

SERDES 8b10b BFM User Guide

1.0 Overview.....	2
2.0 Functional Description.....	2
3.0 Top Level Ports.....	4
4.0 Port Timing Examples (Appendix A: Port Timing Diagrams).....	6

List of Tables

2.1 Source Files Descriptions	3
3.1 encoder interface to testbench + control	4
3.2 decoder interface to testbench + control	5
3.3 encoder interface to line	5
3.4 decoder interface to line	6

1.0 Overview

The SERDES 8b10b BFM is written in Verilog and is designed to be portable to any verification environment that is compatible with Verilog. The interfaces to the BFM are intended to also maximize portability. The testbench interface to the SERDES 8b10b BFM is a simple clocked data interface and configuration signals. The DUT interface is a data lane that is either serial, 10 bits wide (TBI) or twenty bits wide. This allows the BFM to be used in application including the SERDES model as well as those that do not.

Both the XAUI BFM and GIGE BFM use this same SERDES 8b10b layer to interface to the DUT SERDES. The SERDES 8b10b BFM is flexible enough to be used in any 8b10b application with or without other BFMs.

The SERDES 8b10b BFM consists of a transmit path (TX Path) that drives the line interface of the DUT and a receive path (RX Path) that monitors the line interface from the DUT. In cases where the DUT has more than one SERDES interface, the SERDES BFM may be instantiated multiple times.

The TX path receives 8 bit codes over a clocked data interface and drives these codes into the DUT over the line interface. The TX path supports error injection.

The RX path monitors the line interface from the DUT and passes 8 bit codes on to the testbench. Word alignment and disparity checking is performed.

Any errors detected by the RX and TX paths are reported through a single error reporting mechanism that can be customized to best suit the target verification environment.

Finally, a shallow loopback can be achieved by connecting the testbench output of the RX path to the testbench input of the TX path.

2.0 Functional Description

The SERDES 8b10b is partitioned into 3 main parts. Two modules interface to the testbench and DUT. *encoder* is the TX Path module and *decoder* is the RX Path module. The 8b10b code lookup tables and functions are separate from *encoder* and *decoder* and must be instantiated elsewhere in the testbench. This way, the overhead of reading in the memory file with the tables is shared in one place among all the *encoder* and *decoder* instances.

SERDES 8b10b File Name	Description
encoder.v	Module performing 8b10b encode
decoder.v	Module performing 8b10b decode
serdes_8b10b_fun.v	Encapsulates encode and decode tables and functions
serdes_8b10b.h	`defines for SERDES 8b10b BFM
8b10b.mem	Memory accessed by \$readmemb
tbase.h	`defines shared for all BFM modules

Table 2.1 File names and descriptions

At each positive clock edge, *encoder* samples the inputs from the testbench to determine the next 10 bit word to queue for transmission on the line. Nominally, it properly encodes the 8 bit code from the testbench updating the disparity.

However, there are 4 error injection modes available. The first two error injection modes are DC high and low. The BFM can queue 10'b11_1111_1111 or 10'b00_0000_0000 to indicate DC values on the lines. The third error injection mode is disparity error. The BFM can queue the proper encoding of the provided 8bit code, but encoded with the incorrect disparity. In the case where both disparity encodings of the 8bit code are the same, a different 8bit code is chosen to ensure the DUT detects a disparity error.

Finally, a code error can be injected. A bad 10bit code is randomly selected from a table of bad codes. Then, additional checking is done in the BFM. When the DUT is not in sync it will search for a K28.5 in the serial input stream. A sequence of code errors can actually contain K28.5 codes in an unintended alignment position. An additional check is performed on this code error word to ensure that it does not form a K28.5 with the prior 10 bit code.

Once queued for transmission, the 10 bit codes are shifted a programmable number of bits in time. This supports two needed features. First, the TBI and 20 bit line interfaces would otherwise always contain aligned words. By shifting the 10bit code words some number of bits in time, a single 10 bits on a TBI bus could contain some bits for the prior 10bit code as well as some bits of the current.

Second, protocols like XAUI include lane alignment features that require a constant board skew to be modeled. One instance of the SERDES 8b10b BFM could have a small delay while another could have a large delay. This will exercise the DUT code that removes the relative delay between the lanes.

encoder has one feature to assist in waveform debug. It decodes the control codes to assist in easily viewing the transmission of a particular code in a waveform viewer.

decoder performs word alignment to either disparity of K28.5. After bus sync is achieved, it passes along the aligned and decoded 8bit words. Any code or disparity error after bus alignment is achieved is reported as an error.

serdes_8b10b_fun is instantiated in the testbench for use by the *encoder* and *decoder* instances. The interface to *serdes_8b10b_fun* is a pair of function calls: *encode* and *decode* which perform encode and decode functions on one word of data.

