LogiCORE IP 3GPP LTE Turbo Encoder v3.1

DS701 June 22, 2011 **Product Specification**

Introduction

The Turbo Convolution Code (TCC) encoder is designed to meet the 3GPP LTE mobile communication system specification [Ref 1].

Features

- Implements the turbo encoder as defined in *3GPP TS 36.212 v9.0.0 Multiplexing and Channel Coding* specification. [Ref 1]
- Core contains the full 3GPP LTE interleaver
- All 188 3GPP LTE block sizes (40 - 6144) supported
- FIFO-buffered symbol memory for maximum throughput
- Flexible interfacing using optional control signals
- For use with the Xilinx® Core Generator™ software v13.2
- Bit-accurate C model available.

LogiCORE IP Facts Table		
Core Specifics		
Supported Device Family(1)	Virtex-7, Kintex™-7, Artix™-7, Zynq™-7000, Virtex-6, Virtex-5, Spartan-6	
Supported User Interfaces	N/A	
	Resources	**Frequency**
Configuration	See Table 2 to Table 4	
Provided with Core		
Documentation	Product Specification C Model User Guide (Contact Xilinx sales representative)	
Design Files	Netlist	
Example Design	Not Provided	
Test Bench	Not Provided	
Constraints File	Not Provided	
Simulation Model	VHDL Verilog C model	
Tested Design Tools		
Design Entry Tools	CORE Generator 13.2	
Simulation(2)	Mentor Graphics ModelSim	
Synthesis Tools	N/A	
Support		
Provided by Xilinx, Inc.		

1. For a complete listing of supported devices, see the release notes for this core.
2. For the supported version of the tools, see the ISE Design Suite 13: Release Notes Guide

Applications

This version of the Turbo Convolution Code (TCC) encoder is designed to meet the 3GPP LTE mobile communication system specification [Ref 1].

General Description

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

The 3GPP LTE Turbo Encoder input and output ports are shown in Figure 1, and the internal architecture is shown in Figure 2. It is a block-based processing unit where blocks between 40 bits and 6144 bits are input via the DATA_IN port and processed. Each block of data is processed in two identical Recursive Systematic Convolutional (RSC) encoders, which generate high-weight codes. RSC1 processes the raw input data, while RSC2 processes an interleaved version of the input data. The coding operates on the principle that if an input symbol is corrupted in the sequence from RSC1, then it is unlikely also to be corrupted in the interleaved sequence from RSC2, and vice versa.

Often some of the encoded output bits need not be transmitted, so they are omitted, or *punctured* from the output stream. Puncturing offers a dynamic trade-off between code rate and error performance. When the channel is noisy or the data requires more protection, extra redundancy can be added, thereby lowering the code rate. Puncturing is not implemented as part of the core.

