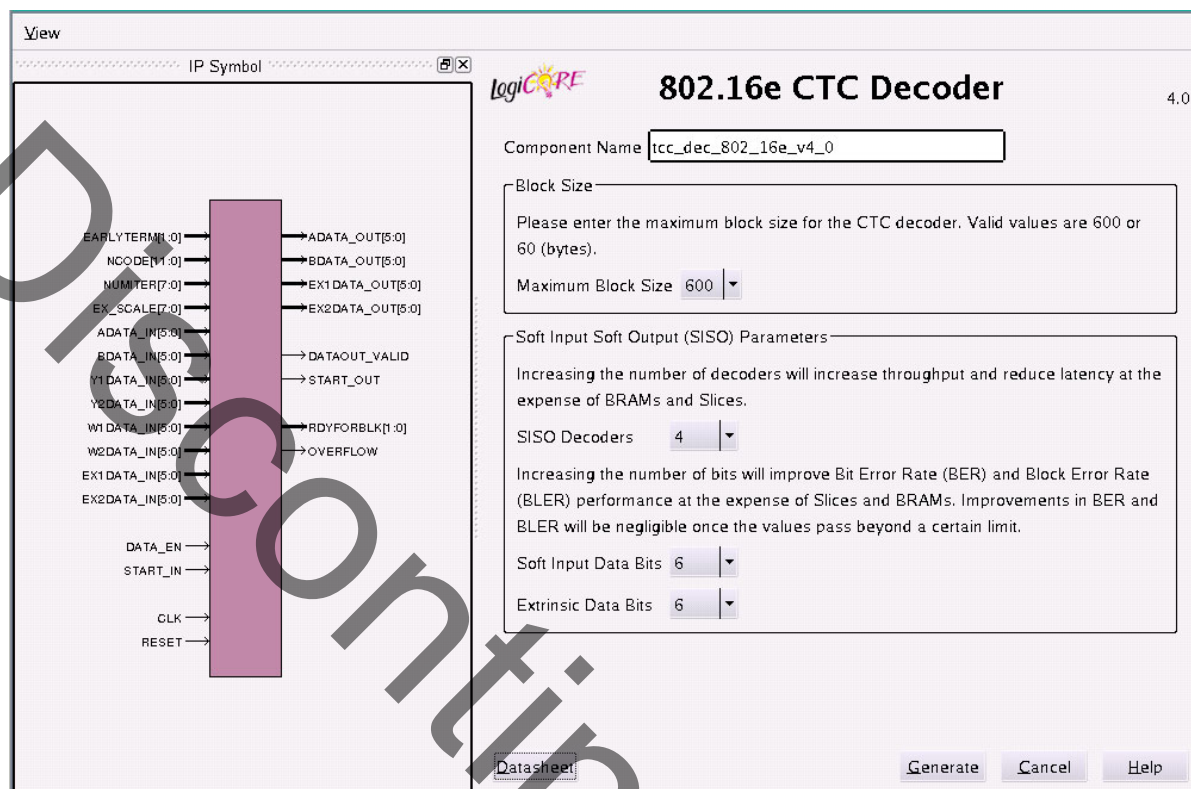


Figure 5: CTC Decoder Core GUI

Table 2: CORE Generator Parameters Description

GUI Parameter	Description
Block Size	Maximum block size supported by the core. Allowed values are 600 or 60 bytes. The 60 byte option is used to support the IEEE 802.16e (WiMax) specification and the HARQ mode, while the 600 byte option is used to support the optional IR HARQ mode.
SISO Decoders	Number of Soft Input Soft Output (SISO) Decoders. Allowed values are 1, 2, 3, 4, 5, 6, or 8.
Soft Input Data Bits	Number of soft input data bits. Allowed values are 4, 6, or 8. Other values like 5 or 7 are supported by generating the core with 6 or 8 soft input bits, and the user has to sign extend the incoming data to the core to 6 or 8 bits.
Extrinsic Data Bits	Number of extrinsic data bits. When soft input data bits is 4, allowed value is 6. When soft input data bits is 6, allowed values are 6 or 8. When soft input data bits is 8, allowed values are 8 or 10.