3.0 Top Level Ports

The RX and TX paths are separate modules in the SERDES 8b10b BFM. Therefore, there are two top level modules that are instantiated, *encoder* and *decoder*.

<i>encoder</i> Interface to Testbench + Control		
Port Type	Port Name	Port Usage Description
input	clk	clock in 8 bit data from testbench
input [7:0]	txoctet	8 bit data from testbench
input	txkcode	indicates <i>txoctet</i> is a special code
input	error_disparity	indicates a disparity error is to be sent
input	error_random	indicates a code error is to be sent
input	error_one	indicates 10'b11_1111_1111 is to be sent
input	error_zero	indicates 10'b00_0000_0000 is to be sent
output	disparity	current running disparity after last encode
input [6:0]	txbit_delay	number of bits (1-129) to shift/delay this lane
input	tbimd	Must be one when TBI output required
input	tbimd20	Must be one when twenty bit output required

Table 3.1 *encoder* Interface to Testbench + Control

<i>decoder</i> Interface to Testbench + Control		
Port Type	Port Name	Port Usage Description
output	rxclk	clock out 8 bit data to testbench
output [7:0]	rxoctet	8 bit data to testbench
output	rxkcode	indicates <i>rxoctet</i> is a special code
output	rxdisparity	indicates a disparity after this decode
output	rxsync	indicates lane is in sync and output data is valid
input	rxresync	tie to zero. For future use.
input	warnonly	All errors reported instead as warnings
input	tbimd20	Must be one when twenty bit output required

Table 3.2 *decoder* Interface to Testbench + Control

<i>encoder</i> Line Interface		
Port Type	Port Name	Port Usage Description
output	txserial_d	serial data output transitions on posedge txserial_clk
input	txserial_clk	txserial_clk must have exactly 10 clock cycles for each cycle of clk. Also the posedges of clk must coincide with posedges of txserial_clk.
output [19:0]	tx_tbid20	little endian twenty bit interface. valid only with tbimd20 asserted
output	txtbic20	twenty bit interface clock with data transitions on negedge. Every negedge of tx_tbic coincides with an edge on txtbic20.
output [9:0]	tx_tbid	little endian TBI data. valid only with tbimd asserted
output	tx_tbic	TBI clock with data transitions on negedge. tx_tbic = !clk && (tbimd tbimd20);

Table 3.3 *encoder* line interface to DUT (serial, 10bit TBI, 20bit)

<i>decoder</i> Line Interface		
Port Type	Port Name	Port Usage Description
input	rxserial_d	serial data input must transition on posedge rxserial_clk
input	rxserial_clk	serial data clock
input [19:0]	rx_tbid20	little endian twenty bit interface. must transition on posedge rx_tbic20 and negedge rx_tbic.
input [9:0]	rx_tbid	little endian TBI data. must transition on negedge rx_tbic.
input	rx_tbic	clock for TBI (must be active only in both tbimd and tbimd20)
input	rx_tbic20	clock for twenty bit interface. (if tbimd20 active, rx_tbic20 must have all edges coincide with rx_tbic edges.)

Table 3.4 *decoder* line interface from DUT (serial, 10bit TBI, 20bit)

4.0 Port Timing Examples (Appendix A: Port Timing Diagrams)

The port timing is illustrated in timing diagrams captured from a waveform viewer in Appendix A. First, the encoder timing is shown in figures A1.0, A1.1, and A1.2 for each of the three interface widths: 1 bit, 10 bit (TBI), and 20 bit. Second, the decoder timing is shown in figures A1.3, A1.4, and A1.5.

4.1 Encoder timing

The 1 bit mode example waveform shows the *txserial_clk* active with a positive edge coincident with each positive edge of *clk*. The *txoctet*, *txkcode*, and *error_* inputs transition on the negedge of *clk*.

There are only 2 of the 6 outputs used when in 1 bit mode. The serial data line is *txserial_d* and it transitions near the posedge of *txserial_clk* and therefore should be sampled on the negedge of *txserial_clk*. The other output is the *disparity* indication which transitions shortly after the posedge of *clk*.