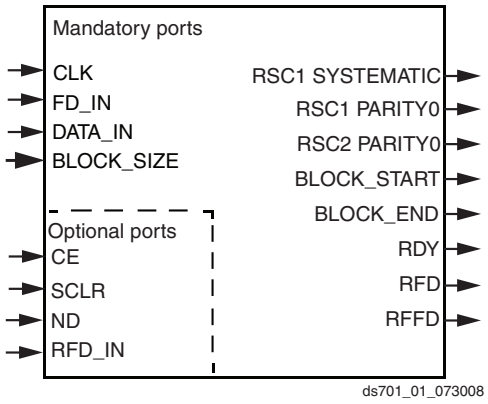

Figure 1: **TCC Encoder Pinout**

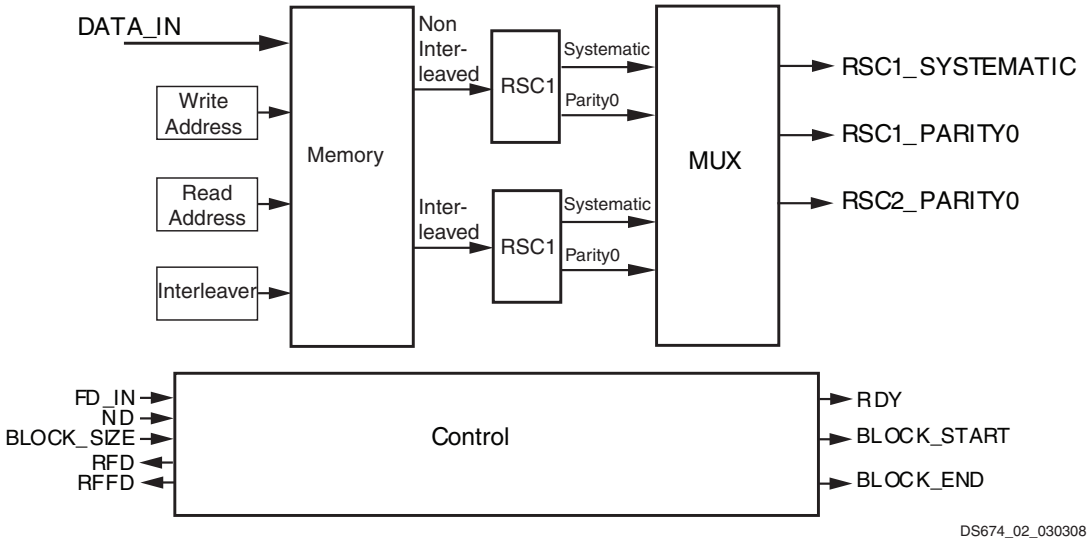

Figure 2: **TCC Encoder Structure**

RSC Encoder Structure

The schematic for each of the two RSCs and the transfer functions for the outputs are shown in Figure 3.

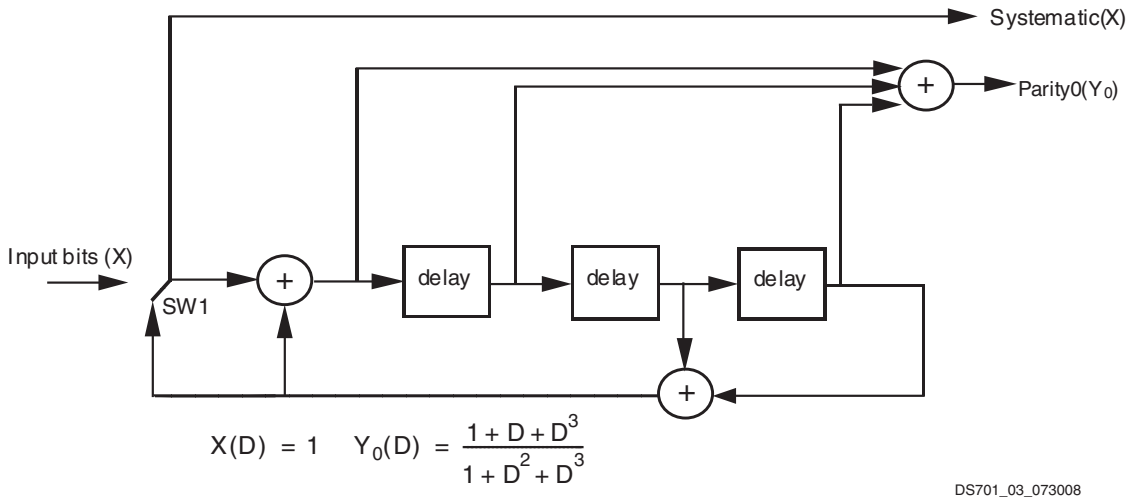

Figure 3: **TCC RSC Structure**

After a block of input data has been coded, the RSCs must return to the initial zero state. To force the RSC back to the all zero state, the input value is set equal to the feedback value by setting the control switch, SW1, to the lower position for three clock cycles. During the first three tail bit periods, RSC2 is disabled, and the control switch of RSC1 is set to the lower position to output the RSC1 tail bits on the RSC1 systematic and parity outputs. During the last three tail bit periods, RSC1 is disabled and the control switch of RSC2 is set to the lower position to output the RSC2 tail bits on the RSC2 systematic and parity outputs.

