802.16e CTC Encoder v3.0

Features

- Drop-in module for Spartan®-6, Spartan-3E, Spartan-3A/3AN/3A DSP, Spartan-3, Virtex®-6, Virtex-5 and Virtex-4 FPGAs
- Implements the IEEE 802.16e Wireless MAN OFDMA PHY CTC encoder specification [1]
- Supports optional 64-QAM modulation
- Supports optional hybrid-ARQ (HARQ)
- Multiplexed symbol memory for maximum throughput
- Simultaneous C_1 and C_2 encoding
- Pipelined data paths for speed
- Flexible interfacing by means of optional control signals
- Designed for minimum area and maximum speed
- Available using the CORE Generator™ software v11.1, which is included with the ISE™ v11.1 tools

Applications

The Convolutional Turbo Code (CTC) encoder meets the Wireless MAN OFDMA PHY CTC encoder specification of the IEEE 802.16e standard [1] and supports the optional hybrid ARQ (HARQ).

General Description

The theory of operation of the Turbo Codes is described in the paper by Berrou, Glavieux, and Thitimajshima [2].

The CTC Encoder core is a parallelized implementation of the convolutional turbo coder specified by the IEEE 802.16e Wireless MAN OFDMA PHY CTC encoder specification [1].

The input and output ports of the CTC encoder core are shown in Figure 1.

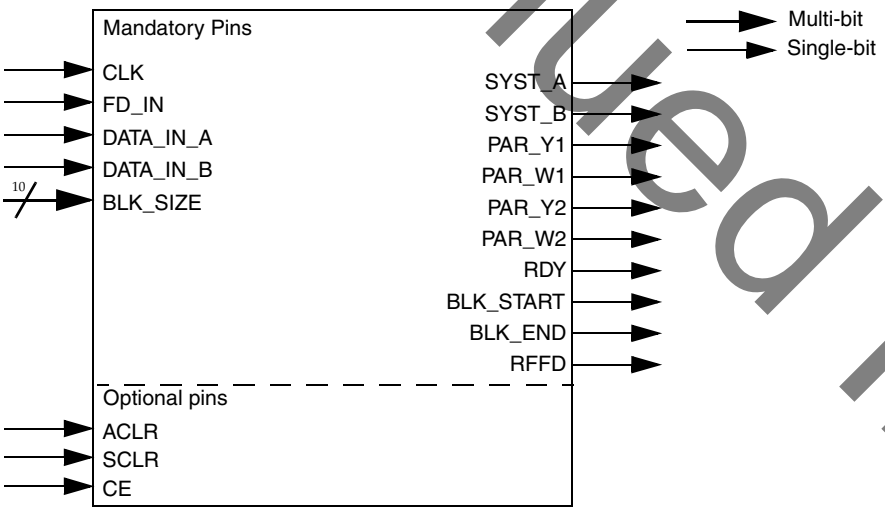

Figure 1: **Pinout**

Two constituent encoders perform C1 (uninterleaved) and C2 (interleaved) encoding. Two other convolutional encoders are employed as precoders that calculate the circulation states for the two main encoders. The constituent encoders are described later in this data sheet.

A triple-buffered rotating memory bank permits C1 and C2 encoding, and the calculation of the circulation states all to be performed simultaneously for maximum data throughput.

The encoder block size can be changed on a block-by-block basis, using the BLK_SIZE input port. All block sizes specified by the 802.16e standard, ranging from 6 to 600 bytes are supported.

Flexible handshaking is facilitated by the FD_IN input port and the output ports RDY, BLK_START, BLK_END, and RFFD.

The fundamental specification of the encoder, including block sizes, determination of the circulation states, and the interleaver specification can be found in the IEEE 802.16e standard[1].

Input/Output Ports

The I/O ports for the basic configuration are summarized in Table 1 and described below.