The internal signals shown reveal the internal operation of the module and are included here to provide some insight into that internal operation. To assist in debug visibility, each of the control codes is monitored. In the waveform one of these monitor signals is included: *tx_28_5*. The vector *code10b* is the encoded word presented on the *txoctet* and

txkcode inputs after any error insertion is performed by the *error_* inputs. Both the *tx_28_5* and *code10b* signals transition when the input is sampled just after the posedge of *clk*. Also on the posedge of *clk*, the 10 bit code in *code10b* is inserted into a queue for future transmission on the serial data output *txserial_d*. The delay through the queue in bit times is indicated by the static input *txbit_delay*. The *wr_ptr* and *rd_ptr* coordinate access to this queue.

The TBI mode example waveform shows the *txserial_clk* inactive and the *tbimd* mode input asserted to 1. The remaining inputs have the same timing and usage as the previously described 1 bit mode example.

Only 3 of the 6 outputs are used. *txtbid* and *txtbic* are the output ten bit interface with the data transitioning on the negedge of the clock so it can be safely sampled on the posedge. The *disparity* output is the same as described previously.

The internal signals are the same as described previously, except the *rd_ptr* advances in increments of 10 as each data transfer on the ten bit interface is 10 bits of data. It is worth noting that some values of the input *txbit_delay* will cause the TBI data output word to include a shifted sampling of the line bitstream and therefore contain the first portion of one 10 bit code and the end portion of the prior ten bit code. The example waveform, however, for ease of illustration does not cause this to occur. The BC on *txoctet* corresponds to 305 on *code10b* and 283 on *txtbid*.

All the inputs for twenty bit mode are the same as ten bit mode except *tbimd20* is asserted instead of *tbimd10*. The outputs are similar with *txtbic20* and *txtbid20* used in place of the TBI. The internal signals shown are identical to those in TBI mode.

4.2 Decoder Timing

Figure A1.3 shows the decoder timing in 1 bit interface mode. The input serial data transitions at the positive edge of the serial clock and it sampled internally on the negative edge. The output decoded *rxoctet*, *rxkcode*, *disparity*, and *rxsync* transition immediately prior to the positive edge of the output *rxclk*.

The internal signals are shown to provide some insight into the inner workings of the module and are included here for that purpose. At each negative edge of *rxserial_clk* the bit position of *code10b* indicated by *iter* is loaded with the input data *rxserial_d*.

Figure A1.4 shows an example with the ten bit interface active and also includes codes with disparity and code errors causing sync to be lost. The input *rxtbid* transitions on the negative edge of the input *rxtbic*. The outputs have the same timing as shown in the prior example of the 1 bit interface.

In the presence of errors, the decoder will immediately loose sync and begin looking for the K28.5. The generated output *rxclk* is not active when the lane is out of sync.

Finally, figure A1.5 shows the twenty bit interface active. Both the *rxbic20* and *rxbic* must be active in twenty bit mode. All timing is the same as previously shown.