In practice, the 12 tail bit values from `RSC1_systematic`, `RSC2_systematic`, `RSC1_Parity0`, and `RSC2_Parity0` are re-multiplexed onto `RSC1_systematic`, `RSC1_Parity0`, and `RSC2_Party0` outputs over four clock cycles. See Trellis Termination.

Input/Output Ports

The I/O ports are summarized and described in Table 1.

Table 1: **I/O Ports**

Pin	Sense	Port Width (bits)	Description
CLK	Input	1	**Clock** – All synchronous operations occur on the rising edge of the clock signal.
CE	Input (optional)	1	**Clock Enable** – When deasserted (low), rising clock edges are ignored and the core is held in its current state.
SCLR	Input (optional)	1	**Synchronous Clear** – When asserted (high) on an active clock edge, the encoder is reset.
DATA_IN	Input	1	**Data Input** – The data to be encoded.
BLOCK_SIZE	Input	13	**Block Size** – The block size of the current encode operation.
ND	Input (optional)	1	**New Data** – When ND is sampled high on a valid clock edge, a new input value is read from the DATA_IN port.
FD_IN	Input	1	**First Data** – FD_IN is asserted (high) on a valid clock edge to indicate that the first bit of a block is on the DATA_IN port and the BLOCK_SIZE port is sampled. FD_IN is qualified by ND.
RFD_IN	Input (optional)	1	**Ready For Data (Input)** – A logic high on RFD_IN indicates that the downstream system is able to accept data from the core. The output side of the core is inhibited if RFD_IN is low.
RSC1_SYSTEMATIC	Output	1	**RSC1_systematic** – The systematic output from RSC1.
RSC1_PARITY0	Output	1	**RSC1_parity0** – The parity0 output from RSC1.
RSC2_PARITY0	Output	1	**RSC2_parity0** – The parity0 output from RSC2.
RFD	Output	1	**Ready For Data** – When asserted (high), the core is ready to accept a new input bit on the DATA_IN port.
RFFD	Output	1	**Ready For First Data** – When asserted (high), the core is ready to accept a new input block. RFFD can be high only if RFD is high.
BLOCK_START	Output	1	**Block Start** – Indicates that the first data bit of an output block is on the systematic and parity outputs when asserted (high).
BLOCK_END	Output	1	**Block End** – Indicates that the last tail bit of an output block is on the systematic and parity outputs when asserted (high).
RDY	Output	1	**Ready** – Indicates that there is valid data on the systematic and parity outputs when asserted (high).

Clock (CLK)

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. If CE is low, the core is held in its current state.

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. It should be noted that CE has a high fanout internal to the core, and this can result in a reduction in performance of about 10%.

Synchronous Clear (SCLR)

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

Data In (DATA_IN)

The DATA_IN port is a mandatory input port that carries the unencoded data. The input process is started with a valid-FD signal and data is read serially into the DATA_IN port on a clock-by-clock basis. Block size clock cycles are, therefore, required to input each block. DATA_IN can be qualified by the optional ND port. See New Data (ND).

New Data (ND)

The optional ND input port is used to indicate that there is new input data to be read from the DATA_IN port. For example, if the input block size is 40, then 40 active high *ND-samples* are required to load a block of data into the encoder. ND should be asserted only when RFD is high. ND is also used to qualify the FD_IN input. See First Data (FD_IN).

First Data (FD_IN)

FD_IN is a mandatory input port used to start the encoder operation. FD_IN is qualified by the optional ND input. Furthermore, the core is sensitive to FD_IN only if the RFFD output is high. A *valid-FD* means that FD_IN and ND are both sampled high on an active rising clock edge when the RFFD output is high.

When a *valid-FD* occurs, the first data bit of a block is read from the DATA_IN port, and the block size is read from the BLOCK_SIZE port. The core then continues loading data, until a complete block has been input.

The FD_IN input should be asserted only when the RFFD output is high. See Ready For First Data (RFFD).