Table 1: **I/O Ports, Basic Configuration**

Pin	Sense	Port Width (bits)	Description
FD_IN	Input	1	**First Data**. When sampled High on an active rising clock edge, the encoding process is started.
ACLR	Input (optional)	1	**Asynchronous Clear**. When this is asserted (High), the encoder asynchronously resets.
SCLR	Input (optional)	1	**Synchronous Clear.** When sampled High on an active rising clock edge, the encoder is reset.
CE	Input (optional)	1	**Clock Enable**. When 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.
BLK_SIZE	Input	10	**Block Size.** Sets up the block size (in bytes) when **First Data** (FD_IN) is sampled high. Ignored if FD_IN is Low.
DATA_IN_A	Input	1	**Data Input A**. The A-channel of the data to be encoded.
DATA_IN_B	Input	1	**Data Input B**. The B-channel of the data to be encoded.
SYST_A	Output	1	**RSC1 Systematic A**.The systematic A output from RSC1.
SYST_B	Output	1	**RSC1 Systematic B**. The systematic B output from RSC1.
PAR_Y1	Output	1	**RSC1 Parity Y.** The Y parity output from RSC1.
PAR_W1	Output	1	**RSC1 Parity W.** The W parity output from RSC1.
PAR_Y2	Output	1	**RSC2 Parity Y.** The Y parity output from RSC2.
PAR_W2	Output	1	**RSC2 Parity W.** The W parity output from RSC2.
RFFD	Output	1	**Ready for First Data**. When the output asserted (High), the core is ready to start another encoder operation.

Table 1: **I/O Ports, Basic Configuration *(Continued)***

Pin	Sense	Port Width (bits)	Description
RDY	Output	1	**Ready**. The output is asserted (High) when there is valid data on the systematic and parity outputs.
BLK_START	Output	1	**Block Start**. The output is asserted (High) for one clock cycle to signal the start of a valid block on the systematic and parity outputs.
BLK_END	Output	1	**Block Start**. The output is asserted (High) for one clock cycle to signal the end of a valid block on the systematic and parity outputs.

Asynchronous Clear (ACLR)

The ACLR input port is optional. When ACLR is driven High, the core resets to its initial state and is ready to process a new block. Following the initial configuration of the FPGA, the core is automatically in the reset state, so no further ACLR is required before an encoding operation can take place. ACLR is the only asynchronous input to the core.

Clock (CLK)

With the exception of asynchronous clear, all operations of the core are synchronized to the rising edge of CLK. If the optional CE pin is enabled, an active rising clock edge occurs only when CE is High.

Clock Enable (CE)

Clock enable is an optional input pin that is used to enable the synchronous operation of the core. When CE is High, a rising edge of CLK is acted upon by the core, but if CE is Low, the core remains in its current state. An active, rising clock edge is one on which CE (if enabled) is sampled High.

Synchronous Clear (SCLR)

The SCLR signal is optional. When SCLR is sampled High on an active rising clock edge, the core is reset to its initial state and is ready to process a new block. Following the initial configuration of the FPGA, the core is automatically in the reset state; no further SCLR is required before an encoding operation can take place. If the CE input port is selected, the SCLR is ignored if CE is Low.

Data Input Ports (DATA_IN_A / DATA_IN_B)

The DATA_IN ports are mandatory input ports that carry the unencoded double-binary data couples. The input process is started with a First Data (FD_IN) signal. When FD_IN is sampled High, the value on the DATA_IN_A port is the MSB of the first byte of unencoded data. Data couples are read serially into the DATA_IN port on a clock-by-clock basis. BLK_SIZE*4 clock cycles are therefore required to input each block.

First Data (FD_IN)

FD_IN is a mandatory input port, which is used to start the encoder operation. When FD_IN is sampled High on an active rising clock edge, the first data couple is read from the DATA_IN_A and DATA_IN_B ports, and the block size is read from the BLK_SIZE port. The core then continues loading data until a complete block has been input.

The FD_IN input should only be asserted when the RFFD output is High (see below). If FD_IN is sampled High when RFFD is Low, this is an *abort* condition, and the behavior of the core is not specified.

Ready for First Data (RFFD)

A logic High on RFFD indicates that the core is ready to accept an FD_IN signal to start a new encoding operation. When an FD_IN signal is sampled High, the RFFD signal goes Low and remains Low until it is safe to start another block.

Block Size (BLK_SIZE)

This 10-bit input port determines the size, in bytes, of the block of data that is about to be written into the encoder. The block size value is sampled on an active rising clock edge when FD_IN is High. If an invalid block size is sampled, the behavior of the core is not specified.