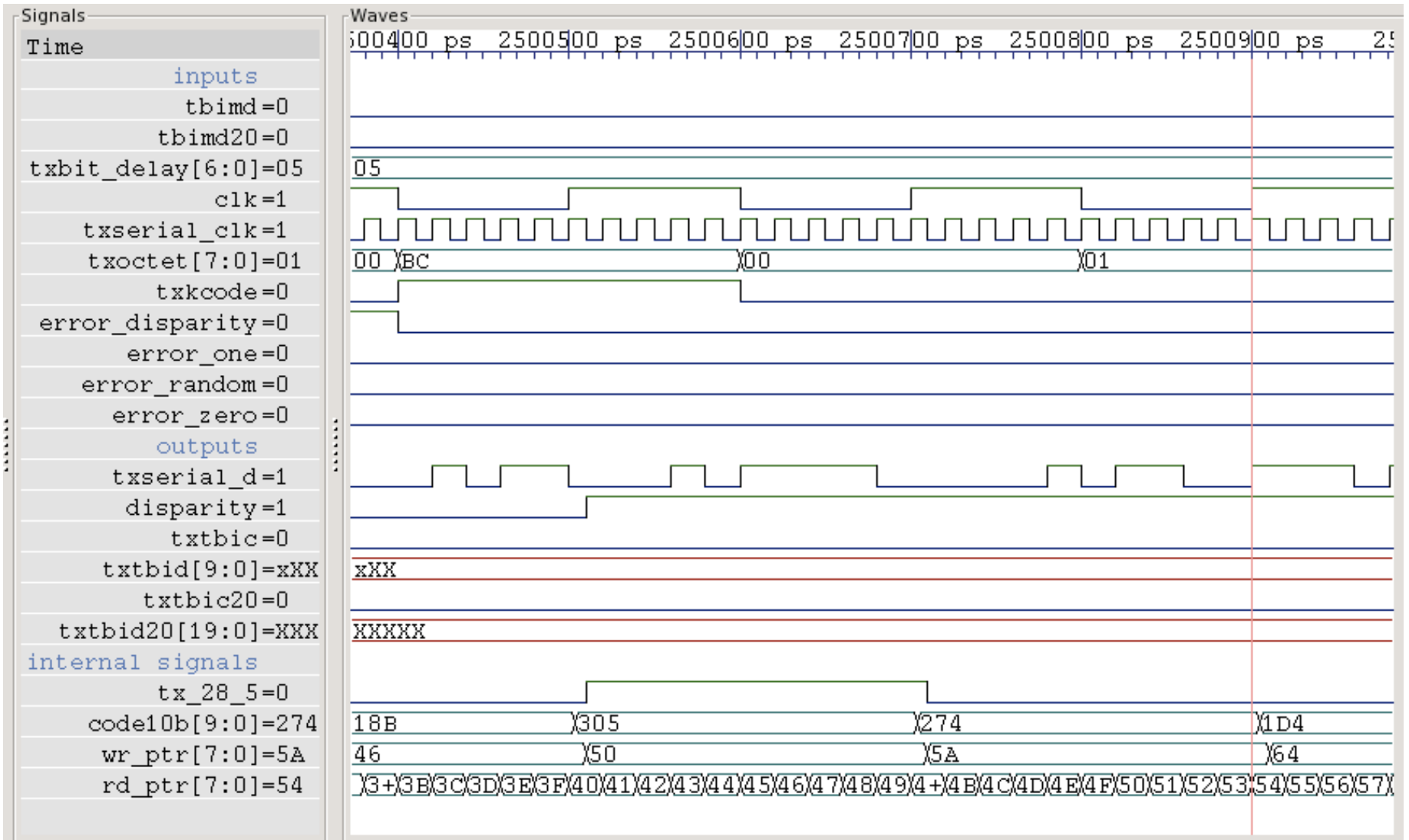


Figure A1.0: Timing Diagram of 8b10b encoder with 1 bit output enabled.