Block Size (BLOCK_SIZE)

This port determines the size 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 and ND (if selected) is high. If an invalid block size (that is, not one of the 188 block sizes specified in standard [Ref 1]) is sampled, the behavior of the core is not specified.

Ready For Data (RFD)

RFD is a mandatory output port that is asserted high when the core is ready to accept new data. If RFD is selected, then it is high during the period that a particular block is input. When *block size* samples of data have been input, the RFD signal goes low to indicate that the core is no longer ready to accept data. The DATA_IN, BLOCK_SIZE, ND, and FD ports are sampled only if RFD is high.

Ready For First Data (RFFD)

RFFD is a mandatory output port that is asserted high when the core is ready to accept an FD_IN signal to start a new encoding operation. RFFD can be high only when RFD is also high. When a valid-FD signal is sampled, the RFFD output goes low and remains low until it is ready to accept another input block. The FD_IN and BLOCK_SIZE ports are sampled only if RFFD is high.

RSC1 Systematic Output (RSC1_SYSTEMATIC)

RSC1_SYSTEMATIC is a mandatory output port that is a delayed version of the uninterleaved input data. During trellis termination, the RSC1_SYSTEMATIC port also carries systematic and parity tail bits.

RSC1 Parity0 Output (RSC1_PARITY0)

RSC1_PARITY0 is a mandatory output port that is the Y0 output from RSC1 (see Figure 2). During trellis termination, the RSC1_PARITY0 port also carries systematic and parity tail bits.

RSC2 Parity0 Output (RSC2_PARITY0)

RSC2_PARITY0 is a mandatory output port that is the Y0 output from RSC2 (see Figure 2). During trellis termination, the RSC2_PARITY0 port also carries systematic and parity tail bits.

Block Start (BLOCK_START)

This signal is asserted high for one clock cycle when the first valid data bit of a block is on the systematic and parity output ports.

Block End (BLOCK_END)

This signal is asserted high for one clock cycle when the last tail bit of a block is on the systematic and parity output ports.

Ready (RDY)

This signal is asserted high when there is valid data on the systematic and parity output ports, including the tail bits. RDY is asserted for block size plus four clock cycles for every encoded block. If the core is running at maximum throughput, RDY does not go low between blocks. For this reason, it is recommended to use the BLOCK_START output to indicate the start of an output block.

Ready For Data In (RFD_IN)

RFD_IN is an optional pin that can be used to inhibit the output side of the core while allowing current input operations to continue. RFD_IN is intended to be used as a means for a downstream system using the encoded data to indicate that it is not ready to handle any further data. If RFD_IN is low, all systematic and parity outputs, BLOCK_START, BLOCK_END, and RDY outputs are inhibited. The operation of RFD_IN is described in further detail later in this document.

Functional Description

FIFO Buffering

Figure 4 illustrates the process by which the core buffers incoming blocks in First-In-First-Out order. Input data bits are written in linear order into the Data Memory at contiguous addresses. Each time a complete input block has been written into the Data Memory, its block size is written to the Block Size FIFO.

When the core is ready to output a block, the block size is read from the Block Size FIFO. This determines the block size parameter for the interleaved and non-interleaved read address generators and is also utilized to calculate the base address for the next block.

As the Data Memory is dual-ported, the write and read operations are independent; this allows maximum throughput to be achieved for any arbitrary sequence of block sizes.

In addition, by using the optional ND and RFD_IN ports, the input and output processes can be independently flow-controlled while still maintaining maximum throughput.

If either the Data Memory or the Block Size FIFO becomes full, the core deasserts RFD to prevent any more data being written. If the Block Size FIFO becomes empty, the core deasserts RDY to indicate that there is no data to output. These processes are described in more detail later in this data sheet.