Systematic Data Output (SYST_A / SYST_B)

SYST_A and SYST_B are mandatory output ports, which are delayed versions of the uninterleaved double-binary input data, DATA_IN_A and DATA_IN_B, respectively.

RSC1 Parity Y Output (PAR_Y1)

PAR_Y1 is a mandatory output port, which is the Y output of the Constituent Encoder RSC1.

RSC1 Parity W Output (PAR_W1)

PAR_W1 is a mandatory output port, which is the W output of the Constituent Encoder RSC1.

RSC2 Parity Y Output (PAR_Y2)

PAR_Y2 is a mandatory output port, which is the Y output of the Constituent Encoder RSC2.

RSC2 Parity W Output (PAR_W2)

PAR_W2 is a mandatory output port, which is the W output of the Constituent Encoder RSC2.

Ready (RDY)

RDY is a mandatory output port, which is driven High when there is valid data on the SYSTEMATIC and PARITY output ports

Block Start (BLK_START)

BLK_START is a mandatory output port, which is driven High for one clock period corresponding to the first valid output sample of a given block on the SYSTEMATIC and PARITY output ports

Block End (BLK_END)

BLK_END is a mandatory output port, which is driven High for one clock period corresponding to the last valid output sample of a given block on the SYSTEMATIC and PARITY output ports.

Functional Description

Constituent Encoder

The schematic of the constituent encoder and the polynomials for the first register input X and parity outputs Y and W are shown in Figure 2.

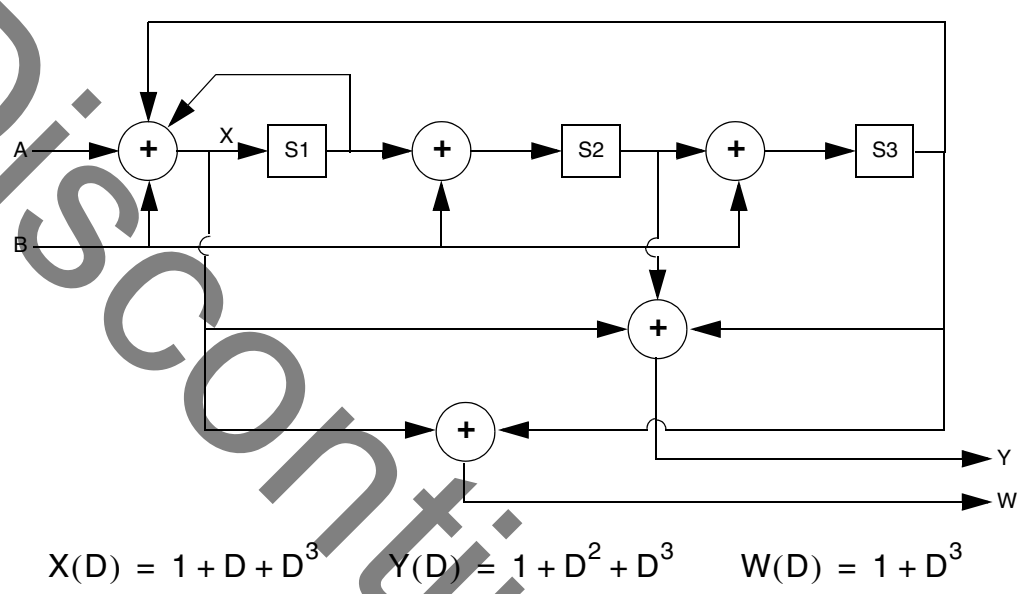

Figure 2: **CRSC Constituent Encoder**

The constituent encoder produces a double-binary Circular Recursive Systematic Convolutional (CRSC) code. This code is more efficient than a basic Recursive Systematic Convolutional (RSC) code, because it does not require three tail bit periods to flush the encoder at the end of each block. Instead, the CRSC is pre-loaded with a *circulation state* at the start of each block.

The CRSC is double-binary. It handles two data bits at a time, and the parity bits are functions of both input bits.

Calculation of the circulation state for the constituent encoder is performed as follows.

1. Initialize the encoder to state 0. Encode the input sequence (in either interleaved or uninterleaved order as required), and obtain the final state of the encoder.
2. Apply a look up table to obtain the circulation state.

In the CTC encoder core, the above operations are performed in parallel by the two *precoders*.

Internal Architecture

The internal architecture of the core is shown in Figure 3.