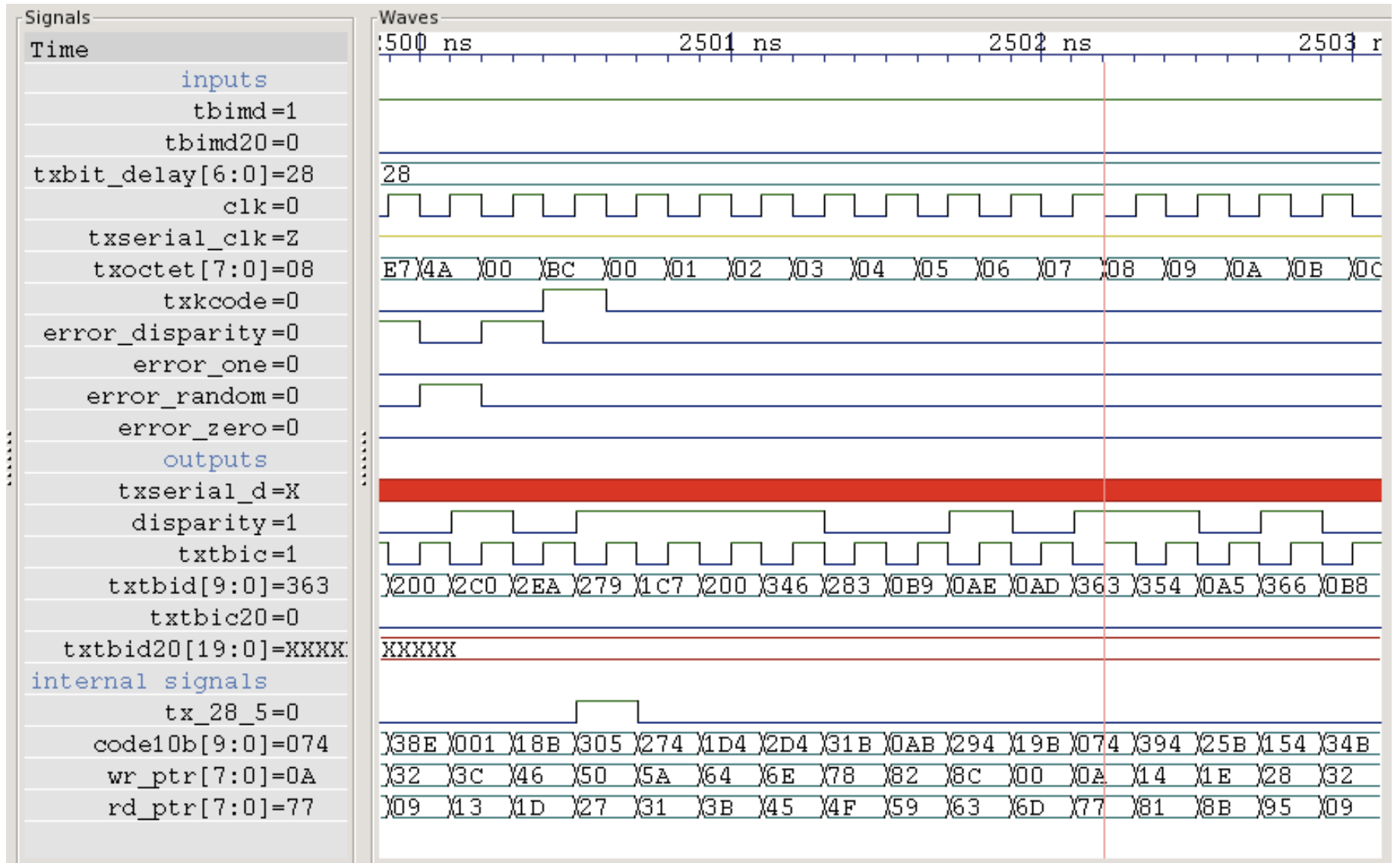


Figure A1.1 Timing Diagram of 8b10b encoder with 10 bit output enabled.