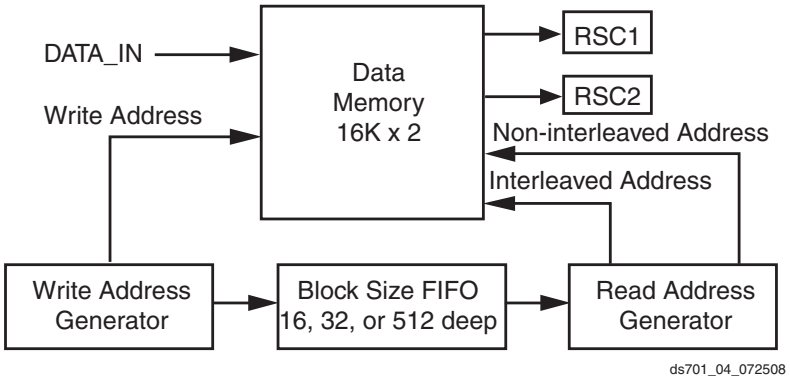

Figure 4: **FIFO Buffering**

Data Memory

The data memory consists of two 16384 x 1 dual-port memories. Data bits are written in linear order into both dual-port memories at the same modulo-16384 addresses. One of the memories is read in non-interleaved order, to provide data for RSC1, while the other is read in interleaved order to provide data for RSC2.

The data memory can accommodate two of the largest 3GPP LTE blocks, that is, 6144 bits. This ensures that maximum throughput can be obtained with a continuous stream of the largest block size.

If the data memory becomes full, the core deasserts the RFD output port to indicate that the core can accept no more data. More precisely, RFD is deasserted if the acceptance of the next input bit would cause the first bit of an unread data block in the data memory to be overwritten. RFD is asserted again when that data block has been completely output.

Block Size FIFO

The Block Size FIFO can be implemented in SRL32 or block RAM, as selected by the Block Capacity menu on the CORE Generator GUI.

Possible values for Block Capacity are 32 or Max.

- Block Capacity value Max selects a 512-deep block RAM FIFO, but reflects that the maximum possible number of 40-bit blocks that can be held in 16K of RAM is 409.
- When the Block Capacity value is 32, it is possible for Block Size FIFO to become full.

If the Block Size FIFO becomes full, the core drives RFD low to inhibit the input until space becomes available in the FIFO, due to the output side of the core fetching a block size from the FIFO.

If the Block Size FIFO becomes empty, then, when the core has output the last bit of the current output block, it deasserts RDY to indicate that there is no data to output.

Input Control Signals

Figure 5a shows the signals associated with the data input side of the core. If, on an active rising edge of CLK, FD_IN and ND (if selected) are both sampled high, this is known as a valid-FD, or valid First Data signal.

When a valid-FD is sampled, the RFFD signal is driven low to indicate that the core is no longer waiting for FD_IN. The block size, in this case *n*, of the current input block is sampled on the BLOCK_SIZE port, and the first data symbol, *d1*, is sampled on the DATA_IN port.

A high on the ND port indicates that the value on the DATA_IN port is new data. If ND is sampled low, then the DATA_IN port is not sampled, and the internal write address does not advance.

The core continues to input data, until *n* new data samples have been accepted, whereupon RFFD is normally driven high to indicate that the core is ready for a new input block.

After asserting RFFD, the core waits until the next valid-FD is sampled, whereupon a new write cycle is started, in this case with block size *N*.

The behavior of the core is not specified if, on a valid-FD, an invalid block size is sampled. If this condition occurs, the core should be reset by asserting SCLR.

Trellis Termination

As specified in the standard [Ref 1], the 12 trellis termination bits of RSC1 and RSC2 (see Figure 3) are re-multiplexed, over four clock cycles, onto the RSC1_SYSTEMATIC, RSC1_PARITY0, and RSC2_PARITY0 output ports. These bits are then transmitted in the following sequence:

$x_{n+1}, z_{n+1}, x_{n+2}, z_{n+2}, x_{n+3}, z_{n+3}, x'_{n+1}, z'_{n+1}, x'_{n+2}, z'_{n+2}, x'_{n+3}, z'_{n+3}$.

Output Control Signals

The RDY port is driven high to indicate that there is valid data on the systematic and parity ports. When the first valid data bit is on the output ports, the BLOCK_START output is driven high, and when the last tail bit is on the output ports, the BLOCK_END output is driven high. The output timing is shown in Figure 5b.