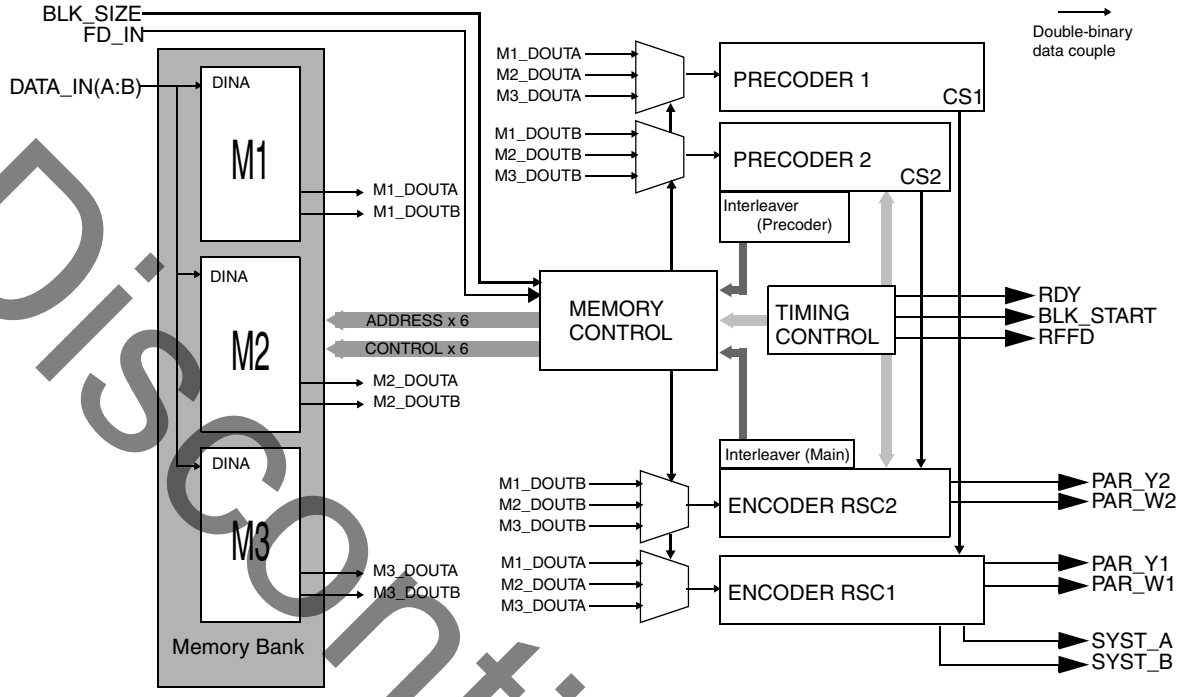

Figure 3: **CTC Encoder Structure**

The sequence of input data couples is written into one of three dual-port memory buffers, while data from the other two dual-port memories are streamed out to the precoders and the encoders. The three dual-port memories are rotated every block.

Data couples read out of the DOUTA ports of the memory bank are in uninterleaved (natural) order, whereas those read from the DOUTB ports are in interleaved order.

Triple-buffering for Maximum Throughput

As described previously, the encoder employs six parallel outputs, and a triple-buffered memory architecture for maximum throughput. Figure 4 illustrates the utilization of the three memories, M1, M2, and M3, and the effects on throughput of changing the block size.

The throughput is at a maximum if the block size is constant, as in Figure 4(a). In this case, the output data is continuous, with the exception that due to internal delays there is a 4-cycle gap every third block. This is signalled on the output side by a Low on RDY and on the input side by a delay in RFFD of five cycles once every third block.

Figure 4(b) shows the effect of increasing the block size. Since it takes longer to write a large block into memory than it does to read a smaller one out, the memory controller needs to wait for the larger block to be written before it can rotate the buffers. This causes a gap in the output data, which is signalled by a Low on RDY.

Figure 4(c) shows the effect of decreasing the block size. Since it takes longer to read a large block out of memory than it does to write smaller one in, the memory controller needs to wait for the larger block to be read out before it can rotate the buffers. To avoid overrun on the input, RFFD is held Low until the memory read operation has completed.