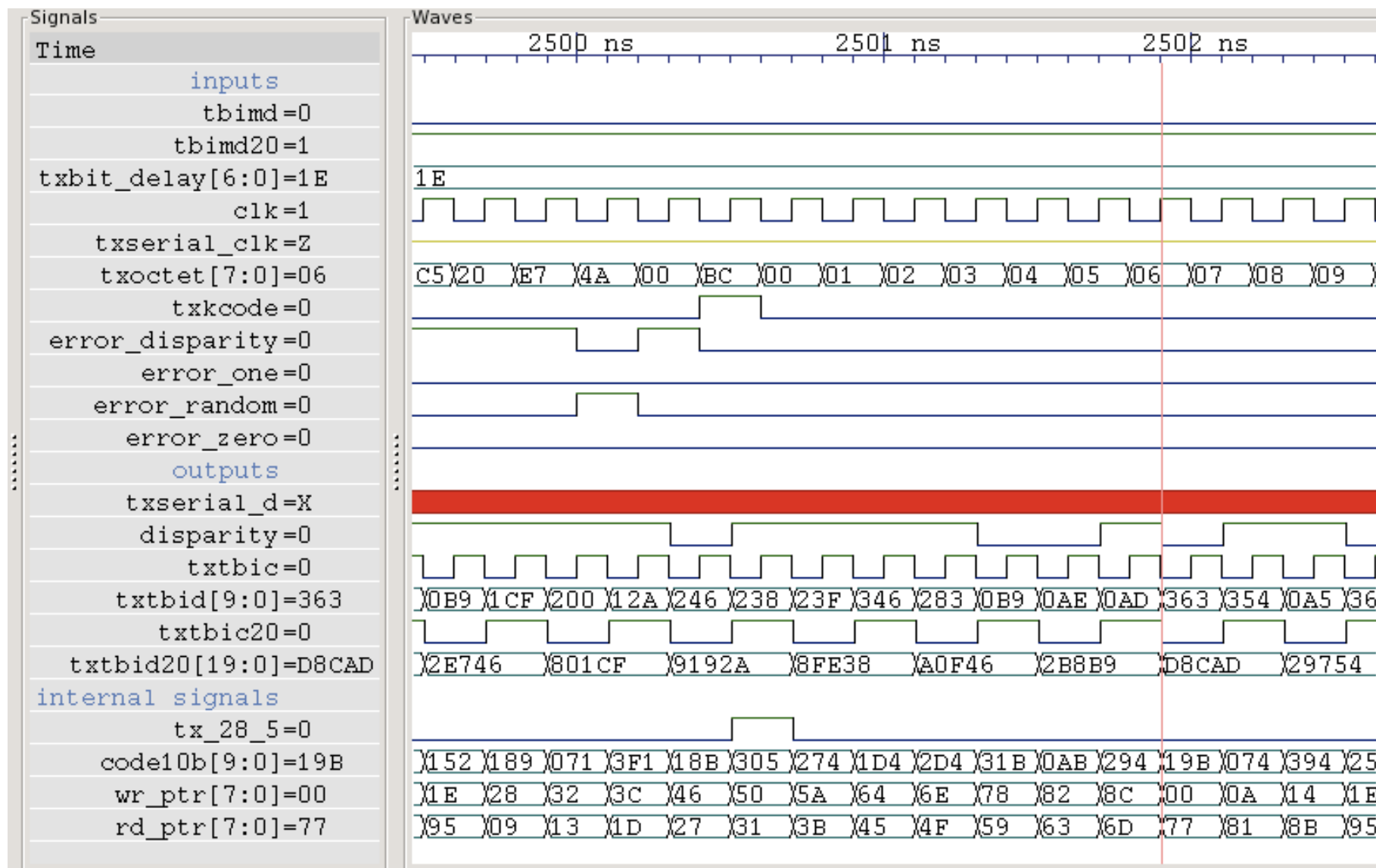


Figure A1.2 Timing Diagram of 8b10b encoder with 20 bit output enabled.