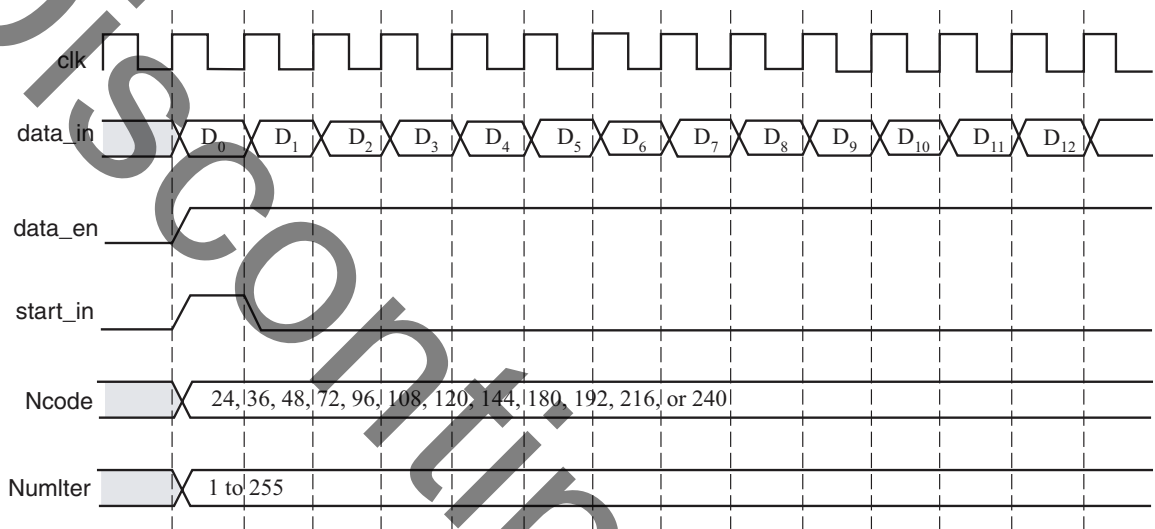
## Timing Interface

All core inputs and outputs are synchronous to the clk input. For best results, the inputs to the core should be driven from registers that are clocked with the rising edge of clk. Likewise, the outputs of the core should drive registers that are clocked with the rising edge of clk.



Figure 6 shows a timing diagram for the decoder inputs. Ncode, NumIter, earlyterm, and ex\_scale must be valid by the time start\_in pulses High. The start\_in signal pulses High for one clock and must occur with the first data\_en of each input code block. The data\_en signal is High for each data\_in (data\_in represents Adata\_in, Bdata\_in, Y1data\_in, Y2data\_in, W1data\_in, W2data\_in, EX1data\_in, and EX2data\_in) sample to be processed. Although the timing diagram shows data\_en High continuously for a block of time, the decoder also supports an arbitrary data\_en pattern.

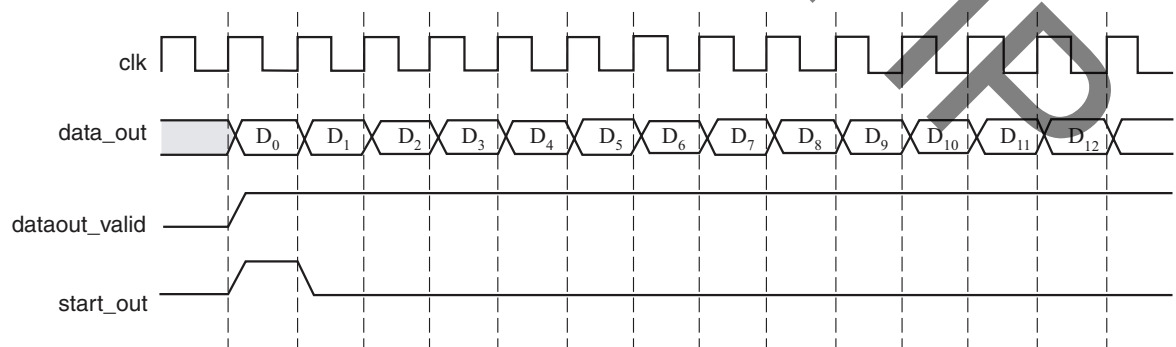
The Ncode, NumIter, earlyterm and ex\_scale signals can change from one block to the next without interruption of the data flow. These signals are latched into the decoder by the start\_in pulse and other internal strobes. This allows the Ncode, NumIter, earlyterm, and ex\_scale parameters to follow the input code block as it is being processed by various stages of the decoder.



ds634\_06\_05/21/07

Figure 6: Timing Diagram for CTC Decoder Inputs

Figure 7 shows a timing diagram for the decoder outputs. The start\_out signal pulses High with the first output bit of each decoded block. The dataout\_valid signal is High for as long as data\_out (data\_out represents Adata\_out, Bdata\_out, EX1data\_out, and EX2data\_out) contains valid data. The dataout\_valid signal can be used as a data enable for downstream processing of data\_out. The dataout\_valid signal bursts High for a duration of N clks. If necessary, the decoder output can be combined with an external FIFO for further data flow control.



ds634\_07\_05/21/07

Figure 7: Timing Diagram for CTC Decoder Outputs

The decoder can accept a new code block only when the output signal `rdyforblk` (`rdyforblk(1)` is High for the 480, 960, 1440, 1920, and 2400 blocks and `rdyforblk(0)` is High for the remaining blocks). Therefore, before sending a new code block to the decoder, the user must check that the `rdyforblk` signal is High before proceeding. If the user pulses the `start_in` signal High when the `rdyforblk` signal is Low, the operation of the core is corrupted and must be reset before correct operation can resume. Upon reset, the `rdyforblk` signal goes High.

If the code block is *not* one of the last five blocks of the H-ARQ mode, then the decoder can process P code blocks, while a second P code block is being written. Therefore, the `rdyforblk(0)` signal remains High after the first P `start_in` pulses. If the first P code blocks are still being processed, the `rdyforblk(0)` signal goes Low upon seeing the second P `start_in` pulses.

If the code block is one of the last five blocks of the H-ARQ mode, then the decoder can process one code block while a second code block is being written. Therefore, `rdyforblk(1)` signal remains High after the first `start_in` pulse. If the first block is still being processed, the `rdyforblk(1)` signal goes Low upon seeing the second `start_in` pulse.