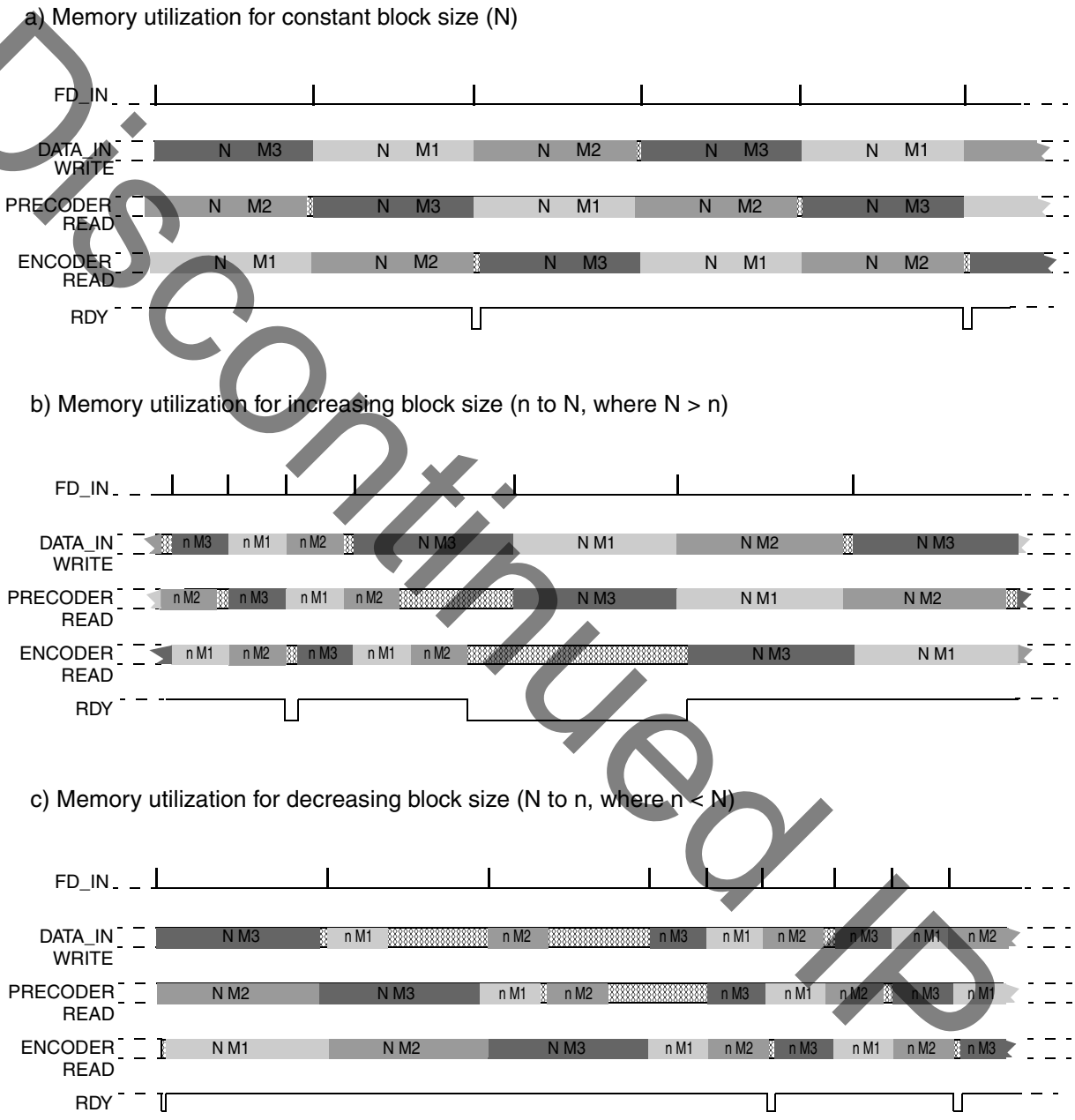

Figure 4: **Triple-Buffering Memory Utilization**

Input Control Signals

Figure 5 shows the signals associated with the data input side of the core.

When FD_IN is sampled High on an active rising edge of CLK, the RFFD signal is driven Low to indicate that the core is no longer waiting for FD_IN. The block size (in bytes), in this case *n/4*, of the current input block is sampled on the BLK_SIZE port, and the first data couple, A0 and B0, is sampled on the DATA_IN ports.