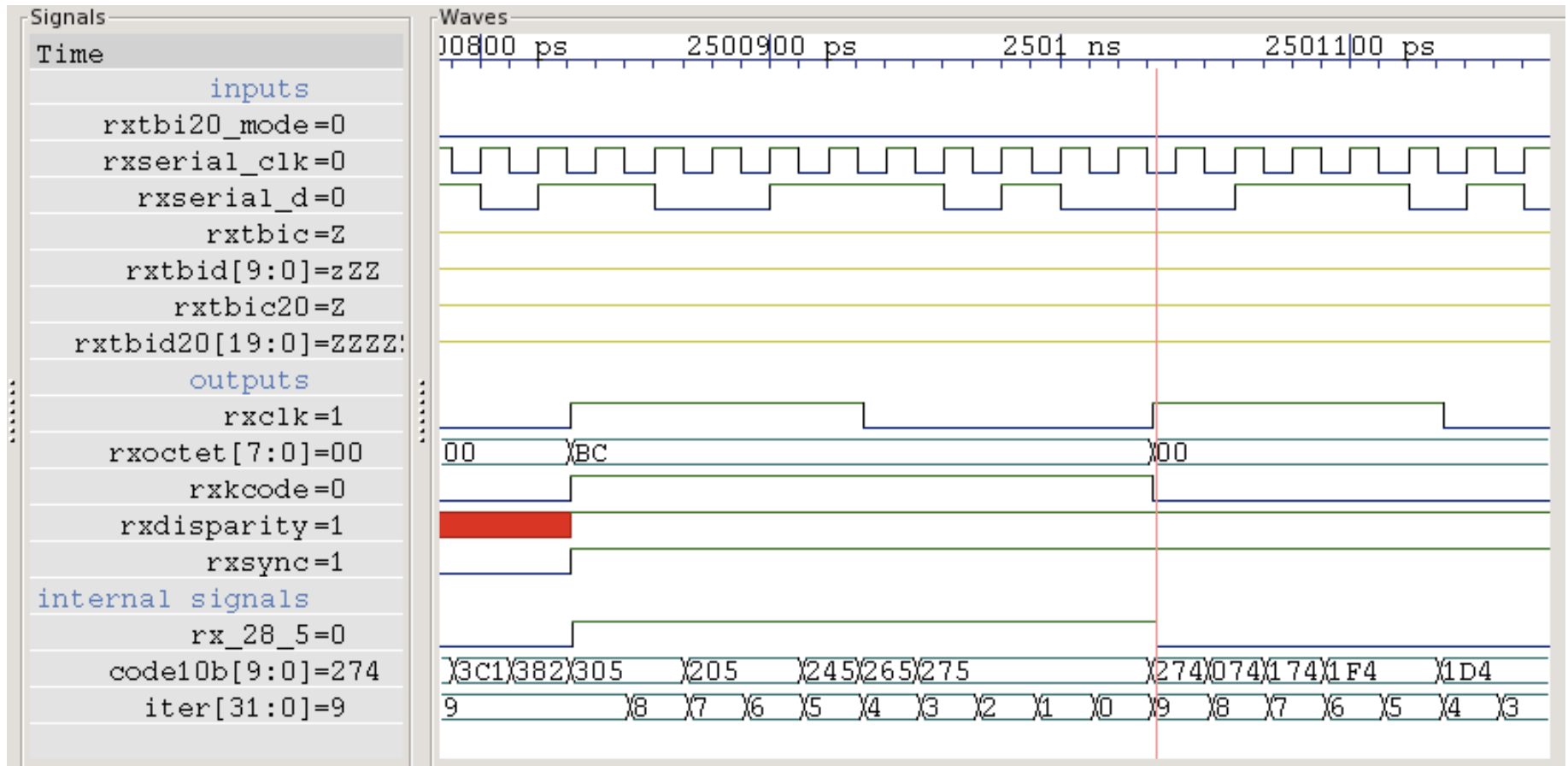


Figure A1.3 Timing Diagram of 8b10b decoder with 1 bit input active

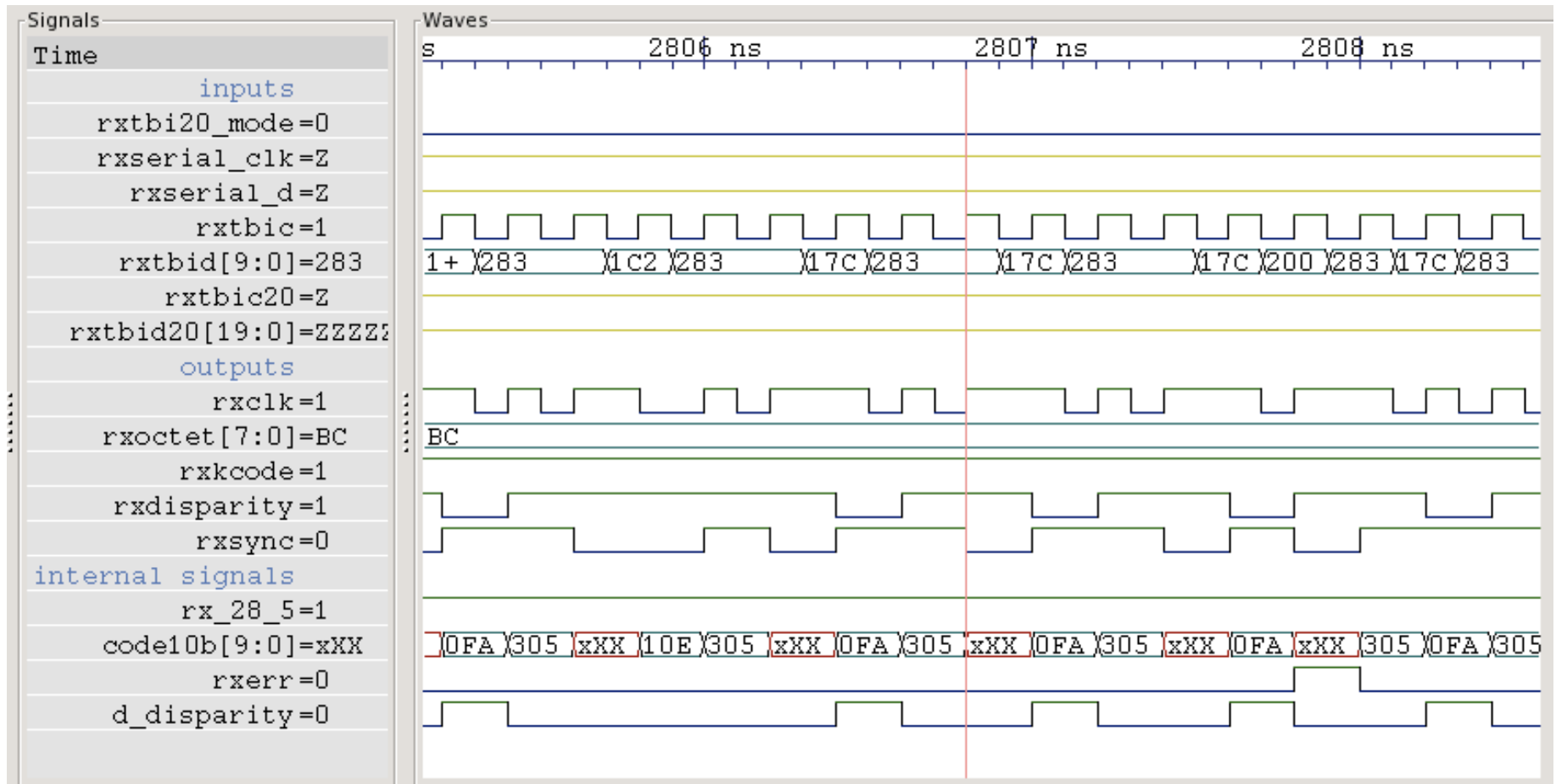