It is assumed that data from the code block continues to arrive even though the `rdyforblk` signal is Low. Thus, the `rdyforblk` signal truly is a “ready for block” indicator and not a “ready for data” indicator. The `start_in` pulse must go High for only one clock and only once per code block.

After the decoder finishes writing the first code block to the output buffer, the `rdyforblk` signal goes High and remains High until another `start_in` pulse arrives (assuming the second code block is still being processed). This sequence of events repeats for each subsequent input block.

## Latency

The latency of the decoder is a function of the block size, the number of iterations, the number of SISO processors, and the clock frequency. It is also dependent upon the current state of the decoder. In particular, the time it takes to decode a given code block can be large if the decoder is still iterating on the previous code block. In other words, part of the latency for the new code block is attributed to waiting for the decoder to finish iterating on the previous code block.

In some cases, the decoder finishes iterating on the previous code block before the new code block is completely written to the input buffer. In these cases, the input buffer is the bottleneck, and the decoding latency for a given block is the same as the *first block latency*.

The latency of the IEEE 802.16e CTC decoder core is defined as the number of clocks from the time the last sample of the first code block is received to the time the first sample of the first code block is coming out of the decoder. Latency in terms of number of clocks is given by [Equation 1](#).

$$L = 2N_i[2N' + C] + 25$$

Equation 1

where  $N_i$  is the number of iterations,  $N'$  is the data block size in pairs ( $N$ ) divided by the number of SISOs ( $N_s$ ), when  $N$  is equal to 480, 960, 1440, 1920, or 2400 pairs; otherwise  $N'$  is the block size in pairs ( $N$ ), and  $C$  is a constant that is equal to 21 when early termination is disabled; otherwise,  $C$  is equal to 25.



The latency in microseconds is defined as in Equation 2:

$$Latency = \frac{L}{f_{clk}}$$

Equation 2

where  $L$  is the latency in terms of number of clocks and  $f_{clk}$  is the system clock frequency in MHz.

Table 3 shows the latency of the IEEE 802.16e CTC decoder core in  $\mu s$  for all the block sizes versus number of full iterations and clock frequency when the number of SISOs is two. Table 4 shows the latency of the IEEE 802.16e CTC decoder core in  $\mu s$  for all the block sizes versus number of full iterations and clock frequency when the number of SISOs is four. Table 5 shows the latency of the IEEE 802.16e CTC decoder core in  $\mu s$  for all the block sizes versus number of full iterations and clock frequency when the number of SISOs is five. The numbers are based on the above two equations. In general, for a fixed number of iterations, latency is reduced by using more SISOs and higher system clock frequencies.

Table 3: Latency of the IEEE 802.16e CTC Decoder Core Using Two SISOs

Data Block Size (N)	fclk = 160 MHz		fclk = 190 MHz		fclk = 215 MHz	
	Latency ( $\mu s$ ) Ni = 4	Latency ( $\mu s$ ) Ni = 5	Latency ( $\mu s$ ) Ni = 4	Latency ( $\mu s$ ) Ni = 5	Latency ( $\mu s$ ) Ni = 4	Latency ( $\mu s$ ) Ni = 5
24	3.61	4.47	3.04	3.76	2.68	3.33
36	4.81	5.97	4.05	5.03	3.58	4.44
48	6.01	7.47	5.06	6.29	4.47	5.56
72	8.41	10.47	7.08	8.82	6.26	7.79
96	10.81	13.47	9.10	11.34	8.04	10.02
108	12.01	14.97	10.11	12.61	8.93	11.14
120	13.21	16.47	11.12	13.87	9.83	12.26
144	15.61	19.47	13.14	16.39	11.61	14.49
180	19.21	23.97	16.17	20.18	14.29	17.84
192	20.41	25.47	17.18	21.45	15.19	18.95
216	22.81	28.47	19.21	23.97	16.97	21.19
240	25.21	31.47	21.23	26.50	18.76	23.42
480	25.21	31.47	21.23	26.50	18.76	23.42
960	49.21	61.47	41.44	51.76	36.62	45.74
1440	73.21	91.47	61.65	77.03	54.48	68.07
1920	97.21	121.47	81.86	102.29	72.34	90.40
2400	121.21	151.47	102.07	127.55	90.20	112.72

Table 4: Latency of the IEEE 802.16e CTC Decoder Core Using Four SISOs

Data Block Size (N)	fclk = 160 MHz		fclk = 190 MHz		fclk = 215 MHz	
	Latency (μs) Ni = 4	Latency (μs) Ni = 5	Latency (μs) Ni = 4	Latency (μs) Ni = 5	Latency (μs) Ni = 4	Latency (μs) Ni = 5
24	3.61	4.47	3.04	3.76	2.68	3.33
36	4.81	5.97	4.05	5.03	3.58	4.44
48	6.01	7.47	5.06	6.29	4.47	5.56
72	8.41	10.47	7.08	8.82	6.26	7.79
96	10.81	13.47	9.10	11.34	8.04	10.02
108	12.01	14.97	10.11	12.61	8.93	11.14
120	13.21	16.47	11.12	13.87	9.83	12.26
144	15.61	19.47	13.14	16.39	11.61	14.49
180	19.21	23.97	16.17	20.18	14.29	17.84
192	20.41	25.47	17.18	21.45	15.19	18.95
216	22.81	28.47	19.21	23.97	16.97	21.19
240	25.21	31.47	21.23	26.50	18.76	23.42
480	13.21	16.47	11.12	13.87	9.83	12.26
960	25.21	31.47	21.23	26.50	18.76	23.42
1440	37.21	46.47	31.33	39.13	27.69	34.58
1920	49.21	61.47	41.44	51.76	36.62	45.74
2400	61.21	76.47	51.54	64.39	45.55	56.91