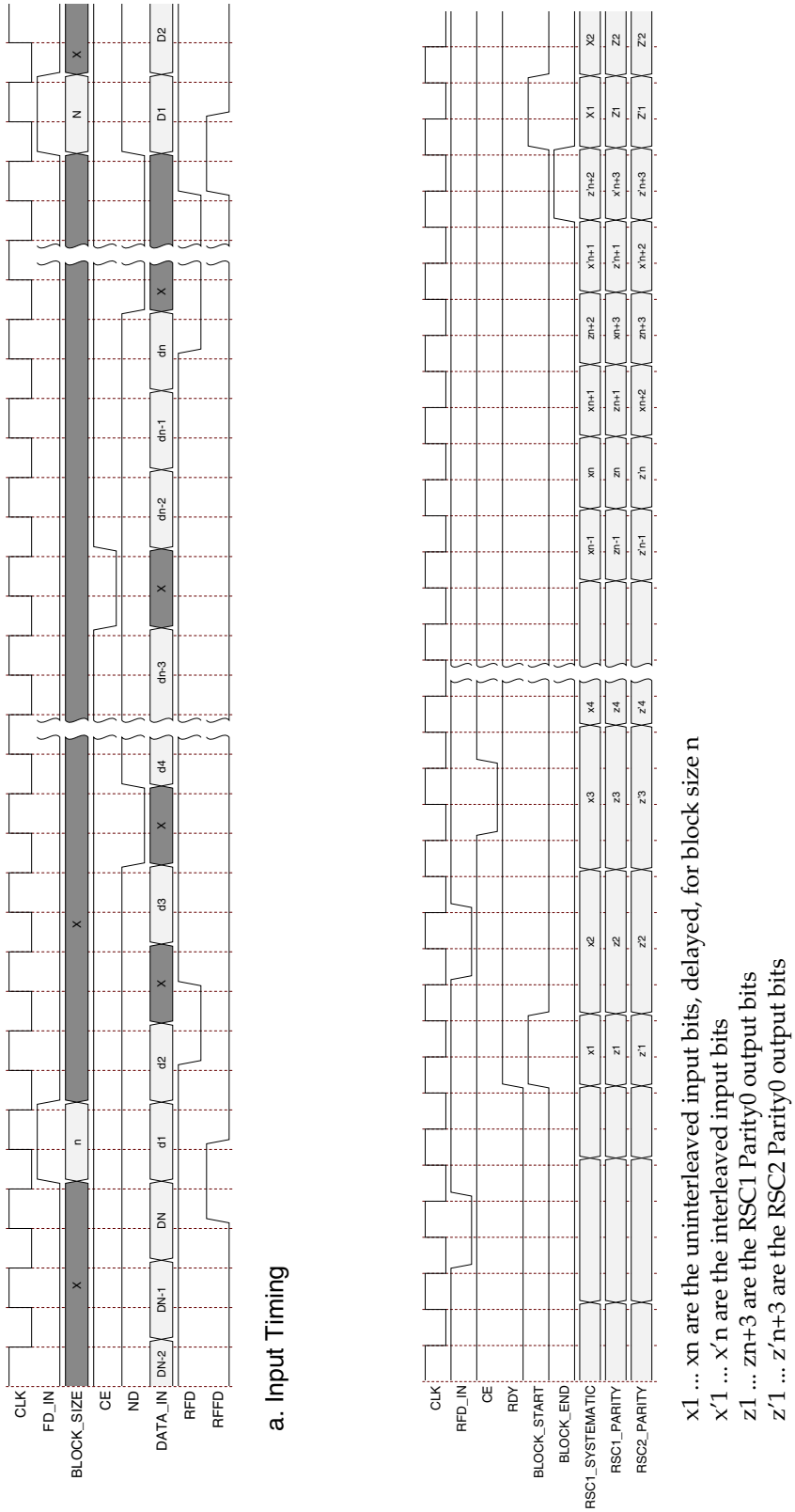

Figure 5: **Input and Output Timing**

Flow control on the output side can be implemented with the optional RFD_IN input port. If the RFD_IN port is sampled low on an active rising clock edge, the RSC output ports, RDY, and the internal circuitry associated with these outputs are frozen.