Figure A1.4 Timing Diagram of 8b10b decoder with 10 bit input active

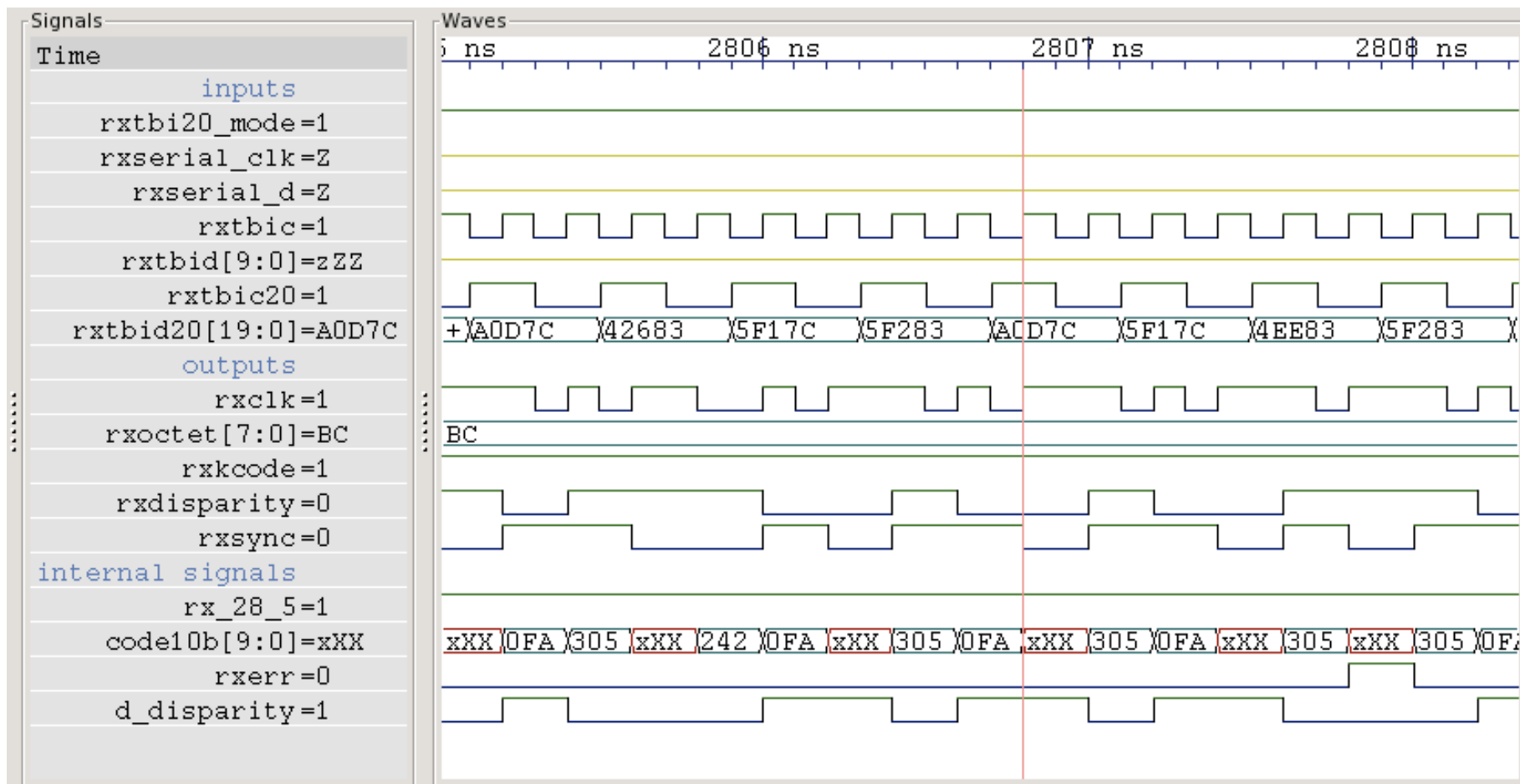


Figure A1.5 Timing Diagram of 8b10b decoder with 20 bit input active