Table 5: Latency of the IEEE 802.16e CTC Decoder Core Using Five SISOs

Data Block Size (N)	fclk = 160 MHz		fclk = 190 MHz		fclk = 215 MHz	
	Latency (μs) Ni = 4	Latency (μs) Ni = 5	Latency (μs) Ni = 4	Latency (μs) Ni = 5	Latency (μs) Ni = 4	Latency (μs) Ni = 5
24	3.61	4.47	3.04	3.76	2.68	3.33
36	4.81	5.97	4.05	5.03	3.58	4.44
48	6.01	7.47	5.06	6.29	4.47	5.56
72	8.41	10.47	7.08	8.82	6.26	7.79
96	10.81	13.47	9.10	11.34	8.04	10.02
108	12.01	14.97	10.11	12.61	8.93	11.14
120	13.21	16.47	11.12	13.87	9.83	12.26
144	15.61	19.47	13.14	16.39	11.61	14.49
180	19.21	23.97	16.17	20.18	14.29	17.84
192	20.41	25.47	17.18	21.45	15.19	18.95
216	22.81	28.47	19.21	23.97	16.97	21.19

**Table 5: Latency of the IEEE 802.16e CTC Decoder Core Using Five SISOs (Cont'd)**

Data Block Size (N)	fclk = 160 MHz		fclk = 190 MHz		fclk = 215 MHz	
	Latency (μs) Ni = 4	Latency (μs) Ni = 5	Latency (μs) Ni = 4	Latency (μs) Ni = 5	Latency (μs) Ni = 4	Latency (μs) Ni = 5
240	25.21	31.47	21.23	26.50	18.76	23.42
480	10.81	13.47	9.10	11.34	8.04	10.02
960	20.41	25.47	17.18	21.45	15.19	18.95
1440	30.01	37.47	25.27	31.55	22.33	27.88
1920	39.61	49.47	33.35	41.66	29.47	36.81
2400	49.21	61.47	41.44	51.76	36.62	45.74

### Decoded Information Data Rate

The achievable information bit rate ( $R_b$ ) in Mbps of the IEEE 802.16e CTC decoder core is given by Equation 3.

$$R_b = N_s \left( \frac{2N'}{L'} \right) f_{clk}$$

Equation 3

where  $N_s$  is the number of SISOs,  $N'$  is the block size in pairs (N) divided by the number of SISOs when N is equal to 480, 960, 1440, 1920, or 2400 pairs; otherwise,  $N'$  is the block size in pairs (N),  $f_{clk}$  is the system clock frequency in MHz, and  $L'$  is latency of the decoder from the time the last sample of the first code block is written into the input buffer to the time the last sample of the first code block is written into the output buffer. Latency for the CTC decoder core is given by Equation 4.

$$L' = 2N_i[2N + C] + 12$$

Equation 4

where  $N_i$  is the number of iterations,  $N'$  is the data block size in pairs (N) divided by the number of SISOs ( $N_s$ ), when N is equal to 480, 960, 1440, 1920, or 2400 pairs; otherwise,  $N'$  is the block size in pairs (N), and C is a constant that is equal to 21 when early termination is disabled; otherwise C is equal to 25.

Table 6 shows the decoded information bit rate of the IEEE 802.16e CTC decoder core in Mbps for all block sizes versus number of full iterations and clock frequency when the number of SISOs is two. Table 7 shows the decoded information bit rate of the IEEE 802.16e CTC decoder core in Mbps for all block sizes versus number of full iterations and clock frequency when the number of SISOs is four. Table 8 shows the decoded information bit rate of the IEEE 802.16e CTC decoder core in Mbps for all block sizes versus number of full iterations and clock frequency when the number of SISOs is five. The numbers are based on Equation 3 and Equation 4. In general, for a fixed number of iterations, the decoded information bit rate is increased by using more SISOs and higher system clock frequencies.

**Table 6: Decoded Data Rate of the IEEE 802.16e CTC Decoder Core Using Two SISOs**

Data Block Size (N)	fclk = 160 MHz		fclk = 190 MHz		fclk = 215 MHz	
	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$
24	27.23	21.88	32.34	25.98	36.60	29.40
36	30.48	24.46	36.19	29.04	40.95	32.87

Table 6: Decoded Data Rate of the IEEE 802.16e CTC Decoder Core Using Two SISOs (Cont'd)