The core continues to input data until *n* new data couples have been accepted, and then the core stops sampling the DATA_IN ports.

When the core is ready to accept a new block of data, RFFD is driven High.

After asserting RFFD, the core waits until the next FD_IN pulse, and then a new write cycle is started, in this case with block size *m/4*.

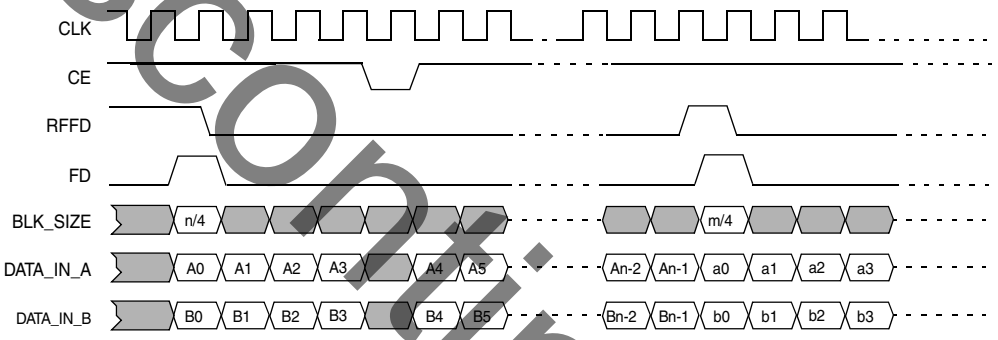

Figure 5: **Input Timing**

Output Control Signals

The RDY signal is driven High to indicate that there is valid data on the systematic and parity ports. The BLK_START signal is driven High for one clock cycle at the start of an output block as shown in Figure 6.

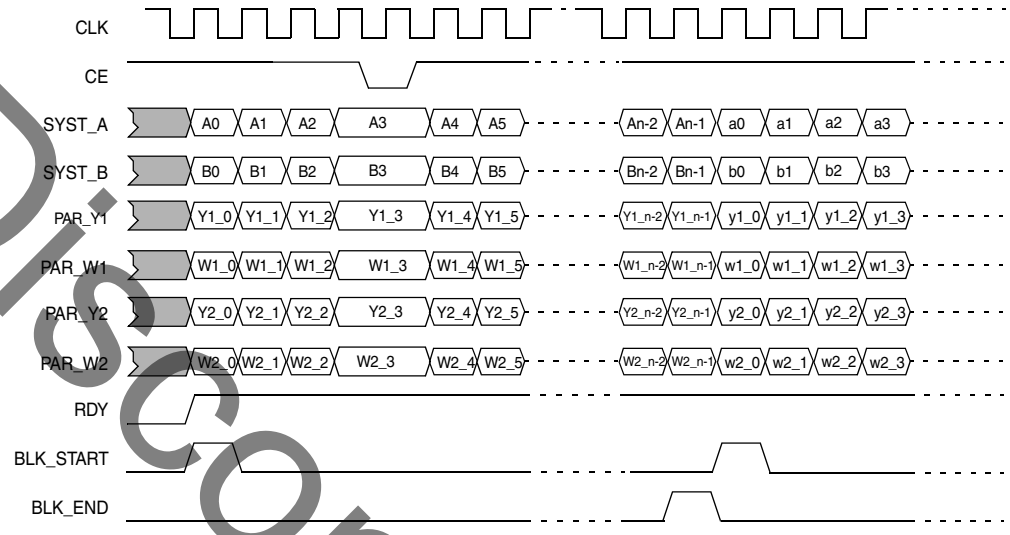

Figure 6: **Output Timing**

Latency

The latency is defined as the number of cycles from when the first input data couple is sampled until the first encoded output. In other words, the time from the FD_IN input being sampled until the BLK_START output being asserted for that block.

Due to the triple-buffered architecture, all data must be written to RAM, read out first to the precoder, and then again to the main encoder. In the case where the block size is invariant, the latency is therefore a few cycles more than twice the block size in couples.

In general, when the block size is not constant, there may be an additional component to the latency, such as a larger block already being processed. The worst case first-in-to-first-out latency for any given block is therefore just over twice the largest block size in couples. This can be understood by referring to the previous section on triple-buffering. Some latency values for fixed block sizes are given in table Table 2.