The latency delay is affected by the optional RFD_IN input. The RFD_IN input should be driven low only to inhibit the core output when RDY is high.

The input side of the core is not directly affected by the RFD_IN port. However, if RFD_IN is deasserted often enough, the time taken to output blocks can be extended, such that either the Data Memory or the Block Size FIFO can become full, whereupon the core deasserts RFD to indicate that it cannot accept any more input bits.

Latency

Latency is defined as the number of active clock cycles from the time the last bit of the first input block is accepted to the time at which RDY and BLOCK_START are asserted to indicate that the first block is ready to output. The latency value is defined only for the first block, because at other times the delay between a given block being input and that block being output depends on the number and size of blocks stored in the Data Memory.

Throughput

In any 3GPP LTE encoder, there must be a minimum of four idle clock cycles at the output due to the time required for the tail bits. In general, the maximum throughput for a given block size K is given by:

(K/(K+4)) x clock frequency (bits/s)

Example:

For a block size of 40 (worst case) and a clock rate of 250 MHz, the throughput is (40/(40+4))*250000000 or 227.3 Mb/s.

For a block size of 6144 (best case) and a clock rate of 250 MHz, the throughput is (6144/(6144+4))*250000000 or 249.8 Mb/s.

Core Resource Requirements and Performance

The resource requirements of the core and the achievable clock rates for the Virtex®-6, Virtex-5 and Spartan®-6 FPGAs are summarized in Table 3, Table 4, and Table 2 respectively.

Table 2: **Core Resource Requirements and Maximum Clock Frequency for Virtex-6 FPGAs**

Options	LUT/ FF Pair	LUTs, FFs	Total Block RAMs (36K or 2x18K)	Max. Clock Frequency (MHz)	
				-3	-1
blks=32	424	397/526	2	523	399
nd,rfd_in,sclr,blks=512	497	457/569	2	435	335

Table 3: **Core Resource Requirements and Maximum Clock Frequency for Virtex-5 FPGAs**

Options	LUT/ FF Pair	LUTs, FFs	Total Block RAMs (36K or 2x18K)	Max. Clock Frequency (MHz)	
				-3	-1
blks=32	569	320/529	2	504	393
nd,rfd_in,sclr,blks=512	635	389/568	2	419	327

Table 4: **Core Resource Requirements and Maximum Clock Frequency for Spartan-6 FPGAs**

Options	LUT/ FF Pair	LUTs, FFs	Total Block RAMs (18K or 2x9K)	Max. Clock Frequency (MHz)	
				-3	-2
blks=32	495	465/531	4	280	224
nd,rfd_in,sclr,blks=512	524	486/569	5	274	205

Notes:

1. Area and maximum clock frequencies are provided as a guide. They can 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.

C Model

A bit-accurate C model that allows system simulations is available for this core. To obtain the C model, contact your local Xilinx sales representative.

References

1. *3GPP TS 36.212 V9.0.0 (2009-12) 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Multiplexing and Channel Coding (Release 9).*
2. *Near Shannon Limit Error-correcting Coding and Decoding Turbo Codes*, C. Berrou, A. Glavieux, and P. Thitimajshima, IEEE Proc 1993, Int. Conf. Comm., pp1064-1070.
3. *3GPP LTE Turbo Encoder v3.1 Bit-Accurate C Model*, Xilinx UG506 (v2.0) April 19, 2010.

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*.

See the *IP Release Notes Guide* (XTP025) for further information on this core. There is a link to all the DSP IP and then to the relevant core.

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

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 software.

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 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.

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revision
09/19/08	1.0	Initial Xilinx release.
04/24/09	1.5	Updated core version to 3.0, and ISE tools to 11.1.
04/19/10	2.0	Updated core version to 3.1, and ISE tools to 12.1.
06/22/11	2.1	Updated for new family support and ISE tools to 13.2

Notice of Disclaimer