Data Block Size (N)	fclk = 160 MHz		fclk = 190 MHz		fclk = 215 MHz	
	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$
48	32.41	25.99	38.48	30.86	43.54	34.92
72	34.59	27.73	41.08	32.92	46.49	37.26
96	35.80	28.68	42.52	34.06	48.11	38.54
108	36.23	29.02	43.02	34.46	48.68	38.99
120	36.57	29.29	43.43	34.78	49.14	39.36
144	37.10	29.71	44.06	35.28	49.86	39.92
180	37.65	30.14	44.71	35.79	50.59	40.50
192	37.79	30.25	44.87	35.92	50.77	40.65
216	38.02	30.44	45.15	36.14	51.09	40.90
240	38.21	30.59	45.37	36.32	51.34	41.10
480	38.21	30.59	45.37	36.32	51.34	41.10
960	39.08	31.28	46.41	37.14	52.52	42.03
1440	39.38	31.51	46.77	37.42	52.92	42.35
1920	39.54	31.63	46.95	37.57	53.13	42.51
2400	39.63	31.71	47.06	37.65	53.25	42.61

Table 7: Decoded Data Rate of the IEEE 802.16e CTC Decoder Core Using Four SISOs

Data Block Size (N)	fclk = 160 MHz		fclk = 190 MHz		fclk = 215 MHz	
	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$
24	54.47	43.76	64.68	51.97	73.19	58.80
36	60.95	48.92	72.38	58.09	81.90	65.73
48	64.81	51.98	76.96	61.73	87.09	69.85
72	69.19	55.45	82.16	65.85	92.97	74.51
96	71.61	57.37	85.03	68.12	96.22	77.09
108	72.45	58.04	86.04	68.92	97.36	77.98
120	73.14	58.58	86.86	69.57	98.29	78.72
144	74.20	59.42	88.12	70.56	99.71	79.85
180	75.29	60.28	89.41	71.59	101.18	81.00
192	75.57	60.50	89.74	71.85	101.55	81.30
216	76.04	60.87	90.30	72.29	102.18	81.80
240	76.42	61.17	90.75	72.64	102.69	82.20
480	73.14	58.58	86.86	69.57	98.29	78.72

Table 7: Decoded Data Rate of the IEEE 802.16e CTC Decoder Core Using Four SISOs (Cont'd)

Data Block Size (N)	fclk = 160 MHz		fclk = 190 MHz		fclk = 215 MHz	
	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$
960	76.42	61.17	90.75	72.64	102.69	82.20
1440	77.58	62.09	92.12	73.73	104.24	83.43
1920	78.17	62.55	92.82	74.28	105.04	84.06
2400	78.53	62.84	93.25	74.62	105.52	84.44

Table 8: Decoded Data Rate of the IEEE 802.16e CTC Decoder Core Using Five SISOs

Data Block Size (N)	fclk = 160 MHz		fclk = 190 MHz		fclk = 215 MHz	
	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$	$R_b$ (Mbps) $N_i = 4$	$R_b$ (Mbps) $N_i = 5$
24	68.09	54.70	80.85	64.96	91.49	73.50
36	76.19	61.15	90.48	72.61	102.38	82.17
48	81.01	64.97	96.20	77.16	108.86	87.31
72	86.49	69.31	102.70	82.31	116.22	93.14
96	89.51	71.71	106.29	85.15	120.28	96.36
108	90.57	72.54	107.55	86.15	121.70	97.48
120	91.43	73.23	108.57	86.96	122.86	98.40
144	92.75	74.27	110.14	88.20	124.64	99.81
180	94.12	75.35	111.76	89.48	126.47	101.26
192	94.46	75.63	112.18	89.81	126.94	101.62
216	95.05	76.09	112.87	90.36	127.72	102.25
240	95.52	76.46	113.43	90.80	128.36	102.75
480	89.51	71.71	106.29	85.15	120.28	96.36
960	94.46	75.63	112.18	89.81	126.94	101.62
1440	96.24	77.03	114.29	91.47	129.32	103.51
1920	97.15	77.75	115.37	92.33	130.55	104.48
2400	97.71	78.19	116.03	92.85	131.30	105.07

Figure 8 shows the decoded information bit rate of the IEEE 802.16e CTC decoder core (in Mbps) with five iterations using two SISOs at 160 MHz clock frequency when early termination scheme 2 is enabled. The curves show the decoded information bit rate for different blocks measured in hardware using the test bench described in the design verification section. The decoded information bit rate is scaled linearly with the number of SISOs even when early termination is enabled.

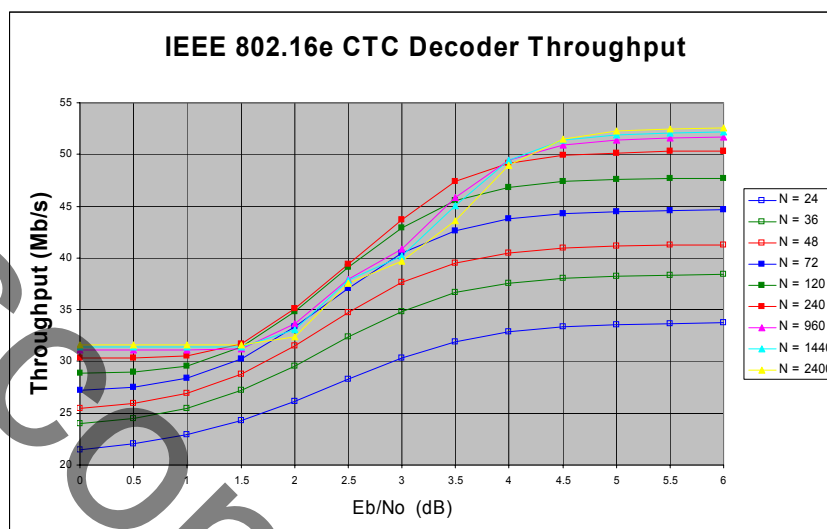


Figure 8: Decoded Information Data Rate with Five Iterations

## Resource Utilization and Static Timing

The following resource utilization estimate is based on 6-bits soft data input and 6-bits extrinsic information. The size of the CTC decoder core depends on the number of SISOs that are used ( $N_s$ ).