Table 2: **Latency Values for Fixed Block Sizes**

Blk_size	Block Size N (Couples)	Latency First Input to First Output (Typical)(Cycles)
6	24	58
9	36	82
12	48	106
18	72	154
24	96	202

Table 2: **Latency Values for Fixed Block Sizes *(Continued)***

Blk_size	Block Size N (Couples)	Latency First Input to First Output (Typical)(Cycles)
27	108	226
30	120	250
36	144	298
45	180	370
48	192	394
54	216	442
60	240	490
120	480	970
240	960	1930
360	1440	2890
480	1920	3850
600	2400	4810

Notes:

1. For a constant block size, First-in-last-out latency= 2N+d cycles, where N is the block size in couples, and d is a fixed delay.
2. On block size changes from block size N_1 to N_2, the First-in-last-out latency is given by:
   $2\max(N_1,N_2)+d$ for the first block of size N_2,
   $\max(N_1,N_2) + N_2 + d$ for the second block,
   $2N_2+d$ thereafter.
   Where: N_1 and N_2 are block sizes in couples.
3. First-in-last-out latency is (First-in-first-out latency)+N_2-1 cycles,
4. The fixed delay d is typically 10 cycles.

Performance and Resource Usage

Some typical performance and resource usage figures for minimal and maximal implementations on Virtex-5 parts are given in Table 3.

Table 3: **Performance and Resource Usage**

With CE, SCLR, and ACLR	Note	Virtex-5 XC5VSX95T-1
Area (LUT6-FF pairs)	1	741
Area (Registers)	1	609
36K Block Memories	1	2
18K Block Memories	1	3
MULT18x18s or DSP48s	1	0
Speed (MHz)	1,2,3	303
Without CE, SCLR, or ACLR	**Note**	**Virtex-5 XC5VSX95T-1**
Area (LUT6-FF pairs)	1	666
Area (Registers)	1	608
36K Block Memories	1	2
18K Block Memories	1	3
MULT18x18s or DSP48s	1	0
Speed (MHz)	1,2,3	303

Notes:

1. Area and maximum clock frequencies are provided as a guide. They may vary with new releases of Xilinx implementation tools, etc.
2. Obtained with the core instantiated within a double registered wrapper, and with the outer wrapper registers in the IOBs. Area figures do not include the inner wrapper registers.
3. Clock frequency does not take clock jitter into account and should be derated by an amount appropriate to the clock source jitter specification.

Throughput

The throughput is greatest when larger block sizes are used. Table 4 shows the throughput values obtained for some typical block sizes.

Table 4: **Throughput Values**

Block Size (Couples)	Input Data Throughput (% of Clock Frequency)
24	94.7
216	99.4
480	99.7
2400	99.9

References

1. *IEEE Std 802.16e-2004/Cor1/D5. IEEE Standard for Local and Metropolitan Area Networks*, Part 16: Air Interface for Fixed Broadband Wireless Access Systems (Draft Corrigendum)
2. C. Berrou, A. Glavieux, and P. Thitimajshima, *Near Shannon Limit Error-correcting Coding and Decoding Turbo Codes*, IEEE Proc 1993 Int Conf. Comm., pp. 1064-1070.

Ordering Information

This Xilinx LogiCORE™ IP 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 core, you will receive instructions for registering and generating a full license. The full license can be requested and installed from the Xilinx IP Center for use with the Xilinx CORE Generator. The CORE Generator software is bundled with all Alliance series software package at no additional charge. Contact your local Xilinx sales representative for pricing and availability of Xilinx LogiCORE products and software.

France Telecom, for itself and certain other parties, claims certain intellectual property rights covering Turbo Codes technology, and is licensing 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.

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.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/18/06	1.1	Initial Xilinx release.
09/28/06	2.0	Updated to v2.0 of Encoder and Xilinx ISE tools v8.2i.
02/15/07	2.1	Updated for v2.1 of Encoder and Xilinx ISE tools v9.1i.
04/02/07	2.5	Added support for Spartan-3A DSP devices.
04/24/09	3.0	Removed support for legacy Virtex devices, added support for Virtex-6 and Spartan-6 FPGAs, updated for v3.0 of the Encoder and Xilinx ISE v11.1 tools.