Table 9 and Table 10 show the resource utilization estimates of the IEEE 802.16e CTC decoder core that are given by the Xilinx ISE 8.1 software when targeting the XC4VLX60 device. Table 11 and Table 12 show the resource utilization estimates of the IEEE 802.16e CTC decoder core that are given by the Xilinx ISE 9.1 software when targeting the XC5VLX85 device. The resource utilization estimates of Virtex-6 and Spartan-6 are similar to that of Virtex-5.

Table 13 shows the static timing results of the IEEE 802.16e CTC decoder core using the Xilinx ISE 8.1 software (PRODUCTION 1.58 2006-02-24, STEPPING level 1). Table 14 shows the static timing results of the IEEE 802.16e CTC decoder core using the Xilinx ISE 9.1 software (ADVANCED 1.51 2006-12-12, STEPPING level 0). Table 15 shows the static timing results of the IEEE 802.16e CTC decoder core using the Xilinx ISE 11.1 software (PREVIEW 0.63 2009-04-27). Table 16 shows the static timing results using the Xilinx ISE 11.1 software (ADVANCED 0.94 2009-04-27).

**Note:** The resource utilization and static timing results can change slightly by using different speed constraints or targeting different devices.



**Table 9: Resource Utilization When the Largest Block Size is 2400 Pairs (600 Bytes) Targeting a Virtex-4 FPGA**

Number of SISOs	I/O	LUTs	FFs	Slices	Block RAMs	DSP48s
1	111	4,070	3,879	2,664	63	2
2	111	8,366	7,482	5,153	60	4
3	111	12,588	10,855	7,591	51	6
4	111	16,569	14,256	9,901	68	8
5	111	20,850	17,493	12,383	50	10
6	111	25,181	20,859	14,881	60	12
8	111	33,510	27,639	19,721	80	16

**Table 10: Resource Utilization When the Largest Block Size is 240 Pairs (60 Bytes) Targeting a Virtex-4 FPGA**

Number of SISOs	I/Os	LUTs	FFs	Slices	Block RAMs	DSP48s
1	107	3,828	3,628	2,477	8	2
2	107	7,520	6,752	4,626	16	4
3	107	11,260	9,871	6,819	24	6
4	107	14,941	12,987	8,959	32	8
5	107	18,638	16,105	11,139	40	10
6	107	22,339	19,223	13,301	48	12
8	107	29,717	25,462	17,628	64	16

**Table 11: Resource Utilization When the Largest Block Size is 2400 Pairs (600 Bytes) Targeting a Virtex-5 FPGA**

Number of SISOs	I/Os	LUTs	FFs	Slices	18k or 36k Block RAMs	DSP48s
1	111	3,505	3,881	1,237	63 or 32	2
2	111	7,211	7,479	2,429	60 or 30	4
3	111	10,766	10,851	3,803	51 or 26	6
4	111	14,300	14,245	4,478	68 or 34	8
5	111	17,742	17,495	5,420	50 or 25	10
6	111	21,558	20,842	6,078	60 or 30	12
8	111	28,543	27,623	8,575	80 or 40	16

**Table 12: Resource Utilization when the Largest Block Size is 240 Pairs (60 Bytes) Targeting a Virtex-5 FPGA**

Number of SISOs	I/Os	LUTs	FFs	Slices	18k or 36k Block RAMs	DSP48s
1	107	3,369	3,638	1,089	9 or 5	2
2	107	6,611	6,767	2,008	18 or 9	4
3	107	9,839	9,891	2,886	27 or 14	6
4	107	13,063	12,945	3,948	36 or 18	8
5	107	16,325	16,133	5,166	45 or 23	10
6	107	19,549	19,158	5,533	54 or 27	12
8	107	26,034	25,490	7,748	72 or 36	16

**Table 13: IEEE 802.16 CTC Decoder Core Static Timing Results Targeting a Virtex-4 FPGA**

Xilinx FPGA	Clock Speed (MHz)
XC4VLX60 -10	162.443
XC4VLX60 -11	191.132
XC4VLX60 -12	216.029

**Table 14: IEEE 802.16 CTC Decoder Core Static Timing Results Targeting a Virtex-5 FPGA**

Xilinx FPGA	Clock Speed (MHz)
XC5VLX85 -1	196.040
XC5VLX85 -2	225.327
XC5VLX85 -3	245.459

**Table 15: IEEE 802.16e CTC Decoder Core Static Timing results Targeting a Virtex-6 FPGA (PREVIEW 0.63 2009-04-27)**

Xilinx FPGA	Clock Speed (MHz)
XC6VLX75T-1	225.683
XC6VLX75T-2	255.102
XC6VLX75T-3	286.041

**Table 16: IEEE 802.16e CTC Decoder Core Static Timing results Targeting a Spartan-6 FPGA**

Xilinx FPGA	Clock Speed (MHz)
XC6SLX45T-2	111.161

## Design Verification

The decoder core was verified through VHDL simulation and hardware testing. Self-checking VHDL test benches were written for many of the design submodules. For example, the SISO design was verified by comparing the output of the VHDL SISO model to that of a bit-true MATLAB® model of the SISO with the same stimulus.

In addition to verifying the function of each subdesign, the top-level decoder design was also verified with a self-checking test bench. The bit-true MATLAB model was used to generate files of input stimulus and the corresponding output vectors. The input stimulus was read by the top-level test bench and applied to the decoder core. The resulting output of the decoder core was then compared to the MATLAB model output vectors read from a file.

Hardware testing of the decoder core was performed using the ML402 Prototyping Board populated with an XC4VSX35 -10 device. A hardware test bench was developed to test the BER performance of the decoder. In addition to the decoder core, the hardware test bench contains an LFSR-based data generator, the Xilinx IEEE 802.16e CTC Encoder core and the Xilinx AWGN core. It also contains other circuitry required to make BER measurements, such as an adder, a bit counter, and an error counter. A simplified block diagram of the hardware test bench is shown in Figure 9.

In addition to the previously mentioned circuitry, the hardware test bench also contains a block that quantizes the signal plus noise data into the 6-bit two's complement data expected by the decoder. The data signal is assumed to be normalized to  $\pm 1$ . The quantization circuit scales the signal plus noise data by 8 (3 fractional bits), rounds the scaled data to the nearest integer, then hard limits to  $\pm 31$  (6 total bits).



Figure 9: Block Diagram of Hardware Test Bench

## Speed and Power Consumption Measurements

Speed and power consumption measurements of the CTC decoder core were performed using the Virtex-5 FF676 FPGA AFX Prototyping Board populated with an XC5VLX50 -1 device. The same hardware test bench described in the design verification section was used to measure speed and power consumption of the CTC decoder core. Table 17 shows the dynamic power consumption versus number of full iterations when running the CTC decoder core at 49 Mbps using 196 MHz clock. Table 18 shows the dynamic power consumption versus number of full iterations when running the CTC decoder core at maximum decoding rate using 196 MHz clock. The dynamic power consumption is defined as the difference in power consumption between the case where the entire test bench including the CTC decoder core is enabled and the case where the test bench is enabled while the CTC decoder core is disabled.

Table 17: IEEE 802.16e CTC Decoder Core Power Consumption at Fixed Decoding Rate

Number of Iterations	Decoded Data Rate	Dynamic Power Consumption (mW)
1	49	178
2	49	356

Table 17: IEEE 802.16e CTC Decoder Core Power Consumption at Fixed Decoding Rate

Number of Iterations	Decoded Data Rate	Dynamic Power Consumption (mW)
3	49	536
4	49	717
5	49	897

Table 18: IEEE 802.16e CTC Decoder Core Power Consumption at Maximum Decoding Rate

Number of Iterations	Maximum Decoded Data Rate	Dynamic Power Consumption (mW)
4	93.61	1,350
5	74.93	1,351
6	62.47	1,350

## BER Performance

All BER results were obtained using the hardware test bench described in "Design Verification." At the lower SNRs, the number of bit errors counted was generally over 4,000. At the higher SNRs, the minimum number of bit errors was over 1,000. The BER results were based on 6-bits input data (3 fractional bits and 3 integer bits), 6-bits extrinsic, and 10-bits accumulated state metric. Figure 10 shows the BER performance of the IEEE 802.16e decoder core using rate 1/2 code and block size 240 pairs versus the number of iterations. Figure 11 shows the BER performance of the IEEE 802.16e decoder core using rate 1/3 code and block size 240 pairs versus the number of iterations.

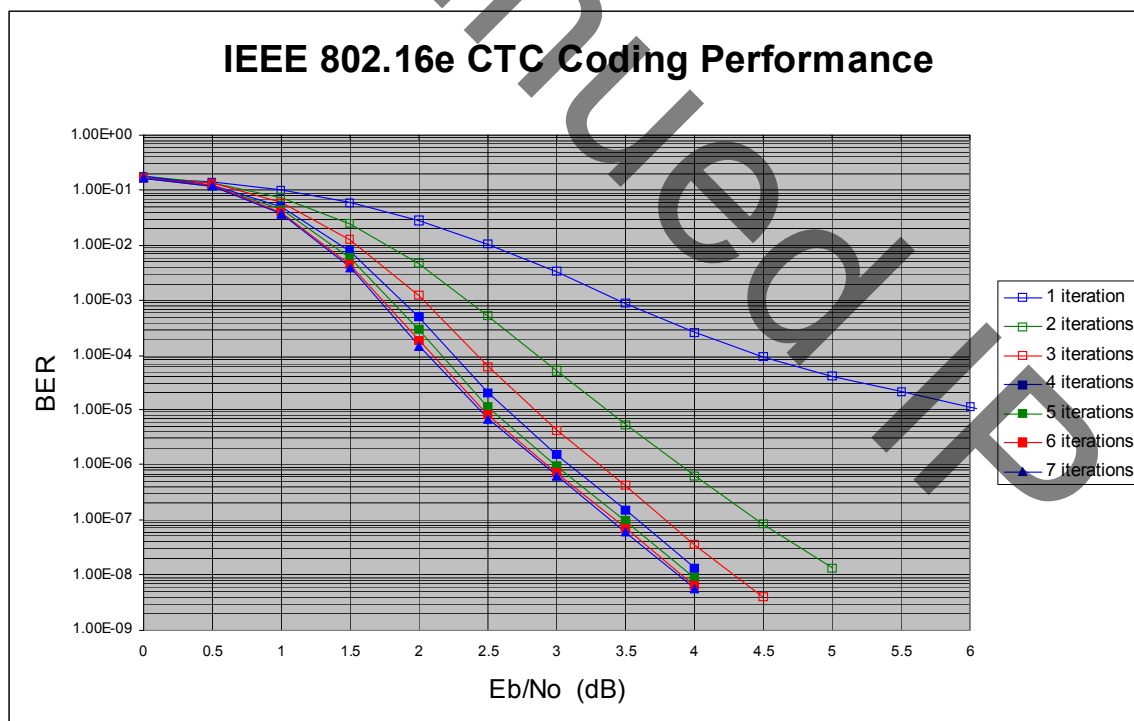


Figure 10: BER Performance of Rate 1/2 CTC Code with Block Size N = 240

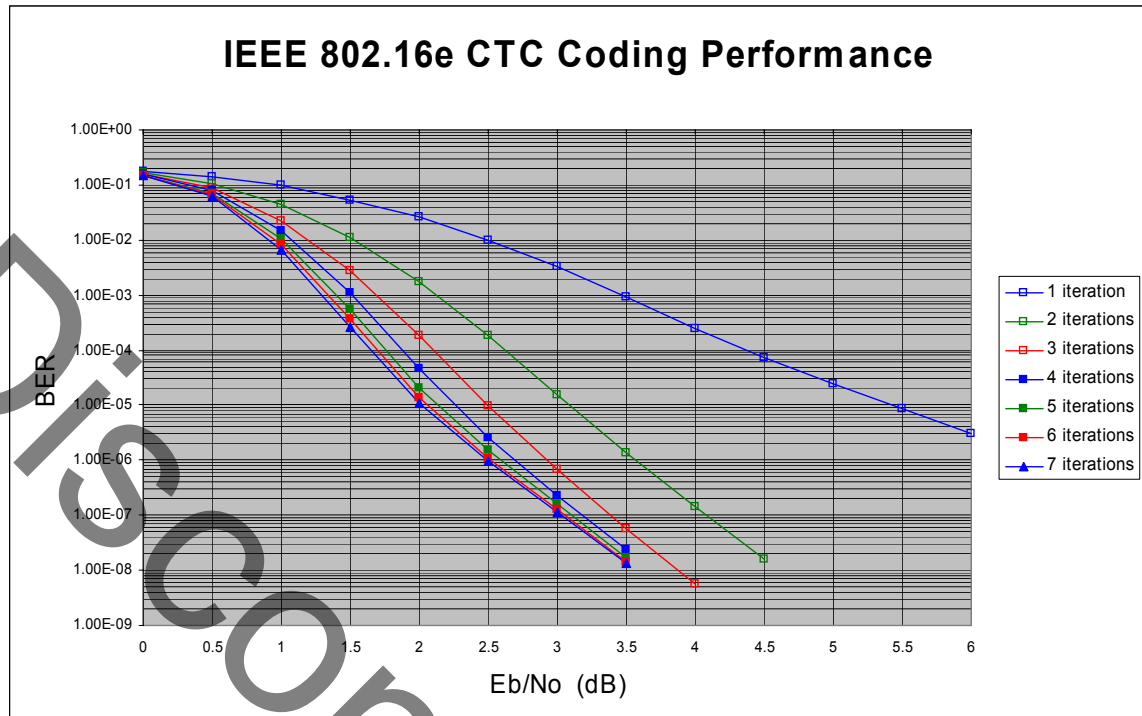


Figure 11: BER Performance of Rate 1/3 CTC Code with Block Size N = 240

## Support

Xilinx provides technical support at [www.xilinx.com/support](http://www.xilinx.com/support) for this LogiCORE 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

The fixed netlist version of the core is provided under the LogiCORE [IP Site License Agreement](#). A free evaluation version is available from Xilinx DSP marketing or through your Xilinx [sales representative](#).

For part number and pricing information, see the core product page on the Xilinx [IP Center](#). To purchase this core, contact your local sales representative.

Information on additional Xilinx LogiCORE IP modules is available on the Xilinx [IP Center](#).

## Revision History

Date	Version	Revision
10/10/07	1.0	Initial Xilinx release.
09/19/08	1.1	Updated for core version 3.1.
06/24/09	1.2	Updated for core version 4.0.
12/02/09	1.3	ISE version numbers revised.

## Notice of Disclaimer

Xilinx is providing this product documentation, hereinafter "Information," to you "AS IS" with no warranty of any kind, express or implied. Xilinx makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx.